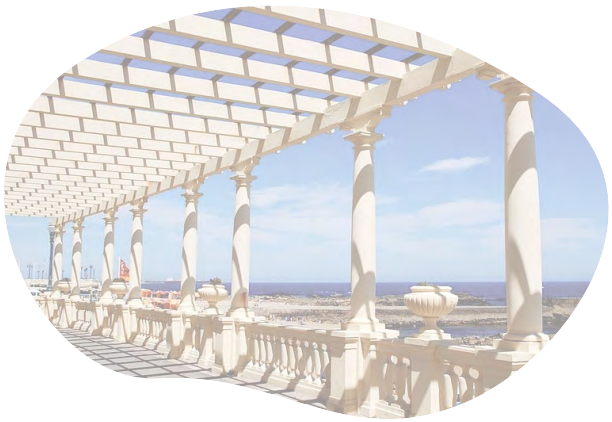
ewsn 2015

12th European Conference
on Wireless Sensor Networks

Poster/Demo Session



Porto, 9-11 of February

EWSN 2015: Posters and Demos

Welcome to Porto, Portugal, and to the Poster and Demo Session of the 12th European Conference on Wireless Sensor Networks (EWSN).

EWSN is a highly reputed and selective conference in the area of sensor networks and the Poster and Demo session provides an ideal platform to disseminate early research results and to showcase exciting demonstrations.

This year we received a total of 18 submissions from which 13 were accepted, in particular 8 demos and 5 posters. After an initial evaluation, submissions were ranked into one of the categories: accept, borderline and reject. The borderline submissions were discussed in detail among the co-chairs. The selected submissions cover a variety of topics ranging from security, localization and energy efficiency, to innovative network designing tools and programming frameworks.

Like in the previous year, in addition to the traditional Poster and Demo session, the authors of the accepted submissions will participate in a "1-minute madness" session where they will have one minute to pitch their work to all EWSN attendees. Don't miss it!

The Poster and Demo session is an integral part of the EWSN conference and we want to express our gratitude to the General Chair, Eduardo Tovar, to the TPC Co-Chairs, Tarek Abdelzaher and Nuno Pereira and to the local organization involved in the session setup.

We hope that you will enjoy the program and that you will strongly consider submitting posters and demos to future editions of EWSN.

The Poster and Demo Co-Chairs of EWSN 2015 edition

Vlado Handziski, TU Berlin, Germany

Raja Jurdak, CSIRO, Australia

Ricardo Severino, CISTER/ISEP, Portugal

Contents

Posters

Demos

Poster Abstract: A Modular Programming Approach for IoT-Based Wireless Sensor Networks

Shashank Gaur, Nuno Pereira, Vikram Gupta, Eduardo Tovar
CISTER Research Institute
ISEP, Polytechnic Institute of Porto, Portugal
Email: {sgaur,nap,vigup,emt}@isep.ipp.pt

***Abstract*—We aim to build a programming framework for IoT-based WSN, where we try to free the programmer from responsibilities of resource management, application distribution, and code deployment. As a foundation of this framework, we propose a modular programming model which lets programmer describe the application functionality with minimum knowledge of resources available in the WSN. We propose that using such programming model, the application code can facilitate the automatic deployment and management of IoT-based WSN applications.**

I. Introduction

Today, the Internet of Things (IoT) plays an important role in the research and development of Wireless Sensor Network(s). IoT has already provided standards such as 6LoWPAN[1] and COAP[2], which has enabled interoperable WSNs. Using these new standards, IoT-based WSNs are able to interact with different kind of systems quite easily, and in such cases, the programmer has access to multiple resources other than just sensor nodes, such as cloud infrastructures, and mobile devices.

To use such systems effectively, the programmer should be freed from many of the details of managing resources (communication, processing, sensing, power) and focus on the functionality. However, the abstraction between programmer and these resources still remains largely unresolved. For any changes in the application or WSN itself, intervention from the programmer or user is required. If a node fails, either the programmer needs to resolve the situation explicitly while writing the application, or the user needs to intervene to resolve the issue on the spot. The situation occurs when it comes to adding new nodes into the network, or deciding how the resources needed by an application (such as sensors, actuators, processing capabilities) can be used. This kind of scenarios requires the programmer to have knowledge about complete WSN and its nodes. There exists some related work in macroprogramming abstractions for WSN. Flask[3] is an example of similar kind where node level code is generated from meta language, but targeted to ensure the space and time behavior in WSN.

In order to extend WSN into a larger user base, it is essential to provide programmer with opportunity to write simple code for a task without worrying about each and every detail of the system. Drawing from works that enable reconfigurable in-network processing in IoT-based WSN such as T-Res[4], we aim to build an easy to program system which can manage the resources on behalf of the user, detect changes in the application according to rules defined by the user, and automatically deploy code accordingly. CITA[5] is an interesting example in such direction, which triggers an application only when necessary conditions are met. But it still doesn't provide a programming approach which let user write simple functionalities without worrying about the system details.

II. Programming Model

Consider the following motivating example depicted in Figure 1. Assume a simple application that activates the A/C in a room if the temperature is above a given temperature, only if someone is present in the room. One way to achieve this would be to collect the output from all the temperature sensors to calculate mean. In case of the mean being high enough, the motion sensor output will be checked before activating the actuator. Let us also assume there is more than one motion sensor in the room. While writing such program in existing systems, a programmer has to define which temperature and motion sensors to use. Additionally, the programmer would have to define what is the behaviour in case of a failure. This simple scenario introduces the need for a programming framework which can alleviate the programmer from such issues.

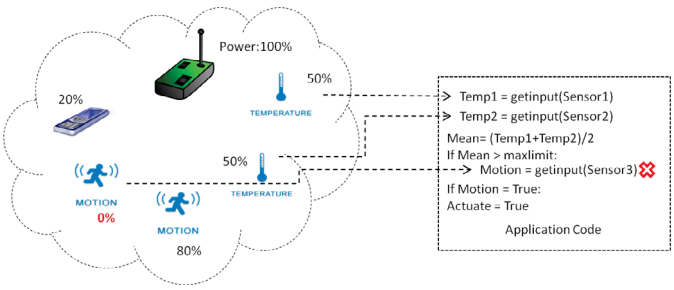


Fig. 1. WSN Motivating example: A programmer has to consider the usage of resources (sensors, processing, ...), and deal with sensor failures

We propose a block-based declarative programming model for IoT-based WSNs. In the proposed programming model, an application is combination of one or more block(s) of code and each block of code defines some functional part of the application. We name these blocks *Jewel*s.

Jewel is a modular piece of code, and one or many Jewels combined together can be represented as a single application. Jewel can be defined as standalone program, which has some essential and optional attributes. These attributes are *Input*, *Output*, *Code*, and *Local Variables* as shown in Figure 2.

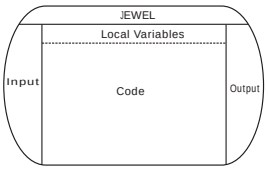


Fig. 2. Jewels Architecture

The functionality part of Jewel is represented by the *Code* attribute. Any simple task can be programmed inside this attribute irrespective of the device(s) connected as input and/or output. Because of this the programmer can be detached from the device details. Also if there are any changes in devices associated with the Jewel, it remains hidden for the code attribute. In order to help execute the code inside jewel, there might be need of some extra information to be provided by programmer. This can be represented using the *Local Variables* attribute.

A Jewel can have more than one *input* from or *output* to Devices and/or Jewels. However it is not necessary to execute the Jewel itself on one of the input or output devices associated with it. Since the code attribute of Jewel is isolated from the device details, a Jewel can be deployed or moved to anywhere in the WSN. This enables the programmer to write complex applications in simple ways without worrying about details of device, where the code will actually be executed.

Let us reconsider the example mentioned in the beginning of this section. The code for such example can be easily divided into two functions, one to keep the track of mean temperature and according to that second function will follow up change in Motion to provide actuation instructions.The two functions are shown below:

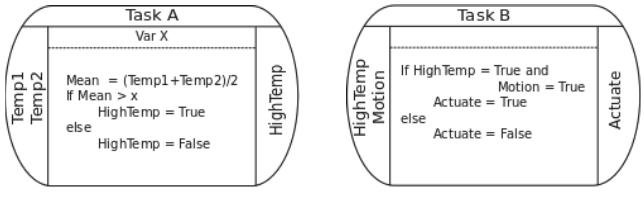
Listing 1. 'Task A'

```
function Taks_A:
{
variable x = 30 C
HighTemp = False
Mean = (Temp1+Temp2)/2
if Mean > x
  HighTemp = True
else
  HighTemp = False
return HighTemp
}
```

Listing 2. 'Task B'

```
function Task_B:
{
if HighTemp = True and
          Motion = True
  Actuate = True
else
  Actuate = False
return Actuate
}
```

These two function can be represented as jewels as shown in Figure 3 below.



(a) Task A Jewel (b) Task B Jewel

Fig. 3. Jewels for the example

The complete example can be represented as a flow graph of Jewels connected to all the devices involved, as shown in Figure 4. The code inside Jewel A and B are not concerned with the details of input and output resources. The Jewel B can have any motion sensor as input and that will not affect the code or whole application. Hence if one motion sensor fails, the framework can easily replace it with another, without any intervention from the programmer. The only constraint with Jewels is that the input should fall under conditions provided by programmer, if any. Due to the modular approach, it is also easy to replace one or many jewels in a flow graph without making changes in complete application.

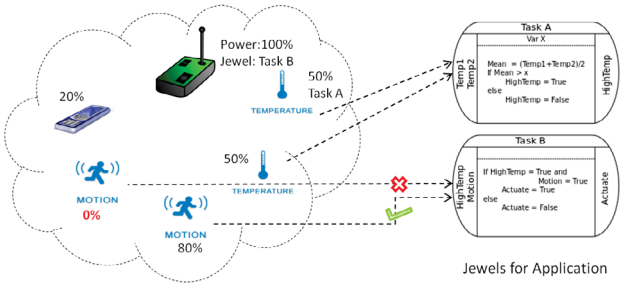


Fig. 4. Jewels and Devices

III. Future of Jewels

The Jewels are the basic building block for the earlier proposed programming framework. When combined together multiple jewels can represent a single application, and change in one jewel in the combination can change the application. Hence we envision a repository of jewels, where programmer can use existing jewels to build desired application instead of writing same code again and again.

With the modular architecture of the Jewel based programming framework, it should be quite efficient to design a smart resource manager for IoT based WSNs. With such programming model and resource manager in place, smarter WSNs can be achieved which can manage changes in application or resources without intervention from the programmer.

References

[1] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons, 2011, vol. 43.

[2] C. Bormann, A. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *Internet Computing, IEEE*, vol. 16, no. 2, pp. 62–67, March 2012.

[3] G. Mainland, G. Morrisett, and M. Welsh, "Flask: Staged functional programming for sensor networks," *ACM Sigplan Notices*, vol. 43, no. 9, pp. 335–346, 2008.

[4] D. Alessandrelli, M. Petraccay, and P. Pagano, "T-res: Enabling reconfigurable in-network processing in iot-based wsns," in *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*. IEEE, 2013, pp. 337–344.

[5] H. B. Lenin Ravindranath, Arvind Thiagarajan and S. Madden, "Code in the air: Simplifying sensing on smartphones."

Poster Abstract: Deployment, Reconfiguration and Adaptation through Modelling and Simulation in DREAMS

Richard Figura, Sascha Jungen, Matteo Ceriotti, Pedro José Marrón
Pervasive Computing, Networked Embedded Systems Group (NES)
University of Duisburg-Essen, Germany
richard.figura, sascha.jungen, matteo.ceriotti, pjmarron@uni-due.de

***Abstract*—Heterogeneous wireless sensor networks offer a pragmatic solution for environmental monitoring. Static sensors provide a reliable infrastructure and mobile unmanned aerial or ground vehicles allow on demand sensing or act as a platform for automatic sensor deployment. However, this heterogeneity of devices immersed in the monitored environment increases the system complexity. Deployment, reconfiguration and adaptation of the services to the dynamics of the environment and of the system itself is challenging, jeopardising reliability. In this work, we present DREAMS, a framework that combines both offline and online planning of stationary and mobile resources to continuously sustain application reliability in dynamic scenarios.**

I. Introduction

Sensor networks are getting deployed in an increasing number of scenarios, including applications for environmental awareness, e.g., pollution monitoring [1]. At the same time, actuation and control of embedded systems have reached a stage in which unmanned aerial or ground vehicles (UAVs, UGVs) are considered more and more reliable solutions for mobile sensing and automatic deployment of sensors. While this heterogeneity provides a new degree of freedom for data collection and environmental monitoring, the planning of such systems becomes more and more difficult. Furthermore, both the system and the environment evolve over time threatening the achieved reliability. Incorrect or suboptimal configurations and adaptations, e.g., positioning of static nodes or trajectories of mobile nodes, can endanger the ability of a system to meet application requirements, e.g., throughput, latency or lifetime.

The traditional approach followed to identify an optimal configuration requires monitoring the system behaviour, detecting configuration flaws or bottlenecks, identifying an alternative configuration and applying such new configuration. In doing this, system developers typically rely on simulation environments, e.g., COOJA [2] or Avrora [3], or on real testbeds in controlled environments, e.g., TWIST[4] or TrainSense [5]. However, the environment dynamics and the system characteristics intertwine with each other resulting in the final application performance. Gathering information about the current environment and system state becomes then crucial. Such information could be used as input for analytic, e.g., pTunes [6], or simulation-based, e.g., DrySim [7], frameworks that exhaustively explore in a single step the complete reconfiguration space to optimise a running stationary system.

A mobile system, instead, is subject to continuous changes, both in the system characteristics and in the impact of the environment on them. Such a scenario requires a continuous process that examines the system and environment dynamics to learn the employed models and analysis techniques online and suggests localised reconfiguration actions. In DREAMS, we aim at addressing such challenges in a comprehensive framework that combines both offline and online planning of stationary and mobile resources to continuously sustain application reliability in highly dynamic scenarios.

II. Approach

Figure 1 depicts the DREAMS framework for deployment, reconfiguration and adaptation. We base the implementation of this framework on COOJA[2] and IRIS [8], open source tools that we adapt to the needs of DREAMS. On the one hand, COOJA allows to accurately simulate large numbers of WSN applications and to evaluate the performance of system configurations prior deployment. IRIS, on the other hand, is a flexible tool for data processing and interaction with deployed systems. For DREAMS, we modified COOJA to allow a flexible remote interaction with the simulator. For example, we enable to remotely configure and start a simulation, change the state of a running simulation instance online and retrieve information, such as performance results, from a running or finished simulation instance. We exploit such extensions to interface COOJA with IRIS, in order to start a new simulation or interact with an existing one. In this way, IRIS can interact with deployed systems to monitor and control the current resources. The gathered data can then provide information to generate different models, e.g., radio models, with which configure the simulation environment and analyse the system behaviour to identify new reconfiguration plans. These adaptations have been integrated into the main IRIS distribution and made available online [9], so that the research community can benefit from our efforts.

The aforementioned modifications allow the interaction between heterogeneous deployed systems, flexible data processing components, and simulation services in a unified framework. We combine the different modules in a workflow,

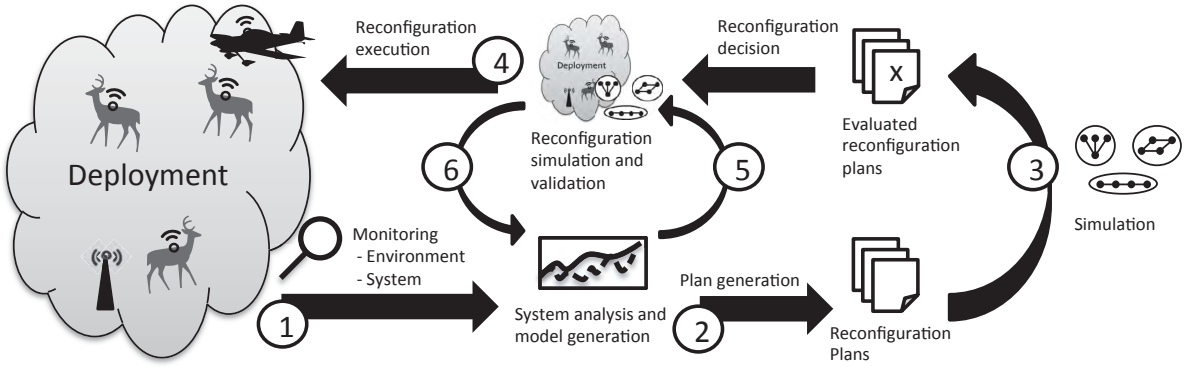


Fig. 1. DREAMS framework and its use for deployment and reconfiguration planning.

depicted in Figure 1, that satisfies the needs of deployment and reconfiguration planning when the system and environment characteristics change during operation. In particular, we follow three phases as described next.

A. Deployment Monitoring and Data Analysis

In the first phase, DREAMS monitors the systems performance and the environmental characteristics ①. This information includes the position of the nodes and their mobility, link quality measurements, e.g., PRR, SNR and LQI, as well as system performance metrics. This information is then analysed to evaluate the achieved performance of the running system and build system and environment models of the observed deployment. In particular, these measurements allow to build a graph based radio model as input for the system simulator. By continuously monitoring the changes in the link qualities and in the system performance, DREAMS allows to analyse temporal effects on the link quality caused by, e.g., daily variations or weather changes, and gather knowledge about the specific impact of the environment on the operational network.

B. Offline Reconfiguration Planning

In a second phase, DREAMS uses the gathered data to detect configuration flaws, e.g., high number of hops between source and sink, over-utilised links or nodes ②. The information about detected configuration flaws is then used to generate a set reconfiguration plans that consider the deployment of additional static or mobile nodes, or removal of existing ones. These reconfiguration plans are evaluated within a simulator ③, which uses the models generated in the previous phase. The outcome of this step is a set of evaluated reconfiguration plans, where each plan is associated with its expected performance metrics, e.g., lifetime, response delay, throughput, and its estimated costs. Finally, the user can evaluate the trade-offs between achievable quality and costs to decide on the final reconfiguration plan for the running system.

C. Online Reconfiguration Adjustment

In the last phase, a chosen reconfiguration plan is applied to the running system ④. Hereby, DREAMS allows to adjust the reconfiguration plan online if the monitored performance of the intended reconfiguration deviates from the expectations. This might happen in the case of unpredictable environmental effects, e.g., short- or medium-scale fading. To find an optimal configuration, DREAMS maintains a synchronised virtual copy of the deployed system in simulation, together with the information about the reconfiguration plans and the estimated system performance. During reconfiguration, the monitoring of the system and the environment ① continues, providing data useful to update the system and environment models. This information can be used to re-synchronise the virtual copy of the deployed system, by adjusting, e.g., the position of the nodes or the properties of the wireless links ⑤. If the resulting system performance deviates from the target goals, adjustments to the reconfiguration plan can be evaluated and applied ④. At the same time, the identified deviations provide valuable information to refine the environment and system models ⑥ and gather factual knowledge about the unexpected interplay between the system and its surrounding environment.

III. Conclusion

Simulation-aided planning based on observations from the operational system provides a simple yet effective solution for deployment reconfiguration, in particular in scenarios with dynamic system and environment characteristics. Our current goals involve the analysis of reconfiguration plans where application requirements are met by adjusting the position of stationary nodes or the trajectory of mobile devices. The same approach can also be used to reconfigure software parameters.

References

[1] Planet. [Online]. Available: www.planet-ict.eu
[2] F. Osterlind *et al.*, "Cross-Level Sensor Network Simulation with COOJA," in *LCN*, 2006.
[3] B. Titzer *et al.*, "Avrora: scalable sensor network simulation with precise timing," in *IPSN*, April 2005.
[4] V. Handziski *et al.*, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Network," in *RealMAN*, 2006.
[5] H. Smeets *et al.*, "TrainSense: A Novel Infrastructure to Support Mobility in Wireless Sensor Networks," in *EWSN*, 2013.
[6] M. Zimmerling *et al.*, "ptunes: Runtime parameter adaptation for low-power mac protocols," in *IPSN*, 2012.
[7] M. Strübe *et al.*, "DrySim: Simulation-aided Deployment-specific Tailoring of Mote-class WSN Software," in *MSWiM*, 2014.
[8] R. Figura *et al.*, "IRIS: Efficient Visualization, Data Analysis and Experiment Management for Wireless Sensor Networks," *EAI Endorsed Transactions on Ubiquitous Environments*, 2014.
[9] WSN Tools. [Online]. Available: http://wsntools.com

Poster Abstract: Performance Analysis of Data Serialization Formats in M2M Wireless Sensor Networks

Francesca Pacini and Femi A. Aderohunmu
and Paolo Pagano
National Inter-University Consortium
for Telecommunications (CNIT), Pisa-Italy
Email: {firstname.lastname}@cnit.it

Andrea Azzara and Matteo Petracca
and Stefano Bocchino
Scuola Superiore Sant'Anna,
Network and Embedded Systems Lab, Pisa-Italy
Email: {firstname.lastname}@sssup.it

***Abstract*—Data and information processing on constrained devices poses new challenges to the accomplishment of a robust cyber-physical system. Advancements in the context of Wireless Sensor Networks (WSN) and the new Internet of Things (IoT) technologies are paving ways for novel applications following the machine-to-machine (M2M) communication paradigm. However, due to the limited resources available on a typical sensor device, and in order to achieve lower energy consumption on the nodes, it is important to reduce the amount of the exchanged data.**

In this paper we conduct an analysis of two foremost data serialization methods viz., JSON and EXI, recommended by the ETSI M2M Technical Specification. To reach our goal, we evaluate the portability of the ETSI M2M data serialization formats on wireless sensor nodes and we measure their performance in terms of execution time, channel usage, and energy consumption. The results from our experiments show that on average EXI, when compared with JSON, is able to achieve an improvement with respect to energy consumption and channel usage, while maintaining a low memory foot-print.

I. Introduction

In order for WSN deployments to achieve the IoT vision, sensor nodes would be required to communicate on the internet and to be interoperable with other IoT systems. This necessitates that nodes would send large amount of data packets on the network, thus leading to huge burden on bandwidth and the network channel. In addition, transmitting several packets due to fragmentation by the OSI higher layers would mean energy limited nodes would die out quickly hence drastically limiting the lifetime of an IoT-enabled sensor node. A number of solutions have been proposed to address this problem. A particular strong candidate is the ETSI M2M protocol [3], defining different data serialization formats specified for M2M applications of which WSN could be part. These data serialization formats specify data compression methods on a network to achieve few fragmentation thus reducing the overall packet retransmission especially on a lossy media.

In today's Internet, Extensible Markup Language (XML) has become the *de facto* standard for data representation due to its flexibility. However, due to the verbosity and the processing overhead associated with XML, new data formats need to be adopted to achieve efficient communication in the world of constrained devices. The ETSI M2M recommends three data serialization formats: 1) Extensible Markup Language (XML) 2) JavaScript Object Notation (JSON) and 3) Efficient XML Interchange (EXI). The EXI format is a light-weight and compact representation of the XML that is targeted for use on constrained devices such as sensor nodes. EXI is designed to meet the requirements of high performance XML applications, significantly reducing the bandwidth overhead and enhancing the encoding and decoding performance. Some studies [1] concentrated on testing the applicability of EXI serialization method on embedded systems. However they do not provide insight on the performance of EXI in the realistic use case of M2M communication. A comprehensive performance evaluation of EXI is still missing in the literature.

We present a comparative study of the data serialization formats recommended by the ESTI M2M working group. In particular, we focus on the comparison of EXI[1] and JSON. To align our work with the ETSI standard, we conduct performance measurements for a declared set of messages deduced from the ETSI CoAP M2M Interoperability Test [2].

II. Implementation and Performance Evaluation

We provide performance evaluation of these data formats to better understand the tradeoffs on wireless sensor nodes. To conduct the performance evaluation, we ported the EXIp [5] on *Wismote* sensor nodes[2], while using the ETSI M2M messages as a payload benchmark. On the software side, we developed several test libraries[3], written in C, implementing the ETSI M2M interoperability test [2]. In addition, in order to perform the evaluations in a controlled environment, we used Cooja, a well-known simulator for WSNs. The choice of using Cooja will allow other researchers to easily replicate our measurements on the platforms supported by the simulator before proceeding to real hardware testing. We based our software implementation on the Contiki Operating System, which provides a full-fledged IoT network stack. Contiki is designed to support severely resource-constrained classes of hardware. Moreover, Contiki OS provides low-power Internet communication by fully supporting IPv4/IPv6 and low-power wireless standards such as the 6LoWPAN, RPL and CoAP.

The experimental setup consists of Wismote wireless sensor nodes, emulated in Cooja. The sensor nodes represent the communication endpoints of a generic M2M scenario. The setup configuration consists of three sensor nodes, acting as client, server, and 6LoWPAN border router. The client and the server are programmed to execute an M2M interaction using the test library we developed implementing the ETSI M2M test suite. The application is realized using the Erbium [4] CoAP library. CoAP endpoints exchange messages using our port of the EXIp library to encode and decode the data payload. The performance of the two ETSI M2M data serialization formats viz., JSON and EXI are compared according to the following criteria:

1) *Performance over ETSI M2M CoAP Interoperability Tests*: This measurement focuses on the execution time, channel usage and energy consumption of several M2M operations.

[1]EXI is a light-weight realization of the XML, thus we omit the comparison of XML

[2]http://www.aragosystems.com/en/wisnet-item/wisnet-wismote-item.html

[3]Available upon request to the authors

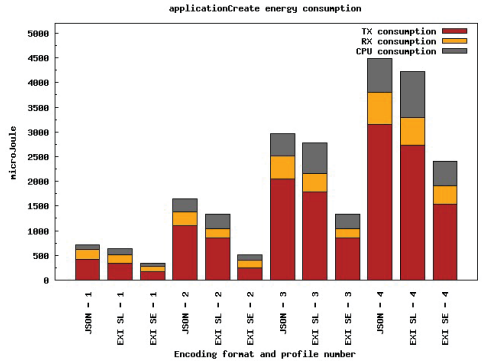


Fig. 1: ApplicationCreate: Energy consumption.

2) *Serialization complexity*: This measures the complexity with respect to the memory usage, serialization and deserialization times.

For evaluating the EXI format two configurations are considered, corresponding to Schema-Enabled (SE) and Schema-Less (SL) serialization. The "Schema-Enabled" compression mode (also known as "schema-informed") can achieve better compression performance compared to the Schema-Less mode, but requires both the encoder and the decoder to share an XML schema. In addition, the EXI SE uses schema information to improve compactness and efficiency.

III. Results and Evaluation

The ETSI M2M CoAP Interoperability Tests: These Tests consist of the execution of a set of operations (create, retrieve, update) on ETSI M2M resources (*application*, *subscription*, *contentInstance*). For instance in our tests we perform an update operation on sensor nodes to manipulate the sensor readings (e.g., temperature and humidity) mapped over ETSI M2M resources. In our setup, we configured one CoAP endpoint to simulate a M2M Device Application and the other endpoint as the M2M Gateway. The ETSI M2M standard defines different profiles for message instances. We tested different message instances sent between the CoAP endpoints, which differ in terms of length and complexity. We categorized the message instances according to Profiles 1 to 4, where Profile 1 is shorter with less complexity and Profile 4 is longer with more complexity.

In Fig. 1 we show the energy consumption of an *applicationCreate* operation with respect to transmission, reception and CPU energy consumptions. From this figure, we can observe that EXI SE performs better compared with EXI SL and JSON. This is due to the fact that a smaller payload (packet) implies fewer transmission hence lower radio usage, which has a higher impact on power consumption than the one deriving from the CPU usage. Table I shows that EXI SL reduces channel usage on average by 15% with respect to JSON, while EXI SE reaches an average of 50%.

TABLE I: applicationCreate channel usage

Profile	JSON	EXI	
		Schema-Less	Schema-Enabled
	bytes	bytes	bytes
1.	245	232 (-5%)	155 (-36%)
2.	640	469 (-27%)	181 (-72%)
3.	1135	953 (-16%)	471 (-58%)
4.	1679	1448 (-14%)	870 (-48%)

To sum up, EXI SE outperforms both JSON and EXI SL configurations in terms of energy consumption, channel usage and the execution times. However EXI has significantly higher memory requirements. For completeness, we summarize in Table II the results obtained from our experiments for all the ETSI M2M resources i.e., *application*, *subscription* and *contentInstance*. We rank the performance from 1 to 3, where

TABLE II: Summary of Results

Criteria	JSON	EXI Schema Less	EXI Schema Enabled
memory occupation			
ROM	1	2	3
RAM	1	2	3
applicationCreate			
Execution Time	2	3	1
Energy Consumption	3	2	1
Channel Usage	3	2	1
applicationRetrieve			
Execution Time Sc. 1	3	2	1
Execution Time Sc. 2	2	3	1
Energy Consumption	3	2	1
Channel Usage	3	2	1

Criteria	JSON	EXI Schema Less	EXI Schema Enabled
applicationUpdate			
Execution Time	-	-	1
Energy Consumption	3	2	1
Channel Usage	3	2	1
subscriptionCreate			
Execution Time	2	3	1
Energy Consumption	-	-	1
Channel Usage	3	2	1
contentInstanceRetrieve			
Execution Time Sc. 1	3	2	1
Execution Time Sc. 2	-	3	-
Energy Consumption	-	-	1
Channel Usage	-	-	1

1 ranks as the best performance and 3 ranks as the worst. The data serialization methods considered are placed in relative order whenever possible. The dash ("-") signifies situations where, according to the chosen benchmark, evaluation of the results is not needed. From our results, we made the following observations:

1) EXI SE performs better in terms of energy consumption, compared with JSON and EXI SL. Similarly, with respect to the execution time, EXI SE performs better in the majority of cases. This is however achieved at the cost of a higher memory occupancy, which can still be accommodated on a sensor node.
2) The average performance of EXI SL is comparable with JSON; in some cases it performs better than JSON with respect to energy consumption. In our implementation, EXI SL execution time is better to an equal extent compared with JSON especially when the serialization time is negligible.
3) With respect to channel usage, in all scenarios considered EXI performs better compared with JSON, especially when EXI SE is used.

IV. Conclusions

In this paper we conducted a comparative study of data serialization formats that are recommended by the ETSI M2M Technical Specification, namely JSON and EXI (Schema-Less and Schema-Enabled). Utilizing a light-weight data serialization format is particularly useful for constrained devices such as wireless sensor nodes. Lots of data would be generated among heterogeneous devices, hence it is necessary to understand the tradeoff between the different recommended data formats. Based on our results, EXI Schema-Enabled outperforms other comparable data formats with respect to energy consumption on the nodes, execution times and channel usage on the network. However JSON has significantly lower memory requirements, and is the recommended solution when working with heavily constrained sensor nodes with low memory. The result of this work will be adopted for the implementation of the ITS architecture designed within the ICSI Project approved by the European Commission.

V. Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 317671 (ICSI Intelligent Cooperative Sensing for Improved traffic efficiency).

References

[1] EXI: Efficient XML Interchange (EXI) Format Evaluation. http://www.w3.org/TR/exi-evaluation/. Accessed: 15-Sept-2014.

[2] ETSI. Machine-to-machine communications (m2m); interoperability test specification for coap binding of etsi m2m primitives. http://www.etsi.org/deliver/etsi_ts/103100_103199/103104/01.01.01_60/ts_103104v010101p.pdf. Accessed: 15-Sept-2014.

[3] ETSI. Machine-to-machine communications (m2m); mia, dia and mid interfaces. http://www.etsi.org/deliver/etsi_ts/102900_102999/102921/02.01.01_60/ts_102921v020101p.pdf, 2013. Accessed 15-Sept-2014.

[4] M. Kovatsch, S. Duquennoy, and A. Dunkels. A low-power coap for contiki. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 855–860, 2011.

[5] R. Kyusakov, J. Eliasson, and J. Delsing. Efficient structured data processing for web service enabled shop floor devices. In *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, pages 1716–1721, June 2011.

Poster Abstract: Towards Developing a Generalized Modeling Framework for Data Dissemination

Kamini Garg and Silvia Giordano
Networking Laboratory, ISIN-DTI, University of Applied Sciences of Southern Switzerland
{kamini.garg, silvia.giordano}@supsi.ch

***Abstract*—In this paper, we present a overview of our Markov chain based generalized modeling framework that takes into account real world characteristics and predicts the performance of data dissemination for a given environment. Our modeling framework allows multiple simultaneous pair-wise contacts among people under heterogeneous mobility, multiple data sources and different dissemination strategies. Our framework can be used to predict the performance of data dissemination process and provide bounds for different evaluation metrics: data dissemination time, optimal number of people required to maximize the spread of information and best relays. We also present initial results obtained from our modeling framework under broadcast data dissemination strategy and predict the upper bound of data dissemination time. Our simulation results show that, our modeling framework can provide tighter upper bound of data dissemination time within 5-10% error.**

I. Introduction

In recent years, proliferation of mobile devices enable people to gather and share information from their neighborhood without the help of a centralized system. With the help of different communication interfaces (like Bluetooth, WiFi), people can collect context based up-to-date information from their neighbors and also further spread it to other geographical regions. The success of any encounter-based data dissemination application critically depends how data gets disseminates in a given region. Therefore, to predict the performance of these applications, there is a need to predict the performance of data dissemination process. Some of the key evaluation metrics to measure the performance are: maximum time taken to spread information i.e data dissemination time; optimal number of people required to maximize the spread of information etc.

There is a need for a generalized model that can mimic and capture real world characteristics like heterogeneous contacts, temporal movements of people, multiple simultaneous contacts and different data dissemination strategies. Existing works do not consider all these aspects thus, fail to provide a realistic evaluation of data dissemination process [1]. In this paper, we propose a generalized modeling framework that collectively considers all real world characteristics of data dissemination process. We summarize our contribution as follows:

- Our Markov chain based modeling framework capture and predict the performance of data dissemination under real world characteristics. To the best of our knowledge, the proposed framework is the first one that collectively considers all real world aspects of data dissemination process.
- As opposed to existing works, the evaluation metrics obtained from our modeling framework is not only limited to data dissemination time. It can be used to predict the data dissemination performance in multiple dimensions like optimal number of people required to maximize the spread of information, finding the best relays to maximize information spread and/or to minimize data dissemination time.

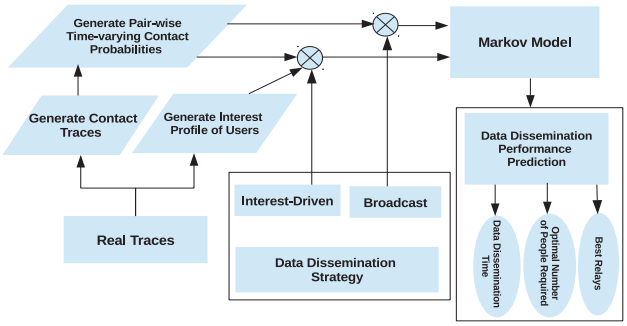


Fig. 1. Basic components for our generalized modeling framework.

Our modeling framework is work in progress and to show the applicability of our framework, we present initial results obtained for data dissemination time using real world traces.

II. Proposed Modeling Framework

Figure 1 outlines the different components of our modeling framework. It utilizes the real world traces to generate contact traces and interests profile of users. Contact traces present the mobility pattern of people and, interests profiles determine the likelihood to exchange data (in case of interest-driven data dissemination strategy). Further from contact traces, we generate the pair-wise time-varying contact probability for all users. In real world heterogeneity lies in contact patterns of people along with temporal variation. Therefore, the inclusion of pair-wise time-varying contact probability will enable our modeling framework to mimic the real world contact patterns of people. Currently our modeling framework is designed to support two data dissemination strategy: broadcast and interest-driven. In case of broadcast data dissemination strategy, it only utilizes pair-wise time-varying contact probability to evaluate the performance of data dissemination process. While, in case of interest-driven strategy both pair-wise time-varying contact probability and interests profile of users will be used.

Markov model is an integral part of our modeling framework. It takes pair-wise time-varying contact probability as an input and models data dissemination under synchronous model (allows multiple simultaneous contacts among people at each time slot). We consider each environment as a network of N mobile users and M data sources. We assume that every data source has a distinct data message and each mobile user is interested to gather messages from all data sources. In the end

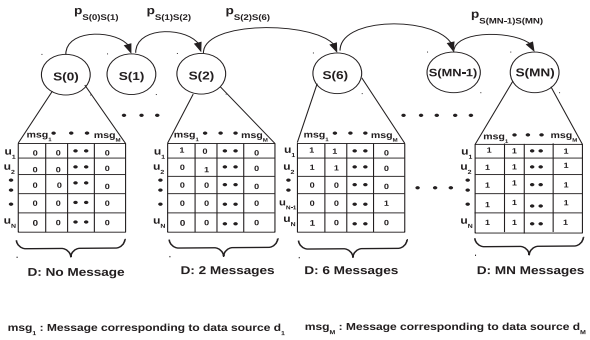


Fig. 2. One realization of our Markov model starting from $S(0)$ to $S(MN)$ with their respective Data matrices D and transition probabilities. This realization also shows the jump from $S(2)$ to $S(6)$ due to simultaneous contact between two mobile nodes and mobile node and data sources.

there will be a total of MN (or $M \times N$) data messages stored in the network (each one of the N users will have M data messages). Each state of our model represents the total number of messages in the network (starting from 0 to MN) where $S(0)$ and $S(MN)$ represents states with no message and MN messages respectively. Figure 2 represents one realization of the Markov chain where each state $S(x), x \in [0, MN]$ can be viewed as data matrix D of size $N \times M$, where number of non zero elements represent the number of messages present in network. The probability of transition can be calculated using our pair-wise contact probability distribution. Once we reach the final state $S(MN)$, the transition probability to remain in the same state will be 1.

Utilizing the Markov model, we can predict the bounds for data dissemination process in multiple dimensions and not only restricted to data dissemination time. The other evaluation metrics are to find the optimal number of people to maximize the data dissemination and to find the best relays in the network to accelerate the information spread in the network. For all evaluation metrics, we can use different algorithms for different data dissemination strategies.

III. Premilinary results: Upper bound of Data dissemination time under broadcast strategy

To show the validity and accuracy of our modeling framework, we present the initial results obtained for data dissemination time under broadcast strategy.

Definition 1 (Data Dissemination Time T_{dss}): We define it as the time until all mobile users receive data from all data sources. The time t at which all N mobile users receive data and all elements of D become 1 is defined as data dissemination time T_{dss}.

A. Estimation of upper bound of T_{dss} from our Markov model

The total time spent in each state before reaching the final state $S(MN)$ will be equivalent to the data dissemination time T_{dss} in a network. Let T_x be the time spent in each state $S(x)$ before transition to any other state. T_{dss} can be calculated as follows:

$$T_{dss} = \sum_{x=0}^{MN-1} T_x \qquad (1)$$

Due to simultaneous multiple contacts and multiple data messages, our Markov chain is not always a 1-step transition chain (refer Figure 2). To provide tighter upper bound of

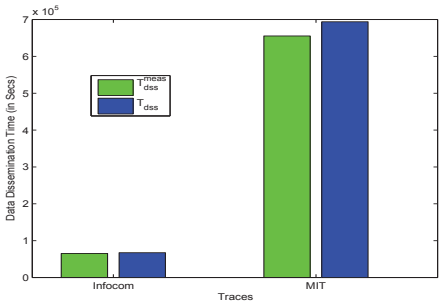


Fig. 3. Comparison of measured Data dissemination time T_{dss}^{meas} against T_{dss} obtained from our model for INFOCOM and MIT traces.

T_{dss}, we utilize the impact of long tail cutoff property of inter contact time on data gathering process [2]. We believe that the consideration of this long tail cut-off can provide a much tighter bound of T_{dss} because it closely articulates the real world data gathering process. For this reason we calculate a cut-off point α beyond which the rate of data collection becomes very slow and exhibits a long tail over time. We first replicate the fast rate of data gathering till cut-off point and calculate the total time required to directly reach till α state i.e. from $S(0)$ to $S(\alpha \times MN)$. Further, to mimic the long tail, we calculate the time spend in each state from $S(\alpha \times MN)$ to $S(MN)$. Finally the data dissemination time T_{dss} can be calculated from summation of all times.

B. Results

To capture the impact of diverse environments and to investigate the accuracy of our model, we use two real-world contact traces: (i) Infocom 2005 trace (INFOCOM) [3], (ii) Reality Mining trace (MIT) [4]. In Figure 3, we present the bounds of T_{dss} obtained from our model under broadcast data dissemination strategy and show that our model is able to mimic the real data dissemination time T_{dss}^{meas} (measured from real traces) within 5-10% error.

IV. Next Steps

Currently our modeling framework predicts the performance of data dissemination time under real world characteristics for broadcast data dissemination strategy. Our next step is to extend our model to predict upper bound of data dissemination time for interest-driven strategy. Further, we also plan to implement algorithms for other evaluation metrics i.e. to find best relays and optimal number of people required to maximize the data dissemination under both dissemination strategies.

Acknowledgment

This work was supported by the EU FP7 ERANET program under grant CHIST-ERA-2012 MACACO.

References

[1] A. Clementi, R. Silvestri, and L. Trevisan, "Information spreading in dynamic graphs," in *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*, ser. PODC '12, New York, USA, 2012, pp. 37–46.

[2] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović, "Power law and exponential decay of inter contact times between mobile devices," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, ser. MobiCom '07, New York, USA, 2007, pp. 183–194.

[3] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2006-01-31)," Downloaded from http://crawdad.org/cambridge/haggle/, Jan. 2006.

[4] N. Eagle, A. S. Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15 274–15 278, 2009.

Poster Abstract: Security Challenges in Indoor Location Sensing using Bluetooth LE Broadcast

Prasant Misra[†], Shahid Raza[‡], Vasanth Rajaraman[†], Jay Warrior[†], Thiemo Voigt[‡⋆]
[†]Robert Bosch Centre for Cyber Physical Systems, Indian Institute of Science, Bangalore, India
[‡]SICS Swedish ICT Stockholm, Sweden
[⋆]Uppsala Universitet, Uppsala, Sweden
Email: {prasant.misra, vasanth.rajaraman, jay.warrior}@rbccps.org† {shahid, thiemo}@sics.se[‡]

***Abstract*—As we consider a new generation of Internet of Things and Humans (IoTH) applications that place humans at the epicenter of the control system the need to gather information from the immediate vicinity, in addition to global clues, is gaining importance. The loosely coupled Bluetooth Low Energy (BLE) data collection framework enables a new way of architecting IoTH systems where resource constrained BLE advertisers broadcast events, and devices inevitably carried by humans (such as smartphones) implicitly gather such notifications. While such a mechanism significantly alleviates data scavenging, it introduces serious limitations in terms of operational security. In this work, we show the applicability of BLE broadcast advertisements for indoor location sensing (as part of an IoTH application) and demonstrate an attack on the same system. Based on this preliminary case study, we discuss other security implications on BLE broadcasting.**

I. Introduction

Location sensing is highly desirable in distributed systems, including many applications in the Internet of Things and Humans (IoTH). It binds humans and 'Things' to a common scope by providing valuable context ('where?') for interpretation ('what?'). For IoTH, Bluetooth Low Energy (BLE) [1] - a subset of the recent Bluetooth v4.0 stack - provides a low-power and loosely coupled mechanism for sensor data collection with ubiquitous units (e.g., smartphones and tablets) carried by humans. This fundamental design paradigm of BLE is enabled by a range of *broadcast advertising* modes. These new modes offer unidirectional communication between two or more LE devices using advertising events, thereby achieving a communication solution without entering into a bonded connection (as required by Classic Bluetooth devices). Such a loosely coupled manner of data transfer is undoubtedly more energy efficient but it also unearths other limitations. For example, BLE broadcast mechanisms are *highly* vulnerable to a range of *security* threats.

To ground this discussion, we first show the applicability of BLE broadcast advertisements for an IoTH application - indoor location sensing (Section II). Based on this system architecture, we demonstrate the feasibility of launching a packet injection attack (Section III) and discuss other possible security implications (Section IV).

II. Indoor Location Sensing: System Overview

Our system is composed of *three* units: proximity beacon, controller and coordinator. The *proximity beacon* unit consists of a resource constrained sensing tag that is deployed in the region-of-interest. It is responsible for engaging in undirected BLE broadcasts, and measurement of simple physical parameters (if necessary). The *controller* unit consists of a resourceful device that is capable of listening and receiving broadcast data contained in BLE advertisements. The *coordinator* unit is a private Cloud service that aggregates high-level location details. For our indoor location sensing study, the beacon unit is a custom designed CC2540Cheep BLE platform, the controller unit is an Android v4.4.4 smartphone that uses Bluedroid, and the coordinator unit is a Amazon EC2 cloud computing service.

The custom designed beacon platform, CC2540Cheep, consists of a TI CC2540 [2] system-on-chip (SoC), a RF balun filter, a chip antenna, and a CR2032 battery holder. It has a dimension of $[28 \times 25 \times 8]$ mm. The SoC comprises an 8051 microcontroller core, a Bluetooth v4.0 compliant 2.4 GHz RF transceiver, 8 kB RAM, 256 kB programmable flash, and a 12 bit ADC.

BlueDroid is the default Bluetooth stack of Android. It consists of two layers: the Bluetooth Embedded System (BTE) layer holds the core Bluetooth functionality and the android application code (at the framework level) that utilizes the APIs of `android.bluetooth` to interact with the bluetooth hardware that internally, through the Binder IPC mechanism, calls the Bluetooth process (both the Bluetooth service and profiles). The packaged android app uses JNI to call into the hardware abstraction layer (HAL) and to receive callbacks.

The basic *ranging* mechanism uses the broadcaster mode of BLE to transmit unsolicited advertisement packets from the beacon units. Broadcast packet transmissions are kept short (10 bytes). They contain only the necessary fields such as the preamble, header, medium access control (MAC) address and checksum, but no application payload. As part of the advertisement event, beacons broadcast advertisement packets on each advertisement channel within a specified time period and transmit power level. Passively listening (mobile) controller units are able to receive such broadcasts when they are in the proximity range of the beacons. If the beacons are placed at regions-of-interest (for contextual tagging), then the proxemic cue helps to correlate the presence of mobile units to the contextual vicinity. Here, context binds 'people' (carrying the mobile units) and 'Things' (beacon units) to a common scope, and hence, eases mining of relevant location information. The beacon-(mobile)controller proximity information is pushed into the coordinator unit for generating a location heat map over a configurable time duration.

In the following section, we demonstrate the effect of a packet injection attack on our indoor location system.

(a) *System under normal conditions* (b) *System under attack conditions*

Fig. 1: **Heat map of the movement of people in the indoor environment.** *(a) shows the ground truth results, while (b) shows the impact of a packet injection attack on the location results.*

III. System under Packet Injection Attack

The experimentation space was the third floor of our office building. It was instrumented with a set of 26 beacon units. The beacons were configured to broadcast advertisement packets at an interval of 750 ms and at their lowest transmit power of -23 dB. For our evaluation, 10 people carried Android smartphone running the location service for about 6 hours. Proximity events were streamed to the Cloud service that aggregated the data from all 10 mobile units and generated a real-time location heat map of people in the office space.

During the experimentation period, a packet injection attack was launched on the location sensing system by impersonating a benign beacon with a malicious one. The malicious beacon was placed in the same zone as the benign beacon so as to tag it with the same context, but was configured to broadcast BLE packet at an interval of 100 ms and at a higher transmit power of -6 dB. However, for the purpose of identifying rogue packets, we added an additional identifier in the advertisement message that could be subsequently filtered with a simple rule.

Fig. 1 shows the raw heat map, and represents the density of user presence at each location over the entire duration, with the color going from light yellow (low density) to deep red (high density). Fig. 1(b) shows the impact of the attack on the location results; while Fig. 1(a) reports the same once all rogue information are filtered. While under a normal situation (as expected), the regions of high interest to people were observed to be the oval meeting room, the office front desk and entrance, and one of the closed cabin spaces; the situation gets entirely flipped under the attack scenario where it appears that people are mostly interested in gathering at one office corner.

IV. Security Implications of BLE Broadcasting

Though BLE broadcasts can effectively distribute data (as shown in previous sections), it is subjected to packet injection attacks. Such an attack state can be mitigated by Source-Specific Multicast (SSM) [3], but is not provisioned in the unsolicited broadcast mode of BLE.

When a malicious device impersonates the *public device address* or *static address* of a legitimate device, the centralized module of the localization service can filter out the injected packets as different locations are reported from the same source address. Though it helps in eliminating the wrong location information, the valid information from the legitimate source with corresponding address is also excluded. An adversary can implant multiple malicious nodes to disrupt the entire system; but then it is a question of cost-benefit analysis. If a service uses the BLE *non-resolvable private address* or privacy aware *resolvable private address*, it is hard to distinguish between fake and legitimate sources. Malicious devices can also launch simple, yet effective attacks to confuse the localization algorithm such as rapid variation in transmit power levels to make the device appear/disappear.

A BLE-enabled application (such as an Android app that is purely based on the broadcast mode[1]) can develop custom designed security solutions at the application layer but this has implications on the interoperability with other applications. The restricted packet size (31 bytes) of BLE broadcast messages further limits the use of a sophisticated broadcast authentication protocol such as Tesla [4].

V. Conclusion

Using a location sensing system as a case study for IoTH applications, we have demonstrated the severity of a packet injection attack on the broadcast mode of BLE communication[2]. Currently, we are working towards understanding and evaluating other such security attacks and designing algorithms for their mitigation.

References

[1] "Bluetooth SIG: Specification Adopted Documents," https://www.bluetooth.org/en-us/specification/adopted-specifications.

[2] "CC2540," http://www.ti.com/product/cc2540l.

[3] S. Bhattacharyya, "An Overview of Source-Specific Multicast (SSM)," RFC 3569 (Informational), IETF, Jul. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3569.txt

[4] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," *CryptoBytes*, vol. 5, no. 2, pp. 2–13, Aug. 2002.

[1] Security features can be enabled in the BLE connection mode, but it supports only unicast communication and requires active input by the user.

[2] While Bluetooth 4.2 [1], the latest update to the core specification, does introduce many additional benefits; yet its security mechanism in the broadcast mode is still weak.

Demo Abstract: Automating WSN experiments and simulations

Rémy Léone*† Jérémie Leguay* Paolo Medagliani* Claude Chaudet†
* Thales Communications & Security – Gennevilliers, France
† Institut Mines-Télécom / Télécom ParisTech / CNRS LTCI UMR 5141 – Paris, France

***Abstract*—In wireless sensor networks (WSNs), as in every other discipline, people willing to evaluate the performance of an application or a protocol rely on modeling, simulation or experimentation. Simulations and models produce results for large-scale networks in a reasonable time, but trade representation accuracy for speed and hence ignore many physical and system effects, such as interference from the outside world or race conditions inside the nodes. Experimentation provides more representative and precise results, but is limited to small networks. Besides, they require more effort to be deployed and to collect results. These approaches are therefore complementary and should all be involved in the evaluation, which is seldom true, as it requires duplicating the deployment and data collection processes.**

In this demonstration, we present MakeSense, a framework that simplifies these tasks for both simulation and real experiments environments by creating a whole experimentation chain from a single JSON description file. By using MakeSense, it is possible to organize the compilation, to orchestrate the firmware deployment, to efficiently collect results and to plot statistics. We illustrate the ease of use and efficiency of the complete MakeSense workflow over a simple RPL-UDP deployment scenario evaluated with the Cooja simulator and the FIT IoT-Lab open testbed.

I. Introduction

Internet of Things (IoT) is growing interest from both industrial and scientific communities. Potentially, 50 billions of smart objects will be connected in 2020. This means that a significant effort of design thinking must be carried out while developing solutions for constrained Wireless Sensor Networks (WSNs). In particular, one of the major concerns of nowadays applications is how existing protocols scale for large networks.

The design and the validation of efficient protocols can be achieved via both simulators and realistic testbeds. However, both approaches present drawbacks. With simulators it is possible to carry out performance evaluation for a very large number of nodes in a reasonable time frame. On the other side, simulators like Cooja or Tossim, designed for constrained IoT networks, make several assumptions on physical and system aspects for tractability. Real testbeds such as FIT IoT-Lab[2] allow very accurate performance analysis. However, scaling experiments to a large number of devices often require a great effort for firmware deployment and statistic collection.

MakeSense[1] unveils the problems of validating IoT applications by offering a whole chain of experimentation. Through *a single JSON configuration file*, it is possible to compile, deploy, and run an experiment in FIT IoT-Lab or in Cooja. MakeSense also allows analyzing results by collecting and parsing the log files, to plot the results, and to generate HTML reports. Each of these steps can be executed separately or take part in a batch sequence.

This demonstration illustrates the ease of use and capacity of MakeSense over a simple application. The whole description of the experimental setup fits in a single JSON file. We present the syntax of this file and apply slight modification to use alternatively the Cooja simulator and the FIT IoT-Lab open experimental platform. The JSON file also contains directives on which parameters to measure, such as the energy consumption or routing overhead, on how to collect data and on how to present results.

II. MakeSense

MakeSense was introduced in a short paper [1] as a tool to easily organize, run and share WSN simulations. It is distributed under the Apache license through GitHub[1]. The version demonstrated here adds features and supports execution of the experiment using ContikiOS on the FIT IoT-Lab [2].

In MakeSense, the experimenter defines his workflow by specifying a sequence of steps, as illustrated on Figure 1. A single JSON configuration file orchestrates the whole chain, from scenario specification to graphs generation. The experimenter usually first invokes the `make` step to produce the binary files from his source code by typing the command `fab make:dummy` which compiles the source code for an experiment called *dummy*. It can then deploy these binaries on the remote platform using `fab push_iotlab:dummy` or in the Cooja simulator. The remote deployment details are configured within the JSON file that should contain a `nodes` directive that specifies how binaries are named and where they shall be uploaded.

Binaries names and destinations can be specified explicitly or by defining a function, whose syntax use variables and loops, as illustrated in listing 1. This example tells MakeSense to write in the `iotlab.json` JSON file lines to produce 43 firmware binaries named `dummy_1.iotlab-m3`, etc. from a template names `dummy_iotlab.json` and to store these binary files in a directory specified in the `path` variable. These binaries go on the Grenoble IoT Lab platform named `m3-1.grenoble.iot-lab.info`, etc. This function is then invoked in the MakeSense workflow by `fab push_iotlab:dummy`. Using templates ease firmwares management while improving its safety and transparency.

[1] http://github.com/sieben/makesense
[2] https://www.iot-lab.info

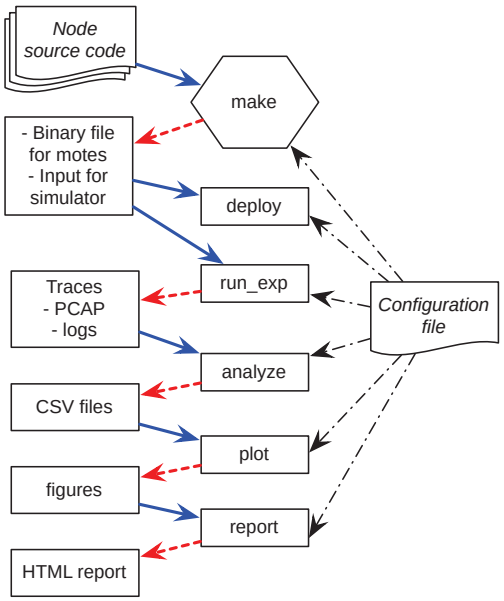


Fig. 1: MakeSense workflow

Listing 1: Configuration file excerpt

```
    def FUNCTION(name):
    [...]
  # Location of a JSON file template to include
  config_template =
         TEMPLATE_ENV.get_template("dummy_iotlab.json")

  # Creation of the array that specifies and associates
  # nodes names and with firmware binaries paths
   res = [
       {"nodes": ["m3-%d.grenoble.iot-lab.info" % num],
      "firmware_path": pj(path, "dummy_%d.iotlab-m3" % num)
      } for num in range(1, 43)]

  # Dump the array in the JSON file
   with open(pj(path, "iotlab.json"), "w") as f:
       f.write(json.dumps(res))
    [...]
```

The experimenter can then execute the experiment with the `run_exp` step. Once the execution is finished, traces and log files are retrieved and analyzed with the `analyze` step, producing CSV files containing the desired metrics. The `plot` step creates graphs that can be inserted automatically in an HTML report with the `report` step.

III. Demonstration scenario overview

The demonstration consists in configuring, building and running an UDP application over a network running the RPL routing protocol. N nodes, ($N \in [10; 40]$) send UDP packets to a single destination, root of the RPL tree during 1 hour. We collect trace files that log the messages that nodes exchange and the RPL periodical control messages.

We show how to run the experiment on the Cooja simulator and on the ARM M3 node made available in the FIT IoT-lab Grenoble testbed. This platform allows monitoring the nodes

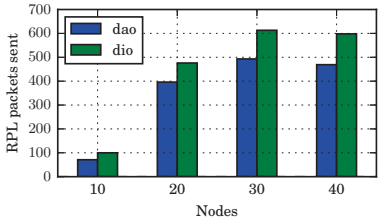


Fig. 2: RPL overhead as a function of the number of nodes.

real energy consumption with an embedded solution whose output can be collected by MakeSense. During experiments execution, different messages are aggregated and analyzed using a MakeSense step.

To illustrate the output of MakeSense, we measure the overhead due to the routing protocol, with the same syntax that was utilized in [1]. Once results are collected they are parsed and analyzed automatically, producing the graph shown in Figure 2, where the network-aggregated RPL overhead is shown as a function of the number of nodes in the network. The step generating the figure is exactly the same as the one used to parse and plot the results of a simulation run.

The last step, *report*, produces an HTML file that presents the results and the JSON configuration file, which can easily be shared. Visitors or collaborators have all the necessary information to re-run the experiment and compare results.

IV. Conclusion

This demonstration shows how to use MakeSense for automating the experimental process, not only in simulation with Cooja, but also using an open remote experimental platform. The heart of the demonstration consists in examining the configuration file and modifying it to change the simulation setup and its output, evaluating a simple RPL-UDP application deployed over FIT-IoT Lab.

MakeSense is currently configured to interact with Cooja and FIT IoT Lab but can easily adapt to other experimental platforms, depending on the existence of certain functionalities such as remote upload of the firmware binaries, remote access and data collection capabilities.

MakeSense was built using Python and uses a few libraries. It can easily be extended and future works will investigate, libraries such as iPython to provide dynamic interaction. iPython allows also the embedding of images, text and functions inside a single notebook along the variables to ease the sharing of an experiment.

V. Acknowledgment

This work is supported by the French National Research Agency (ANR) under grant reference IRIS ANR-11-INFR-0016.

References

[1] Rémy Léone, Jérémie Leguay, Paolo Medagliani, and Claude Chaudet, “Makesense: Managing reproducible wsns experiments,” *Fifth Workshop on Real-World Wireless Sensor Networks*, 2013.

Demo Abstract: BSD-based ECC for the Contiki OS

Oriol Piñol Piñol*, Shahid Raza‡, Joakim Eriksson*‡, Thiemo Voigt‡
*Yanzi Networks AB, Stockholm, Sweden
oriol@yanzi.se
‡SICS Swedish ICT, Stockholm, Sweden
{shahid, joakime, thiemo}@sics.se

***Abstract*—Security has arisen as an important issue for the Internet of Things (IoT). Efficient ways to provide secure communication between devices and sensors is crucial for the IoT devices, which are becoming more and more used and spread in a variety of fields. In this context, Elliptic Curve Cryptography (ECC) is considered as a strong candidate to provide security while being able to be functional in an environment with strong requirements and limitations such as wireless sensor networks (WSN). Furthermore, it is a valid candidate to be used in industry solutions.**

In this demo we show a real use case of Elliptic Curve Cryptography for key establishment in combination with symmetric AES encryption. The demo will show the use of a BSD-licensed ECC library for the Contiki OS running on Yanzi Networks Contiki-based nodes that will securely communicate with a Yanzi Gateway.

I. Introduction

Communication protocols have been twisted to adapt to the constraints that Wireless Sensor Networks (WSN) present and to improve the efficiency of devices in this kind of networks [1] [2]. Even special OS, such as the Contiki OS [3], have been designed especially to adapt to this paradigm [4]. Security is a big issue in this new environment. Privacy, identity management, security and access control are an important challenge [5] and also the issue of how to assign credentials to distinguish access to certain information by the different machines arises. Protocols to establish security in the communications between devices such as those belonging to WSN have been implemented focusing on power efficiency and enabling communication with other devices over the Internet [1]. In this paradigm, public-key cryptography has been shown to be a viable alternative [6]. For many applications in this area, energy is considered the main restriction but public key cryptography has proven to fulfil these requirements and to perform well in such restrained environments.

The protocols and applications used to provide secure communication between devices nowadays are implemented for machines that are not limited by their power or memory in a drastic way. On the other hand, the Internet of Things is mainly composed of devices that have important power restrictions. Therefore, power-efficient, lightweight protocols need to be designed. Since security is such an important issue the security protocols need to be also adapted to this paradigm. Nowadays public key cryptography with TLS and RSA are the de-facto standards in secure communications but they do not fit the aforementioned need. Protocols such as Datagram Transport Layer Security (DTLS) have been adapted to approach this problem, but on the application level security, Elliptic Curve Cryptography (ECC) shows up as an alternative to RSA. Elliptic curves achieve the same level of security as RSA algorithms with less number of bits, implying faster, and less power-consuming schemes that provide the same security [7].

Fig. 1. Yanzi Networks system. The nodes use a STM32W108 system-on-chip and run the Contiki OS.

We implement the first BSD license ECC implementation for the Contiki OS, with the objective of making this cryptography scheme available for use in research and industry and take advantage of the benefits that ECC offers in the context of a versatile implementation that allows the use of different Elliptic Curves and an easily configurable security-level thanks to the parameter configuration such as the key size.

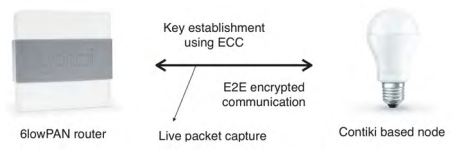


Fig. 2. Demonstration setup consisting of a Yanzi Gateway and a Yanzi LED that run a key exchange using ECDH and communicate using symmetric encryption using the derived secret. To visualise the demonstration a live Wireshark capture of the communication will be performed

II. Demonstration Description

The purpose of the demonstration is to show a functional system using Elliptic Curve Cryptography for key establishment in a low-power Wireless Sensor Network using a lightweight ECC implementation for the Contiki OS.

The setup consists of nodes from Yanzi Networks AB using STM32W108 system-on-chip based on a 32-bits ARM®Cortex™-M3 processor with 16 kb of RAM and 256 kb of embedded Flash memory runing Contiki OS. The nodes establish a secure communication using ECC with a Yanzi Gateway based on Linux and then communicate using 128 bits AES symmetric encryption.

The demonstration scenario is the following. First, when the gateway is discovered by a new node, in this case a Yanzi LED, the gateway and the device exchange their public 256-bits ECC public keys following a Elliptic Curve Diffie-Hellman (ECDH) [8] key exchange scheme. From the combination of the other party's public key and their own private key and, as a result of the ECDH exchange, a shared secret is obtained by both parties. From that shared secret both parties derive a session key to feed the AES ciphers and start a secure communication using symmetric encryption.

To visualise the explained procedures, a live Wireshark capture of the process will be shown. This will allow to observe the public exchange of keys corresponding to the ECDH exchange and the subsequent communication using the key derived from the exchange for the communication between the gateway and the lamp secured using 128-bits AES symmetric encryption.

III. Conclusion and Future Work

Elliptic Curve Cryptography is a noteworthy alternative as a public-key cryptography scheme for the WSN. We show that it is feasible to use this scheme in sensor networks and that it provides a strong security scheme for establishing secure communications between devices. Furthermore, we present with this demonstration that our solution is already in industry as a fully working solution for key establishment.

To add flexibility and usability of the ECC, out-of-the-box integration with established security standards such as the Internet Key Exchange protocol (IKE) or Datagram Transport Layer Security (DTLS) are a possible next step. These implementations, provided with BSD licenses such as the implemented ECC library, would help to increase the use of this security scheme with all its benefits. We also plan to compare our ECC implementation with other ECC implementations for constrained devices such as TinyECC and MoTE-ECC.

Acknowledgment

This research has been funded by VINNOVA.

References

[1] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight Secure CoAP for the Internet of Things," *IEEE Sensors Journal*, vol. 13, no. 10, 2013.

[2] C. Bormann, A. Castellani, and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," *Internet Computing, IEEE*, vol. 16, no. 2, pp. 62–67, March 2012.

[3] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, Nov 2004, pp. 455–462.

[4] M. Kovatsch, S. Duquennoy, and A. Dunkels, "A Low-Power CoAP for Contiki," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, Oct 2011, pp. 855–860.

[5] L. Coetzee and J. Eksteen, "The internet of things - promise for the future? an introduction," in *IST-Africa Conference Proceedings, 2011*, May 2011, pp. 1–9.

[6] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based On-line Energy Estimation for Sensor Nodes," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, ser. EmNets '07. New York, NY, USA: ACM, 2007, pp. 28–32. [Online]. Available: http://doi.acm.org/10.1145/1278972.1278979

[7] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, March 2005, pp. 324–328.

[8] V. S. Miller, "Use of elliptic curves in cryptography," in *Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85*. New York, NY, USA: Springer-Verlag New York, Inc., 1986, pp. 417–426. [Online]. Available: http://dl.acm.org/citation.cfm?id=18262.25413

Demo Abstract: Experiments On Using Vehicles As Data Mules For Data Collection From Urban Sensors

Pedro Santos*, Tânia Calçada*, André Sá*, Diogo Guimarães*, Tiago Condeixa†, Carlos Penichet*, Susana Sargento†‡, Ana Aguiar* and João Barros*†
Instituto de Telecomunicações, *Faculdade de Engenharia da Universidade do Porto, †VENIAM,
‡Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Email: {pedro.salgueiro,tcalcada,ee12256,ee10158}@fe.up.pt,
tcondeixa@veniamworks.com,cpp@fe.up.pt,susana@ua.pt,{anaa,jbarros}@fe.up.pt

***Abstract*—In the UrbanSense platform developed in the Future Cities Project, sensing devices based on the open-source Raspberry Pi platform were sparsely distributed across the city to measure environmental parameters. The data gathered by the sensors needs to be transferred to the cloud, preferably taking advantage of the existent local infrastructures. This work presents a proof-of-concept of a system that will use Porto's vehicular network, composed by over 600 vehicles, to transfer sensor samples from the sensing devices of the UrbanSense platform to a cloud server, in a Delay Tolerant Networking (DTN) fashion. During the demo, we will show data being collected by a moving vehicle and later delivered to a server in the cloud.**

I. Introduction

In recent years, pervasive monitoring became possible by the development of small sensing devices with communication capabilities. One of the major challenges in such large-scale sensing platforms is to perform low-cost aggregation of the collected data at a central database. Solutions such as cellular networks or physical wiring to all sensors may be impractical or costly when compared with data benefits. A purposely-built M2M solution ([1], [2]) would require considerable investment, whereas our implementation takes advantage of an already-deployed, cost-free platform.

An low-cost alternative is provided by vehicular networks. With the Future Cities project [3], the vehicles of Porto's public transportation and municipal services systems have become equipped with wireless communication technologies, creating an operational vehicular network. This set of vehicles covers a significant area of the city on a daily basis, creating numerous opportunities to collect the data acquired by sensors and deliver it to a cloud server. Such strategy is known as *Data muling*, a case of Delay Tolerant Networking (DTN) [4].

The UrbanSense platform proposes to bring together these two realities, pervasive urban sensing and data muling, in order to support large scale data gathering from fixed urban sensors and aggregation at a cloud-based server. A number of processing devices equipped with environmental sensors and WiFi interface are being deployed at several strategic locations of the city of Porto. Their data is collected at a cloud-based database server. Fig. 1 shows one of such units that was deployed at the margin of an important artery of the city where buses of the vehicular network pass by regularly. This demo shows a proof-of-concept scenario that includes one sensor, one mule and one infra-structured node that collects data from the mule and delivers it to the cloud database.

This work was partially funded by three research projects: SenseBusNet (PEst-OE/EEI/LA0008/2013), I-City for Future Mobility (NORTE-07-0124-FEDER-000064), and FP7 - Future Cities (FP7-REGPOT-2012-2013-1). The authors would like to thank the Municipality of Porto for the logistic support, and Porto Digital for providing fiber connection to the RSU.

Fig. 1. Sensing device deployed in R. Damião de Góis, Porto, Portugal. UrbanSense sensing devices are processing devices (Raspberry Pi) equipped with environmental sensors (wind direction and speed, rain, solar radiation, luminosity, humidity, temperature, noise) and a IEEE 802.11b/g/n interface.

II. Data Collection Architecture

There are three main elements in this sensing and communication system: (i) the sensing devices, (ii) the vehicular network, and (iii) the cloud-based server. Fig. 2 presents the elements of the architecture, and the data and control messages exchanged between them.

The sensing devices are processing devices (Raspberry Pi) equipped with environmental sensors and a IEEE 802.11b/g/n

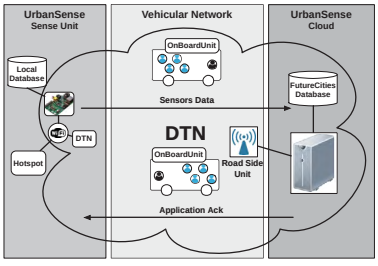


Fig. 2. Global perspective of the architecture. Data is gathered and locally stored by the sensing devices, transmitted through the vehicular network, and delivered to a server in the cloud. The acknowledgements are generated by the server and transmitted through the vehicular network to the sensing devices, so they can delete the local data already received in the server.

Fig. 3. Path travelled for measurements.

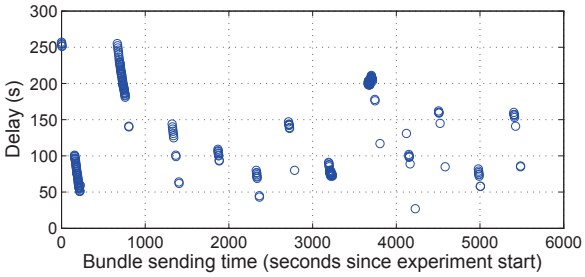


Fig. 4. End-to-end delay measurements.

interface, which are encapsulated in hermetic casing and deployed at the sites of interest. They have a local database that stores the data collected by the sensors.

The vehicular network is composed of two types of elements: mobile and static nodes. The mobile nodes, in the form of buses and garbage-collection trucks, are equipped with a communicating device called On-Board Unit (OBU) that has IEEE 802.11b/g/n and 802.11p interfaces [5], providing the vehicles with wireless networking capabilities. These mobile units communicate with the sensing devices via IEEE 802.11b/g/n interface (WiFi). The static nodes of the vehicular network are infrastructure devices called Road Side Units (RSU). These equipments have high-speed connection to the Internet and communicate with the mobile nodes via the IEEE 802.11p interface. They bridge the communication between the vehicular network and the cloud-based UrbanSense server.

The cloud UrbanSense database server receives, acknowledges and stores the data collected by all sensing devices.

III. Proof-of-Concept

A. Implementation

This architecture was tested in a real-world scenario. For this purpose, we had to create software modules at the end-points (the sensing devices and the cloud-based server) to manage the end-to-end communication, and integrate it with a software implementation that provides support for DTN communication over all elements of the architecture.

Regarding the DTN requirements of this architecture, we chose the *bundle protocol* specified in RFC 5050 [6]. The bundle protocol defines a number of services and primitives tailored specifically to handle the opportunistic nature and long delays that are inherent to Delay Tolerant Networks. The protocol was designed as a layer in the network stack that sits between the network and application layer, the *bundle layer*. The datagrams exchanged at this layer are named *bundles*. We used a specific implementation of the bundle protocol, the open source implementation IBR-DTN [7]. The IBR-DTN software package was installed in all elements of the architecture: sensing devices, OBUs, RSUs and server.

The communication module at the sensing device constantly searches for opportunistic connections to passing OBUs. If detected, it fetches sensor data from the local database and stores it into a bundle, which then delivers to the bundle layer. The information about which data was sent is locally stored. Upon reception of an acknowledgement of the previously sent data, the sensing device deletes that data from the local database. The communication module at the cloud-based server is in charge of receiving and processing the bundles received from the bundle layer, storing them in the global database, and sends acknowledgements to the sensing device that originated the data via the bundle layer.

B. Tests

The proof-of-concept tests were conducted using a sensing device installed at Rua Damião de Góis and a personal vehicle, that was fitted with one OBU. The vehicle performed an itinerary that involved passing by the sensing device, giving the opportunity to establish a connection to the OBU, and continued to a nearby RSU, located at Praça do Marquês de Pombal. This path is depicted in Fig. 3.

This itinerary was made multiple times, over a period of time of 5565 seconds (apr. 93 minutes). There were 11 contacts between the sensing device and the OBU. In total, 250 bundles were transferred, corresponding to 145.85 kilobytes of sensor data. All data bundles successfully arrived to the cloud server.

Fig. 4 shows the delay experienced while transmitting the bundles during the test. The minimum, mean and maximum bundle delays were 27 s, 140 s and 257 s, respectively. This shows that the roads traffic conditions are determinant to bundles delay. All data samples created during the tests were transmitted at the first transmission attempt.

IV. Conclusions

This work presents an architecture to use vehicles as mules for data gathered by environmental sensors deployed in the city. Our proof-of-concept has shown the reliability of the system and how delay is dependent on road traffic.

References

[1] SIGFOX, http://www.sigfox.com.
[2] M2M Spectrum Networks, http://www.m2mspectrum.com.
[3] "Future Cities Project," http://futurecities.up.pt/, 2014.
[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgeson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture. RFC 4838 (Experimental)," April 2007.
[5] C. Ameixieira, J. Matos, R. Moreira, A. Cardote, A. Oliveira, and S. Sargento, "An IEEE 802.11p/WAVE implementation with synchronous channel switching for seamless dual-channel access (poster)," in *Vehicular Networking Conference (VNC), 2011 IEEE*, Nov 2011, pp. 214–221.
[6] K. Scott and S. Burleigh, "Bundle Protocol Specification. RFC 5050 (Experimental)," November 2007.
[7] M. Doering, S. Lahde, J. Morgenroth, and L. Wolf, "IBR-DTN: an efficient implementation for embedded systems," in *Proceedings of the 3rd ACM Workshop on Challenged Networks*. ACM, 2008, pp. 117–120.

Demo Abstract: ProFuN TG: A Tool Using Abstract Task Graphs to Facilitate the Development, Deployment and Maintenance of Wireless Sensor Network Applications

Atis Elsts, Farshid Hassani Bijarbooneh, Martin Jacobsson, and Konstantinos Sagonas
Uppsala University, Sweden

***Abstract*—In this demo abstract we present ProFuN TG (Task Graph), a tool for sensor network application development using the data-flow programming paradigm. The tool has support for the whole lifecycle of WSN application: from the initial design of its task graph, task placement on network nodes, execution in a simulated environment, deployment on real hardware, to its automated maintenance through task remapping. ProFuN TG allows to program applications that incorporate quality-of-service requirements, expressed through constraints on task-to-task data flows.**

I. Introduction

Programming wireless sensor network applications is difficult, especially if certain reliability and data quality properties are desired together with energy efficiency.

We take an existing WSN programming methodology, the Abstract Task Graph [1], and implement it in ProFuN Task Graph [1] , a tool that facilitates the development of such applications. ProFuN TG not only allows the user to describe the functionality of an application with a task graph, but also to incorporate non-functional requirements [2] in that description. The requirements are expressed in form of probabilistic *constraints* on minimal packet delivery rate (PDR) and delivery delay, and set on data flows between tasks. The tool allows users both to use predefined tasks from a palette and to write their own tasks in C or SEAL programming languages.

The tool includes support for mapping these task graphs on network nodes, for macrocompilation of their code, and for their deployment both in simulated and real networks. The supporting run-time middleware gathers application performance statistics and determines whether the conditions of the constraints hold, enabling maintenance alert notifications, as well as automated maintenance through task remapping.

II. Architecture

Under the hood, ProFuN TG uses a number of well-known software tools and libraries: Contiki for system-level functionality, Cooja for network simulation, Gecode for constraint solving (used in the task allocation algorithm). For the visual frontend, an adapted version of Node-RED is employed.

ProFuN TG joins these components in a distributed microservice architecture (Fig. 1). The components communicate by passing JSON messages through HTTP, with the exception of WSN middleware, which uses an efficient binary format.

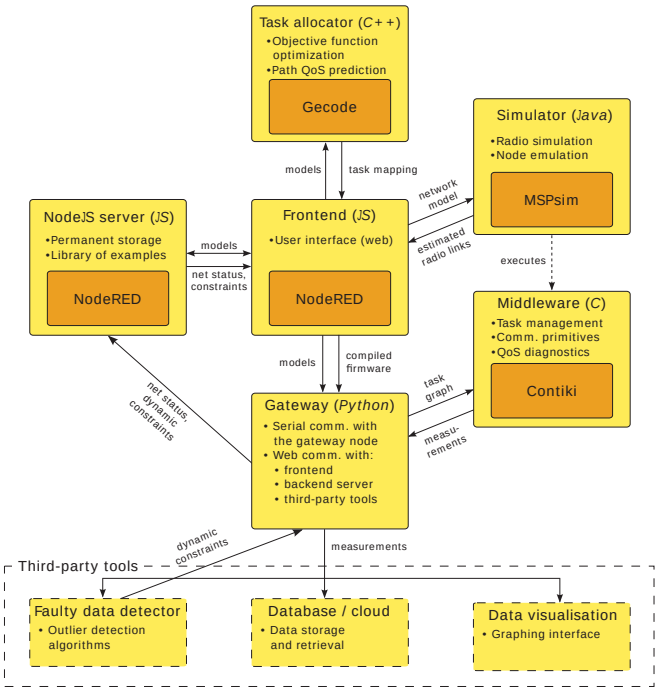


Fig. 1: Architectural overview

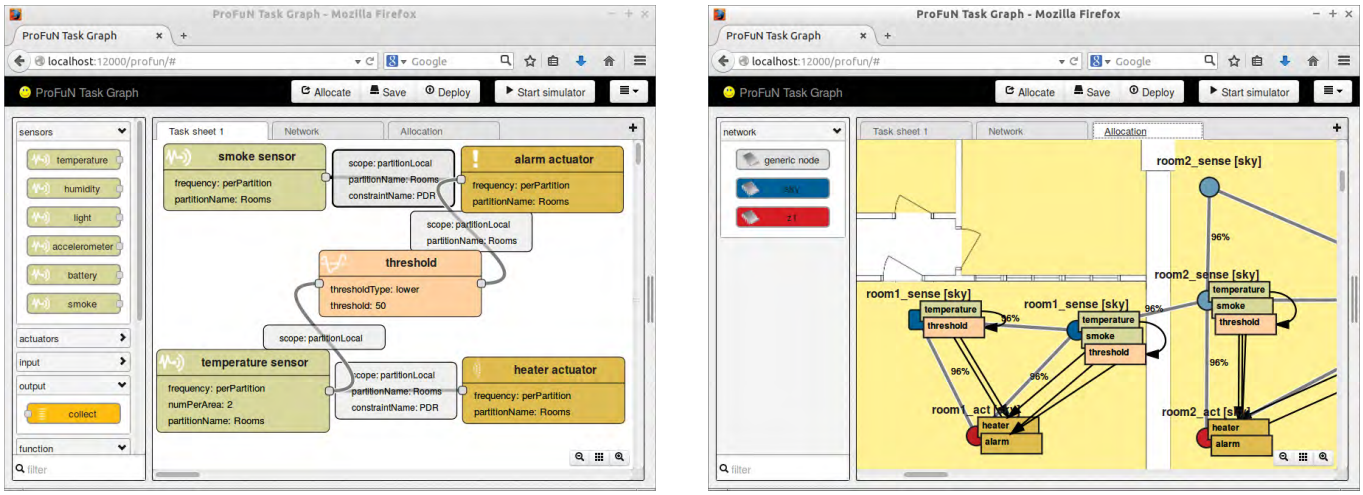
The tool provides an HTTP interface for data export in JSON format. Through it, custom or third-party tools can access the data, provide feedback for ProFuN TG , and impose dynamic constraints on the task mapping algorithm. In our demo setup we are going to use a custom principal component analysis (PCA) tool that detects sensor faults by learning acceptable sensor covariance bounds from past datasets.

ProFuN TG task sheet view (Fig. 2a) shows the task graph of a sample application. Network view (Fig. 2b) shows node placement in the network and tasks mapped on the nodes.

III. Application example

To explain the typical user workflow with the tool, let us take an example application that uses temperature sensors and

[1]http://parapluu.github.io/profun/



(a) Task graph view. Shows sensor, actuator, and processing tasks connected with data flows. On some of flows, PDR constraints are configured

(b) Network view. Shows sensing and actuating nodes (as blue and red circles, respectively) connected with network links (grey lines), and mapped tasks (rectangles) connected with data flows (black arrows)

Fig. 2: The visual interface of ProFuN TG showing a heater control application with fire detection

heater actuators to a maintain stable temperature in a number of rooms. The tool allows the user to:

- Create a task graph model for the application. The model consists of two tasks connected with a data flow.
- Set a constraint for minimally required PDR on the flow.
- Describe the model of the network: node locations, capabilities, and links between nodes. Probabilistic parameters such as link delay are described by probability distributions. In absence of explicit configuration, link existence and quality parameters are estimated by the simulator.
- Partition the network into regions (rooms) and configure task allocation frequency: one-pair-per-room.

Subsequently, the tool:

- Maps the tasks on network nodes, taking into account the network model, with the objective to minimize energy usage and satisfy the PDR constraints.
- If desired, simulates the setup to see if the constraints hold in the simulation environment.
- Deploys the task graph on a real WSN.
- Continuously tests for satisfaction of the constraints and requests task remapping when the test fails.

IV. Demo setup

We plan to demonstrate a fault-tolerant light-sensing application developed with the tool.

The setup is going to consist of a laptop and a number of sensor nodes equipped with light sensors. On the laptop, ProFuN TG will be running, and the measured light intensity will be displayed. Light sensing and data collection tasks will be activated on the nodes.

Interested attendees are going to be invited to try to "break" the application by covering some of the light sensors and turning off some of the nodes, while the system is expected to demonstrate robustness by ignoring readings of the affected sensors and reallocating tasks to a different set of nodes.

V. Concluding remarks

ProFuN TG enables design of data quality requirement-aware task graph applications by allowing the user to write PDR and delay constraints on data flows between tasks. The tool also enables deployment and maintenance of these applications by providing middleware that checks for faults at runtime and triggers reallocation in case a violation is detected.

A major difficulty for the tool to provide quality-of-service guarantees is caused by the fact that the inherent unreliability of wireless communications makes it hard to predict the performance of an application before actually deploying it. To make the runtime system more predictable, advanced link-layer protocols such as Glossy [3] should be used instead of the current Contiki network stack.

Acknowledgments

The authors acknowledge support from SSF, the Swedish Foundation for Strategic Research.

References

[1] A. Bakshi, V. K. Prasanna, J. Reich, and D. Larner, "The abstract task graph: a methodology for architecture-independent programming of networked sensor systems," in *Proceedings of the 2005 workshop on End-to-end, sense-and-respond systems, applications and services*. USENIX Association, 2005, pp. 19–24.

[2] F. H. Bijarbooneh, A. Pathak, J. Pearson, V. Issarny, and B. Jonsson, "A constraint programming approach for managing end-to-end requirements in sensor network macroprogramming." in *SENSORNETS*, 2014, pp. 28–40.

[3] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE, 2011, pp. 73–84.

Demo abstract: Radio Interferometric-Based Indoor Tracking

Gergely Zachár, Gyula Simon
University of Pannonia, Veszprém, Hungary
e-mail: {zachar, simon}dcs.uni-pannon.hu

***Abstract*—In this demo, a radio interferometry based indoor object tracking will be demonstrated. The system consists of several infrastructure nodes in fixed and known positions and the tracked objects are equipped with the same type of devices. The tracking method does not require special hardware components, only the onboard radio chips. The experiment is performed on low-end devices utilizing distributed phase difference calculation and the real-time tracking results can be seen on the host computer.**

I. Introduction

Nowadays tracking and localization services are important building blocks of several application e.g. surveillance, supply chain management or robot control. While outdoors several well-known and widely used systems exists (e.g. GPS), indoors a large variety of competitive methods try to offer reliable positioning and tracking. Some of the possible solutions utilize radio interferometry, which do not require special or expensive hardware elements, but use only the onboard radio modules of the sensor nodes. The accuracy of the radio interferometric localization and tracking can be in the range of a few centimeters.

In this real-time application an indoor radio interferometric object tracking will be demonstrated, based on the Radio Interferometric Object Trajectory Estimation [1].

II. Radio Interferometric Measurement

The basic radio interferometric measurement was designed to provide a low cost alternative for localization and tracking in sensor networks [2], utilizing the inexpensive onboard radio modules on the nodes. Instead of high frequency signal processing, radio interferometry performs low-frequency processing of the interference signal. The operation of the basic radio interferometric measurement can be seen in Fig. 1. Two devices (A and B) are transmitting carrier waves at almost the same frequency ($f_A \approx f_B \approx f$), thus generating an interference signal with a low-frequency envelope, where the beat frequency is $\Delta f = |f_A - f_B|$. This low-frequency beat signal is the actual receive signal strength (RSS) which can be measured with most RF receivers.

By recording the RSS signals, detected by the two receivers (C and D), the phase difference ϑ between the two signals can be measured.

This research was supported by the Hungarian Government and the European Union and co-financed by the European Social Fund under project TÁMOP-4.2.2.C-11/1/KONV-2012-0004.

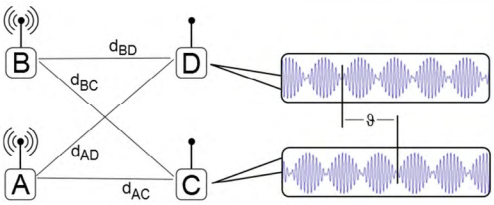


Fig. 1. Radio interferometric phase measurement

Using the measured phase difference ϑ, the following relationship holds for the distances between the four nodes [2]:

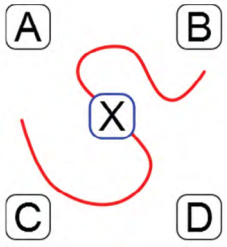
$$\vartheta(f) = 2\pi \frac{d_{AD} - d_{BD} + d_{BC} - d_{AC}}{c/f} \pmod{2\pi} \qquad (1)$$

where c is the speed of light, and the distance notations are shown in Fig. 1. Note that if the position of three nodes is known then in (1) only two unknown ranges remain. Also notice the modulus operator in (1), resulting in a phase ambiguity problem. In order to calculate the unknown ranges multiple measurements are necessary [1], [2].

III. Radio Interferometric Tracking

A. Infrastructure

In the demonstration one moving node will be tracked, and fixed infrastructure nodes at known locations will be utilized to perform the measurements. To provide redundant measurements, in the demo four infrastructure nodes will be utilized, each of which can be either receiver or transmitter. In each measurement configurations three of the infrastructure nodes will be active (two transmitters and one receiver), and the tracked device will play the role of a receiver. Using four infrastructure nodes, 12 possible configurations are possible, as shown in Fig.2. The operation of the measurement is illustrated in Fig. 3. In step (a) the base station broadcasts a synchronization and measurement configuration command. Based upon the received configuration command, in step (b)



Config	A	B	C	D	X
C1	T	T	R		R
C2	T	R	T		R
C3	T	R		T	R
C4	R	T	T		R
C5	R	T		T	R
C6	R		T	T	R
C7	T	T		R	R
...					
C12		R	T	T	R

Fig. 2. The 12 possible configurations in the demo infrastructure, where four infrastructure nodes (A, B, C, D) and one tracked node (X) are utilized.

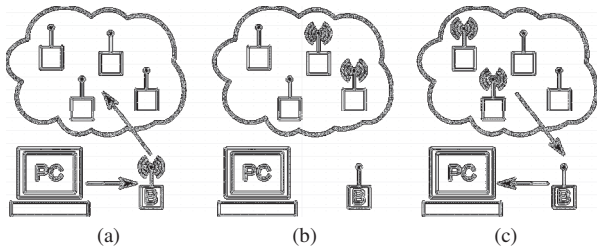


Fig. 3. Measurement steps for measuring phase differences: (a) broadcast of measurement and synchronization command, (b) data collection and phase estimation, (c) transmission of measurement results

nodes start operating in their designated roles as transmitters or receivers, and perform synchronized phase measurements. The results are transmitted back to the base station in step (c).

B. *Object tracking*

For sake of simplicity the method is discussed in 2D. In each measurement round (i.e. measurements in the 12 possible configurations) the phase differences are collected and a confidence map is created over the possible locations of the tracked object [1]. A confidence map is shown in Fig. 4, where blue peaks show locations where the tracked object can be with high confidence, and red areas shows unlikely locations. Due to the phase ambiguity problem of (1), the confidence map contains multiple possible solutions: one solution corresponds to the true location, the other solutions are phantoms. As the tracked object moves, the peaks on the confidence map shift, following the movement of the object. The true solution follows the object along its trajectory, while the phantoms eventually disappear. Thus the method can be used in two operating modes: (i) after sufficiently long track record the full trajectory can be determined (retrospectively), or (ii) the track can be followed, if the initial position is known.

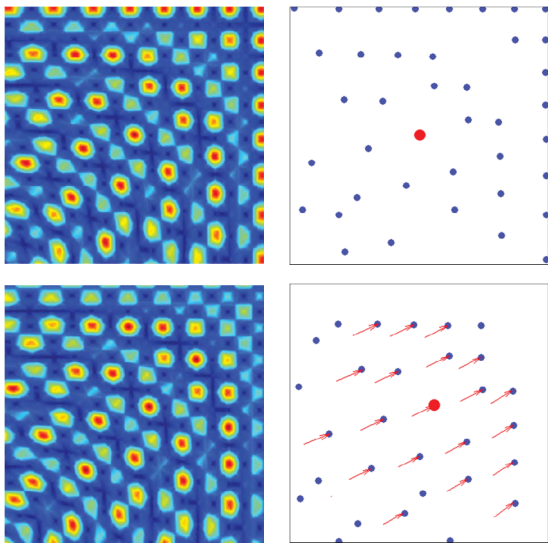


Fig. 4. Calculated confidence maps (left) in two positions along the trajectory of the tracked object. The shift in the confidence map can be seen, as the object moved about 250 millimeters. Red and blue colors in the maps correspond to high and low confidence values, respectively. The peak positions, identifying possible locations are shown in the right hand side, arrows showing the direction and size of the shift between the two positions. Red and blue dots identify true and phantom positions, respectively.

IV. Deployment

A. *Hardware*

The node used in the demonstration can be seen in Fig. 5. These special dual-radio nodes are based on Atmel's ATmega128RFA1 microcontroller and transceiver. The RFA1 radio is used to transmit measurement results, start commands, and setup messages. The node also features another radio chip (Silicon Labs Si4432), which has the ability of fine tuning the transmission frequency. This feature makes the chip suitable for radio interference based tracking, where the two transmission frequencies should nearly be the same. This radio is used to perform the interferometric measurements in the 868MHz ISM frequency band. Before the measurements transmission frequencies are calibrated on one of the devices for each transmitter pairs. RSSI values were recorded with sampling frequency of 62.5kHz. Synchronized receivers perform frequency and phase difference measurements, using the stored RSSI data with length of approximately 8ms.

B. *Demo scenario*

In the demo application the four infrastructure devices are placed on tripods, approx. 1.5m above ground, while the tracked device can be carried in hand. The movement of the node should be performed in the area bounded by the four infrastructure nodes for proper tracking.

The phase measurements are controlled by the host computer utilizing a base node as a bridge device, as shown in Fig 3. The real-time tracking results with the confidence maps can be seen on the laptop screen.

V. Conclusion

A sensor fusion algorithm for radio interferometric tracking was demonstrated. The algorithm utilizes redundant phase measurements to estimate the position of the tracked device. The method is able the track objects with known initial positions in real time, or can determine the full track of an object retrospectively. As opposed to earlier solutions [2], the phase ambiguity problem is resolved by redundant and sequential measurements, allowing real-time tracking. The accuracy of the tracking method is in the range of 5-15 centimeters.

References

[1] G. Zachár, G. Simon, „Radio-interferometric Object Trajectory Estimation", in Proc. 3rd International Conference on Sensor Networks, SENSORNETS 2014, Lisbon, Portugal, Jan. 7-9, 2014, pp. 268-273

[2] M. Maroti et al, "Radio Interferometric Geolocation", ACM Third International Conference on Embedded Networked Sensor Systems (SenSys 05), San Diego, CA, pp. 1--12, November, 2005.

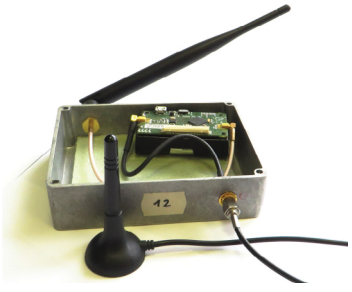


Fig. 5. A measurement device with dual-radios used in the demonstration

Demo Abstract: Towards the Development of XDense, A Sensor Network for Dense Sensing

João Loureiro, Raghuraman Rangarajan, Eduardo Tovar
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal
Email: {joflo, raghu, emt}@isep.ipp.pt

***Abstract*—Many applications require deployments of thousands of sensors per square meter with high sampling rates to observe specific phenomena of interest. We proposed XDense as a wired mesh grid sensor network architecture tailored for scenarios that can benefit from such a deployment. We evaluated and validated the XDense model together with its application specific potentials through extensive simulations. In this paper, we discuss the practical implementation issues of XDense and the steps for its experimental validation. Using an emulation setup, we demonstrate and evaluate XDense potentials practically, and show the results using a supervisory system.**

I. Introduction

In order to observe a phenomena of interest, numerous applications rely on extremely dense deployments of sensor networks (SN) to achieve very high spatial resolution, likewise high temporal resolutions is also desirable. Such sensors can be of different natures, and some applications may require deployments as dense as hundreds to thousands of units per square meter. For example, artificial skins for robotics, biomedical devices, such as electroencephalographs and brain implantable retinal prosthesis. Fluid dynamics is another potential application field which has tight spatial and temporal sensing requirements, specially if closed control loops are desirable, for example, for active flow control on aircrafts [1]. For such dense deployments, sensor network technology faces scalability issues in some key aspects as such as cost, communication time, interconnectivity, processing time, power, and reliability.

We proposed XDense to alleviate these issues [2]. XDense has a wired mesh grid network of sensors and actuators that resembles Network-on-Chip topologies and is tailored to address the challenges of extremely dense sensor deployments. It enables efficient data extraction of observed phenomena, without the need of collecting the data from each individual node centrally. It allows the user to program each node, to exploit their computational power, with distributed algorithms, in order to decrease communication load and response time. The XDense architecture is presented in Figure 1(a), with its details summarized underneath.

Similar deployments are found in literature [3], but authors utilize different approaches for communication, such as shared medium or sensors multiplexing, limiting scalability and distributed processing opportunities.

We first conceptualized XDense and present preliminary simulation results in [4]. More recently, extensive simulation results were performed and presented in [2]. We shown two different algorithms for efficient data extraction of a network of 101×101 sensors and one sink. Both results provided greater understanding of the network principle of operation, and evaluated, with simulations, the potential gains offered.

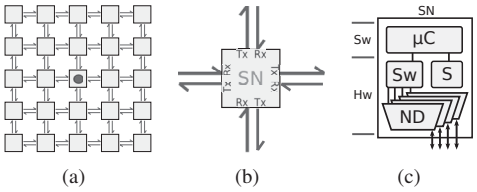


Fig. 1. Overview of XDense architecture (a): An example 5×5 network with one sink in the center; (b) Node pinout: one full-duplex port in each direction; (c) Node's model architecture: a software/application layer (μC), and the hardware layer, which includes the switch (Sw), the net-device (ND) and the sensor (S)

II. Systems Requirements

Although we have developed extensive simulations, a hardware solution is desirable to validate and consolidate our model's feasibility in real application scenarios. In this sense, the hardware platform must fulfill some basic performance and cost requirements: 1) Simplicity: Nodes should be simple in terms of hardware and software, for enabling cost effective deployments with miniaturized nodes. 2) Performance: The controller should allow fast enough communication links and processing with respect to the application requirements. 3) Communication ports: Our sensor node requires at least four serial ports, one for each direction. An extra port is desirable for debugging proposes, or for an external link to a supervisory system; 4) Sensors/actuators: The attachment of any kind of sensing or actuation should be possible through the available interfaces. 5) Power consumption: Node's should be energy efficient due to the density of deployment, and its impact on the overall system power consumption.

Usually, for realising the above, a custom design integrated circuit (IC) obviously provides the best-fit solution. But, this reduces design flexibility and might become a single-application solution. It is desirable to have a hardware platform and software framework that is flexible enough to allow customization of applications to a variety of scenarios. Moreover, the overhead of development and initial costs of production of a custom design IC is prohibitive.

III. Systems building

Keeping these requirements in mind, and other practical aspects, we use COTS to prototype XDense. With the require-

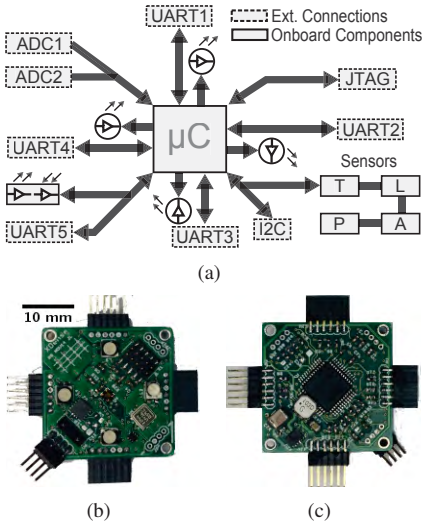


Fig. 2. XDense sensor node schematic and prototype. Figure (a), the node's schematic shows each of major components of the system. Dotted lines are actually headers for external connections. Others are on-board components, like the LED's, the IrDA transceiver and the temperature (T), pressure (P), light (L) and acceleration (A) sensors. (b) and (c) shows the top and bottom side of the PCB respectively.

ments stated in the previous section, we are also restricted to a limited number of candidate μCs, especially concerning the number of high-speed serial ports available. Availability of Direct Memory Access (DMA) is also desirable, to reduce processing time spent on communication.

Our choice, the Atmel IC ATSAM4N8A, is based on the 32-bit ARM Cortex-M4 RISC processor, a mid-range general purpose μC. It runs at up to 100 MHz and has a good balance between power consumption and processing power. It has a small 48-pin footprint, with five high speed serial ports (one USART and four UART ports), and 23 DMA channels that allows efficient communication and sensor reading. With a digital signal processing extensions (DSP) and floating point unit (FPU) co-processor, the IC chosen has enough computational power to allow us to demonstrate the benefits of distributed processing.

The prototype node is shown in Figure 2 with the detailed explanation of each of the main components seen in the board. We placed four different sensors on the top of the board for sensing: a 3-axis accelerometer (A), a pressure sensor (P), a temperature (T), and a visual-range light sensor (L). Along with this, for actuation, we have four RGB LEDs to transduce sensed values to colors, and represent any kind of distributed actuation for debugging proposes.

Given the above requirements and design considerations, constructing an XDense node is an iterative process involving two distinct steps [5]: 1) system conception, manufacturing and assembly, and 2) initial software setup and device driver development. The details of the steps are given below.

1) System conception, manufacturing and assembly Circuit schematic definition and pin-to-peripheral mapping; Placement of the components and routing of the printed circuit board (PCB); PCB manufacturing and testing; Power supply assembly and tests; Assembly of the remaining components and tests; 2) Initial software setup, and device driver development Setup the initialization routines of the μC and programming/debugging interfaces; Development and operationalization the drivers of internal and external peripherals; Development of unit tests to each peripheral; Implementation of a bootloader, to allow in-network programming.

We then started porting our protocol model, previously developed with the ns-3 simulator, to our μC. Some abstractions of the model, such as the net-device and sensor components, are translated into device drivers that interact with the actual hardware peripherals. The switch and packet processor are implemented in software as part of the communication stack. Algorithms for data processing and feature detection and extraction are also implemented in software, but in a modular fashion, to allow easy re-programming.

The FreeRTOS real-time operating system is used. It provides drivers for asynchronous communication, and some additional abstractions desired such as multi tasking, and tools for timing analysis and predictability.

IV. Current status and next steps

Before performing real-life experiments, a basic testbed is required to validate our assumptions using a restricted application scenario. For this purpose, we have built a simple 3×3 XDense network. Our first experiment at analyzing this architecture will be to use the light sensor to demonstrate distributed data processing capabilities. The results will be compared with our previous simulation results in [2]. We project controlled light source on the deployment using a video projector. This can be artificial computer generated data or representational data from a natural phenomena.

This input will involve gradients and geometric shapes to test distributed data processing algorithms for data aggregation and edge detection. Comparison of input data with data acquired will allow us to comment on granularity of results, data consistency and network performance for different scenarios. This results will be shown using a supervisory user interface running on a PC, wirelessly connected to the sink. With this demonstration, we will be able to validate and iterate through our implementation. This feasibility study will allow us to then proceed with testing XDense in some of the real-life application scenarios we have talked about [2].

References

[1] N. Kasagi, Y. Suzuki, and K. Fukagata, "Microelectromechanical systems-based feedback control of turbulence for skin friction reduction," *Annual review of fluid mechanics*, vol. 41, pp. 231–251, 2009.

[2] J. Loureiro, R. Rangarajan, E. Tovar, and N. Pereira, "Xdense: A dense grid sensor network for distributed feature extraction (submitted, under revision)."

[3] J. A. Paradiso, J. Lifton, and M. Broxton, "Sensate mediamultimodal electronic skins as dense sensor networks," *BT Technology Journal*, vol. 22, no. 4, pp. 32–44, 2004.

[4] J. Loureiro, V. Gupta, N. Pereira, E. Tovar, and R. Rangarajan, "Xdense: A sensor network for extreme dense sensing."

[5] P. Marwedel, *Embedded system design*. Springer, 2006, vol. 1.

Demo Abstract: Virtual Experimental Evaluation of RF-based Indoor Localization Algorithms

Filip Lemic*, Vlado Handziski*, Niklas Wirström†, Tom Van Haute‡,
Eli De Poorter‡, Thiemo Voigt†, Adam Wolisz*
*Telecommunication Networks Group (TKN), Technical University of Berlin (TUB)
†Swedish Institute of Computer Science (SICS)
‡Department of Information Technology (INTEC), Ghent University - iMinds

***Abstract*—This demonstration presents a set of services for streamlined experimental evaluation and benchmarking of RF-based indoor localization algorithms using previously collected raw measurements. The platform consists of an online service for storing and managing raw indoor localization data collected in a set of extensive experiments. The platform also integrates a cloud-based service for calculation of a standardized set of metrics for characterizing the performance of indoor localization algorithms. To simplify the access to the above services, we also offer a set of Software Development Kits (SDKs) for their use from Python and MATLAB. Experimenters are able to "link" the platform to their indoor localization algorithms, use previously collected data to evaluate the performance of their algorithms and calculate a set of metrics for characterizing their performance.**

I. Introduction

Indoor localization algorithms are usually benchmarked in different environments and scenarios, mostly using different hardware. Thus, even with the usage of a standardized set of metrics the results from experimental valuation are not easily comparable. In addition, experimental benchmarking of indoor localization algorithms is labor, time and cost expensive.

Within the EVARILOS project [1], we address these drawbacks by providing a set of online services for experimental benchmarking of Radio Frequency (RF)-based indoor localization algorithms without the overhead of running real measurement campaigns. Any algorithm can be evaluated and compared with other ones by running it on exactly the same raw datasets, where the raw data is a typical low-level input to RF-based localization algorithms like Received Signal Strength Indicator (RSSI), Time of Arrival (ToA), etc. The focus on raw input data differentiates our approach from the one taken in related efforts like VirTIL [2], which exports already processed range (distance) values as the basic datum. For the purpose of virtual evaluation we provide two online services: a service for access and management of database of raw localization data collected in extensive measurement campaigns and a service for calculation of an extensive set of metrics for characterizing the performance of indoor localization algorithms. The platform also includes two SDKs, for the Python and MATLAB programming languages, providing functions for easy interaction with the above introduced services.

II. Platform Overview

The overview of the platform is given in Figure 1. The service for managing the raw data is responsible for storing and making available measurement datasets collected in experimental campaigns. A detailed description of the service and its functions is given in [3]. The measurements are stored together with the locations where they are taken, annotated with the locations of the transmitting devices, metadata describing the environment and hardware used for the collection. The service provides a publicly available Application Programming Interface (API) for managing the stored data, where the user can "browse" the stored datasets and select the desired one. We further provide the visualization tool that enables users to easily visualize the collected information stored in the provided measurement collections.

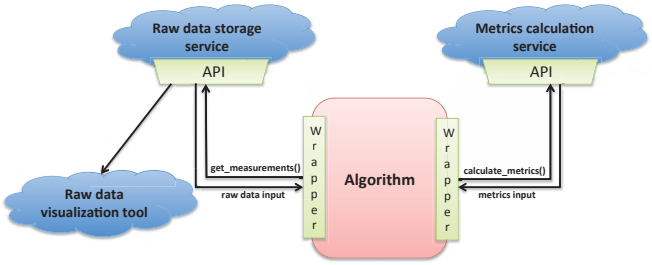


Fig. 1. Overview of the platform

The platform further consists of a set of software SDKs developed for Python and MATLAB programming languages, which are selected because they are widely used for prototyping various kinds of software algorithms, including those for indoor localization. The user can use the wrappers to fetch the desired measurements through a single function call. The user is then able to input the fetched data to the algorithm to be evaluated. The output of the algorithm, i.e. the estimated location can, together with the ground-truth coordinate where the measurement was taken, be sent to a cloud service for calculation of the performance metrics using a single function call provided by the wrappers. In that way, the user is able to easily evaluate the performance of its algorithm, using the experimentally collected data as the input and receiving a set

of standardized metrics as the output of the procedure [4]. A snapshot of the raw data is given with Listing 1. The data consists of RSSI measurements, accompanying metadata (timestamp, transmitter ID, run number, etc.) and locations of transmitting and receiving devices.

Listing 1. Raw data format

```
{
receiver_id: "MacBook Pro",
run_nr: 13,
timestamp_utc: 1373126790,
sender_id: "tplink08",
sender_bssid: "64:70:02:3e:aa:11",
rssi: -42,
channel: "11",
receiver_location: {
    room_label: "FT226",
    coordinate_z: 9.53,
    coordinate_y: 1.67,
    coordinate_x: 23.9},
sender_location: {
        room_label: "FT226",
    coordinate_z: 10.9,
        coordinate_y: 0.7,
        coordinate_x: 31},
}
```

The SDKs wrap the interaction with the cloud service for the data storage using a simple API shown in Listing 2. Using the "get_measurements" command, the experimenter is able to fetch the data from an experiment or a specific measurement. With the "filtering" command, it is possible to filter the fetched data based on desired parameters, such as number of measurements, wireless channel or transmitting device. Finally, using the "calculate_metrics" call, the experimenter is able to obtain the standardized set of the performance metrics.

Listing 2. Python API overview

```
def get_measurements(database, experiment, measurement);
def filtering(measurement, num_meas, channel, sender);
def calculate_metrics(data);
```

As the API shows, the platform offers a set of services for easy "scoping and filtering" this data, so that experimenters can selectively ask for specific record at a set of location coordinates and for a given technology and then get this dataset in an efficient way. This approach, in comparison to plain "downloading" of measurement traces, offers several benefits. Experimental raw datasets for evaluation of indoor localization algorithms can be very large. Especially for "universal" data sets that can be used for evaluation of different localization algorithms, the aim is to collect data at high spatial sampling densities and using diverse hardware equipment. At the same time, any particular algorithm would likely use only a small subset of this data in a given evaluation campaign. Approach of disseminating the whole raw data sets as "downloadable" files is thus very inefficient and leaves to the user the burden of finding the nuggets of relevant data from the whole dataset. The alternative that we offer, an online service for management of this data, is much more convenient for the users.

III. Demo Description

In this demonstration we show how the service for storage of the raw data from indoor localization benchmarking experiments can be accessed and how one can "browse" the available data collections. We also present the functionalities of the visualization tool and how it can be used to easily access the raw data and the metadata related to each data collection. Finally, we show the fetching and filtering capabilities of the SDKs for Python and MATLAB, how the data can be used by a simple WiFi-based fingerprinting algorithm and how the metrics can be calculated with one function call by interacting with the online service.

The above features will be shown on the basis of a dataset collected in the TWIST tesbted [5]. It contains multiple collections of IEEE 802.11 beacon packets RSSI values from APs distributed in locations depicted as blue squares in Figure 2. The dataset also contains collections of beacon packets from IEEE 802.15.4 nodes deployed on positions depicted with dots.

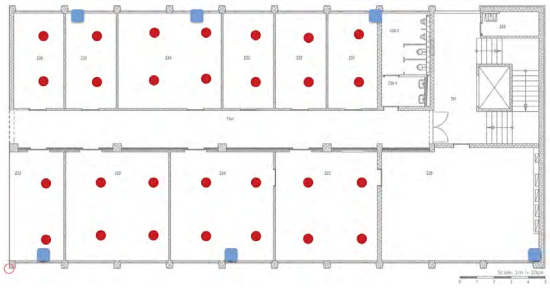


Fig. 2. Transmitting nodes locations in the testbed

IV. Conclusion and Future Work

This work demonstrates a set of tools and measurements collections that can easily be used for experimental benchmarking of IEEE 802.11 and IEEE 802.15.4 RSSI-based indoor localization algorithms, without a need for a testbed and all complexities and costs that usage of testbed introduces. Future work will be focused on collections of different types of data, such as Time of Arrival and Angle of Arrival (AoA).

Acknowledgments

This work has been partially funded by the European Commission (FP7-ICT-FIRE) within the project EVARILOS (grant No. 317989). The author Filip Lemic was partially supported by DAAD (German Academic Exchange Service).

References

[1] *Project EVARILOS*, 2013. [Online]. Available: www.evarilos.eu.

[2] S. Schmitt *et al.*, "A Virtual Indoor Localization Testbed for Wireless Sensor Networks," in *SECON'13*, 2013.

[3] F. Lemic and V. Handziski, "Data Management Services for Evaluation of RF-based Indoor Localization," Telecommunication Networks Group, Tech. Rep. TKN-14-002, 2014.

[4] F. Lemic, "Service for Calculation of Performance Metrics of Indoor Localization Benchmarking Experiments," Telecommunication Networks Group, Tech. Rep. TKN-14-003, 2014.

[5] V. Handziski *et al.*, "TWIST: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Network," in *RealMAN'06*, 2006.

Demo Abstract: Voltage Scheduling of Peripheral Components on Wireless Sensor Nodes

Ulf Kulau*, Stephan Friedrichs † and Lars Wolf*

*Technische Universität Braunschweig
Institute of Operating Systems and Computer Networks (IBR)
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
Email: [kulau,wolf]@ibr.cs.tu-bs.de

†Max Planck Institute for Informatics
Saarbrücken, Germany
Email: sfriedri@mpi-inf.mpg.de

***Abstract*—While existing work focuses on the transceiver and the processing unit to increase the energy efficiency of wireless sensor nodes, it is missed that peripheral energy consumption may dominate that of the entire node. Related to Dynamic Voltage Scaling (DVS), even peripherals' energy efficiency benefit from a downscaled voltage level, but different peripherals require different minimum voltage levels. With this demo we combine theory and practice to present the implementation of an algorithm weighing off the benefits of a downscaled voltage level against the switching overhead, e.g. for calculating an optimal peripheral voltage schedule.**

I. Introduction

As the dynamic power consumption of CMOS gates shows a quadratic dependency on the voltage level, DVS helps to significantly improve the energy efficiency of microelectronic systems [1]. Hence, several existing DVS approaches [2], [3] lead to an increase of WSN lifetime. Nevertheless, not only MCUs but also peripherals like memory devices, sensors, or actuators benefit from a downscaled voltage level.

Each peripheral hardware device requires a minimum voltage to be properly operated. The common practice is to statically configure the lowest peripheral voltage conform to all peripheral devices' voltage requirements. This can be very inefficient, because most hardware consumes more energy when exposed to higher voltage. Hence, we seek to exploit a sensor node's mechanism to dynamically switch the peripheral voltage.
The crux is that switching the voltage does not come for free. If it would, one could simply operate every peripheral device with its minimum required voltage. But switching the voltage consumes energy as well: The additional time interfacing a scalable voltage supply prolongs the duty-cycle of a processing unit, leading to a higher energy consumption.
The concept of incorporating switching cost among power different radio modes was discussed in [4], but is focussed on the radio transceiver only. In the following we outline the algorithm introduced in [5] which will be showed in our demo. This algorithm allows for peripheral voltage scheduling that weighs off the energetic benefits of switching to a lower peripheral voltage against the switching overhead without violating the minimum voltage requirements of active hardware.
Consider a sensor node with a set S of peripheral hardware devices. In order to assess the benefits of switching to a lower peripheral voltage before using $s \in S$, we need to know how much energy is consumed when querying s using the peripheral voltage v. $e_s(v)$ depends on the time t_s necessary to query s, the peripheral voltage v, and the accumulated current $I_s(v,t)$ flowing through s as well as through the inactive peripheral hardware $S \setminus \{s\}$:

$$e_s(v) = v \int_0^{t_s} I_s(v,t)\,\mathrm{d}t \tag{1}$$

Each $s \in S$ has two attributes: 1. a minimum voltage $v_{\min}(s)$ required to properly operate s, and 2. the energy consumption $e_s(v)$ of all peripherals while only s is active, depending on the peripheral voltage v, see above. Throughout this work, we assume $e_s(v)$ to be a monotonically increasing function, i. e., that a reduction of the peripheral voltage never results in an increased energy consumption. For a constant amount C of energy, the *switching overhead*, the sensor node can adapt its peripheral voltage. The sensor node is presented a sequence of queries denoted by $[1,\ldots,n]$, so that the energy consumption E of a voltage schedule is given by:

$$E = \sum_{i=1}^{n} e_{s_i}(v(i)) + \sum_{i=2}^{n} \begin{cases} C & \text{if } v(i-1) \neq v(i), \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Our goal is to minimize E, so we call a voltage schedule *optimal* if E is minimal. It follows from the monotonicity of $e_s(v)$ that an optimal schedule only uses $v(i) \in \{v_{\min}(s) \mid s \in S\} = \{V_1,\ldots,V_m\}$ with $V_1 < \ldots < V_m$.

II. Algorithm

Let us, for a pair of query and voltage (i, V_j), determine the minimum amount of energy $E_{i,j}$ necessary to reach it using a feasible schedule $v(1),\ldots,v(i)$ while assuming an infinite energy consumption for infeasible configurations. For the first query, we have:

$$E_{1,j} = \begin{cases} \infty & \text{if } V_j < v_{\min}(s_1), \\ e_{s_1}(V_j) & \text{otherwise.} \end{cases} \tag{3}$$

For $2 \leq i \leq n$, there is the mandatory energy consumption $e_{s_i}(V_j)$ to answer the query i itself, as well as the accumulated costs for traversing $i-1$ preceding configurations. There are two ways to reach the configuration (i, V_j) with an optimal energy consumption: Either the peripheral voltage from the previous query is kept, or it is changed. The former case yields an additional energy consumption of $E_{i-1,j}$. In the latter case we require the minimum amount of energy $\hat{E}_{i-1}$ to reach the

cheapest feasible predecessor configuration and the additional costs C for switching the voltage, where $\hat{E}_{i-1} = E_{i-1,\hat{\jmath}}$ with $\hat{\jmath} := \arg\min_j E_{i-1,j}$. This yields, for $2 \leq i \leq n$:

$$E_{i,j} = \begin{cases} \infty & (i, V_j) \text{ is infeasible,} \\ e_{s_i}(V_j) + E_{i-1,j} & E_{i-1,j} < \hat{E}_{i-1} + C, \\ e_{s_i}(V_j) + \hat{E}_{i-1} + C & \text{otherwise.} \end{cases} \quad (4)$$

We use dynamic programming to efficiently solve the recursion by determining $E_{i,\cdot}$ before $E_{i+1,\cdot}$; the optimal overall schedule is that ending in the configuration (n, V_j), where $E_{n,j} = \hat{E}_n$ is minimal. For a detailed description of the algorithm and some extensions please refer to [5].

III. Implementation

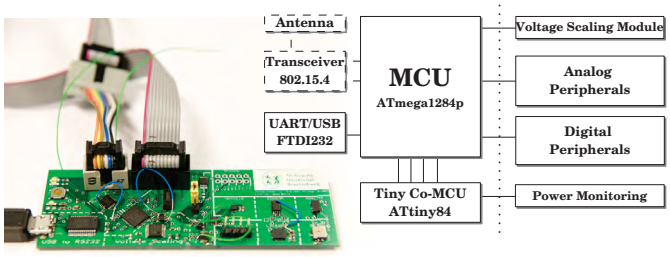


Figure 1. Block diagram and a picture of the actual prototype implementation.

Related to the INGA sensor node [6], we use an 8-bit Atmel ATmega1284p MCU as processing unit. Thus, the low computational capabilities of this MCU demonstrates that our approach is sufficiently lightweight for WSN requirements.
Figure 1 shows a picture and a block diagram of the prototype. Compared to ordinary sensor nodes a voltage scaling module is connected to the processing unit via I2C-bus. This module provides a voltage level of $1.8\,\text{V} \leq v \leq 3.3\,\text{V}$ with an 8-bit resolution to the peripherals. In this case, the overhead of switching to an arbitrary voltage level is $C \approx 7.76\,\mu\text{J}$. This includes the increased active time of the MCU and the voltage scaling module's static power dissipation, refer to [3], [5] for details.
Our prototype's sensing unit is divided into an analog and a digital section. The analog section offers the ability of connecting fully analog sensors to the ADC channels of the ATmega1284p, while the digital section includes the devices of Table I. All of them are connected via I2C bus. In order to

Table I. Equipped peripherals for demonstration.

Peripheral s	Device	Description	$v_{\min}(s)$ [V]
A	ADXL345	Accelerometer	2.000
E	AT24C08C	EEPROM	1.800
P	BMP085	Pressure Sensor	1.800
G	L3G4200D	Gyroscope	2.400
M	MAG3110	Magnetometer	1.950

calculate an optimal schedule, we need information describing the overall peripheral energy consumption.For this reason, we added a tiny co-MCU to the prototype, which is able to concurrently sample the current consumption of the peripherals (a shunt is used in connection with current sense amplifiers) and to measure the time (the co-MCU can be triggered by the ATmega1284p via digital GPIOs). Hence, $e_s(v)$ can be measured for any given values of s and v.

IV. Evaluation

Although the demonstration will give the user already the chance to optimize custom schedules, the following table depicts some exemplary schedules to show the general benefit of our approach. The energy savings are compared against three classical strategies. ConstDefault is what happens when a sensor node has no mechanism to adapt the peripheral voltage. A constant peripheral voltage of 3.3 V is kept. ConstMaxMin is the trivial strategy that uses the maximum minimum voltage, i. e., $\max_{s \in S} v_{\min}(s)$, for all queries. AlwaysSwitch always switches the voltage to its minimum requirement. It ignores the switching overhead.

Table II. Sample schedules to show the benefit of peripherals' voltage scheduling compared to naive approaches.

	Energy saved by Scheduled compared to		
Query Sequence	ConstDefault	ConstMaxMin	AlwaysSwitch
AEPGMAEPGM	45.80 %	17.13 %	0.97 %
GAMGAMGAM	46.15 %	17.04 %	0.49 %
GAMPE	46.91 %	18.52 %	1.40 %
GPGPGPGPGP	31.54 %	0.00 %	20.29 %
PAMPE	47.90 %	20.41 %	2.53 %

V. Demonstration

With a GUI a custom query of peripherals (cf. Table I) can be created. This query is transferred to the prototype board via USB. As the implementation follows a fully self-optimizing approach, the prototype board firstly self-parametrizes the energy functions $e_s(v)$ of involved peripherals. Afterwards, the board executes the optimization algorithm to get the optimal voltage schedule for the given query. Finally the query is processed while the optimal schedule is compared against the trivial voltage strategies as described in the previous section. For this purpose, the second tiny MCU samples the current consumption of ConstDefault, ConstMaxMin, AlwaysSwitch and of course Scheduled. The results are send back to the PC, where the GUI displays an oscilloscope of the current consumptions as well as an analysis of the saved energy.

References

[1] U. Tietze and C. Schenk, *Electronic Circuits: Handbook for Design and Application*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[2] L.B. Hoermann, P.M. Glatz, C.Steger and R.Weiss, "Energy Efficient Supply of WSN Nodes using Component-Aware Dynamic Voltage Scaling," *Wireless Conference 2011 - Sustainable Wireless Technologies (European Wireless), 11th European*, pp. 1 –8, april 2011.

[3] U. Kulau, F. Büsching, and L. C. Wolf, "A node's life: Increasing WSN lifetime by dynamic voltage scaling," in *The 9th IEEE International Conference on Distributed Computing in Sensor Systems 2013 (IEEE DCoSS 2013)*, Cambridge, USA, May 2013.

[4] R. Jurdak, A. G. Ruzzelli, and G. M. P. O'Hare, "Radio sleep mode optimization in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, 2010.

[5] S.Friedrichs, U.Kulau and L.Wolf, "Energy-efficient voltage scheduling of peripheral components on wireless sensor nodes," in *Communications Workshops (ICC), 2014 IEEE International Conference on*, June 2014, pp. 860–865.

[6] F. Büsching, U. Kulau, and L. Wolf, "Architecture and Evaluation of INGA - An Inexpensive Node for General Applications," in *Sensors, 2012 IEEE*. Taipei, Taiwan: IEEE, oct. 2012, pp. 842–845.