Brandenburg
University of Technology
Cottbus

Faculty of Mathematics,
Natural Sciences and
Computer Science

Institute of Computer Science

COMPUTER SCIENCE REPORTS

Report 03/13

September 2013

12. GI / ITG FACHGESPRÄCH SENSORNETZE

Jörg Nolte

Computer Science Reports
Brandenburg University of Technology Cottbus
ISSN: 1437-7969

Send requests to: BTU Cottbus
Institut für Informatik
Postfach 10 13 44
D-03013 Cottbus

Jörg Nolte
jon@informatik.tu-cottbus.de
http://www.tu-cottbus.de/fakultaet1/de/betriebssysteme/

12. GI / ITG
Fachgespräch Sensornetze

Computer Science Reports
03/13
September 2013

Brandenburg University of Technology Cottbus

Faculty of Mathematics, Natural Sciences and Computer Science

Institute of Computer Science

Computer Science Reports
Brandenburg University of Technology Cottbus
Institute of Computer Science

Head of Institute:
Prof. Dr. Ingo Schmitt schmitt@tu-cottbus.de
BTU Cottbus
Institut für Informatik
Postfach 10 13 44
D-03013 Cottbus

Research Groups:	Headed by:
Computer Engineering	Prof. Dr. H. Th. Vierhaus
Computer Network and Communication Systems	Prof. Dr. H. König
Data Structures and Software Dependability	Prof. Dr. M. Heiner
Database and Information Systems	Prof. Dr. I. Schmitt
Programming Languages and Compiler Construction	Prof. Dr. P. Hofstedt
Software and Systems Engineering	Prof. Dr. C. Lewerentz
Theoretical Computer Science	Prof. Dr. K. Meer
Graphics Systems	Prof. Dr. D. Cunningham
Systems	Prof. Dr. R. Kraemer
Distributed Systems and Operating Systems	Prof. Dr. J. Nolte
Internet-Technology	Prof. Dr. G. Wagner

CR Subject Classification (1998): C.3, J.7

Printing and Binding: BTU Cottbus

ISSN: 1437-7969

INHALTSVERZEICHNIS

SIMULATION, REALE MESSUNGEN UND ERFAHRUNGSBERICHTE

PROTOKOLL- UND ANWENDUNGSENTWICKLUNG

PLATTFORMENTWICKLUNG UND TESTBEDS

Real-world Bluetooth Master-Slave Bridge Deployment

Nicole Todtenberg, Paweł Kornecki
IHP
Im Technologiepark 25
15236 Frankfurt (Oder)
Germany
Email: todtenberg@ihp-microelectronics.com

Matthias Mahlig
lesswire AG
Rudower Chaussee 30
12489 Berlin
Germany
Email: mahlig@lesswire.com

***Abstract*—For Wireless Sensor Networks (WSN) the limited number of active Bluetooth slaves in a piconet is a constraint. Scatternet topologies are described in the Bluetooth specification for use cases requiring many devices. However, scatternet related aspects are not addressed in detail in the Bluetooth specification and research work was primarily evaluated by simulation or analytical analysis. In contrast, we implemented a real-world scatternet topology providing the services HSP and A2DP. Our experiments show that real-world scatternet deployment of time-critical data is not straightforward.**

***Index Terms*—Bluetooth, Scatternet, HSP, A2DP**

I. Introduction

Bluetooth wireless communication standard is very popular because of its claim of simplicity and compactness, its interference resilience and power efficiency. For wireless sensor network (WSN) applications the limited number of seven active slaves in a piconet is a handicap. The Bluetooth specification provides the scatternet mode for such use cases: Several piconets can be connected to form a network. One or more devices need to act as participants in multiple piconets (PMP) [1] that need to apply time division multiplex [2]. Although the scatternet mode is described throughout the specification possible challenges and scatternet related aspects are not covered in detail [3].

A considerable amount of research effort was directed to Bluetooth scatternets. Three challenges were addressed primarily: Formation algorithms, traffic scheduling algorithms and routing protocols. Reviews of scatternet formation algorithms are given in [4], [5] and [6]. Different scheduling strategies are categorized and summarized in [7], [8], [9], [10], [5] and [11]. A review of scatternet routing protocols is given in [12], [13] and [14]. One common aspect to all these works is: The evaluation of the proposed concept is either performed by simulation or analytical analysis. To gather real-world experiences we implemented a scatternet topology in order to investigate the performance of a PMP device.

Our scatternet conists of a headset, a mobile phone and a proprietary PMP device. Headset and mobile are common consumer class devices whose firmware was not changed during our investigation. A development board (BlueSy, lesswire AG) acts as PMP device that forwards the payload from the mobile to the headset and vice versa. The PMP device connects two piconets: One piconet is built by the mobile and the PMP device, the other by the headset and the PMP device (see Fig. 1). Accordingly, the PMP device links two net segments together and therefore is named *BT Bridge* hereinafter.

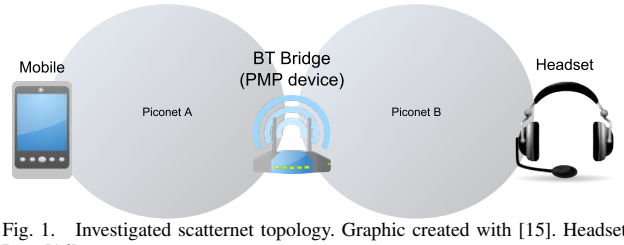


Fig. 1. Investigated scatternet topology. Graphic created with [15]. Headset Icon [16].

The Bluetooth Headset Profile (HSP) and the Advanced Audio Distribution Profile (A2DP) are used as applications. HSP provides telephony whereas A2DP offers high quality audio distribution service. This selection enables the investigation of both Bluetooth connection types Asynchronous Connection-oriented (ACL) and Synchronous Connection-oriented (SCO) on the basis of time critical payload data. Realization of these services is a challenge concerning throughput and delay of BT Bridge. Thus, these kinds of services are predestined to evaluate the performance of the scatternet.

II. Bluetooth Characteristics

A Bluetooth piconet is built by one device operating as *Master* and at least one device acting as *Slave* [17]. The communication topology of a piconet is a star - consequently slaves are not able to communicate directly [17], [18]. The Bluetooth master applies Time Division Multiplex (TDM) in order to coordinate data transfers from and to its slaves [17]. Typically, one time slot has a duration of 625μs (1600 slots per second) [19]. The master of a piconet polls its slaves [19], [20] by sending a packet in a time slot t. Following slot (t+1) is reserved for a response from that slave [18], [20]. Furthermore, Bluetooth uses Frequency Division Multiple Access (FDMA) in terms of Frequency Hopping Spread Spectrum (FHSS) [17].

One carrier frequency of a set of 79 frequencies is chosen per packet [19] which results in maximum 1600 frequency shifts per second [19]. In each piconet a unique frequency hopping sequence is used [18]. Thus, devices that participate in several piconets need to switch frequency hopping sequence during operation. It is not possible to switch frequency hopping sequence between two consecutive Bluetooth time slots [20].

A. ACL versus SCO

HSP is based on SCO connections for which fixed time slots are allocated [18]. Therefore, SCO connections can be regarded as circuit switching [18]. In contrast, A2DP uses ACL connections that realize the concept of packet switching [21]. Furthermore, HSP and A2DP differ in means of data flow direction: HSP payload data is sent from mobile to headset and vice versa, whereas A2DP data is transferred from mobile to headset only. An ACL connection is established automatically if two Bluetooth devices get connected. This default ACL connection is used for exchange of payload data as well as signalling information [18]. SCO connections are not reliable in contrast to ACL ones. Conceptually, SCO connections are predestined for transport of time-dependent data whereas ACL connections provide a reliable link that guarantees data integrity [18], [21].

B. HCI

A Bluetooth product consists of a host and at least one controller [17]. The physical layer, the link controller, the baseband resource manager, the link and the device manager are implemented in the controller [22]. In our case the controller is a separate chip (BlueCore 4) of the BlueSy development board. All layers below non-core profiles and above the layers of the controller - namely Logical Link Control and Adaptation Protocol (L2CAP), Security Manager Protocol (SMP), Attribute Protocol (ATT)/Generic Attribute Profile (GATT) and Service Discovery Protocol (SDP) - are located in the host [17], [22]. In our use case the host is built by the microcontroller of the BlueSy development board and the Bluetooth functionality is provided by a software Bluetooth stack. Host and controller implement an intermediate layer in order to communicate with each other. This layer is the Host Controller Interface (HCI). Considering Bluetooth HCI specification all control messages sent to Bluetooth controller are called *commands* whereas all received control messages from Bluetooth controller are called *events*.

III. Concept

BT Bridge acts as intermediate station between exactly two user devices (see Fig. 1). Once the user initiates a call or starts to play music all data of the mobile is transferred to BT Bridge which forwards the information to the headset and vice versa for HSP. All data BT Bridge receives is forwarded at the level of HCI layer (see Fig. 2). Accordingly, BT Bridge is fully transparent for user devices. Furthermore, complexity of application logic of BT Bridge is kept to a minimum and the device is not restricted to certain profiles. Due to the fact

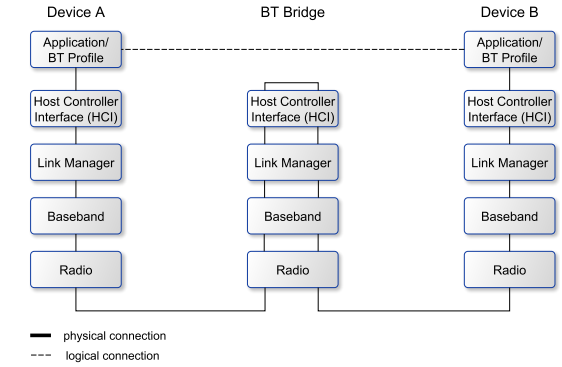


Fig. 2. Bluetooth protocol layers of BT Bridge and its connected devices.

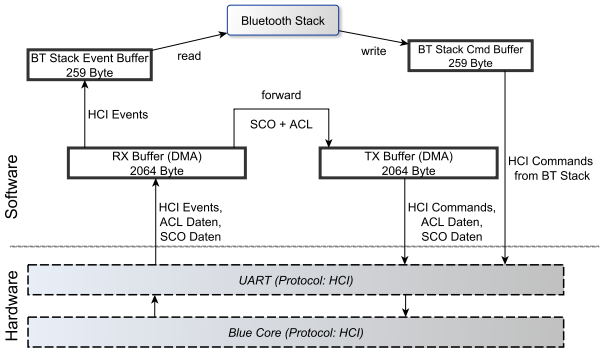


Fig. 3. Functional structure of BT Bridge.

that BT Bridge forwards data packets at HCI layer it has to differentiate received data. All Bluetooth HCI events have to be sent to Bluetooth stack in order to maintain Bluetooth state machine. In contrast, all data packets (ACL and SCO) have to be handled by BT Bridge application itself (see Fig. 3). This behaviour distinguishes BT Bridge implementation from ordinary Bluetooth applications. Common Bluetooth software design maintains well defined protocol stack layers so that user data is sent and received solely through the Bluetooth stack. Separate buffers are needed to ensure correct behaviour of BT Bridge. This way data scrambling of both entities communicating with the controller is prevented. Fig. 3 illustrates the main data buffers and the data flow of BT Bridge software.

The operating mode of BT Bridge is freely configurable: It is either able to operate as master for both user devices (piconet) or to connect two individual piconets of device A and B (scatternet). Current implementation supports master/slave bridge. Theoretically, there could be a slave/slave bridge, too. But this configuration would require a role switch during connection establishment, and acceptance of role switch requests is not mandatory [23]. Therefore, every other operating mode than master/slave bridge could lead to a loss of interoperability.

Configuration of device pairs is static: There is a table of configured device pairs in the permanent memory of BT Bridge which can be modified through specific configuration commands.

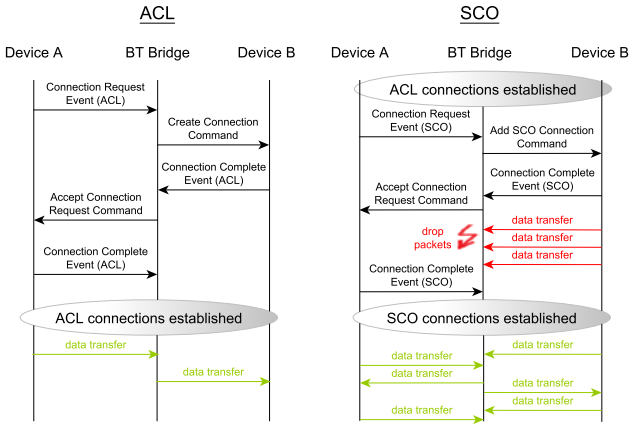


Fig. 4. Connection establishment procedures for ACL and SCO from BT Bridge's HCI perspective. Concept of both link types are different: Data transfer for ACL is unidirectional whereas it is bidirectional in case of SCO.

TABLE I
SCO HIGH QUALITY VOICE (HV) PACKET FORMATS [24].

Packet Type	Size [bit]	Payload [bit]	FEC [bit]
HV1	240	80	160
HV2	240	160	80
HV3	240	240	0

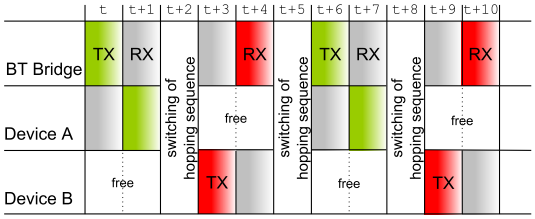


Fig. 5. BT Bridge in scatternet mode as master/slave bridge. In piconet composed by BT Bridge and Device A BT Bridge is master whereas it operates as slave in the piconet that consits of BT Bridge and Device B.

A. Connection Establishment

BT Bridge establishes ACL and SCO connections to its clients. When the application starts it initializes all hardware and software components and goes to an idle state where it waits for incoming connections.

1) Asynchronous Connection-Oriented (ACL): When an incoming connection request from device A is received, the internal device table is checked for existing entries. If A is not present in database or it does not have a corresponding device B the request is rejected. Otherwise a link to B is created and the request from A is accepted afterwards. A responses with a confirmation and devices are connected. The described procedure is illustrated in left part of Fig. 4. Further actions depend on user's choice. For example, when the user starts the music player the mobile can establish a L2CAP connection to provide A2DP. In this case, since all L2CAP commands and events are sent over an existing ACL link BT Bridge application only forwards every packet to the second device and no additional application logic is necessary. If the mobile receives a call request it could initiate SCO connections follow the steps below (III-A2).

2) Synchronous Connection-Oriented (SCO): The algorithm for setting synchronous connections is similar to ACL links, but data transfer is bidirectional. Thus, it is not possible to ensure both connections are established before one device begins to transfer data. From a device's point of view its connection is established first, data transfer could start immediately. This device is not aware that it is connected to an intermediate device. To deal with this situation BT Bridge application drops the first SCO packets as long as the second SCO connection setup is not finished (right part of Fig. 4). It is assumed that no relevant information is lost and the user does not perceive a loss in quality. Reason behind this presumption is that there is no conversation without both links set up.

B. Packet Type for SCO in Scatternet

For ACL links master arbitrarily chooses non-reserved time slots for packet transmissions [21] whereas slot reservation is static for SCO links [18]. For SCO connections a datarate of 64 kbit/s is mandatory. SCO packet types and their characteristics are listed in Tab. I. With respect to SCO datarate one of two slots (64 kbit/s : 80 bit = 800 slots/s) has to be reserved for SCO when using HV1 packets. Therefore all slots are occupied due to the slave responding in the remaining slots. If all slots of a device are occupied by SCO and that device needs to transfer control information, SCO information is discarded [21], causing a deliberate loss in quality. BT Bridge needs to maintain two SCO connections. Accordingly, it is not possible to use HV1 packets in the given scenario. For HV2 packets every fourth slot and for HV3 packets every sixth slot is needed for a transfer from the master to a slave. Traffic of different piconets that cover the same area is separated by FHSS with unique frequency hopping sequences per piconet. Because it is not possible to switch the frequency hopping sequence between two consecutive slots it is only possible to use HV3 packets for SCO in scatternet mode. Two HV3 SCO connections in scatternet mode lead to an occupation of all available slots at BT Bridge (see Fig. 5).

C. Processing of received HCI packets

ACL and SCO packets are forwarded by BT Bridge at HCI level. In order to forward HCI data packets to a remote device it is necessary to change HCI connection handles. This is done by swapping the connection handle in every received packet with the connection handle of the corresponding device (the device specified in the pair).

IV. RESULTS

We evaluated the feasibility of our concept with five different mobile phones ranging from low-budget to high-end devices and two different headsets (see Tab. II). A2DP worked with all mobile-headset combinations except for Nokia C1

TABLE II
RESULTS

Phone	Headset	A2DP	HSP
iPhone 4	Nokia BH-503	yes	no
	Jabra BT530	yes	yes
Samsung Nexus S	Nokia BH-503	yes	no
	Jabra BT530	yes	yes
Sony Ericsson p990i	Nokia BH-503	yes	no
	Jabra BT530	yes	yes
Sony Ericsson w580i	Nokia BH-503	yes	no
	Jabra BT530	yes	yes
Nokia C1	Nokia BH-503	no A2DP	no
	Jabra BT530	no A2DP	yes

which does not support this profile. Nevertheless, the quality of the transmitted information was moderate due to regular short interruptions. In contrast, the sound quality of A2DP was fine with all devices in piconet mode.

The performance of the telephony service via HSP was very poor: It was not possible to establish the SCO connections as described above with the Nokia Headset although the profile should be supported. Furthermore, it was not feasible to maintain SCO connections over an extended period with the remaining combinations. After a few seconds the Bluetooth links broke. We also analyzed HSP in piconet configuration. The speech quality was even impaired in this case compared to the direct communication of mobile and headset. This observation led to the assumption that the integration of BT Bridge in data flow infects the speech quality. We assume that this degradation of performance is associated with integration of additional delays. SCO slot allocation is fixed. That means that the controller of BT Bridge needs to send or receive data in every SCO slot. Degredation at BT Bridge in piconet can only occur if data from host is not available in time. Real-world SCO deployments are challenging even in piconet mode but in scatternet operation we could not even manage to maintain the SCO connections.

V. CONCLUSIONS

Our experiments show that Bluetooth scatternet operation is still challenging. Even for a small network of simple topology the real-world deployment of time-critical data is not straightforward. The performance of different mobile phones differed substantially for the analyzed services HSP and A2DP. HSP and A2DP forced BT Bridge to operate near the upper limit of performance and revealed the difficulties of a real-world scatternet deployment. In our view it is not reasonable to continue the work in this configuration. We observed different phenomena whose causes cannot be determined due to the end user devices that appeared as black boxes. Accordingly, it would be necessary to replace consumer class end devices by full accessible development boards in order to investigate the reasons of degradation in detail.

REFERENCES

[1] *Bluetooth Specification Version 4.0. Volume 2. Part B. 1 General Description*, Bluetooth SIG Std., 2010.

[2] *Bluetooth Specification Version 4.0. Volume 1. Part A. 4 Communication Topology and operation*, Bluetooth SIG Std., 2010.

[3] V. B. Mišić, J. Mišić, and K. L. Chan, "Performance of adaptive bridge scheduling in a scatternet with a slave-slave bridge: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 4, no. 1, pp. 85–98, Feb. 2004.

[4] S. Basagni, R. Bruno, G. Mambrini, and C. Petrioli, "Comparative performance evaluation of scatternet formation protocols for networks of bluetooth devices," *Wirel. Netw.*, vol. 10, no. 2, pp. 197–213, Mar. 2004.

[5] R. Roy, M. Kumar, N. Sharma, and S. Sural, "Bottom-up construction of bluetooth topology under a traffic-aware scheduling scheme," *Mobile Computing, IEEE Transactions on*, vol. 6, no. 1, pp. 72–86, 2007.

[6] Z. Wang, R. J. Thomas, and Z. J. Haas, "Performance comparison of bluetooth scatternet formation protocols for multi-hop networks," *Wirel. Netw.*, vol. 15, no. 2, pp. 209–226, Feb. 2009.

[7] W. Priess, J. Rezende, and L. Pirmez, "Enhancing scatternets performance via scheduling algorithm parametrization," in *Personal Wireless Communications*, ser. Lecture Notes in Computer Science, M. Conti, S. Giordano, E. Gregori, and S. Olariu, Eds. Springer Berlin Heidelberg, 2003, vol. 2775, pp. 741–755.

[8] V. B. Mišić and J. Mišić, "Polling and bridge scheduling algorithms in bluetooth," Department of Computer Science, University of Manitoba, Tech. Rep., 2003.

[9] V. Mišić, J. Mišić, and K. Chan, "Walk-in bridge scheduling in bluetooth scatternets," *Cluster Computing*, vol. 8, no. 2-3, pp. 197–210, 2005.

[10] J.-P. Sheu, K.-P. Shih, S.-C. Tu, and C.-H. Cheng, "A traffic-aware scheduling for bluetooth scatternets," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 7, pp. 872–883, 2006.

[11] G. Ramana Reddy, S. Bhatnagar, V. Rakesh, and V. P. Chaturvedi, "An efficient algorithm for scheduling in bluetooth piconets and scatternets," *Wirel. Netw.*, vol. 16, no. 7, pp. 1799–1816, Oct. 2010.

[12] C.-Y. Chang, P. Sahoo, and S.-C. Lee, "A location-aware routing protocol for the bluetooth scatternet," *Wireless Personal Communications*, vol. 40, no. 1, pp. 117–135, 2007.

[13] X. Li, C. Men, M. Li, and L. Sun, "Hierarchical routing for large scale bluetooth network," in *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. International Conference on*, vol. 2, 2009, pp. 415–418.

[14] K. Persson and D. Manivannan, "Hybrid bluetooth scatternet routing," in *Ubiquitous Intelligence and Computing*, ser. Lecture Notes in Computer Science, D. Zhang, M. Portmann, A.-H. Tan, and J. Indulska, Eds. Springer Berlin Heidelberg, 2009, vol. 5585, pp. 163–177.

[15] yed graph editor. [Online]. Available: http://www.yworks.com/de/products_yed_about.html

[16] License: GNU Lesser General Public License. [Online]. Available: http://www.iconarchive.com/show/oxygen-icons-by-oxygen-icons.org/Devices-audio-headset-icon.html

[17] *Bluetooth Specification Version 4.0. Volume 1. Part A. 1 General Description*, Bluetooth SIG Std., 2010.

[18] *Bluetooth Specification Version 4.0. Volume 1. Part A. 3.3 Physical Channels*, Bluetooth SIG Std., 2010.

[19] *Bluetooth Specification Version 4.0. Volume 2. Part B. 2 Physical Channels*, Bluetooth SIG Std., 2010.

[20] *Bluetooth Specification Version 4.0. Volume 2. Part B. 8.6 Active Mode*, Bluetooth SIG Std., 2010.

[21] *Bluetooth Specification Version 4.0. Volume 2. Part B. 4 Logical Transports*, Bluetooth SIG Std., 2010.

[22] *Bluetooth Specification Version 4.0. Volume 1. Part A. 2 Core System Architecture*, Bluetooth SIG Std., 2010.

[23] *Bluetooth Specification Version 4.0. Volume 2. Part C. 4.4 Role Switch*, Bluetooth SIG Std., 2010.

[24] *Bluetooth Specification Version 4.0. Volume 2. Part B. 6.5.2 SCO Packets*, Bluetooth SIG Std., 2010.

Wireless in The Woods: Experimental Evaluation of IEEE 802.11a/b/g in Forested Environments

Margit Mutschlechner, Patrick Baldemaier, Philipp Handle, and Falko Dressler
Computer and Communication Systems, Institute of Computer Science, University of Innsbruck, Austria
{mutschlechner,dressler}@ccs-labs.org
{patrick.baldemaier,philipp.handle}@student.uibk.ac.at

***Abstract*—We study the feasibility of using IEEE 802.11a/b/g in forested environments. Our particular interest is to identify potential wireless communication technologies for spanning a ground network in the woods to study the foraging and hunting behavior of bats in the wild. We are working on ultra-low power communication devices to monitor contact times and to localize bats in their natural habitat. For collecting and aggregating the received information, a stationary ground network is planned but little is known about the signal attenuation due to shadowing and fading and the resulting packet error rates in such environments. Thus, we experimentally studied selected wireless LAN technologies in an extensive set of measurements. We report our findings that also help selecting protocols and configurations in other sensor networking applications.**

I. Introduction

The use of Wireless Sensor Networks (WSNs) has become more ubiquitous in the last years [1]. Under the umbrella of Cyber Physical Systems (CPSs), many research activities are going on investigating topology management, wireless access, routing and data aggregation, security, and many others [2]. Still, many of the early findings in this domain have rarely been applied in practical applications. In the scope of the BATS research group, we explore the feasibility of using sensor networking technology for tracking bats in their natural habitats.[1] State of the art technology for such observations is still radio telemetry [3]. This, however, is extremely labor expensive and allows to track single individuals with limited localization accuracy. In order to study the animals' behavior, contacts and precise tracking would be beneficial.

To this end, we plan a stationary sensor network deployed in a forest environment that collects and aggregates such contact information. In a first step, we explore the feasibility of different wireless communication technologies for this ground network. Given that distributed real-time localization algorithms require a certain (depending on the algorithm even very high) data rate between the nodes, wireless LAN according to IEEE 802.11a/b/g might be a good candidate.

We realized that signal attenuation due to shadowing and fading constitutes a substantial problem in forest environments. To the best of our knowledge, there is no study comparing the performance of IEEE 802.11a, IEEE 802.11b, and IEEE 802.11g in such environments. Thus, we experimentally studied the applicability of the mentioned technologies in an extensive set of measurements. In the following, we report our findings that also help selecting protocols and configurations in other sensor networking applications.

[1]Dynamic Adaptable Applications for Bats Tracking by Embedded Communicating Systems, http://www.for-bats.de/

II. Related Work

In the literature, the performances of the IEEE 802.11a/b/g standards has been discussed in depth for line-of-sight scenarios as well as indoor and urban environments [4]–[6]. Yet, knowledge about the behavior in the countryside and wilderness is not fully understood, especially for forested surroundings.

Research on the impact of vegetation on wireless communication has been focused mainly on general radio wave propagation and attenuation models, which have been either developed from empirical data [7]–[11] or have been derived analytically [12].

Some studies have been focusing on the communication through the canopy where one station is situated at a prominent height over the coverage area [7], [13]. However, placing the communicating nodes in or above the canopy would make deployment and maintenance of our sensor network more difficult, and, as bats are flying blow the canopy to hunt for prey on the ground, this would influence communication negatively. Hence, the decision was taken to place the nodes near the ground.

Most models depend on the type of trees and on the used frequency, even though only few studies have been conducted in the ISM band. Therefore, most models are not applicable to the IEEE 802.11a/b/g standards as they do not cover the used frequencies of 2.4 GHz and 5.8 GHz, respectively [10], [11], [14]–[16]. A model, that also covers the ISM band, is proposed in [17]. The model is valid for frequencies of 1 GHz to 60 GHz and is very accurate as it combines edge diffraction, ground reflection and the signal going through vegetation. As it takes into consideration the occurring tree species, tree height, tree spacing and leaf dimension it can be applied only to a well-structured and precisely describable environment, which is not given in a naturally grown forest. Therefore, the model is not applicable if the performance in more general scenarios is of interest. The performance of the 2.4 GHz band in comparison to the 5.8 GHz band is evaluated in [8], but only in two forest types, an oak tree forest and an eucalyptus woodland.

The impact of pine trees on the communication using IEEE 802.15.4 has been studied in [18]. In a similar way, the throughput and received signal strength of IEEE 802.11b/g have been evaluated in a wooded area in [9]. However, to the best of the authors' knowledge, there is no study comparing

(a) Light forest

(b) Dense forest

(c) Light forest with thick undergrowth

Figure 1: Pictures of the three environments

the performance of IEEE 802.11a, IEEE 802.11b, and IEEE 802.11g in various different forest environments.

III. Measurement Environment

Three different forested environments were chosen for the measurement campaign: a light forest (cf. Figure 1a), a dense forest (cf. Figure 1b), and a light forest with thick undergrowth of about 2 meters height (cf. Figure 1c). As a reference, free space measurements were performed on a grassland.

The light forest and the undergrowth environments consist of conifers with an average height of 20 meters, a diameter of up to 50 centimeters and no branches for the lower four meters. In the dense forest, the large conifers were accompanied by smaller trees. The light forest was free of ground vegetation, opposite to the other forests.

The most appropriate environment in the scope of the BATS research group is the light forest, as the lack of ground vegetation is preferred by the bats according to [19]. The other forests were included into the measurement campaign to complete the picture.

In the light forest environment three different scenarios were taken into account: line of sight, a few trees between the two stations, and as many trees as possible between the two stations. In the undergrowth and dense forest environments neither line of sight nor a few trees between the stations was achievable.

Overall, we took measurements in six different scenarios, in the remaining part referred to as free space, line of sight, few trees, many trees, dense forest, and light forest with undergrowth.

IV. Measurement Setup

The measurements were performed using two measurement stations, both serving as transmitter and receiver (in the following called Station 1 and Station 2, respectively). We used laptop computers (Ubuntu Linux 12.04) connected via USB to a WLAN stick (Airlive X.USB dual band IEEE 802.11a/b/g/n USB WLAN stick with Atheros chipset,[2]) and an omnidirectional antenna (VERT2450 dual band 2.4-2.5 GHz and 4.9-5.9 GHz with 3 dBi gain[3]) attached to the top of a pole (cf. Figure 2) to reduce ground level influences.

[2]Airlive X.USB, http://www.airlive.com/product/X.USB

[3]Ettus Research VERT2450 Antenna, https://www.ettus.com/product/details/VERT2450

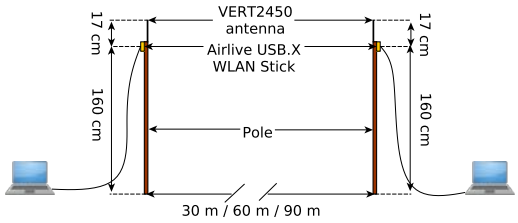
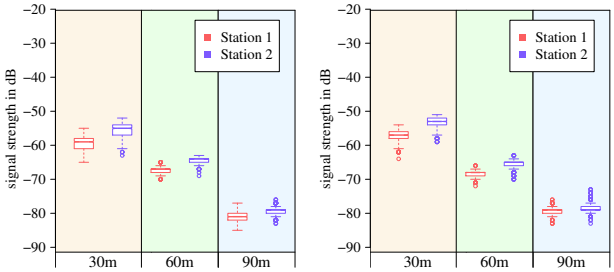


Figure 2: Measurement setup. We used two poles to mount the antennas at a height of 1.6 m in a distance of 30 m, 60 m, and 90 m



(a) Bit rate of 1 Mbit s^{-1}

(b) Bit rate of 11 Mbit s^{-1}

Figure 3: Results for IEEE 802.11b and the dense forest scenario

One measurement station was placed on a fixed position, whereas the other was moved according to the distance to measure (30 m, 60 m, and 90 m). We used two different modulations for each technology, one for the slowest supported bit rate and one for a higher bit rate. The same frequency was used for both modulations. Table I summarizes all the used configurations.

Therefore, in total six measurements per distance and scenario were performed. For each measurement run, both stations alternated sending 900 packets of 256 B containing a sequence number with a transmission power of 20 dBm. The receiving station logged the received signal strength and the sequence number.

V. Evaluation of the Results

In the following, we discuss the measurement results with the aim to identify best suited technologies for the different scenarios.

A. Impact of Distance

The first and obvious observation is the fact, that with increased distance the received signal strength decreased and, in some cases, not all packets were received. This effect can be seen in Figure 3, which (as an example) shows the measurement results for IEEE 802.11b in the dense forest. The difference between the received signal strength for both stations is due to the fact, that the forest is not really homogeneous.

Up to 60 m, we did not experience any packet loss (data not shown), but at a distance of 90 m, when sending with a bit rate

Protocol	IEEE 802.11a		IEEE 802.11b		IEEE 802.11g	
Channel	Channel 44		Channel 1		Channel 1	
Frequency	5.220 GHz		2.412 GHz		2.412 GHz	
Modulation	BPSK-½	QAM16-½	DBPSK	CCK	BPSK-½	QAM16-½
Data rate	6 Mbit s^{-1}	24 Mbit s^{-1}	1 Mbit s^{-1}	11 Mbit s^{-1}	6 Mbit s^{-1}	24 Mbit s^{-1}

Table I: Configurations of IEEE 802.11a/b/g used in the measurements

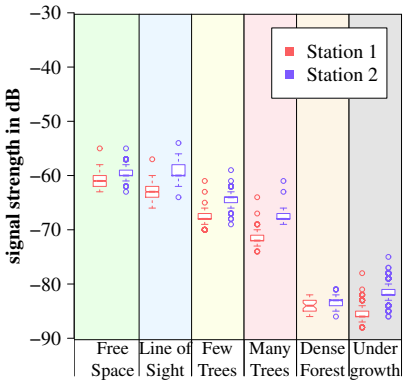
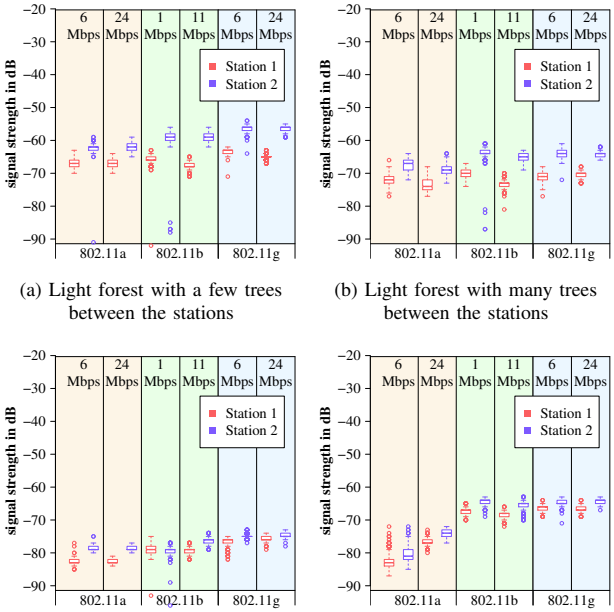


Figure 4: Results for IEEE 802.11a using a bit rate of 6 Mbit s^{-1} and a distance of 90 m



(a) Light forest with a few trees between the stations

(b) Light forest with many trees between the stations

(c) Light forest with undergrowth

(d) Dense forest

Figure 5: Comparison of the protocol standards in various environments at a distance of 60 m

of 11 Mbit s^{-1}, the receiver is not able to decode 25 % to 55 % of the packets. When the bit rate is throttled down to 1 Mbit s^{-1}, the transmission is very stable and the loss rate is close to 0 %. A similar behavior can be observed for IEEE 802.11a and IEEE 802.11g as well as for the other environments (data not shown). The packet loss ratio for the largest measured distance of 90 m increased to almost 100 % when the forest becomes thicker and more impenetrably.

B. *Influence of the Scenario*

When comparing all six scenarios, we see a steady decrease in the received signal strength as the environment becomes thicker and more impenetrably. This trend is shown in Figure 4 for IEEE 802.11a, a bit rate of 6 Mbit s^{-1}, and a distance of 90 m.

As we are interested in very sparsely crowded forests, we are primarily interested in the differences compared to the free space (grass land) measurements. We see that the received signal strength slightly differs (this is for the two leftmost scenarios in our figure). Although in both scenarios the measurement stations had a direct line of sight between each other, the received signal strength clearly decreases in the presence of some trees. An explanation for this behavior could be multipath effects, which are even more significant in the light forest scenario.

The same trend of a decreased received signal strength as the environment becomes thicker can be observed also for IEEE 802.11b and IEEE 802.11g as well as for other distances (data not shown).

C. *Comparison of the Communication Standards*

In order to help taking decisions which wireless LAN standard to choose in which scenario, we finally compare the results according to the different technologies and configurations. Figure 5 gives an overview of the most expressive results regarding the comparison of the standards. Here, we fixed the communication range to 60 m. As can be seen, none of the three standards is definitively outstanding.

In all four graphs a slightly better performance of IEEE 802.11b and IEEE 802.11g can be observed. This is most probably due to the fact that shadowing and fading have a stronger impact with increasing frequency. In the light forest with a few trees between the stations, many trees between the stations, and a thick undergrowth (cf. Figure 5a, Figure 5b, and Figure 5c, respectively), the difference between the two protocol standards sending in the 2.4 GHz band and IEEE 802.11a is negligible. As can be seen, the trend increases as the forest becomes thicker and more impervious. In comparison with the results from the dense forest, the received signal strength of IEEE 802.11a is reduced by 10 dBm to 20 dBm (cf. Figure 5d).

A rather unexpected behavior can be observed when taking into consideration the percentage of received packets. As with increased distance the packet loss becomes a more and more considerable issue, we focus on the largest measured distance of 90 m, shown in Figure 6. For the sparser scenarios, we experienced almost no packet loss (at least 90 % reception rate),

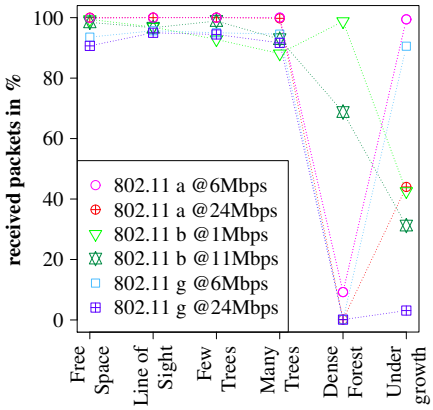


Figure 6: Percentage of received packets at a distance of 90 m

independent of the protocol standard and bit rate. However, this is not the case for the two scenarios dense forest and undergrowth. Comparing the percentage of received packets for IEEE 802.11a with a bit rate of 6 Mbit s^{-1} with the received signal strength shown in Figure 4, we see that the two measures do not coincide. Although the received signal strength of the two scenarios is in the same range, we experience a huge drop in the number of received packets in the dense forest scenario. A similar behavior can be observed for the other protocol standards and bit rates (data not shown).

VI. Conclusion and Further Work

In order to evaluate the performance of the different IEEE 802.11 protocol variants in forested environments, we performed measurement campaigns in three different forests as well as in a grassland scenario. The evaluation of the packet loss rate and the signal strength shows that the performance is influenced by distance as well as the density of the forest. In particular, the environment has an even bigger impact. Slightly moving a node (e.g., one meter to the side) can influence the performance even more than changing the distance between the stations.

It turned out that there is no clear winner when looking at IEEE 802.11a/b/g. Depending on the scenario, the performance of the protocol variants changes substantially. The results clearly show that the upper bound for the distance between the two communicating stations in a forested environment is about 90 m, or, depending on the scenario and standard, even below. In the scope of our BATS project, this finding makes it necessary to deploy a rather dense ground network.

We can also conclude that further investigations have to be performed to get deeper insights about the influence of different kinds of forests on IEEE 802.11a/b/g. This could lead, for example, to some best practices definition for optimally placing nodes in the woods.

VII. Acknowledgments

This work was supported by the German Research Foundation (DFG) under grants no. FOR 1508 (subproject TP4).

References

[1] F. Wang and J. Liu, "Networked Wireless Sensor Data Collection: Issues, Challenges, and Approaches," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 4, pp. 673–687, November 2011.

[2] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, "From wireless sensor networks towards cyber physical systems," *Elsevier Pervasive and Mobile Computing*, vol. 7, no. 4, pp. 397–413, August 2011.

[3] C. Drescher, "Radiotracking of Myotis myotis (Chiroptera, Vespertilionidae) in South Tyrol and implications for its conservation," *MAMMALIA*, vol. 68, no. 4, pp. 387–395, 2004.

[4] D. Cheung and C. Prettie, "A Path Loss Comparison Between the 5 GHz UNII Band (802.11a) and the 2.4 GHz ISM Band (802.11b)," Intel Labs, Technical Report, January 2002.

[5] A. Doufexi, S. Armour, B.-S. Lee, A. R. Nix, and D. R. Bull, "An Evaluation of the Performance of IEEE 802.11a and 802.11g Wireless Local Area," in *IEEE International Conference on Communications (ICC 2003)*. Anchorage, AK: IEEE, May 2003, pp. 1196–1200.

[6] D. B. Faria, "Modeling Signal Attenuation in IEEE 802.11 Wireless LANs," Computer Science Department, Stanford University, Technical Report TR-KP06-0118, July 2005.

[7] ITU-R, "Attenuation in vegetation," Recommendation P.833-7, February 2012.

[8] I. Cuiñas, J. A. Gay-Fernández, A. V. Alejos, and M. G. Sánchez, "A comparison of radioelectric propagation in mature forests at wireless network frequency bands," in *4th European Conference on Antennas and Propagation (EuCAP 2010)*. Bacelona, Spain: EurAAP, April 2010, pp. 1–5.

[9] K.-C. Wang, G. Venkatesh, S. Pradhananga, S. Lokala, S. Carter, J. Isenhower, and J. Vaughn, "Building Wireless Mesh Networks in Forests: Antenna Direction, Transmit Power, and Vegetation Effects on Network Performance," in *3rd ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH 2008)*. San Francisco, CA: ACM, September 2008, pp. 97–98.

[10] J. A. R. Azevedo and F. E. S. Santos, "An Empirical Propagation Model for Forest Environments at Tree Trunk Level," *IEEE Transactions on Antennas and Propagation*, vol. 59, no. 6, pp. 2357–2367, June 2011.

[11] S. Phaiboon and S. Somkuarnpanit, "Mobile Path Loss Characteristics for Low Base Station Antenna Height in Different Forest Densities," in *1st IEEE International Symposium on Wireless Pervasive Computing (ISWPC 2006)*. Phuket, Thailand: IEEE, January 2006.

[12] L.-W. Li, T.-S. Yeo, P.-S. Kooi, M.-S. Leong, and J.-H. Koh, "Analysis of Electromagnetic Wave Propagation in Forest Environment Along Multiple Paths," *Progress In Electromagnetics Research*, vol. 23, pp. 137–164, 1999.

[13] J. Richter, R. F. S. Caldeirinha, M. O. Al-Nuaimi, A. Seville, N. C. Rogers, and N. Savage, "A Generic Narrowband Model for Radiowave Propagation through Vegetation," in *61st IEEE Vehicular Technology Conference (VTC2005-Spring)*. Stockholm, Sweden: IEEE, May 2005, pp. 39–43.

[14] D. Dence and T. Tamir, "Radio Loss of Lateral Waves in Forest Environments," *Radio Science*, vol. 4, no. 4, pp. 307–318, April 1969.

[15] T. Tamir, "Radio Wave Propagation Along Mixed Paths in Forest Environments," *IEEE Transactions on Antennas and Propagation*, vol. 25, no. 4, pp. 471–477, July 1977.

[16] N. Savage, D. Ndzi, A. Seville, E. Vilar, and J. Austin, "Radio wave propagation through vegetation: Factors influencing signal attenuation," *Radio Science*, vol. 38, no. 5, October 2003.

[17] N. C. Rogers, A. Seville, J. Richter, D. Ndzi, N. Savage, R. F. S. Caldeirinha, A. K. Shukla, M. O. Al-Nuaimi, K. Craig, E. Vilar, and J. Austin, "A Generic Model of 1-60 GHz Radio Propagation through Vegetation - Final Report," Radiocommunications Agency, Project Report QINETIQ/KI/COM/CR020196/1.0, May 2002.

[18] J. A. Gay-Fernández, M. G. Sánchez, I. Cuiñas, A. V. Alejos, J. G. Sánchez, and J. L. Miranda-Sierra, "Propagation analysis and deployment of a wireless sensor network in a forest," *Progress In Electromagnetics Research*, vol. 106, pp. 121–145, 2010.

[19] B.-U. Rudolph, A. Liegl, and O. Von Helversen, "Habitat Selection and Activity Patterns in the Greater Mouse-Eared Bat Myotis myotis," *Acta Chiropterologica*, vol. 11, no. 2, pp. 351–361, 2009.

Selbstorganisierende drahtlose Vernetzung in Photovoltaik-Kraftwerken

Stefan Lange
IHP
Im Technologiepark 25
D-15236 Frankfurt(Oder)
E-Mail: lange@ihp-microelectronics.com

***Abstract*—Drahtlose Sensornetze (Wireless Sensor Networks/WSN) sind Gegenstand breit angelegter Forschung. Zahlreiche theoretische und praktische Arbeiten im Bereich des Netzaufbaus, des Routing und der Weiterleitung von Daten sowie der Datenverarbeitung in WSN wurden veröffentlicht. Berichte über den konkreten Einsatz dieser Forschungsergebnisse z.B. in Industrieanlagen sind jedoch selten. In dieser Arbeit soll diese Lücke für die Forschungsergebnisse des Projekts SolarFlex geschlossen werden. Es wird über den Einsatz eines WSN in einem Photovoltaik-Kraftwerk berichtet. Der Text gibt einen Überblick, wie beginnend mit Forschung über die Produktentwicklung, Feldtests bis hin zur ersten Installation eine im industriellen Umfeld einsatzbare WSN-Lösung entsteht.**

Keywords—Deployment; Real World Scenario; Photovoltaic

I. EINLEITUNG

Die Überwachung großer Gebiete ist das Hauptanwendungsgebiet drahtloser Sensornetze (Wireless Sensor Networks/WSN). Die einzelnen Knoten erfassen vor Ort Messdaten über ihre Sensoren und versenden diese über das Netzwerk an eine Datensenke. Die Knoten im Netz haben dabei folgende Aufgaben:

- Die von den Sensoren empfangenen Daten codieren und an die Datensenke weiterleiten. Ist die Datensenke nicht direkt zu erreichen, müssen die Daten über Zwischenstationen in einem Multi-Hop-Verfahren an die Senke versendet werden.
- Datenpakete von anderen Sensorknoten entgegennehmen und in Richtung Datensenke weiterleiten.

Photovoltaik-Kraftwerke bieten ein passendes Einsatzszenario für WSN. Die Kraftwerke bestehen aus einer großen Anzahl von über große Flächen verteilt aufgebauten Photovoltaik-Modulen, die aus der Energie des Sonnenlichts eine Gleichspannung erzeugen. Diese Gleichspannung wird von Wechselrichtern in Wechselspannung umgewandelt, die schließlich am Netzeinspeisepunkt dem Verbundnetz übergeben wird. Die Wechselrichter erfassen dabei die unter anderem für die Abrechnung benötigten Ertragsdaten. Diese Ertragsdaten müssen an zentraler Stelle erfasst und gespeichert werden. Die Wechselrichter sind dazu über ein Ethernet-basiertes Netzwerk mit einem Gateway mit dem Internet verbunden. Über das Internet fragt der erfassende Rechner die Daten der Wechselrichter einzeln ab. Üblicherweise verwaltet dieser Rechner mehrere Anlagen. [1] Die Datenübertragung auf der Anlage zwischen den einzelnen Wechselrichtern und dem Gateway zum Internet erfolgt dabei drahtgebunden. Im ZIM-Projekt SolarFlex wurde diese Datenübertragung unter Einsatz eines selbstorganisierenden Bluetooth[2]-Netzwerks drahtlos realisiert.

Der nachfolgende Text ist wie folgt gegliedert. Im Anschluss an die Einleitung wird zuerst die im Projekt verwendete Sensorknotenplattform in Bezug auf Hardware und Software beschrieben. Darauffolgend werden die zu beachtenden gesetzlichen und privatrechtlichen Regularien erläutert. Im anschließenden Abschnitt werden die einzelnen Entwicklungsschritte von der Algorithmen-Entwicklung in der Simulationsumgebung bis hin zum Aufbau der ersten Pilotinstallation erläutert. Ein Fazit und ein Überblick über weitere realisierte Sensornetze schließen die Arbeit ab.

II. SENSORPLATTFORM

Die Sensorknoten wurden von unserem Projektpartner – der Fa. lesswire AG – entwickelt. Da die Knoten durch die Photovoltaik-Anlage mit Energie versorgt werden, konnte leistungsfähigere Hardware eingesetzt und auf energiesparende Maßnahmen verzichtet werden.

A. Hardware

Der Sensorknoten verfügt über einen ARM9-basierten Mikrokontroller der Fa. Atmel. Der Mikrokontroller stellt eine Ethernet-Schnittstelle, die der Sensorknoten als Datenschnittstelle verwendet, bereit. Als Bluetooth-Model kommt das BlueBear-HCI[3] zum Einsatz. Das Modul kombiniert den Bluetooth-Chip BlueCore04[4] von CSR mit einem hochempfindlichen Eingangsverstärker, so dass die Eingangsempfindlichkeit von -94 dBm liegt. Bei einer Ausgangsleistung von 20 dBm liegt die Reichweite bei über 500 m.

B. Software

Als Betriebssystem wurde Embedded-Linux gewählt. Embedded-Linux beinhaltet den vorzertifizierten Bluetooth-Protokoll-Stack BlueZ[5], der die benötigte Treiber- und Protokollunterstützung bereitstellt.

C. Zulassung und Zertifizierung

Im akademischen Umfeld spielen Zulassungsfragen im Allgemeinen keine Rolle, da die Sensorknoten innerhalb der Einrichtung verbleiben und nicht „in Verkehr gebracht“ werden. Sollen die Sensorknoten jedoch verkauft und in Wohn- und/oder Industrieanlagen eingesetzt werden, sind gesetzliche Vorschriften zu beachten. Das Gesetz über Funkanlagen und Telekommunikationsendeinrichtungen (FTEG), das die EU-Telekommunikations-Richtlinie 1999/5/EG umsetzt, regelt die Zulassung von Funkanlagen in Deutschland. Für jedes Produkt muss dessen Inverkehrbringer eine Konformitätserklärung zur Erlangung des CE-Zeichens erstellen und unterschreiben. Mit diesem Dokument bestätigt er, dass das Produkt den geforderten Normen entspricht. Auf ein Bluetooth-basiertes Gerät sind folgende Normen anzuwenden:

- EN 60950-1 für die elektrische Sicherheit,
- EN 300 328 für die elektromagnetische Verträglichkeit,
- EN 301 489-17 für die Einhaltung der Anforderungen an Funkanlagen, die das Frequenzband 2,4 GHz verwenden.

Verwendet ein Produkt Bluetooth und soll es als Bluetooth-fähig gekennzeichnet werden, ist eine Listung des Geräts bei der Bluetooth-SIG(Special Interests Group) notwendig. Dazu muss die Interoperabilität des Geräts mit anderen Bluetooth-Geräten nachgewiesen werden. Der Nachweis erfolgt für jedes vom Produkt unterstütztes Protokoll des Bluetooth-Protokoll-Stack unter Anwendung standardisierter Testpläne. Die Bluetooth-SIG stellt ihren Mitgliedern eine Software zur Verfügung, die aus Menge der durch das zu zertifizierende Produkt unterstützten Protokolle einen Testplan erstellt. Zur Vereinfachung des Zertifizierungsprozesses kann auf die Testergebnisse von bereits zertifizierten Komponenten zurückgegriffen werden. In diesem Projekt z.B. waren der Bluetooth-Chip BlueCore4, das Bluetooth-Modul BlueBearHCI und der Bluetooth-Protokoll-Stack BlueZ bereits zertifiziert, so dass nur wenige Tests zum Nachweis der korrekten Integration der einzelnen Komponenten durchzuführen waren. [6]

III. PROJEKTABLAUF

A. Simulation des Netzaufbaus

Im ersten Schritt wurde der in [7] beschriebene Netzaufbaualgorithmus im Netzwerksimulator entwickelt. Aus Mangel an Bluetooth-Unterstützung durch die verbreiteten Netzwerksimulatoren OMNeT++ und NS3, wurde eine eigene ereignisorientierte Simulationsumgebung eingesetzt. Die Implementierung des Algorithmus erfolgte so, dass derselbe Quelltext ohne Anpassung im Simulator und auf der Sensorplattform eingesetzt werden kann.

B. Routing und Forwarding

Die Kontrolldaten des Scatternet-Algorithmus werden über RFCOMM[8]-Verbindungen übertragen. Um eine saubere Architektur zu realisieren, sollte der Scatternet-Algorithmus nicht das Routing und Forwarding der Anwendungsdaten übernehmen. Die Netzknoten sollen über ihre Ethernet-Schnittstellen die Funktionalität eines Ethernet-Switch bereitstellen. Im Projekt wurde das erreicht, in dem parallel zu jeder Kontrollverbindung eine Datenverbindung unter Verwendung des Protokolls BNEP [9] aufgebaut wurde. Die Endpunkte dieser Verbindungen werden unter Linux als virtuelle Netzwerkgeräte realisiert. Unter Verwendung der Linux-Software-Bridge werden alle BNEP-Netzwerkgeräte und das lokale Ethernet-Netzwerkgerät jedes Knotens über einen Software-Switch gekoppelt.

C. In-House-Entwicklungs- und Testnetz

Parallel zur Entwicklung des Netzaufbaualgorithmus produzierte unser Projektpartner lesswire AG die ersten Sensorknoten. Mit den ersten zur Verfügung stehenden Knoten wurde ein In-House-Entwicklungs- und Testnetz aufgebaut. Die Knoten wurden im Institutsgebäude verteilt und so konfiguriert, dass sie ihr Betriebssystem über Ethernet-Verbindungen von einem zentralen Server beziehen. Das ermöglicht den einfachen Austausch von Software-Komponenten während der Entwicklung und beschleunigt den Entwicklungsprozess.

Abbildung 1 zeigt die Sensorknoten für das In-House-Testnetz im Probebetrieb. Um die Knoten auf dem engen Raum betreiben zu können, wurden die Antennen der Knoten mit Dämpfungsgliedern versehen, um die Übersteuerung der Eingangsstufen zu verhindern.

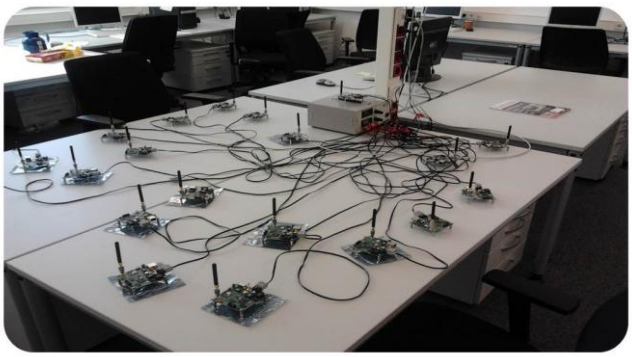


Abbildung 1. Die Knoten des In-House-Testnetz im Labor vor dem Deployment.

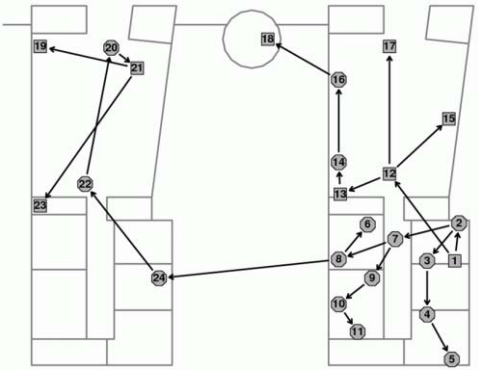


Abbildung 2 Deployment und Topologie des aufgebauten Scatternet des In-House-Testnetzes.

Für das Deployment im Institutsgebäude wurden die Dämpfungsglieder entfernt. Abbildung 2 beschreibt die Verteilung der Knoten über zwei Gebäudeteile und zwei Etagen. Gezeigt wird ein Ausschnitt des Grundrisses des Institutsgebäudes. Eingezeichnet sind die Knoten als nummerierte Symbole. Quadrate stehen für Knoten in der ersten Etage, Kreise für Knoten in der zweiten Etage.

Die Sensorknoten ermöglichen das Tracken von anderen Bluetooth-Geräten wie z.B. die Smartphones der Angestellten und die Aufzeichnung derer Bewegungsprofile [10]. Folglich wäre für die Errichtung des Testnetzes die Zustimmung des Betriebsrats notwendig gewesen [11]. Um dies zu umgehen, verwenden die Sensorknoten des Testnetz statt des Generic Inquiry Acces Code(GIAC) einen Dedicated Inquiry Access Code(DIAC) für den Inquiry-Prozess. Von Standard-Bluetooth-Geräte können so nicht mehr per Inquiry ermittelt werden, die Netzknoten können sich nur untereinander finden.

D. Installation auf einer Testanlage

Mit der Verfügbarkeit zertifizierte Hardware und eines stabilen Netzaufbaualgorithmus erfolgte die erste Installation in einer realen Photovoltaik-Anlage, die als Testumgebung neuer Wechselrichter-Firmware dient. Die PV-Module der Anlage sind auf den Dächern zweier Werkshallen montiert; die Wechselrichter befinden sich geschützt im Innern einer Werkhalle. Dort wurden auch die Netzknoten installiert.

Die Anlage besteht aus 30 drahtgebunden mit dem Firmennetz verbundenen Wechselrichtern. Von 29 Wechselrichtern wurde die drahtgebundene Verbindung zum Firmennetz getrennt und durch einen Netzknoten ersetzt. Ein 30. Netzknoten wurde als Gateway an das Firmennetz angeschlossen. Der 30. Wechselrichter diente als Referenz zum Test der Netzwerkverbindung zur Testanlage.

Die verbauten Netzknoten speichern die Firmware im Flash-Speicher. Updates und das Herunterladen der Log-Dateien erfolgten über FTP- und Telnet-Zugang auf den einzelnen Knoten unter Verwendung des aufgebauten Scatternet. Der Zugriff auf Knoten, auf denen der Scatternet-Algorithmus versagte, wäre weder über das Funknetz noch über das drahtgebundene Netzwerk möglich gewesen. Um an die Log-Dateien dieser Knoten zu gelangen, wurde zusätzlich ein 31. Knoten, auf dem der Scatternet-Algorithmus deaktiviert war, an das drahtgebundene Netz angeschlossen. Von diesem Knoten aus wurden manuell Bluetooth-Verbindungen zu „abgestürzten“ Knoten aufgebaut, um die Log-Dateien zur Analyse herunterzuladen.

Der Testbetrieb zeigte, dass der entwickelte Scatternet-Algorithmus häufig versagte, wenn die Netzknoten nicht gleichzeitig sondern erst nach und nach booten. Dieser Effekt tritt in den PV-Anlagen bei Sonnenaufgang auf, da das Licht der aufgehenden Sonne nicht alle PV-Module gleichzeitig erreicht. Grund war die ursprüngliche Initialisierungssequenz der Netzknoten. Die Knoten wurden bereits kurz nach dem Start für die Knoten in ihrer Umgebung sichtbar, konnten aber noch keine Verbindungen entgegennehmen, was im Scatternet-Algorithmus nicht vorgesehen war. Zur Lösung des Problems wurde die Initialisierungssequenz angepasst, so dass der Knoten erst sichtbar wird, wenn die Initialisierung abgeschlossen ist. Außerdem wurde der Scatternet-Algorithmus erweitert, um den Fall stabil weiterarbeiten zu können. Ein weiteres Fehlverhalten wurde ausgelöst, wenn ein Knoten plötzlich abgeschaltet wurde, wenn z.B. der Wechselrichter nicht mehr ausreichend Energie liefert. War der Knoten z.B. gerade dabei, sich ins Netz einzuhängen, verblieb das Netz nach Ausfall des Knotens in diesem Zustand. Um dies zu verhindern, wurden sämtliche Zustände des Scatternet-Algorithmus mit Zeitgrenzen versehen. Wird eine Zeitgrenze überschritten, wird die aktuelle Aktion als fehlgeschlagen gewertet und der Algorithmus geht in einen definierten Ausgangzustand zurück. Weiter zeigte sich, dass das gewählte Verfahren für Routing und Forwarding praxistauglich ist. Im Laufe des Testbetriebes wurde der Algorithmus sukzessive verbessert bis schließlich ein fehlerfreier Betrieb möglich war.

E. Pilotinstallation

Nachdem der Betrieb auf Testanlage mehrere Monate zuverlässig lief, wurde ein Photovoltaik-Kraftwerk mit Knoten ausgestattet. Bei der Anlage handelt es sich um eine mehrere Hektar umfassende Freifeldanlage, auf der im endgültigen Ausbau 39 Knoten installiert wurden. Plan und eine beobachtete Netztopologie sind in Abbildung 3 zu sehen.

Als wesentliches Problem stellte sich die Installation der Antennen der Netzknoten heraus. Die Netzknoten wurden in der Nähe der Wechselrichtern, die sich wie in Abbildung 4 zu sehen unterhalb der PV-Module befinden, installiert.

Sind die Antennen unterhalb der PV-Module installiert, werden die Funksignale durch PV-Module abgeschattet. Funkverbindungen können nur parallel zu den Modulreihen aufgebaut werden. Überragen die Antennen die PV-Module zu weit, kann es auf benachbarten Modulreihe zu Schattenwürfen durch die Antenne und zu Ertragseinbußen kommen.

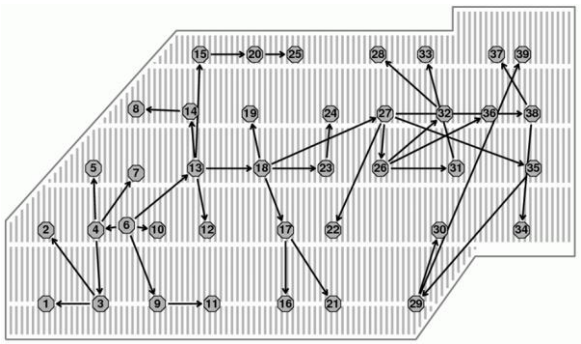


Abbildung 3 Deployment der Knoten auf der PV-Anlage. Die Pfeile stellen die aufgebauten Bluetooth-Verbindungen da.

Abbildung 4 Position der Wechselrichter zwischen den PV-Modulreihen in der Pilotanlage.

Nach anfänglichen Versuchen mit SMD-Keramikantennen, die auf Grund ihrer ungünstigen Ausbreitungscharakteristik nur schlechte Ergebnisse lieferten, kommen jetzt Stabantennen, die maximal 20 cm über die Modelkante herausragen, zum Einsatz. [12]

IV. Fazit

An Hand des Beispiels einer drahtlosen Vernetzungslösung für Photovoltaik-Anlagen wurde aufgezeigt, wie die Entwicklung eines industriellen drahtlosen Sensornetzes erfolgte. Schwerpunktmäßig wurden die praktischen Probleme, die während des Projekts auftraten und die dafür gefundenen Lösungen dargestellt.

V. Verwandte Arbeiten

A. *Great Duck Island*

Die Knoten des drahtlose Sensornetzes auf der Insel Great Duck Island erfassen das Brutverhalten der Inselvögel und das Mikroklima auf der Insel. Die Daten werden an einer Basisstation gesammelt. Über das Internet können verschiedene Forschungseinrichtungen auf die erfassten Daten zugreifen. [13] Die Quelle enthält keine Informationen über den Zulassungsprozess der Sensorknoten. Jedoch ist im Gegensatz zum EU-Raum in den USA eine Konformitätserklärung des Inverkehrbringers nicht ausreichend. Die Zulassung muss bei der FCC (Federal Communications Commission) beantragt werden. Zu den einzureichenden Unterlagen gehören neben Messprotokollen über Abstrahlungen des Produkts und Produktbeschreibung auch die vollständigen Stücklisten und Konstruktionspläne des Produkts. Weisen die eingereichten Unterlagen die Konformität des Produkts mit den Regularien der FCC nach, wird dem Produkt eine FCC-ID zugeteilt. Alle eingereichten Unterlagen werden von der FCC auf ihrer Web-Seite http://www.fcc.gov/ veröffentlicht. Will ein Hersteller die Veröffentlichung von Stücklisten und Konstruktionsplänen verhindern, muss er seinen Unterlagen einen Antrag auf Vertraulichkeit hinzufügen. [6]

B. *WSAN4CIP*

Im Rahmen des Projekts WSAN4CIP wurde die Frischwasserleitung der Wasserwerke von Frankfurt(Oder) mit eine Kette von Sensorknoten, die entlang der Rohrleitung installiert sind, überwacht. Die Sensorknoten lösten die Überwachung mittels eines kostenintensiveren Glasfaserkabels ab. [14] Die Quelle nennt ebenfalls keine Details über die Zulassung der Sensorknoten. Es kommen jedoch dieselben Vorschriften wie für im Projekt SolarFlex zur Anwendung, wobei entsprechend Frequenzband und Einsatzumgebung der Sensorknoten andere Normen nachzuweisen sind.

Danksagung

Diese Arbeit wurde durchgeführt ZIM-Projekts SolarFlex gefördert vom Bundesministerium für Bildung und Forschung (BMBF) unter der Referenznummer KF2123403DF9. Die Netzknoten sind eine Entwicklung der Fa. lesswire AG.

Referenzen

[1] "Photovoltaik-Wechselrichter-Programm," ed: REFUsol, 2013.
[2] "Specification of the Bluetooth System 2.1+EDR," ed, 2007.
[3] "BlueBear - Industrielles Long Range HCI-Bluetooth® 2.0 Modul mit EDR," lesswire AG, Berlin, Data sheetApril 2009.
[4] *BlueCore4-External Product Data Sheet*, 2005.
[5] *BlueZ - Official Linux Bluetooth protocol stack*. Available: http://www.bluez.org/
[6] *Bluetooth SIG Homepage*. Available: http://www.bluetooth.org/
[7] M. Methfessel, S. Peter, and S. Lange, "Bluetooth Scatternet Tree Formation for Wireless Sensor Networks," in *MASS*, 2011, pp. 789-794.
[8] "Part F:1 RFCOMM with TS 07.10 Serial Port Emulation," in *Bluetooth Specification Version 1.1*, ed: Bluetooth SIG, 2003, pp. 393 - 424.
[9] "Bluetooth Network Encapsulation Procotol (BNEP) Specification," Bluetooth SIG2003.
[10] M. Haase and M. Handy, "BlueTrack - Imperceptible Tracking of Bluetooth Devices," *Ubicomp Poster Proceedings,* 2004.
[11] § 87 Abs. 1 Nr. 6 BetrVG - Einführung und Anwendung von technischen Einrichtungen, die dazu bestimmt sind, das Verhalten oder die Leistung der Arbeitnehmer zu überwachen. Available: http://www.kanzlei-hessling.de/de/inhalte-Betriebsratsratgeber/_87_Abs_1_Nr_6_BetrVG_technische_Arbeitnehmerueberwachung/
[12] Technisches Datenblatt REFUconnect [Online]. Available: http://europe.refusol.com/fileadmin/user_upload/pdf/products/Technisches_Datenblatt_REFUconnect_DE.pdf
[13] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," presented at the Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, Atlanta, Georgia, USA, 2002.
[14] S. Peter and G. Weber, "Monitoring and Control of a Drinking Water Pipeline - An Application of Secure Wireless Sensor and Actuator Network," *EURESCOM mess@ge - The Magazine for Telecom Insiders,* vol. 1, p. 15, 2011.

On the Comparability of Indoor Localization Systems' Accuracy

Sebastian Fudickar, Sebastian Amend, Bettina Schnor
Institute of Computer Science
University of Potsdam, Germany
Email: [fudickar, samend, schnor]@cs.uni-potsdam.de

***Abstract*—While indoor localization is gaining increased relevance, the comparability of localization systems' accuracies is rarely given due to variations of the evaluation environments and evaluation methods. To overcome this limitation, the article at hand proposes a simulator for indoor localization systems that assures their comparability and optimization under recorded, realistic conditions. The practicability of the proposed approach is shown by optimizing the parameters and algorithms of a model-based indoor localization system, which achieves lower error-distances and higher success rates than the model-based RADAR system as shown by evaluation.**

I. Introduction

Smart phone applications (such as navigation, information or social-applications) can be enhanced with location awareness. While outdoor localization-technologies such as GPS or cell-tower communication are not applicable for indoor environments due to radio-signal shielding, the radio frequencies of 2.4 GHz (as used by WiFi) or Sub GHz (such as 868 MHz) are well suited for indoor localization and apply the following mechanism. Beacons are placed within buildings and regularly transmit messages. Mobile devices estimate their current position from the received messages typically via the received signal strength (RSS). A challenging aspect of RSS based localization is the influence of the radio wave distribution and environmental noise - resulting in varying localization accuracies (typically measured as error-distances) for different setups and buildings. Therefore, a meaningful comparison of the quality of specific algorithms requires the exclusion of factors such as the influence of the radio wave distribution and environmental noise (e.g. by testing the systems under identical conditions or recorded measurements).

The article at hand proposes a simulator that enables the comparison of RSS-based localization algorithms with prerecorded radio-traces, to overcome these drawbacks and enable a meaningful comparison of localization algorithms and their optimization.

Section II gives an overview of current indoor localization systems and evaluation techniques. The simulator is discussed in section III and the new indoor localization system which's parameter settings and algorithms are optimized for optimal accuracy with the simulator is proposed in section IV. As shown in the evaluation in section V, the developed localization system achieves a median error distance of up to 4.14 m in the unoptimized nursery home, which is more than 25% more accurate than the RADAR localization system [1].

II. Related Work

Existing radio based indoor localization systems (as summarized in Table I) typically use either model-based or empirical localization algorithms that estimate the position based on large-scale variations of the RSS instead of applying distance measures and triangulation (as used by GPS). Therefore, the RSS of multiple router is matched with a radio-propagation map of the building which is either generated by empirical measurements or modeled based on the position of the transmitting nodes. Empirical algorithms are more common since achieving lower error-distances than model-based ones as shown by the RADAR indoor localization system [1], while the primer requires initial laborious empirical measurements in each supported building.

However, the comparability of localization algorithms' accuracy is challenging since the error-distances are significantly influenced by the following varying measurement conditions The **node densities** among the evaluations of these systems typically varies significantly (ranging from 0.4 to 0.003 $\frac{nodes}{m^2}$). Denser node placement increases the probability of line of sight transmissions (and thereby are more accurate). Also the **size of evaluated areas**, which is typically smaller than $1000\,m^2$ are chosen heterogeneously. The maximal error distances are equidistant to the size of the evaluation area and thereby affect localization runs.

Aside, the heterogeneous specification of the resulting error distances (e.g. by median or average) is as well challenging for a meaningful interpretation of the influence of specific algorithms and radio frequencies on the localization accuracy. Most evaluations determine only the error-distance and do not investigate other relevant metrics such as the availability and floor error rate.

The *EvAAL* competition [2] offers a normed benchmark of indoor localization and tracking systems and evaluates the complexity of deploy and configuration, the user acceptance, the interoperability, availability and accuracy within a specific environment. The applied metrics cover the essential criterion for the applicability of indoor localization systems, but the participants have free hand in preparing the environment (e.g. by selecting random and undocumented node densities). Thereby, the accuracy is not related to a specific node density and can not be used for direct comparison of localization algorithms' accuracy.

TABLE I
LOCALIZATION ACCURACY

Localization system	error-distance (m)	Precision	Node-density	Nodes	Size (m^2)	Year	Type	Transceiver
ITRI	2.13 - 7.17	average	0.0125	5	400	2005	Empirical	Wi-Fi
Ekahau	3.72	average	0.04	5	108	2009	Empirical	Wi-Fi
RADAR [1]	2.94	median	0.003	3	980	2000	Empirical	Wi-Fi
RADAR [1]	4.3	median	0.003	3	980	2000	Model based	Wi-Fi
n-Core Polaris	0.97	average	n.a.	n.a.	70	2011	n.a.	ZigBee
Compass	1.65	average	0.028	9	312	2006	Empirical	Wi-Fi
Horus	1.4	90%	0.011	21	1766	2005	Empirical	Wi-Fi
IIS-Frauenhofer	2.5	n.a.	0.1	8	80	2005	n.a.	868 MHz
OWLPS	4.52	average	0.007	5	690	2010	Empirical	Wi-Fi

III. SIMULATION ENVIRONMENT

The proposed simulator enables the meaningful comparison of localization systems regarding their accuracy and availability. By using recorded RSS readings of tracks (track records) and specific positions, the influence of unintended factors (such as node-density) is excluded and even a validation for multiple buildings can be achieved. Furthermore, specific localization algorithms can be optimized by automatic testing of parameter levels. Aside of the optimization and comparison of localization algorithms, the simulator can optimize pathloss-models (for several buildings at once) by multi variance testing based on steady records.

Steady records consist of RSS measurements, taken at various specific positions with the measurement device facing to the four cardinal points (to cover influence of human shielding). These recordings are used to optimize the pathloss models by parameter tuning. The records specify the utilized radio frequency, the measurement time, the orientation and the position (as x and y coordinates and the floor).

Aside, **Track records** contain RSS readings of surrounding nodes, which are collected while walking on predefined paths. To identify a measurement position, tracks consist of straight lines that connect so called checkpoints, where the proband confirms its arrival and departure, which are recorded for later position estimation. The RSS readings are stored with a timestamp and the ID of the transmitting node. These records have precise check-point positions annotated to achieve accurate positioning. The track passed a straight line between two checkpoints with a steady speed and the position is interpolated during simulation from the time of recording and the arriving at the next checkpoints. Consequently, passing a path element in steady speed is essential for accurate interpolation results.

A. RSS record collection process and Environment

The simulator evaluates localization systems based on RSS records which were recorded under the following setups.

The recordings took place in the computer science faculty building at Potsdam University and a nursery home (see Table II). The records of the nursing home were collected during representative daily activities, while the records in the faculty building were collected on a weekend with low activity.

The node placement based on practicability - without interrupting the daily work.

The recorded RSS samples were collected by a TinyOS application (running the B-MAC protocol) and the Sub GHz (CC1101) transceiver of the *Efficient Mobile Unit* (EMU)[3]. Mica2 nodes transmitted with 0 dBm over 868 MHz and are placed at a height of ca. 1 m in the building. In both measurement setups, the EMU is worn at the hip and is connected via USB to a laptop that is worn at the shoulder (to minimize the influence on signal distribution). The laptop runs a *measurement controller* in which the current position is specified (for steady records) and arriving or leaving checkpoints is confirmed (for track records). The controller also triggers recording on the EMU.

Aside of track-records, steady records were collected at a grid of one m^2 for accessible areas. For each position and direction 10 seconds of Sub GHz messages were recorded (which let in average to 11 Sub GHz messages in the nursing home and 13 Sub GHz messages and 50 Wi-Fi messages in the faculty building).

B. Simulator

The simulator consists of two separate modules for the evaluation of path-loss models and for the evaluation of the localization systems' accuracy.

The **pathloss-model simulator** evaluates the accuracy as the distance between the modeled signal strength in [dBm] and the average of the received signal strength measurements for each measured position and transmitter. Furthermore, an accurate pathloss model can be calculated by linear regression.

The **localization system simulator** encapsulates the evaluated localization system and operates event-based to represent the recorded conditions realistically. It can either optimize parameter settings and algorithms or determine the accuracy of localization systems.

Therefore, track records are loaded and the included RSS measurements are precisely timed handed to the localization system. Aside of the RSS measurements, the building information (specifying wall and node positions) and the configuration, which specifies either concrete parameter settings or parameter levels (for multi variant tests) are loaded.

Once the localization system has generated the radio-propagation model based on the building information, the

TABLE II
CHARACTERISTICS OF THE ENVIRONMENTS AND RECORDS

Building	Size m^2 (incl. outdoors)	Floors	Sub GHz nodes	Positions (in rooms)	Tracks	Average Track duration (minutes)	Track distance (m)
Faculty Building	3110	3	26 (0.008)	166 (10.2%)	6	13.78	355
Nursing Home	2627 (3532)	2	18 (0.007)	121 (5.8%)	5	17.88	443

simulation starts and events for each RSS readings are forwarded to the localization system. Localization requests are sent from the simulator to the localization system according to the configured localization interval. These requests initiate localization runs which estimate the current position based on the recently forwarded RSS readings. The position that is estimated by the localization algorithm is then stored aside with the calculated position of the recording unit at this time, for later processing at the end of a simulation run. Both positions consist of the floor level and the x, y distances from a point of origin. The estimated position may be null, indicating a failed localization run.

$$Pos_{cur} = Pos_{n-1} + \frac{(Pos_n - Pos_{n-1})}{(Time_n - Time_{n-1})} * (Time_{cur} - Time_n) \quad (1)$$

The measurement position Pos_{cur} at a timestamp $Time_{cur}$ is calculated based on the previous Pos_{n-1} and the upcoming checkpoint Pos_n. If both checkpoints are identical, they represent the measurement position. Otherwise, the position is interpolated by the time of departure from the last checkpoint and the time of arrival at the next checkpoint as shown in Equation 1.

After simulating all track records the results of all localization runs are evaluated. The accuracy is evaluated regarding the following metrics:

- **Localization errors** occur if a localization did not return any position and thereby was unsuccessful.
- **Floor errors** occur if the localization estimated building floor is incorrect.
- The **error distance** is calculated for successful localizations, that determined the correct floor, as the Euclidean distance between the estimated and interpolated positions.

With this simulator the localization system that is described in the next section was optimized for these metrics.

IV. LOCALIZATION SYSTEM

The indoor localization system runs on mobile devices and estimates the device positions via the RSS of messages that were received from surrounding beacons. A model-based approach was chosen to overcome the laborious collection of empirical data. As shown in Figure 1, the device's position is estimated by matching the RSS values to a radio-propagation model, which is generated (during system initialization) by a pathloss-function from a building-map that specifies the position of beacons and walls. The pathloss can be calculated with several *pathloss-functions* (see Table III), which i.a. vary regarding the considered attenuations (e.g. absorption of walls and floor). The modeled signal strength is calculated by subtracting the calculated pathloss from the beacon's transmission power and for each beacon at a position-grid which's *granularity* can be configured.

Once the initialization phase is completed and the radio-propagation model is generated, localization runs are started regularly (e.g. every 20 seconds), which consist of the following processing steps (see Figure 1).

The received messages are transmitted regularly (e.g. with a maximal frequency of 1 Hz) by neighboring beacons according to a *transmission interval* and are considered by the localization for *RSS TTL*. The message fields specify i.a. the beacon ID, a building ID and after reception are extended by the RSS.

Messages with a RSS below a threshold (of *minimal strength*) and from *other floors* are omitted. Furthermore, the amount of considered messages per beacon is limited to the last *maximal messages*.

The floorlevel of the beacon with the strongest RSS is used for localization.

The remaining RSS values per beacon are combined to a single RSS value by a *signal model*, which might consider the distance passed by the mobile device between message receptions.

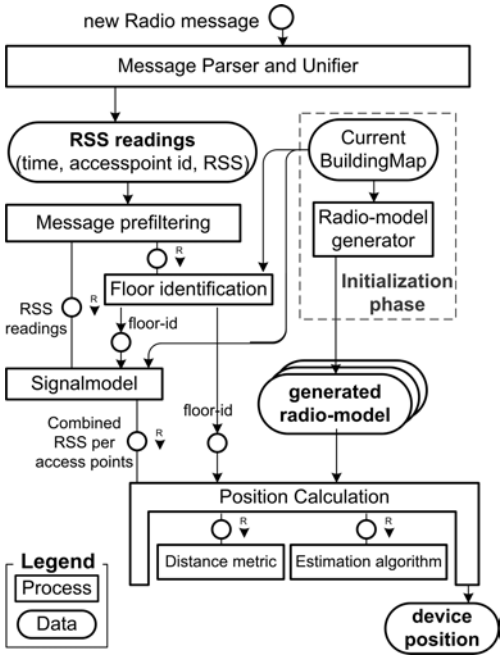


Fig. 1. Schematic of the localization-system

TABLE III
EVALUATED AND CHOSEN FACTORS AND LEVELS OF THE LOCALIZATION SYSTEM VERSIONS

Parameter	Range	Best Guess	Radar	Optimized
Pathloss function	FreeSpace, Linear Attenuation, Log-Distance, Gahleitner, Devasirvatham, Log-Distance Zhao, KeenanMotley, ITU Indoor (with/without) Walls	ITU with Walls	ITU with Walls	ITU with Walls
RSS TTL	1 - 40 s	10 s	1 s	10 s
Granularity	0,5 - 5 m	1 m	1 m	4.5 m
Signal model	Last, Best, Average, Weighted Average, Median, KernelSmoother, Linear Regression	Average	Last	KernelSmoother
Min strength	-95 - -80 dBm, none	none	none	-91 dBm
Max messages	1-10, all	all	all	all
Transmission Interval	1 - 40 s	1 s	10 s	1 s
Localiser	Nearest neighbours (NN), Weighted nearest neighbors (WNN)	NN	NN	NN
K-Neighbors	1 - 50	10	1	11
Distance metric	Euclidean, Manhattan, Mahalanobis	Manhattan	Euclidean	Euclidean
Other floors	true, false	true	false	true

For the identified floor, the distance (in dBm) between the modeled and the summarized RSS is calculated by a *distance metric* per beacon and position of the radio-propagation model.

Next, the device position is calculated by a *localiser* function such as the K-nearest neighbor algorithm, where the positions of the *K-Neighbours* with the lowest RSS distance are interpolated to estimate the final device position.

The localization system can be implemented with various algorithms. E.g. various pathloss models and signal models can be applied and various parameter settings can be adjusted to optimize the accuracy. Therefore, the optimal algorithms and parameter settings were identified via simulation.

V. EVALUATION

The accuracy of the optimized indoor localization system is evaluated and compared with educated best guess settings and the model-based RADAR localization algorithm (used as reference) for the nursery home and the faculty building. In advance, nine pathloss models were optimized and the optimal accuracies per model are compared with each other for steady records of both buildings. The identified optimal pathloss-model and its configuration is used further on. The parameter settings and algorithms of the localization system are optimized for the faculty building (see Table III for the identified optimal settings). Next, the accuracies (concerning the error-distance, floor-error rate and localization-error rate) of the resulting optimized version, a version based on educated best guess and the reported settings of the model-based RADAR system are evaluated for both buildings (see Table IV).

TABLE IV
ERROR DISTANCES OF THE LOCALIZATION SYSTEMS

Algorithm	Building	Error-distance (m)				Floor errors	Local. errors
		1.IQR	2.IQR	3.IQR	avg		
Best Guess	faculty	3.27	5.61	8.8	6.56	4%	12%
Optimized	faculty	2.62	4.40	7.19	5.50	4%	12%
Radar	faculty	4.10	6.05	9.80	7.77	2%	40%
Best guess	nursery	2.6	4.02	6.79	5.06	0%	6 %
Optimized	nursery	2.58	4.14	6.28	4.71	6%	1%
Radar	nursery	2.95	5.21	10.07	8.37	3%	16%

The optimized version has a median **error-distance** below 4.5 meters in both buildings. The error distances at the first and the third inter quartile range (IQR) are both lower than the ones of the RADAR system and the educated best guess. Furthermore, the optimized version achieves a far better **localization error rate** than the *Radar* algorithm. While the **floor error rate** is (with up to 6%) slightly higher than the one of the Radar algorithm it is still sufficient to prevent wrong estimations by considering two consecutive floor interpretations.

Aside of the high accuracy in buildings with a low beacon-density, the chosen model-based localization system is even applicable to buildings (of similar types) without further optimizations as indicated by the results of the nursery home. By achieving a similar accuracies in this building, the deployment in additional buildings may be limited to installation of beacons and the creation of a building map, but might exclude manual recalibration, while achieving error-distances that are sufficient for the localization in rooms.

VI. CONCLUSION

The article at hand proposes a simulator for the comparison and optimization of RSS based indoor localization systems. It enables the comparison of localization systems' accuracy under prerecorded conditions and enables reproducibility. The simulator overcomes the necessity of regular test runs, while assuring comparability of (intermediate) results. Furthermore, the simulator (e.g. used on computing clusters) enables the automated multi variant testing of large parameter sets.

The practicability of the simulator was shown by optimizing and evaluating a new model-based localization system, which achieves lower error-distances and significantly less localization errors than the model-based RADAR algorithm under identical conditions.

REFERENCES

[1] 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*, vol. 2, no. c, pp. 775–784, 2000.

[2] http://evaal.aaloa.org/; accessed May 1, 2013.

[3] S. Fudickar, M. Frohberg, S. Taube, P. Mahr, and B. Schnor, "An energy efficient mobile device for assisted living applications," in *2012 IEEE Online Conference on Green Communications (IEEE GreenCom'12)*, Sep. 2012.

Towards Application-Centric Deployment of Low-Power Wireless Networks

Matteo Ceriotti, Alexandr Krylovskiy, Klaus Wehrle
Chair of Communication and Distributed Systems (ComSys),
RWTH Aachen University, Germany
http://comsys.rwth-aachen.de

***Abstract*—Low-power wireless embedded networks have reached the stage of being deployed in an increasing number of application scenarios. However, despite the plethora of proposed solutions, deploying such systems remains a challenging task. In particular, there is still a clear discrepancy between having a generically functional network and a system capable of efficiently matching specific application requirements in the given scenario at hand. In our current line of research, we aim at identifying a deployment approach that accounts for application requirements and the specificity of the scenario where the system is installed. In particular, we identify the need to observe the network topology during deployment and feed this information into a simulator capable of exploring different application configurations. Here, we introduce our approach and present some preliminary results, highlighting our current research agenda.**

I. Introduction

Low-power wireless networks, most commonly referred to as Wireless Sensor Networks (WSNs) or Cyber Physical Systems (CPSs), have been proven to be effective solutions in several applications. They can be employed as scientific instruments in harsh environments, e.g., volcanos [1], as well as industrial products in operational systems [2]. As a result, low-power wireless networks are slowly being deployed in various scenarios, ultimately at the benefit of society at large.

In contrast to this trend, the deployment of operational networks is still a tedious task, which mostly relies on the experience of the people installing the system. As different environments have clearly different impacts on the deployed network [3], even the acquired experience has limited applicability: minor changes in the environment will result most likely in a different system behavior. Due also to the inherent complexity of these networks, systems are currently deployed with rule of thumb, ultimately resulting in an undependable fulfilling of the intended application requirements.

By looking at the typical application life cycle, we can approximately identify the following steps: 1) the software and hardware is developed, debugged, and validated in controlled scenarios, e.g., laboratories; 2) the system is deployed in the operational scenario and let run; 3) the application requirements are evaluated over a long period of time, as application data are gathered; 4) upon failure or clear discrepancy between the deployed network and the application needs, the system configuration is adjusted. By looking at this process, we ask ourselves: why do not we guide the system configuration by evaluating the application requirements *during* deployment?

We identify a possible approach to address such a challenge by incorporating application requirements in the deployment process. We describe the expected system performance as an application profile composed of several metrics, e.g., throughput, latency, and lifetime. We then simulate the deployed system by taking as input the observed network topology of the installed network. Currently, we are working on minimizing the gap between the simulated and the real network, so that the simulation can reliably reproduce the running system. Afterwards, we aim at identifying the application profiles that a deployed network can support, exploring possible system configurations and highlighting network bottlenecks. With this information, we will be able to guide the system reconfiguration during deployment to reliably support the user needs.

II. Main Approach

We aim at evaluating application requirements as early as possible in the deployment process, enabling the estimation of the target application performance in the network under deployment. With such information, decisions can be made on resources to add or replace; furthermore, the system configuration can be optimized to best satisfy the expected usage.

To support this vision, we need primarily to acquire extensive information about the deployed network so that a reliable and accurate model of its behavior can be built. This is a challenge in itself, as monitoring protocols running in parallel to the actual application in the real system impact the observed behavior. Therefore, such type of observations are typically severely restricted in the amount of information they collect, resulting in overall insufficient visibility on the internal system functioning. Similarly, many different configurations may need to be tested to explore the available application tradeoffs, which would require an increasing time and deployment effort if explicitly tested in the operational network.

We address these challenges by reproducing the behavior of the network in simulation. In particular, we (i) accurately measure the network topology, by observing the complete network connectivity graph and characterizing each link of the deployed system. We use this information to (ii) build accurate link models for the target network in the simulator and (iii) simulate individual nodes, their energy consumption, and the overall network behavior. With this information we are finally able to (iv) evaluate the satisfiability of application

Fig. 1. Smartphone connected to a TelosB, gathering connectivity information; on the smartphone, link qualities are visualized through semaphores.

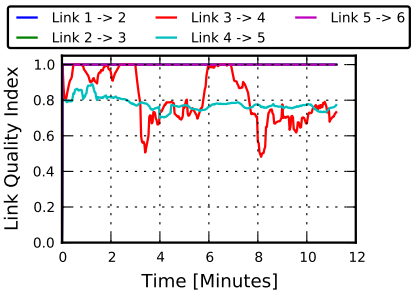
requirements in a controllable environment, so that many simulations can be run and different configurations tested.

The resulting architecture takes as input both the application requirements and the network connectivity measurements. Through a device interfaced with a smartphone, the user can conveniently gather data and visualize network connectivity while the system is deployed; a dedicated monitoring protocol runs on each device to observe link characteristics. With such information, we run network simulations in COOJA [4]; with MSPsim [5], it is possible to accurately emulate the behavior of MSP430-based platforms, such as the common TelosB [6]. We are currently addressing the discrepancy between the behavior of the real network and the simulated one. In our future work, we aim at using this set of tools to provide the user with information about application requirements satisfiability and critical network areas while deploying the network.

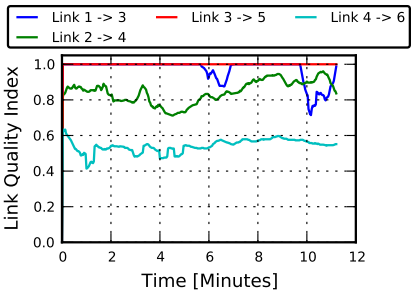
III. Observe the Network Topology

Obtaining accurate measurements of link characteristics in a running system is challenging. In particular, we need to gather complete information about the connectivity graph in order to correctly analyze the impact of possible interferences happening in the wireless broadcast shared medium. Therefore, each link needs to be probed in isolation, avoiding the impact of concurrent accesses to the channel in the same collision domain. This requires a careful scheduling of collision-free probing messages to measure each link. Moreover, during the deployment process, we cannot rely on the presence of any sink, asking for fully decentralized solutions. Finally, we would like to expose such information conveniently to the user.

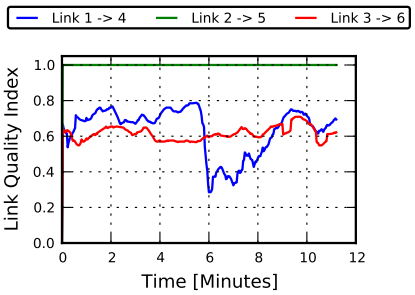
To observe the links, we use Reins-MAC [7], a fully-decentralized TDMA protocol that schedules collision-free access to the medium and dynamically adjusts the schedule depending on the devices in the same collision domain. Following the created schedule, nodes can efficiently and accurately measure the link characteristics, building a local view of all the available incoming links and their statistics (over a sliding time window). At the same time, nodes can use the communication schedule to spread in the network the gathered information about the neighborhood. Nodes periodically transmit their own local view of the network, and also forward the information



(a) 5 m links



(b) 10 m links



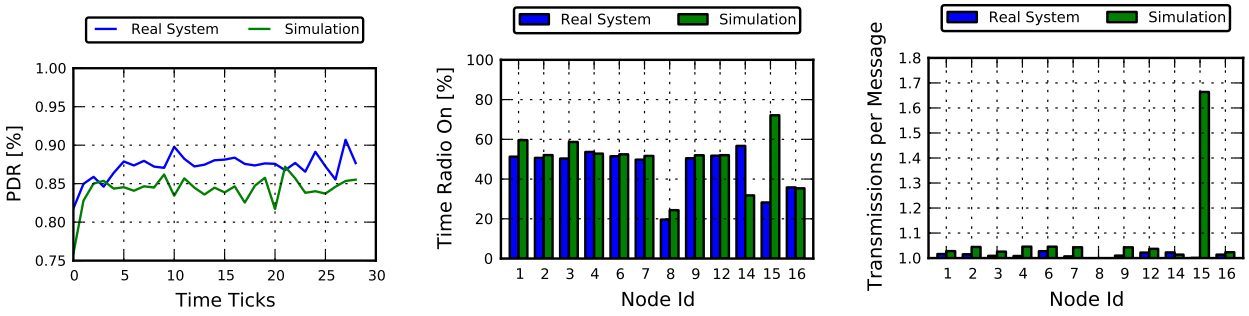
(c) 15 m links

Fig. 2. Link quality index for different link lengths in an indoor scenario; the index is a weighted average of RSSI and PDR.

received from other devices. As a result, each node in the network has complete knowledge of the whole network graph.

Given that each node has complete view of the system connectivity, we can either connect to a deployed device or use one additional node to gather the complete knowledge of the network graph. In particular, to make the deployment process more practical, we developed a simple Android application that processes data sent from a TelosB device to a smartphone connected via USB (see Figure 1). On the smartphone, we conveniently visualize the quality level of the links through simple semaphores. In this way, not only we are able to gather information that we can later exploit in the validation of the supported application profiles, but the user can also identify connectivity problems and deploy a functional network.

We ran several experiments at our chair, by deploying different network setups in an indoor scenario. In Figure 2, the measurements taken in one experiment are depicted. For each link, we compute a weighted average of RSSI and PDR; as a result, also links with high PDR but low RSSI (close to the receiver sensitivity) are detected as unreliable. The nodes were deployed at a fixed distance of 5 m one from the other.



(a) Network delivery rate (b) Radio duty cycle (c) Average number of send operations

Fig. 3. Comparison of simulated and real network behavior.

Despite the indoor communication range of 50 m indicated in the datasheet, unreliable links already appear at 15 m. More interestingly, lower distances already manifest high variation in the link quality, even at a short distance of 5 m. Modeling these effects without direct measurements requires a very detailed description of the scenario; e.g., doors, windows, and walls, will have different impact on the wireless signal propagation, also depending on the involved materials. Providing such a detailed description is complex and arguably impractical.

IV. Towards Validation of Application Profiles

With the data collected as described in Section III, we have available the description of the network in its operational environment. We can, therefore, build an equivalent model of the network and run simulations of the application behavior in a controlled scenario, gaining reproducibility and visibility. Having the network graph on the smartphone, we could upload such information to a remote, more powerful server where multiple application configuration are tested.

Before testing application configurations, it is important to measure the adherence of simulation results to the behavior of the real network. For this reason, we execute some preliminary experiments in an indoor testbed at our chair. We ran a simple data gathering application based on CTP [8] as routing protocol, and BoX-MAC [9] as underlying CSMA MAC protocol with radio duty cycling. Before running the application in the real system, the network connectivity is measured; these measurements are used later as input to the simulation.

In Figure 3, a simulation run is compared against the behavior of the application in the real testbed. From the results, a rough match is evident. The discrepancies may be caused by different timings, e.g., in the order in which the nodes are booted or in the backoff timers used to access the shared medium. These preliminary experiments show that it is possible, based on the connectivity measurements taken directly from the deployed network, to analyze the real system in simulation. We are now working to further reduce the mismatch by incorporating more information in the simulated radio model, in particular improving the interference model.

V. Research Agenda

In this work, we aim at introducing a technique to explore the application requirements that a real network can satisfy while being deployed. We have successfully built a monitoring tool to accurately measure the network graph in an installed system. With this information, we have been able to closely approximate the behavior of the real network in simulation. This will offer the possibility to test different system configurations and verify which requirements are likely to be satisfied by the deployed system. After increasing the adherence of the simulation to the real system behavior, our long-term goal is to explore solutions to guide modifications to the deployed network so that requirements, currently not satisfied, can be met. We will then provide the user with a comprehensive tool to guide the deployment process and match the expectations of the application for which the system is installed.

References

[1] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. of the 7th Symp. on Operating Systems Design and Implementation (OSDI)*, 2006.

[2] M. Ceriotti, M. Corra, L. D'Orazio, R. Doriguzzi, D. Facchin, Ş. Gună, G. P. Jesi, R. Lo Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele, "Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels," in *Proc. of the 10th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2011.

[3] L. Mottola, G. P. Picco, M. Ceriotti, Ş. Gună, and A. L. Murphy, "Not all wireless sensor networks are created equal: A comparative study on tunnels," *ACM Transactions on Sensor Networks*, vol. 7, September 2010.

[4] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. of the 31st Conf. on Local Computer Networks (LCN)*, 2006.

[5] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt, "Mspsim – an extensible simulator for msp430-equipped sensor boards," in *Proc. of the 4th Europ. Conf. on Wireless Sensor Networks (EWSN), Poster/Demo session*, 2007.

[6] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proc. of the 4th Int. Symp. on Information Processing in Sensor Networks (IPSN)*, 2005.

[7] M. Ceriotti and A. L. Murphy, "A MAC contest between LPL (the champion) and Reins-MAC (the challenger, an anarchic TDMA scheduler providing QoS)," in *Proc. of the 8th Conf. on Embedded Networked Sensor Systems (SenSys), Demo session*, 2010.

[8] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *(To appear) ACM Transactions on Sensor Networks*, vol. 10, February 2014.

[9] D. Moss and P. Levis, "BoX-MACs: Exploiting physical and link layer boundaries in low-power networking," Stanford University, Tech. Rep. SING-08-00, 2008.

The Crux of OMNeT++ on development for a specific Wireless Sensor Node Platform, A Progress Report

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

***Abstract*—OMNeT++ is a discrete event-driven simulator and is intensively used for testing and validating research results in the area of Wireless Sensor Networks (WSN)s. In the context of the IQlevel project we developed a Wireless Sensor Node Platform (WSNP) and a Multi-Hop Routing (MHR) Protocol, which benefits by our configurable protocol stack model. It allows us to use a single protocol implementation in different environments without any adaptations. Driven by the impressive advantages of the OMNeT++ simulator we wrapped into the early decision to continue our protocol development for our WSNP with Castalia. In this progress report we will depict why these strategy was an aberration, which finished in a sloppy protocol implementation. We will describe five hard to find traps that we had to solve for porting our successfully simulated implementation to our target WSNP. As a conclusion we will mentioned an alternative approach for testing and simulating implementation for specific WSNPs.**

I. Introduction

In the recent years the research in the area of Wireless Sensor Network (WSN)s becomes more and more enlarged. Publications cover a broad variety of topics, e.g. hardware, operating systems, low power systems and wireless communication protocols. Especially the design and the implementation of wireless Medium Access Control (MAC) schemes and routing protocols are deeply investigated. The addressed features as ad-hoc structures, self-organization, self-healing or mobility make the protocols and their development more and more complex. Furthermore, testing and validation become a time-consuming process and cannot be handled with real WSNPs.

Due to the complexity and size of WSN protocols and their deployments the usage of simulation frameworks becomes very attractive. Driven by a similar motivation we used Castalia, which is based on OMNeT++, for testing and validating our Fair Energy Trade (FET)-Multi-Hop Routing (MHR) protocol [1]. The protocol was developed in the context of the IQlevel project [2] and is currently in extension in context of the Aeternitas project [3]. The protocol aims to provide a long living multi-hop network with a mesh structure. We started our development on a real Wireless Sensor Node Platform (WSNP), but due to the complexity of the test and validation runs we early decided to continue our work with the Castalia framework. After finishing the simulation with promising results we were confident that we can immediately move our implementation to our MSP430-based WSNP [4], [5]. But we had to learn in a long nerve-racking process that a successful running implementation in Castalia has not to be suitable for a real WSNP. In this progress report we will describe five traps that were very hard to find and that we had to solve during our porting work. We are convinced that most of these traps may be clear for a skillful WSN engineer, but we also hope that our progress report can motivate researchers to be more carefully when using Castalia / OMNeT++ for their protocol development as well as it helps to explain why OMNeT++ and Castalia are not platforms for testing compiled code for a specific WSNP.

In the following we shortly explain few fundamentals of the OMNeT++ / Castalia framework. In section III we present our approach of a configurable protocol stack model, which allows us to use an unmodified implementation in different environments. In section IV we describe the five major traps that we had to solve for porting our protocol to a real WSNP. We conclude this short paper with a mention of an alternative approach for simulating and testing protocol implementation for a specific WSNP and a short summary.

II. OMNeT++ / Castalia

OMNeT++ is a discrete event simulator for modeling communication networks, multiprocessors and other distributed systems [6]. It is open-source and fully implemented in C++ and designed to support network simulation in large scale. Its development was motivated by the needs of a powerful open-source simulation tool for academic, education and research use. OMNeT++ is available since 1997 and has a large community [7]. There have been registered downloads from over forty universities worldwide and the number of OMNeT++-related projects is still growing.

The OMNeT++ model consists of modules that communicate with message passing. A message can be either objects holding arbitrary data or simple events. Modules are instances of module types, which are written in C++ and use the simulation class library. The user can add functionality to its module

via one of two alternative programming models: (I) co-routine-based programming and (II) event-processing functions. A module of the co-routing-based model runs in its own thread and contains an infinite loop with send and receive calls. The thread gets control by the simulation kernel each time the module receives a message. The event-processing function is called by the simulation kernel and returns immediately after processing the message. Modules can be hierarchically grouped into a compound module, where the number of levels is unlimited. This helps the user to transparently split its implementation in several modules. OMNeT++ supports message passing within a single level of module hierarchy or within a compound module's hierarchical structure. To the outside a compound module acts as a 'cardboard box', transparent relaying messages from their inside. Furthermore, connections can be parametrized with delay, data rate or bit error rate. Especially the module structure and the strict separation of "nodes" and "links" provided by OMNeT++ improve the reusability of the user's model components and simplify their deployment to a real application. An important requirement for OMNeT++ was easy debuggability and traceability of simulation modules. It offers module output windows, inspectors and automatic animation as well as tracking object ownerships, doing ownership-based automatic deallocations and detecting bugs caused by aliased pointers and misuse of shared objects.

Castalia is a simulator for WSN, Body Area Network (BAN) and generally networks of low-power embedded devices. It is based on OMNeT++ and has an advanced channel model, radio model, extends sensing modeling provisions and supports node clock drift. Furthermore, Castalia inherits the excellent modularity, realiability and simulation speed from the OMNeT++ framework [8], [9].

III. A CONFIGURABLE PROTOCOL STACK MODEL

Although the implementation of our FET-MHR protocol was driven by the requirements of the IQlevel project, where Operating System (OS) and hardware were developed from scratch, we decided to use a configurable protocol stack model. This model basically divides the protocol stack in a driver, a protocol and an application layer. These layers provide a basic separation among Hardware Abstraction Layer (HAL), Hardware Independent Layer (HIL) and application.

We are convinced that a well-defined protocol as part of the HIL could be easily ported to different environments by adding small adaptation layers. Although a network protocol may make an extensive use of OS primitives like timers or tasks and needs functions for sending and receiving data via the radio interface, these functions are typically provided by the application layer or the HAL and can be wrapped by the adaptation layers. The authors of [10] show that this approach is feasible for a various set of OSs and causes a minimal runtime effort and code size impact.

We implemented adaptation layers to encapsulate our FET-MHR protocol layer and to make it portable to multiple OSs and simulators. Figure 1 shows the adaptation layers for the different environments that already exist and the integration of our FET-MHR protocol.

Castalia and the Reflex OS [11] are implemented in C++ and must be compiled by the GNU Compiler Collection (GCC). Our FET-MHR protocol and the IQlevel OS are bare C code and can be compiled by the GCC as well as by the Texas Instruments (TI) compiler. This allows us to use the TI compiler for our specific WSNP, which generates more efficient machine code. Nevertheless, our build environment supports both compilers so that our protocol can be used in all these environments without any additional implementation effort.

IV. CRITICAL TRAPS ON PORTING SOFTWARE

As just mentioned our protocol stack model and our build environment allow us to use our implementation in different environments as well as compile it with environment specific compilers. But, we learned that the compiler and architectural differences are causing a various number of pitfalls. In the following subsections we will explain the most critical traps that we found during porting our protocol from a Castalia simulation, running on an x86 architecture, to a real MSP430-based WSNP.

A. Unaligned data access

The MSP430 has a 16-bit Reduced Instruction Set Computer (RISC) architecture, where any memory access must be aligned to a 16-bit address. Nevertheless, an alignment by software is not necessary. The Micro Controller Unit (MCU) masks the lowest address bit at any memory access so that an unaligned access never traps an error. For accessing 8-bit memory addresses the compiler uses special instructions, which mask the higher or lower byte of the 16-bit data word internally.

The x86 architecture handles an 8-bit memory access in a more native way. Special instructions are unnecessary and a programmer has not to take care about memory alignment. But, when porting software from x86 to the MSP430 architecture a type cast from a 8-bit data array to a larger data type can cause an unaligned memory access and due to the MSP430's architecture the processor will not trigger any runtime error. It reads at the wrong address instead.

Especially in network protocol implementations packets are initially stored in an unspecific 8-bit data array. Later, an array's subset is casted to protocol header specific structures to simplify access onto header elements. While this cast is working at any address on the x86 architecture on the MSP430 the structure must be aligned. To avoid this problem aligned protocol headers can be used only, any access must be done with 8-bit operations or the data must be copied to an aligned address.

B. Stack size

Due to the limited resources of a low power MCU the stack size is often few bytes only. Furthermore, a task implementation with isolated memory sections does not exist. The

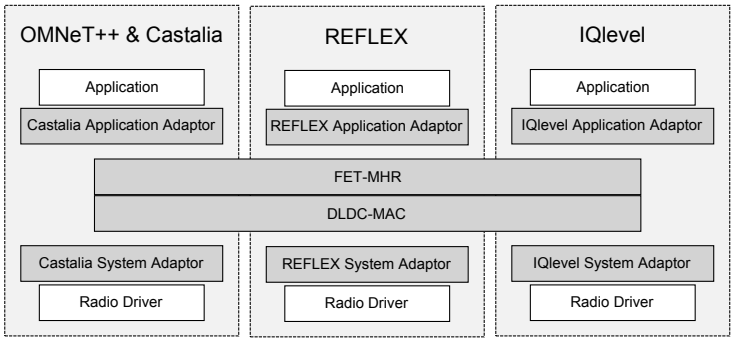


Fig. 1. Configurable protocol stacks model for using a single protocol implementation in different Operating Systems (OS)s and simulators. FET-MHR and DLDC layer are encapsulated by adaption layers to meet environment requirements and make it portable without implementation effort.

MSP430 does not have a memory protection unit to monitor memory access. A low power sensor node OS running on an MSP430 shares a single address space and stack among all its tasks.

The stack size is growing with function calls. The processor writes at least the return address and the frame pointer to the stack at each call. While these information do not consume as much memory each local variable, which can not be stored in a register, is stored on the stack too. Especially, large local objects as structures or arrays consume a lot of stack memory.

While in the Castalia framework running on a PC the stack size is quasi unlimited and local objects are very comfortable to use on a low power MCU the programmer must take special care about this. As a result a software written for the Castalia framework, which runs in a proper way, can cause a stack overflow on a low power MCU. Due to the single address space of a sensor node OS a stack overflow can overwrite any data without causing a runtime error. This makes this kind of error very hard to find.

A way to avoid this pitfall is to take special care about the usage of local variables and to forbid the implementation of uncontrolled recursions. Furthermore, barrier variables can be used to detected a possible stack overflow during runtime.

C. Interrupt handling

As mentioned in section II the OMNeT++ framework supports objects and events for passing messages to a simulation module. While objects are a good abstraction for receiving network packets the asynchronous behavior of an interrupt can not be sufficiently emulated. The delivery of an event is controlled by the simulation kernel and its execution is based on the kernel's scheduling scheme. In addition to this the OMNeT++ supports an 'unlimited' event queue. Events are not dropped in case that the programmer is not explicitly implementing this.

The MSP430 serves an interrupt immediately after finishing the current machine code instruction. Furthermore, interrupts are blocked by the MCU during a running interrupt service. By that reason an well-designed OS interrupt service is separated into a bottom and a top half section. The bottom half function runs in the interrupt context where interrupts are disabled. It does the most necessary operations only. Complex operations are done in the top half of an interrupt service, which runs in an application context. This scheme guarantees that interrupts can be served as fast as possible and the loss of interrupts can be avoided.

During the comfortable event handling scheme of the OMNeT++ framework the user is wrapped into a sloppy interrupt service implementation. Complex operations are not identified and not moved to a top half function running outside the interrupt context. On a real WSNP such a sloppy interrupt service implementation causes an unpredictable behavior and its solution requires a lot of re-implementation work, which affects the simulation results too.

D. Run-time constrains

Beside interrupt handling the run-time constrains of low power MCU are quite hard to emulated by the OMNeT++ framework. Although Castalia supports packet delivery delays the emulation of execution delays caused by the restricted computing power of a low power MCU must be explicitly implemented and can not reflect the real behavior in a sufficient manner.

Especially the low power modes of the MSP430 make an useful emulation very hard. The execution time of an interrupt is significantly influenced by the current low power mode. Furthermore, the MCU is often running with a lower clock speed and is not equipped with a floating point unit. Both make the usage of complex mathematic operation very time-consuming. These run-time constrains must be kept in mind during protocol implementation, otherwise they will influence the behavior of the protocol in a non-negligible manner.

E. Compiler bugs

As mentioned before we used the TI compiler to build the IQlevel OS for our WSNP. Although the IQlevel build environment also supports the GCC the TI compiler is preferred by us. But the Castalia framework running on the x86 architecture can not be compiled by the TI compiler. We build it with the GCC.

While porting the FET-MHR protocol to our WSNP we learned that the TI compiler generates incorrect machine code

for call by value function calls with large objects. The MSP430 uses 16-bit registers to pass arguments to the called function. For objects larger than 16-bits more than one register must be used to hold all data. Due to a bug the compiler generates the correct machine code for the function call preparation, but the generated machine code of the called functions does not use the additional registers. It uses the first register only. The additional registers are overwritten without reading the data before. Again, this bug does not generate a run-time error and the trap is very complex to solve.

Although this trap is basically a compiler bug. It shows that the usage of different compilers for the Castalia framework and the real WSNP opens the possibility to toddle into additional errors, which are very hard to tackle later.

V. Cycle Accurate Simulator (CAS)

We are still convinced that the OMNeT++ simulation framework is a powerful tool set for testing and validating new, complex network protocols. It can be used for interoperability testing and allows an efficient testing of large networks. But the program code is compiled for the simulation's host, which can differ from real WSNP in a significant manner. The traps that we found and explained in the previous section were quite hard to tackle.

But we are also convinced that for developing new network protocols for specific WSNP a simulator is necessary. In the last recent years a large set of CASs are developed. A CAS emulates the sensor node at the machine code instruction set level and uses the program code compiled for this machine type. We are confident that COOJA, simulator for the Contiki sensor node operating system, should be used for interoperability tests [12]. It is based on the MSPsim, a Java-based CAS of the MSP430 MCU [13]. It provides both a realistic simulation with accurate timing and good debugging. While using a CAS the program code must be compiled for the target hardware and the simulation covers all hardware specific constrains. Although this kind of simulation is less efficient than the OMNeT++ framework, we are convinced that the development effort is even less.

In [14] we describe an extension of the MSPsim by using a SystemC interface. The presented hybrid simulator can be used for a cycle accurate simulation in combination with testing new hardware components written in SystemC. It is planned to integrated the hybrid simulator in COOJA, so that the simulator can cover simulation from hardware up to interoperability test in an efficient manner.

VI. Conclusion

We presented our approach of a configurable protocol model for sensor node operating systems, which aims to reduce the effort for porting from one OS to another. Furthermore, we introduced our FET-MHR that we have implemented in the configurable protocol model. We used the Castalia framework for testing and validating the protocol. Due to our configurable protocol model approach we were convinced that we can use the same implementation in the Castalia and in our real WSNP. But we had to learn that by using the Castalia framework we were wrapped into a sloppy protocol implementation, which was very hard to port to our specific WSNP. In this short progress report we presented five major traps that we found during our porting work. We described the trap's background and explained why these are very hard to find and solve. In the last section we mentioned an alternative approach based on a CAS, which supports testing of compiled native code. We are convinced that this approach can help to avoid these kind of errors and helps to expose problems earlier.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the Federal Ministry of Education and Research (BMBF) under grant agreement No. 16 BN1110.

References

[1] 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.

[2] IHP, "IQlevel: Innovative high Quality level meter," 2010. [Online]. Available: http://www.ihp-microelectronics.com/en/research/wireless-systems-and-applications/projects/iq-level.html

[3] ——, "Aeternitas: Energieeffizientes Wakeup-System für drahtlose Sensorknoten," 2012. [Online]. Available: http://www.aet-projekt.de/partner_IHP.html

[4] 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.

[5] 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.

[6] A. Varga, "The omnet++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference*, ser. ESM 2011, Prague, Czech Republic, June 2001.

[7] OMNeT++, "Network Simulation Framework." [Online]. Available: http://www.omnetpp.org/index.php

[8] A. Boulis, "Castalia: revealing pitfalls in designing distributed algorithms in WSN," in *Poster proceedings of the 5th international conference on Embedded networked sensor systems*, ser. SenSys'07, Sydney, Australia, November 2007.

[9] Castalia, "Wireless Sensor Network Simulator." [Online]. Available: http://castalia.research.nicta.com.au/index.php/en/

[10] M. Brzozowski and P. Langendörfer, "Is Cross-Platform Protocol Stack Suitable for Sensor Networks? Empirical Evaluation," in *Proceedings of the 6th Joint IFIP Wireless and Mobile Networking Conference*, ser. WMNC 2013, Dubai, UAE, April 2013.

[11] K. Walther and J. Nolte, "A Flexible Scheduling Framework for Deeply Embedded Systems," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 01*, ser. AINAW '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 784–791.

[12] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proceedings of the 1st IEEE International Workshop on Practical Issues in Building Sensor Network Applications*, ser. SenseApp, Tampa (Florida), USA, November 2006.

[13] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, T. Voigt, and N. Tsiftes, "Demo abstract: MSPsim - an extensible simulator for MSP430-equipped sensor boards," in *Proceedings of the 5th European Conference on Wireless Sensor Networks*, ser. EWSN '08, Bologna, Italy, Jan. 2008.

[14] O. Stecklina, F. Vater, T. Basmer, and E. B. und Hannes Menzel, "Hybrid Simulation Environment for Rapid MSP430 System Design Test and Validation using MSPsim and SystemC," in *Proceedings of the 14th Symposium on Design and Diagnostics of Electronic Circuits and Systems*, ser. DDECS 2011, Cottbus, Germany, April 2011.

Semantische Annotationen für das IoT

Henning Hasemann und Alexander Kröller
Institut für Betriebssysteme und Rechnerverbund, Algorithmik
Technische Universität Braunschweig
38106 Braunschweig
{h.hasemann,a.kroeller}@tu-bs.de

***Abstract*—Dieser erweiterte Abstract gibt einen Überblick über den *Wiselib RDF Provider* und den *Wiselib TupleStore*, zwei Komponenten, die es ermöglichen, eingebettete Systeme mit dem Semantic Web zu verbinden. Diese legen den Grundstein für eine nicht vorher dagewesene Flexibilität in der Anwendungsentwicklung und eine vollständige Integration von eingebetteten Systemen in das Future Internet. Unsere Software speichert semantische Daten komprimiert auf eingebetteten System und ist durch die Verwendung der Wiselib portabel auf zahlreiche Plattformen (unter anderem Contiki und TinyOS). Die semantischen Daten können im RAM oder auf einem Block Device (bspw. SD-Karten) abgelegt werden. Durch Modularisierung werden verschiedene Trade-Offs im Bezug auf Codesize, Energie und Speicherverbrauch erlaubt, die wir in dieser Arbeit kurz beleuchten werden.**

I. Einleitung

Seit einigen Jahren sind wir Zeuge des Trends, eingebettete Systeme in das existierende Internet zu integrieren; insbesondere im Endanwender-Bereich ist diese Entwicklung wahrnehmbar. Im Internet of Things (IoT) werden eingebettete Geräte mit limitierten Ressourcen zu First-Class-Citizens im Internet. Dies steht im Kontrast zu der bisherigen Rolle dieser Geräte als reine Datenlieferanden an leistungsstärkere Systeme. Dies bringt einige Herausforderungen mit sich:

Auto-Konfiguration: Komponenten im IoT sollten selbstbeschreibend und selbstkonfigurierend sein, und nicht abhängig sein von spezieller Infrastruktur oder manueller Konfiguration.

Heterogenität: Eingebettete System werden von vielen verschiedenen Herstellern für unterschiedlichste Einsatzgebiete angeboten. Dies führt zum einen zu einer großen Zahl verschiedener Prozessor- und Betriebssystem-Plattformen, aber auch zu einer Vielzahl an verschiedenen Ressourcen-Konfigurationen im Bezug auf verfügbare Prozessorleistung und Energiequellen sowie nutzbaren Speicher. Ein erfolgreiches System für das IoT muss also nicht nur hohe Portabilität aufweisen, sondern auch die Möglichkeit bereitstellen, den Ressourcen-Verbrauch der Komponenten zu balancieren.

Daten-Integration: Das IoT hat heute schon viele Anwendungsbereiche und wir erwarten, dass deren Anzahl steigen wird. Um Zukunftssicherheit zu gewährleisten, müssen Systeme Fakten auf universelle Art speichern und verarbeiten können. Eine rein Menschen-orientierte Darstellung wie im bisheringen Internet vorwiegend anzufinden ist nicht vorteilhaft. Eingebettete Systeme erzeugen in der Regel dynamische Daten (z.B. Sensordaten), die erst im Verbund mit anderen (statischen) Daten für Anwendungen sinnvolle Informationen ergeben. Somit ist eine maschinenverarbeitbare Darstellung wünschenswert.

Mit dem Standard 6LowPAN [1] existiert seit einiger Zeit die Möglichkeit, Netzwerke aus eingebetteten Systemen direkt mit dem Internet zu verbinden. 6LowPAN erlaubt dabei Autokonfiguration auf IP-Ebene. Eine weitere neue Entwicklung ist das Constrained Application Protocol (CoAP) [2], das eine Ressourcen-effiziente Alternative zu HTTP für eingebettete Systeme zur Verfügung stellt und damit RESTful Webservices auf diesen Systemen ermöglicht, so dass diese leicht an des Web angebunden werden können.

Diese Arbeit beschäftigt sich mit der Entwicklung eines portablen und modularen Systems, das semantische Daten-Integration für das IoT zur Verfügung stellt. Somit adressiert es die übrigen beiden Herausforderungen. Verschiedene Ansätze für die Darstellung von Sensor-, Meta- und Anwendungsdaten auf eingebetteten Systemen existieren. Diese Ansätze haben jedoch gemein, dass sie inhärent auf einen bestimmten Anwendungsbereich der Daten beschränkt sind. Dies macht sie ungeeignet als Basis für das IoT, das unserer Ansicht nach fortlaufend neue Anwendungsbereiche erschliessen wird.

Das *Semantic Web* [3] bezeichnet eine Entwicklung, Daten im Web in semantischer Form verfügbar zu machen. Die Kodierung von Wissen erfolgt dabei mittels des *Resource Description Framework* (RDF) [4], welches semantische Fakten in Subjekt–Prädikat–Objekt-Tripeln ausdrückt. Semantische Objekte, die in Rolle von Subjekt, Prädikat oder Objekt stehen, werden dabei über eine URI identifiziert, deren Verfolgung weitere RDF-kodierte Informationen zu dem jeweiligen Objekt liefert. Auf diese Weise ist es möglich, beliebige Sachverhalte zu beschreiben und miteinander in Verbindung zu setzen. Die Anzahl der im Semantic Web verfügbaren RDF-Dokumente in Form von *Linked Data* [5] ist in den vergangenen Jahren drastisch angewachsen und stellt eine enorme semantische Wissensbasis dar, die leicht erweiterbar ist und es ermöglicht, aus der Kombination von verteilt gelagerten Fakten Schlussfolgerungen zu ziehen und somit neues Wissen zu erlangen. Wir glauben, dass es möglich ist, das IoT und das Semantic Web sinnvoll miteinander zu verbinden und eingebettete Systeme zu schaffen, die sich selbst semantisch beschreiben und auf diese Weise die Integration in das Future Internet auf Daten-Ebene ermöglichen.

Der Rest dieses Dokuments ist wie folgt aufgeteilt: In Abschnitt II gehen wir auf verwandte Arbeiten ein, Abschnitt III erläutert den Aufbau unseres Systems, das wir in Abschnitt IV experimentell evaluieren, bevor wir in Abschnitt V zu einem Schlusswort kommen.

II. Related Work

Rao *et al.* schlagen vor, Sensor-Datenströme mit RDF zu annotieren [6], allerdings wird dabei ein zentraler Proxy vorausgesetzt, der die eigentliche Annotation vornimmt, so dass das eingebettete Gerät nicht selbstbeschreibend ist. Das Open Geospatial Consortium (OGC) schlägt einen ähnlichen Ansatz vor, bei dem SensorML [7] verwendet wird um Komponenten zu beschreiben. SensorML wird üblicherweise in XML serialisiert, weswegen auch hier meist ein Proxy für die Annotation oder Übersetzung verwendet wird, wie bei Bröring *et al.* [8] und D'Aquin *et al.* [9]. Desweiteren hat SensorML ein klar definiertes Anwendungsgebiet und ist somit nicht universell für zukünftige Anwendungen einsetzbar.

Zahlreiche Datenbanken für eingebettete Systeme wurden in letzter Zeit vorgestellt: Für den Anwendungsfall von Delay Tolerant Networks (DTN) haben Sadler und Martonosi eine spezielle Datenbank entwickelt [10]. Die Systeme TeenyLime [11], [12] und Agilla [13] stellen verteile Tuple-Store-Systeme zur Verfügung. Tsiftes *et al.* präsentieren eine kompakte relationale Datenbank für eingebettete Systeme [14]. Keines dieser Systeme geht dabei allerdings speziell auf das Speichern von RDF ein. Aufgrund der hohen Redundanz dieser Daten, ist eine Betrachtung der Kompressionsmöglichkeiten jedoch unabdingbar.

Fernández und Martínez-Prieto stellen mit HDT [15] eine fortschrittliche Kompressionsmethode für RDF-Daten zur Verfügung. Diese ist allerdings auf die Serialisierung von RDF-Daten ausgelegt und erlaubt es nicht, die enthaltenen Tupel kosteneffizient zu aktualisieren. Su und Riekki [16] schlagen einen gänzlich anderen Ansatz vor, bei dem die eingebetten Geräte ein Template einer RDF-Beschreibung und die einzufügendenen Rohdaten (wie z.B. Sensorwerte) im Speicher halten. Dieser Ansatz ist zwar speicher-effizient, verhindert aber eine echte Universalität in den Beschreibungen und damit eine zukunftssichere Daten-Integration.

III. Architektur

A. Heterogenität

Um ein System entwickeln zu können, das auf verschiedener Hardware verfügbar ist, ist es notwendig, auf fortgeschrittene Features wie dynamische Speicherverwaltung, Exception Handling, Run-Time Type Identification und Virtual Inheritance zu verzichten, da diese nicht auf allen Plattformen verfügbar sind. Desweiteren ist die Speicherplatz für Programmcode in der Regel limitiert, was Laufzeit-Abstraktionen unattraktiv macht.

Die Wiselib [17] ist eine Algorithmenbibliothek für eingebettete Systeme, die sich dieser Problemstellung annimmt. Durch die Verwendung von C++-Templates stellt sie — ähnlich Boost oder der C++ Standard Template Library (STL) — einen potenten Mechanismus zur Verfügung, um Modularität zur Übersetzungszeit abzuhandeln und vermeidet so zusätzlichen Code für Laufzeit-Abstraktionen. Die Wiselib stellt, neben Hardware-Abstraktion und einer Fülle von Algorithmen aus verschiedenen Kategorien, eine Sammlung von Container-Datenstrukturen analog zu der STL bereit.

Der Wiselib RDF Provider ist implementiert als eine Reihe von Wiselib-Komponenten, die zur Übersetzungzeit austauschbar sind und so verschiedene Ressourcen-Tradeoffs erlauben. Betriebssystem-Details wie das Versenden von Nachrichten werden von der Wiselib gekapselt. Der Wiselib RDF Provider ist dadurch für eine Vielzahl von Plattformen verfügbar, darunter TinyOS [18] und Contiki [19].

B. Dokumentstrukturen

Es liegt zunächst die Vermutung nahe, dass das eingebettete System vollständig und sinnvoll mit einem einzelnen RDF-Dokument beschrieben werden kann. Obwohl dies ohne weiteres möglich ist, stellt sich die Frage, ob ein Client bei jeder Anfrage die gesamte semantische Beschreibung benötigt. Es wäre zum Beispiel annehmbar, dass große Teile der Beschreibung statisch sind, während andere, wie aktuelle Sensorwerte, sich häufig ändern. Im Interesse der Energieeffizienz erlaubt der Wiselib RDF Provider daher, mehrere sich überlappende RDF-Dokumente zu verwalten. So ist es zum Beispiel möglich, neben einem Dokument mit einer kompletten Beschreibung des Systems ein Teildokument anzufordern, welches nur aktuelle Sensorwerte enthält, weil davon ausgegangen werden kann, dass sich Metadaten wie die Maßeinheit oder andere Annotationen seit der letzten Messung nicht geändert haben.

C. Aufbau

Das von uns vorgestellte System besteht aus einer Anzahl Protokollhandler, dem RDF Service Broker und dem Wiselib TupleStore. Anfragen an semantische Dokumente von Clients können über CoAP oder andere, modular ergänzbare Protokoll-Handler an das System gesendet werden. Der RDF Service Broker schlägt im Wiselib TupleStore die entsprechenden Tripel nach, die Teil des angefragten Dokuments sind, und antwortet mit einem serialisierten RDF-Dokument. Sensordaten und Sensor-Metadaten werden in Form von RDF-Tripeln in den TupleStore eingefügt. Für Protokolle die das publish/subscribe-Pattern unterstützen, stellt der RDF Service Broker zusätzlich die Möglichkeit bereit, Protokollhandler über Änderungen an Dokumenten im TupleStore zu informieren, was zum Beispiel mittels CoAP Observe [20] dazu verwendet werden kann, Clients automatisch über Änderungenen an relevanten Teilen der semantischen Beschreibung eines Knoten zu benachrichtigen. Im Folgenden beschreiben wir die Komponenten des Systems:

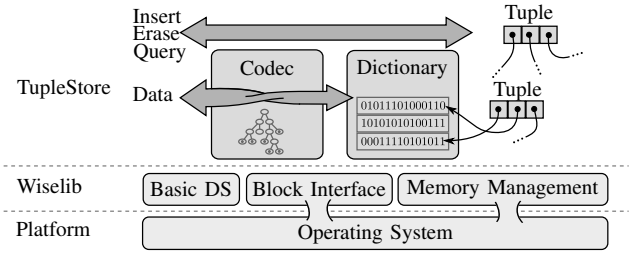


Fig. 1. Überblick über die TupleStore Architektur. TupleContainer (rechts) und Dictionary speichern die Tupel. Der (optionale) Codec stellt eine transparente Huffman-Kompression zur Verfügung.

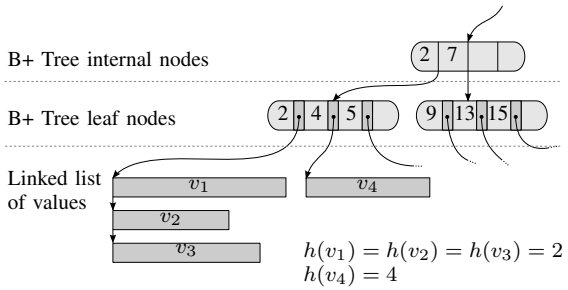


Fig. 2. HashSet-Implementierung basierend auf einem B+ Baum [23]. Baum-Schlüssel sind Hash-Werte von Tupel-Elementen (Dictionary) oder Hash-Werte über Dictionary-Key-Tupel (Tupel-Container). Kollisionen werden in einer Linked List verwaltet. Als Keys für sofortiges Abrufen von Dictionary-Einträgen werden die Speicheradressen der Einträge verwendet.

1) Wiselib TupleStore: Der Wiselib TupleStore hat die Aufgabe, semantische Tupel zu speichern. Da diese zum großen Teil aus URIs bestehen, die eine hohe Redundanz aufweisen und häufig mehrfach auftauchen, ist eine komprimierte Speicherung der Daten naheliegend. Zu diesem Zwecke besteht der TupleStore aus drei Komponenten: Tuple Container, Dictionary und Codec, wie in Abbildung 1 dargestellt.

Der Codec komprimiert alle Strings von einzufügenden Tupeln unabhängig voneinander unter Benutzung eines komprimiert abgelegten, auf typische RDF-Daten optimierten statischen Huffman-Baums [21]. Alle eingefügten (kodierten) Strings werden im Dictionary abgelegt, das Wiederholungen und andere Redundanzen zwischen mehreren Tupel-Elementen zur Speicherplatzersparnis ausbeutet. Wie auch der Codec ist das Dictionary modular austauschbar. Wir stellen zwei RAM-basierte Dictionaries zur Verfügung: Das AVL-Dictionary, welches Element in einem AVL-Baum verwaltet, und das *Prescilla Dictionary*, welches eine Variante des PATRICIA Tree [22] verwendet um gemeinsame Präfixe in Tupel-Elementen auszunutzen.

Beide Dictionary-Implementierungen verwenden Pointer auf entsprechende Baum-Knoten als Keys, mit denen die gespeicherten Strings in konstanter Zeit gefunden werden können. Der TupleContainer hält Tupel von Dictionary Keys und damit indirekt eine Menge von Tupeln von Strings. Alle Container-Typen, die die entsprechenden grundlegenden Methoden bereitstellen, können verwendet werden, darunter die in der Wiselib bereits vorhandenen `vector_static`, `vector_dynamic`, `list_static`, `list_dynamic` und `set_static`.

Zusätzlich zu diesen RAM-basierten Ansätzen stellen wir eine TupleContainer- sowie eine Dictionary-Komponente bereit, die auf einem B+ Hashset basieren, siehe Abbildung 2. TupelStore und Codec führen Kompression und Dekompression von Tupeln dabei transparent aus, so dass der RDF Provider und andere Nutzer des TupleStore von der Verwendung von Codec

und Dictionary entkoppelt sind.

2) Protokolle und Serialisierungen: Der Wiselib RDF Broker stellt zwei Interfaces zur Bearbeitung der TupleStore Daten zur Verfügung: Das *Document Level Interface* und das *Command Interface*. Über das Document Level Interface können RDF-Dokumente erzeugt, gelesen und gelöscht sowie Listener für das publish/subscribe-Pattern registriert werden. Dabei ist ein Dokument eine Menge von Tupeln im TupleStore, die vom Broker über eine zusätzlich gespeicherte Bitmaske als zusammengehörig identifiziert werden. Auf diese Weise koennen Dokumente sich überlappen. Das Command Interface auf der anderen Seite ermöglicht ressourcensparend einzelne Tupel mit Transaktionslogik einzufügen oder zu löschen.

Wir stellen eine CoAP-Implementierung zur Verfügung, die eine CoAP-Ressource mit RESTful interface pro RDF-Dokument bereitstellt. Die Wiselib CoAP Implementierung unterstützt publish/subscribe sowie Blockwise Transfer und ermöglicht damit den Transport von Dokumenten, die länger als eine MTU sind. Für Systeme, deren Ressourcen für CoAP zu beschränkt sind, stellen wir das einfache *Broker-To-Broker Protocol* vor, das das Command Interface des Clients benutzt um Dokumente des Server-Knoten entgegenzunehmen und somit auf zusätzlichen Code für die Verwaltung von Protokollzuständen verzichten kann.

RDF, welches als solches ein reines Datenmodell darstellt, kann auf verschiedene Weisen serialisiert werden. Bekannte RDF-Serialiserungen beinhalten RDF/XML, Turtle und Notation 3 (N3). Wie die meisten RDF-Serialisierungen sind diese allerdings auf Menschen-Lesbarkeit ausgerichtet und daher sehr redundante Formen der Darstellung. Eine sehr viel kompaktere Darstellung liefert HDT, welches zunächst ein Dictionary mit allen verwendeten Strings serialisiert, die gefolgt wird von einer kompakten Darstellung von Dictionary-Key-Tupeln. Für die Implementierung auf eingebetteten Systemen hat diese Darstellung jedoch den Nachteil, dass vor dem Iterieren über die Tupel eines Dokuments nicht ohne zusätzlichen (ggf. erheblichen) Speicherverbrauch ein zu kommunizierendes Dictionary zusammengestellt werden kann.

Unsere Adaption dieses Verfahrens auf eingebettete Systeme, *Streaming HDT*, kann den Datenstrom Paket für Paket erstellen. Dies wird wie folgt erreicht: Der Versender einer RDF-Dokuments legt zunächst die Tabellengröße entsprechend seines verfügbaren Speichers fest und beginnt dann über die Tupel zu iterieren. Jedes angetroffene Tupel-Element wird in der Tabelle nachgeschlagen. Falls es dort noch nicht vorhanden ist, wird es eingefügt und ein entsprechendes `insert`-Kommando als Teil des Streams zum Empfänger gesendet. Ähnlich wie bei HDT bestehen Tupel in SHDT aus Dictionary- bzw. Tabellen-Keys. Im Unterschied zu HDT wird außer für die Tabelle fester Größe und dem aktuell erzeugten Datenpaket kein Speicherplatz belegt. Da die Tabellengröße aber unabhängig von der Anzahl der zu versendenden Elemente ist, kann es notwendig sein, Tabelleneinträge zu überschreiben und `insert`-Kommandos zu wiederholen. Alternativ stellen wie eine Serialisierung zur Verfügung, die kompatibel zu Google Protocol Buffers[1] ist.

IV. Evaluation

A. Codesize

Tabelle I zeigt den Wiselib RDF Provider kompiliert für verschiedene Plattformen und veranschaulicht die möglichen Trade-Offs: Unter Verwendung des Broker-to-Broker Protokolls für Datenübertragung können wir einen TelosB-Knoten mit TinyOS mit 8kB Code mit einem kompletten RDF Store ausstatten, der Zugriff auf Tupel- oder Dokumentebene erlaubt und es somit ermöglicht, den Knoten in das Semantic Web zu integrieren. Für zusätzliche 5kB Codesize können wir Broker-to-Broker austauschen gegen CoAP in Kombination mit der komprimierenden SHDT Serialisierung und auf diese Weise die Kompatibilität auf Protokollebene drastisch erhöhen. Gleichzeitig senkt diese Änderung das transferierte Datenvolumen und damit den Energieverbrauch.

B. Tuplestore Performance

Die Auswahl verschiedener TupleStore-Komponenten ermöglicht einen Trade-Off zwischen Speicherverbrauch und Energieverbrauch. Im einfachsten Fall verwenden wir einen einfachen TupleStore ohne Dictionary oder Codec. Wie Abbildung 4 veranschaulicht, kann diese Variante allerdings eine sehr schlechte Speicherbilanz bedeuten. Aufgrund seiner geringen Code-Komplexität ist dieser Ansatz dabei einer der energieeffizientesten, siehe dazu Abbilding 3. Durch Verwendung des Prescilla-Dictionaries können wir die Kompression in der Regel drastisch erhöhen, ohne große Einbußen bei der Energieeffizienz hinnehmen zu müssen.

Im Falle von URIs mit wenig Präfix-Gemeinsamkeiten, wie in der Billion Triple Challenge vorkommen, kann das AVL Dictionary durch seinen etwas geringeren konstanten Speicher-Overhead besser Kompressionsergebnisse erzielen, siehe Abbildung 4. AVL hat aber aufgrund der komplexen Rotationsoperationen einen merklich höheren Energiebedarf. Auf Systemen, auf denen Block-Storage (z.B. eine SD-Karte) verfügbar ist, kann eine deutlich größere Menge an Daten untergebracht werden, so dass eine elementweise Kompression oftmals nicht mehr notwendig wird. Falls genug RAM verfügbar ist, kann dieser als Cache genutzt werden um Zugriffszeit und Energieverbrauch drastisch zu senken, siehe Abbildung 3.

C. Streaming HDT

Durch die variable Tabellengröße liefert SHDT einen direkten konfigurierbaren Trade-Off zwischen RAM-Verbrauch und Länge der Serialisierung, welche, z.B. beim Versenden über Funk direkte Auswirkungen auf den Energieverbrauch hat. Die Tabelle in Abbildung 4 vergleicht die Serialisierung einer RDF-Knotenbeschreibung im N-Triples-Format und der SHDT-Serialisierung des selben Dokuments für verschiedene Tabellengrößen.

V. Fazit

Wir haben gezeigt, dass es möglich ist, semantische Daten auf eingebetteten Systemen zu speichern und verarbeiten. Dank konsequenter Modularität erlaubt das System eine Balancierung von Codesize, Energie- und Speicherverbrauch. Dies ermöglicht die Anbindung von eingebetteten Systemen an das Semantic Web und stellt damit die Grundlage für die Integration des IoT mit dem Future Internet dar, welche die Verbindung von dynamischen Informationen wie Sensordaten mit einer Fülle an bereits existierenden semantischen Daten ermöglicht.

Acknowledgment

Diese Arbeit wurde teilweise unterstützt von der Europäischen Union (ICT-2009-258885, SPITFIRE).

References

[1] G. Mulligan, "The 6LoWPAN architecture," in *Proc. 4th workshop on Embedded networked sensors (EmNets '07)*. ACM, 2007, pp. 78–82.

[2] B. Frank, Z. Shelby, K. Hartke, and C. Bormann, "Constrained application protocol (CoAP)," IETF draft, Jul. 2011. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-core-coap/

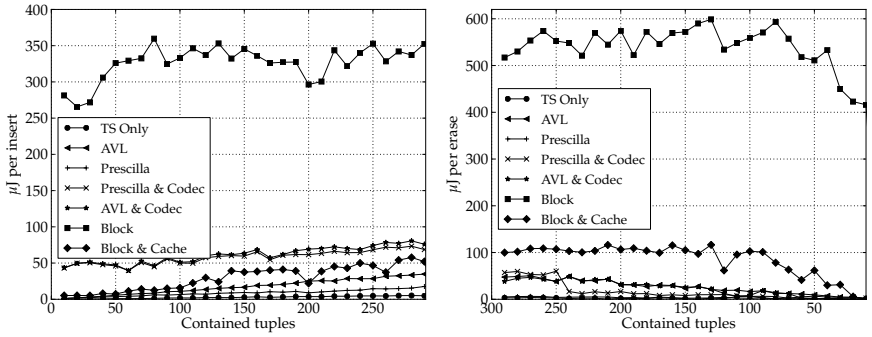
[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.

[4] G. Klyne and J. J. Carroll, "Resource Description Framework: Concepts and Abstract Syntax," W3C recommendation, W3C, Tech. Rep., Feb. 2004. [Online]. Available: http://www.w3.org/TR/rdf-concepts/

[1] https://developers.google.com/protocol-buffers/

		Protokolle		Serialisierungen	
Plattform	RDF Broker	CoAP	Broker-to-Broker	Protobuf	Streaming HDT
iSense 5139	11048 / 264	9136 / 1964	3836 / 424	3120 / 4	1792 / 40
iSense 5148	5824 / 264	6240 / 1968	2072 / 424	1616 / 8	1372 / 44
Contiki / MicaZ	8580 / 522	7796 / 1816	3666 / 343	2840 / 13	1630 / 20
TinyOS / TelosB	6146 / 134	5798 / 1840	1898 / 266	1464 / 2	970 / 20
TinyOS / MicaZ	8648 / 524	7188 / 1816	3348 / 339	2840 / 13	1588 / 20

TABLE I. CODESIZE EINZELNER KOMPONENTEN. SCHREIBWEISE: "ROM/RAM". ROM IST DIE GRÖSSE DES `.text`-SEGMENTS, RAM DIE GRÖSSE VON `.bss` UND `.data`. DAS BROKER-TO-BROKER PROTOCOL (B2B) BENÖTIGT KEINE SERIALISIERUNGSKOMPONENTE.



Konfiguration	µJ / find
TS Only	6.71
AVL	42.44
AVL & Codec	62.23
Prescilla	2.15
Prescilla & Codec	57.69
Block	203.85
Block & Cache	68.15

Fig. 3. Energieverbrauch beim Einfügung (links), Löschen (mitte) und Suchen nach Tupeln auf einem iSense 5148 Sensorknoten.

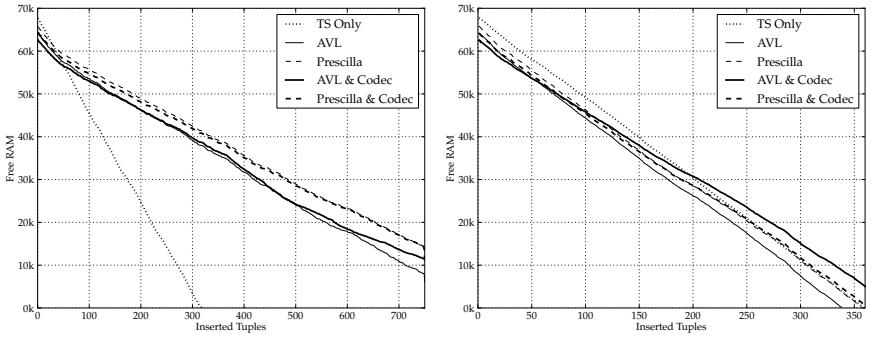


Tabelle	Serialisierung
N-Triples	
-	7686
SHDT	
3	6928
8	5682
16	4277
32	3489
64	3046
128	2851

Fig. 4. Freier Speicher bei Verwendung des TupleStores auf einem iSense 5148 Sensorknoten. Die Abbildung zeigt den freien Speicher nach Einfügen von Tupeln. Die eingefügten Tupel sind typischer Smart Service Proxy output[2] (links) bzw. zufällige Tupel aus der Billion Triple Challange[3] (mitte). Tabelle Rechts: Größe eines mit SHDT serialisierten Dokuments im Vergleich zur N-Triples-Serialisierung, abhängig von der verwendeten Tabellengröße.

[5] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 5, no. 3, pp. 1–22, 2009.

[6] K. R. Rao, T. R. Kumar, and M. N. Rao, "Architecture for automatic semantic annotation to discover knowledge from heterogeneous sensor data," *International Journal of Computer Applications in Engineering Sciences*, vol. 1, no. 2, pp. 137 –140, 2011.

[7] M. Botts, G. Percivall, C. Reed, and J. Davidson, "Sensor web enablement: Overview and high level architecture," OGC, Tech. Rep., December 2007.

[8] A. Bröring, J. Krzysztof, C. Stasch, and W. Kuhn, "Semantic challenges for sensor plug and play," in *Proc. 9th International Symposium on Web and Wireless Geographical Information Systems*, 2009, pp. 72–86.

[9] M. d'Aquin, A. Nikolov, and E. Motta, "Enabling lightweight semantic sensor networks on android devices," in *Proc. 4th International Workshop on Semantic Sensor Networks (SSN)*, 2011.

[10] C. M. Sadler and M. Martonosi, "DALi: A Communication-Centric Data Abstraction Layer for Energy-Constrained Devices in Mobile Sensor Networks," in *Proceedings of the 5th international conference on Mobile systems, applications and services (MobiSys)*. New York, New York, USA: ACM Press, Jun. 2007, p. 99. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247660.1247674

[11] P. Costa, L. Mottola, A. L. Murphy, and G. P. Picco, "Programming Wireless Sensor Networks with the TeenyLime Middleware," in *Proceedings of the International Conference on Middleware*, Nov. 2007, pp. 429–449.

[12] M. Ceriotti, M. Corra, L. D'Orazio, R. Doriguzzi, D. Facchin, S. Guna, G. P. Jesi, R. Lo Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele, "Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels," in *Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN)*, 2011, pp. 187–198.

[13] C.-L. Fok, G.-C. Roman, and C. Lu, "Agilla: A Mobile Agent Middleware for Self-Adaptive Wireless Sensor Networks," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 3, pp. 1–26, Jul. 2009.

[14] N. Tsiftes and A. Dunkels, "A database in every sensor," in *Proc. 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2011, pp. 316–332.

[15] J. D. Fernández, M. A. Martínez-Prieto, and C. Gutierrez, "Compact representation of large RDF data sets for publishing and exchange," in *Proceedings of the 9th International Semantic Web Conference (ISWC)*. Shanghai: Springer, 2010, pp. 193–208.

[16] X. Su and J. Riekki, "Bridging the Gap between Semantic Web and Networked Sensors: A Position Paper," in *Proceedings of the 3rd International Workshop on Semantic Sensor Networks (SSN)*. Shanghai: CEUR, 2010.

[17] T. Baumgartner, I. Chatzigiannakis, S. P. Fekete, C. Koninis, A. Kröller, and A. Pyrgelis, "Wiselib: A generic algorithm library for heterogeneous sensor networks," in *Proc. 7th European Conference on Wireless Sensor Networks (EWSN 2010)*, 2010, pp. 162–177.

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

[19] A. Dunkels, B. Gronvall, 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*, 2004, pp. 455–462.

[20] K. Hartke, "Observing resources in CoAP," IETF draft, 2013. [Online]. Available: http://datatracker.ietf.org/doc/draft-ietf-core-observe/

[21] D. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[22] D. Morrison, "PATRICIA – Practical Algorithm To Retrieve Information Coded in Alphanumeric," *Journal of the ACM*, vol. 15, no. 4, pp. 514–534, 1968.

[23] R. Bayer and E. M. McCreight, "Organization and maintenance of large ordered indexes," *Acta Informatica*, vol. 1, no. 3, pp. 173–189, 1972.

[2] http://spitfire-project.eu/ssp

[3] http://challenge.semanticweb.org/

Directed Link Utilization with Mahalle+

Gerry Siegemund, Volker Turau
Institute of Telematics
Hamburg University of Technology
Email: {gerry.siegemund, turau}@tu-harburg.de

Stefan Lohs, Jörg Nolte
Distributed Systems \ Operating System Group
Brandenburg University of Technology Cottbus
Email: {slohs, jon}@informatik.tu-cottbus.de

***Abstract*—Self-stabilization provides non-masking fault tolerance in distributed systems. Self-stabilizing algorithms (SSA) are defined on the assumption that either the system's topology is fixed over time or topology changes are isolated events occurring at a very low rate. These assumptions are not valid in wireless sensor networks (WSNs) where link qualities change rapidly. Therefore, neighborhood management protocols (NMP) are used to ensure the stability of the network topology for a longer time period. Furthermore, symmetrical links between nodes are more desirable than unsymmetrical ones, therefore, unidirectional links are often omitted. This paper presents an augmentation of the NMP Mahalle$^+$ to transparently utilize certain unidirectional links to increase the performance of SSAs running on top of Mahalle$^+$.**

I. Introduction

Self-stabilizing distributed systems are guaranteed to converge to a desired state or behavior in finite time, regardless of the initial state. Convergence is guaranteed, i.e., after the system is affected by transient faults of unknown scale or nature, it will return to the desired behavior. Hence, self-stabilization is a powerful approach for non-masking fault-tolerance. The actions of each individual node of a self-stabilizing system lead to a global behavior possibly not known to each entity. A node can only evaluate its local view, i.e., its own state and that of its neighbors. Alterations of the neighborhood relation may force nodes to invoke rules leading to updates of their states possibly triggering more rule executions in the entire network. Ultimately, an algorithm running on top of a vigorously changing topology will not converge to a stable state. This situation occurs in WSNs where links frequently disintegrate while others are established regularly.

The goal of a NMP is to maintain a neighborhood relation such that the resulting topology is stable and fulfills certain criteria (first of all the connectedness of the induced graph). These protocols often store histories of link quality parameters leading to a considerable memory consumption. Since memory is a limited resource in WSNs each node maintains only few neighbors, resulting in new challenges. Agility and stability are two essential abilities of neighborhood protocols. Agility ensures that the protocol adjusts quickly to new or failing nodes (or links) respectively. On the other hand, transient faults, like burst errors, need to be ignored to keep the topology stable.

Application algorithms running on top of any NMP are influenced by the performance of it. The more stable the produced topology the better the overall performance. Many SSAs use the assumption of bidirectional links, i.e., each communication channel between two nodes always works in both ways. This assumption does not hold, according to [5] unsymmetrical links can make over half of all connections in a network. It has to be noted that the definition of a unidirectional link varies greatly. One can argue that any missed message in either direction can qualify for unidirectional communication. While others argue that 50 % message loss in either way might be expectable.

Our definition is as follows, if the link quality (defined by a link quality estimator) in either direction is below a threshold (Q) and in the other direction it is above, then the connection is not symmetrical. Q may be adjusted for application needs. Section III will offer more details on this matter.

On the other hand, self-stabilizing algorithms are not restricted to symmetrical links. In [6] Masuzawa et al. developed a unidirectional maximal independent set algorithm, nevertheless claiming that this tasked proved to be a difficult one. In addition, [1] defines certain bounds for self-stabilization in unidirectional networks, mainly for the vertex coloring problem.

Mahalle$^+$ [8] naturally favors symmetric links when building its neighbor list, but as long as there is space in the neighborhood list, also unidirectional links will be recorded. (Since WSNs are not synchronized from a node's perspective all connections are unsymmetrical at one point.) An application running on top of Mahalle$^+$ might only utilize the symmetrical links.

The contribution of this paper is an augmentation of the NMP Mahalle$^+$, to transparently utilize directed links to increase the number of available symmetric links for applications running on top of Mahalle$^+$.

II. Related Work

Herman [3] introduced the first model for the usage of self-stabilizing algorithms in WSNs. A fixed topology with message loss and corruption is initiated, hence, there is no necessity for a NMP. His main contribution is the cached sensornet transformation (CST) where each node maintains a copy of the state of each neighbor (with respect to the fixed topology). Nodes periodically broadcast their state. They only execute an action when an uncorrupted message from each neighbor, since its last action, was received. Under the assumption that each message is received with a fixed

probability and that message transmissions are independent events the system will eventually reach a legitimate state with probability 1 (see also [9]).

The intuitive way to set up a NMP is to add node ids of presumable neighbors into a list. Due to the restrictions of sensor nodes (e.g., low memory) the list will mostly have a constant size. Should the list be full either no new neighbor can be added or neighbors have to be removed first. Many compared NMPs, as well as the naive implementation, will not further update the list once it is full. Woo uses the following replacement schemes to keep the list of neighbors up to date: First in first out, least-recently heard, and frequency [10].

The *Link Estimation Exchange Protocol* (LEEP) [2] from TinyOS, uses a fixed neighborhood table size of 10. Every node sends out its current knowledge about all its neighbors. The eviction policy for a full neighborhood table always replaces the node with the least quality value of LEEPs link quality estimator. Hence, in dense networks, when there exist more than 10 neighbors with a high quality value, link changes occur permanently. The quality value and LEEPs broadcasting data can be used to determine if a node is a symmetrical one or not.

None of the above mentioned protocols utilize unidirectional links and we are not familiar with any NMP which takes care of such a case. In [4, pp.106-108] a neihborhood discovery algorithm for the Unidirectional Link Triangle Routing (ULTR) protocol is introduced. The basic idea of the routing algorithm is similar to our approach while the neighborhood discovery can not be understood as a NMP.

III. Mahalle+ and Forwarded Messages

This section illustrates the neighborhood algorithm Mahalle$^+$. It forms a stable network topology on top of a physical network despite transient faults. Mahalle$^+$ uses a link quality estimator, two lists, and a set of eviction rules to gather a number of good neighbors. The basic state diagram of Mahalle$^+$ is depicted in Figure 1.

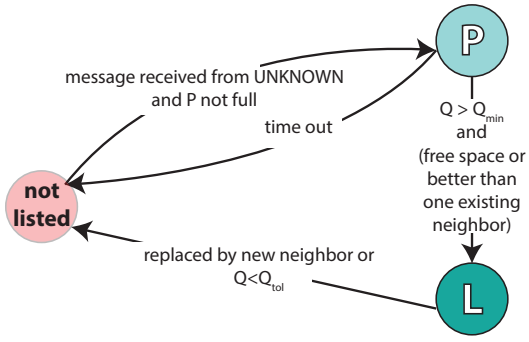


Fig. 1: Basic state diagram of the Mahalle$^+$ NMP (neighbor list L and preparation list P)

For a node to enter the preparation list P simply a message containing Mahalle$^+$ maintenance information needs to be received. Concurrently received messages stimulate the assessment of information about a neighboring node.

A	L	**A**	C		
	F	L	**A**	C	
	B	**A**	I		
	E	F	G	H	I

Fig. 2: Mahalle$^+$ example neighborhood view of a single Node A.

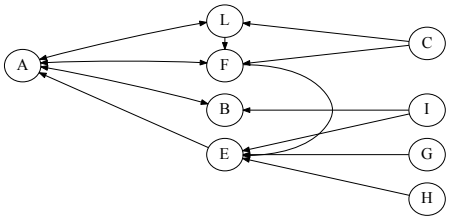


Fig. 3: Local view of a Node A

The link quality estimator values for a neighboring node need to exceed a certain threshold Q_{min} and stay above another threshold Q_{tol}, for a node to be at all considered as an actual neighbor. Any link estimator can be used, while we decided to use the very potent Holistic Packet Statistic (HoPS) [7]. HoPS produces four different estimations which all aid the channel evaluation.

If a nodes link quality exceeds Q_{min} it can be put onto the neighborlist L. L's size ($|L|$) is restricted, therefore, certain replacement policies are in place to keep the best possible neighbors [8].

The originating id of all messages travailing through the neighborhood layer is screened. Messages from unknown senders will not be passed to upper layers. Figure 2 shows a typical example Mahalle$^+$ neighborhood table of a Node A. The graphical representation of the local view of Node A can be seen in Figure 3. Note that the arrows point out the travel direction of messages.

In this example Node A and E are not symmetrical neighbors because Node E choose not to add Node A to its neighbor list, possibly due to a unidirectional communication channel. Mahalle$^+$ provides two-hop neighborhood information, therefore, Node A can logically compute that communication between Node A and Node E is possible, even with a unidirectional channel, if Node F forwards all messages from Node A.

Only slight modifications to Mahalle$^+$s implementation were necessary to find the relay node and change the message format to introduce the forwarding of messages. Firstly, a new message-type was introduced for *forwarding* and *forwarded* messages. All nodes that are supposed to forward a message are added at the end of such a Mahalle$^+$ maintenance message, including a field for the number of forwarding nodes. If the additional information does not fit into a maintenance message, then one, until now, unused flag in the message is set to

Tab. I: Mahalle$^+$ new message types

Message Type	Flag 1	Forwarding Ids
FORWARD	0	Number of and Forwarding IDs follow; containing own ID: forward message
FORWARD	1	Every node which receives this message may forward it
FORWARDED	*	Message was forwarded; has one extra field for preliminary sender ID

indicate that every node receiving this message shall forward it (Table I).

Mahalle$^+$ sends maintenance data in regular time intervals, it attaches its data either to application messages or sends out the information by itself. Forwarded messages need not to be acknowledged. They are just forwarded once by the respective node, which also handles the neighborhood data attached.

Nodes receiving relayed messages evaluate the data differently than regular messages. First, it might happen that a Node V accepts another Node F as a neighbor through relayed data, but the direct link between both nodes improves (due to some positive change in the environment). The forwarding of data is then just useless overhead. Therefore, Node F demanding the forwarding of its messages, should stop doing so. Furthermore, if messages, forwarded or not, are handled the same way, then the Node V will attach the information that Node F is a neighbor of V to the maintenance messages. If Node F receives such a message it computes that V and F are symmetric neighbors, therefore, it will stop to ask for relaying of its maintenance messages. That again will cause the *pseudo symmetric* link between both nodes to break, and causing a restart of the forwarding procedure. This would have the effect of destabilizing the whole topology, introducing a number of unwanted link changes.

Hence, a node in the neighbor list is either treated as a *direct* neighbor or as a *pseudo direct* neighbor. Nodes receiving both, direct messages of a node and forwarded messages of the same node treat them as two different nodes (direct: F_d, pseudo direct: F_p). Both nodes build up their link quality and both can be added to the preparation list (P). If F_d advances from P to L the maintenance message will include the id of F_d, hence, node F will stop asking for forwarding of its messages. That means that, the quality value of F_d will drop until the information is cleared from P. Should F_d advance to L then there will be no mention of this in the maintenance messages, i.e., the message flow of forwarded messages will continue.

The basic functionality of Mahalle$^+$ is not compromised by this approach. The eviction algorithm may choose to keep pseudo symmetrical neighbors or not. Their link quality is influenced by two connections (sender $\rightarrow$ relay $\rightarrow$ receiver), possibly resulting in increased message loss, this might make them less desirable as neighbors.

IV. Evaluation

The evaluation was accomplished using simulations with OMNeT++ and the MiXiM framework, path-loss models and log-normal shadowing are used to influence the channel behavior.

For our approach to have any effect on the system, the build topology, and the dependent application we first assessed the occurrence of such possible pseudo symmetric communication patterns. In several simulations over differed topologies (quadratic, random, or line) we evaluated the number of symmetrical and pseudo symmetrical links in the emerged topology. Figure 4 shows these findings for different neighborhood table sizes and for the following test setup: A grid topology with 50 to 200 nodes, a varying neighborhood table size $|L|$ of 5 to 10 and an average degree of 11. The bottom bars represent the pseudo symmetrical neighbors and the top bars the symmetrical once respectively. The x-axis states the number links on average over all nodes found and utilized by Mahalle$^+$.

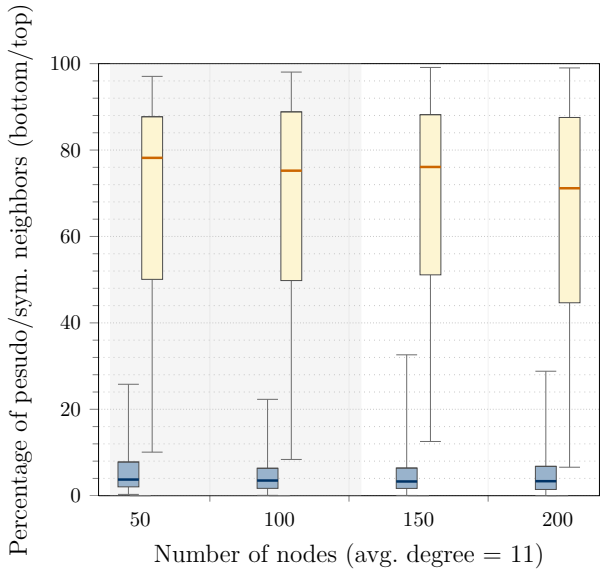


Fig. 4: Symmetric and possible pseudo symmetric neighbors; different neighborhood list sizes; without forwarding messages

We have found, that the higher the number of possible neighbors, i.e, the size of the neighbor table, the higher the number of pseudo symmetric neighbors. Even though, on average less than ten percent of all neighbors of a node are pseudo symmetrical, as can be seen in Figure 4, with 200 nodes and a maximum of 2000 edges, that still concerns roughly 100 connections. Occasionally , the number of pseudo symmetric nodes can even reach as high as forty percent.

After applying our approach of forwarding beneficial messages an increase of symmetrical neighbors, as depicted in Figure 5, can be found. With increasing degree our approach yields no increase in the average utilization of symmetric links, because the availability of symmetric links is already very high, i.e., with a degree much higher than the possible neighbors per node the number of symmetrical neighbors is sufficient.

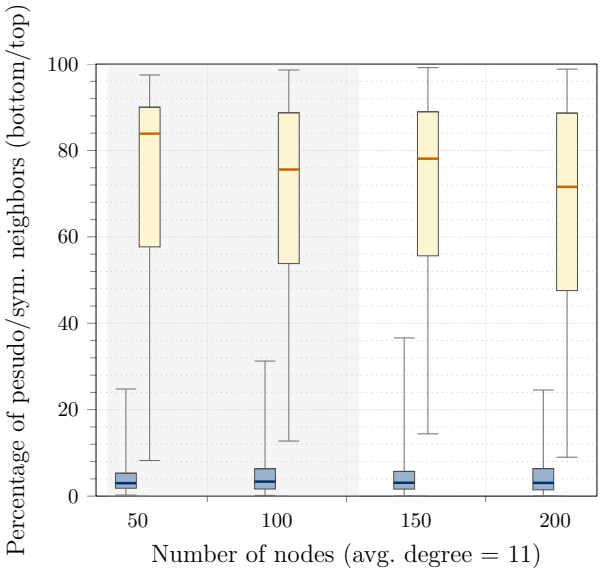


Fig. 5: Symmetric and possible pseudo symmetric neighbors; different neighborhood list sizes; with forwarding messages

Nevertheless, our approach increases the symmetrical links to be used by an application running on top of Mahalle$^+$, i.e., symmetrical plus pseudo symmetrical links. Therefore, e.g., tree construction can benefit. Short cuts trough pseudo symmetrical neighbors can shorten the overall tree depth. This, on the other hand, is very topology dependent, in simulation we found that this happens rarely, but the outcomes where never worse than with the original approach.

The presented figures only show the overall improvement of the build topology, which are quantified as a small increase of the average symmetrical links. In special cases, on the other hand, our approach might help to overcome bigger problems. Consider the node setup in Figure 6, with Mahalle$^+$ the network will not be symmetrically connected, with our augmentation it will be. (Note: The directed arrows point out the chosen neighbors, the message flow is vice versa; unlike Fig. 3).

Node 1 chooses Node 2 as a neighbor, i.e., it can receive messages from it. Node 3 chooses Node 2 and it is symmetrically connected to Node 1. Since there is no symmetrical connection between Sub-network I and II upper layer algorithms relying on these connections will not work correctly. Node 1 can compute that Node 2 did not choose it as a neighbor, possibly due to a directed channel, and that Node 3 can act as a relay Node. Again, Node 1 hears Node 2 and Node 2 hears Node 3, even though the relay node 3 does not hear Node 2, it will forward the messages from Node 1. The green arrow in the figure implicates the pseudo symmetrical connection between Node 1 and 2. In the same way also nodes with naturally few neighbors, like nodes on the outside or edge of a network, can benefit from our augmentation.

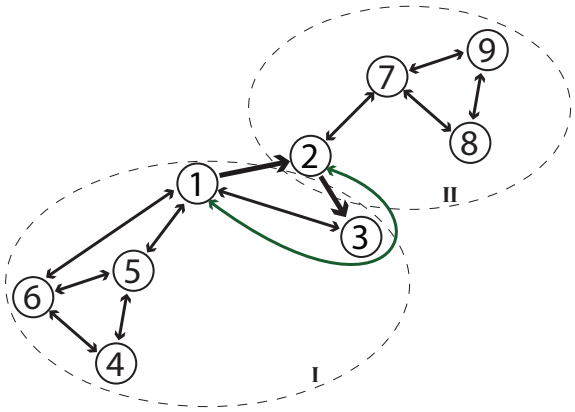


Fig. 6: Pseudo symmetric connection, Node 3 as a relay node

Conclusion

We presented an augmentation to the Mahalle$^+$ neighborhood protocol, to utilize a group of unidirectional links to increase the usefulness of Mahalle$^+$ for applications. Our approach does not interfere with the self-stabilizing abilities of Mahalle$^+$. Especially for networks with a low to medium average degree our approach is beneficial.

Acknowledgments

This research was funded by the Deutsche Forschungsgemeinschaft (DFG), contract number TU 221/6-1.

References

[1] Bernard, S., Devismes, S., Potop-Butucaru, M.G., Tixeuil, S.: Bounds for self-stabilization in unidirectional networks. arXiv preprint arXiv:0805.0851 (2008)
[2] Gnawali, O.: The link estimation exchange protocol (LEEP) (2007), TinyOS Extension Proposal (TEP)
[3] Herman, T.: Models of self-stabilization and sensor networks. In: Distributed Computing - IWDC 2003, LNCS, vol. 2918, pp. 205–214. Springer (2003)
[4] Karnapke, R.: Unidirectional Links in Wireless Sensor Networks. Ph.D. thesis, Universitätsbibliothek (2012)
[5] Lohs, S., Karnapke, R., Nolte, J.: Link stability in a wireless sensor network–an experimental study. In: Sensor Systems and Software, pp. 146–161. Springer (2012)
[6] Masuzawa, T., Tixeuil, S.: Stabilizing maximal independent set in unidirectional networks is hard. arXiv preprint arXiv:0903.3106 (2009)
[7] Renner, C., Ernst, S., Weyer, C., Turau, V.: Prediction accuracy of link-quality estimators. In: Proc. 8th Europ. Conf. on Wireless Sensor Networks (EWSN) (2011)
[8] Siegemund, G., Turau, V., Lohs, S., Karnapke, R., Nolte, J.: Agile and stable topology control for wireless sensor networks
[9] Turau, V., Weyer, C.: Fault tolerance in wireless sensor networks through self-stabilization. Int. J. of Com. Networks & Distribyuted Systems 2(1), 78–98 (2009)
[10] Woo, A.: A holistic approach to multihop routing in sensor networks. Ph.D. thesis, Berkeley, CA, USA (2004)

Sens4U: A Modular Approach Towards the Ideal Sensor Node Software and Hardware

Krzysztof Piotrowski and Jürgen Lösche
IHP
Frankfurt(Oder), Germany
{piotrowski|loesche}@ihp-microelectronics.com

***Abstract*—Low-level hardware and software development generates most of the effort in the area of specialized wireless sensor network (WSN) applications. Additionally, testing of the results is a complex task as well. Thus, especially for the hardware, most of the solutions focus on some predefined or only slightly customizable platforms. The software platform allows more flexibility and re-usability, but the solutions are usually limited to the software modules already known to the application developer. These two layers are crucial, because the reliability of the application logic developed on top of that hardware and software platform relies on these.**

The Sens4U modular approach proposes a framework that provides speed-up to the hardware and software development, for both, the human driven and the automated development. This framework allows describing the hardware and software modules, so that finding the right modules can be realized easier. The proposed abstraction is very powerful–the modules provide specific functionality via defined interfaces and can be connected using these to create the desired target configuration. And the choice between several modules providing the same interface is driven by the values of the interface parameters that describe the features of the provided functionality in more detail. The modularity improves the testability and reuse of components and thus, improves their reliability and the reliability of the generated configurations. Having functionality encapsulated in modules allows easier testing, what gives a good base for further application tests and debugging.

I. Motivation

Developing a wireless sensor network (WSN) or a cyber-physical system (CPS) application is not a trivial task. And this remains true, even if the WSN applications seem simple, i.e., they usually collect some data, process it and perform some actions based on the data or deliver the processing results to some specific nodes in the network. However, implementing the distributed application logic and choosing the right data processing algorithms, like network protocols or aggregation algorithms, requires knowledge on WSN programming and a lot of testing effort. The development process may be simplified by splitting the complete application into the individual layers and developing them separately, i.e., the hardware platform, the low-level software service layer and the application logic on top of these two layers. This solution allows also developing tailor-made and optimized hardware and software platforms for the given application or a family of applications.

To address this issue on the software level, the WSN operating systems, like TinyOS [1] or Contiki [2], usually provide kind of modularity and a library base, so that some of the required functionality do not have to be programmed from scratch. But unfortunately, the choice of modules, i.e., the converting of application needs into the right set of functions, is complicated, even if the most of the functionality is available in the library. Additionally, in those operating systems, the choice of one module from several providing the same functionality is not a trivial task.

On the hardware level there are several predefined hardware platforms, mainly designed for research purposes, like the well known Mica family [3], [4], [5] or TelosB [6], also known as TmoteSky. These provide some predefined hardware configuration and extensions to this configuration require designing additional hardware or attaching external modules to the limited set of external interfaces. A more flexible solution is to use predefined hardware modules that allow building a customizable hardware platform from several individual printed circuit boards (PCBs) providing some chosen functionality. Such solutions are provided for instance by the IHPStack [7] and by the iSense module family by coalesenses [8]. The advantage of both these approaches is that they allow building a prototyping hardware platform in a very short time. On the other hand, these platforms are rather suboptimal with respect to the node size and the flexibility is reduced to the available modules, i.e., the extensions may still require hardware development.

All the above mentioned solutions require the knowledge of the software or hardware modules as well as their interfaces before building the platforms. Our goal is to simplify the building process by allowing identifying the required interfaces of both hardware and software modules and by that locating the potential modules satisfying this first level requirement. Further, the interface parameters allow choosing the right module out of the ones providing the required interface. This allows specifying higher level requirements, e.g., memory usage estimation or a network protocol providing some specific feature. This approach can be also used to provide a generation of hardware and software platform configurations in an automatic way, e.g., providing the interfaces required by the application logic together with some set of high level requirements allows creating an ideal hardware and software sensor network platform for the given application or a family of applications.

This approach is used even further in the Sens4U project,

where we want also to apply the modularity for developing the application logic in the semi-automatic way.

The remaining part of the paper is structured as follows. The following section presents the Sens4U concept followed by the explanation of the way we exploit modularity. The paper concludes with the outlook and the future work.

II. The Sens4U Concept

This section describes the details of the approach we want to implement within the Sens4U project. First, we sketch the design flow and the roles of the users.

The main participants in the envisioned development process are the customer, the tool chain or infrastructure provider and the module developer. The **customer** is the person or organization who will compose and deploy the sensor network. The **infrastructure provider** sets the infrastructure and also chooses and manages the set of available domain-specific components and supported requirements. The **developer** provides the implementation and description of the basic building blocks–hardware and software modules.

The development process starts with the definition of requirements, done by the customer. These are chosen from the **requirements repository**, which contains a set of selectable and parameterizable requirements. In the following step–the composition–the components are selected and assembled to form a system that promises to satisfy the customer's requirements. The selection and assessment process employs the **component repository** containing models of components (modules) together with their properties and implementations. Based on the resulting configuration, the actual system is integrated–compiled and assembled. Finally, the resulting sensor nodes are programmed with the resulting code images and are deployed at the application site.

The vision is that customers only define the requirements and finally perform the physical deployment, while composition and integration are executed automatically. We want to realize this vision by applying the proposed development tools (see Figure 1) that implement the above sketched concept and are going to be realized within the Sens4U project. In Figure 1 we explicitly split the customer expertise into the application domain expertise owned by the actual **customer** and the, at least basic, WSN expertise owned by the **integrator**. The aim here is to provide the real-life applicability of the approach and thus, to introduce the integrator role to support the customer in requirement specification.

The customer explains the target application to the integrator and together they identify the most important features of the application that are further provided to the **planning tool** as input. They specify the non-technical requirements–the target deployment area coverage, the kind of communication obstacles in that area, the required sensing capabilities, the spacial placement of these measurement units, the required data flow and processing, together with the required security, reliability and maintainability features. The planning tool generates the set of technical requirements containing the required functionality and the required parameters that are further

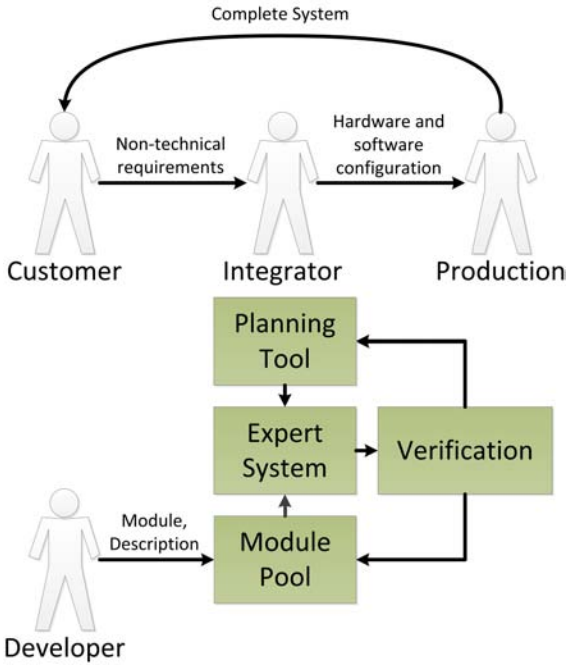


Figure 1. The detailed tool-chain-oriented development flow and user roles

forwarded to the **expert system**. Here, the hardware and software configurations are generated, based on the available modules in the **module pool** and the above mentioned extract from the technical requirements.

The configurations can be verified in simulation before sending them for production. In case of a negative verification result or in case of negative composition result the actions can be twofold, the application requirements defined using the planning tool may be relaxed or a request for developing new modules may be issued to a developer, e.g., if none of the available modules provides some functionality with the requested parameters. Of course, these actions define a trade-off between the quality and costs of the final application.

During the deployment, the sensor network can be further verified with respect to communication conditions by using the **deployment tool**. This tool analyses the actually achieved network topology and the quality of links between individual nodes in the sensor network. If the quality of links is not sufficient, additional repeater nodes may be installed to improve the connectivity. The deployment tool requires some parts of the configuration, in order to be able to communicate with the nodes and to evaluate the deployed network.

After the deployment, the network can be monitored and managed using the **management tool**. This tool allows monitoring all the parameters of each individual sensor node in the network as well as changing their parameters on-the-fly. It also allows forcing special conditions to verify the network robustness as well as allows fine tuning the already deployed nodes, e.g., to reduce their energy consumption or to increase the transmission power.

III. The Module Concept

Practically, a module in the pool is an abstraction of (an implemented) function or set of functions. And each functionality is defined by an interface that specifies the ways the functionality is to be used together with the parameters describing this functionality. Thus, a module is a self-contained building block with a coherent functionality and well defined interfaces so that it can be deployed independently and is subject to composition by third parties. Modules can represent software modules, such as functions, implemented classes, implemented algorithms, or services, but also hardware modules with diverse granularity, including hardware platforms (sensor nodes), hardware components (chips, sensors, memory extensions), or implementable hardware description modules (HDL). So, the granularity of a component can be very fine (one mathematical operation) or coarse (an entire platform). It is also allowed that modules combine other modules together and by that provide some larger set of functions. All modules are stored in the module pool.

As already mentioned, each interface has its associated parameter set that further specifies the detailed attributes of the functionality provided using this interface. These parameters are specified while defining the interface and they specify the qualitative and quantitative aspects of the functionality implemented by a given module, i.e., the values of the parameters are module related. If a module provides an interface, it has to have defined values for all the parameters associated with that interface. This allows comparing several modules that provide a given functionality to choose one that suits the requirements the best.

As part of the system assembly process, modules are aggregated, in order to combine their functionality into the target configuration. Similar to the well-known composition concepts [9], [10], modules are composed by connecting their interfaces. A module that provides a specific interface may be connected to another module that requires the same, or a compatible, interface.

The association of interfaces and parameters also helps to improve the testability of modules. The quality of the functionality a module promises can be verified before using a module in the target system. This is the task of the infrastructure provider to verify the quality of modules that are added to the module pool. Additionally, one module that uses an interface can be configured according to the values of the parameters of a second module providing the interface, before these two modules are integrated.

The result of the composition is a structural description of the system in form of a graph of components connected via interfaces. This graph is the blueprint for the system integration. An example graph with configuration options for an example application is shown in Figure 2. Here we also have shown the split between the different layers of the application on the node. Specifying the interfaces required by the application logic and a set of requirements allows generating the ideal lower layers, i.e., the software and hardware platform.

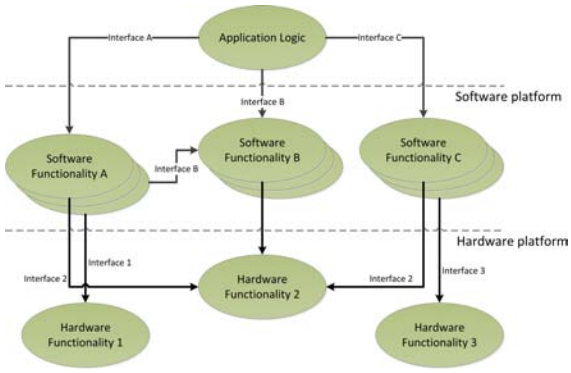


Figure 2. The graph representing the configuration options for an example application

IV. The Application Logic

The application logic is placed on top of the software and hardware platform. One of the most tricky problems in semi-automatic application development is the abstraction of the distributed application logic and the required data flow in the target application. Depending on the required WSN expertise there are several solutions, starting from implementing the application logic from scratch. More elegant solutions can be realized by some kind of macro-programming, either by some scripting means or by composition of modules available in the module pool, representing the chosen sensors and specific operations on the data they generate. In any case, the application logic defines the interface set that it requires from the lower layers and some high level requirements that define the parameters of the interfaces. Thus, it defines the requirements and the software and hardware platform can be constructed according to these. Having the requirements defined for several applications, it is possible to define the ideal software and hardware configuration that optimally satisfies all of them.

V. Conclusions

This paper presents the concept of modularity in software and hardware development for wireless sensor networks that may help in achieving the ideal or optimal platform for a given application or set of applications. This concept is developed within the Sens4U project where we research the methodology and tools to simplify and automate application development for environment monitoring. The aim of the project is to provide a working demonstrator as a proof of concept.

The concept of modularity developed in the Sens4U project and presented in this paper:

- provides a module description framework to support computer aided application development and module testing,
- allows creating application specific/optimized software and hardware configurations,
- does not only allow software compositions, but also supports hardware components with diverse granularity

(platform, chip, module) and functionality (accelerators, sensors, actuators) to fulfill the given task,

By this, our approach goes beyond pure software development. Further, this concept can be easily applied for embedded system application development in general, due to the powerful abstraction and the tool chain.

ACKNOWLEDGEMENTS

This work was partially funded by the German government under grant 03WKP26A.

REFERENCES

[1] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "Tinyos: An operating system for sensor networks," *Ambient Intelligence*, pp. 115–148, 2005.

[2] A. Dunkels, B. Grönvall, and T. 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, Nov. 2004.

[3] M. Inc., *IRIS Wireless Measurement System*, 2011, http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135:iris. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135:iris

[4] ——, *MICA2 Wireless Measurement System*, 2011, http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=147:mica2. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=147:mica2

[5] ——, *MICAz Wireless Measurement System*, 2011, http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148:micaz. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148:micaz

[6] ——, *TELOSB Mote Platform*, 2011, http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152:telosb. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152:telosb

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

[8] coalesenses GmbH, *coalesenses GmbH-Homepage*, 2013, http://www.coalesenses.com/.

[9] G. Coulson, G. Blair, P. Grace, F. Taiani, A. Joolia, K. Lee, J. Ueyama, and T. Sivaharan, "A generic component model for building systems software," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 1, p. 1, 2008.

[10] 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*, 2003.

A GNU Radio-based IEEE 802.15.4 Testbed

Bastian Bloessl, Christoph Leitner, Falko Dressler, and Christoph Sommer
Computer and Communication Systems, Institute of Computer Science, University of Innsbruck, Austria
{bloessl,christoph.leitner,dressler,sommer}@ccs-labs.org

***Abstract*—We present a Software Defined Radio (SDR) based IEEE 802.15.4 transceiver testbed for GNU Radio. Our testbed is Open Source and fully interoperable with off-the-shelf TelosB sensor motes and the Contiki sensor mote operating system, from the physical layer to the network stack. The testbed can be setup and configured easily via a graphical user interface and applications can interface with the SDR using TCP sockets. Furthermore, the SDR is able to log traffic in PCAP format to investigate networks with common software like Wireshark.**

We believe the main applications of the transceiver to be two-fold. First, the communication stack has a modular, layered structure, which allows for rapid prototyping, is educational, and is easy to grasp, lowering the steep learning curve that SDRs typically have. Secondly, the integration of a network stack in GNU Radio pushes interoperability from the physical to the network and application layer and thus enables the investigations of higher layer metrics with SDRs.

I. Introduction

Due to their multitude of possible applications, Wireless Sensor Networks (WSNs) are still an active research topic. The application scenarios for these networks range from medical monitoring for drug dispense, environmental monitoring for precision agriculture, over industrial real-time monitoring of production systems, to event detection systems like burglar alarms. Since its release in 2003, IEEE 802.15.4 [1] emerged as the de-facto standard for these networks, enabling higher layer standards like WirelessHart, ZigBee or 6LoWPAN.

The success of the standard is also visible in the fact that, today, there are twelve different physical layers proposed [2]. Starting with an O-QPSK physical layer with channels in the 700 MHz to 900 MHz and 2.4GHz ISM-bands, IEEE 802.15.4 networks are now even considered as secondary users on locally unoccupied parts of the spectrum like TV white space.

For that reason, system designers not only have to decide between different frequency bands, but also between completely different wireless technologies, like O-QPSK, GFSK, or OFDM, i.e., selecting from a spectrum that ranges from single to multi-carrier systems. It is therefore crucial to be able to compare and evaluate the performance of these different physical layer technologies. Furthermore, as WSNs are extremely application specific, it is important not to be limited to physical layer metrics like Signal to Noise Ratio (SNR) and packet loss, but to have the opportunity to compare application layer behavior and metrics like true goodput.

Experimentation is a valuable instrument that can complement analytic and simulative performance evaluation of wireless communication systems to considerably increase the confidence in the results.

Currently, there are two general approaches to conduct experimental research. First, one can rely on off-the-shelf sensor motes. This approach is easy to realize and, due to typically relatively cheap motes, a larger network can be investigated with moderate costs. The drawback of using real motes is, however, that the insights are limited to the information that the transceiver chip provides. Off-the-shelf sensor mote testbeds are, therefore, well-suited to investigate application layer metrics, but might lack the possibility to explain some of the effects, as advanced metrics are not accessible. For example, it can be hard to determine if outage occurred due to interference or due to noise, as ordinary transceiver chips do not provide any information about packets that could not be decoded. Another potential drawback is that off-the-shelf motes can not be used to investigate new physical layer strategies. Furthermore, for some of the proposed physical layers, there are no consumer devices available yet.

A second approach for conducting experimental research is the use of Software Defined Radios (SDRs), where signal processing is done in software instead of being hidden inside a transceiver chip. An SDR system consists of a software framework for real-time signal processing and a hardware RF frontend to send and receive the signal. With such a reprogrammable system the user has full control over all signal processing steps. The drawback of SDRs is that they add cost and complexity. SDR-based systems are often not accessible for non-experts in signal processing. Furthermore, these systems are often limited to physical layer implementations only and, thus, are not interoperable with real sensor motes on higher layers, i.e., the SDRs can not become nodes in a WSN.

We bridge this gap by providing an SDR based IEEE 802.15.4 testbed that provides a fully interoperable network stack. The testbed is implemented based on GNU Radio and implements the communication stack from the physical up to the network layer, where applications can be attached easily. As network layer, we choose the Rime stack [3]. Rime is a modular, lightweight network stack, which is part of the Contiki operating system. Contiki [4], in turn, is a state-of-the-art operating system for research in WSNs. With the help of the Rime stack, the SDRs can be easily integrated into a WSN, or form a heterogeneous network consisting of sensor motes and SDRs.

We make all GNU Radio code, a demo application, and a Contiki firmware available as Open Source software in the hope that it might be useful for others.[1]

[1] All source code can be downloaded from http://www.ccs-labs.org/software/.

II. Related Work

The O-QPSK physical layer of the IEEE 802.15.4 standard was first implemented by Thomas Schmid in 2006 [5]. This implementation featured separated receive and transmit chains and was verified to work with Crossbow MicaZ motes.

This implementation provided the base for several further research projects. It was used to study properties of the physical layer and the applicability of SDRs to conduct wireless research in general.

The potential of SDR based testbeds for wireless research in sensor networks was already realized by Ali et al. [6], urging the community to move from simulations to SDR based prototypes. In [7], the authors go one step further and discuss the idea of *Cognitive Radio Sensor Networks*. These networks exploit the flexibility of SDRs by applying cognitive radio strategies to sensor networks. An actual testbed for wireless research is presented in [8], where the authors prototype protocols in SDRs and on real hardware, but focus only on the MAC layer.

In all SDR implementations latency is a crucial factor, as the standard mandates strict maximum response times. For that reason, the latency of the GNU Radio implementation is studied by Thomas Schmid et al. in [9] by means of Round Trip Time (RTT) measurements.

Using SDRs connected via USB 2.0, the RTT of the SDR was an order of magnitude higher than the RTT of MicaZ motes. Furthermore, the RTT of the SDR suffered from high variances, whereas the RTT of the MicaZ motes was very deterministic. Given these results, the latency bounds that the IEEE 802.15.4 standard mandates could not be met.

The more recent USRP2 SDRs are connected via Gigabit Ethernet. This, together with the advances of GNU Radio, most notably the Vectorized Library of Kernels (VOLK) [10], should lower latency considerably. The VOLK library is part of GNU Radio and allows the use of Single Instruction Multiple Data (SIMD) operations. As SIMD instructions operate on vectors instead of scalars, the performance can be improved considerably.

In [11], the IEEE 802.15.4 implementation was further extended to support multi-channel reception with the N210 USRP SDRs. Compared to the USRP1, the N210 provides a higher bandwidth and allows to decode five adjacent channels in parallel.

III. Transceiver Architecture

Figure 1 gives an overview of the transceiver structure as exposed to GNU Radio Companion, a GUI that can be used to setup and configure signal processing flow graphs. The layered structure of the communication system can be identified clearly. In the following, we describe the individual components.

A. Physical Layer

For our tests and development of the receiver, we used USRP N210 SDRs from Ettus Research, equipped with XCVR2450 daughterboards as radio frontend. These daughterboards can operate on the 2.4 GHz ISM band in half-duplex mode.

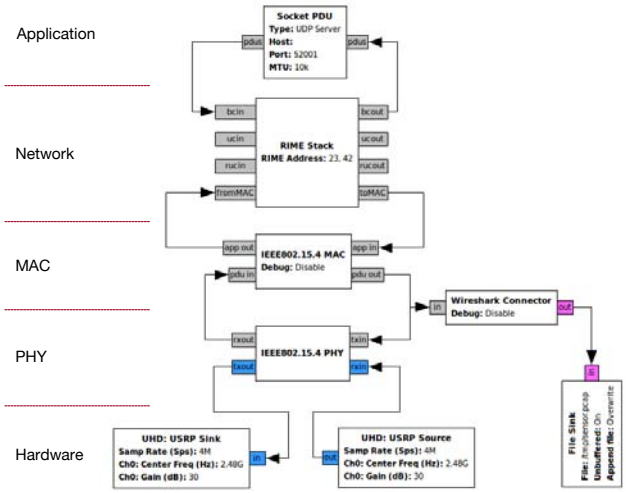
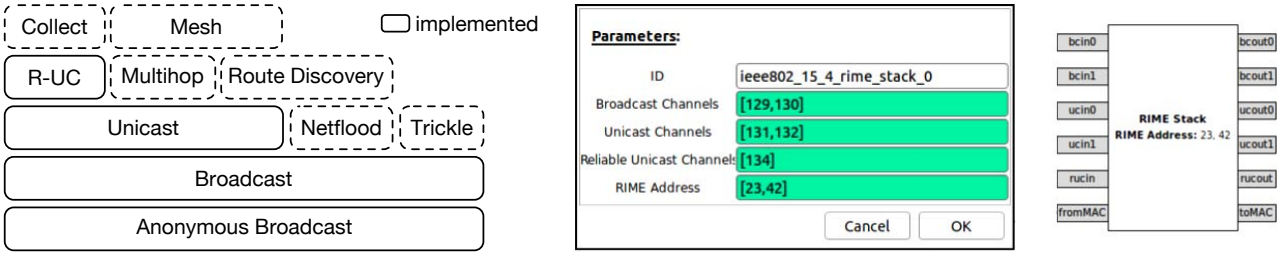


Figure 1. The modular, layered structure of the SDR transceiver as exposed to GNU Radio Companion. The physical Layer is encapsulated in a hierarchical block. The packets between the MAC and the physical layer are captured by the *Wireshark Connector* (in this case only outgoing packets to preserve clearness of the Figure).

The *USRP source* and *USRP sink* blocks in Figure 1 are interfacing this hardware. These blocks are connected to the physical layer, which is encapsulated in a hierarchical block, that hides all details of the modulation process. In GNU Radio, a hierarchical block does not implement an algorithm itself but contains another flow graph. This concept supports modularity and allows a clearer structure.

From the multitude of physical layers that are included (or, rather, are proposed to be included) in the IEEE 802.15.4 standard, currently only the O-QPSK PHY is implemented. This physical layer defines 16 channels in the 2.4 GHz band. The implementation is based on the UCLA ZigBee PHY of Thomas Schmid [5]. We extended it by porting it to version 3.7 of GNU Radio, we added GNU Radio Companion bindings in order to access the blocks in the graphical user interface, we reimplemented all python functions in C++, we changed the transmitter to GNU Radio blocks (i.e., removed custom blocks that are not needed anymore), and finally merged the separated receive and transmit chains to operate in parallel to form a transceiver system.

Creating a transceiver out of separated receive and transmit chains is not as straight forward as it might sound. Since we use half-duplex radio frontends, the USRP has to switch between send and receive mode for every packet that is sent. This happens automatically, in the sense that by default the USRP receives and switches to transmission mode if it receives samples from GNU Radio. The switch from receiving to transmitting is no problem, however the other way round is. When a packet (i.e., a burst of samples) is sent and the sample stream to the device stops, the USRP first assumes that an underflow occurred, i.e., the PC can not deliver the samples fast enough. For that reason, the device does not switch back



(a) Overview of the modular structure of the Rime stack. Each protocol makes use of its underlying primitives. Solid components have been implemented in our testbed.

(b) Configuration interface of the Rime block with ports dynamically created by the given parameter set.

(c) Resulting block instance.

Figure 2. Overview of the Rime stack and implementation in GNU Radio.

to receive mode immediately, but instead waits until a timeout occurs. The problem, however, is that when communicating with real hardware like the TelosB motes, this timeout period is too long and immediate responses like acknowledgments are missed, as the device is still in transmit mode.

To deal with that issue, we insert a tag at the start of each burst that indicates its length, i.e., the number of samples that the burst spans.[2] We added a block just before the USRP sink, which reads the length tag and, with this additional length information, is able to signal the USRP when the end of the burst is reached. Thus, the timeout before switching back to receive mode is avoided.

B. MAC Layer

As depicted in Figure 1, the MAC layer is implemented on top of the physical layer. The colors of the ports encode the type of data that is exchanged between the blocks. Gray is used to depict asynchronous messages ports, which were introduced in GNU Radio v3.6.3. Asynchronous messages allow to work packet-based, as opposed to stream-based – the mode in which most of the physical layer blocks are working in. Asynchronous messages can encapsulate arbitrary information by the use of polymorphic types. The GNU Radio developers, however, agreed on a Protocol Data Unit (PDU) format. A PDU is represented by a pair consisting of a dictionary and a character buffer. The buffer is used to store the actual data of the packet, while the dictionary can contain arbitrary metadata.

The MAC block is currently limited to the most basic functionality that enables connectivity. It encapsulates the packets from the higher layers with a valid IEEE 802.15.4 header and calculates and appends the CRC checksum. On the receive path, it does the reverse, removing the MAC header and checking whether the CRC is correct.

In particular, the MAC layer does not perform carrier sensing, but instead sends a message immediately. For that reason, the MAC layer does not yet support any other CSMA/CA functionality like backing off in time.

[2]In GNU Radio, tags can annotate specific samples of the sample stream with arbitrary information.

C. Rime Network Stack

The Rime communication stack is a lightweight network stack, designed for use in WSNs. It was implemented for Contiki by Dunkels et al. [3]. Contiki is a state-of-the-art operating system for WSNs and is, like TinyOS, heavily used and well accepted in the research community.

As depicted in Figure 2a, Rime has a modular, layered structure, where more complex connection primitives extend the underlying simpler ones to offer advanced features. The solid blocks in Figure 2a depict the connection primitives that are currently included in our SDR implementation.

The communication stack can be setup completely with a graphical user interface. Figure 2b shows the configuration interface of the Rime stack in the *GNU Radio Companion* GUI frontend. The user can open new Rime connections by adding channels numbers to the list corresponding to the desired connection type. We tweaked GNU Radio so that it is possible to dynamically create input and output message ports. This way, we can add a new pair of input and output port per connection. Figure 2c depicts the Rime block, generated by the given configuration. We see two pairs of broadcast and unicast connections (labeled bcinX, bcoutX, ucinX, and ucoutX) and a reliable unicast connection (labeled with rucin and rucout). Furthermore, we can configure the Rime address of the transceiver.

Considering unicast connections, the application has to specify the destination node on a per-packet basis, like it is the case for common UDP sockets. This is required as Rime operates connection-less. We decided to prefix the actual packet payload with the destination address of the target node in order to provide an easy to use interface to the SDR.

IV. PCAP and Wireshark Connector

To ease debugging and to allow monitoring communications in the WSN, we implemented a module that logs all transmissions in PCAP format[3], the de-facto standard format for packet dumps. PCAP is understood by all network monitoring tools like Wireshark or tcpdump. These tools also provide useful additional functionality like throughput and delay calculations.

[3]http://www.tcpdump.org/

We dump the network traffic with the *Wireshark Connector* block, which is depicted at the right hand side of Figure 1. When the block is started it writes a PCAP file header that includes global parameters like maximum packet size and the utilized technology. Every packet that is passed between MAC and physical layer is prefixed with per-packet information and logged. The per-packet header contains information about the packet size and includes a time stamp that indicates when the packet was received. The PCAP file can either be written to disk or to a Linux pipe where Wireshark can be attached to. With the help of the pipe the network can be monitored live. We utilized the Wireshark connector to get first insights into the latency of the presented transceiver. We ran the transceiver on a laptop with an Intel i5 CPU (2.6 GHz) and measured the RTTs between an SDR and a TelosB mote to be around 5 ms.

Wireshark supports ZigBee and, thus, can dissect the IEEE 802.15.4 MAC format, but not the Rime protocol headers. We therefore also implemented a dissector for Rime in LUA.

To demonstrate the capabilities of the testbed, we implemented and made available a Contiki firmware for the TelosB sensor mote platform. By default this firmware opens a broadcast connection and periodically disseminates the values of the light sensor on that connection. Furthermore, a notification is sent over that connection when the button is pressed.

It is extremely easy to connect to the flow graph with the help of the *Socket PDU* block. Connecting to the Rime connection of the flow graph and printing of the sensor values can be done with a line of shell code:

```
nc -u localhost 52001 | od -vsw2
```

As shown in Figure 3, we also provide a more visual representation of the results in a GUI that uses *matplotlib* to draw a live graph of the sensor values. Furthermore, the firmware includes a shell application where the user can connect via a serial connection over USB and dynamically open new connections and send messages on them.

V. Conclusion

We created a GNU Radio-based IEEE 802.15.4 testbed. The testbed is interoperable with real sensor motes on physical layer, and with Contiki, a state-of-the-art operating system for WSNs, on network layer. With this level of interoperability it is possible to set up a mixed network, consisting of off-the-shelf sensor motes and SDR-based transceivers. The SDR-based transceiver can be setup and configured quickly with the help of a graphical user interface. The whole communication stack, from physical to application layer, is implemented within the SDR system. Applications can be attached to the transceiver via TCP or UDP sockets.

Debugging, logging, and monitoring of the communication is aided by an option to capture all traffic in PCAP format. To further increase usability, we implemented a Wireshark dissector that parses the utilized protocols. Finally, the platform is accessible as it is Open Source.

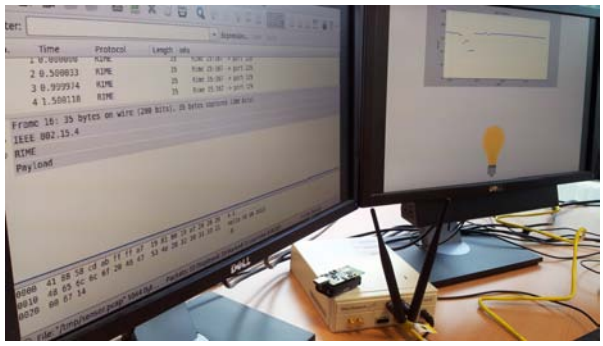


Figure 3. An example setup, consisting of a USRP N210 and a TelosB mote that periodically sends light sensor data that is displayed in the graph on the right screen. On the left screen, the whole communication is monitored in Wireshark.

This transceiver represents a proof of concept implementation that is interoperable with real sensor motes. With this as a starting point, we hope to provide a tool that facilitates rapid prototyping of new physical layers, and that allows the investigation of application layer metrics with SDRs.

References

[1] "Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE, Std 802.15.4-2011, June 2011.

[2] C.-S. Sum, L. Lu, M.-T. Zhou, F. Kojima, and H. Harada, "Design Considerations of IEEE 802.15.4m Low-Rate WPAN in TV White Space," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 74–82, April 2013.

[3] A. Dunkels, F. Österlind, and Z. He, "An Adaptive Communication Architecture for Wireless Sensor Networks," in *5th ACM Conference on Embedded Networked Sensor Systems (SenSys 2007)*. Sydney, Australia: ACM, November 2007.

[4] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th IEEE International Conference on Local Computer Networks (LCN 2004)*, Tampa, FL, November 2004, pp. 455–462.

[5] T. Schmid, "GNU Radio 802.15. 4 En-and Decoding," Networked & Embedded Systems Laboratory, UCLA, Technical Report TR-UCLA-NESL-200609-06, June 2006.

[6] M. Ali, U. Saif, A. Dunkels, T. Voigt, K. Römer, K. Langendoen, J. Polastre, and Z. A. Uzmi, "Medium Access Control Issues in Sensor Networks," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 36, no. 2, pp. 33–36, April 2006.

[7] O. B. Akan, O. B. Karli, and O. Ergul, "Cognitive Radio Sensor Networks," *IEEE Network*, vol. 23, no. 4, pp. 34–40, July 2009.

[8] X. Zhang, J. Ansari, L. M. A. Martinez, N. A. Linio, and P. Mähönen, "Enabling Rapid Prototyping of Reconfigurable MAC Protocols for Wireless Sensor Networks," in *IEEE Wireless Communications and Networking Conference (WCNC 2013)*. Shanghai, China: IEEE, April 2013, pp. 47–52.

[9] T. Schmid, O. Sekkat, and M. B. Srivastava, "An Experimental Study of Network Performance Impact of Increased Latency in Software Defined Radios," in *2nd ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and Characterization (WiNTECH'07)*. Montréal, Québec, Canada: ACM, September 2007, pp. 59–66.

[10] T. Rondeau, N. McCarthy, and T. O'Shea, "SIMD Programming in GNU Radio: Maintainable und User-Friendly Algorithm Optimization with VOLK," in *Conference on Communications Technologies and Software Defined Radio (SDR'12)*. Brussels, Belgium: Wireless Innovation Forum Europe, June 2012.

[11] L. Choong, "Multi-Channel IEEE 802.15.4 Packet Capture Using Software Defined Radio," Networked & Embedded Systems Laboratory, UCLA, Technical Report TR-UCLA-NESL-200904-01, April 2009.

Extending Wireless Body Sensor Networks Using Intelligent Implants

Thomas Basmer*, Mario Birkholz*
*IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany,
Email: basmer@ihp-microelectronics.com

***Abstract*— For monitoring vital parameters Wireless Body Area Networks (WBAN) or body sensor networks are a major research field in Wireless Sensor Networks (WSN). Extending WBAN to encompass medical implants would improve applications and develop new ones. We point out the challenges developing intelligent implants for WBANs. Wireless in-body communication, software development, security aspects, housing and energy management are discussed.**

I. Introduction

Today WBANs are monitoring vital parameters, like temperature, oxygen content and pulse, outside human bodies to monitor patients [1], firemen [2] or used for ambient assisted living. Here, transmitter and receiver are outside the human body and wireless communication is a so called on-body link. It would be beneficial to include intelligent implant devices into WBANs to optimize therapies, keep patients mobile and provide non-steady 24/7 monitoring. Patients would benefit due to better treatment while costs could be reduced at the same time. WBANs using on-body and in-body communication are introduced in [3]. The development of implants has to cope with many tasks related to bio-compatibility, wireless in-body communication (receiver or transmitter are inside human body), housing, energy management, reliability and sensor functionality.

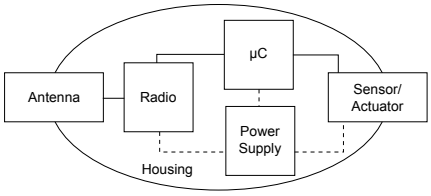


Fig. 1. Sensor node architecture used for typical WSN and implant devices.

An implant consists of comparable components like WBAN or WSN nodes (Fig. 1): Microcontroller and software, radio module for communication, sensors to detect or monitor phenomena and actuators to react in answer to them. Power supply and energy management ensure operation over the designated period. Housing protects the system against external impacts. Scope of this work are all these aspects regarding to specifics of implantable medical devices. It includes the experiences we made during GlucoPlant project [4], [5], [6] aiming at a continuous monitoring of blood sugar levels. The following sections provide an overview of implants and their requirements ranging from components over software to system aspects.

II. Implant Sensor Node

Well-known types of implanted sensor nodes are modern pacemakers or defibrillators. These devices supporting heart functionality and send therapy information to the outside world. It is also possible to configure functionality after implementation using a wireless link. Other examples are the insulin pump [7] or Pillcam [8]. All these devices communicate with a basestation only. The idea is to include implants into a WBAN to use extra information like strain, movement and other parameters to evaluate situations more precisely. Implant devices are intended to have a life cycle of at least 10 years. Meanwhile they are not physical accessible so functionality and power supply must be guaranteed. In order to not impair patients implants must be miniaturized as far as possible. Top priority for a design is that human beings must not be harmed under any circumstances.

III. Wireless Communication

WBAN and WSN applications typically use license-free Industrial, Scientific and Medical (ISM) bands for wireless communication. Popular ISM frequencies in Europe are 433 MHz, 868 MHz and 2.4 GHz. They are regulated by the European Commission in [9]. Wireless in-body communication for implant devices is regulated to minimize mutual reaction with human body (i.e. heating) and to reduce interference with other bands. For implant devices two frequency bands are approved: The Medical Implant Communication Service (MICS) and Radio Frequency Identification (RFID) presented in the next subsections. These bands are not part of ISM so a gateway is necessary between implant and WBAN nodes. In section III-C antennas in lossy medium (human body) are discussed.

A. RFID

RFID systems consist of a reader device and a so called tag device. Reader is master during communication using varying magnetic fields for transmission of power and information. A tag is called passive if it has no own power supply and is powered via magnetic induction. Otherwise it is an active device. RFID is available for frequencies from kHz up to GHz range enabling distances up to 10 m. As implant RFID tag is used for animal identification (125 kHz, 134.2 kHz), human identification (134.2 kHz), pacemaker (175 kHz) [10] or hip prosthesis (125 kHz) [11]. Low frequencies require large coils/antennas leading to large reader devices not easily applicable in WBAN nodes. Also permitted data rate and range are very small (< 0.5m) in this frequency range. In case of power consumption RFID systems could be beneficial. A passive tag does not consume limited energy resources.

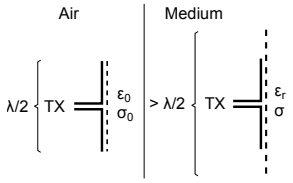


Fig. 2. Physical (solid) and electrical (dashed) antenna size (based on [18]).

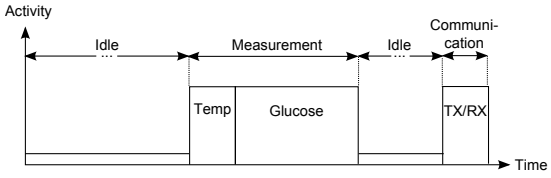


Fig. 3. Application schedule of the glucose implant device [5].

B. MICS Band

MICS band is frequently used for in-body wireless communication. It is described and regulated in [12], [13], [14]. Available frequencies are 402-405 MHz. The maximum Equivalent Isotropically Radiated Power (EIRP) on the body surface is limited to $25\mu W$ (-16 dBm). Bandwidth is separated into 25 kHz channels. Each implant may occupy up to 300 kHz. Sending range can be up to 5 m.
To our knowledge the only commercially available MICS band radio devices are offered by Microsemi (former Zarlink). ZL70321 [15], used in GlucoPlant project, is a user friendly device needing just a few external components and can be accessed via Serial Peripheral Interface (SPI). Power consumption during communication is 5 mA and in idle mode just 1 mA. A low-power mode can be accessed reducing power consumption to 10 nA. Maximum sending power is -2 dBm and receiver sensitivity is -99 dBm. Data rate is up to 800 kb/s. Modulation modes are 2-Frequency Shift Keying (FSK), 4-FSK or Gaussian-FSK (GFSK). In low-power mode the whole system can be started externally using 2.4 GHz wake-up signals. Therefore a wake-up receiver is included consuming 290 nA constantly. It polls periodically for incoming wake-up messages. If a message is received an interrupt signal is generated waking up a microcontroller from its deepest low-power mode. Only the ZL70120 [16] has a 2.4 GHz wake-up transmitter (CC2550) so it should be used for base station or gateway devices.

C. MICS Band Antennas

This subsection is limited to MICS band antennas due to the rare usage of RFID communication in WBANs and WSNs. Low frequency of MICS band ($\sim$403 MHz, wavelength λ: 0.74 m in air) requires long antennas ($\lambda/4$ wire antenna: 0.18 m) not suitable for miniaturized implants. This problem is extenuated because of the fact that electrical characteristics of enclosed implant antennas are changed by lossy media. Mainly permittivity ϵ affects antenna capacity. ϵ is 1.00 As/Vm in vacuum. Other media have higher permittivity increasing capacity and also electrical size of the antenna (Fig. 2) [17]. To have similar performance as in air/vacuum physical antenna size can be decreased.

We examined small antennas in a lossy medium for sensor node devices. In [18] antenna measurements for a sensor capsule are described. Capsule is intended for a bio-reactor filled with algae. Measurements were done at 433 MHz. For GlucoPlant project similar measurements for MICS band were done at 403 MHz in a body phantom [19]. We identified small sized antennas, i.e. a ceramic chip antenna (11 x 3 x 1 mm) [20] and a multi-turn printed loop antenna (10.2 x 10.0 x 1 mm) [21] enabling wireless communication up to a distance of 5 m.

IV. Control Unit

A. Microcontroller

Microcontrollers are well-known and widely used as processing units on wireless sensor nodes. They are usually based on a Reduced Instruction Set Cycle (RISC) architecture with bus width of 8, 16 or 32 bit. Available memory is up to 512 kB of non-volatile memory and up to 64 kB RAM. Todays microcontrollers support multitude of embedded and WSN applications coming with integrated Analog Digital Converters (ADC), timers, interfaces (SPI, UART) to connect external components (i.e. radio, sensors) and I/O ports including interrupt functionality. Low-power modes are available that are realized by switching off components. MSP430F67xx comes with eight different power modes to finely adjust power management to the application's need [22]. Usable voltage is between 1.8 V and 3.6 V. The power consumption of the device is adjustable between 265 μA/MHz in Active Mode and 0.78 μA in its deepest low-power mode (LPM4.5).

B. Software

Software for wireless sensor nodes has to run the application and to control the power management. Fig. 3 shows application schedule of GlucoPlant device [5] as an example. This scheme is typical for most WSN applications. In idle period all non-required components are set to low-power mode. Idle mode is left for measurement or communication. In our case body temperature and a so-called switching time representing glucose concentration are measured [6]. Wireless communication can be done at different points: Directly after measurement, initiated externally by WBAN node acting as base station or on fixed dates.

Due to limited memory in todays microcontrollers, the main challenge of WSN software development is to design very efficient code. In general implementation is done from scratch. Only special operating systems for WSN support the software development process. Available WSN operating systems are TinyOS [23], Contiki [24], REFLEX [25] and eCos [26] supporting engineers with main functionalities. But such an implementation from scratch is not standardized and it is hard to ensure that software is reliable in any use case. Especially

in critical applications where in worst case scenarios patients could be harmed, software development must be transferred from scratch development to standardized reliable approaches as known from industry. Model-based development, known from automotive industry [27], is one approach to overcome this. Therefore an accurate model of the target system must be available. The model enables a fast software algorithm development and structured tests. The difficulty in case of medical devices is to describe a patient and not a technical system. Also automatic code generators used in automotive industry could generate code that follows standardized rules, implementing well tested algorithms and would be less fault-prone. Tests and certification can be performed very fast. Unfortunately the drawback of these approaches is a code overhead.

V. Security

In previous WSN applications for measurement of environmental parameters (i.e. temperature, humidity, pressure) security was not a key aspect because security operations are very resource consuming. However, in WBAN applications personal data is measured and therapies are controlled. Therefore privacy aspects must be taken into account and unauthorized changes in therapies must be prohibited.
In [28] an attack against a defibrillator using wireless communication at 175 kHz is presented. An oscilloscope and an universal software radio peripheral are used to intercept communication. Oscilloscope was used to reverse engineer the wireless communication and software radio to emulate programmer outside the body. This setup enables a read out of plain data i.e. patients name, date of birth, patients history, treating physicians telephone number, implant serial ID. It was also possible to change patient's name, turn off therapy and induce fibrillations. A common way to enable privacy is encryption of measured data and communication. Common cryptographic operations are Advanced Encryption Standard (AES) or Elliptic Curve Cryptography (ECC). These algorithms can be done in software or hardware. In general special hardware accelerators are faster and more energy efficient than software implementation but less flexible. Such hardware accelerators are only seldom available in integrated circuits. The MSP430x59xx microcontrollers have AES hardware accelerators included with an encryption runtime of 168 clock cycles and a decryption runtime of 215 clock cycles for 128-AES [29]. In comparison an AES software implementation needs 5228 cycles for encryption and 6857 for decryption. At IHP hardware accelerators for 128-AES [30] and 233-ECC [31] were developed and successfully integrated into a 32-bit LEON processor [32] and 16-bit microcontroller [33]. The IHP AES needs 78 clock cycles for encryption and decryption only. A 233-ECC point multiplication in software needs 13 Mio. clock cycles, the IHP hardware accelerator only 13164 clock cycles.

Data and communication encryption is a necessity for medical WBAN applications because privacy of personal data and security of the therapy must be guaranteed.

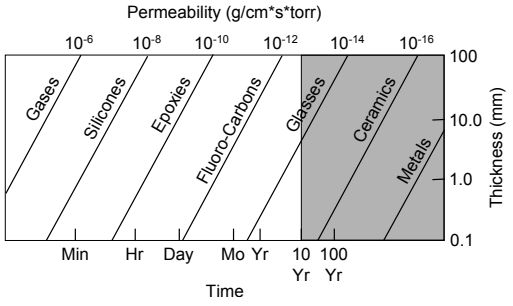


Fig. 4. Time after interior of package (different materials and thickness) reaches 50% relative humidity [39]. Grey shaded area is relevant for implants.

VI. Assembly

A. Power Supply

The power supply must provide implant lifetime of up to 10 years. First pacemakers were equipped with radioisotope thermoelectric generators using the decay of plutonium to generate energy. Todays implants use batteries as energy source. The GlucoPlant device is powered by a titanium sealed 1200 mAh Litronik pacemaker battery (31 x 28 x 5 mm) [34]. Thin-film batteries and super capacitors are not widely used in implants yet. Thin-film batteries are not commercially available and super capacitors are used in pacemakers to offer high discharge currents but not as primary power supply. Another approach is an external power supply like shown in the Fraunhofer hip prosthesis [11] where externally induced energy activates the device (RFID). Future technologies include energy harvesters generating electrical energy from physical forces. In [35] a generator using vibration caused by body movement is presented. Another idea is to use the heart beat to generate energy [36] sufficient to power a pacemaker.

B. Housing

Housing of implant devices is very challenging as influences from and to the outside have to be minimized. In Fig. 4 it can be seen that only metals and ceramics are valid housing materials for implants. Pacemakers and defibrillators usually use titanium housings. Housings made of high performance plastics like polyether ether keton (PEEK) are advancing. This material has a density similar to ceramics and metals, is easy to deform at high temperature and is resistant against chemicals. The bio-compatibility is also an important feature because body internals are a very aggressive environment attacking and dissipating every debris [37], [38].

VII. Conclusion

Implantable devices represent a valuable component of WBANs. Contemporary applications can be upgraded and new areas developed. The design of such a device is still complex and challenging. To protect patients, implants are strictly regulated. Available modules for wireless communication integrate MICS and RFID bands with limited EIRP of 25 μW. Both bands are not part of ISM bands used by most WBANs so a

gateway node is necessary. Software used for such extended WBANs must be certified but common microcontrollers have limited resources, so software is often implemented from scratch. Strategies (model-based, code generators) used in automotive industry could be beneficial. These extended WBANs process private and personal data, so security aspects must be taken into account. There is literature available coping with pacemakers sending personal data as plain. Life time of implants should be at least 10 years exceeding life time of todays WBANs. Therefore energy management strategies are crucial (i.e. use low-power components, switch off everything that is not needed). Housing is an additional challenge for such a long life time. Only few materials (metals ceramics, high performance plastics) can protect implants against aggressive environment inside human bodies.

ACKNOWLEDGMENT

The investigations presented here were performed within the development project of an implantable glucose sensor funded by the BMBF under contract number 16SV3934 (Vollimplantierbarer Glucosesensor für Diabetesdiagnostik und -therapie GlucoPlant).

REFERENCES

[1] (2013, June) Stroke back project website. [Online]. Available: http://www.strokeback.eu

[2] H. Will, T. Hillebrandt, and M. Kyas, "Wireless sensor networks in emergency scenarios: the feuerwhere deployment," in *Proceedings of the 1st ACM international workshop on Sensor-Enhanced Safety and Security in Public Spaces*, ser. SESP '12. New York, NY, USA: ACM, 2012, pp. 9–14.

[3] S. Ullah, P. Khan, N. Ullah, S. Saleem, H. Higgins, and K. S. Kwak, "A review of wireless body area networks for medical applications," *CoRR*, 2010.

[4] T. Basmer, P. Kulse, and M. Birkholz, "Systemarchitektur intelligenter Sensorimplantate," in *Biomedical Engineering / Biomedizinische Technik*, vol. 55, 2010, pp. 43–46.

[5] T. Basmer, D. Genschow, M. Fröhlich, and M. Birkholz, "Energy budget of an implantable glucose measurement system," in *Biomedical Engineering / Biomedizinische Technik*, vol. 57, 2012, pp. 259–262.

[6] M. Birkholz, K.-E. Ehwald, T. Basmer, P. Kulse, C. Reich, J. Drews, D. Genschow, U. Haak, S. Marschmeyer, E. Matthus, K. Schulz, D. Wolansky, W. Winkler, T. Guschauski, and R. Ehwald, "Sensing glucose concentrations at GHz frequencies with a fully embedded biomicro-electromechanical system (BioMEMS)," in *Journal of Applied Physics*. AIP, 2013, vol. 113, no. 24, p. 244904.

[7] E. Renard, "Implantable closed-loop glucose-sensing and insulin delivery: the future for insulin pump therapy," *Current Opinion in Pharmacology*, vol. 2, no. 6, pp. 708 – 716, 2002.

[8] (2013, June) Pillcam website. [Online]. Available: http://www.pillcam.ch

[9] *ERC Recommendation 70-03*, Conférence Europenne des Administrations des Postes et des Télécommunications Std., May 2013.

[10] M. Schuettler and T. Stieglitz, "Intelligent telemetric implants," in *Biomedical Engineering / Biomedizinische Technik*, vol. 57, 2012, pp. 967–970.

[11] "Intelligente implantate," November 2012, flyer of the Fraunhofer IPMS hip prosthesis.

[12] *The 47 CFR 95.601-95.673 Subpart E Rules applicable to the MedRadio Service*, Federal Communications Commission Std., 1999.

[13] *ETSI EN 301 839-1: Electromagnetic compatibility and Radio spectrum Matters (ERM);Radio equipment in the frequency range 402 MHz to 405 MHz for Ultra Low Power Active Medical Implants and Accessories;Part 1: Technical characteristics, including electromagnetic compatibility requirements, and test methods*, European Telecommunications Standards Institute Std.

[14] *MICS Band Plan Part 95*, Federal Communications Commission Std., january 2003.

[15] *ZL70321 Implantable Radio Module MICS RF Telemetry*, Zarlink Semiconductor Std., march 2011.

[16] "Zl70120 mics-band rf base station module (bsm)," Microsemi, Tech. Rep., February 2013.

[17] F. Merli, B. Fuchs, J. Mosig, and A. Skrivervik, "The effect of insulating layers on the performance of implanted antennas," *Antennas and Propagation, IEEE Transactions on*, vol. 59, pp. 21–31, 2011.

[18] N. Todtenberg, T. Basmer, J. Klatt, and K. Schmalz, "Estimation of 433 MHz path loss in algae culture for biosensor capsule application," in *43st European Microwave Conference (EuMC)*, 2013, accepted.

[19] T. Basmer, N. Todtenberg, F. Popiela, and M. Birkholz, "Examination of antennas for use in implant mics band applications," in *IEEE MTT-S International Microwave Workshop Series on RF and Wireless Technologies for Biomedical and Healthcare Applications*, 2013, submitted.

[20] *Chip Antennas*, Mitsubishi Materials Corperation Std., 2009.

[21] K. Yazdandoost and R. Kohno, "An antenna for medical implant communications system," in *European Microwave Conference, 2007*, october 2007, pp. 968–971.

[22] "Msp430f672x , msp430f673x mixed signal microcontroller datasheet," Texas Instruments, Tech. Rep., February 2013.

[23] (2013, June) Tinyos website. [Online]. Available: http://www.tinyos.net

[24] (2013, june) Conitki-os website. [Online]. Available: http://www.contiki-os.org

[25] (2013, June) Reflex website. [Online]. Available: http://idun.informatik.tu-cottbus.de/reflex/cgi/reflex-trac.fcgi

[26] (2013, June) ecos website. [Online]. Available: http://ecos.sourceware.org/

[27] T. Felderhoff, "Teaching model-based development methods for biomedical signal processing to sensitize and motivate for accuracy and product quality," in *Biomedical Engineering / Biomedizinische Technik*, vol. 57, 2012, pp. 962–965.

[28] D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero power defenses," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, may 2008, pp. 129–142.

[29] "Msp430fr58xx and msp430fr59xx family user's guide," Texas Instruments, Tech. Rep., January 2013.

[30] F. Vater and P. Langendörfer, "An area efficient realization of aes for wireless devices," in *IT - Information Technology*, 2007, pp. 188–193.

[31] S. Peter, "Evaluation of design alternatives for flexible elliptic curve hardware accelerators," Master's thesis, BTU Cottbus, 2006.

[32] P. Langendörferfer, F. Vater, T. Basmer, O. Stecklina, F. Gehring, and C. Wieschebrink, "Abschlussbericht - trusted sensor node," IHP and BSI, Tech. Rep., October 2011.

[33] G. Panic, T. Basmer, S. Henry, S. Peter, F. Vater, and K. Tittelbach-Helmrich, "Design of a sensor node crypto processor for ieee 802.15.4 applications," in *SoCC*, 2012, pp. 213–217.

[34] *LiS 3150 M LITRONIK Li-Manganese Dioxide Medium Rate Battery*, MST Litronik Std., february 2010.

[35] S. Platt, S. Farritor, and H. Haider, "On low-frequency electric power generation with pzt ceramics," *Mechatronics, IEEE/ASME Transactions on*, vol. 10, no. 2, pp. 240–252, 2005.

[36] M. A. Karami and D. J. Inman, "Powering pacemakers from heartbeat vibrations using linear and nonlinear energy harvesters," *Applied Physics Letters*, vol. 100, no. 4, p. 042901, 2012.

[37] M. Birkholz, K.-E. Ehwald, D. Wolansky, I. Costina, C. Baristiran-Kaynak, M. Frhlich, H. Beyer, A. Kapp, and F. Lisdat, "Corrosion-resistant metal layers from a CMOS process for bioelectronic applications," *Surface and Coatings Technology*, vol. 204, pp. 2055 – 2059, 2010, proceedings of the European Materials Research Socierty (E-MRS)Spring Meeting 2009 Symposium.

[38] M. Fröhlich, M. Birkholz, K. E. Ehwald, P. Kulse, O. Fursenko, and J. Katzer, "Biostability of an implantable glucose sensor chip," *IOP Conference Series: Materials Science and Engineering*, vol. 41, 2012.

[39] R. K. Traeger, "Nonhermeticity of polymeric lid sealants," *Parts, Hybrids, and Packaging, IEEE Transactions on*, vol. 13, no. 2, pp. 147–152, 1977.

Optimization of Point-to-Point Communication in Wireless Sensor Networks

Tsvetko Tsvetkov[1], Alexander von Bodisco[2], and Georg Carle[1]

[1]Chair for Network Architectures and Services, Technische Universität München
[2]Industrial Data Communication, University of Applied Science Kempten
Email: tsvetkov@net.in.tum.de, alexander.von.bodisco@tn-allgaeu.de, carle@net.in.tum.de

***Abstract*—Usually, Wireless Sensor Network (WSN) nodes are scattered over a large area. Their primary task is to collect data from the environment and to forward it to a base station. In order to overcome network topology changes and failures, the Routing Protocol for Low-Power and Lossy Networks (RPL) was developed and standardized by the Internet Engineering Task Force (IETF). The protocol is designed for constructing and maintaining a routing topology which is optimized for bidirectional communication between the nodes and the sink.**

There are, however, use-cases where instead of a sensor to sink traffic pattern, a sensor to sensor communication is required. For example, a node may delegate computation tasks to another node due to lack of energy. Since RPL basically requires all data packets to pass through the data sink, suboptimal routes may be selected in this kind of scenarios. Moreover, severe traffic congestion may occur near the base station. To overcome these challenges, an extension of RPL has been developed. It is called P2P-RPL, and it enables a node to discover on demand routes to other devices that do not necessarily go through the data sink.

In this paper, an assessment of RPL and P2P-RPL for TinyOS is given and an evaluation of both protocols on the MoteLab testbed is made. Furthermore, implementation specific design choices of both approaches are discussed.

I. Introduction

Over the last years WSNs have become a very important and challenging research field. Such networks consist of spatially distributed autonomous devices which usually operate untethered and additionally have limited power resources. This limits all aspects of their construction, architecture and communication capabilities. To overcome these challenges, the IETF Routing Over Low Power and Lossy networks (ROLL) Working Group [1] designed a new routing protocol, called RPL [2]. The protocol supports the latest version of the Internet Protocol which results from the research made by different organizations such as the IP for Smart Objects (IPSO) Alliance [3]. The highest goal of RPL is to provide efficient routing paths for Multipoint-to-Point (MP2P) traffic from devices inside the network towards a data sink as well as for Point-to-Multipoint (P2MP) traffic from the central control point to the remaining nodes. For this purpose, a Directed Acyclic Graph (DAG) which is rooted at the data sink is established. The DAG is actively maintained by all devices and its links are used to transport data from the sink to the network and vice versa.

However, there is a third traffic pattern that was neglected by the developers of the protocol: the Point-to-Point (P2P) connectivity between two arbitrary nodes. In case two devices need to communicate and neither of them is the sink, the transmitted data is restricted to travel only along the links in the DAG. As discussed in [4], this may result in the usage of significantly suboptimal routes and severe traffic congestion near the DAG root. Therefore, we decided to develop an extension to RPL for TinyOS. This new mechanism is commonly referred to as P2P-RPL and allows a node to reactively discover routes to one or more network participants. A node having the need to communicate with another non-sink device sends a route request message that travels through the network in a multicast manner. This message accumulates the addresses of the nodes that forward it until it reaches the desired destination. Upon reception of this message, the destination stores the accumulated route and generates a reply message containing the route. The reply message travels back to the source along the specified route. After receiving the reply, the source is able to use the learned route and transmit data packets without necessarily going along the links in the existing DAG.

This paper is organized as follows. Section II provides a detailed description of the protocols. An evaluation of both approaches is made in Section III. Our work is concluded in Section IV with a summary.

II. Protocols

A. RPL

RPL is a distance vector routing protocol for low-power and lossy networks. Network devices running the protocol are connected in such a way that no cycles are present. For this purpose a Destination Oriented Directed Acyclic Graph (DODAG), which is routed at a *single* destination, is built. The RPL specification calls this specific node a DODAG root. The graph is constructed by the use of an Objective Function (OF) which defines how the *routing metric* is computed. In other words, the OF specifies how routing constraints, certain Key Performance Indicators (KPIs) or other functions are taken into account during topology construction by every device separately.

In some cases a network has to be optimized for different application scenarios and deployments. For example, a DODAG may be constructed in a way where the Expected Number of Transmissions (ETX) or where the current amount

of battery power of a node is considered. For this reason, RPL allows to build a logical routing topology over an existing physical infrastructure. It specifies the so-called RPL Instance which defines an OF for a set of one or more DODAGs.

The protocol tries to avoid routing loops by computing a node's position relative to other nodes with respect to the DODAG root. This position is called a *Rank* and increases if nodes move away from the root and decreases when nodes move in the other direction, respectively. The calculation of the Rank can be based on a simple hop-count metric or a function which leaves more room for flexible adaptive routing.

The RPL specification defines four types of control messages for topology maintenance and information exchange. The first one is called a DODAG Information Object (DIO) and is the main source of routing control information. It may store information like the current Rank of a node, the current RPL Instance, the IPv6 address of the root, etc. The second one is called a Destination Advertisement Object (DAO). It enables the support of downward traffic and is used to propagate destination information upwards along the DODAG. The third one is named DODAG Information Solicitation (DIS) and allows a node to require DIO messages from a reachable neighbor. The fourth type is a DAO-ACK and is sent by a DAO recipient in response to a DAO message. The RPL specification defines all four types of control messages as ICMPv6 information messages with a requested type of 155. This new type has been officially confirmed by the Internet Assigned Numbers Authority (IANA) [5].

The support of downward routing is an important key feature of the protocol. By supporting P2MP traffic it is possible for a network administrator to control nodes that are not within direct communication range. This is very useful for performance evaluation purposes where usually several hundred nodes are spread over a large area. If such traffic is not supported, even the slightest changes, such as a timer value, may require to find the node, disconnect it from the network and upload a new code image. Moreover, if the idea of the Internet of Things is considered, P2MP becomes a must for WSN routing protocols [6].

The RPL specification defines two modes of operation for supporting P2MP. First, the non-storing mode which makes use of source routing. In this mode each node has to propagate its parent list up to the root. After receiving such topology information, the root computes the path to the destinations. Second, the storing mode which is fully stateful. In this mode each non-root and non-leaf network participant has to maintain a routing table for possible destinations. Note that the latter one is employed by TinyRPL, the TinyOS RPL implementation, and evaluated in this paper.

Another important fact about the protocol's design is the maintenance of the topology. Since most devices in a WSN are typically battery powered, it is crucial to limit the number of control messages transmitted over the network. Many routing protocols broadcast control packets at a fixed time interval, which causes energy to be wasted when the network is in a stable condition. Thus, RPL adapts the sending rate of DIO messages by extending the Trickle algorithm [7]. In a network with stable links the control messages will be rare which is in contrast to an environment with frequent topology changes where control messages have to be sent at a very high rate.

B. P2P-RPL

The P2P-RPL specification distinguishes between two types of nodes: an *origin* and a *target*. The discovery process begins by forming a DODAG rooted at the origin, the node that wishes to establish a P2P connection. Unlike RPL, the formed DODAG is temporal in nature and is only used to discover routes to one or more target nodes. For this reason, DIO messages are not only used for establishing the DODAG, but they also serve as route discovery messages. The specification defines them as P2P-DIO messages, and they are extended to carry additional information such as an unicast or multicast target address. Moreover, such control messages accumulate the route from the origin to the target as nodes join the DODAG. Each node determines its Rank and selects the appropriate parent node as defined by the used OF.

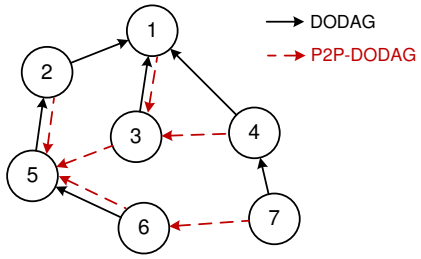
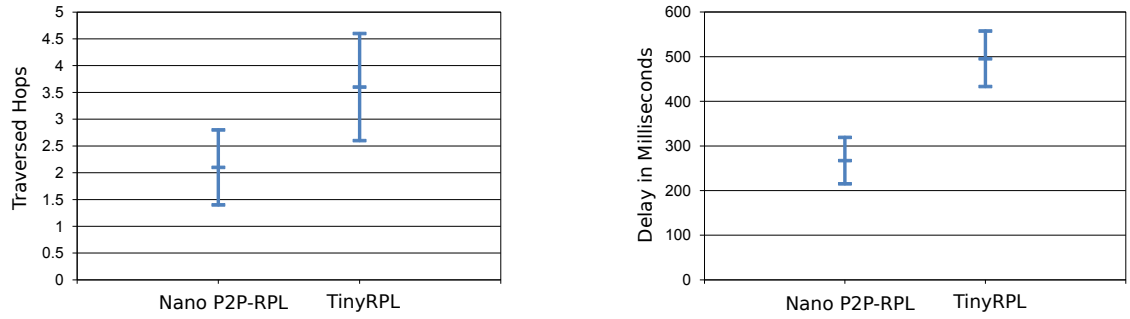


Fig. 1. RPL and P2P-RPL

Upon reception of a P2P-DIO, the target remembers the discovered route and sends it via a reply message to the origin. This reply message, which is called a Discovery Reply Object (DRO), travels back to the origin along the discovered route. The target node may additionally request the origin to acknowledge the DRO message by sending back a DRO Acknowledgment (DRO-ACK). After the route discovery process completes, both nodes are able to communicate with each other by using source routing, as shown in Figure 1.

The implementation discussed in this work is called Nano P2P-RPL and is based on the 9th revision of the P2P-RPL Internet draft published in March 2012 [4]. It supports only a single P2P-RPL instance which means that nodes are not able to participate in concurrent route discoveries, i.e. once a node joins a P2P-DODAG it will not partake in any other route discovery until the lifetime expires. Note that the term 'P2P-DODAG' represents a temporal DODAG established during route discovery. Since the RPL as well as the P2P-RPL specification use the term 'DODAG', the decision is made to define a second one in order to prevent misinterpretation. Furthermore, Nano P2P-RPL does not include an hop-by-hop route establishing mode, thus requiring an origin and a target to employ source routing after the route discovery process



(a) Average Route Length

(b) Average End-to-End Packet Delay

Fig. 2. Nano P2P-RPL Route Optimization

completes. By default, OF0 is used during the setup of the P2P-DODAG.

Since P2P-RPL operation requires bidirectional links, every Nano P2P-RPL node that receives a P2P-DIO sends the message back to the sender. However, it contains an infinite Rank and the neighbor's address as destination. The selection of a unicast address is motivated by the fact that the BLIP stack [8] does not acknowledge packets containing a multicast destination address. By selecting a unicast address, Nano P2P-RPL is able to make use of the `linkResult` event signaled by the forwarding engine and determine whether its neighbor is reachable.

III. Evaluation

The evaluation part consists of three experiments: (1) an estimation of the average route length between source and destination, (2) a computation of packet forwarding overhead of the data sink, and (3) a stability measurement of the discovered route. The first two compare TinyRPL with Nano P2P-RPL and depict the advantages of employing the route discovery mechanism. Each one of them lasts 6 hours. The last one evaluates the selection of three different P2P-DODAG lifetimes. It lasts 18 hours since the experiment is repeated three times.

During every experiment, 59 MoteLab nodes are actively participating. A new origin is selected every 30 minutes and a new target is selected every 5 minutes. In other words, as soon as a node becomes an origin, it tries to find a new target every 5 minutes. In total, 12 origins and 72 targets are chosen. They are selected according an uniform distribution over the node IDs. The data generation time interval is set to 2 seconds and the shown 99 % confidence intervals are computed around the sample mean.

A. Route Optimization

The average route lengths between source and destination are presented in Figure 2(a). While data packets sent only over the DODAG established by TinyRPL need to traverse 3.6 hops on average, the average route length is almost halved when nodes additionally employ Nano P2P-RPL. The reduction of the average route length has also an impact on the end-to-end packet delay. As shown in Figure 2(b), the average delay of packets sent over a P2P route equals to 267 ms whereas packets sent over the RPL DODAG links require 495 ms to reach the destination.

Two factors may have an negatively influence the end-to-end packet delay: hardware quality and environmental disruption. A malfunctioning antenna of a node may lead to suboptimal parent selection and delay increase. Interference from another network may lead to packet collisions and packet loss. For example, WiFi interference may lead to poor performance when the WiFi channel overlaps the one, selected by the sensor devices [9]. Unfortunately, none of these factors can be monitored and controlled over distance. MoteLab neither provides information about the current state of the hardware, nor gives details about interference and disruption from other networks.

B. Packet Forwarding Overhead

The results from this experiment show that only 648 of the generated packets traverse data sink when Nano P2P-RPL is employed. The forwarding overhead experienced by this node increases in comparison to the situation where nodes use only the DODAG links provided by TinyRPL: 2916 of the generated packets are received and forwarded by the DODAG root which leads to an increase by a factor of 4.5.

Despite the positive results, a very important remark should be made. The formed topology plays a crucial role when it comes to decreasing the data packet forwarding overhead of the data sink. In case the topology resembles an unbalanced tree, there is a low probability for P2P traffic to reach the sink even when the links established by RPL are used. For example, if the formed DODAG resembles a chain, none of the generated data packets will reach the DODAG root unless it is chosen as destination.

C. Stability of the Discovered Route

A short P2P-DODAG lifetime influences Nano P2P-RPL's ability to discover the best route to a target node. As shown in Figure 3, the usage of a source route discovered over a

DODAG with a lifetime of 5 seconds results in 1.3 route failures whereas the usage of a route discovered over a 10 second time period results in 0.9 route failures. When a P2P-DODAG lifetime of 15 seconds is selected, the reliability of the discovered route further increases: the reported route failures are 0.6. The results are computed in the following manner. Every time a new target is selected, the origin reports the number of received ICMPv6 destination unreachable messages as it tried to reach the previous target. The sample mean is computed over these values.

Furthermore, an estimation of the route changes is made. In terms of P2P-RPL, a route change occurs when an origin receives a DRO message containing a higher sequence number than the last one seen. Therefore, it will overwrite the last advertised route and use the new one instead. As indicated by the results in Figure 3, route changes start to occur more frequently as the P2P-DODAG lifetime increases. When a lifetime of 5 seconds is selected, the observed sample mean of route changes equals to 1.6. In case the lifetime is increased to 10 seconds, the sample mean equals to 2.2. It raises up to 3 when the lifetime is set to 15 seconds.

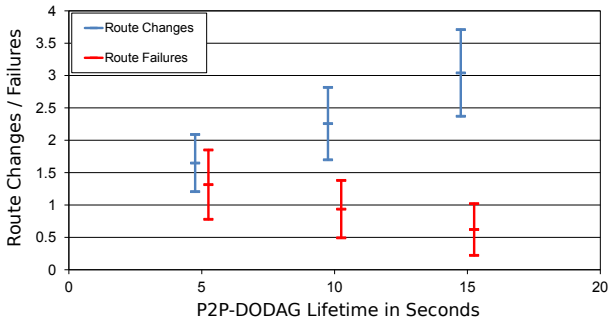


Fig. 3. Route Changes and Failures

Another aspect which is of particular interest is the packet delivery ratio. In case an intermediate node does not manage to forward a packet as specified by the source routing header, it will drop it and generate ICMPv6 destination unreachable messages to the origin and the target. These messages will be forwarded along the DODAG established by TinyRPL unless the originator of the messages has source routes to these destinations. However, such messages are also prone to loss and delay. If the origin is not informed on time that the selected route is not valid anymore, it may continue using it and packet loss may occur. During the three tests, Nano P2P-RPL proves to be highly reliable in detecting invalid routes. In case a P2P-DODAG lifetime of 5 seconds is selected the observed packet delivery ratio is 92 % and when a lifetime of 10 seconds is picked, the packet delivery ratio increases by 1 %. It further increases by 1 % when a P2P-DODAG lifetime of 15 seconds is chosen. The packet delivery ratio is computed as the ratio of the correctly received packets over all sent packets.

IV. Conclusion

In this paper, a new mechanism for establishing P2P routes in WSNs is described and evaluated. This approach, also known as P2P-RPL, establishes a DODAG rooted at the origin, the node initiating the route discovery process. The multicast P2P-DIO messages used for creating the DODAG are extended to accumulate the addresses of the nodes they traverse. The purpose of these messages is to discover routes to a target, the node at the other end point of the P2P route. Upon reception of such a message, the target remembers the accumulated route and sends it back to the origin via a reply message. This message travels along the specified route until it reaches the node which has triggered the route discovery mechanism. Unlike the RPL protocol, the established DODAG is only valid for a certain period after which nodes leave the DODAG.

P2P-RPL proves to be a reliable and efficient approach, allowing an origin and a target to exchange data packets over considerably shorter routes than the ones provided by the RPL protocol. The experiments show that the average route length is almost halved when nodes employ Nano P2P-RPL, the implementation of P2P-RPL for TinyOS. This decrease of the average route length affects the end-to-end packet delay as well. The results indicate that it is reduced by a factor of 1.85 when motes make use of the P2P-RPL mechanism. Furthermore, the packet processing overhead experienced by the data sink is significantly reduced. The experiments show that forwarding overhead of the sink is reduced by a factor of 4.5 when the links established by P2P-RPL are used.

In addition, a lifetime change of P2P-DODAG influences the ability of the origin to successfully discover the target. The results show that when the origin picks a lifetime of 5 seconds it manages to receive a reply message from only 50 % of the targets it selects. A lifetime of 15 seconds on the other side results in target discovery success rate of 90 %.

References

[1] Michael Richardson, Jean Philippe Vasseur, Adrian Farrel, and Rene Struik, *Routing Over Low Power and Lossy networks (ROLL) Working Group*, http://datatracker.ietf.org/wg/roll/, September 2012.

[2] Jean Philippe Vasseur, Navneet Agarwal, Jonathan Hui, Zach Shelby, Paul Bertrand, and Cedric Chauvenet, *RPL: IPv6 Routing Protocol for Low power and Lossy Networks*. IPSO Alliance, April 2011.

[3] Geoff Mulligan, Pete Pierre, Tim Hirou, Nick Ashworth, Anton Pfefferseder and others, *The IPSO Alliance*, September 2012.

[4] Mukul Goyal, Emmanuel Baccelli, Matthias Philipp, Anders Brandt and Jerald Martocci, *Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks (draft-ietf-roll-p2p-rpl-09)*, Internet Draft, Work in Progress, March 2012.

[5] Internet Assigned Numbers Authority (IANA), *Internet Control Message Protocol version 6 (ICMPv6) Parameters: ICMPv6 Type Numbers*, http://iana.org/assignments/icmpv6-parameters/, April 2013.

[6] Jean Philippe Vasseur, *The Internet of Things: Dream or Reality*. SENSORCOMM 2010: The Fourth International Conference on Sensor Technologies and Applications, July 2010.

[7] Philip Levis, Neil Patel, David Culler and Scott Shenker, *Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks*. In Proceedings of the USENIX NSDI Conference, San Francisco, CA, USA, March 2004.

[8] The TinyOS Community, *BLIP 2.0 - TinyOS Documentation Wiki*, http://docs.tinyos.net/tinywiki/index.php/BLIP_2.0_Tutorial, October 2011.

[9] Matteo Bertocco, Giovanni Gamba and Alessandro Sona, *Is CSMA/CA Really Efficient Against Interference in a Wireless Control System? An Experimental Answer*. In Proceedings of Emerging Technologies and Factory Automation, September 2008.

Weniger ist Mehr: Leichtgewichtige Metriken zur Erkennung von Denial-of-Service Angriffen in Drahtlosen Sensornetzen

Michael Riecker und Matthias Hollick
Secure Mobile Networking Lab
Technische Universität Darmstadt
Mornewegstr. 32, 64293 Darmstadt
{michael.riecker, matthias.hollick}@seemoo.tu-darmstadt.de

***Zusammenfassung*—Drahtlose Sensornetze (WSN) sind anfällig für Angriffe. Insbesondere Denial-of-Service (DoS) Angriffe können sich stark auf die ressourcenschwachen Knoten auswirken. In dieser Ausarbeitung identifizieren wir eine große Anzahl an Metriken, die sich leicht auslesen und berechnen lassen ohne zuviel Overhead zu erzeugen. In einem weiteren Schritt soll untersucht werden, wie sich diese Metriken im Angriffsfall verhalten und ob sie sich zur Erkennung von Angriffen eignen.**

I. Einführung

In letzter Zeit wurden drahtlose Sensornetze (WSN) zunehmend zur Lösung von unterschiedlichen, praktischen Anwendungen eingesetzt. Die Einsatzgebiete in der echten Welt, in denen WSN eine wichtige Rolle spielen, reichen von der Beobachtung von industriellen Anlagen wie beispielsweise das Erkennen von undichten Stellen und das Messen des Drucks in einem Rohr [1] bis hin zur Überwachung kritischer Infrastrukturen wie der Golden Gate Brücke [2]. Das Gewährleisten von Sicherheit in diesen beispielhaften Anwendungen stellt ein Problem dar. Eine Analyse der Literatur in diesem Bereich hat ergeben, dass Sicherheit bisher hauptsächlich von einem kryptographischen Standpunkt her betrachtet wurde, wobei der Schwerpunkt auf Vertraulichkeit und Integrität der Daten liegt. So wurden beispielsweise nicht nur sehr effiziente symmetrische Algorithmen für die ressourcenbeschränkten Knoten entwickelt, sondern auch Public-Key Kryptographie ist auf Sensorknoten einsetzbar [3], [4]. Was den Einsatz von kryptographischen Schlüsseln betrifft, so wurden mehrere Schlüsselmanagementsysteme vorgeschlagen [5], [6], [7]. Nichts desto trotz sollte aufgrund der oftmals leichten Zugänglichkeit ein Angreifer angenommen werden, der in der Lage ist die Knoten physisch zu kompromittieren und somit Zugriff auf das Schlüsselmaterial zu erhalten. Aus diesem Grunde sind Mechanismen zum Schutz vor Angriffen auf die Verfügbarkeit des WSN, wie z.B. Denial-of-Service (DoS) Angriffe, nötig. Ferner muss die Sicherheit im laufenden Betrieb gewährleistet werden.

Eine klassische Möglichkeit zur Erkennung von Angriffen stellen Intrusion Detection Systeme (IDS) dar. Die bestehenden IDS für drahtlose Sensornetze unterscheiden sich nicht nur hinsichtlich der Architektur, sondern auch in den eingesetzten Erkennungstechniken sowie den potentiell erkennbaren Angriffen. Die überwiegende Mehrheit dieser IDS arbeitet dezentral, d.h. die Knoten versuchen die Angriffe lokal zu erkennen, teilweise unter Einbeziehung der Nachbarknoten. Einige IDS können nur ein bestimmtes Fehlverhalten wie Sinkhole- oder Wormholeangriffe erkennen, während andere in der Lage sind, unterschiedliche Anomalien zu erkennen. Ein Algorithmus zur Detektion von Insider-Angriffen mit Hilfe von lokal verfügbaren Informationen wurde in [8] vorgestellt. Diese Arbeit erforscht den Einfluss räumlicher Nähe auf das Netzverhalten von Sensorknoten in direkter Nachbarschaft. Jeder Knoten überwacht dabei seine direkten Nachbarn, identifiziert Ausreißer mittels Mahalanobis-Distanzen und bestimmt durch Mehrheitswahl unter Einbeziehung der Nachbarn die endgültige Liste mit anormalen Sensorknoten. Das Netzverhalten wird charakterisiert durch die Rate an verworfenen Paketen, die Paketsenderate, die Verzögerung beim Weiterleiten und die tatsächlichen Sensormesswerte. Ein weiteres beispielhaftes IDS wurde in [9] entwickelt, welches ein Modell vom normalen Verhalten anhand einer Vielzahl von Metriken erstellt, wie z.B. die Paketkollisionsrate, der Energieverbrauch und das Messintervall.

Bisher wird die Entscheidung, welche Metriken für die Angriffserkennung relevant sind, mehr willkürlich als wissenschaftlich begründet getroffen. Um die passendsten Metriken zu identifizieren, müssen wir die tatsächlichen Effekte von Angriffen auf drahtlose Sensornetze verstehen. Wir gehen davon aus, dass bestimmte DoS-Angriffe drastische Auswirkungen auf verschiedene Metriken haben und sich diese somit im Angriffsfall signifikant von den normalen Werten unterscheiden.

II. Leichtgewichtige Metriken

Was könnten nun leichtgewichtige Metriken sein, die sich ohne viel Aufwand lokal am Knoten auslesen lassen? Wir haben mit dem Collection Tree Protokoll (CTP) ein typisches Anwendungsszenario von WSN näher betrachtet und verschiedene Metriken identifiziert, die sich potentiell eignen könnten, um DoS-Angriffe zu erkennen. Neben diesen CTP-spezifischen Metriken wurden auch allgemein verwendbare

Metriken herausgearbeitet, welche zunächst vorgestellt werden:

1) *Received Signal Strength Indicator (RSSI):* Der RSSI-Wert stellt die momentane Stärke des Funksignals dar, wie es auf Empfängerseite gemessen und üblicherweise in dBm ausgedrückt wird. Ein hoher RSSI-Wert könnte auf einen Angriff deuten, bei dem ein Angreifer mit höherer Sendeleistung operiert als reguläre Sensorknoten.
2) *Sende-/Empfangszeit:* Diese Metriken geben an, wieviel Zeit das Funkmodul pro Periode im Sende-/Empfangsmodus verbringt. Jamming und Blackhole-Angriffe sollten diese Zeit beeinflussen.
3) *Gesendete/Empfangene Pakete auf Netzwerkebene:* Auf Netzwerkebene werden alle ausgehenden und eingehenden Pakete gezählt. Abhängig von der Stärke eines Jamming-Angriffs könnte sich die Anzahl der empfangenen Pakete entweder erhöhen oder ganz auf Null reduziert werden, wenn keine Kommunikation mehr möglich ist. Der Blackhole-Angriff könnte ebenfalls zu zwei unterschiedlichen Effekten führen. Einerseits könnten Knoten in direkter Nachbarschaft zum Angreifer eine erhöhte Anzahl an empfangenen Paketen verzeichnen. Andererseits könnte das Verwerfen von Paketen durch den Angreifer bei bestimmten Knoten zu einer Abnahme an empfangenen Paketen führen.
4) *Gesendete/Empfangene Pakete auf MAC-Ebene:* Diese Paketraten entsprechen den Metriken auf Netzwerkebene, zählen aber die ausgehenden und eingehenden Nachrichten auf MAC-Ebene.
5) *Pakete mit ungültiger CRC-Prüfsumme:* Ein empfangenes Paket mit ungültiger CRC-Prüfsumme wird verworfen. Diese Metrik stellt die Häufigkeit des Auftretens solcher Ereignisse dar. Falls ein Angreifer Pakete sendet, die nicht der IEEE 802.15.4 Spezifikation entsprechen, sollte sich die Anzahl an Paketen mit ungültiger CRC-Prüfsumme erhöhen. Darüber hinaus könnte höherer Netzverkehr zu Interferenzen und der Zerstörung von regulären Paketen führen.
6) *Energieverbrauch durch das Funkmodul:* Der Energieverbrauch durch Aktivitäten des Funkmoduls wie das Abhören des Kanals, das Senden von Nachrichten oder der Energieverbrauch im Bereitschaftsmodus können gemessen werden.
7) *Energieverbrauch durch den Mikrocontroller:* Diese Metrik misst den Energieverbrauch des Mikrocontrollers, der von der Betriebszeit und -spannung abhängt.
8) *Contention Rate:* Wenn ein Knoten aufgrund eines belegten Kanals kein Paket senden konnte, wird der Zähler dieser Metrik hochgesetzt.
9) *Wartende Pakete:* Diese boolesche Metrik gibt an, ob der Knoten noch nicht verarbeitete Pakete im Buffer hat.
10) *Zu kurze Pakete:* Pakete die kürzer sind als der Footer plus die Prüfsumme könnten durch einen Jamming-Angriff hervorgerufen werden.

Das Collection Tree Protokoll stellt zusätzliche statistische Daten das Routing betreffend zur Verfügung, welche als mögliche Erkennungsmetriken eingesetzt werden können:

11) *Gesendete/Empfangene Datenpakete:* Diese Metriken zählen, wieviele Datenpakete unter Einsatz des Collection Tree Protokolls gesendet/empfangen wurden.
12) *Gesendete/Empfangene Acknowledgement-Pakete:* Hier wird angegeben, wieviele Acknowledgement-Pakete gesendet und empfangen wurden.
13) *Empfangene Duplikate:* Diese Metrik beschreibt die Anzahl der empfangenen Duplikate. Angriffe, bei denen die selben Datenpakete ohne jede Veränderung wiedereingespielt werden, könnten dadurch erkannt werden.
14) *Verworfene Pakete aufgrund eines überladenen Buffers:* Sollte der Buffer für eingehende Datenpakete überladen sein, werden die nächsten ankommenden Pakete verworfen. Diese Metrik zählt, wie oft dieses Ereignis eintritt.
15) *Paketzustellrate:* Die Paketzustellrate (PDR) basiert auf den gesendeten Datenpaketen und den empfangenen Acknowledgements. Sie wird wie folgt berechnet

$$PDR_{Sender} = \frac{empfangene\_ACKs}{gesendete\_Pakete}$$

und beschreibt den Anteil der erfolgreich gesendeten Datenpakete.
16) *Wechseln des Vaterknotens:* Im Collection Tree Protokoll kommt eine baumbasierte Topologie zum Senden von Daten zum Einsatz. Falls ein Sensorknoten nicht direkt mit der Basisstation kommunizieren kann, wählt er einen Vaterknoten aus, welcher seine Daten weiterleitet. Die Anzahl der Wechsel des Vaterknotens wird durch diese Metrik gezählt. Diese Information könnte nützlich sein, um Blackhole-Angriffe zu identifizieren.
17) *Schätzung des besten Nachbarknotens:* Ein Knoten wählt denjenigen Nachbarn als Vaterknoten, der die geringsten Routingkosten aufweist. Diese Kosten werden entweder berechnet durch Schätzen der Linkqualität oder durch Auslesen der Headerinformationen von eingehenden Paketen. Diese Metrik schätzt die Linkqualität zum besten Nachbarknoten und kann dadurch evtl. generelle Netzwerkanomalien wie Sinkhole-Angriffe erkennen.

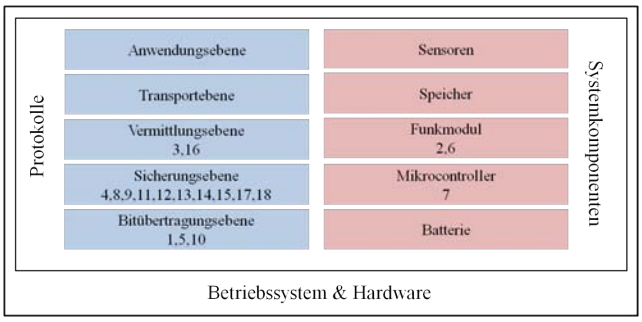


Abbildung 1. Architektur eines Sensorknotens. Die Zahlen geben an, welche Metrik auf welcher Ebene / welchem Modul ausgelesen wird.

18) *Anzahl der Nachbarknoten:* Dieser Wert gibt die Anzahl der erreichbaren Knoten in der Nachbarschaft an.

Abbildung 1 zeigt die Architektur eines typischen Sensorknotens und gibt an, an welcher Stelle die Metriken ausgelesen werden bzw. welche Ebenen betroffen sind. Alle genannten Metriken lassen sich mit Hilfe von Standard Contiki- / CTP-Funktionen entweder direkt auslesen oder sind leicht anhand der ermittelten Werte berechenbar.

III. Verwandte Arbeiten

Die systematische Analyse der Auswirkungen von Denial-of-Service Angriffen auf drahtlose Sensornetze wurde bisher vernachlässigt.

Xu et al. [10] untersuchten mehrere Erkennungsvarianten für Jamming-Angriffe. Die Autoren verwendeten drei verschiedene Metriken, um zwischen normalem und durch Jamming beeinträchtigtem Verkehr zu unterscheiden: (1) den durchschnittlichen RSSI-Wert, (2) die Abhörzeit des Trägers, und (3) die Paketzustellrate (PDR). Sie zogen den Schluss, dass die Kombination aus PDR und RSSI zuverlässig Jamming-Angriffe erkennen kann. Die selben Autoren erweiterten ihre Arbeit zu Jamming-Angriffen in [11], und erforschten die Auswirkung von Jamming-Angriffen auf die Paketzustellrate und implementierten „Channel Surfing“ - Techniken, um Interferenzen entgegen zu wirken.

Ebenso führten Zhao et al. [12] eine Studie zur Paketzustellrate in dichten WSN durch. Obwohl sie störende Übertragungen auf der MAC-Ebene berücksichtigen, wird kein systematischer Angriff in ihrer Arbeit betrachtet.

Eine weitere Arbeit, die sich mit der Paketzustellrate befasst, wurde in [13] vorgestellt. Hauer et al. evaluierten die Auswirkungen von WLAN-Interferenz auf die Paketzustellrate in IEEE 802.15.4 Body Area Networks.

Kürzlich stellten Lu et al. [14] ein System zur Erkennung von Jamming-Angriffen in zeitkritischen Netzwerken vor. Sie schlugen eine neue Metrik zur Performanzmessung vor. Eine Nachricht wird hierbei als ungültig angesehen, wenn die Nachrichtenverzögerung einen bestimmten Grenzwert überschreitet. Sie analysierten die Auswirkungen von Jamming-Angriffen auf diese spezielle Metrik.

Mehrere Arbeiten untersuchten Linkqualitätsmetriken, z.B. um die Dienstgüte zu verbessern. Liu et al. [15] führten einen Linkschätzer basierend auf maschinellen Lernverfahren ein. Ihre Modelle können die Linkqualität vorhersagen, indem sie eine Kombination aus PHY-Parametern (Signal to Noise Ratio (SNR), Link Quality Indicator (LQI) und RSSI) und der Packet Reception Rate (PRR) als Eingabe verwenden. Boano et al. [16] analysierten die Kombination von PRR, SNR und LQI zu einer einzigen, robusteren Metrik um die Linkqualität zu schätzen. Boano et al. [17] entwickelten auch $JamLab$, ein System zur Erzeugung von Interferenz, und untersuchten den Einfluss auf die PRR.

IV. Zusammenfassung und Ausblick

Wir identifizieren eine große Anzahl an leichtgewichtigen Metriken, die lokal an den Sensorknoten ausgelesen werden können. Inwieweit sich diese Metriken eignen, um DoS-Angriffe zu erkennen, muss in einem weiteren Schritt untersucht werden. Hierzu haben wir ein Angriffs-Testbed entwickelt, um die Auswirkungen von Jamming- und Blackhole-Angriffen auf die Metriken zu analysieren. Die Auswertung dieser Ergebnisse steht teilweise noch aus, jedoch lässt sich bereits sagen, dass gewisse Metriken wie die Paketzustellrate statistisch signifikant unterschiedliche Werte im Angriffsfall aufweisen und sich somit sehr gut zur Angriffserkennung eignen.

Danksagungen

Diese Arbeit wurde im Rahmen des Projektes EMERGENT im Software-Cluster (www.software-cluster.org) erstellt und mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) unter dem Förderkennzeichen „01IC10S01“ teilweise gefördert und wurde unterstützt durch LOEWE CASED (www.cased.de). Die Verantwortung für den Inhalt liegt bei den Autoren.

Literatur

[1] P. Suriyachai, J. Brown, and U. Roedig, “Time-critical data delivery in wireless sensor networks,” in *DCOSS*, 2010.
[2] 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 *IPSN*, 2007.
[3] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, “Energy analysis of public-key cryptography for wireless sensor networks,” in *PerCom*, 2005.
[4] A. Liu and P. Ning, “Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks,” in *IPSN*, 2008.
[5] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in *CCS*, 2002.
[6] D. Liu and P. Ning, “Establishing pairwise keys in distributed sensor networks,” in *CCS*, 2003.
[7] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, “A key management scheme for wireless sensor networks using deployment knowledge,” in *INFOCOM*, 2004.
[8] F. Liu, X. Cheng, and D. Chen, “Insider attacker detection in wireless sensor networks,” in *INFOCOM*, 2007.
[9] Z. Yu and J. J. Tsai, “A framework of machine learning based intrusion detection for wireless sensor networks,” in *SUTC*, 2008.
[10] W. Xu, W. Trappe, Y. Zhang, and T. Wood, “The feasibility of launching and detecting jamming attacks in wireless networks,” in *MobiHoc*, 2005.
[11] W. Xu, W. Trappe, and Y. Zhang, “Channel surfing: Defending wireless sensor networks from interference,” in *IPSN*, 2007.
[12] J. Zhao and R. Govindan, “Understanding packet delivery performance in dense wireless sensor networks,” in *SenSys*, 2003.
[13] J.-H. Hauer, V. Handzinski, and A. Wolisz, “Experimental study of the impact of wlan interference on ieee 802.15.4 body area networks,” in *EWSN*, 2009.
[14] Z. Lu, W. Wang, and C. Wang, “From jammer to gambler: Modeling and detection of jamming attacks against time-critical traffic,” in *INFOCOM*, 2011.
[15] T. Liu and A. Cerpa, “Foresee (4c): Wireless link prediction using link features,” in *IPSN*, 2011.
[16] C. Boano, M. Zuniga, T. Voigt, A. Willig, and K. Römer, “The triangle metric: Fast link quality estimation for mobile wireless sensor networks,” in *ICCCN*, 2010.
[17] C. Boano, T. Voigt, C. Noda, K. Romer, and M. Zuniga, “Jamlab: Augmenting sensornet testbeds with realistic and controlled interference generation,” in *IPSN*, 2011.