Proceedings of the
9th European Conference on Wireless Sensor Networks
Posters/Demos Session

11th – 13th February 2009, Cork, Ireland

www.ewsn.org

First published in 2009 by
Department of Computer Science,
University College Cork,
Cork,
Ireland.

ISBN 978-1-906642-06-8

Cover picture: Widyawan
Location: Cork, Ireland

Table of Contents

Message from the Posters/Demos Co-Chairs

We would like to extend a warm welcome to all delegates of the poster/demo session at the 6th European Conference on Wireless Sensor Networks (EWSN 2009) in Cork, Ireland. The posters/demos session presents late breaking results in form of posters and showcases research results in tangible form through demonstrations. We received 63 submissions, quite a large number compared to previous versions of the conference. The quality of submissions was high and we selected 9 demonstrations and 13 posters for presentation.

The accepted demos and posters cover a broad spectrum of topics in the area of wireless sensor networks from applications of sensor networks, to programming and development tools for sensor networks and management and control aspects of sensor networks. The posters and demos will be presented in a special session during EWSN 2009 on February 11th from 6pm to 8pm. However, we are grateful to the organisers of the conference for making the poster/demo session room available during the 11th and 12th of February for the full day so that delegates have ample additional opportunity to view posters/demos and interact with presenters.

We would like to thank the general co-chairs Prof. Cormac Sreenan and Dr. Utz Roedig for the general organisation of the conference and providing us with much help in organising the poster/demo session. We hope you will find the posters and demonstrations interesting and thought provoking. We encourage you to use the presentations to provide feedback to presenters and exchange and discuss ideas to make the session a valuable experience for all attendees. Finally, we hope you will enjoy your stay in Cork and spend some time to visit various places of interest around this city and its surroundings.

Dirk Pesch
Cork Institute of Technology
Cork, Ireland

Sajal Das
University of Texas at Arlington
and National Science Foundation
Arlington, USA

Demo Abstract: Wireless quality monitoring in the food chain

Reiner Jedermann, Eelco de Jong, Leon Kleiboer, Alexander Wessels, Shaoping Yuan and Walter Lang

Abstract—**The pre-processing of measurement data inside a wireless sensor node reduces the volume of communication. A model that predicts the effects of temperature deviations on the quality of fresh food products was developed for two different sensor platforms. Optimized integer arithmetic allows the model to be calculated using the resources of the existing hardware.**

Index Terms—**Food logistics, shelf life modeling, wireless sensor networks.**

I. Introduction

Monitoring cool chain transports is an important application for wireless sensor networks. Food products are very sensitive to temperature mismanagement. According to the U.S. Food and Drug Association (FDA), 20% of all perishable food is wasted during transport [1].

Spatial temperature deviations between 1 °C and 5 °C, which can be found in almost any transport, can reduce the remaining product lifetime by several days. Commercial solutions for wireless monitoring of single packages became available in 2008. But so far, these tools have not yet been able to evaluate the effects of temperature deviations on product quality. Because of high costs, neither the manual processing of one temperature chart per pallet nor the transmission of the complete sensor data over cellular networks are possible.

A. Definition of shelf life

The concept of shelf life has become a useful concept to describe the current state of the quality of a product. The shelf life of a product batch gives the number of remaining days until its quality falls below an acceptable limit and the consumer would reject purchasing the product. In other words, it states how many days the product can be kept in the shop 'on the shelf'. The shelf life depends on the product's temperature history, but must also be scaled to the temperature of the shelf or the final storage condition T_S.

B. Modeling shelf life

There are several approaches to calculate shelf life by using a mathematical model:

a) A model can directly describe the chemical processes which contribute to color changes and other decay processes [2].
b) Another approach measures reference curves for a quality attribute at certain constant temperatures. The prediction for changeable temperatures uses an interpolation between the reference curves [3].
c) For implementation on low-power micro controllers, we used a simplified model that calculates the loss of shelf life per day L as a function of the temperature T. The remaining shelf life $Q(t)$ at time t is calculated by the initial quality Q_R minus the integral over the loss per day [4], [5].

The temperature dependency of the decay process is assumed to follow Arrhenius' law for reaction kinetics [6], which calculates the reaction rate k by equation (1) with T_R as reference temperature, k_R as reaction rate at T_R, E_A as activation Energy and R_{Gas} as gas constant = 8.314 $J \cdot mol^{-1} \cdot K^{-1}$:

$$k = k_R \cdot e^{\frac{E_A}{R_{Gas}} \cdot \left(\frac{1}{T_R} - \frac{1}{T} \right)} \quad (1)$$

Two Arrhenius functions can be combined to achieve a better fit of the physical properties of the product. The loss per day is calculated according to equation (2):

$$L(T) = \frac{k_1(T) + k_2(T)}{k_1(T_S) + k_2(T_S)} \quad (2)$$

Fig. 1 shows the typical loss per day functions for fruits and vegetables, scaled to a standard temperature T_S of 15 °C. These products are affected not only by accelerated decay at high temperatures, but also by chilling injuries at low temperatures.

II. Implementation of a shelf life model on sensor hardware

In the first step, the model was implemented on the well-known TelosB / TmoteSky platform [7] with a MSP430 16-bit processor and a Chipcon CC2430 radio chip. The restricted resources of this microcontroller proved to be a challenge to the project. First, the model was required to fit into the limited program and user memory of the microcontroller. Second, the energy usage of the CPU to

Manuscript received December 1, 2008. This research was supported by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes".

R. Jedermann is with the Microsystems Center Bremen (MCB) at the University Bremen, Otto Hahn Allee NW 1, D-28359 Bremen, Germany (phone: +49 421 218 4908; fax: +49 421 218 4774; e-mail: rjedermann@imsas.uni-bremen.de).

Eelco de Jong is currently Director Sales & Marketing at Ambient Systems. Leon Kleiboer is responsible for customer support at Ambient Systems. Ambient Systems BV, Capitool 22 7521 PL Enschede, The Netherlands. Eelco.dejong@ambient-systems.net, leon.kleiboer@ambient-systems.net

Alexander Wessels, Shaoping Yuan and Walter Lang are with the MCB, Bremen. alexander.wessels@gmx.de, syuan@imsas.uni-bremen.de, wlang@imsas.uni-bremen.de

calculate the model should not significantly shorten the sensor life time. In order to adhere to these restrictions, we avoided using floating point operations in the model implementation. All mathematical operations were scaled to 16-bit integer arithmetic, except for a few critical variables that were handled as 32-bit values. The exponential function was replaced by an interpolation of value tables.

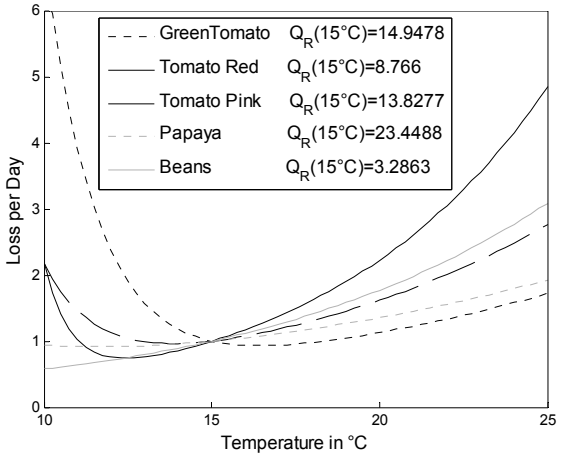


Fig.1. Loss of shelf life as function of temperature (Final storage temperature T_S = 15 °C)

A. Measurement of calculation time and memory requirements

The performance of the model implementation was tested on the TmoteSky platform. The source code was compiled by the IAR "C" Compiler. The code requires 1216 bytes of program memory. The calculation of one model step required 1999 instruction cycles. At a clock speed of 8 MHz, this is equivalent to 0.24 ms or an energy consumption of 5.8 µJ for a 3 volt supply. Compared to other operations of a sensor note this value might possibly be neglected. For example, sending the message over the radio takes 15 ms and consumes 780 µJ of energy.

B. Implementation on an 8-bit controller

Unfortunately, the TmoteSky sensors can only be used for prototype tests using real-world transport conditions. The absence of water-protected housing and its high price prevent a commercial use of the system. In cooperation with a manufacturer of sensor systems, a second microcontroller implementation was developed. The Ambient 3000 SmartPoint contains a Chipcon CC2430 radio chip with a built-in 8-bit 8051 processor.

C. Accuracy of integer arithmetic

The shelf life prediction by the sensor was compared to a floating point model that was simulated in Matlab. The error for the loss per day calculated by integer arithmetic was between 0.68 % and 1.37 % for the different products in Fig. 1.

III. The Sensor Network

Ambient's networks consist of three elements: A GateWay, each network contains a single GateWay, which provides the interface between corporate IT systems and the Ambient network. It is mains powered and provides a serial interface (RS-232) for communication with an external device. Each network contains up to 255 MicroRouters, which provide the backbone of the wireless infrastructure. The MicroRouters wirelessly communicate with the GateWay, with each other and with the SmartPoints. SmartPoints are battery-powered devices that have a similar role as traditional active RFID tags, but with significantly enhanced capabilities. SmartPoints monitor their environment using a default temperature sensor. SmartPoints can be extended with additional sensors (I2C, 1-Wire, SPI, ADC, and GPIO). SmartPoints are self-locating. Based on the wireless communication with Ambient's infrastructure, a SmartPoint is able to calculate its own location in the network.

IV. Demonstrator Setting

Each type of fruit has an individual set of parameters for use with the Arrhenius model and a different reaction to temperature deviation. A demonstration of the enhanced wireless sensors compares the shelf lives for different food products that are exposed to the same temperature conditions during a mixed transport (Fig 2). One sensor is programmed with the parameters for 'Strawberries' and another one for 'Lettuce'. The sensors are placed inside a box with the corresponding product. The differences in the speed of quality decay are displayed by the graphical interface of the control software. A second demonstration setting shows the difference in shelf life for two batches of one product that are stored with different temperature conditions.

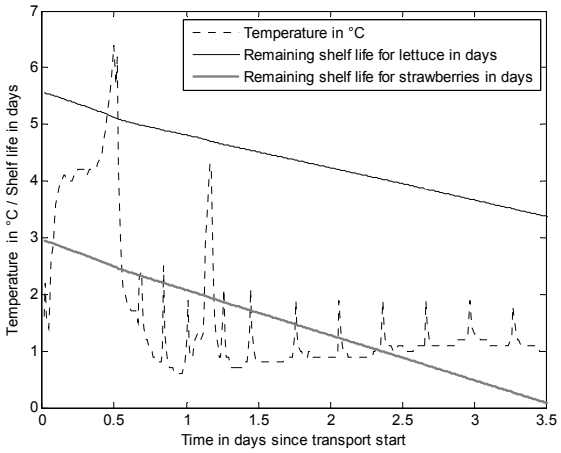


Fig. 2. The quality curves for two different food products stored at the same temperature in a chilled transport (T_S = 5 °C).

References

[1] K.C. Gross, "Fresh vegetable quality in the United States: Current issues" Acta Hort. (ISHS) vol. 483, pp. 39-47, 1999.

[2] E. Bobelyn, M. L. A. T. M. Hertog and B. M. Nicolaï, "Applicability of an enzymatic time temperature integrator as a quality indicator for mushrooms in the distribution chain", Postharvest Biology and Technology, vol.42, no.1, pp. 104-114, 2006.

[3] R. Jedermann, J. P. Edmond and W. Lang, "Shelf life prediction by intelligent RFID", Proc. Dynamics in Logistics. First International Conference, LDIC 2007, Berlin Heidelberg, pp. 231-240, 2008.

[4] L. M. M. Tijskens and J. J. Polderdijk, "A generic model for keeping quality of vegetable produce during storage and distribution", Agricultural Systems, vol.51, no.4, pp. 431-452, 1996.

[5] R. Jedermann, R. Schouten, A. Sklorz, W. Lang and O. van Kooten, "Linking keeping quality models and sensor systems to an autonomous transport supervision system", Proc. of the 2nd international Workshop Cold Chain Management, pp. 3-18, 2006.

[6] S. Arrhenius, "Über die Reaktionsgeschwindigkeit bei der Inversion durch Säuren", Zeitschrift für physikalische Chemie, vol.4, 1889.

[7] MoteIV Corporation, "Tmote sky - Ultra low power IEEE 802.15.4 compliant wireless module datasheet", 2006.

Demo Abstract: Seamless Sensor Network IP Connectivity

Geoff Mulligan•, Colin O'Flynn†, Julien Abeille*, Mathilde Durvy*, Patrick Wetterwald*,
Blake Leverett⋆, Eric Gnoske⋆, Michael Vidales⋆, Nicolas Tsiftes‡, Niclas Finne‡, Adam Dunkels‡
geoff@mulligan.com, coflynn@newae.com, {jabeille,mdurvy,pwetterw}@cisco.com,
{blake.leverett,eric.gnoske,michael.vidales}@atmel.com, {nvt,nfi,adam}@sics.se
• Proto6 LLC, † NewAE, * Cisco Systems, ⋆ Atmel Corporation, ‡ Swedish Institute of Computer Science

***Abstract*—Most existing sensor network systems require custom gateways and tools to interface to the Internet. We present a seamless IP-based sensor network connectivity mechanism that allows a sensor network to be instantly connected to a PC and additionally to the Internet. We use an IP stack on the sensor network and a USB stick on the PC that acts as a wireless network card. The USB stick carries its own network drivers for both Linux and Microsoft Windows. By inserting the stick in a PC, the PC is instantly connected to the sensor network. This allows the PC to use standard tools, such as ping and web browsers to access the sensor network nodes and allows the nodes to send packets both to the PC and the Internet.**

I. Introduction

Even though many wireless sensor networks use IEEE 802.15.4, none provide seamless and direct IP connectivity of the sensor network to IP-based systems and the Internet. Such sensor networks require translation gateways to map their proprietary protocols to IP. This adds costs, complexity, requires additional training and knowledge and breaks many of the end to end models used in the Internet for reliability, confidentiality and authentication.

By using an IPv6-compliant protocol stack, the sensor network can be easily integrated into IPv6 networks and leverage existing tools, protocols, knowledge and networking infrastructure [4], [5], [9]. The IP-enabled sensor network can be managed with existing or readily available tools. Knowledge of IP network management can be applied to the sensor network. Additionally, development and programming of the sensor network does not require learning new network protocols or paradigms. Recent work has shown that by using a power-saving MAC protocol in the sensor network, IP-based sensor networks are as power-efficient as sensor networks based on proprietary or specialized mechanisms [7].

We run uIPv6, the world's smallest fully compliant IPv6 stack [6], together with the SICSlowpan implementation of IPv6 transmission over 802.15.4 networks [8], [9] in the Contiki operating system [3].

We show how a set of IP-based sensors can be networked together and connected easily to a standard PC and via the PC to the Internet using only standard IP based protocols [2] and therefore without the need for special translation software. We also show how the nodes, the PC and Internet devices can communicate using standard software and tools, such as ping, telnet and HTTP. Contiki has supported IP-based sensor networks for several years [5] but until now, their integration with general-purpose IP-based networks have either required additional software on the router or non-standard access mechanisms such as Serial Line IP (SLIP).

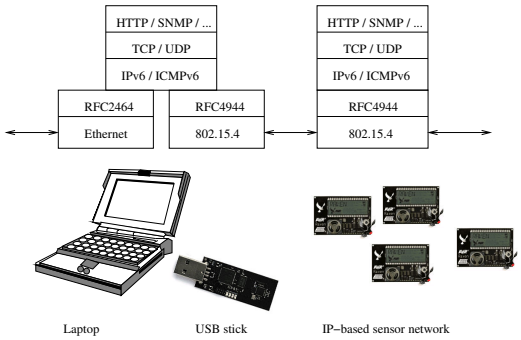


Fig. 1. The IP-based sensor network is seamlessly integrated with external networks through a USB stick inserted in a laptop or router. The sensor nodes and laptop/router runs an IP stack.

II. System Design

Our IP-based sensor network consist of a set of Atmel Raven boards, one PC laptop acting as an IP router, and one USB stick that bridges IP packets from the PC laptop to the sensor network. The USB stick provides a generic network interface to the PC laptop. The USB stick is supported in Linux, Windows XP, and Windows Vista. Linux does not require any driver file; Windows requires a simple INF file that tells Windows to use its built-in drivers.

The Raven boards include two Atmel microcontrollers, an ATMega 1281 with 128K Flash and 16K RAM to handle all the communications stack, an ATMega 3290 with 32K Flash and 16K RAM to control the onboard LCD and an Atmel AT86RF230 IEEE 802.15.4 compliant radio. In addition, the end nodes have temperature, light and audio sensors. The USB stick includes a single microcontroller, an Atmel AT90USB1287 with 128K of FLASH and 8K of RAM and an Atmel AT86RF230 radio. This microcontroller handles both the USB interface and the communications stack.

The Raven boards run uIPv6, our IPv6 Ready protocol

stack [6], [1], along with the SICSlowpan implementation of the 6LoWPAN adaption layer [8], [9], and a simplified, compliant IEEE 802.15.4 MAC on top of the Contiki operating system [3]. On top of uIPv6, we run a web server that provides access to the on-board sensors.

The USB stick runs the SICSlowpan header compression and decompression software and the necessary software for the USB interface and to present an Ethernet like interface to the connected PC. The USB stick acts like a stand-alone network card to the PC and a Zone Area Network controller (ZAN-controller) to the sensor nodes. The USB stick additionally presents a serial port to allow debug messages from the stack to be viewed.

A. Network Architecture

Figure 1 shows the network architecture of our demonstration. IP packets are routed by the PC laptop to and from the sensor network. The USB stick bridges the IP packets from the laptop and compresses and decompresses the IPv6 headers with 6LoWPAN header compression.

The sensor network uses a star and leaf topology. Within the Sensor Zone Area Network (S-ZAN), all sensor nodes communicate via a simple star. End-node to end-node communication is accomplished via forwarding by the ZAN controller. This initial demonstration does not provide multi-hop forwarding with the ZAN.

B. Address Assignment

The sensor nodes use IPv6 Stateless Address Autoconfiguration [11] to determine and set their global IP addresses. When the network is started, by the PC and USB stick, the PC requests an IPv6 prefix for the sensor network using DHCPv6-PD. Next, the PC starts listening for Neighbor Discovery (ND) [10] requests through the USB network interface. When the sensor nodes are powered up, the sensors first create a link-local IPv6 address by combining the standard IPv6 link-local prefix - fe80::0/64 - with its MAC address. The sensor node then performs Duplicate Address Detection (DAD) to ensure this address is not in use. Next, the node transmits an ND Router Solicitation message to autoconfigure its network address. The ZAN-controller responds with a Router Advertisement packet to the node.

C. Routing and Packet Forwarding

In general, packets in the sensor network can be sent from any node to any node in the sensor network, but in our demonstration, packets sent by the sensor node are sent through the default router, the ZAN-controller running on the USB stick. The ZAN-controller forwards any packets it receives that are not addressed to itself. If the Network Prefix is that of the local sensor network, the ZAN-controller forwards the packet to the destination endnode. In this way, sensor nodes can communicate with other sensor nodes within the local sensor network (the ZAN). If the Network Prefix is not on the local IPv6 network, the ZAN-controller forwards the packet via its USB interface to the PC. The PC will then intern forward the packet as necessary and usually via its Default Route.

III. Evaluation

We demonstrate the usefulness of IP-based sensor networking by showing that regular IP-based network tools can be used to access, inspect, and manage the sensor network. Because the sensor nodes have IP addresses, a general-purpose web browser can be used to access the sensor nodes. Similarly, IP-based tools such as ping, traceroute, and telnet can be used to communicate with the network.

Because the USB stick acts as regular network interface on the PC, communication in the sensor network can be captured using a standard IP network capture tool such as Tcpdump or Wireshark. Similarly, the USB stick can be used on any nearby PC to capture the traffic without requiring physical access to the network router.

IV. Conclusions

Traditional sensor networks have required proprietary tools and mechanisms for access and management. IP-based sensor networks simplify access to and management of sensor networks by using the open standard IP set of protocols. We leverage the base of existing protocols and technology to further extend the capabilities of sensor networks.

Our demonstration shows an IP-based sensor network that seamlessly connects a sensor network the Internet. We show that the severely constrained sensor nodes with limited Flash program space and even more limited RAM buffer space, it is possible to deploy these types of embedded IP systems allowing IP connectivity and network management and control without the need for complex translating gateways.

References

[1] The IPv6 Ready Logo Program. http://www.ipv6ready.org.

[2] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, IETF, December 1998.

[3] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, November 2004.

[4] A. Dunkels and J-P Vasseur. IP for Smart Objects, September 2008. The Internet Protocol for Smart Objects Alliance. White Paper 1.

[5] A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proc. EWSN'04 work-in-progress session*, January 2004.

[6] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008)*, Raleigh, North Carolina, USA, November 2008.

[7] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proceedings of the 6th International Conference on Embedded Networked Sensor Systems (SenSys 2008)*, Raleigh, North Carolina, USA, November 2008.

[8] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Internet proposed standard RFC 4944, September 2007.

[9] G. Mulligan. The 6lowpan architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 78–82, New York, NY, USA, 2007. ACM.

[10] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861, IETF, September 2007.

[11] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862, IETF, September 2007.

Demo Abstract: Efficient Building Management with IP-based Wireless Sensor Network

Rostislav Spinar[1], Panneer Muthukumaran[1], Rodolfo de Paz[1], Dirk Pesch[1], Weiping Song[2], Shafique Ahmad Chaudhry[2], Cormac J. Sreenan[2], Essa Jafer[3], Brendan O'Flynn[3], James O'Donnell[4], Andrea Costa[4], Marcus Keane[4]

{rostislav.spinar, panneer.muthukumaran, rodolfo.depaz, dirk.pesch}@cit.ie, {w.song, s.chaudhry, cjs}@cs.ucc.ie, {essa.jafer, brendan.oflynn}@tyndall.ie, {jtod15, andre.costa.phd}@gmail.com, marcus.keane@nuigalway.ie

[1]Centre for Adaptive Wireless Systems, Cork Institute of Technology , Ireland

[2]Mobile Internet Systems Lab, Department of Computer Science, University College Cork, Ireland

[3]Tyndall National Institute, Lee Maltings, Cork, Ireland

[4]Informatics Research Unit for Sustainable Engineering, National University of Ireland, Galway

***Abstract*—Existing Building/Energy Management Systems (BMS/EMS) fail to convey holistic performance to the building manager. A 20% reduction in energy consumption can be achieved by efficiently operated buildings compared with current practice. However, in the majority of buildings, occupant comfort and energy consumption analysis is primarily restricted by available sensor and meter data. Installation of a continuous monitoring process can significantly improve the building systems' performance. We present WSN-BMDS, an IP-based wireless sensor network building monitoring and diagnostic system. The main focus of WSN-BMDS is to obtain much higher degree of information about the building operation then current BMSs are able to provide. Our system integrates a heterogeneous set of wireless sensor nodes with IEEE 802.11 backbone routers and the Global Sensor Network (GSN) web server. Sensing data is stored in a database at the back office via UDP protocol and can be access over the Internet using GSN. Through this demonstration, we show that WSN-BMDS provides accurate measurements of air-temperature, air-humidity, light, and energy consumption for particular rooms in our target building. Our interactive graphical user interface provides a user-friendly environment showing live network topology, monitor network statistics, and run-time management actions on the network. We also demonstrate actuation by changing the artificial light level in one of the rooms.**

***Index Terms*— Wireless sensor networks, network management, policy-based management, building management systems**

I. System Architecture

Enhanced understanding of building operation is required to achieve improved performance, with a 20% reduction of energy consumption achievable by efficiently operated buildings compared with typical practice [1]. Currently, building managers rely heavily on the BMS/EMS to acquire performance related information. The BuildWise project [2] has identified that the level of information available with existing BMS\EMS is insufficient to perform the required performance based assessment of building operation [3]. This assessment is essential in order to reduce building energy consumption, operational costs, and to comply with European Legislation (Directive 2006/32/EC). We strive to fill this gap by deploying the WSN-BMDS as a base technology platform to facilitate this assessment. The system has been developed by the WISEN Emnets project [4] and deployed as a part of the BuildWise project in the Environmental Research Institute (ERI) [5] building in Cork. The WSN-BMDS test-bed is a 3-tiered framework with the following components:

1. IEEE 802.15.4 sensor nodes forming 6LoWPAN network

For sensing purposes, we use Tyndall [6] and TMote Sky [7] sensor nodes. The Tyndall board is equipped with an Atmega1281 MCU and EM2420 radio chip and the TMote Sky features a MSP430 MCU with CC2420 radio chip. Both platforms include sensors for monitoring air-temperature, air-humidity and light. Moreover, we utilize recently developed sensor boards for the Tyndall platform that incorporates electricity meters as well as the interface for controlling (on/off) an AC load. Both platforms run the recently released b6LoWPAN stack [9].

2. IEEE 802.11 gateways as 6lowPAN/IPv6 routers

Soekris embedded PC boards [10] with Atheros CM9 Wi-Fi cards and a single IEEE802.15.4 node form a backbone network spanning all three floors of the ERI building.

3. User-end PC for presenting network variables

The network data is gathered from the first and ground floor to the local PC in the basement. The sensor readings are processed and stored by the GSN. An interactive network monitoring and management system described in section III.

II. Node Locations

In order to demonstrate the inefficiencies in building operation three main zones within the ERI building have been chosen according to the needs of the BuildWise project.

- Seminar Room and Break-Out Space on the first floor (nodes 1, 2, 3) with a room where the manifold for this space is located (node 4)

- Open Office space on the first floor (nodes 8, 9, 10) with a room where the manifold for this space is located (node 7)

- Immunology lab on the ground floor (nodes 11, 12)

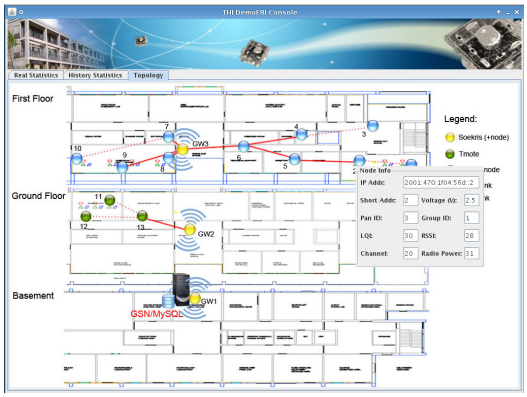


Fig. 1: The WSN-BMDS deployed in the ERI, spans three floors of the ERI building.

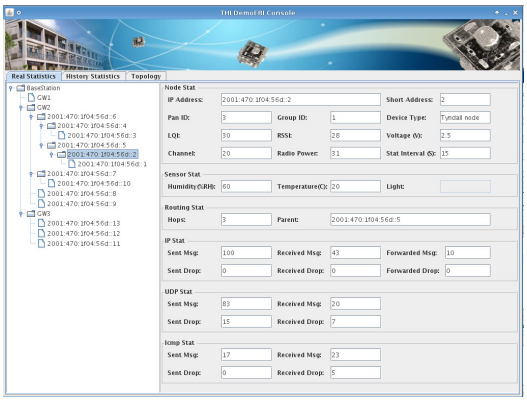


Fig. 2. Management console

Nodes 5, 6, 13 are intermediate nodes maintaining connectivity to the gateway. Moreover, node 5 provides data sets from a room where no wired sensors are located as this room was created later by partitioning one bigger space into two smaller rooms. Nodes 6 and 13 also provide a data calibration measure since they are located on top of wired sensors of the current BMS.

III. Network Monitoring and Management

Network management of our testbed is essential to keep the WSN-BMDS always operational. EmNets [4] Management Protocol (EMP), runs as the management data collection and dissemination protocol. EMP not only collects node and network level management information, it can also be used to execute management operations on the sensor nodes.

Our interactive graphical user interface provides a user-friendly environment to view live network topology, monitor network statistics, and run management actions on the network. Fig.1. is a snapshot of the deployed sensor network topology. Users, by just navigating the topology map, can get the basic node information e.g. node ID, node's IP address, and sensed variables.

Fig.2. shows the detailed information related to the sensor nodes under various categories. Run-time network probing is also supported to collect network management information and network statistics.

IV. Data Storage and Presentation

Accurate measurements from wireless sensors along with an appropriate interface to integrate these measurements into the current BMS are essential parts required by the BuildWise project. Therefore, we have chosen to make use of the Global Sensor Network GSN middleware [11] which acts as a local database as well as provides XML-based interface for applications such as BMS. Furthermore, GSN also provides a Web-based interface to view real-time as well as historical sensor data.

V. Demonstration

We will demonstrate the operation of the WSN-BMDS system as deployed in the ERI building, which presents a base line for research in the area of wireless sensor and actuator networks for wireless building performance management. The demonstration will provide a real-time view and historical measurements of air-temperature, air-humidity, light, and energy consumption from wireless sensors. Through the system, users will be encouraged to control the lighting level in a room with immediate feedback using USB cameras recently deployed in the ERI. The possibility to connect directly to any node within the network using IP-compatible applications such as ping or netcat will be also demonstrated.

VI. Conclusion and Future Work

We show how the WSN-BMDS can facilitate the objectives of the BuildWise project towards a comprehensive understanding of complex building management and control operations. The system will also provide an experimental facility to conduct research in indoor wireless sensor and actuator networks. More specifically, we make a contribution to the recently released b6lowPAN stack by interfacing it with our new IEEE802.15.4 compatible MeshMAC [8].

Please refer to: http://erideployment.ucc.ie:8080/ for the current status of this demonstration.

References

[1] James O'Donnell, Marcus Keane and Vladimir Bazjanac, Specification of an Information Delivery Tool to Support Optimal Holistic Environmental and Energy Management in Buildings in 'SimBuild 2008' IBPSA-USA. Berkeley, CA, USA.
[2] BuildWise Project Website http://zuse.ucc.ie/buildwise
[3] Martin Keller, James O'Donnell, Karsten Menzel, Marcus Keane, Ufuk Gokce Integrating the Specification, Acquisition and Processing of Building Performance Information, in Tsinghua science and technology
[4] EmNetS Project: http://www.cs.ucc.ie/emnets/
[5] Environmental Research Institute building http://zuse.ucc.ie/buildwise/eri/e_overview.html
[6] Tyndall Platform http://www.tyndall.ie/mai/25mm.htm
[7] TMote Sky Platform http://www.sentilla.com/moteiv-transition.html
[8] P. Muthukumaran, R. Spinar, K. Murray, D. Pesch, "Enabling Low Power Multi-hop Personal Area Sensor Networks", in Proc. of 10th WPMC, Jaipur, India, December 2007
[9] b6lowPAN Project Website http://smote.cs.berkeley.edu:8000/tracenv/wiki/b6loWPAN
[10] Soekris Embedded Boards: http://www.soekris.com/net4826.htm
[11] GSN Middleware http://gsn.sourceforge.net

Demo Abstract: Survivable and Scalable WSN Solution for Environmental Monitoring in Harsh Conditions

Krisakorn Rerkrai, Christine Jardak,
Aleksandar Kovacevic, Janne Riihijärvi and Petri Mähönen
Department of Wireless Networks, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
Email: kre@mobnets.rwth-aachen.de

Alban Hessler
NEC Europe Ltd.
Kurfürsten-Anlage 36
D-69115 Heidelberg, Germany
Email: alban.hessler@nw.neclab.eu

***Abstract*— Designing scalable and self-organizing sensor networks is vital for enabling the real-world deployment of large-scale applications. In this demo we show a fully functional, but scaled-down system that was developed for agricultural WSNs. The full-scale system with 64 sensor nodes was deployed with a backend server to a medium-sized vineyard. The protocols used ensure reliable local data storage, robust communication and energy-efficient data gathering. The backend server provides a user-friendly interface offering two main functionalities: logging communication messages of the employed protocols and providing end-user support for both on-demand and periodic data requests.**

I. Introduction

The use of networked sensors in agriculture is becoming more feasible and important. Environmental sensors such as temperature, light and soil moisture can be employed across agricultural fields to provide real-time data to farmers. Environmental data need to be constantly and automatically monitored. Such systems are generally developed to support precision agriculture in plant care. In particular, our system helps farmers to control irrigation, and to adjust fertilizer and pesticide deployments more precisely for individual parts of the vineyard [6].

In this demonstration, we present an autonomous monitoring system for vineyards that is capable of collecting and storing data as well as providing measured data upon request with reliable routing and transport protocols. The system provides centralized and distributed data storage. Data from the network are regularly backed up at a backend server. Such a centralized approach is naturally prone to failures. Therefore, we implemented as well a distributed data backup which is performed by constantly replicating the aggregated data in a neighboring cluster. The graphical user interface allows end-users to retrieve the measured data upon request or to send notification emails or SMSes to registered users. The protocols used in this demonstration have been developed in the the UbiSec&Sens project [3]. The system has been successfully deployed at a commercial vineyard with 64 sensor nodes at the city of Neustadt, Germany, during vine growing season until harvest time in 2008.

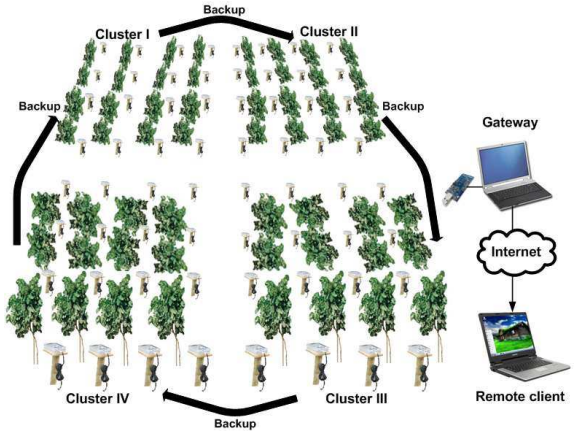


Fig. 1. System architecture.

II. System Overview

The architecture of the UbiSec&Sens vineyard monitoring system is shown in Figure 1. The system is composed of two main parts: a collection of wireless sensor nodes and a backend server. The general idea is to collect the environmental data either locally from the network if a user is in the vineyard or remotely through the backend server through Internet.

We have chosen the Telos sensor platform [1] as a basic platform for WSNs. Telos motes enable the addition of extra sensor or devices with ADC connections, which are useful for our purposes. Built in temperature and light sensors are also employed as a part of our demonstration. We use an additional ECHO EC5 soil moisture sensor [2] depicted in Figure II to measure soil moisture values in the ground. Each node is housed in a waterproof box with a transparent lid which allows the embedded light sensors on the Telos motes to measure the light intensity.

The sensor nodes are distributed in a loose grid topology. A group of sensor nodes are geographically clustered

and the data from the nodes are aggregated at each cluster head. Consequently, the aggregated data are transmitted to a gateway node over the multihop network constructed from different clusters. A collection of protocols used in this demo include aggregator node election (PANEL [8]), lightweight hybrid routing (TinyLUNAR [5]), persistent data storage (TinyPEDS[7]) and reliable transport protocol (NanoTCP [4]). These protocols together make the entire system persistent, reliable and self-organizing. The system also has a self-healing mechanism to recover system functions in the case of failures.

The backend server, on the one hand, handles on-demand queries from a local user who wants to monitor environmental changes. The results are displayed through a graphical user interface we developed. The basic window of GUI is shown in Figure 3. A user can also register himself for periodic notifications through emails. The configuration can be modified upon the need of users. This allows both technical and non-technical users to customize the system if necessary.

(a) (b)

Fig. 2. Sensor node in a weather-proof box and soil moisture sensor.

III. Demo Description

We propose to demonstrate a fully functional but scaled-down version of vineyard monitoring system. The on-site demo is deployed with 16 nodes divided into 4 clusters. Its functionality is identical to the system running in the vineyard. A smaller subset of hardware components is demonstrated due to space limitations. The demo is comprised of 3 main parts: auto-configuration, normal operation and self-organization.

First, the nodes get started once they receive a synchronization packet from a gateway node. This packet is periodically sent to the network to ensure that all the nodes are synchronized with the same epoch number and perform a normal operation at the same time. We demonstrate the bootstrapping process, and also the fact that the system can continue a normal operation even when replacing batteries of some nodes. Every N epochs, each cluster elects its cluster head as an aggregator node. The aggregators constantly collect sensor readings from its members through a multihop routing protocol. Aggregated data will be stored locally on the aggregators and on the neighboring clusters as backup information and also collected at the gateway node. These storing strategies allow the users to retrieve the information either on-demand directly from the network or periodically from the database. The former case is more practical when the users are in the field. They can

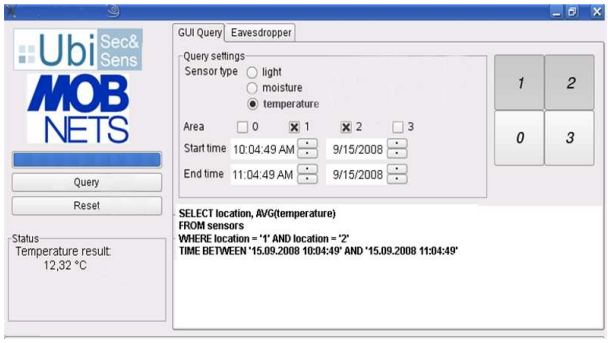


Fig. 3. Graphical user interface at the eavesdropper.

construct their own query and deliver it to the network on the fly. Each and every step can be shown to the conference participants on the GUI.

We will also highlight the use of the self-organizing system in harsh conditions. For example, lopping and harvesting machines can physically damage some nodes in the network. However, this will not influence the whole network since the data is aggregated throughout the entire clusters and routing is done dynamically. If the broken node is an aggregator, a new aggregator will be elected in the next election period. In a worse case situation, a forest fire can burn some regions of the sensor nodes. In such a case, we can still retrieve the backup information from a neighboring cluster. Additionally, spurious or erroneous data which are produced by hardware failures can be eliminated by data filtering process.

We will additionally display a video demonstration of our full-scale system consisting of 64 nodes deployed in a vineyard during the vine growing season. In summary, we show in this demo a complete and scalable environmental monitoring system that can survive in harsh conditions.

Acknowledgment

We would like to thank European Union (Project UbiSec&Sens) for providing financial support for our work. The authors from Aachen would like to thank RWTH Aachen University, Deutsche Forschungsgemeinschaft through the UMIC research center and the European Union (Project WASP) for providing financial support. We also thank Junaid Ansari and Dirk Westhoff for their support.

References

[1] Crossbow website, http://www.xbow.com.
[2] Decagon website, http://www.decagon.com.
[3] UbiSec&Sens project website, http://www.ist-ubisecsens.org.
[4] C. Jardak et al. Implementation and Performance Evaluation of nanoIP Protocols: Simplified Versions of TCP, UDP, HTTP and SLP for Wireless Sensor Networks. In *Proc. of IEEE WCNC 2008*, Las Vegas, NV, 2008.
[5] E. Osipov. TinyLUNAR: One-Byte Multihop Communications Through Hybrid Routing in WSNs. In *Proc. of NEW2AN*, Russia, 2007.
[6] J. Burrell et al. Vineyard Computing: Sensor Networks in Agricultural Production. *IEEE Pervasive Computing*, 3(1), 2004.
[7] J. Girao et al. TinyPEDS: Tiny Persistent Encrypted Data Storage in Asynchronous WSNs. *Elsevier Ad Hoc Journal*, 5(7), 2007.
[8] L. Buttyán et al. PANEL: Position-based Aggregator Node Election in Wireless Sensor Networks. In *Proc. of IEEE MASS*, Pisa, Italy, 2007.

Demo Abstract: WiLab, a real-life Wireless Sensor Testbed with Environment Emulation

Lieven Tytgat, Bart Jooris, Pieter De Mil, Benoît Latré, Ingrid Moerman and Piet Demeester
Ghent University – IBBT, Department of Information Technology (INTEC) – IBCN
Gaston Crommenlaan 8, bus 201, B-9050 Ghent, Belgium
firstname.lastname@intec.ugent.be

***Abstract*—Current wireless sensor testbeds have a 'passive' nature, meaning that once the testbed has been set up, the configuration remains the same. It is not possible to change the characteristics of the device such as the available battery power. This shortcoming severely effects the types of simulations that can be evaluated. The WiLab testbed solves this issue by using an active element, the Environment Emulator. This device is capable of emulating several events (e.g. battery depletion, energy harvesting power sources, node failure, sensor events, actuator events, audio streams etc.) in a real-life environment without the need for real sensor hardware or real sensor applications. Using this approach, novel wireless sensor network architectures and protocols can be evaluated in a true real-life environment with realistic user scenarios. In addition, the real-time power consumption of every node can be monitored, enabling the development of new and verification of existing power optimized protocols and applications. In this demo, we will demonstrate the possibilities of WiLab.**

I. Introduction

More and more researchers have started to realize that only relying on simulations as a validation tool for new wireless protocols and applications will not guarantee its proper working in a real-life environment. Existing simulators do not have the possibility to cover all important real-life effects of wireless communication, often leading to a considerable discrepancy between experiments and simulations [1]. In order to solve these issues, wireless sensor testbeds are used to validate the correctness of developed protocols.

Wireless sensor testbeds enable to effectively test the wireless sensor protocols or applications in a real-life environment. In such an environment, the spectrum is not 'clean', and the locations of the nodes are fairly random. Some testbeds got deployed in real 'office-like' buildings [2], [3]. The TWIST testbed [3] has implemented a simple, binary power supply control, making it possible to emulate the death of a node. Emulating battery depletion or energy harvesting is not considered. In these testbeds, it is not possible to act upon real-life changes. E.g., suppose an emergency protocol for a wireless fire alarm is being developed. During a fire, the temperature sensors should start propagating warnings and it is quite possible that an explosion destroys some of the sensors. In existing testbeds it is not possible to test the protocol under such real fire conditions.

In this demonstration, we will show the possibilities of the WiLab testbed. This testbed is equipped with a new piece of hardware, the Environment Emulator (EE). Using this, it is possible to emulate the behavior of any type of sensor or actuator without the need for real sensor/actuator hardware or the development of a full-blown sensor application. It is possible to emulate the battery depletion, depending on the real life power consumption of the sensor node. When the node's battery is depleted or the node is destroyed (e.g., in an explosion), the node can be switched off. The EE can be programmed to emulate a sensor event (e.g., temperature rise, motion detection), an actuator event or to support voice streams. Further, the EE can be used to monitor the energy consumption of each individual sensor. Altogether, this means that it is possible to assess the complete usability of a certain wireless sensor and actuator network application or protocol in a real-life environment. This is the first testbed that supports these features on all nodes.

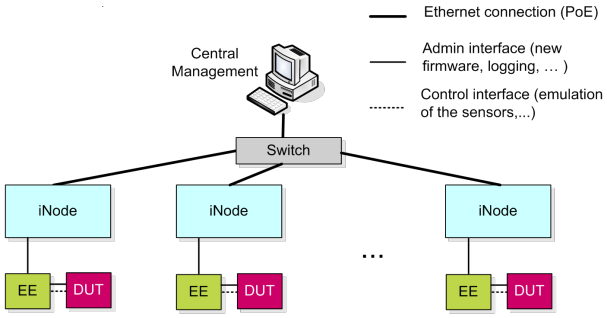


Figure 1. The WiLab Architecture

In the following, the testbed's architecture and the characteristics of the environment emulator is given in Section II. In Section III the demonstration will be described.

II. WiLab testbed

A. Architecture

The WiLab testbed is deployed in an office building of 12x80m and is spread out over three floors. It consists out of more than 200 sensor nodes. Figure 1 shows the architecture of the testbed. The core is based on the widely used MoteLab testbed from Harvard University [2]. The sensor nodes are Tmote Sky motes, although the hardware setup allows for a large variety of sensor nodes as long as they can be programmed through a USB or serial interface. The intermediate

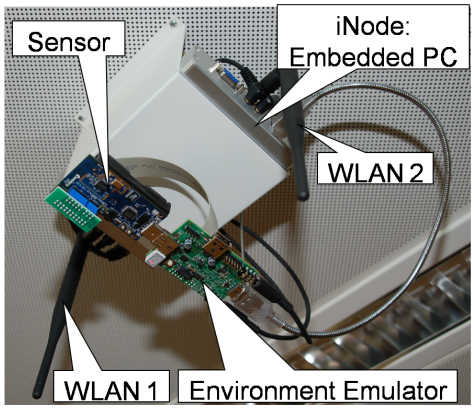


Figure 2. A complete sensor node set up (iNode + Environment Emulator + sensor node)

nodes (called iNodes) are Alix 3C3 [4] devices running Linux. These are mini PC's equipped with Ethernet, USB, serial, vga, audio and two 802.11bg interfaces. All the iNodes are connected to the management backbone using Power-over-Ethernet switches, making it possible to power up/down the iNodes as needed without physical interaction. Finally, the Environment Emulator is located in between the iNode and the sensor node (=Device under Test or DUT), see Fig. 2. Finally, all the possibilities of the complete testbed, the environment emulator scenarios and events, a visualization and a graphical analysis tool, are accessible through a web interface. The visualization tool can visualize any type of node status and/or link information on a map of the building, while the graphical analyzer plots out the data. The information for both tools is gathered from the MoteLab database through the use of user customizable MySQL statements, making it extremely flexible.

B. Environment Emulator (EE)

In Fig. 3 the connection diagram of the environment emulator is shown. Its major possibilities are described below: (1) The EE can disconnect the USB power from the DUT, and power it with its own regulating voltage source. This enables the EE to emulate the real behavior of a battery depleting, energy harvesting power sources, failing of the mote,...(2) The current used by the DUT can be measured with a sample frequency of 10kHz. This makes it very easy to determine the exact power consumption when tested. An example output can be seen in Fig. 4 (3) The EE has some General Purpose digital Input / Output pins connected to the DUT. This allows for real-life, real-time digital sensor/actuator emulation. (4) Some analogue input/output pins are also connected to the DUT, making the emulation of real-life, real-time analogue sensors/actuators possible. (5) Audio input/output signals can be injected and extracted from the iNode making all sorts of audio over sensor network experiments very easy to conduct.

The EE can programmed to emulate events using scenarios. These allow for the efficient comparison of different protocols under exactly the same situations. On the other hand, it is also possible to issue real sensor events (e.g., 'button press'), during an experiment.

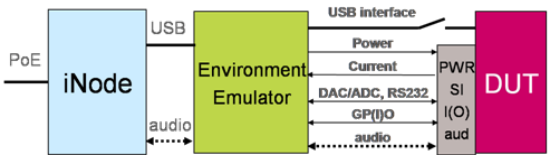


Figure 3. The environment Emulator connection diagram

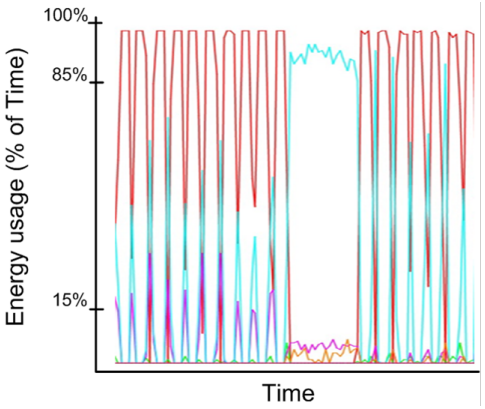


Figure 4. Example of a current measurement. It can be seen that the node is switched off periodically.

III. Demo Setup

The demo will consist out of two major parts. First of all, there will be a small local testbed setup on which the features of the environment emulator will be displayed in real-life. More specifically: the battery emulation, sensor and actuator emulation and node failure will be demonstrated. The second part of the demo will contain a demo on the testbed located at IBBT. In this demo, a real-time visualization of the testbed will demonstrate the advantages when doing protocol validation on an active testbed compared to doing it on a passive testbed.

Acknowledgment

We would like to thank the Interdisciplinary Institute for Broadband Technology in Flanders (IBBT) for supporting the creation of this testbed within the framework of the Wireless Building Automation (IBBT-WBA) and Deployment and Easy Use of wireless Services (IBBT-DEUS) projects.

References

[1] Kotz D., Newport C., Gray R. S., Liu J., Yuan Y., and Elliott C.,"Experimental evaluation of wireless simulation assumptions". 7th ACM intl. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Venice, Italy, pp. 78-82, October 04-06, 2004

[2] Werner-Allen G., Swieskowski P. and Welsh M.,"Motelab: A wireless sensor network testbed". 4th Annual Conference on Information Processing in Sensor Networks (IPSN) 2005, pp. 483-488, 15 April 2005

[3] Handziski V., Köpke A., Willig A. and Wolisz, A.,"TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks." In Proceedings of the 2nd international Workshop on Multi-Hop Ad Hoc Networks: From theory To Reality (REALMAN '06), Florence, Italy, pp. 63-70, May 26 - 26, 2006

[4] Alix 3C3 [online] http://www.pcengines.ch/alix3c3.htm

Demo Abstract: DHV - An Efficient Code Consistency Management Protocol for Wireless Sensor Networks

Thanh Dang and Nirupama Bulusu and Wu-chi Feng and SeungWeon Park
Department of Computer Science
Portland State University
Portland, OR, USA, 97201
Email: {dangtx,nbulusu,wuchi,swpark}@cs.pdx.edu

***Abstract*—Ensuring that every sensor node has the same code version is challenging in dynamic, unreliable, multi-hop sensor networks. When nodes have different code versions, the network may not behave as intended, wasting time and energy. We demonstrate DHV, a code consistency maintenance protocol to ensure that every node in a network will eventually have the same code. DHV is based on the simple observation that if two code versions are different, their corresponding version numbers often differ in only a few least significant bits of their binary representation. DHV allows nodes to carefully select and transmit only necessary bit level information to detect a newer code version in the network. DHV can detect and identify version differences in $O(1)$ compared to the logarithmic of current protocols in terms of messages and latency. Simulations and experiments on a real MicaZ testbed show that DHV reduces the number of messages by 50%, converges in half the time, and reduces the number of bits transmitted by 40-60% compared to DIP, the state-of-the-art protocol.**

I. Introduction

Experience with wireless sensor network deployments across application domains has shown that sensor node tasks typically change over time, for instance, to vary sensed parameters, node duty cycles, or support debugging. Typically, this leverages the ability to reprogram sensor devices using wireless communication. The goal of network reprogramming is to not only reprogram individual sensors but to also ensure that all network sensors agree on the task being performed.

Network reprogramming creates the need for a protocol to ensure that all network nodes have the same up-to-date code items, which can be binary images, code segments, or configuration parameters. We refer to such a protocol as a *Code Consistency Maintenance Protocol* (CCMP). Depending upon the application, the bandwidth and energy consumption of a CCMP can be comparable to sensing, particularly for networks with mobility or large churn. Key requirements for a CCMP are:

- A CCMP must ensure that the all network nodes *eventually have the same updated code*.
- A CCMP should enable a node with an old code item to discover a newer code item and update it with *low latency*.
- A CCMP should conserve energy and bandwidth.
- A CCMP should scale with both the network size and the total number of code items.

Code items are often represented as a set of *(key, version)* tuples. Each *key* uniquely identifies a code item. The corresponding *version* indicates if the code item is old or new (the higher the version, the newer the code). Previous CCMP protocols like DRIP [1] and DIP [2] incur high latency, high cost and may be complicated. Both DRIP and DIP are built on top of Trickle [3], a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In DRIP, a node randomly broadcasts each of its own *(key, version)* tuples to its neighbors separately. DRIP scales linearly with the total number of items ($O(T)$). DIP improves the total number of messages and latency to $O(log(T))$ by searching for a different *(key, version)* using hashes of the *(key, version)* tuples. A common feature in both approaches is that they try to detect and identify differences of versions as a whole.

We demonstrate a new code consistency maintenance protocol called *DHV*. We observe that *if two versions are different, they often only differ in a few least significant bits of their version number rather than in all their bits.* Hence, it is not necessary to transmit and compare the whole version in the network. DHV aims to detect and identify differences of version-levels for code items with the goal of transmitting much less data in the network compared to DRIP and DIP. The name DHV comes from the three steps in the protocol — Difference detection, Horizontal search, and Vertical search. Each step requires only $O(1)$ total messages and latency with respect to the total number of items. So DHV can identify a difference with as few as 3 transmissions.

II. The DHV Protocol

DHV has two main phases — *detection* and *identification*. In detection, each node broadcasts a hash of all its versions called a *SUMMARY message*. Upon receiving a hash from a neighbor, a node compares it to its own hash. If they differ, there is at least one code item with a different version number.

In identification, the *horizontal search* and *vertical search* steps identify which versions differ. In horizontal search, a

node broadcasts a checksum of all versions, called a *HSUM message*. Upon receiving a checksum from a neighbor, the node compares it to its own checksum to identify which bit indices differ and proceeds to the next step. In vertical search, the node broadcasts a bit slice, starting at the least significant bit of all versions, called a *VBIT message*. If the bit indices are similar, but the hashes differ, the node broadcasts a bit slice of index 0 and increases the bit index to find the different locations until the hashes are the same. Upon receiving a VBIT message, a node compares it to its own VBIT to identify the locations corresponding to the differing *(key, version)* tuples. After identifying which *(key, version)* tuples differ, the node broadcasts these *(key, version)* tuples in a *VECTOR message*. Upon receiving a VECTOR message, a node compares it to its own *(key, version)* tuple to decide who has the newer version and if it should broadcast its DATA. A node with a newer version broadcasts its DATA to nodes with an older version.

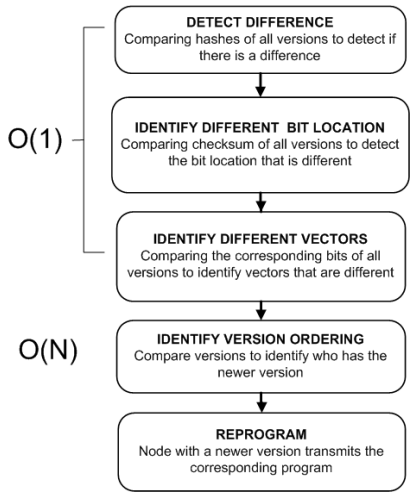


Fig. 1. DHV main components

Figure 1 illustrates the DHV protocol steps to complete a search to identify which vectors differ. DHV requires $O(1)$ in both latency and cost to identify all the differences, in contrast to DIP, which requires $O(log(T))$ in both latency and cost for searching.

III. Demonstration

We will demonstrate that the DHV protocol helps reduce network reprogramming latency and conserve energy consumption in communication. The demonstration is on a testbed of 16 MicaZ nodes. A code item can be updated through the default setting at boot time or by dynamically injecting packets from a laptop. A node will act as a base station to broadcast the packets injected from the laptop and sniff the traffic of the network. Based on the sniffed packets and the debugging LEDs on each node, we can determine the state of the network. We will demonstrate the protocol with a testing program for different scenarios. Readers are encouraged to look at the complete paper [4] for detailed results.

Network programming varying the number of total code items. In this scenario, we vary the total number of code items in the network. Users can choose to update one or more arbitrary code items. We will show that the DHV protocol helps discover if nodes need to update code in O(1) latency with respect to the total number of code items.

Network programming varying the number of new code items. In this scenario, we let users vary the number of new code items to be updated. We will show that the DHV protocol helps the network converge in $O(N)$ where N is the total number of new code items.

Network programming varying the number of nodes in a network. We will vary the number of nodes in the network and observe the total number of transmitted messages and the network convergence time. We will show that the total number of transmitted messages and the convergence time increase slightly with the total number of nodes in the network.

Network programming with nodes joining and leaving the network dynamically. We will also show that the network can quickly discover if there is a newly added node with older code versions and reprogram it accordingly. This is an important scenario showing that a CCMP and specifically the DHV protocol is necessary in real situations.

IV. Conclusion

We demonstrate the DHV protocol to maintain code consistency in wireless sensor networks. The key innovation in DHV is that it reduces the number of transmitted bytes in the network by carefully selecting and transmitting only absolutely necessary information at the bit level to detect and identify which code items need updates. Together with a carefully designed suppression mechanism, DHV is able to reduce the total number of messages significantly. Theoretically, DHV can identify differences with $O(1)$ complexity in the total number of items instead of logarithmically compared to DIP. Simulations and real-world experiments validate that DHV performs at least twice better than the state-of-the-art DIP protocol. We believe that DHV can not only be used in wireless sensor networks but also in other distributed applications that require data consistency. The preliminary version of the DHV source code can be downloaded at http://sys.cs.pdx.edu/home/dhv.

References

[1] G. Tolle and D. Culler, “Design of an application-cooperative management system for wireless sensor networks,” in *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, Istanbul, Turkey, 2005.

[2] K. Lin and P. Levis, “Data discovery and dissemination with dip,” in *Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (IPSN 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 433–444.

[3] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation (NSDI 04)*. Berkeley, CA, USA: USENIX Association, 2004, pp. 2–2.

[4] T. Dang, N. Bulusu, W. chi Feng, and S. Park, “Dhv: A code consistency maintenance protocol for multi-hop wireless sensor networks,” in *Proceedings of the sixth the European conference on Wireless Sensor Networks, EWSN 09*, Cork, Ireland, February 11–13, 2009.

Demo Abstract: Software Factory for Wireless Sensor Networks

T. Naumowicz, B. Schröter, and J. Schiller

Freie Universität Berlin, Institute of Computer Science
{naumowic, schroete, schiller}@inf.fu-berlin.de

***Abstract*—Wireless Sensor Networks (WSNs) are not widely used in field research because of poor tool support and high complexity. To address this issue, we designed and developed a Software Factory for WSNs that hides the complexity of software development for embedded systems and exposes a visual domain-specific modeling language. Our solution makes WSNs more attractive as a tool for researchers from outside the computer science field and enables a wider adoption of WSNs in field research.**

We will demonstrate our Software Factory in a real-world scenario replicating our latest WSN deployment.

***Index Terms*— Wireless Sensor Network, Software Factory, Domain-Specific Languages**

I. Introduction

The vast majority of publications in the field of WSNs refer to environmental or habitat monitoring as the typical application for WSNs. You could expect that WSNs are already widely used in field research because they are often advertised with high sensing accuracy, long runtimes, and easy deployment. However, this is still not the case.

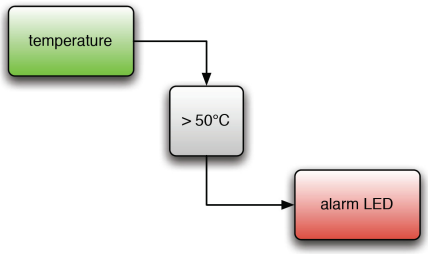


Figure 1: A conceptual example of a simple data flow

Research in the area of WSNs has previously focused on hardware design, self-organization, various routing algorithms, or energy saving patterns. This trend is already changing [1, 2] but the available tools typically target experienced software developers and not researchers from outside the computer science field. As of today, domain experts such as environmental scientists need extensive support from hardware and software engineers during planning, deploying and management of WSNs. This makes a wide adoption of WSNs in real-world scenarios difficult and, in combination with poor tool support, makes such adoption slow and error prone.

We have designed and developed a Software Factory to address this gap. Software Factories are model-driven development environments. Our solution is based on a data-centric programming model where data flows are used to describe how data should be processed (e.g. Figure 1). We will refer to our Software Factory as *Flow*.

We prototyped *Flow* for the resource constrained ScatterWeb WSN platform MSB-430 [3] with 55kB flash memory and 5kB RAM. The prototype is available for download and evaluation [4].

II. The Software Factory in Brief

Flow provides a visual editor for modeling of applications for WSNs and a native code generator. *Flow* focuses on the visual representation and execution of data flows at a very high abstraction level (e.g. Figure 2). It is not a visual programming language for describing program control flow – in such scenarios the textual representation is very often easier to be written and understood.

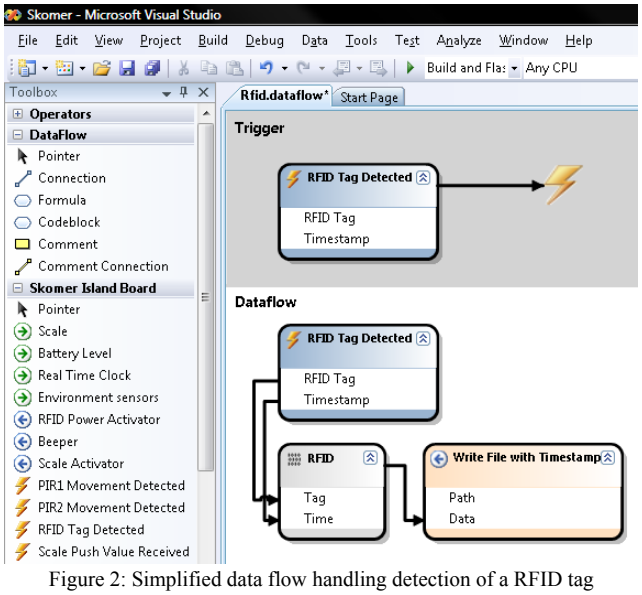


Figure 2: Simplified data flow handling detection of a RFID tag

Our Software Factory does not leverage a Virtual Machine. With this approach, we expect to achieve higher execution performance. The use of native C application code has an important advantage: it makes the

implementation of drivers for *Flow* less time-consuming and allows easy integration of specialized C code blocks to handle advanced cases.

Domain experts use the provided visual Domain-Specific Language (DSL) to define data flows and thus specify the behavior of nodes in the network. The modeled data flows are validated at design time and the user is provided with visual feedback. After successful validation, application code is generated. With *Flow,* domain experts no longer need to develop or maintain native C application code.

III. Real-World Scenario: Skomer Island

We deployed a WSN for environmental monitoring on Skomer Island in the United Kingdom in March 2007 [5]. The WSN was used to monitor the behavior of Manx Shearwaters, a burrow nesting seabird. We recorded birds' activity around entrances to the burrows as well as the identity of tagged individuals. In addition, we collected high resolution environmental data about the temperature and humidity inside and outside the burrows. In March 2008 an updated version of the system was deployed.

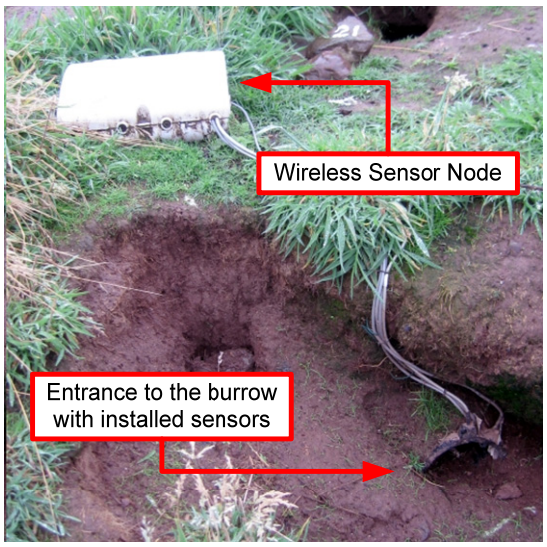


Figure 3: Sensor Node deployed on Skomer Island in 2008

During the deployments we observed that our expertise in software development for embedded systems was required in every phase of the project. The research goals shifted often and we had to update the deployed applications each time. High complexity of required changes prevented domain experts from modifying the application code themselves. The dependency on us was a big disadvantage – it resulted not only in delays but also distracted scholars in their research.

We use the Skomer Island scenario in our research to investigate how high levels of programming abstraction can be used in order to make WSNs more attractive as a tool in areas outside the computer science field.

IV. Demonstration Setup

The demonstration is built from a subset of our real-world WSN deployment to the Skomer Island in 2008. The functionality of the WSN from the deployment is now modeled using *Flow*. The WSN is enabled, i.e. conference attendees can interact with the installed demonstration. Visitors are encouraged to modify data flows using *Flow*, deploy the generated code, and evaluate the results.

The demonstration consists of the following components:

1) Wireless sensor node used during the expedition to the Skomer Island. The node is in its original packaging from the deployment (see Figure 3) and it's equipped with environmental sensors, movement detectors, and a RFID reader.
2) Wireless data logger node used as a data sink on the network. The node collects data from the network and stores it on a SD card.
3) Wireless gateway node attached to a notebook. The node provides access to the network for .NET based applications executing on the PC.
4) Notebook with installed *Flow* and .NET development environment. The notebook is used to model data flows and to present the capability of easy integration of WSNs with end user applications.

Interested visitors use the provided notebook to modify existing data flows and design their own ones. This is how they are able to modify the behavior of the network. *Flow* generates native code for the sensor nodes based on the designed data flows and compiles it for the selected hardware platform. Proxy libraries for the PC are generated as well. This enables easy access to the data in the WSN from the PC.

The generated code handles network discovery, gateway resolution, data communication and marshaling between the WSN and the client application. Visitors can access data and activate data flows using easy to handle .NET objects that represent wireless sensors in the network.

V. Conclusions

The proposed demonstration illustrates the achieved level of programming abstraction. Data flows can be easily created, read and modified by researchers with no experience in development for embedded systems.

In our opinion, the proposed solution has the potential to shorten the development cycles dramatically during WSNs deployments in field sciences, to reduce the dependency on hardware and software engineers, and to lead to a wider adoption of WSNs outside the computer science field.

References

[1] Martinez, K., Padhy, P., Riddoch, A., Ong, R., Hart, J.: Glacial Environment Monitoring using Sensor Networks. In: Proceedings of the Workshop on Real- World Wireless Sensor Networks (REALWSN'05), Stockholm, Sweden (2005).

[2] Talzi, I., Hasler, A., Gruber, S., Tschudin, C.: PermaSense: Investigating Permafrost with a WSN in the Swiss Alps. In EmNets 2007, Cork (2007).

[3] ScatterWeb, http://cst.mi.fu-berlin.de/projects/ScatterWeb

[4] Flow, http://cst.mi.fu-berlin.de/projects/flow

[5] Naumowicz, T., Freeman, R., Heil, A., Calsyn, M., Hellmich, E., Braendle, A., Guilford, T., Schiller, J.: Autonomous Monitoring of Vulnerable Habitats using a Wireless Sensor Network. In REALWSN'08, Glassgow, UK (2008).

Demo Abstract: Mountainview – Precision Image Sensing on High-Alpine Locations

Matthias Keller, Jan Beutel, Lothar Thiele
ETH Zurich, Computer Engineering and Networks Lab
8092 Zurich, Switzerland
Email: kellmatt,beutel,thiele@tik.ee.ethz.ch

***Abstract*— We describe a novel optical sensor system designed for high-fidelity image acquisition in wireless sensor networks. The system uses precision optical equipment as well as a high-resolution imager yielding accurate observations - a prerequisite for many image processing techniques. In order to deliver the lifetime and durability required by long-term environmental monitoring campaigns we present a custom solution that is optimized to integrate seamlessly into a power efficient wireless sensor network. The imaging system presented is based on a state-of-the-art wireless digital SLR camera, that has been packaged, augmented with the necessary power supply, heater, system supervision and control to operate reliably under the most hostile and extreme environmental conditions.**

I. Introduction

The PermaSense project [1] strives for collecting geo-physical data in the high-altitude environment of the Swiss Alps with a wireless sensor network (WSN). To this day, a WSN consisting of one base station and 16 motes has been successfully deployed on the Matterhorn at 3.500 meters above sea level and ambient temperatures down to -40°C since July 2008. In this deployment, simple analog and digital sensors representing properties such as temperature, humidity or electrical resistivity are sampled every few minutes. While this sensor data is highly valuable for the geophysical research of our joint project partner, there are contexts that require more complex data, e.g. fidelity and rates two or three orders of magnitudes higher than the current set of simple sensors.

In this demo we present a novel, high-precision wireless image sensor for the PermaSense project. This sensor allows to assess temporal variability in the snow cover as well as rock and ice movement with great detail. For the application of advanced image processing algorithms and a meaningful analysis of the image data gathered from the environment, the fidelity and optical precision are of the utmost importance. For this purpose a digital SLR camera is mounted in a ruggedized housing. The housing includes an optically corrected lens port, heating system, system supervision, power supply and integrated into our wireless sensor network infrastructure. The main challenges are reliability concerns, issues concerning power efficiency and the available energy supply, data throughput, image quality as well as flexibility in the deployment location to obtain an optimal field of view of the inspection site. The system design is optimized to operate on a very low duty cycle, i.e. with a few pictures per day. It should still allow for a flexible and quick wakeup upon the request of a user or another sensor.

II. System Concept

The general concept of our sensor system is depicted in Fig. 1. A Nikon digital SLR camera with interchangeable lenses is housed in a steel enclosure. A lens port, derived from an underwater still camera housing, using non-reflecting, optical glass opens the field of view for the camera. Since the anticipated data volume per image (∼11MB RAW, ∼6MB JPEG) by far exceeds the performance of any low-power, mote-based WSN the camera uses a dedicated WLAN link to transmit images to a base station. An additional heater allows to defog the lens port and heat the camera/lens to an acceptable ambient level. Since all three consumers consume considerable amounts of energy a power switch controlled by a low-power TinyNode sensor node allows to power-cycle the individual components when not in use. Effectively, this reduces standby power consumption to an acceptable minimum. Furthermore, the sensor node serves as an internal health indicator providing regular ambient temperature, humidity and battery fill levels. Depending on the intended application scenario the sensor system can either be powered from an integrated battery allowing to take a finite amount of pictures or by a solar system (solar panel, charger, rechargeable battery).

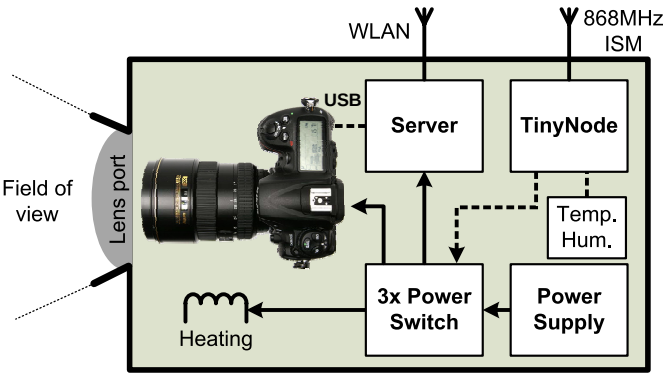


Fig. 1. The Mountainview sensor contains a professional digital SLR camera, a heater and an embedded computer with WLAN connectivity. Power consumption is controlled and monitored via a TinyNode sensor node. An adjustable mount supports different lenses and focal points for enhanced image quality.

III. Implementation

The Mountainview prototype is based on a Nikon D300 DSLR with a 12.3 MegaPixel image sensor. An attached Gumstix embedded computer is used for camera control. Adding a WLAN interface allows to control the camera, download thumbnails and full images over the Internet from a remote computer. A detailed power profile of the camera at startup and during image capture is given in Figure 2. The camera consumes an average of 220 mA with peak currents up to 3 A at 9V. The power consumption of the embedded computer is in a similar range, measured at 3 W average when connected to a WLAN access point. Alternative WLAN microservers, e.g. optimized for long-range point-to-point directional links are known to require even more power with about 10-20W average power consumption.

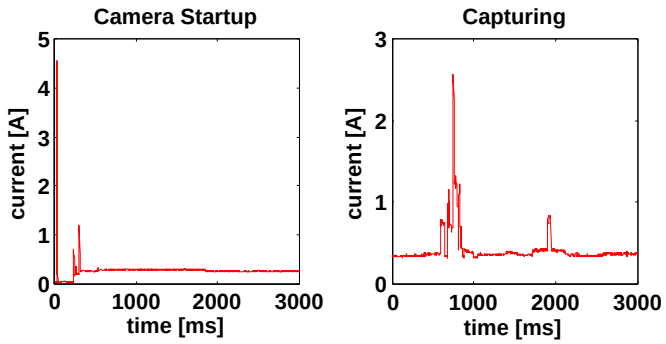


Fig. 2. Current draw during camera startup (left), image capture and storage.

It is thus necessary to power cycle all power hungry consumers and only enable them when their function is required. By using a TinyNode sensor node running a custom version of the ultra-low power application Dozer [2] the power consumption is reduced to a mere 0.148mA. By sending a broadcast packet over the wireless sensor network the user can control when and for how long to enable heater, camera and embedded computer. The monitoring of temperature, humidity and battery fill levels is not only a necessity while operating the camera/heater but also when the system is in long sleep phases to alert in case of water ingress. A simple timer to wake up the system, transmit an image and go back to sleep is not an option as it would (i) limit the flexibility/reactivity of the system and (ii) lack the online system supervision that is critical to successful long-term environmental monitoring, especial in the environmental extremes of PermaSense.

IV. Related Work

Depending on the application, the desired geophysical data can also be gathered by aircraft or satellites [3], [4]. However, there are certain constraints concerning costs, availability, spatial resolution and applicability of remote sensing techniques. For the price of a smaller covered area, our approach focuses on a high spatial resolution and more flexibility as the later application is virtually only dependent on the number and location of the installed cameras.

Sharing the same motivation, Kodak DC20 cameras with a timer-based trigger mechanism for automatic image gathering are used in [5]. In contrast, our work first of all offers a much higher image quality concerning image resolution and optical system used. Additionally, our prototype allows remote control and near real-time data access.

Several projects in the WSN field have proposed the usage of image sensors for sensing, control and verification purposes. These often low-resolution CMOS camera modules or webcams are directly attached to sensor nodes [6], [7]. These applications have quite different requirements concerning image quality, installation location, density and durability when compared to our project.

V. Conclusions and Future Work

The current progress of our work promises a great tool for geophysical and possibly other scientific experiments at a new level of detail and interactive control for reasonable costs. While our current experiments have only been carried out in our lab, we are intending to deploy our prototype on the Matterhorn field site. A detailed assessment of the optimal duty cycle is currently under investigation. However, it is likely to depend much on the actual requirements of system operation at the field site as the main consumers of power (heater, WLAN) depend on the actual circumstances at the site. Future work encompasses the use of smaller/different sensors for triggering the main sensor and 3-D modeling using multiple cameras.

Acknowledgment

The work presented was supported by NCCR-MICS, a center supported by the Swiss National Science Foundation under grant number 5005-67322. We also thank Stefan Kronig and Michel Müller for their work on the prototype implementation.

References

[1] A. Hasler, I. Talzi, J. Beutel, C. Tschudin, and S. Gruber, "Wireless sensor networks in permafrost research - concept, requirements, implementation and challenges," in *Proc. 9th Int'l Conf. on Permafrost (NICOP 2008)*, vol. 1, pp. 669–674, June 2008.

[2] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: ultra-low power data gathering in sensor networks," in *Proc. 6th Int'l Conf. Information Processing Sensor Networks (IPSN '07)*, pp. 450–459, ACM Press, New York, Apr. 2007.

[3] W. Rosenthal and J. Dozier, "Automated mapping of montane snow cover at subpixel resolution from the Landsat Thematic Mapper," *Water Resources Research*, vol. 32, no. 1, pp. 115–130, 1996.

[4] W. Smith, D. Zhou, F. Harrison, H. Revercomb, A. Larar, H. Huang, and B. Huang, "Hyperspectral remote sensing of atmospheric profiles from satellites and aircraft," in *Proc. SPIE*, vol. 4151, p. 94, SPIE, Bellingham, MA, 2001.

[5] J. Hinkler, S. Pedersen, M. Rasch, and B. Hansen, "Automatic snow cover monitoring at high temporal and spatial resolution, using images taken by a standard digital camera," *International Journal of Remote Sensing*, vol. 23, no. 21, pp. 4669–4682, 2002.

[6] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior, and M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proc. 3rd ACM Conf. Embedded Networked Sensor Systems (SenSys 2005)*, pp. 192–204, ACM Press, New York, 2005.

[7] A. Basharat, N. Catbas, and M. Shah, "A framework for intelligent sensor network with video camera for structural health monitoring of bridges," in *Proc. 3rd IEEE Int'l Conf. Pervasive Computing and Communications (PerCom 2005)*, Mar. 2005.

Demo Abstract: Towards Interoperability Testing for Wireless Sensor Networks with COOJA/MSPSim

Joakim Eriksson, Fredrik Österlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt
Swedish Institute of Computer Science
{joakime,fros,nfi,nvt,adam,thiemo}@sics.se
Robert Sauter, Pedro José Marrón
University of Bonn and Fraunhofer IAIS
{sauter,pjmarron}@cs.uni-bonn.de

***Abstract*—Wireless sensor networks are moving towards emerging standards such as IP, ZigBee and WirelessHART which makes interoperability testing important. Interoperability testing is today performed through black-box testing with vendors physically meeting to test their equipment. For interoperability testing of these standards, a white-box approach is desired, since it gives both details on performance and clues about why tests succeeded or failed. To enable white-box testing, we propose a simulation-based approach. We extend our MSPSim emulator and COOJA wireless sensor network simulator to support interoperable simulation of sensor nodes with firmware from different vendors. To show the benefits of white-box interoperability testing, our demonstration runs Contiki and TinyOS nodes in a single simulation and demonstrates power profiling over both operating systems.**

I. Introduction

Recently, wireless sensor networks have proven their industrial potential and, consequently, a number of standards have emerged. These include WirelessHART, ZigBee, and IPv6-based sensor networks [3]. The standardization of sensor networks makes interoperability a new requirement when implementing communication stacks and applications. Interoperability is difficult to verify, however, since sensor network simulators and prototyping tools lack the possibility to simulate heterogeneous sensor networks. Many sensor network simulators do not simulate sensor node operating systems. The others only simulate nodes running one specific operating system. These tools are thus not able to perform interoperability tests. Instead, vendors are today required to physically meet to perform black-box testing of their equipment. Black-box testing can test interoperability but does not provide detailed information about the node internals during testing.

To meet the need for white-box testing of interoperability, we extend the Contiki simulator COOJA [5] and MSPSim [4], an instruction level emulator that is integrated in COOJA. Our simulator extensions enable simulations of heterogeneous sensor networks that consist of nodes running the TinyOS and Contiki operating systems. We also extend MSPSim with a power profiling mechanism similar to the software-based power profiler of the Contiki operating system [2]. Our experiments demonstrate that COOJA/MSPSim is not only a feasible tool for interoperability testing; it is also able to power profile nodes running different operating systems.

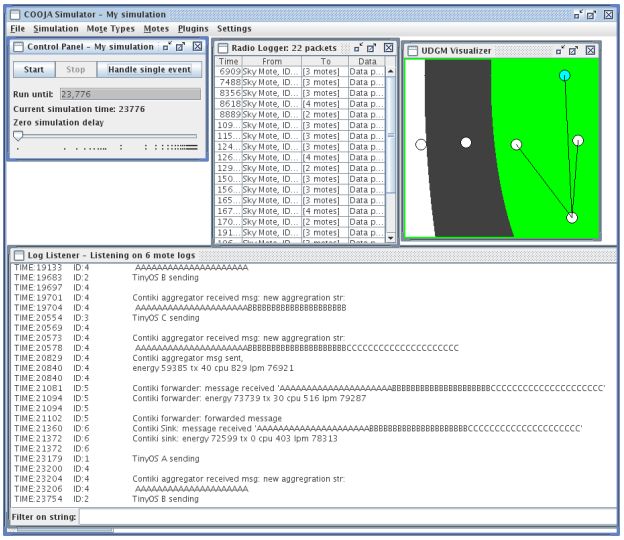


Fig. 1. COOJA/MSPSim simulating a hierarchical, heterogeneous sensor network of TinyOS and Contiki nodes

II. Background and Implementation

COOJA [5] is a flexible Java-based simulator initially designed for simulating networks of sensors running the Contiki operating system. COOJA enables simultaneous simulation at different levels. MSPSim [4] is a Java-based instruction level emulator of the MSP430 microprocessor series. In contrast with CPU-level emulators, it emulates complete sensor networking platforms such as the Tmote Sky and ESB/2 [6].

For this demonstration, we integrate MSPSim more closely into COOJA. We add COOJA support for also simulating TinyOS nodes which required some COOJA changes to handle node IDs since TinyOS uses two IDs per node; one for the node itself and one as a mutable Active Message address. In MSPSim, we made the emulation of the MSP430 and the CC2420 radio chip more accurate and complete. The initial emulation was limited to the subset of features used in Contiki.

Contiki has an on-line power profiling mechanism [2] that estimates the energy consumption by measuring the duration each component is in various modes, e.g. the time the CPU spends in low-power mode. Using this mechanism, all emulated nodes can be power profiled in COOJA/MSPSim.

To power profile non-Contiki firmwares in COOJA we extended MSPSim with more detailed statistics for external components such as the CC2420 radio chip. The mechanism is similar to the built-in power profiling mechanism in Contiki. The information can be accessed per node from COOJA when power profiling is needed.

III. Evaluation

A. A Heterogeneous, Hierarchical Sensor Network

To demonstrate interoperability between Contiki and TinyOS in COOJA/MSPSim, we simulate a heterogeneous, hierarchical sensor network shown in Figure 1. The hierarchical sensornet consists of both TinyOS and Contiki nodes. The three TinyOS nodes placed in the right part of the top right plug-in in Figure 1 act as data sources that periodically send a data message with 20 bytes payload (“As”, “Bs” or “Cs”) to an aggregator node running Contiki. The aggregator node aggregates the data and sends it via a forwarder node in a multi-hop fashion to the sink (left-hand node in the plug-in). Both the forwarder and the sink node run Contiki.

Since the TinyOS frame format differs from the Contiki frame format, we have modified the Rime stack and the Chameleon module [1] to allow Contiki to understand packets from the TinyOS nodes. Our experiments have demonstrated successful interaction between TinyOS and Contiki nodes and the ability to measure the radio duty cycle for both TinyOS and Contiki nodes.

B. Interoperability Tests

To validate COOJA/MSPSim’s capability for interoperability tests we perform an experiment where we develop the same networked application in both TinyOS and Contiki.

We implement an application that broadcasts packets to its neighbors and when it receives packets it counts the neighbors and visualizes the neighbor count. The application for TinyOS uses the standard Active Message communication framework and the Contiki application is designed to replicate the TinyOS application. Since TinyOS uses IEEE 802.15.4 we replace the default Contiki MAC protocol with IEEE 802.15.4 and add the TinyOS active message type as the first byte of the payload.

Figure 2 shows the result of the test. All nodes communicate and count neighbors as expected. This interoperability test validates a very basic application protocol on top of 802.15.4 and shows that it is possible to perform interoperability tests using COOJA/MSPSim.

IV. Conclusions

In this demo we present the COOJA/MSPSim simulator. COOJA/MSPSim enables white-box testing which makes interoperability testing more practical and transparent by allowing repeatable and fine-grained control of experiments. COOJA and MSPSim can be downloaded from `http://www.sics.se/contiki/` and `http://www.sics.se/project/mspsim/`.

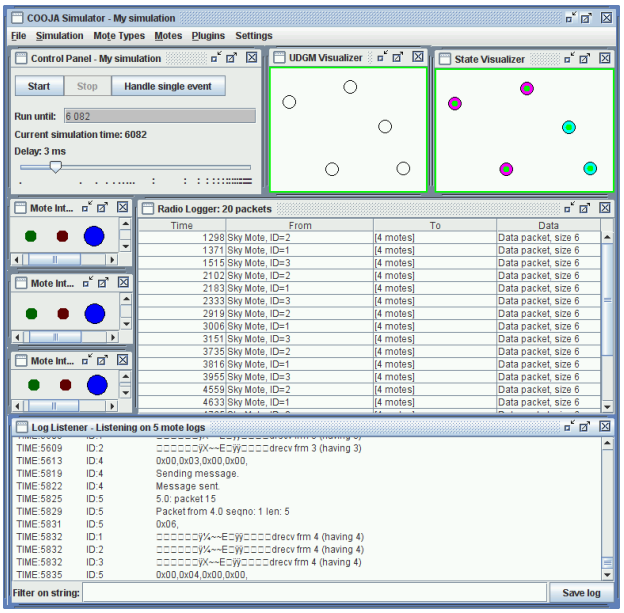


Fig. 2. Basic interoperability test with three TinyOS nodes and two Contiki nodes communicating. Some of the nodes led-panels are shown, and debug printouts are visible in the log window.

V. Acknowledgments

This work has been partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053. This work was also partly financed by VINNOVA, the Swedish Agency for Innovation Systems.

References

[1] A. Dunkels, F. Österlind, and Z. He. An adaptive communication architecture for wireless sensor networks. In *Proceedings of the Fifth ACM Conference on Networked Embedded Sensor Systems (SenSys 2007)*, Sydney, Australia, November 2007.

[2] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He. Software-based online energy estimation for sensor nodes. In *Proceedings of the Fourth Workshop on Embedded Networked Sensors (Emnets IV)*, Cork, Ireland, June 2007.

[3] M. Durvy, J. Abeillé, P. Wetterwald, C. O’Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of the Sixth ACM Conference on Networked Embedded Sensor Systems (ACM SenSys 2008)*, Raleigh, North Carolina, USA, November 2008.

[4] J. Eriksson, A. Dunkels, N. Finne, F. Österlind, and T. Voigt. MSPSim – an extensible simulator for MSP430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, January 2007.

[5] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with COOJA. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, November 2006.

[6] J. Schiller, H. Ritter, A. Liers, and T. Voigt. Scatterweb - low power nodes and energy aware routing. In *Proceedings of Hawaii International Conference on System Sciences*, Hawaii, USA, 2005.

Poster Abstract: Rupeas - An Event Analysis Language For Wireless Sensor Network Traces

Matthias Woehrle [#], Christian Plessl [*], Lothar Thiele [#]

[#]*ETH Zurich, Computer Engineering and Networks Lab*
8092 Zurich, Switzerland
[*]*University of Paderborn, Paderborn Center for Parallel Computing*
33102 Paderborn, Germany
matthias.woehrle@tik.ee.ethz.ch

***Abstract*— Wireless Sensor Networks (WSNs) are distributed systems of embedded sensor nodes loosely coupled by wireless communication. Debugging, profiling and testing of these systems is inherently challenging due to resource constraints of the system components, a vast state space and tight integration with the environment. A prominent validation strategy is to use testbeds running an instrumented version of the software on distributed sensor nodes and to collect data during execution for off-line analysis. We provide a novel programming language named Rupeas which especially targets the analysis of these WSN logs.**

I. Introduction

Testing of WSNs is a crucial part of the development process as a functional, performing and resource effective system at deployment time is of utmost importance [6]. A main methodology for testing is using a WSN specific test platform such as a testbed [1] or simulator [2] and logging test specific information. Tests result in a substantial amount of log data, which needs to be analyzed in the context of the test platform and the application. Such an analysis requires adequate tools.

With *Rupeas*, a **R**uby **P**owered **E**vent **A**nalysi**s** language, we provide a tool for WSN analysis. It facilitates analysis of WSN logs by using an event abstraction for representing the execution traces. It provides operators to formulate queries on sets of events. Rupeas provides a high ease of use with its simple, concise notation. It greatly simplifies WSN trace analysis and testing, which are often perceived as meticulous tasks.

As an example, an interesting analysis of typical WSN applications for data gathering is whether and to what extent sensory data can be extracted from the system. Typical questions are: What is the data yield from each node? What do the routing paths look like? Are the routing paths efficient?

These are sample questions that can be answered by Rupeas queries. Rupeas allows for determining routing paths by formulating a relation on send and receive events gathered in traces, even for multi-sink systems as shown in Fig. 2.

Previously proposed approaches cannot help with this analysis: Diagnostic simulation [4] using data mining techniques on simulation data, can help you with outlier detection, but does not provide a comprehensive analysis. Assertions on distributed, global state [3], [5] can help you identify problems in parent selection of the routing protocol, but do not help in the analysis of taken routes.

II. Rupeas

Rupeas is based on ideas first introduced in the EvAnT framework [7] for analyzing WSNs. Both, Rupeas and EvAnT, use an event abstraction: Individual log entries are events, tuples of key-value pairs; logs are represented as a set of events. Both are based on four generic operators for processing events sets and allow for formulating event set queries and assertions for testing.

Rupeas provides two major enhancements over EvAnT: Firstly, Rupeas is a novel programming language for WSN analysis and testing, a Domain Specific Language (DSL). The EvAnT framework merely provides limited API's on top of a general purpose language, which lacks a focus on the problem domain of the analysis of WSN logs. Secondly, Rupeas introduces descriptions for the event types in the trace to specify format and expected content of each event.

Rupeas is especially designed to allow for an intuitive formulation of queries on event sets. As a DSL, Rupeas is a language specifically designed for the given problem domain, capturing its essence and neglecting any general purpose aspects. Rupeas is an internal DSL embedded in Ruby. Ruby as a host allows for elaborate analysis of the processed event sets and integration into a comprehensive test framework [6].

A Rupeas program features two main sections:

(1) The *event description* section specifies keys and expectations on values for the each type of event. The individual events are verified when being loaded from the trace. One specific feature is to provide periodic value types for events, which are used for wrapping counters such as sequence numbers to allow for determining a total order. An abbreviated description of send events in Rupeas is shown in Figure 1.

(2) The *event processing* specifies the processing that is applied to the event set. The DSL is designed to facilitate transformations and queries of event sets by means of event set operators using a simple and concise notation. An example for such processing is the analysis of routing paths. Starting with a single packet originating at a source node, Rupeas' fixed point operator allows for iterative merging of send and receive events. After the fixed point is reached, the merged

Event Declaration

```
Type :senddone do
 with :name =>:seqNo, :fieldtype=> :periodic, :range => 0..255
 with :name =>:origin, :fieldtype => :integer, :range => 0..100,:notification => :error
 with :name =>:destination, :fieldtype => :integer, :range => 0..100, :notification => :warning
 with :name =>:nodeid,:fieldtype=> :integer, :range => 0..100,:notification => :error
 with :name =>:time, :fieldtype => :float, :notification => :warning
end
```

Event Transformation

```
routes = routestart.transform(:iterative) do |sent, receive|
  constraint sent[:origin] == receive[:origin]
  constraint sent[:seqNo] == receive[:seqNo]
  select sent[:nodeid] == receive[:nodeid] and receive[:type] == :route and sent[:type] == :senddone
  select sent[:dest] == receive[:nodeid] and sent[:type] == :route and sent[:type] == :receive
  merge :type => :route, :nodeid => receive[:nodeid], :seqNo => sent[:seqNo], :origin => sent[:origin]
end
```

Log file

```
type seqNo origin dest node time
receive 0 50 51 51 6.009753
senddone 0 50 51 50 6.009921
receive 0 34 44 44 6.009966
senddone 0 34 44 34 6.010134
receive 0 72 62 62 6.010889
receive 0 28 27 27 6.010983
senddone 0 72 62 72 6.011057
senddone 0 28 27 28 6.011151
receive 0 12 13 13 6.011532
senddone 0 12 13 12 6.011700
receive 0 37 100 100 6.021597
senddone 0 12 13 12 6.011700
receive 0 37 100 100 6.021597
senddone 0 37 100 37 6.021765
receive 0 76 77 77 6.023131
senddone 0 76 77 76 6.023299
receive 0 31 41 41 6.025161
senddone 0 31 41 31 6.025329
receive 0 60 61 61 6.025801
```

Fig. 1. Rupeas event description of :senddone events, transformation syntax of fixed point operator and a sample of the according trace.

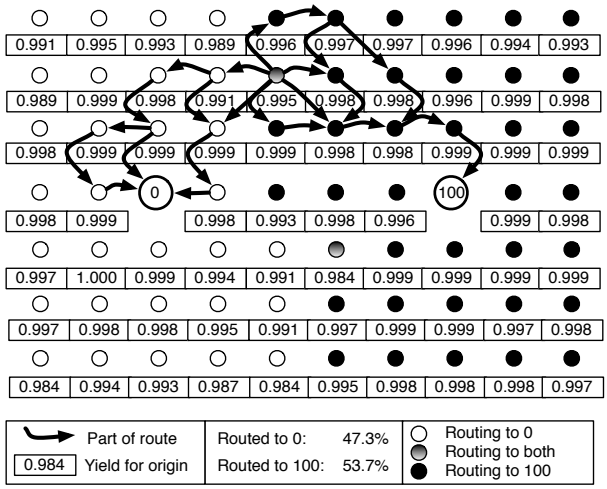


Fig. 2. Visualization of Rupeas results.

send-receive pairs represent all routing paths with detailed information on the exact paths taken for each origin node and sequence number.

III. Case Study

As a preliminary example for an analysis of a data gathering application, we apply Rupeas to data from simulating a TinyOS 2.x application. We instrument the Collection Tree Protocol to log sent and received packets and simulate the system in TOSSIM [2]. The simulation topology is a 70 node grid (cf. Fig. 2) including *two sink nodes* (0, 100). The simulation uses gain and noise models based on USC's Realistic Wireless Link Quality Model and Generator [1]. The log file and input for Rupeas captures data from a 6 hour run resulting in over 2 million events.

The analysis features two basic steps: Loading the event trace using the event description and using the fixed point processor(cf. Fig. 1). Rupeas allows for filtering routing paths for final destination, route selection or hop count. The data can be easily extracted with Ruby into different file formats. The main results obtained are visualized in Figure 2. It depicts the yield of individual origin nodes and their associated sink. The data yield of the simulation is high and the traffic is routed evenly among the sinks. As an example, Figure 2 displays the links of nine different routing paths actually taken in the experiment for node 24. The packets are routed via minimally two and maximally six intermediate hops. 50.4% of the packets are routed to sink 0.

This analysis is only a single example of Rupeas capabilities. Further details on Rupeas are available on the Rupeas project page[2].

IV. Summary

We presented Rupeas, a novel language targeted at analyzing WSN logs. The DSL provides a high ease-of-use by capturing the essence of the WSN test domain. Rupeas is an open-source project trying to facilitate testing and thereby stimulating testing efforts in the WSN community.

V. Acknowledgments

The work presented here was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

[1] M. Dyer et al. Deployment support network - a toolkit for the development of WSNs. In *Proc. EWSN 2007*, pages 195–211, 2007.
[2] P. Levis et al. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proc. SenSys 2003*, pages 126–137, 2003.
[3] M. Lodder et al. A global-state perspective on sensor network debugging. In *Proc. HotEmNets 2008*, 2008.
[4] M. Maifi et al. Towards diagnostic simulation in sensor networks. In *Distributed Computing in Sensor Systems*, pages 252–265. Springer, 2008.
[5] K. Römer and M. Ringwald. Increasing the visibility of sensor networks with passive distributed assertions. In *Proc. REALWSN '08*, 2008.
[6] M. Woehrle et al. Increasing the reliability of wireless sensor networks with a distributed testing framework. In *Proc. EmNetS 2007*, pages 93–97, 2007.
[7] M. Woehrle et al. EvAnT: Analysis and checking of event traces for wireless sensor networks. In *Proc. SUTC 2008*, pages 201–208, 2008.

[1] http://anrg.usc.edu/www/index.php/Downloads

[2] http://code.google.com/p/rupeas/

Poster Abstract: A Knowledge Based Wireless Sensor Network

Canada-Bago J., Gadeo-Martos, M. A., Fernandez-Prieto, J. A.
Telecommunication Engineering Department
University of Jaen (Spain)
jcbago@ujaen, gadeo@ujaen.es, jan@ujaen.es

Velasco-Perez, Juan R.
Department of Automatica
University of Alcala (Spain)
juanra@aut.uah.es

***Abstract* — Nowadays, there is a trend to include more and more functionality in sensors; in this sense, some works about intelligent sensors have been presented. However, few attentions have been paid to knowledge based sensors. In this work we present a knowledge based WSN in order to control a plague in agriculture. Each sensor of the network has a Fuzzy Rule Based System, which consists of a knowledge base composed of a set of IF-THEN fuzzy rules, fuzzyfication and defuzzyfication interfaces, and an inference engine adapted to the sensor. Before designing the WSN, a set of experiments was carried out to measure the performance of the intelligent sensor in different conditions. The results show that the sensor can execute properly this kind of knowledge based system, and that it is possible to use it in a large number of applications.**

***Index Terms* — Fuzzy logic, intelligent sensors, knowledge based system.**

I. Introduction

Intelligent sensors, in which artificial intelligence plays a key role, have been studied in recent years, for instance, Deckneuvel[1] and Shan[2]. There are a few types of artificial intelligence systems that can be used in sensors, such as neural network, fuzzy logic and a hybrid neuro-fuzzy system. Although some studies have presented sensors that use fuzzy logic such as Benoit et al. [3], few attentions have been paid to knowledge based sensors. In this work, a complete Fuzzy Rule Base System (FRBS) (figure 1), adapted to a 32 bit sensor, has been designed. The FRBS is composed of a knowledge base (linguistic fuzzy variables and rules), interfaces, and an inference engine. Besides, some tests have been carried out in order to measure its performance. Finally, a WSN application of intelligent sensor with FRBS is presented.

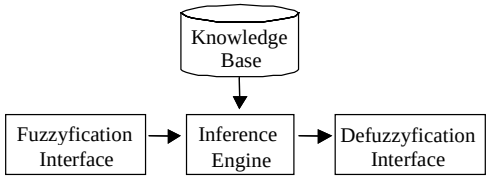


Fig. 1. Inner structure of Fuzzy Rule Based System

II. Intelligent Sensor Performance

In order to verify the performance of the sensor, some tests have been carried out with a knowledge base in which the number of rules (figure 2) and the number of antecedents of the rules (figure 3) are modified. Figure 2 shows the number of inferences per second versus the number of rules in the knowledge base.

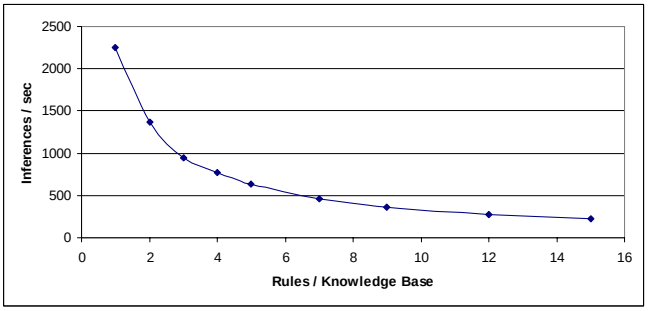


Fig. 2. Inferences/second vs. number of rules in the knowledge base

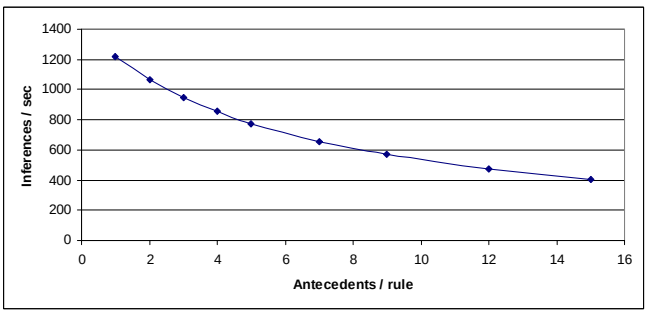


Fig. 3. Inferences/second vs. the number of antecedents in the rules

As can be observed, the intelligent sensor executes 223 inferences per second (in a knowledge base with 15 rules), so the sensor reaction time is 4.4 ms, which is sufficient for a large number of applications.

III. Application

A control system of plagues in the culture of the olive tree is shown as application of the knowledge based WSN. The

development of the olive tree fly is strongly related to the conditions of temperature and humidity of the environment.

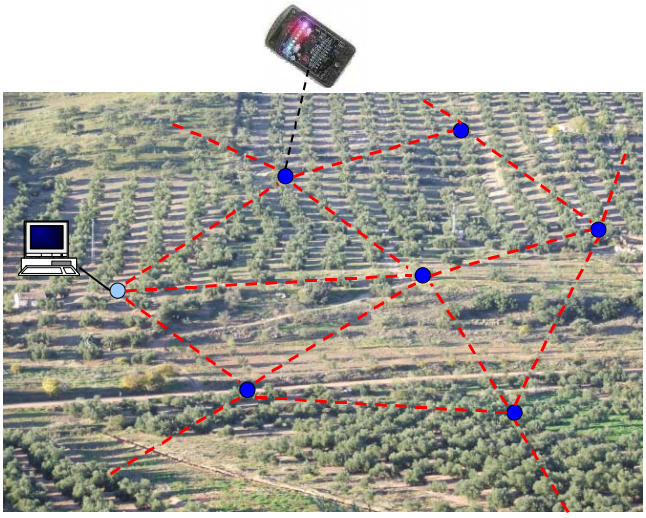


Fig. 4. The knowledge base WSN. Environment: land of olives

A knowledge based WSN, in which every sensor uses these variables as input and generates an alert status, is proposed to control the plague. If the alert status, which shows the risk of the plague appearance, surpasses a threshold level (i.e. 0.75), the suitability of insecticide treatments applications should be evaluated. The knowledge base, which governs the behaviour of the fuzzy system, is composed of inputs and output fuzzy variables, membership functions (figure 5), and the rule base (table I).

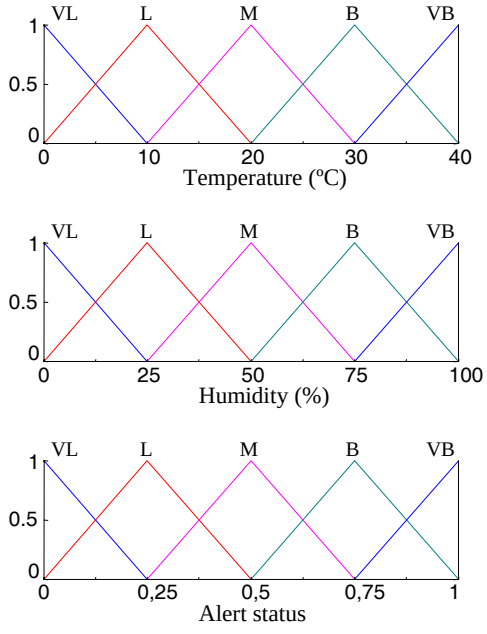


Fig. 5. Membership functions of the inputs and output variables fuzzy sets

TABLE I
BASE OF RULES USED IN THE CONTROL OF THE OLIVE TREE FLY

Alert	Temperature					
Humidity		VL	L	M	B	VB
	VL	VL	VL	L	M	VL
	L	VL	L	M	M	VL
	M	VL	M	B	B	L
	B	VL	B	VB	B	L
	VB	VL	B	VB	VB	L

Figure 6 shows the input-output control surface which relates the values of the temperature and the humidity with the alert status.

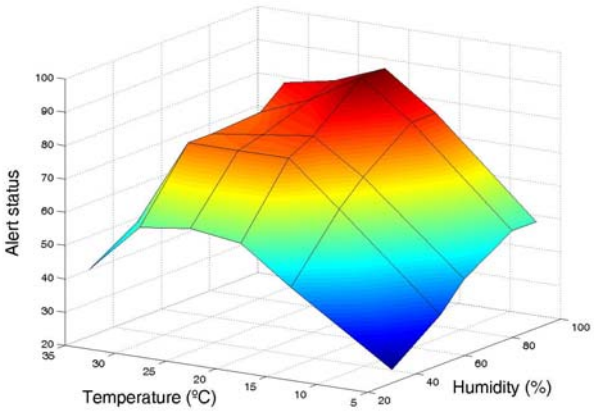


Fig. 6. Input – output control surface

IV. CONCLUSION

The paper presents an effective approach, a knowledge based WSN, in order to control a plague in agriculture. The tests have demonstrated the effectiveness of the Sun SPOT to support a knowledge based system and its use in a large number of applications.

ACKNOWLEDGMENT

This work has been partially supported by University of Jaen and Caja Rural de Jaen (project UJA_08_16_18).

Authors would like to thank Sun Microsystems Inc. for its support under the grant Sun SPOT Development.

REFERENCES

[1] E. Dekneuvel. “Intelligent Sensor: Analysis and Design”, Embedded System Handbook, Edited by R. Zurawski, Tailor and Francis Group, 2006.

[2] Shan Q, Liu Y, Prosser G, Brown D. “Wireless intelligent sensor networks for refrigerated vehicle”, Proceedings of the 6th IEEE Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication, vol. 2, pp. 525-528, 2004.

[3] Benoit E, Dapoigny R, Foulloy L. “Fuzzy-Based Intelligent Sensor: Modelling, Design, Application”, Proceedings of the 8th IEEE International Conference on Emerging Technologies, pp. 401-407, 2001

Proc. EWSN 2009 – Demos/Posters Session, Cork, Ireland, February 11th – 13th, 2009

Poster Abstract: Gradient-based Integral Sampling for WSNs in Building Automation

Joern Ploennigs, Volodymyr Vasyutynskyy, Mario Neugebauer, and Klaus Kabitzsch

***Abstract*—Energy efficiency is an elementary requirement for battery-operated wireless sensor networks, which concerns not only the device hardware and communication protocols, but also the device applications. This paper introduces a new gradient-based integral sampling approach that reaches an equal sampling quality with less messages and wake-ups.**

***Index Terms*—Wireless sensor networks, adaptive sampling, networked control, building automation.**

I. Introduction

Building automation is often envisioned as future mass market for wireless sensor networks [1] due to its opportune requirements of large networks with up to thousands of devices performing heterogeneous, decentralized tasks. WSN are also attractive for building automation primarily due to their easy installation without wiring. However, building automation is a conservative automation domain with high demands on WSNs. So, the networks have to work reliable for 10 to 30 years with a warranty given by the system integrators. The requested battery lifetime starts at 5 years [1]. Additionally, the real-time requirements are down to 100 milliseconds with a reliable message transmission. The common approaches to achieve the trade-off between real-time and energy efficiency focus on hardware [2] , operating systems [3] and protocols [4].

Whichever hardware or protocol are used, the device application defines the practical energy consume in the end. *Adaptive sampling* approaches like send-on-delta and integral sampling [5], [6] transmit messages only when their content has changed significantly and, thus, save transmission energy. They therefore sample the signal periodically with time T and decide for each sample, if it is transmitted. This makes them easy to implement without communication or control overhead. However, the device still has to wake-up for sampling that also requires energy. This drawback is intended to be remedied by the new sampling approach introduced in this poster, which uses the signal gradient to estimate wake-ups.

II. Gradient-based Integral Sampling

The limiting factor in the energy consumption of send-on-delta and integral sampling [5], [6] are the necessary periodic wake-ups for sampling. The sampling times are usually limited by the Nyquist-Shannon theorem and have to be set correspondingly to the worst-case signal frequency. For example, the temperature control cycle examined in Sec. III requires a periodic sampling every 10 s. Then, the energy consumption for wake-ups e_{wk} is twice as large than for message transmission e_{sd} in case of *send-on-delta sampling* (SoD), where a sample is only send if it changed more than a limit δ since the last transmitted value.

The idea of *gradient-based integral sampling* is to use the actual gradient in the signal to estimate the next sampling time and optimize the number of wake-ups. Assuming, the temperature raised 0.15 °C since the last sample 15 s ago, then it will estimably require only 10 s next time to reach a delta of $\delta = 0.1\,^{\circ}\mathrm{C}$. Thus, the time of the next sampling can be estimated by the linear extrapolation of the gradient. The absolute gradient m of the signal since the last sample $(i-1)$ is estimated at the sampling event i with time t_i by

$$m_i = \frac{|y_i - y_{i-1}|}{t_i - t_{i-1}} = \frac{\Delta y(i)}{T_i}. \tag{1}$$

The *integral absolute error of sampling* E_i in the sensor since the last transmitted message at event $k < i$ is estimated by

$$E_i = \sum_{j=k}^{i} \frac{|y_j - y_k|}{2} T_j. \tag{2}$$

Integral sampling limits this error and sends a message whenever it rises above a parameter $E_i \geq \delta_{\mathrm{I}}$. Now, the time for this event should be estimated as next sample time T_{i+1}. If the signal is extrapolated linearly with gradient m the error E will estimably change during this time to

$$E_{i+1} = E_i + \frac{|(y_i + m_i T_{i+1}) - y_k|}{2} T_{i+1}. \tag{3}$$

The next sampling interval T_{i+1} should be chosen in that way, that $1/S$ of the limit δ_{I} is reached, i. e. the equation $E_{i+1} = \delta_{\mathrm{I}}/S$ is met. Solving the quadratic equation above leads to

$$T^*_{i+1} \approx -\frac{|y_i - y_k|}{2m_i} + \sqrt{\frac{|y_i - y_k|^2}{4m_i^2} + \frac{2\delta_{\mathrm{I}}}{m_i S} - \frac{2E_i}{m_i}}. \tag{4}$$

The equation permits to estimate the time of the next sampling and message creation with a stepping factor $S = 1$. This saves unnecessary wake-ups, when new messages are unlikely and permits if necessary a minimum sampling interval even below the fixed T used by the other approaches.

Different cases can occur on device wake-up after the estimated sampling interval. In the best case, the signal changed as estimated and E_i approached δ_{I} within a tolerance value ε. If

J. Ploennigs is a Feodor-Lynen fellow at the Department of Civil and Environmental Engineering, University College Cork, Ireland and wants to thank the Humboldt-Foundation and the BMBF for their support, e-mail: J.Ploennigs@ucc.ie

V. Vasyutynskyy, K. Kabitzsch are with the Department of Computer Science, Dresden University of Technology, 01062 Dresden, Germany, e-mail: {vv3, kk10}@inf.tu-dresden.de.

M. Neugebauer is with ubigrate, Noethnitzer Str. 46, D-01187 Dresden.

the signal changed less than expected, the device can estimate a new sampling interval and enter sleep mode again. The condition of transmitting a value, which is evaluated at each wake-up, is:

$$((E_{i+1} \geq (1-\varepsilon)\delta_\mathrm{I}) \wedge (t_i - t_k \geq T_\mathrm{L})) \vee (t_i - t_k \geq T_\mathrm{U}). \quad (5)$$

with the parameters T_L and T_U to limit the minimum and maximum message interval.

If the signal changes significantly more than expected, it will be undersampled. To prevent this, it is necessary to wake up the device within the sampling interval to correct the gradient and the sampling interval if necessary. For this aim, the stepping factor S was introduced above. With a maximum stepping factor $S_\mathrm{max} = 2$, the device will wake up one time within the sampling interval. At this inter-sampling the next wake-up time is computed with a smaller stepping factor $S-1$, to prevent the endless bisection of the sampling interval. Each step, S is computed relative to the percentage of E_i within δ_I from the starting stepping factor S_max to

$$S = \min(S_\mathrm{max}, \max(1, \lfloor (1 - E_i/\delta_\mathrm{I})\,(S_\mathrm{max}+1) \rfloor)). \quad (6)$$

S_max is incremented by 1 in the equation to avoid further bisection if the next sample was nearly missed. This means, that if E_i reaches 48 % of δ_I at the inter-sampling step, the stepping factor S is computed to 1 with $S_\mathrm{max} = 2$, meaning that the next sampling interval is estimated directly and not bisected again.

To avoid infinite sampling intervals in case of slow signal changes, a mandatorily *max-sleep-time* T_W is defined, which wake-ups the device, but sends a message only after condition (5) is evaluated true. The *max-send-time* T_U in (5) defines a so-called heartbeat that wakes the device up and transmits a message even without value changes, so that the receiver is aware of its proper function. A *min-send-time* is useful to define a minimum sampling interval for gradient-based sampling. Assuming that $T_\mathrm{L} \leq T_\mathrm{W} \leq T_\mathrm{U}$ is fulfilled, the limited sampling time can finally estimated from (4) to

$$T_{i+1} = min(T_\mathrm{W}, max(T_\mathrm{L}, T^*(i+1))). \quad (7)$$

III. Results

A realistic simulation model of a single room in Matlab/Simulink is used to evaluate the new sampling approach. It simulates a highly automated room over one year with a temperature control including many realistic disturbances like weather, the heat load of the sun and people, set point changes, etc. The energy consumption for wake-ups e_wk and transmissions e_sd at the sensor is estimated with the energy model of a Telos rev B node presented in [3]. Also, the *number of wake-ups* N_wk and *sent messages* N_sd is counted. Transmission delays or message losses in the network are not considered to focused on the gradient sampling. They will be topic of further publications.

This scenario offers also a realistic feedback about the sampling quality. Large sampling times T create approximation errors and alias frequencies in the control cycles that drastically reduce the control loop performance (quality-of-control). It is expressed as common [6] by the *mean overshoot* $\bar{O}$, the *mean settling time* $\bar{T}_\mathrm{S}$ and the *integral absolute error of control*

$$IAE_\mathrm{C} = \int |w(t) - y_k(t)|\,\mathrm{d}t \quad (8)$$

where $w(t)$ is the set point value and y_k is the latest transmitted sample value.

	T, T_W	δ	N_wk	N_sd	IAE_C	$\bar{O}$	$\bar{T}_\mathrm{S}$	e_wk	e_sd
Peri	10	0	$3.2\mathrm{E}^6$	$3.2\mathrm{E}^6$	$5.7\mathrm{E}^5$	7%	819	29	716
SoD	10	0.05	$3.2\mathrm{E}^6$	$6.1\mathrm{E}^4$	$7.3\mathrm{E}^5$	8%	997	29	14
Int	10	0.05	$3.2\mathrm{E}^6$	$1.6\mathrm{E}^5$	$5.9\mathrm{E}^5$	8%	920	29	36
Grad	10	0.05	$3.2\mathrm{E}^6$	$1.6\mathrm{E}^5$	$5.9\mathrm{E}^5$	8%	961	29	37
Peri	120	0	$2.6\mathrm{E}^5$	$2.6\mathrm{E}^5$	$7.2\mathrm{E}^5$	15%	1497	2	60
SoD	120	0.05	$2.6\mathrm{E}^5$	$4\mathrm{E}^4$	$8.9\mathrm{E}^5$	15%	1747	2	9
Int	120	0.05	$2.6\mathrm{E}^5$	$5.8\mathrm{E}^4$	$8.7\mathrm{E}^5$	15%	1732	2	13
Grad	120	0.05	$3.3\mathrm{E}^5$	$6.3\mathrm{E}^4$	$7.8\mathrm{E}^5$	10%	1410	3	14
Peri	300	0	$1.1\mathrm{E}^5$	$1.1\mathrm{E}^5$	$1.6\mathrm{E}^6$	32%	3125	1	24
SoD	300	0.05	$1.1\mathrm{E}^5$	$3.8\mathrm{E}^4$	$1.8\mathrm{E}^6$	32%	3124	1	9
Int	300	0.05	$1.1\mathrm{E}^5$	$4.3\mathrm{E}^4$	$1.7\mathrm{E}^6$	32%	3096	1	10
Grad	300	0.05	$1.9\mathrm{E}^5$	$4.6\mathrm{E}^4$	$1.2\mathrm{E}^6$	17%	1787	2	11

TABLE I
Results of the sampling for a temperature control cycle
($[T_*]$ =s; $[\delta]$ =°C; $[N_*]$ = Nº; $[IAE_\mathrm{C}]$ =°Cs; $[\bar{O}]$ =%; $[e_*]$ =As;)

The results of the comparison are shown in Tab. I for different parameterizations and sampling algorithms. Send-on-delta (SoD) and integral sampling (Int) save already a lot of energy in comparison to periodic sampling (Peri) by not sending unnecessary messages. Gradient-based sampling (Grad) seems to have a comparable energy consumption on the first look, because the max-sleep-time T_W was set equal to the sample time T to keep the parameterizations comparable. In result, the device wake-ups even more often.

However, the benefit of gradient-based sampling is the improved quality-of-control with a lower IAE_C, mean overshoot $\bar{O}$ and settling time $\bar{T}_\mathrm{S}$. The device is able to reduce the sampling interval in dynamic situations to limit the sampling error especially for large sampling times T and max-sleep-times T_W. Hence, the energy saving rises from the fact that a higher T_W can be used to provide the same quality-of-control and with $T_\mathrm{W} = 120\,\mathrm{s}$ the device requires only 15 % of the energy of an integral sampling with $T = 10\,\mathrm{s}$ and 2.3 % of a periodic sampling.

References

[1] C. Reinisch, W. Kastner, G. Neugschwandtner, and W. Granzer, "Wireless technologies in home and building automation," in *INDIN - 5th IEEE Int. Conf. on Industrial Informatics*, vol. 1, 23.–27. July 2007, pp. 93–98.

[2] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *IPSN - 21th Int. Symp. on Information Processing in Sensor Networks*, 15. Apr. 2005, pp. 364–369.

[3] K. Klues, V. Handziski, C. Lu, A. Wolisz, D. Culler, D. Gay, and P. Levis, "Integrating concurrency control and energy management in device drivers," in *SOSP - 21th ACM SIGOPS Symp. on Operating Systems Principles*. New York, NY, USA: ACM, 2007, pp. 251–264.

[4] K. Langendoen, *Medium Access Control in Wireless Sensor Networks*. Nova Science Publishers, 2007, pp. 535–560.

[5] M. Neugebauer and K. Kabitzsch, "A new protocol for a low power sensor network," in *IPCCC - 23rd IEEE Int. Performance Computing and Communications Conf.*, Phoenix, AZ, USA, 15.–17. Apr. 2004, pp. 393–399.

[6] V. Vasyutynskyy and K. Kabitzsch, "Deadband sampling in PID control," in *INDIN - 5th IEEE Int. Conf. on Industrial Informatics*, vol. 1, 23.–27. July 2007, pp. 45–50.

Poster Abstract: Energy-efficient Rate Adaptive MAC Protocol (RA-MAC) for Long-lived Sensor Networks

Quanjun Chen*, Wen Hu†, Peter Corke†
*School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia,
Email: quanc@cse.unsw.edu.au
†ICT Centre, CSIRO Brisbane, Australia
E-mail: {wen.hu, peter.corke}@csiro.au

***Abstract*— This paper introduces an energy-efficient Rate Adaptive MAC (RA-MAC) protocol for long-lived Wireless Sensor Networks (WSN). Previous research shows that the dynamic and lossy nature of wireless communication is one of the major challenges to reliable data delivery in a WSN. RA-MAC achieves high link reliability in such situations by dynamically trading off radio bit rate for signal processing gain. This extra gain reduces the packet loss rate which results in lower energy expenditure by reducing the number of retransmissions. RA-MAC selects the optimal data rate based on channel conditions with the aim of minimizing energy consumption. We have implemented RA-MAC in TinyOS on an off-the-shelf sensor platform (TinyNode), and evaluated its performance by comparing RA-MAC with state-of-the-art WSN MAC protocol (SCP-MAC) by experiments.**

I. Introduction

Previous research shows that the dynamic and lossy nature of wireless communication is one of the major challenges to reliable data delivery in Wireless Sensor Networks (WSN). The traditional WSN approach is to use sophisticated Media Access Control (MAC) and routing protocols to compensate for poor link quality by using acknowledgments and retransmissions. However, these introduce extra traffic, increase energy consumption, and increase the size and complexity of the node's software.

There has been much less work focused on addressing the underlying link quality. Well known approaches include: adding relay nodes, using higher transmit power, applying frequency/antenna diversity or spread spectrum modulation techniques. All have difficulties or cost implications.

Coding and data bit rate are one of the remaining degrees of freedom in this problem. With suitable coding we could for example halve the data bit rate and achieve 3 dBm of extra signal processing gain. The resulting increased Signal to Noise Ratio (SNR) can move us from the lossy link zones to the good reception zones allowing error free transmissions without the need for multiple retransmissions.

Conventional Wireless Local Area Networks (WLAN) use rate adaptive MAC to optimise throughput, rather than overall transmission *energy cost*. To the best of our knowledge, RA-MAC is the *first rate-adaptive Media Access Control protocol for WSN* with the purpose of minimizing network energy consumption.

Most of the work was done while Quanjun chen was an intern in CSIRO ICT Center, Australia.

II. Rate Selection Model

To reduce radio energy consumption, previous research shows that it is very important for the radio transceiver to be in sleep mode as much as possible. To wake a receiver from sleep mode, a sender sends a special signal (called a "tone") before transmitting a data packet. For a given link condition, expressed by Receiver Signal Strength Indicator (RSSI), and a data bit rate (R), the energy consumption to successfully transmit a data packet is

$$E(RSSI, R) = \frac{P_{tx}(t_o + \frac{f}{R}) + nP_{rx}(t_o + \frac{f}{R}) + \sigma}{PRR(RSSI, R)} \quad (1)$$

where t_o is the tone transmission duration, f is packet frame size in bits, and n the number of neighbors that overhear the packet, P_{tx} and P_{rx} are the power consumption for transmitting and receiving respectively. σ is the energy consumed to wake up the radio chip and to listen on the channel before transmitting. $PRR(RSSI, R)$ is the percentage of packets that can be successfully received at the receiver for a given $RSSI$ and R. The inverse proportion of $PRR(RSSI, R)$ is the number of retransmissions required to successfully deliver a packet.

The optimal bit rate $\widehat{R}$, given the link condition $RSSI$, is the one that minimizes the energy consumption

$$\widehat{R}(RSSI) = \underset{R}{\operatorname{argmin}} \quad E(RSSI, R) \quad (2)$$

III. Rate Adaptive MAC (RA-MAC) Algorithm

The analytical results in Section II show that, in order to select the optimal data rates, the sender should monitor channel characteristics, which include $RSSI$ and the function $PRR(RSSI, R)$, continuously. Our proposed RA-MAC is designed to adapt transmission data rates to the channel characteristics dynamically for the purpose of minimizing transmission energy consumption.

When a sender sends a data packet, it first checks the delivery status of the last data transmission, which was transmitted at bit rate R_{last}. If the last packet was unsuccessful, the node decreases the bit rate level by 1 and transmits the data packet. Otherwise, the sender *predicts* the RSSI of the current packet transmission based on the moving average of recent RSSI samples, which have been piggybacked on the previous ACK packets. The sender then uses the predicted RSSI and the learned $PRR(RSSI, R)$ to select the current transmission bit rate R according to Eq. (2). Finally, the sender looks up the number of consecutive packets for which it has used the current R to transmit. If the number is larger than a certain threshold, the sender increases the current R by 1 and transmits the packet. Otherwise, the sender transmits the packet at the current R.

To adapt to changing channel condition more quickly, once the channel condition becomes bad (i.e., the last packet transmission fails), the sender decreases the data transmission rate R aggressively. Conversely as the channel improves (i.e., multiple successful recent packet transmissions), the sender increases the data transmission rate R in order to explore the performance function ($PRR(RSSI, R)$) at a higher bit rate.

The receiver collects current $RSSI$ and bit rate R after each received data packet. After the CRC verification process, the receiver updates the statistical performance function ($PRR(RSSI, R)$). If the packet is correctly received, the receiver sends an ACK packet which piggybacks the $RSSI$ and the learned performance ($PRR(RSSI, R)$) so that the sender can collect these performance values to guide its future packet transmissions.

IV. Implementation and Evaluation

We implemented RA-MAC in TinyOS on the TinyNode platform [1], which uses the XE1205 radio transceiver (from Semtech). This radio is highly programmable, and its physical data rates can be adjusted from 9.6kbps to 76kbps. By lowering transmission data rates, we can increase the transceiver's sensitivity (processing gain) and extend transmission ranges.

Fig. 1 shows the basic operations of RA-MAC in one duty cycle. Synchronized nodes poll the channel periodically. A sender firstly transmits a short tone to wake up a receiver before sending a one-byte data rate indicator to the receiver. Both tone signal and data rate indicator are transmitted at the base bit rate (i.e., 9.6 kbps). Now that the receiver is ready to receive at the specified data rate (i.e., 9.6, 20, 38 or 76kbps), the sender transmits the preamble and the data frame to the receiver at the specified data rate. Finally, the receiver transmits an ACK, together with RSSI and PRR, to the sender at the same data rate. To handle the clock drifts in the nodes, a synchronization packet is sent out every minute at the base transmission rate.

We ran the experiments on an *outdoor wireless link* (without line of sight, the distance between two nodes is approximately 30 meters) for multiple periods of 10 hours. We compare the performance of RA-MAC to that of SCP-MAC with fixed bit rates. The evaluation metric is *Energy consumption per received packet*, which is total energy consumption divided by the number of packets received successfully. For ease of comparison, we normalize the energy consumption of different strategies with respect to RA-MAC.

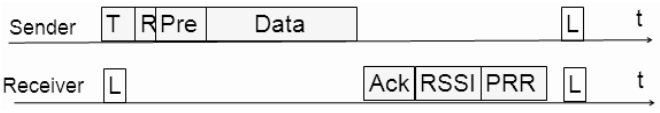
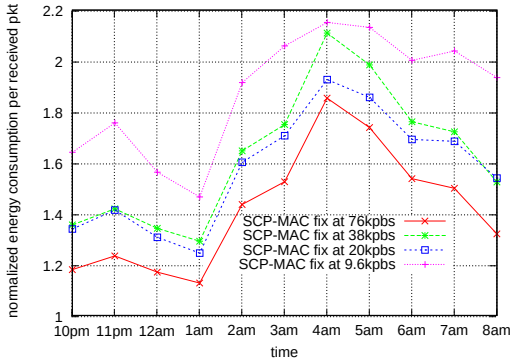
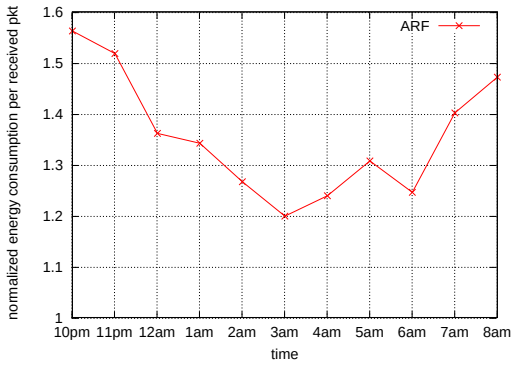


Fig. 1. The basic operations of RA-MAC. L: Polling. T: Tone. R: Data rate. Pre: preamble. Data: data frame. Ack: Acknowledgment bytes.



(a) RA-MAC vs. SCP-MAC with fixed bit rates



(b) RA-MAC vs. ARF

Fig. 2. Normalized (against RA-MAC) energy consumption per received packet vs. time

Fig. 2(a) shows that RA-MAC reduces energy consumption significantly compared to SCP-MAC. For example, RA-MAC only consumes around 50% of the energy of SCP-MAC with various fixed bit rates, between 3am-4am. We also compared RA-MAC to Auto Rate Fallback (ARF), a popular used WLAN rate adaptive algorithm. Fig. 2(b) shows that RA-MAC outperforms ARF significantly.

V. Conclusion

We have introduced RA-MAC, which employs an adaptive approach to select the optimal data rates based on estimated link conditions to reduce network energy consumption. Our outdoor experiment evaluation showed that RA-MAC outperformed both WSN (SCP-MAC) and WLAN (ARF) MAC significantly.

References

[1] H. Dubois-Ferrière, L. Fabre, R. Meier, and P. Metrailler. Tinynode: a comprehensive platform for wireless sensor network applications. In *Proceedings of IPSN*, pages 358–365, New York, NY, USA, 2006. ACM.

Poster abstract: Gumsense - a high power low power sensor node

Kirk Martinez, Philip Basford, Joshua Ellul, Robert Spanton

Electronics and Computer Science, {km, pjb08r, je07r, rds}@ecs.soton.ac.uk
University of Southampton, UK.

I. Introduction

The development of increasingly complex algorithms for sensor networks has made it difficult for researchers to implement their design on typical sensor network hardware with limited computing resources. The demands on hardware can also mean that small microcontrollers are not the ideal platform for testing computationally and/or memory intensive algorithms. Researchers would also like access to high level programming languages and a wider range of open source libraries.

To address this problem we have designed and implemented an architecture, Gumsense [2] which combines a low power micro-controller (8MHz MSP430) with a powerful processor (100-600MHz ARM) on a Gumstix board [1] running Linux. This Open Embedded OS supports a wide variety of programming languages, package management and development tools. A similar hybrid approach was also used in the LEAP platform [3]. The microcontroller wakes up frequently to manage tasks such as activating sensors and gathering data. The intended use-case is to power-up the ARM board and storage only during the brief periods it is needed, for example performing computation or communication.

Fig. 1 shows the basic architecture of the platform with I^2C being used to communicate between the MSP430 and ARM platforms. The Gumsense board provides adjustable instrumentation amplifiers to allow the connection of a wide range of sensors as well as a USB port for additional connectivity.

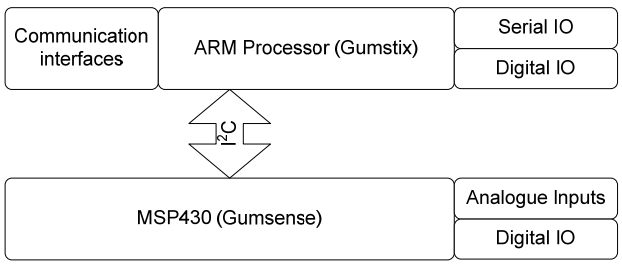


Fig. 1. Gumsense architecture

II. Hardware Design

The Gumstix does not have a suitably low power sleep mode so when it is not in use it is completely powered down, unlike the use of a sleep state used by LEAP[3] meaning that energy cost of sleeping is down to the MSP430 and some support circuitry. The Gumsense board comprises power control, I/O pins and the microcontroller with its analogue and digital I/O. The MSP430 is responsible for sensor reading, setting the next wakeup and controlling power rails according to the schedule set by the Gumstix. The MSP has 60k of flash memory which is used as a ring buffer for sensor values, which are then fetched by the Gumstix when it next wakes.

The design is shown in Fig. 2. In the top view of the 104mm wide PCB the Gumstix is visible in the upper half, surrounded by headers which make I/O and controlled power available for sensors and a radio. The underneath shows the potentiometers for gain settings, MSP430 and backup battery.

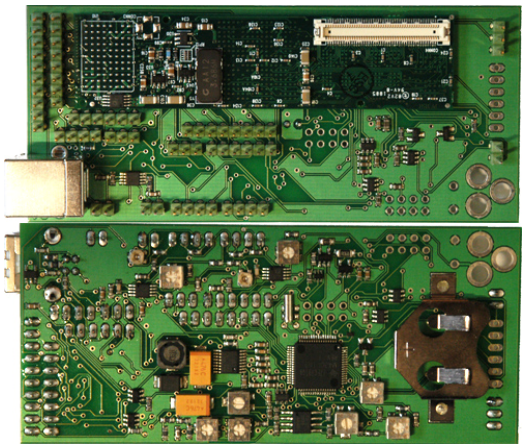


Fig. 2. PCB showing Gumstix (darker pcb on top picture) and underside view (lower).

III. Power Consumption

Typical sensor nodes will employ a periodic sleep-resume cycle in order to prolong node lifetime. These nodes have three different power levels as shown in Table 1, the majority of the time the node will be in state 0, with sensor readings being taken in state 1. State 2 will only be used for short periods in which communication or high powered computation is being performed.

Table 1 Power draw of a node in different states

ID	State	Power (mW)
0	Sleep	0.00720
1	MSP Active (Gumstix off)	52.4
2	MSP & Gumstix powered	900

As an example if the board is used to take 4 measurements (each taking 10 seconds) an hour, and then wakeup the ARM board for 2 minutes a day for processing

as well as performing 15s of communication a day then with a 12volt 2AH battery the predicted lifetime is in excess of a year. This estimate shows that although the Gumstix energy use is a large part of the overall budget, real system deployments are possible.

IV. PROGRAMMING

In order to make the system easy to control by Linux programmers the interchange of schedule and data has been made as simple as possible. The MSP430 maintains a list of recurring jobs with a set of analogue inputs, set of power rails to manage and sampling interval. The Gumsense also provides an adjustable delay time between the powering on of sensors and taking the readings to allow voltages to settle.

Communication between the Gumstix and Gumsense modules is achieved using nine registers on the I^2C bus with sensor readings being provided in 32byte blocks. Functions to manage this communication are provided as a kernel module for Gumstix with additional bindings to allow access from scripting languages.

V. POSSIBLE DEPLOYMENT SCENARIOS

There are three different deployment scenarios for which this platform is suited. The first as a gateway node for other sensors; secondly as a mesh of cluster heads to form a two-tiered network; finally as a complete network of homogeneous sensors. In the first scenario the gumsense platform can be equipped with an interface to allow communication with the sensors, the node could then perform any calculations required and forward the data on. In the second scenario the sensors can provide in network processing before transmitting the data to researchers, the amount of storage available on the platform would also allow data to be backed up over multiple nodes. Finally if the entire network consists of Gumsense boards then it can provide a useful test bed for algorithms, as powerful debugging environments can be run on the individual nodes if required.

A complete test of the system was carried out in Iceland in the summer of 2008 (see Fig. 3) when a system was used as the gateway for subglacial probes [5]. This included control of a dGPS and used a GSM/GPRS unit for communications home. The improved sleep power meant the system could run through the winter easily and batteries were seen to remain full thanks for solar/wind charging. It has been running since August 2008.

VI. CONCLUSION AND FUTURE WORK

The design provides a powerful processor (100-600MHz) with 64-128MB of memory (additional storage can be added in the form of SD or compact flash cards if required) while allowing for real deployments where the data sampling is generally handled by a low power microcontroller. A test deployment within the department will be used to develop this concept as a prototyping platform for researchers. These can use WiFi cards for convenient configuration and easy updating as well as a low power radio for research.

At the moment the MSP430 runs a custom operating system, however this could be changed to tinyOS2[6] or Contiki[7]. If done in a manner adhering to the I^2C protocol currently in use then this change would be transparent to the higher levels.

This platform overcomes a number of limitations of existing low power sensor networks by combing very low sleep power consumption (approx 72µW) with sufficient computational capability to support, for example, complex sensor network behaviour algorithms, asymmetric cryptography and autonomous experimentation algorithms.

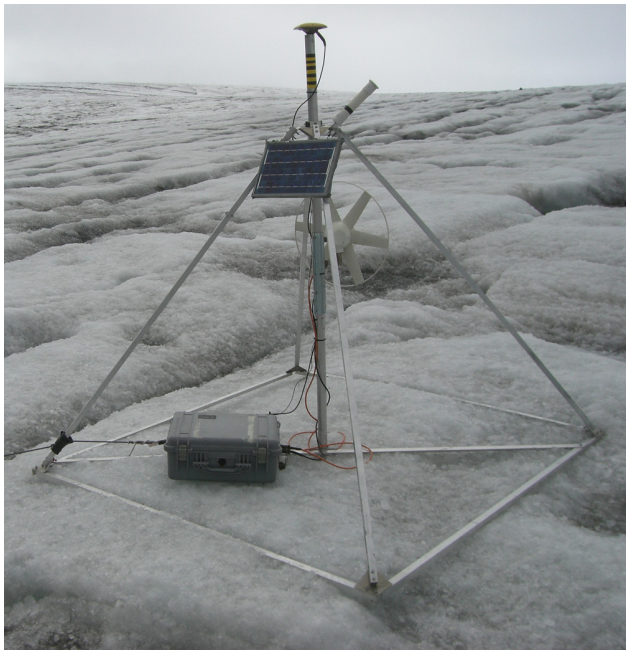


Fig. 3 deployment on the Skalafellsjokul glacier in Iceland

ACKNOWLEDGMENTS

Thanks to Royan Ong (Monash University Malaysia) for the version 1 design, Klaus Peter Zauner for many discussions, Mareike Bauer for prototype assembly and Jeff Gough and Tom Bennellick for work on the Iceland deployment. This research was funded in part by the EPSRC project Glacsweb, The Nuffield Foundation (URB/34245), and the ECS dept.

References

[1] Gumstix: www.gumstix.org
[2] Gumsense: gumstix.ecs.soton.ac.uk/wiki/index.php?title=Gumsense
[3] D. McIntire, K. Ho, B. Yip, A. Singh, W. Wu, and W.J. Kaiser. “The low power energy aware processing (LEAP) embedded networked sensor system”, *Proceedings of the fifth international conference on Information processing in sensor networks*, pp. 449–457, 2006.
[4] Martinez, K., Hart, J. and Ong, R. “Environmental Sensor Networks”, *IEEE Computer*, 37 (8). pp. 50-56. 2004.
[5] Martinez, K., Padhy, P., Elsaify, A., Zou, G., Riddoch, A., Hart, J. K. and Ong, H. L. R. (2006) Deploying a Sensor Network in an Extreme Environment. In *Proceedings of Sensor Networks, Ubiquitous and Trustworthy Computing*, pp. 186-193, Taiwan.
[6] TinyOS2 www.tinyos.net/
[7] Contiki www.sics.se/contiki/

Poster Abstract: Harvester – Energy Savings Through Synchronized Low-power Listening

Roman Lim, Matthias Woehrle, Andreas Meier, Jan Beutel
Computer Engineering and Networks Lab
ETH Zurich
8092 Zurich, Switzerland
lim@tik.ee.ethz.ch

***Abstract*— Long-term data gathering with Wireless Sensor Networks (WSNs) requires drastic optimization of power consumption to provide a long system lifetime, especially for battery operated motes. In this work, we discuss Harvester, a low power data collection application that leverages recent advancements over the standard MAC and network layer components provided by the TinyOS-2.x operating system. More specifically we contribute an energy saving strategy based on a per-hop synchronization mechanism on the MAC layer. Our open-source implementation achieves over 70% energy savings over the standard LPL layer currently provided by TinyOS.**

I. Introduction

WSN systems are networks of embedded systems, where each mote has severe resource limitations e. g. memory or energy. Ample energy resources necessitate a focus on low power consumption operation. Especially for data-gathering applications requiring long-term operation, it is not feasible to periodically change the batteries. Instead, the protocols used for collecting data over multiple hops need to be optimized.

TinyOS provides MAC and network layer components to create multi-hop data gathering applications: Low-power listening (LPL) and the Collection Tree Protocol (CTP) [3] are readily available and commonly used. While the TinyOS developers are steadily improving this combination forming a low-power data gathering stack, it is still inferior over custom solutions with respect to power consumption [1], [4]. We present the protocol stack of Harvester, a low-power data gathering application developed over the last 2 years as contributed project to TinyOS[1]. Utilizing time synchronization, our radio stack saves considerable amounts of energy compared to the standard LPL+CTP combination on the order of 70%.

In the following, we present Harvester and especially the synchronization enhancements made on the LPL layer. We present a case study based on a data collection application underlining the substantial savings in energy savings.

II. Synchronized Low Power Listening

Low power MAC protocols for WSNs duty cycle the radio as it is the main power consumer of the sensor node. LPL, introduced in BMAC [6], is a random access MAC protocol, where nodes wake-up independently and have no coordination for message exchange. Using LPL, a node signals the transmissions of a packet by sending a preamble. Its neighbors periodically sample the channel, listening for preambles. WiseMAC [2] is a particular LPL scheme, which tracks wake-up phase offset between the neighbors. Hence, a sender can adapt its wake-up time to send a preamble just before the destined neighbor wakes up, considerably saving energy and bandwidth. In packet-based radios such as the CC2420 used on the Tmote Sky, long preambles can be generated by transmitting the actual packet in succession [5].

In order to achieve a lower energy footprint, we added per hop synchronization to the default TinyOS CC2420 radio stack. We used local synchronization (similar to WiseMAC) and adapted it to packet-based radios. Differing from global, local synchronization is independent of network size.

The set of modifications introduced in the LPL protocol are the following: (1) We changed of the power cycling to a *predictable wake-up schedule*, (2) added the ability to *send packets exactly at a given time* with 32 kHz clock granularity, (3) added *time information* to each packet sent, in order to share wake-up times and clock skews with neighboring nodes and (4) added a robust *wake-up prediction for neighboring nodes* utilizing drift estimation techniques.

A. Short vs. Long Preambles

Every sent packet is extended by time information about the wake-up schedule of the sender node including the offset since the last wake-up and the sleep interval. Initially, packets are sent with a ”long preamble” (cf. Fig. 1), e. g. as a packet burst. Every node gradually builds up a table of time information about its neighborhood. Using the augmented neighborhood table and timed packet transmission, it is possible to exactly schedule packets without long preambles. Packets which cannot be sent due to channel occupation are sent in a later period.

Short packets can only be used for unicast traffic, broadcast messages still occupy a whole period.

B. Wake-up prediction

Consecutive time information of a node allows to calculate its drift by comparing the measured wake-up intervals against the claimed interval. In order to get accurate drift measurements by using the limited measurement resolution of

[1] http://tinyos.cvs.sourceforge.net/ in /tinyos-2.x-contrib/ethz/snpk/

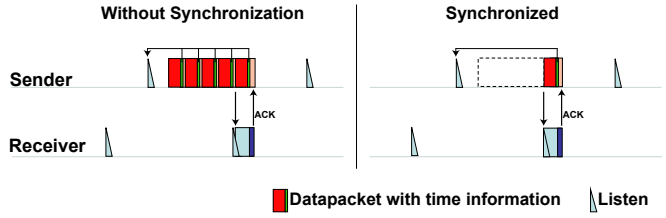


Fig. 1. All messages contain information about the senders wake-up offset. As long as there is no information about the wake-up offset of a neighbor, long preambles are used (1). Gradually, the offsets of frequent communication partners are known and a long preamble can be omitted (2).

the 32 kHz clock, a history of several measurement points is stored. From this information, we calculate the average drift. The longer the history, the more accurately the drift to every neighbor can be computed. Hence, a node can longer sustain local synchronization without message exchange.

C. Timed Packet Transmission

Switching the CC2420 hardware on and off as well as sending a packet consists of a sequence of actions, which have non-deterministic execution times. Additionally, there could also be tasks interfering. Hence we introduce an upper time bound for the whole sequence. Within this window, the radio is turned on, and the packet is loaded into the FIFO buffer. Once the FIFO is loaded, a very short command sequence, triggered by a timer, starts the actual transmission.

D. Tuning for Collection Protocols

We add two features that provide additional performance when used with a collection protocol. Collection protocols in wireless sensor networks generally only generate transmissions towards the sink. This leads to parent nodes being well informed about wake-up times and drift of their children, but not vice-versa. We add a resynchronization flag to the packet header to cope with this asymmetry. If time information becomes inaccurate, a node can request a synchronization update. A node replies with a packet containing time information.

Secondly, in very low duty cycle networks, throughput can be reasonably increased by transmitting several data packets in one wake-up interval. In order to make this possible, the receiver node has to stay awake after packet reception. We introduced another header flag called “More-Flag” to support this feature. In conjunction with CTP, this requires an enhancement to the forwarding engine.

III. Comparison

As Harvester uses CTP for data gathering, we used this protocol to compare our synchronized LPL with the existing MAC. CTP creates a routing tree based on estimated numbers of transmissions (ETX) to the sink. Link quality estimation based on sequence numbers of control packets and data acknowledgement render CTP platform independent. The Trickle algorithm allows for adapting the amount of control traffic to the fluctuations in network connectivity. CTP achieves more than 97% reliability[3] with a non-duty cycled MAC.

The following test runs were all made on a 25 node Tmote office building testbed. We introduced a patch to the original CTP to allow low power listening. We compared the TinyOS CTP/LPL combination against CTP with our synchronized MAC protocol. In both cases, the sink node does not duty-cycle the radio. The nodes run for ten minutes without data collection for initialization. Subsequently, nodes generate and send data packets with a rate of 1 packet per 20 seconds for a period of 30 minutes. We measured the effective radio duty cycle on every node. Fig. 2 displays boxplots for the individual measured duty-cycles. Our tests show that both protocols achieve comparable data yield (97.20% - 99.78%) but the synchronized LPL is superior as it achieves energy savings of 71.6% to 75.6%.

The essence of this work is very similar to the local synchronization mechanisms proposed in recent related work [1], [4]. However the solutions provided here are (i) more general in their applicability to a wide family of TinyOS-2.x based applications than custom solutions [1] and (ii) freely available as open-source. Further information and source code is provided in the contributed TinyOS project.

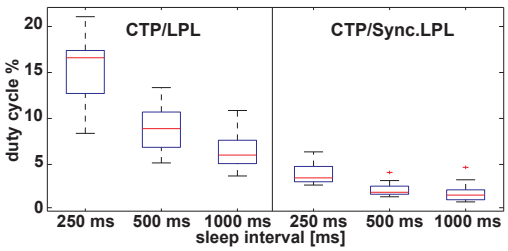


Fig. 2. Boxplot of effectively measured duty cycle. The values are measured over all 24 non-sink nodes with different sleep periods.

IV. Acknowledgments

The work was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

[1] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: ultra-low power data gathering in sensor networks. In *Proc. 6th Int’l Conf. Information Processing Sensor Networks (IPSN ’07)*, pages 450–459. ACM Press, New York, Apr. 2007.

[2] A. El-Hoiydi and J.-D. Decotignie. Wisemac: An ultra low power mac protocol for multi-hop wireless sensor networks. In *ALGOSENSORS*, pages 18–31, 2004.

[3] O. Gnawali et al. Ctp: Robust and efficient collection through control and data plane integration. Technical report, Univ. of Southern California, UC Berkeley, MIT CSAIL, Stanford Univ., 2008.

[4] J. Hui and D. Culler. Ip is dead, long live ip for wireless sensor networks. In *Proc. 6th ACM Conf. Embedded Networked Sensor Systems (SenSys 2008)*, pages 15–28, New York, NY, USA, Nov. 2008. ACM Press, New York.

[5] D. Moss et al. *TinyOS 2.x, TEP 126: CC2420 Radio Stack*.

[6] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004)*, pages 95–107. ACM Press, New York, 2004.

Poster Abstract: Achieving latency restrictions in heterogeneous Sensor Networks

J. Borms*, K. Steenhaut*†, B. Lemmens*†, W. Colitti*
*Vrije Universiteit Brussel - Department of Electronics and Informatics ETRO
†Erasmus Hogeschool Brussel - Departement IWT
contact: {jborms, ksteenha, blemmens, wcolitti}@etro.vub.ac.be

***Abstract*—One can imagine a lot of realistic scenarios, such as home or factory automation, where a wide variety of sensors and actuators form a wireless network in order to perform various tasks. There is still a lot of work to be done to ensure scalable and reliable functionality of such a heterogeneous network. We present a MAC layer algorithm that ensures end-to-end latency requirements while minimizing maintenance. We demonstrate that our algorithm scales down the duty cycles along constrained paths in such a way that nodes with more capabilities take on more responsibility in meeting the end-to-end application constraints.**

I. Introduction

In recent studies [1][2] a lot of interest has been shown in the development of a communication stack or, more generally, a service platform to run applications on heterogeneous sensor networks. Not only do the nodes in such a network have highly variable power constraints (some may even be mains-powered), the network should be able to support applications with strong demands in terms of end-to-end latency and reliability. Moreover, not only sensor-to-sink traffic is important, we also have to consider point-to-point traffic patterns within the network.

Even though we have to manage additional constraints, such a network also offers opportunities: it can mitigate the power limitations of some nodes by giving extra responsibility to more powerful nodes. In this work we demonstrate this principle for the MAC layer.

In related work, [3] describes how to adapt the duty cycle of self-recharging nodes based on remaining battery capacity. Paper [4] proposes an algorithm to control the sensing and sending rate based on the power budget of a node. These approaches however focus only on energy-aware behavior and not on application level QoS constraints.

II. MAC Properties

We want the nodes to be able to differentiate their duty cycle individually, since the network is heterogeneous. The nodes should also be able to adapt to variable traffic load and network conditions. For these reasons, we prefer an asynchronous MAC layer such as X-MAC [5], in order to avoid resynchronization whenever there are changes in the network.

In X-MAC nodes periodically wake up for a short time denoted t_{wake} and sleep in between for a period denoted t_{sleep}. As depicted in fig. 1, nodes communicate only when there is data to be sent. When a node A wants to send data to B, it starts sending out polls to B, denoted **B?**. When B wakes up, it hears a poll message and acknowledges it. Having received an acknowledgment, A can now send its data.

In the following we motivate our choice of t_{sleep} for each node in the network, t_{wake} is fixed.

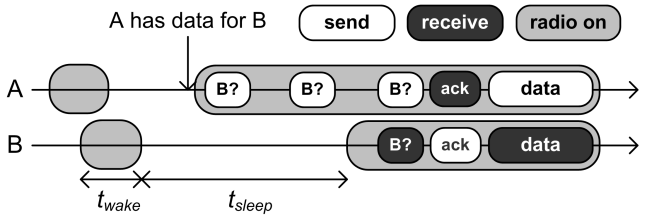


Fig. 1. X-MAC communication scheme

III. Algorithm

Looking at fig. 1 we observe the following: if the sleep time of B increases, A will spend more energy sending data to B since it will have to poll longer on average. However, if B sleeps more, B itself will spend less energy. In the original paper that introduces X-MAC, Buettner et al. present an algorithm to scale the sleeping period of X-MAC depending on the amount of traffic. They consider three costs to each message transmission: the energy consumption of the receiver, the energy consumption of the sender and latency. In our study, we modified this algorithm to take into account the heterogeneity of the network. In a first step, energy consumption of the nodes is optimized. Next, end-to-end latency along along a path is monitored and the duty cycles of nodes along that path are tuned to fit the end-to-end constraints.

A. Optimal sleep time

If we define a power P_{max} for each node which represents a target for its maximum average power consumption (for example, based on desired lifetime and battery capacity) we calculate the the optimal sleep time for a node j as follows:

$$t^*_{sleep,j} = \sqrt{\frac{t_{wake} \cdot \left(\frac{P_{wake,j}}{P_{max,j}}\right)}{\sum_{i=1}^{n} \left(\frac{p_{i,j}}{2} \cdot \frac{P_{wake,i}}{P_{max,i}}\right)}} - t_{wake} \qquad (1)$$

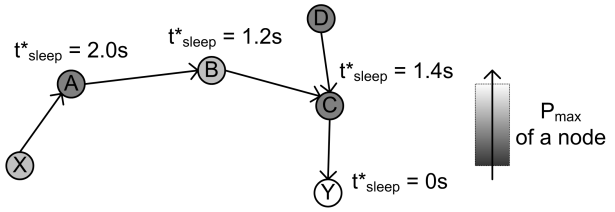


Fig. 2. heterogeneous network with optimal sleep times shown, without end-to-end latency constraints

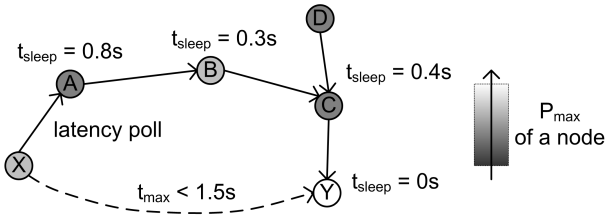


Fig. 3. optimal sleep times adapted to meet the end-to-end latency constraint

where j has n neighbors and $p_{i,j}$ is the measured message rate from neighbor i to node j (# messages per second). $P_{wake,j}$ is power consumed by node j when the radio is on. Formula (1) allows that each node computes its optimal sleep time with minimal overhead; maintaining a traffic history table and communicating P_{wake}/P_{max} to all neighbors. Practically, we also set an upper limit to t_{sleep} to avoid bad network responsiveness after a long period of relative silence (from (1) would follow that as $p \to 0$, $t_{sleep} \to \infty$).

If nodes periodically adjust their sleep times according to formula (1) sender and receiver energy will be traded off, while nodes are forced to contribute proportionally to their energy budget. This is depicted in fig. 2 where nodes with less capabilities are represented in darker color. Node A and C have the same capabilities, but as C has to forward more traffic, its optimal sleep time is lower.

B. Latency constraints

We observe that the one-hop latency in X-MAC is dominated by the sleep time of the receiving node. When transmitting over multiple hops the worst-case latency will be the sum of all worst-case one-hop latencies. If there are no queuing delays and packets are short, it is reasonable to assume the sum of all sleep cycles (multiplied by a worst-case estimate on transmissions per hop n_{tx}) of all receivers on a path will be a good estimate of worst-case latency.

$$\text{latency}(path) \leq \sum_{j \in path} (t_{sleep,j} + t_{wake}) \cdot n_{tx,j} \qquad (2)$$

This means that if a latency restriction along a certain path can not be guaranteed, we should reduce the sleep times of all nodes on that path.

Looking at (1), we notice that a weighed reduction of sleep times on a certain path can be achieved by increasing traffic on that path. Obviously we do not want to congest the network, since this would increase latency again (and violate our earlier assumption that queuing can be neglected) but we could artificially increase the perception of traffic through a node.

C. Putting it together

When an application requests a latency constraint from one node to another, the node where constrained traffic will originate polls the target node to make sure a route exists. All intermediate nodes add their sleep time to a field in this poll message, and finally the target node will echo back this message so the initiator can evaluate the total latency, based on (2). Since link conditions may vary over time and nodes may be added to or removed from the network, these route polls should be repeated to ensure there is always a good route to the target node. This also has the advantage that there should be no route setup needed when actual constrained traffic is sent out.

Because of the traffic balancing, these polls may suffice to drop the latency along a path below the maximally allowed latency. If not, the polling application will add a "traffic weight" to the poll messages so nodes along the constrained path take additional virtual traffic into account when calculating their optimal sleep time. This is illustrated in fig. 3.

IV. Implementation

Currently we are developing software in the Contiki OS [6], using the existing X-MAC layer and Rime routing layer, to test the feasibility of this concept on a small testbed. We offer an `add_latency_constraint` primitive to the application layer, which is registered by a MAC layer "learning module" that performs the tasks described in the previous section. In future work we intend to broaden the scope to the routing layer as well.

Acknowledgments

This work has partly been done in the scope of the ITEA ESNA project with the support of IWT Flanders.

References

[1] M. Yarvis, A. Kushalnagar, H. Singh, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *in Proc. of the IEEE Infocom*, 2005, pp. 366–377.

[2] E. Troubleyn, E. D. Poorter, I. Moerman, and P. Demeester, "Amoqosa: Adaptive modular qos architecture for wireless sensor networks," *Sensor Technologies and Applications, International Conference on*, pp. 172–178, 2008.

[3] C. Vigorito, D. Ganesan, and A. Barto, "Adaptive control of duty cycling in energy-harvesting wireless sensor networks," *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on*, pp. 21–30, June 2007.

[4] K.-W. Fan, Z. Zheng, and P. Sinha, "Steady and fair rate allocation for rechargeable sensors in perpetual sensor networks," in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*. New York, NY, USA: ACM, 2008, pp. 239–252.

[5] M. Buettner, G. Yee, E. Anderson, and R. Han, "X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks," Department of Computer Science University of Colorado at Boulder, Tech. Rep., 2006.

[6] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, Nov. 2004. [Online]. Available: http://www.sics.se/ adam/dunkels04contiki.pdf

Poster Abstract: Combining Positioning & Communication Using Ultra Wideband Transceivers

Paul Alcock and Utz Roedig
Infolab21, Lancaster University, UK
Email: {{p.alcock|u.roedig}@lancaster.ac.uk}

***Abstract*— A new generation of ultra wide band (UWB) communication transceivers are becoming available that supports both positioning and communication tasks. Transceiver manufacturers envision that communication and positioning features are used separately and asymmetrically. We believe that this is an unnecessary restriction of the available hardware and that positioning and communication tasks can be active concurrently. This paper presents and investigates a medium access control (MAC) protocol which combines communication and positioning functions. The presented MAC protocol extends an established protocol concept which is used, for example, in the standard TinyOS low power listening component. Our experiments show that the existing data communication of a network can be exploited to gather position information efficiently.**

I. Introduction

Many systems of positioning have been developed which use the existing communication transceiver of a sensor node. These currently used transceivers are however not well suited for positioning tasks due to the physical properties of the signal. Positioning systems relying on these conventional low-power communication transceivers typically make use of either the received signal strength indicator (RSSI) or the measured time-of-flight (TOF) of a signal as input for the positioning algorithm. Both methods can be used to determine the distance between transceivers and ultimately the position of all transceivers in relation to each other. These methods of distance measurements have been investigated at length and reports show that using current transceivers yield unreliable and inaccurate results. Patwari et al. [1] present an in-depth report of their findings of how multipath signals and shadowing obscure distance measurements.

The recent development of low-power, ultra wide band (UWB) transceivers for use in sensor nodes, overcomes the aforementioned ranging inaccuracies. The physical signal properties of UWB communication make it possible to accurately determine the time-of-arrival (TOA) of signals, by utilizing either clock synchronization or two-way-ranging it is therefore possible to accurately determine the time of flight (TOF) or the signal. Thus, the distance between communicating transceivers can be determined. The IEEE 802.15.4a physical layer specification [4], standardized in 2007, specifies the use of UWB transceivers for use in wireless personal area networks and the functionality of positioning. The Nanotron nanoLOC TRX [2] transceiver is one such transceiver which adheres to this standard.

UWB transceiver manufacturers envision that communication and positioning features are used separately and one at a time. Either the transceiver is used to transfer data packets between sender and receiver OR the transceiver is used to send ranging packets to determine the TOF between nodes. We argue that this leads to inefficient transceiver usage as excess packets might be generated unnecessarily. If an exchange of data packets is currently taking place between two nodes using a send and acknowledge scheme, the round-trip time can be used to determine the distance between nodes to avoid inter-node time synchronization. Therefore the distance can be estimated using the existing data packets, alleviating the need to transmit specialized ranging packets. If however there is currently not enough data communication taking place between nodes to accurately satisfy positioning needs, ranging packets may still need to be exchanged. Evidently the interplay of the transmission of data packets and ranging packets has to be organized to achieve the set communication AND positioning goals. Transceiver usage for communication purposes is defined by the MAC layer. The MAC layer determines when packets are transmitted and how their transmission shall be organized. Thus we propose to combine positioning and communication tasks within the MAC layer.

This paper describes how the existing FrameComm MAC layer [3] can be extended to support positioning tasks. The resulting system can be used with any transceivers adhering to the 802.15.4a IEEE standard.

II. Protocol Definition

FrameComm: FrameComm [3], like many wireless contention based MAC protocols, performs duty cycling of the node's transceiver. To ensure that rendezvous between transceivers occur, FrameComm deploys a method in which a trail of identical packets of data, called framelets, is transmitted by the sender with gaps between each. The receiver sends an acknowledgement to the sender after successfully receiving a framelet. Upon the reception of this acknowledgement, the sender may then cease sending and yield control of the channel (See Figure 1).

FrameComm with Positioning: The basic principle of FrameComm is ideally suited for the integration of positioning functions. The method of exchanging packets and acknowledgements mirrors that of the two-way-ranging method of determining the round-trip-time, and ultimately the TOF of signals. If the sender records the time of transmission of its last

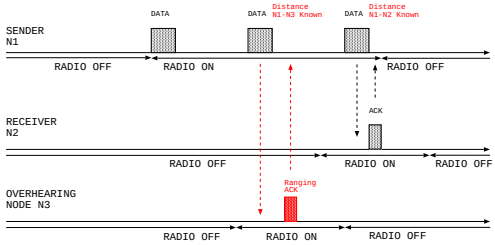


Fig. 1. FrameComm and FrameComm with Positioning

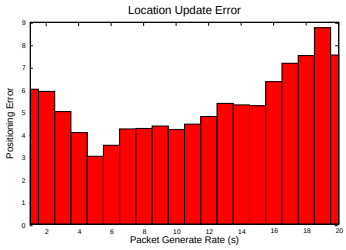


Fig. 2. Experiment 2 : Positioning error while tracking node movement.

framelet, and the time upon receiving its acknowledgement, the distance between nodes can be determined.

We propose extending this positioning enhancement further, in such a manner that the sender may derive not only the distance to its intended recipient, but potentially the location of any node within transmission range. During the exchange of framelets between the sender and receiver, a third node may enter its listening period and overhear a framelet. Before discarding the framelet and returning to sleep, the node exploits this overhearing and sends what we call a ranging acknowledgement. This ranging acknowledgement is offset to allow the communication acknowledgement from the intended receiver, to be processed without collision. Upon receiving the ranging acknowledgement the sender knows the distance to this third node (See Figure 1).

The described method allows us to implement positioning functionality without reducing the available bandwidth for communication. Minimal additional energy in form of ranging acknowledgements is consumed using positioning features.

III. Evaluation

Application Scenario: A warehouse (size $20m \cdot 30m$) is used to store crates which are equipped with a sensor node (12 nodes). Additional static nodes (14 nodes) are deployed such that a sensor node in a crate has always connectivity to at least one static sensor node. The sensor nodes in the crates transmit periodic sensor readings (packet generation rate of λ packets per second) through the network to a central sink. Collected distance measures are transported with these sensor readings in the same message to the sink. The sink employs a central location algorithm to calculate locations of all crates.

Experiment Setup: The evaluation consists of two simulation experiments. The first is designed to compare the original FrameComm protocol with our implementation including positioning, and is designed to measure differences in communication overheads. The second experiment evaluates the ability for our implementation to track mobile nodes (moving crates) within the network.

Traffic	FrameComm			FrameComm + Positioning		
λ (packets/s)	Transceiver On Time (s)	Throughput (packets/s)	Latency (s)	Transceiver On Time (s)	Throughput (packets/s)	Latency (s)
1	128.74	0.17	23.64	134.15	0.15	24.45
0.2	61.92	0.17	1.81	65.68	0.17	2.09
0.1	52.41	0.09	1.01	51.49	0.09	0.98
0.05	34.64	0.05	1.32	36.83	0.05	1.43

TABLE I
Experiment 1 Results : Communication overheads.

Results of Experiment 1: Table I shows data collected from one node in the deployment. The results show that the additional positioning mechanisms do not significantly decrease the nodes throughput or increase the nodes energy consumption. Only under a heavy traffic load ($\lambda = 1$) the achievable throughput drops as ranging acknowledgements are interpreted as a busy channel when nodes probe a channel before transmission.

Results of Experiment 2: Fig. 2 presents the findings of our second experiment where one node is mobile within the network (random waypoint movement, 1 m/s). The graph displays the distance (positioning error) between the true position of the node and the position estimated by the positioning algorithm on the sink for different traffic loads λ. For high traffics rates the positioning error is relatively high as packets carrying TOF measurements are lost in the network. For low traffic rates, the error is high as well as fewer packets containing TOF measurements reach the sink. For high traffic rates the problem could be corrected by ensuring that each node obtains a fair share of the available bandwidth. For low traffic rates, the problem could be corrected by sending TOF measurements to the sink even if no data has to be transported.

IV. Conclusion

In this paper we have presented a Medium Access Control Protocol which combines communication and positioning tasks. We have shown by simulation the general feasibility of the design. In further developments we plan to implement the protocol on real nodes that are equipped with 802.15.4a compliment UWB transceivers to evaluate the proposed protocol in a real-world deployment.

References

[1] N. Patwari, J.N. Ash, S. Kyperountas, A.O. Hero, R.L. Moses, N.S. Correal. “Locating the nodes.” Signal Processing Magazine, IEEE, Vol. 22, No. 4. (2005), pp. 54-69.
[2] Nanotron nanoLOC TRX Data Sheet. http://www.nanotron.com. 2007.
[3] J. Benson, T. O’Donovan, U. Roedig C. Sreenan. “Opportunistic Aggregation over Duty Cycled Communications in Wireless Sensor Networks.” In Proc. IPSN April 2008.
[4] IEEE 802.15 WPAN Low Rate Alternative PHY Task Group 4a (TG4a). http://ieee802.org/15/pub/TG4a.htmlKyperountas, A.O. Hero, R.L. Moses, N.S. Correal. “Locating the nodes.” Signal Processing Magazine, IEEE, Vol. 22, No. 4. (2005), pp. 54-69.

(Poster Abstract) PerDB: Performance Debugging for Wireless Sensor Networks

Veljko Pejovic
University of California, Santa Barbara
Email: veljko@cs.ucsb.edu

Cormac J. Sreenan
University College Cork, Ireland
Email: cjs@cs.ucc.ie

***Abstract*—The characteristics of wireless sensor networks are such that traditional approaches to network management and monitoring are inappropriate. In this paper a system for performance debugging in wireless sensor networks is presented. It tracks two key metrics - the delay and reliability of message delivery. Its principle of operation is to reactively isolate and diagnose problems once they occur, rather than proactively probe for data in order to provide real time monitoring - an approach that would be less resource efficient and would interfere with normal traffic flows. The system passively gathers information from the existing application traffic and reacts upon signs of problems, selectively probing misbehaving areas until the causes are determined and reported. The paper gives an outline of our performance debugging system PerDB along with preliminary results.**

I. Introduction

Debugging a wireless sensor network is not a straightforward process, and while there has been some work on software debugging, network management and monitoring (e.g. [2],[3],[1]) there has been little attention to *performance* debugging of a WSN. Our goal is a performance-focused debugging system for wireless sensor networks, emphasizing the difference between problems that manifest in performance failures as opposed to functional failures. Obviously the determination as to whether a network is operating within or outside acceptable performance bounds must be application-specific. Our aim is not to design a system that will provide a complete picture of the network state; rather, we strive towards debugging - recognizing faulty situations, localizing them, and finding out the true causes of the undesired events. The contribution of this paper is to present a performance debugging system that passively monitors a WSN in order to detect when it is operating outside of the given thresholds for message delivery latency and reliability, and then uses selective probing to localize and identify the root causes.

II. Approach

The focus of measurement in the proposed system is message delivery, which represents the fundamental service provided by a WSN. While message delivery can be characterized by several metrics, the key generic metrics of interest to most applications are clearly the delivery latency and delivery reliability, both observed from end-to-end. These two metrics encompass a wide range of applications that require performance assurances for message delivery in a WSN.

Our basic approach to measurement is passive, and recognizes that with small modifications in place, the existing traffic in a WSN can be used to piggyback information useful for detecting and locating problems. In contrast to an active, probing-based, approach, the use of a passive approach has significant benefits for energy consumption and is less intrusive. Yet, for the successful isolation of the problems the use of probing cannot be avoided. When probing, we adopt an approach in which nodes aim to maximize to use of space to include as much debugging information in the replies as possible, so that a minimal number of probes is sent and no significant interference with the data traffic is introduced.

III. System Design

In PerDB each packet gathers key performance information as it travels from source to sink. Delay information is added to the cumulative delay field contained in the packet payload. In a similar manner, aggregate packet retransmissions are counted on each of the nodes, together with the total hop count a packet takes to reach the sink. In our implementation for TinyOS we use an additional fours-bytes for carrying this information.

Protocol: Although this approach results in relatively little additional overhead, it is not always feasible to isolate errors via analyzing the relevant information in just a few bytes of each message. In that case the sink relies on a request-response protocol to probe the network. It can probe a specific node in the network, eliciting a response message that is used to gather more detailed performance information about the path. The node's reply consists of information similar to that described above; however, the fact that the entire message can be used allows detailed information for each hop along the path to be obtained. Information such as a node's number of neighbors, battery levels, sources of delay and others can be harvested as well. The probing process consists of the following steps: (a) A probe is sent from the sink towards a problematic node; (b) Each node on the probe's path is "told" what to measure, and how long to wait for its child's answer; a timeout value is set from the moment a probe visits a node and is directly proportional to the number of nodes yet to be visited by a probe; (c) If it gets a probe reply from the child, a node combines its own reply with it and sends to the sink; (d) Otherwise if after the timeout the node does not receive the

answer from its child it generates its own answer and sends it to the sink.

Algorithm: After the data arrives to the sink it is analyzed according to the application's needs. Consider the basic metrics: a network does not perform well if the amount of delivered data is less than expected, or if the average traffic delay is above a certain threshold.

Since the system assumes a simple tree topology it may be able to localize clusters of nodes that are experiencing high delay, in that case the probes are sent towards the clusters' heads since all delayed traffic must go via these common-parent nodes. On the other hand, the paths experiencing the delay may not be under a shared parent node, but we can still try to find some overlapping of the delayed paths along the way; in that case the probes are sent to the overlapped parts first. If there is no overlap then every single node is probed.

In comparing paths, the system uses information derived from "healthy" traffic, i.e. traffic delivered without severe delay and without per-hop retransmissions. Overlapping between a path that a packet which was not delayed and the one which was delayed took can tell us where to look for errors, so that probes are not sent to explore links that belong to the overlapped problematic paths. However, if "healthy" data traffic cannot be utilized then every node whose packets are delayed may need to be probed. This is optimized by probing nodes on the most important paths - the ones whose links are used by the largest number of nodes in the network. It is more likely that fixing errors along these lines will allow enough nodes to function properly.

PerDB deals with the problem of low traffic or an absence of it by probing clusters that are furthest from the sink.

IV. Experimental Evaluation

In order to conduct a preliminary evaluation of PerDB we modified TOSSIM source code to enable the necessary features our system uses. On top, a debugging layer was built above which a stub application is running. We focused on four aspects: (1) time to debug as a function of number of nodes, (2) benefit of using probing in terms of correctly identifying probematic links, (3) impact of number of probe packets on correctness and (4) time to debug relationship with average node degree in topology. Due to space restrictions we only show the proof-of-concept results in this paper.

We argue that even three bytes reserved for debugging purposes in each of the data messages can substantially improve the search for problems, pointing out to problematic areas; otherwise the probes are sent randomly to different parts of the network, in a greedy manner, trying to cover as many nodes as possible. In the experiments the number of nodes is incrementally increased, while keeping the same ratio of errors (lossy links, congested areas) versus well performing links. A comparison is made between the time needed for a system with and without piggybacked debug data to isolate and correctly determine the types of errors. As shown in figure 1, greedy probing does not scale - once the network size gets larger, it becomes much faster to probe the problematic areas

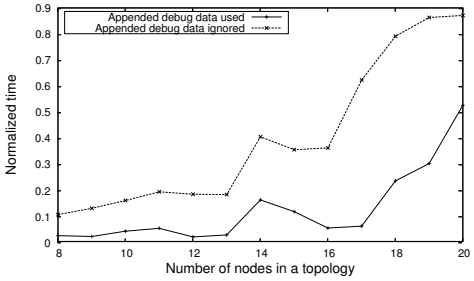


Fig. 1: Normalized debugging time as a function of number of nodes

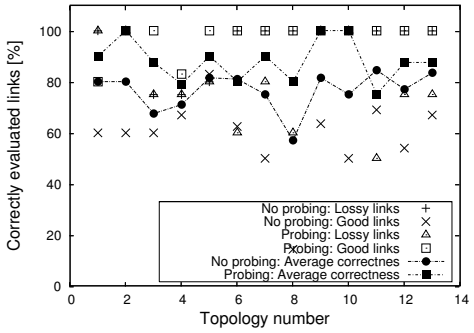


Fig. 2: Percentage of correctly evaluated links for the two approaches

only. It is worth noting that the additional negative effects of probing such as interference with the regular traffic were not considered. However, the probing is still necessary in a number of cases as the piggybacked debug information is insufficient for successful problem isolation. We examined a set of different (in size, ordering of nodes and placement of erroneous links) topologies and compared the correctness of reporting from the system that does not probe and the one that probes if application specific performance thresholds are exceeded. More precise diagnosis is observed when probing is in place (Figure 2). Without probing it is hard to conclude about the source and location of problems. Naturally a tradeoff exists regarding the speed and accuracy of diagnosing performance violations versus the energy required for probing.

V. Conclusion

We have presented a concise summary of PerDB, a performance debugging protocol for wireless sensor networks. PerDB uses a passive approach to gather information about network performance, together with probing when necessary. Ongoing work is to extend PerDB with a more sophisticated approach to inferring sources of performance violations and to deploy on our sensor node network management testbed.

References

[1] W. L. Lee, A. Datta, and R. Cardell-Oliver. *Handbook on Mobile Ad Hoc and Pervasive Communications*, chapter Network Management in Wireless Sensor Networks. American Scientific Publishers, 2007.

[2] Nithya Ramanathan, Eddie Kohler, and Deborah Estrin. Towards a Debugging System for Sensor Networks. *International Journal of Network Management*, 15(4):223–234, July 2005.

[3] M. Ringwald and K. Romer. SNIF: A Comprehensive Tool for Passive Inspection of Sensor Networks. In *GI/ITG KuVS Fachgesprach Sensornetze*, Aachen, Germany, July 2007.

Poster Abstract: Enabling Real-Time in WSN Applications

© Marcel Baunach, Clemens Mühlberger, Christian Appold 2009
Department of Computer Engineering, University of Würzburg, Am Hubland, 97074 Würzburg, Germany
Email: {baunach, muehlberger, appold}@informatik.uni-wuerzburg.de

***Abstract*—Increasing complexity of today's WSN applications can quickly boost the real-time requirements of the underlying sensor nodes. Using preemptive operating systems with special scheduling, resource and timing mechanisms is one way to retain acceptable reactivity for single tasks, nodes and finally for the overall system.**

I. Why Real-Time matters

Current research on Wireless Sensor Networks (WSN) is still mainly focused on networking itself. Indeed, increasing performance and computational power of recent sensor nodes allows a much wider scope of useful application scenarios. Especially Sensor Actor Networks (SANET) are central for the upcoming Ubiquitous Computing but also establish new challenges and require a clear focus shift from pure communication towards real-time capability within such distributed systems. The inherently strong interaction with the surrounding environment as well as the fast cooperation of participating nodes and embedded components will commonly need a much higher level of system reactivity. Sensing, interpreting and propagating environmental conditions will not suffice any more, but active participation and proactivity gain in importance, too. Additionally, complex applications often apply a modular design in which even simple software compositions can rapidly result in serious runtime problems and significant performance loss of the overall system. Still, available operating systems for sensor nodes often address these problems just insufficiently. However, special support within multi-threaded preemptive systems might help to retain acceptable reactivity for individual nodes and the overall WSN. Initially, we will point out quite typical real-time aspects within a complex real-world scenario and describe the various emerging problems regarding task scheduling, resource and time management. Finally we will present our concepts and solutions within the fully preemptive real-time operating system *Smart*OS.

II. Requirements of Real-World Scenarios

In this section we will present the potential application complexity by presenting a real WSN based tracking and steering control system for indoor vehicles.

The central localization algorithm periodically estimates the vehicle's current position from distance measurements between a mobile node mounted on the vehicle itself and static nodes within the environment. Then, a fuzzy controller computes the subsequent deviation from the desired track and readjusts the vehicle movement. The core software system comprises seven concurrently running main tasks for distance measurement, position estimation, radio communication, fuzzy logic, motor control as well as the main application and a remote maintenance task for software updates. Nevertheless, even a single one of today's low performance sensor nodes can be sufficient if the various emerging problems from the application's compositional complexity are handled adequately. The prerequisites will be addressed next.

Since the vehicle perpetually moves around, at least localization and fuzzy control must be implemented as *periodic tasks* with a suitable frequency to avoid crashes at the current velocity. The distance measurement in particular, which is based on TDoA between radio and ultrasound, requires a constantly precise, high-resolution and system load independent *timestamping* of signal arrival times. This also implies the timely allocation and configuration of the sender and receiver hardware, which requires *resource sharing* among several tasks. The aggregation of distance information at the mobile node uses a TDMA radio protocol and thus requires node *synchronization* for exact adherence to the time slots. Thus, data must be available on time at the sender and needs a high *reactivity* for fast processing at the receiver. For further improving the localization frequency, distance measurement and data aggregation tasks are both executed while the position estimation task still processes the information from the last measurement *simultaneously*. Thereby, the significant CPU load needs sophisticated *scheduling* and must provide *preemption* to avoid long-term blocking of highly reactive tasks. These real-time requirements, however, vary according to each individual task's current conditions. E.g. during distance measurement, the corresponding task temporarily needs a high priority. Beyond, other tasks should be privileged by means of *dynamic priorities*.

The just described tracking system was developed completely independent from the steering control and remote maintenance systems. Still, their tasks require interaction and *inter-task-communication*. Additionally, they must be composable even in case of potentially overlapping resource requirements. E.g. the motor control and measurement tasks share a common hardware timer to generate a preferably uninterrupted PWM signal for speed control and a sporadic square wave for driving the ultrasound transmitter, respectively. Finally, the fuzzy logic also produces high CPU load in parallel to the localization process and the remote maintenance system shall always be ready for external commands.

Next, we will outline problems of existing operating systems and our solution strategies regarding real-time operation.

III. Problems of current Systems and our Solution Strategies

Many available operating systems for sensor networks lack special support for such complex real-world scenarios. They often do not support preemptive or prioritized tasks, which hardly allows dynamic adaption to changing system requirements at runtime. In particular, the consequently priority unaware resource management policy further reduces the desired reactivity of time critical tasks. Some systems like TinyOS [1] and Contiki [2] follow an event-driven approach, where processes are implemented as event handlers that run to completion. Problematic in purely event-driven implementations is, that a lengthy computation completely monopolizes the CPU, which makes the system temporarily unable to quickly respond to external events. Additionally, it prohibits the interleaved execution of several computations. One solution to these problems is to provide a multi-threaded operation model. Although multi-threading extensions exist for some event-driven systems, they often lack important features like priorities or intelligent scheduling strategies (e.g. TOSThreads in TinyOS or Protothreads in Contiki). The basic architecture of these inherently non-preemptive systems also often limits the applicability of the multi-threaded approach and therewith the expected advantages.

Systems which inherently offer preemptive multi-threading are e.g. Mantis [3] and Nano-RK [4]. Nano-RK lacks dynamic priorites and uses the preemptive Priority Ceiling Protocol emulation Highest Locker Priority for resource assignment. This scheduling policy has the disadvantage that low prioritized tasks can block intermediate prioritized ones though the high priority task that defined the ceiling priority not even currently demands the resource in question. In fact, this leads to unnecessarily bad reactivity and can cause late results of the intermediate prioritized tasks. Mantis uses preemptive time-sliced priority based thread-scheduling with round-robin semantics within a priority level. Indeed, it provides no mechanism to limit priority inversion [5]. Although both operating systems provide a local system time, they do not offer the possibility to wait on events or resources with a limited timeout for reacting on various runtime imponderabilities. Finally, the low resolution of $1\ \mathrm{ms}$ prohibits precise event capturing as already required for various measurements like ultrasonic position estimation.

To meet the special requirements regarding reactivity and modularity of complex WSN/SANET applications, we developed the *Smart*OS real-time kernel to integrate four basic concepts. Most central is the inherent system timeline with a resolution of $1\ \mu\mathrm{s}$. It grants an individual local time for each node and is used for automatic timestamping of external events. Furthermore, it allows periodic tasks and limited waiting for events and resources. By definition, tasks are fully preemptive and possess a variable priority, each. This provides dynamic adjustment of their individual reactivity to changing system requirements and environmental conditions. The dynamic (de)allocation of resources at runtime allows tasks to use them just on demand and is essential for free composition of tasks to a more complex total system. While the priority inheritance protocol speeds up the resource handover from lower to higher prioritized tasks, we also implemented a mechansim of task cooperation which improves reactivity and even grants deadlock detection and recovery at runtime. Finally, an event concept is available for task interaction and synchronization. These events can be triggered by either tasks or external occasions.

In spite of its complexity regarding scheduling, task-interaction, time measurement and resource management, the presented application was successfully implemented on a single SNOW^5 [6] sensor node (MSP430@8MHz, 10KB RAM, 48KB ROM). With it, we achieved a localization frequency of approximately $6\ \mathrm{Hz}$ at a precision of around $2\ \mathrm{cm}$ and a track deviation of about $20\ \mathrm{cm}$.

IV. Conclusion

We showed, that real-time can already be relevant for today's WSN/SANET applications. Especially when task-parallelism and intense environmental interactions are required, available operating systems commonly lack special support. For enabling even resource constrained sensor nodes to execute complex real-time applications, we developed the embedded operating system *Smart*OS. Within $1.8\ \mathrm{kB}$ of ROM, *Smart*OS offers concepts which allowed us to successfully implement a complex vehicle tracking and steering application within a WSN. Therewith even difficult real-time problems can be mastered by using optimized software. For the future we expect that more powerful sensor nodes will enlarge the realization of even more ambitious applications. The design of *Smart*OS with its preemptive multi-threaded approach and its optimized scheduling and resource management policies suits a new broad class of applications which sensor networks already are or soon will be able to execute.

References

[1] U. Berkeley, "TinyOS," Web site http://www.tinyos.net/, UC Berkeley, 2004. [Online]. Available: http://www.tinyos.net/

[2] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 455–462.

[3] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms," *ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on Wireless Sensor Networks*, vol. 10, no. 4, pp. 563–579, August 2005.

[4] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-RK: An energy-aware resource-centric RTOS for sensor networks," in *RTSS '05: Proceedings of the 26th IEEE International Real-Time Systems Symposium*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 256–265.

[5] O. Babaoglu, K. Marzullo, and F. Schneider, "A Formalization of Priority Inversion," 1993. [Online]. Available: citeseer.ist.psu.edu/babaoglu93formalization.html

[6] M. Baunach, R. Kolla, and C. Mühlberger, "SNoW^5: A versatile ultra low power modular node for wireless ad hoc sensor networking," in *5. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze*, P. J. Marrón, Ed. Stuttgart: Institut für Parallele und Verteilte Systeme, 17.–18. Jul. 2006, pp. 55–59.

Proc. EWSN 2009 – Demos/Posters Session, Cork, Ireland, February 11th – 13th, 2009

Poster Abstract: Power Reduction by Adapting Strobed Preambles in Wireless Sensor Networks

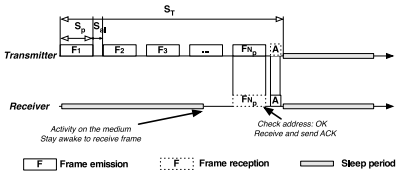
Erwing R. Sanchez*†, Claude Chaudet* and Bartolomeo Montrucchio†
*Département Informatique et Réseaux
TELECOM ParisTech, Paris, France
†Dipartimento di Automatica e Informatica
Politecnico di Torino, Torino, Italy

I. Introduction

Wireless Sensor Networks (WSN) are composed by small, battery-operated devices usually deployed to report measurement values to a central collection node. The primary objective of these dedicated networks is to fulfill their task during a long period of time. Hence, energy consumption is a constant concern for protocol designers in WSN. An important effort has been dedicated to optimize channel access mechanisms, since it is a critical point where a considerable amount of energy can be saved.

Duty cycling is one of the main approaches to reduce power consumption in the MAC layer. It consists on switching off the network interface as much as possible. Ideally, nodes sleep and wake up only to receive messages. This technique substantially reduces *idle listening*, which is the active time that nodes spent with radio frequency circuits turned on to wait for incoming frames. Duty-cycled medium access protocols can be either synchronous or asynchronous. Synchronized MAC protocols, such as SCP-MAC [1], require nodes to explicitly agree on a common schedule for sleeping and listening. They save energy effectively due to the reduction of idle listening because nodes are awoken only in specific slots of time. Asynchronous MAC protocols do not rely on a precise synchronization and use techniques such as preamble sampling, also known as *low power listening* (LPL), to facilitate communication between a sender with data and a duty-cycling receiver. Protocols belonging to this family, such as B-MAC [2] and X-MAC [3], do not restrict the time when a node can access the channel and may, in some cases, remove completely the problem of synchronizing nodes' clocks. This provides a greater flexibility to handle nodes densities, traffic loads and dynamic changes.

Our work focuses on such asynchronous MAC protocols and more specifically on improvements for the preamble sampling mechanism. We use a strobed data-preamble medium access protocol. This gives the possibility to a fast acknowledgement by the receiver, reducing the average length of the preamble. In addition, we utilize state information of the node in order to change the behavior of the protocol according to the position of the node in the topology tree.



(a) Transmission and reception (b) Tree topology

Fig. 1: Preamble sampling

II. Strobed Preamble Sampling with State Information

The general procedure of asynchronous duty cycling is presented in Figure 1a. While the sender transmits packets repeatedly, simulating a long preamble, the receiver deactivates its radio transmitter and turns it on only to sense the channel. Every potential receiver in the network schedules its radio to wake up and checks immediately for a carrier on the channel. If there is activity on the medium, the radio keeps listening to receive the subsequent frame. Otherwise, if the channel is idle, the node goes back to sleep and the whole cycle is repeated again. A node that has successfully detected energy on the channel prepares itself to receive a frame, the duty cycle procedure is momentarily blocked, and the radio is kept on for a sufficient time to receive the frame and to acknowledge it. After receiving the packet and sending the acknowledgment frame, the node goes back to its normal duty cycle schedule.

In our approach transmissions are constituted by strobed preambles, i.e., the transmitter sends the packet intermittently. We do not use dummy packets to form the strobed preamble. Transmitters send frames during a total time of S_T if no acknowledgment is received, otherwise they may return to sleeping mode after receiving an acknowledgement. S_T is compound by preamble frames and acknowledgement gaps, which have duration of S_p and S_{al} respectively. S_T must be at least as long as the standard sleeping period of the potential receiver in order for the preamble to be detected effectively.

When a transmission is performed, all nodes that are in the transmission range receive a frame preamble. Neighbors that are not the intended receiver can go back to sleep as soon as they realize the packet is not for them. Differently, the actual

receiver acknowledges the transmitter immediately in the gap following the transmission of the frame. Once the transmitter receives the acknowledgment message, it stops sending the preamble frames, thus, producing an overall reduction of power consumption.

Nodes closer to the sink have a high volume of data to forward, while leaf nodes simply transmits their own measurement results. Therefore, it seems sub-optimal to enforce the same duty cycle at every stage of this hierarchy. Information related to the position of the node in a tree topology, i.e. the number of hops separating it from the sink, can be used to improve the performance of the network.

In any level of the hierarchical tree, power consumption of a transmitting node depends on the intended receiver duty cycle. Nodes need to transmit their strobed preambles at least $\frac{R_l+Rs}{S_p+Sal}$ times to be certain that the receiving node obtains the packet correctly. R_s and R_l are the receiver's sleeping and waking time respectevely. This means that while for a receiver it is preferable to have long sleeping times, for a transmitter it is better that its intended receiver has a short sleeping time. Optimal energy consumption is reached when optimizing these conflicting statements. Since in a tree-like routing, the load of a given node is the sum of the loads of its children in the tree, it is expected that nodes closer to the network sink transmit more frequently. Hence, our approach consists on increasing the duty cycle of nodes closer to the sink.

III. Experimental Results

We run our MAC protocol through an empirical testbed of TelosB motes with TinyOS. We implemented an application that sends data in a frequent basis to a fixed destination. We implement counters in the MAC protocol that keep track of the time used to perform the various operations. Specifically, we implemented timers to track the time that the radio was in active or sleeping mode. The deployment topology used is shown in Figure 1b. The data flowed upwards, from low levels toward node N_1 which was the top-level sink. Children nodes provided their packets to their parent node. N_1 was the base station and was in the radio range of nodes N_2 and N_3. Differently, nodes N_2 to N_6 shared the same radio range. This setup shows the topology tree that can be formed in a wireless sensor network where a node is chosen as the parent and, hence, as the only recipient of a set of nodes.

The processor of the platform consumes lower energy compared with the radio module, thus in our experiments we allowed the application to run continuously and did not attempt to force it to idle mode. The duty cycle, then, is related to the radio module and the time it is turned on and off. For our experiments, the awakening period R_l is always 10 ms. Data packets were sent only by leaf nodes, i.e., N_4, N_5 and N_6. Internal nodes N_2 and N_3 did not generate data traffic by themselves. First, we executed our experiments under constant preambles (CP) for each node of the network, and second, we performed the same experiments with variable preambles (VP). We executed the experiments for different data rates: each 5 s, 2.5 s and 1.75 s. Figure 2 shows the

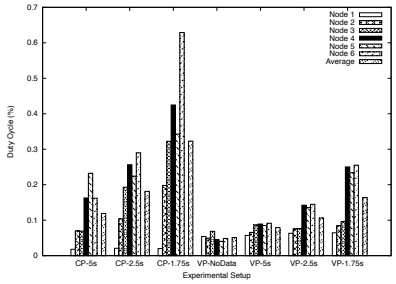


Fig. 2: Experimental duty cycle for every node and average.

TABLE I: Experimental energy consumption - High data rate.

Approach	N_1	N_2	N_3	N_4	N_5	N_6	Total
Constant	1,04	10,33	16,82	22,16	17,88	32,83	101,09
Adaptable	3,80	4,98	5,70	14,78	13,81	15,06	58,17

experimental duty cycle for every node and the average for the entire network. Duty cycles in the variable preamble (VP) experiments present a fair tendency. In the experiments where data is sent, the improvements compared to the constant approach are evident. In the least efficient case, which is the case with lower data rate – one packet every 5 seconds, we find an improvement of almost 34 % with the adaptable preamble approach. In the highest data rate experiment, the improvement in duty cycle is close to 50 % when using adaptable preambles.

Table I shows the energy utilized by the whole network when we compare constant and adaptable preamble approaches in the highest data rate instance. We suppose, for the adaptable preamble case, that every time the radio is turned on it is receiving a packet (highest consumption), while for the constant approach we suppose that every time the radio is turned on it is transmitting a packet (lower consumption). The energy consumed by the adaptable preamble approach is consistently lower than the constant case.

IV. Conclusions

We present a technique to improve overall energy consumption by adapting the preamble duration according to the position of the node. Further work will consist on characterizing the relationship that exists between the shape of the routing tree and the optimal duty cycle.

References

[1] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle mac with scheduled channel polling," in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2006, pp. 321–334.

[2] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004, pp. 95–107.

[3] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks," in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2006, pp. 307–320.

Proc. EWSN 2009 – Demos/Posters Session, Cork, Ireland, February 11th – 13th, 2009

Poster Abstract: Detection of abandoned/removed objects with a video sensor node aided by IR Sensor

©Michele Magno, Davide Brunelli, Luca Benini, 2009

Department of Electronics, Computer Sciences and Systems, University of Bologna
viale Risorgimento, 2 - 40136 Bologna, Italy - tel: +39 051 209 3787
{michele.magno,davide.brunelli,luca.benini}@unibo.it

***Abstract*—The interest in low-cost and small size video surveillance systems able to collaborate in a network has been increasing over the last years. Thanks to the progress in low-power design, research has greatly reduced the size and the power consumption of such distributed embedded systems providing flexibility, quick deployment and allowing the implementation of effective vision algorithms performing image processing directly on the embedded node.**

In this paper we present a multi-modal video sensor node designed for low-power and low-cost video surveillance able to detect changes in the environment. The system is equipped with a CMOS video camera and Pyroelectric InfraRed (PIR) sensors exploited to reduce remarkably the power consumption of the system in absence of events. We analyze different configurations and characterize the system in terms of runtime execution and power consumption.

I. Introduction

Nowadays video surveillance systems and cameras aimed at monitoring private and public areas are gaining increasing popularity. In this paper we propose a system for video analytics which deploys synergically a PIR sensor and a smart camera. The aim of our method is to automatically detect structural changes occurring in a monitored scene which are due to events such as abandoned or removed objects in the scene. This class of events is often of critical importance for security reasons: as a matter of fact, a typical video analytics issue is the automatic detection of abandoned objects in venues such as airports or railway stations or the theft of objects in places such as museums and private properties [1].

A major issue for a reliable detection of such events is the presence of stationary cluttered parts of the scene caused. This can easily cause an automatic video-surveillance system to produce false alarms. Our idea is to use the information coming from the PIR sensor to help increasing the robustness of a change detection algorithm used to detect this class of events.The PIR sensor can trigger the camera sensor only in absence or intrusion in the monitored scene to reduce the numbers of false alarms which could be produced by the change detection algorithm.

Moreover, power management is a critical issue when dealing with wireless and distributed embedded systems and it is well known that batteries do not scale as much as electronic devices [2]. We will show that by adopting PIR sensors the average power consumption of the platform will be reduced remarkably.

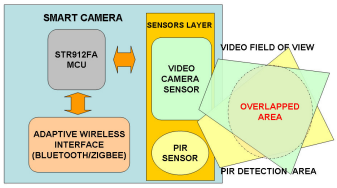


Fig. 1. Video sensor node architecture.

II. System Architecture

The block-level architecture of our video sensor is displayed in Fig. 1 and consists of several modules: the vision board which hosts both the CMOS imager and the PIR sensor with a common area under monitoring, the wireless module, the microprocessor and other peripherals.

The node architecture is centered around an ARM9 from STMicroelectronics. It is employed mainly for digital image processing and provides configurable and flexible power management control. Power consumption can be dynamically managed by firmware. For example a typical current consumption for this microcontroller is about $1,7\,mA/MHz$ in RUN mode and only a few mA in sleep mode which is an attracting feature for wireless sensor networks design where the power consumption is a main constraint.

The goals guiding the electrical design of this node architecture were the integration of a video camera and a Pyroelectric sensor. The whole system is designed with low power consumption as the primary goal. The vision module includes a SXGA CMOS color digital camera targeted for mobile applications and a Pyroelectric Infrared Detector, whose detection area is overlapped with the field of view of the video sensor. Wireless communication capabilities have been supported through a suitable interface for both ZigBee and Bluetooth compliant transceiver.

III. Video Processing Algorithm

The following application example demonstrates the use of a smart camera with PIR. Note that in this application the PIR sensor plays a two-fold role: it is adopted both to wake up the camera and to provide information for a more accurate processing of the image. Since the field of view of the PIR overlaps the field of view of the image sensor, the main advantage of using the PIR sensor is the awareness of objects presence/absence in the surveilled area.

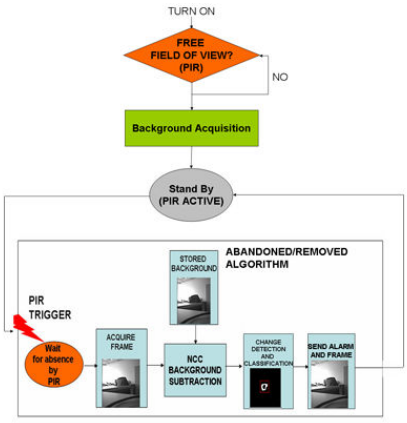


Fig. 2. Flow chart of the background change detection task.

Fig. 2 depicts the main steps of the implemented algorithm. After switching on the system, the microcontroller reads the digital output of the PIR sensor and does not start acquiring the background until the PIR signal is low for at least five seconds.

When the Background Process is over, the system switches to a stand-by mode. In this mode the ARM9 microcontroller runs in SLEEP mode and its power consumption becomes notably lower, as it will be shown later. In addition, also the camera is switched to a stand-by mode to reduce the power consumption. Instead the PIR sensor stays in run mode since it is used to wake up the microcontroller.

When an intrusion event which occurred within the field of view of the smart camera is over, the PIR sensor alerts the microcontroller by turning high its output signal. At this point, the ARM waits a certain amount of time for the output of the PIR to remain low, so to make sure no intrusion is currently occurring in the scene and thus to enhance robustness towards false alarms. Then the microcontroller switches on the camera sensor and starts acquiring image frames and a motion detection algorithm based on NCC background subtraction (BS) is applied.

Once obtained the changed mask from BS, a multi blob detection starts, it searches all the regions of interest (ROI) in the changed mask deleting the ROIs not greater than a fixed value. Finally the classification algorithm takes every ROI one by one and analizes all the edges of the current frame to understand if the edge is present in the current frame (abandoned object) or not (removed object).

When the algorithm is done an alarm is communicated to a host station via Bluetooth. It can be a simple message that is send to tell, if the object is removed or abandoned, or the whole frame or only the ROI of the current frame.

IV. Experimental Results

The above-mentioned application was fully implemented in ARM9 firmware. In the following we will focus on mote sensor power and performance.

A. Sensor Node Characterization

The ARM microprocessor STR912F offers configurable and flexible power management control which allows dynamic power consumption reduction. We can switch the microcontroller in three global power control modes: RUN, IDLE and SLEEP.

Mode	Power $[mW]$
Active with video sensor	626,5
Active, without video sensor	484,5
Alarms Transmission	572,5
Sleep, only PIR is Active	51

TABLE I
Power consumption of the video sensor node.

Task	Energy $[mJ]$	Time $[ms]$
Frame Acquisition	72	115
NCC Background Subtraction	417,5	860
ROI detection	8,5	17.5
Abandoned/revoved detection Worse case ROI 160x160	19,5	40
Image Transmission	801,5	1400

TABLE II
Energy/Time requirements of the Hardware/Software system.

B. Detection of Abandoned/Removed Objects

Fig. 3 shows screenshots of the proposed change detection example. In the first preliminary test, only with 60 frames for the abandoned object and 60 frames for the removed object, the accuracy is 90% for abandoned and 88,33% for removed. In the future the tests and the classificator algorithm will be improved.

V. Conclusions

An integrated video sensor node for energy efficient surveillance able to detect changes in the scene has been proposed. The adoption of a multimodal platform equipped with different family of vision sensor with heterogeneous features of power consumption and resolution allow to achieve longer autonomy reducing considerably the overall power consumption of the system.

References

[1] M. Beynon, D. Van Hook, M. Seibert, A. Peacock, and D. Dudgeon. Detecting abandoned packages in a multi-camera video surveillance system. In *Proc. IEEE Conf. on Advanced Video and Signal Based Surveillance (AVSS 03)*, pages 221–228, 2003.

[2] T. Paradiso, J.A.; Starner. Energy scavenging for mobile and wireless electronics. *Pervasive Computing, IEEE*, 4(1):18–27, Jan.-March 2005.

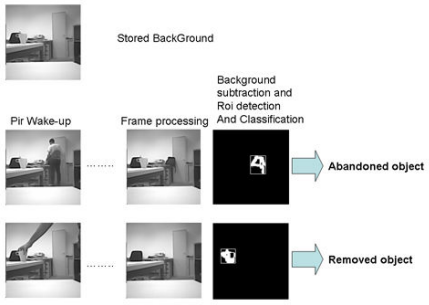


Fig. 3. Screenshots of abandoned/removed object detection example

Poster Abstract: An Extensible Dashboard for Sensor Networks Control and Visualisation

A.G. Ruzzelli, M. Dragone, R. Jurdak†, C. Muldoon, A. Barbirato, and G. M. P. O'Hare
CLARITY: The Centre for Sensor Web Technologies
University College Dublin Belfield, D4, Ireland
† CSIRO ICT Centre, Australia.

I. Introduction

A tool to dynamically reconfigure and monitor online the wellness of Wireless Sensor Networks (WSNs) is fundamental to developers both for debugging purposes and for gaining a deep understanding of inter-node communication. For the final users, the tool should provide a user-friendly interface for setting events and tuning application parameters. We present OctopusJB, an extensible JavaBeans-based dashboard to control the topology and behaviour of the network through a network map and interact with the network by reconfiguring and monitoring a number of parameters of application and radio, e.g frequency channel, nodes duty-cycle and monitoring energy consumption. The dashboard allows formulating and injecting composite queries into the network while data can be logged into a file for network analysis. A Network Chart plots live data for network analysis while a 2D Floor Plan allows node localization support.

OctopusJB [2] is open-source software developed specifically for TinyOS version 2 that provides a modular programming architecture. OctopusJB can be utilised as an effective debugging and assessment tool for new underlying algorithms and modules by simply adding them in the OctopusJB configuration. The standard configuration of the dashboard utilises the collection tree protocol (CTP) and the low power listening (LPL) access control defined in TinyOS 2.

Facilitating future extensions and code reuse conforming to certain standards is an important objective of OctopusJB that implements a JavaBeans component-based architecture to distinguish between a component-based infrastructure framework and a set of functional components. The architecture is based on the features associated to the BeanContext class. This eases much of the component composition while the abstract API enables different possible implementations of these features. This architecture enables new or existing components to be plugged-in without altering the remainder of the system. The objective is to provide a high degree of configurability and the creation of different versions of OctopusJB in order to satisfy specific deployment scenarios and application requirements.

OctopusJB greatly improves the state of the art of network state monitoring and network control. Currently, Mviz [1] is the standard tool for TinyOS v.2 that provides a basic visualization with no features for remote network control.

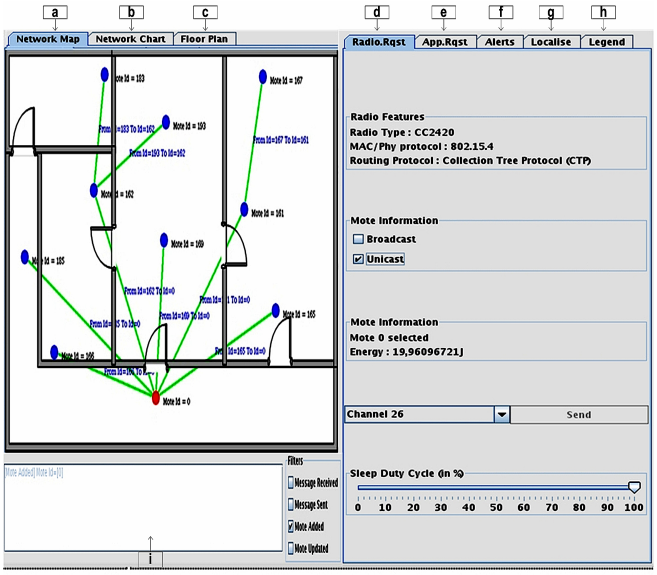


Fig. 1. A Screenshot of OctopusJB with connections and location of nodes.

II. Panels Highlights

The dashboard provides 3 main views namely Network Map *NP*, Network Chart *NC*, and Floor Plan *FP* as shown in Figure 1 tabs a,b, and c. On the right-hand side of the dashboard, a side menu contains support tools for the user to formulate application requests, tune radio parameters, set email and sms alerts, and localise nodes in the network. Incoming packets can be visualised live by the Consol panel at the bottom of the board as shown in Figure 1 tab i that provides also some checkboxes to filter the messages displayed. A Legend panel allows selecting more visualisation options on the main board and logging the data into a file.

We now illustrate the basic architecture and logical organization of OctopusJB, highlighting the basic functionalities of the Radio and the Application panels. Then, we detail some of the advanced functionalities of the Alert and Localise panels.

A. The Basic Functionalities

The basic functionalities are provided by the panels of Radio Requests (*RR*), Application Requests (*AR*), and Legend accessible through the tabs d,e and h in Figure 1, respectively. At network setup, the *NM*, in Figure 1, is populated with nodes

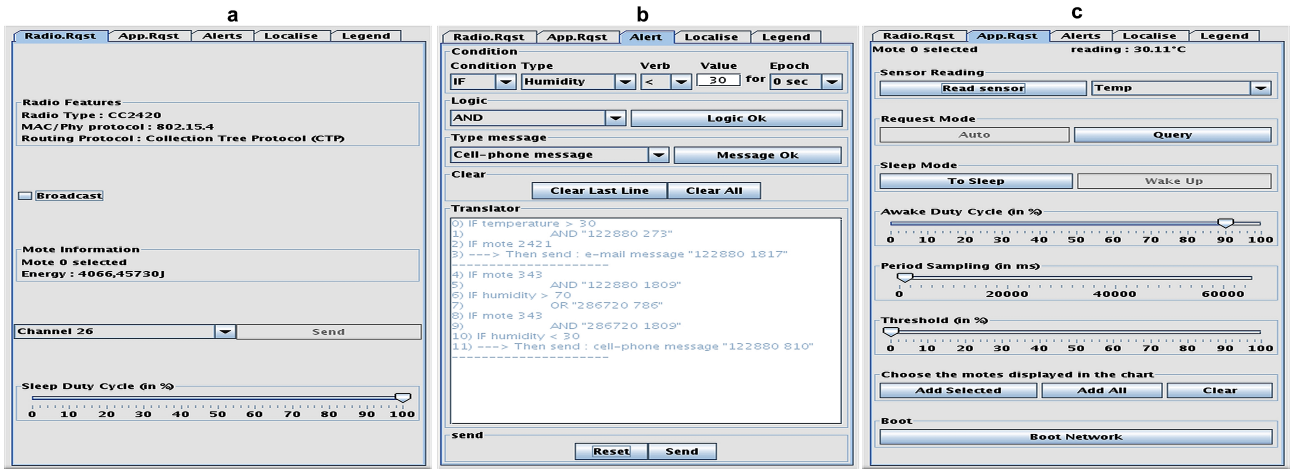


Fig. 2. A Screenshot of the Radio Requests, Alerts, and Application Requests panels in OctopusJB.

and routing connections from nodes to the gateway. Figure 2a presents the *RR* panel more in detail. The panel provides a "*sleep duty cycle (in %)*" slider for changing the node duty cycle. Dragging the slider to the wanted value and releasing the mouse generates a request packet from the gateway to the network. The request packet can be broadcast to the entire network or by selecting individual nodes on the *NM* and unicasting to them. The panel also lists the available radio channels to allow switching to a new frequency channel of operation. This will generate a new request which is broadcast by the gateway and must be acknowledged by all nodes. The *RR* panel provides estimation of the energy spent by each node since start-up. This is calculated by timestamping radio, CPU and sensor activities. Nodes calculates the energy according to provided data-sheet parameters then transmit the value to the gateway at a very low frequency interval. From the *AR* panel shown in Figure 2c, the user can select sensed values to be displayed live in the *NC*. From this panel, the user can select either time-driven or event-driven data collection from nodes. Selecting the Request Mode "*Auto*" activates transmission at a fixed interval while a "*threshold*" slider regulates a basic event-driven data reports. Setting a threshold to 0 is equivalent to activating nodes in auto mode. The *FP*, selected through the tab in Figure 1c, displays a floor map of the physical location of the network. *NM*, *NC* and *FP* can also be displayed in overlapping modalities. Figure 1 shows the *NM/FP* view. Initially, nodes are located randomly on the board, however, the combined view allows to run an interactive localisation algorithm, see Section II-B. Finally, a Legend panel allows the setting and the visualisation of further network parameters.

B. The Advanced Functionalities

The Advanced Functionalities include (1) the Alert panel for injecting composite queries into the network and send alerts to the users; (2) the Localise panel provides support for the nodes localisation.

The Alert panel, in Figure 2b, provides a user-friendly system to compose sentence-like queries. The panel consists of 3 blocks: the "*Query Sentence*", the "*Logic*" and the "*Alert Type*". Each query sentence consists of a *subject*, a *verb*, a *value*, and an *epoch*. The subject allows for the selection of packet fields like nodeID, sensor reading or link quality. A list of fields is generated automatically by the Message Interface Generator (MIG) provided in TinyOS. The *verb* represents the verifying condition while the *epoch* denotes the duration of the event before the sentence is validated. The *Logic* interlaces sentences to formulate composite queries such as "*if a node is within certain xy coordinate and temperature is greater than z then report*". Prior to setting an alert, the correctness of a sentence is evaluated on the screen through the "*Translator*" toolbox. At present, OctopusJB supports the composition of a maximum of 10-sentence query, converting each sentence into an efficient 40-bit data representation. The composite query is injected into the network when the "*Message Ok*" button is pressed. The GUI sets a countdown timer and waits for notification from the nodes before alerting the user. Nodes are provided with an NesC interpreter module to decode the sentence and activate the enquired mode. Queries also have an expiration time set on the node-side.

Finally, the dashboard provides support for interactive node localization (*ILS*) and comparison. By the Localise Panel, in Figure 1 tab g, the user can first upload an XML-based map onto the board containing coordinates, shape and size of the area network. Following, the user can drag and drop nodes of known location (anchors) in the correct position on the *FP*. Releasing the mouse generates a packet transmitted to the anchors or broadcasted to the network. This allows nodes to run their localization algorithm. After completion, nodes transmit their estimated location to the GUI that enables localizing them on the *FP*. If the accuracy obtained is not sufficient, the user may decide to repeat the process by dragging a further anchor to the correct position on the *FP*. Nodes' placements can also be saved to avoid running the algorithm every time the network is rebooted.

REFERENCES

[1] Mviz http://www.tinyos.net/tinyos-2.x/apps/MViz/
[2] The Octopus Dashboard http://www.csi.ucd.ie/content/octopus-dashboard-sensor-networks-visual-control