Proceedings of the First

REALWSN 2005

Workshop on Real-World Wireless Sensor Networks

Stockholm, Sweden
20-21 June 2005

SICS Technical Report T2005:09
ISSN 1100-3154
ISRN:SICS-T–2005/09-SE

Welcome to the first REALWSN workshop!

It is our greatest pleasure to welcome you to the first REALWSN workshop on Real-World Wireless Sensor Networks in the beautiful city of Stockholm during the brightest days of the year. We first want to thank you all, both authors, attendees and members of the technical program committee for make this event possible.

As the name of the workshop suggests, REALWSN is a forum for people interested in real world issues in the fascinating research area of wireless sensor networks. While analysis and simulation of sensor networks are indispensable, the key difference of wireless sensor networks to other related research areas is the number of possible applications and the possible impact on society, industry and our daily lives. Deploying real sensor networks is a challenging task. The main objective of REALWSN is to bring together researchers and practitioners to boost the state of the art in this exciting field.

The program with carefully chosen contributions from the 43 submissions reflects the real-world focus of this workshop. The technical program covers topics from sensor network hardware and software development for sensor nodes to real-world sensor network applications.

Thanks again to all people who contributed to the workshop: the technical program committee, Kersti Hedman and L-H Orc Lönn at SICS and not to forget our sponsors VINNOVA, the Swedish Agency for Innovation Systems and the EU E-Next Network of Excellence for providing travel grants.

We are looking forward to an exciting and inspiring workshop in Stockholm.

The REALWSN organization team
Adam Dunkels
Bengt Ahlgren
Per Gunningberg
Sverker Janson
Christian Rohner
Thiemo Voigt

Organization committee

Adam Dunkels, SICS, Sweden
Bengt Ahlgren, SICS, Sweden
Sverker Janson, SICS, Sweden
Per Gunningberg, Uppsala University, Sweden
Thiemo Voigt, SICS, Sweden

Program committee chairs

Thiemo Voigt, SICS, Sweden
Christian Rohner, Uppsala University, Sweden

Program committee

Tarek Abdelzaher, University of Virginia, USA
Leif Axelsson, Ericsson Microwave Systems, Sweden
Mats Björkman, Mälardalen University, Sweden
Torsten Braun, University of Berne, Switzerland
Erdal Cayirci, Istanbul Technical University, Turkey
Jerker Delsing, Luleå University of Technology, Sweden
Adam Dunkels, Swedish Institute of Computer Science, Sweden
Jakob Engblom, Virtutech AB, Sweden
Kevin Fall, Intel Research Berkeley, USA
Laura Feeney, Swedish Institute of Computer Science, Sweden
Per Gunningberg, Uppsala University, Sweden
Paul Havinga, University of Twente, Netherlands
Holger Karl, University of Paderborn, Germany
Jim Kurose, University of Massachusetts, USA
Pedro José Marron, University of Stuttgart, Germany
Prasant Mohapatra, University of California, Davis, USA
Chiara Petrioli, University of Rome, Italy
Hartmut Ritter, Free University Berlin, Germany
Kay Römer, ETH Zürich, Switzerland
Jochen Schiller, Free University Berlin, Germany
Cormac Sreenan, University College Cork, Ireland
Ivan Stojmenovic, University of Ottawa, Canada
Andras Veres, Ericsson Research, Hungary

Reviewers

Zinaida Benenson, RWTH Aachen, Germany
Frank Eliassen, Simula Research Laboratory, Norway
Joakim Eriksson, SICS, Sweden
Sverker Jansson, SICS, Sweden
Gábor Németh, Ericsson Research, Hungary
Martin Nilsson, SICS, Sweden
Fergus O'Reilly, Cork Institute Of Technology, Ireland
Miklós Aurél Rónai, Ericsson Research, Hungary

Program

Session 1: Applications

- **Glacial Environment Monitoring using Sensor Networks**, Kirk Martinez, Paritosh Padhy, Alistair Riddoch, Royan Ong, Jane Hart
- **The Heathland Experiment: Results And Experiences**, Volker Turau, Christian Renner, Marcus Venzke, Sebastian Waschik, Christoph Weyer, Matthias Witt
- **SensorScope: Experiences with a Wireless Building Monitoring Sensor Network**, Thomas Schmid, Henri Dubois-Ferriere, Martin Vetterli

Session 2: Localization and location dependent services

- **Lost in Space Or Positioning in Sensor Networks**, Michael O'Dell, Regina O'Dell, Mirjam Wattenhofer, Roger Wattenhofer
- **Improving Location Accuracy by Combining WLAN Positioning and Sensor Technology**, Paul Hii, Arkady Zaslavsky
- **Using Wireless Sensors as Selection Devices for a Multimedia Guidebook Scenario**, Montserrat Ros, Matthew D'Souza, Michael Chan, Konstanty Bialkowski, Adam Postula, Neil Bergmann, Andras Toth

Session 3: Software development for sensor nodes

- **Timber as an RTOS for Small Embedded Devices**, Martin Kero, Per Lindgren, Johan Nordlander
- **Using Protothreads for Sensor Node Programming**, Adam Dunkels, Oliver Schmidt, Thiemo Voigt
- **Driving Forces behind Middleware Concepts for Wireless Sensor Networks**, Kirsten Terfloth, Jochen Schiller

Session 4: Dealing with limited resources

- **Processor Choice For Wireless Sensor Networks**, Ciaran Lynch, Fergus O'Reilly
- **Power Characterization of a Bluetooth-Equipped Sensor Node**, Magnus Lundberg, Jens Eliasson, Jason Allan, Jonny Johansson, Per Lindgren
- **Realizing Robust User Authentication in Sensor Networks**, Zinaida Benenson, Nils Gedicke, Ossi Raivio

Session 5: Short presentations

- **Using REWARD to detect team black-hole attacks in wireless sensor networks**, Zdravko Karakehayov
- **Sensor Networks and The Food Industry**, Martin Connolly, Fergus O'Reilly
- **Experimental construction of a meeting model for smart office environments**, Daniel Minder, Pedro Marron, Andreas Lachenmann, Kurt Rothermel
- **TinyREST - A Protocol for Integrating Sensor Networks into the Internet**, Thomas Luckenbach, Peter Gober, Andreas Kotsopoulos, Kyle Kim, Stefan Arbanowski

- **Real World Issues in Deploying a Wireless Sensor Network for Oceanography**, Jane Tateson, Christopher Roadknight, Antonio Gonzalez, Taimur Khan, Steve Fitz, Ian Henning, Nathan Boyd, Chris Vincent, Ian Marshall

Poster presentations

- **Embedding a Microchip PIC18F452 based commercial platform into TinyOS**, Hans-Jörg Körber, Housam Wattar, Gerd Scholl, Wolfgang Heller
- **ZigBee-ready modules for sensor networking**, Johan Lönn, Jonas Olsson, Shaofang Gong
- **Use of Wireless Sensor Networks for Fluorescent Lighting Control with Daylight Substitution**, Fergus O'Reilly, Joe Buckley
- **Wireless sensor networks in precision agriculture**, Aline Baggio
- **Intelligent Sensor Networks - an Agent-Oriented Approach**, Björn Karlsson, Oscar Bäckström, Wlodek Kulesza, Leif Axelsson
- **A Tree-Based Approach for Secure Key Distribution in Wireless Sensor Networks**, Erik-Oliver Blass, Michael Conrad, Martina Zitterbart
- **Wireless Sensor Network-Based Tunnel Monitoring**, Sivaram Cheekiralla
- **Simulation of Real Home Healthcare Sensor Networks Utilizing IEEE802.11g Biomedical Network-on-Chip**, Iyad Al Khatib, Axel Jantsch, Mohammad Saleh

Session 1: Applications

Glacial Environment Monitoring using Sensor Networks

K. Martinez, P. Padhy, A. Riddoch
School of Electronics and Computer Science
University of Southampton
United Kingdom
+44 (0)2380 594491
{km, pp04r, ajr}@ecs.soton.ac.uk

H.L.R. Ong
Department of Engineering
University of Leicester
United Kingdom
+44 (0) 116 252 5683
hlro1@leicester.ac.uk

J.K. Hart
School of Geography
University of Southampton
United Kingdom
+44 (0)23 80594615
jhart@soton.ac.uk

ABSTRACT
This paper reports on the implementation, design and results from GlacsWeb, an environmental sensor network for glaciers installed in Summer 2004 at Briksdalsbreen, Norway. The importance of design factors that influenced the development of the overall system, its general architecture and communication systems are highlighted.

General Terms
Measurement, Documentation, Performance, Design.

Keywords
Low Power, Radio Communications, Environmental monitoring, Glaciology, sensor networks

1. INTRODUCTION
Continuous advancements in wireless technology and miniaturization have made the deployment of sensor networks to monitor various aspects of the environment increasingly feasible. Unfortunately, due to the innovative nature of the technology, there are currently very few environmental sensor networks in operation that demonstrate their value. Examples of such networks include NASA/JPL's project in Antarctica [1], and Huntington Gardens [2], Berkeley's habitat modelling at Great Duck Island [3], the CORIE project which studies the Columbian river estuary [4], deserts [5], volcanoes [6] and glaciers [7]. The research efforts in these projects are constantly thriving to a pervasive future in which sensor networks would expand to a point where information from numerous such networks (e.g. glacier, river, rainfall, avalanche and oceanic networks) could be aggregated at higher levels to form a picture of the environment at a much higher resolution. This paper highlights real-world experiences from a sensor network, GlacsWeb, which was developed for operation in the hostile conditions underneath a glacier.

To understand climatic change involving sea-level change due to global warming, it is important to understand how glaciers contribute by releasing fresh water into the sea. This could cause rising sea levels and great disturbances to the thermohaline circulation of the sea water. The behaviour of the sub-glacial bed determines the overall movement of the glacier and it is vital to understand this behaviour to predict future changes. During the summer of 2004, we deployed our network in Briksdalsbreen glacier, Norway. The aim of this system is to understand glacier dynamics in response to climate change. Section 2 of this paper provides a simple overview of the system architecture. Section 3 highlights a list of factors that helped design the system. Section 4 presents a synopsis of results obtained from the system post deployment. Section 5 concludes with future work and the summary of the system.

2. SYSTEM ARCHITECTURE
The intention of the environmental sensor network was to collect data from sensor nodes (*Probes*) within the ice and the till (sub-glacial sediment) without the use of wires which could disturb the environment. The system was also designed to collect data about the weather and position of the base station from the surface of the glacier. The final aspect of the network was to combine all the data in a database on the *Sensor Network Server* (SNS) together with large scale data from maps and satellites. Figure 1 shows a simple overview of our system.

The system is composed of *Probes* embedded in the ice and till, a *Base Station* on the ice surface, a *Reference Station* (2.5 km from the glacier with mains electricity), and the *Sensor Network Server* based in Southampton.

Before deployment into the ice, the probes were programmed to wake up every 4 hours and record various measurements that included, the temperature, strain (due to stress from the ice), the pressure (if immersed in water), orientation (in the 3 dimensions), resistivity (to determine if they were sitting in sediment till, water or ice) and their battery voltage. This method provided 6 sets of readings for each probe everyday.

The base station was programmed to talk to the probes once a day at a set time. It is powered up from its standby state for approximately 5 minutes everyday, during which, it collects data from the probes and reads the weather station measurements. Once a week it also records its location with the differential GPS, which takes 10 minutes. This time is often used to remotely login from the UK for maintenance. After it has performed these tasks, it sends all the collected information to the reference station PC via long range radio modem. Figure 2 shows the sequence of events occurring during and beyond its operating window describing the communication process between probes, base, reference station and the Southampton Server.

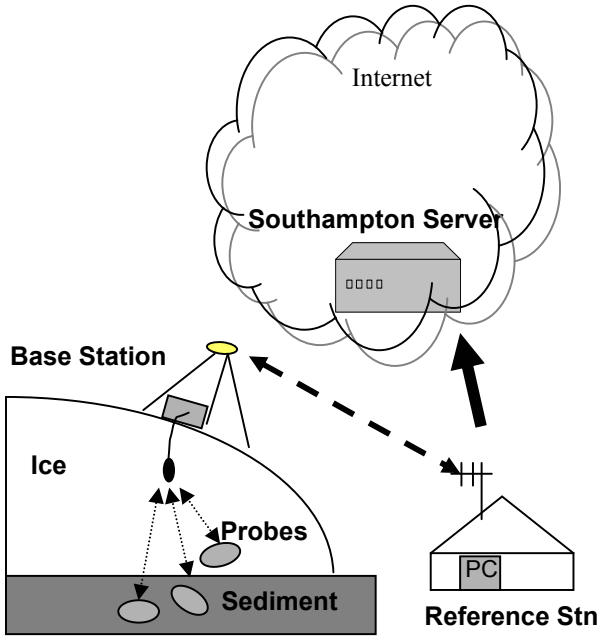

Figure 1: Simple Overview of the System

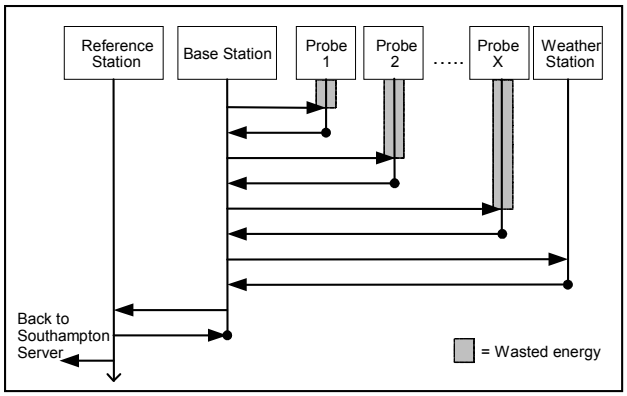

Figure 2: Sequence of Events during Communication

The reference station is configured to upload all unsent data to the SNS via an ISDN dial-up every evening. This data is stored in a database where it is being used by glaciologists to interactively plot graphs for interpretation.

3. DESIGN FACTORS

In a sub-glacial environment, nodes can be subject to constant immense strain and pressure from the moving ice. Therefore, a robust sensor design, integrated with high levels of fault tolerance and network reliability was developed. The design of the system was influenced by a comprehensive list of factors including scalability, power consumption, production costs and hardware constraints [8]. These factors served as essential guidelines for the design structure of the network and the chosen protocol for communication. The rest of the section discusses the impact of each factor on the design.

3.1 Production Cost

It is usually the case that sensor networks consist of a large number of sensor nodes and more often than not if the cost of the network is more expensive than the cost of deployment, the sensor network is not cost-justified. Taking into consideration, however, the hostile environment and the hazards that the nodes were expected to face without failing over a long duration of time, it was a pragmatic decision to invest substantially in the development of the nodes. The final cost of each probe came to an estimated £177. A total of 8 probes were deployed.

3.2 Power Consumption

3.2.1 Probes

Each probe was powered with six 3.6V Lithium Thionyl Chloride cells providing 6AH worth of energy. The cells were chosen due to their high energy density and good low temperature characteristics. The probes were designed to consume only 32µW in their sleep mode, where only the real time clock and voltage regulators are powered. In power mode the probe consumes 15mW when the transceiver is disabled, 86mW when it is on but idle, 370mW when receiving, and 470mW whilst transmitting state. The probes wake up every 4 hours for 15 seconds to take measurements and then go back to sleep. They were programmed to communicate with the base station once a day when they power up for a maximum of 3 minutes. During this window they attempt to send their data readings to the base. An approximate calculation of a probe's daily power consumption turns out to be 5.8mWH. Theoretically, this means at this rate the probe could last for at least 10 years.

3.2.2 Base Station

The Base station was powered with lead-acid gel batteries powered with a total capacity of 96AH (1152WH). These batteries fed power to a StrongARM-based embedded computer (BitsyX), GPS, GSM and long range communication modules. The Bitsy consumes 120mW in sleep mode and 1.45W when operating. The base station is powered up for a maximum of 15 minutes a day during which it communicates with the probes, takes measurements, reads weather station and sends data to the reference station. The estimated power consumption during this job is approximately 4W (1WH hour per day). This combined with a consumption of 170mW (120mW *BitsyX* + 50mW Weather Station average) in sleep mode, the total estimated daily consumption is 5WH. This means that the batteries should last approximately 230 days. The batteries were connected in parallel with two solar panels (15W in total) to produce 15WH per day during summer that would approximately provide an additional 100 days of energy. This implied that base station would survive for almost up to a year without being attended to.

3.3 Transmission Media

The communication module for our probes like most other sensor networks was also based on RF circuit design. There were, however, a few variations to our design to accommodate better transmission through ice. Based on the failure of the previous version of the probes [7], the communication frequency between probes and base station was halved from 868MHz to 433 MHz. Antenna size grows problematically with any further decrease in frequency. The presence of liquid water presents a problem when trying to use radio waves in glaciers especially during summer

because the englacial water scatters and absorbs the radio signals making it difficult to receive coherent transmissions [9]. Thus by halving the frequency, one is essentially doubling the wavelength which would be larger than the size of the majority of water bodies that could impede the signal. The radiated RF power was also increased significantly by using transceiver modules that incorporated a programmable RF power amplifier that boosted the transmission power to over 100mW to improve the signal penetration through ice. To further improve communications, base station transceivers were also buried 30-40m under the ice connected via serial (rs232) cables.

3.4 Scalability

The system is infrastructure based, i.e. all nodes are only one hop away from the base station. The polling mechanism used for communication between the probes and the base station, although has a natural advantage over other contention based protocols due to reduced duty cycles and no overhead and collisions, one could argue problems arising with the deployment of additional new nodes. The base station runs Linux, using sequence of shell scripts and a custom "cron"-like scheduler to complete its daily jobs. One can assume full control of the system and reconfigure the scripts to update the communication schedule that would adapt to the new probes without hampering the system's operation.

3.5 Fault tolerance

Most sensor networks are catered to face multiple sensor node failures without upsetting the functioning of the entire network. In a system like ours where only a limited number of nodes are available for disposal in the glacier, it is very crucial that all aspects of the system are robust. The glacier's environment is nevertheless very hostile to allow smooth operation of the system including communication. Therefore some very vital measures were taken in order to sustain the network functionalities, even at the cost of time delay, during breakdown of the system.

3.5.1 Probe Failure

The probe's firmware was designed to have a segment called user space (3k words) that could be altered. It can hold programs that are autonomously executed whenever the Probe awakens. Programs could be loaded or removed from the user space and this provides flexibility to alter the probes functioning from anywhere in the world. A watchdog timer placed on the firmware ensured that any rogue programs loaded into the user space were terminated if it exceeded some preset timeout. It also ensured that the program was not automatically executed next time the probe awakened.

3.5.2 Base Station Failure

In an event where the base loses communication with the reference station over the long range modem, the GSM modem is activated. This allows data to be sent directly to the UK server via text messages (SMS). The probes house a 64Kb Flash ROM which is organized as a ring buffer. The six sets of measurements recorded by the probe over one day use 96 bytes and are time stamped and stored in the Flash ROM. This allows the probe to store up to 682 days worth of data in the event of a short range link failure where the base fails to communicate with the probe.

3.5.3 Communication Failure

An authentic communication packet was developed to specially cater for the system due to the limited resources provided by the PIC microcontroller embedded in the probes. The packet size varies between 5 and 20 bytes. The gap between each transmitted byte was set to a maximum of 3ms to ensure spurious data didn't inhibit valid communication. The packet incorporates a checksum byte that allows checking the packet's integrity at the receiver. If a communication error is detected, the sender can retry sending. The limit on the number of retries during failures was set to 3 as a compromise between reliability and power consumption. In practice few retries are ever seen.

3.6 Hardware Constraints

3.6.1 Probe constraints

A typical sensor node comprises of 4 basic modules. These are a power module, a sensing module, a processing module and a transceiver module. All these units needed to fit into a palm-sized module that could be easily dispatched into the glacier's bed via holes 70m long and 20 cm wide. As shown in figure 4, all the electronics were enclosed in a polyester egg-shape capsule measuring 14.8 x 6.8cm. The round shape simplified insertion into the drilled holes.

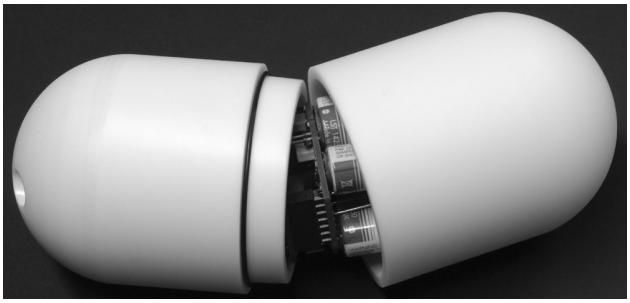

Figure 4: Probe shown open

Our probe electronics was divided into 3 sub-systems: digital, analogue and radio each of which was mounted on separate octagonal PCBs. This efficiently utilized the available volume and modularized the design.

PIC microcontrollers are low-cost, small sized RISC computers with low power consumption. The probes used embedded PIC processors to configure, read and store the attached sensors at user-specified times, handle power management and communicate with the base station. The length of the capsule was designed so that it could also accommodate a conventional ¼ wavelength "stubby" helical antenna fixated on the radio module.

3.6.2 Base Station constraints

The base station was one very critical aspect of the network as the entire operation of the network depended on it. Due to its location on top of the surface of the glacier, several measures were taken in order to ensure safety and efficiency. The base station was held together with the help of a permanent weather and movement tolerant pyramid structure as seen in figure 4. The electronics and the batteries were housed in two separate sealed boxes. Their weight in total stabilized the entire base station by creating a flat even surface as they melted the ice beneath. The long pole in the middle of the pyramid was used to mount the GPS antenna, the

long range modem antenna to communicate with the reference station and the anemometer connected to the weather station in the box. The solar panels were attached directly on top of the boxes in order to minimise wind-drag.

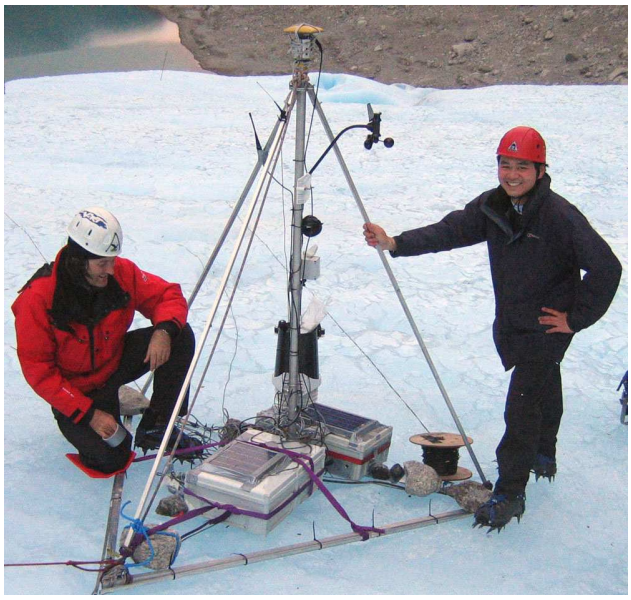

Figure 4: Base Station and the Pyramid, showing solar panels, battery box, antennas and weather station.

3.7 Topology

Unlike many sensor networks, we decided not to deploy the probes in an arbitrary fashion. The deployment site of the glacier was surveyed before hand using Ground Penetrating Radar (GPR) to determine any sub-glacial geophysical anomalies (e.g. a river). Based on this survey, the 8 probes were deployed in holes within 20m of a relay probe which was suspended 25m into a central hole. The main reason why this was done was due to the range of the probes. In air the probes can communicate over a distance of 0.5km. In ice, however, their range decreases considerably.

4. RESULTS

4.1 Probe Data

8 probes were deployed in August 2004. At the end of deployment, the base station managed to collect data from 7 probes. During the course of the next few months, however, communication access was reduced to only 3 probes. Namely probes 4, 5 and 8. This failure can be attributed to one or all of the following three reasons.

4.1.1 Range of Probe Transceivers

As discussed before, the range of the probe transceivers was restricted to just under 40m in ice. Although the base station was attached to wired-transceivers inserted in the ice to improve data gathering, the loss of communication with the probes may imply that the sub-glacial movement of the ice could have carried the probes out of transmission range. This was not unexpected, however, in the time available it was not feasible to develop a multi-hop ad-hoc network of probes that could cater for such problems.

4.1.2 Base Station Breakdown

The base station operated properly from August until November when it experienced power failure. This meant that although the probes were still functioning, data could not be retrieved from them with a dead base station. A small team went to the glacier for two days to repair and reactivate the base station.

It is estimated that the probes' real time clocks drift up to 2 seconds everyday. In order to synchronise them, the base station updates their clocks everyday during its 15 minute window using broadcast packets. The base clock is set to GPS time once a week. Failure of the base station for a period this long may imply that the probes could have drifted a minute at most outside the base station's polling window. This problem is currently being investigated by a trial and error method where the polling window is shifted slightly everyday.

4.1.3 Probe Breakdown

Another simple explanation for this communication failure could be that the probes died due to various reasons such as immense stress of the ice or short circuits due to the presence of water. These causes of failure are very hard to avoid and the only way to overcome them is to make more probes that could increase the chances of data gathering. Internal sensors to monitor health may also help in the future. The probes that did communicate with the base station for the duration till now have shown a significant improvement over the previous system. The previous version saw only 1 probe operating over a period of 14 days. Figure 5 shows a sample of data gathered by probe 8 during the month of January 2005.

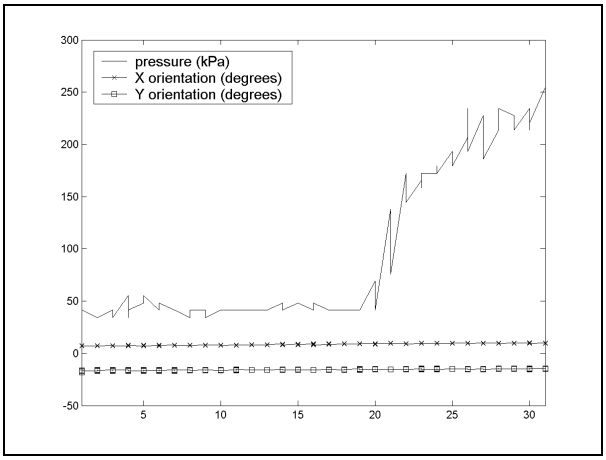

Figure 5: January readings from Probe 8

The graph indicates us how the probe is undergoing an increase in pressure as the month progresses. This means that the probe is being subjected to the full pressure of the ice. A graph from the weather station in figure 6 during the same period shows increasing humidity towards the end of the month which could imply there was rain or snow. The graph also indicates stability in the probe's its x and y axis orientation. This could mean that probe is fixed in one position and thus, still communicating.

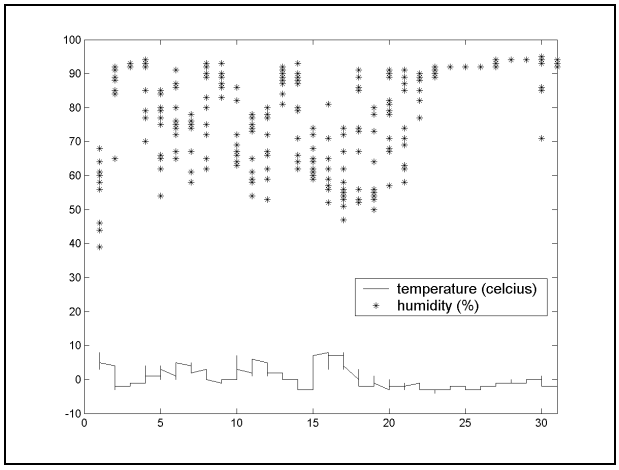

Figure 6: January weather readings from base station

4.2 Power Issues

The base station ran out of power during the peak of winter. A possible explanation for this could be that snowfall had covered the solar panels not allowing the batteries to charge. It was discovered that there is enough wind on the surface of the glacier throughout the year to produce electricity using a wind turbine. This has been noted and the next version of the system will involve a small wind generator in addition to solar panels.

Figure 2 shows power being wasted by probes whilst waiting for the base station to poll. A better protocol could not be implemented due to time constraints and risks, the cost of probes and the nature of the deployment environment. This issue is important as probe power savings would be extremely crucial in a future network where an ad-hoc protocol would be implemented.

5. CONCLUSION AND FUTURE WORK

This study is one of the first in a glacial environment and we managed to talk to 3 probes regularly out of the 8 deployed. This was a significant achievement that demonstrated that this system is robust and can be operated in the hostile environment of a glacier. We believe the reason for failure to communicate with the remaining probes is due to their non ad-hoc nature as they must have moved out of communication range from the base station. Our future aim is to implement a multiple hop, self-organising ad-hoc network of probes that would not only ensure scalability but also reduce power consumption. These aims are fostered by keeping into consideration that a future network would involve more nodes covering a larger area and more than one base station. Use of a much more standardized protocol would improve communication with more probes and ensure a better understanding of the sub-glacial environment.

6. ACKNOWLEDGMENTS

The authors thank the Glacsweb partners Intellisys and BTExact. Topcon and HP for equipment support. Thanks also Harvey Rutt, Sue Way, Dan Miles, Joe Stefanov, Don Cruickshank, Matthew Swabey, Ken Frampton and Mark Long, Sarita Ward, Hanna Brown and Katherine Rose. Thanks to Inge and Gro Melkevol for their assistance and for hosting the *Reference Station*.

7. REFERENCES

[1] K.A. Delin, R.P. Harvey, N.A. Chabot, S.P. Jackson, Mike Adams, D.W. Johnson, and J.T. Britton, “Sensor Web in Antarctica: Developing an Intelligent, Autonomous Platform for Locating Biological Flourishes in Cryogenic Environments,” 34th Lunar and Planetary Science Conference, 2003.

[2] http://sensorwebs.jpl.nasa.gov/resources/huntington_sw31.shtml

[3] R. Szewczyk, et al., “Lessons from a Sensor Network Expedition,” Proceedings of the 1st European Workshop on Wireless Sensor Networks (EWSN '04), January 2004, Berlin, Germany, pp 307-322.

[4] D.C. Steere, et al., “Research Challenges in Environmental Observations and Forecasting Systems,” Proc. ACM/IEEE Int. Conf. Mobile Computing and Networking (MOBICOMM), 2000, pp. 292-299.

[5] K.A. Delin, S.P. Jackson, D.W. Johnson, S.C. Burleigh, R.R. Woodrow, M. McAuley, J.T. Britton, J.M. Dohm, T.P.A. Ferré, Felipe Ip, D.F. Rucker, and V.R. Baker, “Sensor Web for Spatio-Temporal Monitoring of a Hydrological Environmental,” 35th Lunar and Planetary Science Conference, League City, TX, 2004.

[6] K. Lorincz, D. Malan, Thaddeus R. F. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G.Mainland, S. Moulton, and M. Welsh, “Sensor Networks for Emergency Response: Challenges and Opportunities”, Special Issue on Pervasive Computing for First Response, Oct-Dec 2004.

[7] Martinez, K., Hart, J.K., Ong. R. (2004). Environmental Sensor Networks. Computer, 37 (8), 50-56.

[8] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., Cayirci, E., A Survey on Sensor Networks., IEEE Communications Magazine, August 2002.

[9] Gades, A.M., C.F. Raymond, H. Conway, H. and R.W. Jacobel, 2000. Bed properties of Siple Dome and adjacent ice streams, West Antarctica, inferred from radio echo-sounding measurements, Journal of Glaciology, 46(152), 88-94

The Heathland Experiment: Results And Experiences

V. Turau, C. Renner, M. Venzke, S. Waschik, C. Weyer, and M. Witt

Hamburg University of Technology, Department of Telematics
Schwarzenbergstraße 95, 21073 Hamburg, Germany
turau@tu-harburg.de

ABSTRACT

This paper reports on the experience gained during a real-world deployment of a sensor network based on the ESB platform in the heathlands of Northern Germany. The goal of the experiment was to gain a deeper insight into the problems of real deployments as opposed to simulated networks. The focus of this report is on the quality of radio links and the influence of the link quality on multi-hop routing.

1. INTRODUCTION

In recent years wireless sensor networks have been attracting research interest given the recent advances in miniaturization and low-cost, low-power design. Many algorithms have been proposed to solve the problems inherent to sensor networks, foremost resource limitations and high failure rates. The vast majority of algorithms has not been implemented on real sensor networks, but evaluated using simulation tools. Simulations are a valuable and cheap means to compare specific aspects of different algorithms solving the same problem (e. g., routing or data aggregation), but currently no simulation tool is capable to allow for all imponderabilities of a real deployment of a sensor network in a harsh environment over a longer period of time. To attain a deeper insight into sensor networks, experiments with real deployments are indispensable. But up to today the number of deployed wireless sensor networks is extremely low compared with the number of publications.

Environmental monitoring is a significant driver for wireless sensor network research, promising dynamic, real-time data about monitored variables of an area and so enabling many new applications. Because of this, it comes as no surprise, that almost all real experiments were conducted with this application background. In particular, the first published experience with real deployments of sensor networks were about habitat monitoring [4]. Only recently other application backgrounds such as wildfire monitoring were considered in real experiments [1].

In this paper we report about an experiment with a real deployment of a sensor network with 24 nodes running for two weeks in March 2005 in the heathlands of Northern Germany. After the deployment the application ran without any human attention. At the time of writing this report the experiment was just finished. The preliminary results allow an insight into the problems emerging during the deployment and operation of a sensor network and provide valuable information for other installations. The focus of the following analysis is on the communication between the nodes.

2. THE GOALS

The Heathland experiment is to our knowledge the first outdoor long term usage of the Embedded Sensor Board (ESB) platform [3]. Naturally, we were interested in the overall performance of the system. Specifically, this experiment provided a basis to evaluate our neighborhood exchange and routing protocol and acted as an indicator for the feasibility of our deployment and debugging strategy. In particular the following aspects are analyzed in this paper:

- General radio performance (packet loss, packet errors)
- Distribution and constancy of transmission ranges
- Stability and quality of links as determined by the neighborhood protocol
- Communication in a multi-hop environment

To conduct this analysis, a considerable amount of data was logged by the sink node (about 6 MB per day). Whenever a packet with a sensor reading was sent to the sink, the state of the node (neighborhood list, remaining energy etc.) was included in the packet. Some data was also stored in the EEPROM of each node; this was used to analyze causes of failure of a node, e. g., when the node was no longer reachable by other nodes.

The Heathland experiment was not aimed at evaluating the long term operation of the sensor network, hence the application code was not optimized to achieve an increased expectation of life.

3. THE EXPERIMENT

3.1 The Hardware

For the experiment, the ESB nodes from the Free University Berlin were used [3]. They consist of the micro controller MSP 430 from Texas Instruments, the transceiver TR1001, which operates at 868 MHz at a data rate of 19.2 kbit/s, some sensors, and a RS232 serial interface. The radio transmit power can be tuned in software, with 0 being the minimum and 99 the maximum. Each node has 2 KB RAM and 8 KB EEPROM. The nodes were powered by three AA batteries. The sink had a permanent power supply. The power consumption of the nodes according to the specifications of the vendor varies from 8 μA in sleep mode up to 12 mA when running with all sensors. A description of the sensors of the ESB nodes can be found in [3].

3.2 The Packaging

The nodes had to be prepared for diverse weather conditions including snow, rain, and sunshine (see Figure 1). Waterproof packing was essential. Therefore the nodes were shrink-wrapped together with desiccant bags. The foils were then placed in waterproof boxes, which again were put into plastic bags. These bags were affixed on trees, windows etc. using gaffer tape. The recorded temperature during the experiment varied in a range of almost 40 °C. Figure 2 shows the developing of the temperature during the two weeks of the experiment as measured by the sensor nodes (in- and outdoor temperature). Our results indicate that the applied packaging was appropriate (no node failed due to extraneous cause). The applied packaging formed an insulation of the nodes and affected the measurements of the sensors. Since this was not tangent to our goals, it was accepted.

Figure 1: The packaging of the sensor nodes

3.3 The Application

A broad class of applications of wireless sensor networks are so-called *sense-and-send* applications. They share a common structure, where sensors deployed in a wide-ranging area are tasked to take periodic readings, and report results to a central repository. The implemented application follows in principle this *sense-and-send* pattern. The nodes periodically send readings from their five sensors supported by the hardware [3] to the sink. To account for topology changes caused by node failures, deployment of new nodes, decreasing communication radii due to fading battery energy, and moving nodes, routing trees and neighbor relationships were recomputed regularly and unlike other experiments (e. g. [5]) a multi-hop network was used.

The design of the software follows the style of separate layers. The bottom layer consists of a proprietary firmware that is shipped together with the ESB nodes. The included radio transmission protocol is very simple; its unreliableness is the source of many problems. The protocol silently discards a packet after 15 unsuccessful trials of submission. On top of this a neighbor discovery protocol called *Wireless Neighborhood Exploration* (WNX) has been implemented. It is a slightly modified implementation of TND, the proactive neighbor discovery protocol of TBRPF as defined in RFC 3684 [2]. WNX determines uni- and bidirectional links

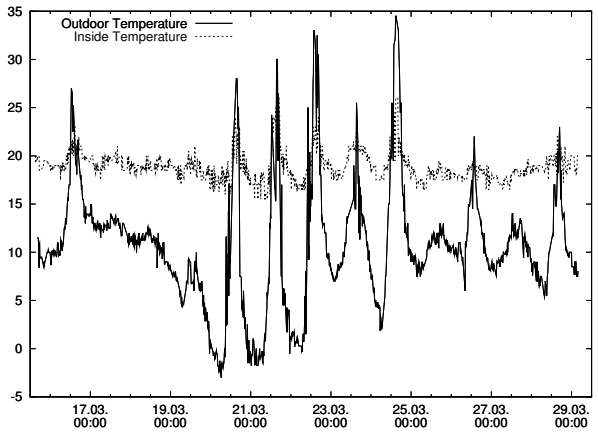

Figure 2: Recorded temperatures during experiment

and adds a quality descriptor to every unidirectional link. A link becomes a bidirectional link if the qualities of both link directions exceed a given limit. Based on the bidirectional links provided by WNX, a depth-first spanning tree rooted at the sink is built. The depth-first search is performed using a distributed algorithm developed by Tsin [6]. Each node keeps a list of at most eight neighbors, high-quality links are preferred over low-quality ones. The spanning tree is used to route messages from any node to the sink. The reason for using depth-first spanning trees is twofold:

- In order to better test multi-hop communication, routing trees with a higher depth are needed. Depth-first trees usually have a higher depth than breadth-first trees, which can easily be computed using flooding.
- Since depth-first search is the basis for many algorithms which are likely to be employed in a sensor network, we were interested in the performance of our distributed implementation.

The neighborhood relation is used to build the routing tree. Therefore, WNX is only executed at the beginning of every application cycle. For this reason a suspend mode was added to WNX. Since the leaves of a routing tree are not needed to route messages to the sink, they only turn on their radio:

- while running WNX,
- during the depth-first search, and
- when they send their sensor readings to the sink.

As a consequence, leaf nodes turn off their radio for about 46 minutes during every hour.

After the completion of the depth-first search all nodes send their neighbor list including the quality factors to the sink. After this step all leaf nodes send their data in intervals of 10 minutes to the sink. To reduce the likelihood of interferences, the times a leaf node sends its readings are randomly distributed in this interval. Apart from the sensor readings the following data is included in every packet:

- time-stamp with respect to local clock
- remaining battery energy
- number of packets received and sent
- number of packet transmission retries
- information about clock drift

Since it was known from preliminary tests with the ESB nodes that the quality of communication links varies considerably over time, it was decided to repeat periodically the determination of the neighbors of each node and to build a new depth-first tree thereafter. This step was accomplished at the start of every clock hour.

Due to the uncertainty in the radio communication it was decided to use a time triggered scheme in which activities are initiated by the progression of a globally synchronized time-base. Since a considerable clock drift between individual ESB nodes was observed, the sink sends periodically its local time into the network. Table 1 lists the points in time within the span of one hour and the associated actions. All nodes including the sink have the same code; to turn a node into the sink a single flag needs to be set. This considerably facilitated the deployment process. The sink node sent all data over a standard serial port interface to a PC, that stored the data in files.

Time	Action	Result
0	Reset	Resets state of node
1	Start WNX	Nodes send Hello packets, compute link qualities and determine bidirectional links
9	Suspend WNX	List of bidirectional links with link quality
12	Start depth-first search	Depth-first search tree
14	Start measurements	Leaf nodes turn off radio, inner nodes turn off sensors, in intervals of 10 minutes • leaf nodes turn on radio • all nodes send measured data and link states to sink, leaf nodes turn off radio • the sink sends its local time to all nodes in tree

Table 1: Periodic sequence of triggered actions

3.4 The Deployment

The nodes were deployed in a rectangular area with dimensions 140 times 80 meters. The territory was mainly heathland, some spots with taller trees and three smaller buildings. For most pairs of nodes the line-of-sight was obstructed. The majority of the nodes was attached to trees at a height of about 4 meters, some on poles just above the ground surface and 4 nodes were positioned on different floors inside the main building (including the sink). Figure 3 depicts the location of the nodes and of the main building, the sink (square node) and marks the three nodes that failed immediately after the start of the experiment (empty circles). Furthermore, it shows a sample depth-first tree.

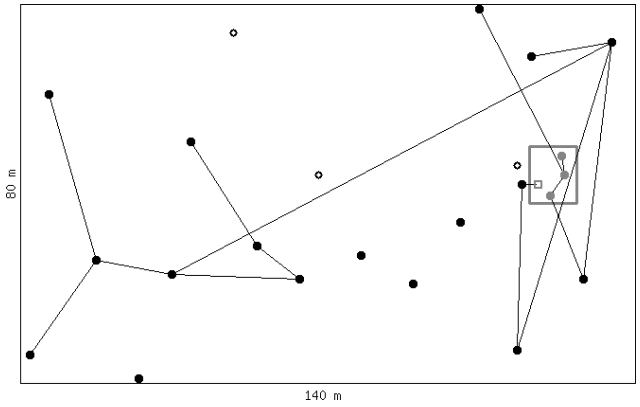

Figure 3: Location of sensor nodes

The software allowed the nodes to operate in two different modes: deployment and application mode. After the nodes were attached to their positions, the software was in deployment mode. Using an additional node attached to a portable computer it was possible to configure the nodes in place by walking through the area. Within the transmission range packets could be sent directly to a node using its identifier. Among other things this allowed

- to query the reachable neighbors of a node (including the quality factors)
- to adjust the transmission range
- to switch into the application mode

The deployment mode made it possible to create a topology according to the experiment's needs. In particular, the numbers of neighbors of a node was limited in order to reduce radio interferences. This approach proved to be useful.

4. ANALYSIS

The following discussion is confined to an analysis of the quality of the communication and the consequences for the application, i. e., the depth-first search. Altogether 24 nodes were used in the experiment, three of the nodes did not send or receive any data after the first day (the reasons are not known). Among the remaining 21 nodes, out of the 210 different pairs of nodes, 45 demonstrably appeared as links in a search tree, but the quality of these links varied considerably.

Link-quality for each neighbor was computed based on the *hello-history*, a bit-field indicating whether or not a hello has been received in the past. Each node kept track of the last 32 expected hellos from each neighbor. Periodically, received hellos – weighted with a coefficient – were summed up to represent link-quality. These coefficients have been selected to be a linear function of the hello-age: more recent hellos were assigned a higher one than very old hellos. To be precise, the latest expected hello was weighted with a value of 11.84, whereas the value of each older hello was

reduced by 0.25. Hence, link-quality was an integer between 0 and 255, with the latter being the optimum. In addition, very old hellos had less influence. This approach was chosen to support the suspending phase of WNX: the link-quality must not only indicate how many hellos have been received on the long run, but also how many hellos have been received shortly before.

The application mainly used a unicast transmission scheme with acknowledgments provided by the firmware. In case acknowledgments were not received up to 15 retransmissions were tried. This leads to a maximal latency of 1143 ms, not including the waiting time for channel access. We observed latencies up to 3 seconds. About 50 % of all unicasts were successful. Out of these transmissions the average number of retries was 3.89 with a standard deviation of 2.19. This suggests, that the limit of 15 maximal retransmissions was chosen too high. A lower value may have led to less congestion and may have increased the overall success rate.

Figure 4 depicts the quality values of a link between two nodes inside the building. On the average the quality is above 200 with rather limited variation over time (the following three figures also display the mean value and the standard deviation). Figure 5 depicts the quality values of

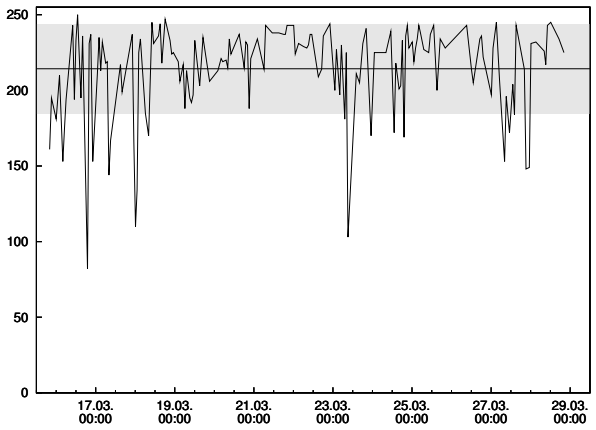

Figure 4: Quality of a link between two indoor nodes

a link between two outdoor nodes. The average link quality is well below 200 with a rather high variation over time. This is a typical example for outdoor links, in some cases the variation was even higher (see Figure 6). Outdoor nodes were on the average much farther apart from each other than indoor nodes. For some links the number of entries in our log file was too low to make a fair judgment. Overall it can be said, that the a priori assessment of the quality of a link is very difficult. We observed cases where a pair of nodes could not communicate despite line-of-sight, while the opposite phenomenon also occurred. Similar to [7] nodes received packets frequently from high quality neighbors, but also occasionally from more remote nodes. Many nodes had asymmetric links, i. e., the link quality in one direction was high, and low in the opposite direction. These links are not used in the application. Moreover, we experienced a correlation between packet size and transmission success: The larger the packets, the lower the probability that the packets reached their destination.

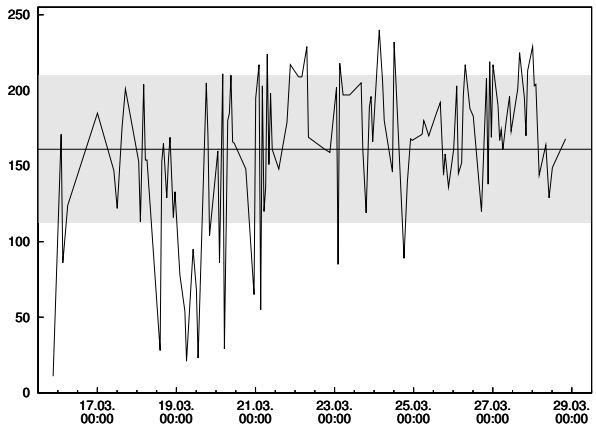

Figure 5: Quality of a link between two outdoor nodes

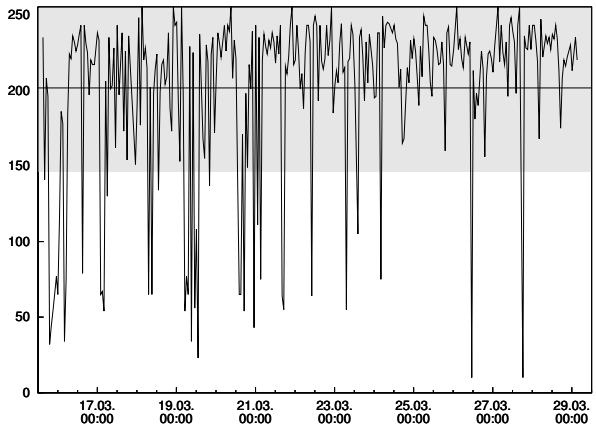

Figure 6: Large variation of link quality over time

Despite the discouraging results about the quality of the links, the depth-first search produced surprisingly good results. The distributed depth-first search was started from the sink 317 times in a period of two weeks. The search successfully terminated in 205 of these cases (in four cases the sink did not reach any node). A successful depth-first search does not imply that all nodes of the network were visited. The largest tree consisted of 19 nodes (over 90 % of all active nodes). Figure 7 displays the distribution of depth-first trees with respect to their size. More than 50 % of all successfully built trees included more than 10 nodes (i. e. 50 % of the nodes). Trees with 17 nodes even occurred 25 times; this is a rather astonishing result, against the background of the quality of links discussed earlier. The trees varied considerably, which is another sign of the changing quality of the links. Not surprisingly, links with high average quality appeared more often as edges of the depth-first tree. The link whose quality is depicted in Figure 4 arose in more than 50 % of all trees.

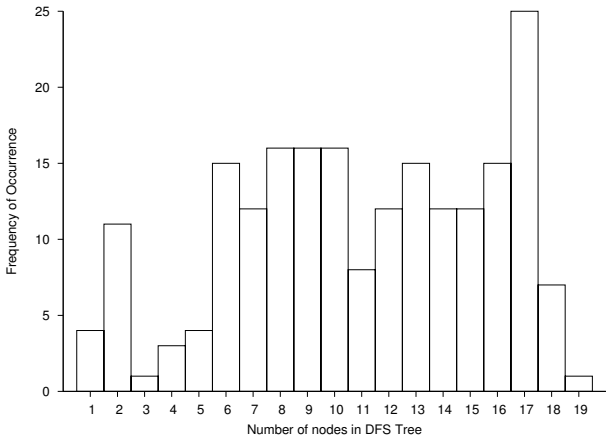

Figure 7: Distribution of total number of nodes in depth-first search trees

The depth-first search trees were used as routing trees, to forward the data measured at nodes towards the sink. Obviously, the successful transmission of a packet towards the root correlates with the depth of the node in the routing tree. Figure 8 displays the relationship between successful delivery of a measurement packet and the depth of the corresponding node in the routing tree. Expectedly, the rate of success drops approximately exponentially with the depth d. The plotted curve $100 \cdot 0.8^d$ closely approximates the delivery rate of the measurement packets. Hence, the average delivery rate can fairly well be predicted based on the average qualities of the individual links. This allows an estimation of the maximal acceptable hop count for multi-hop routing.

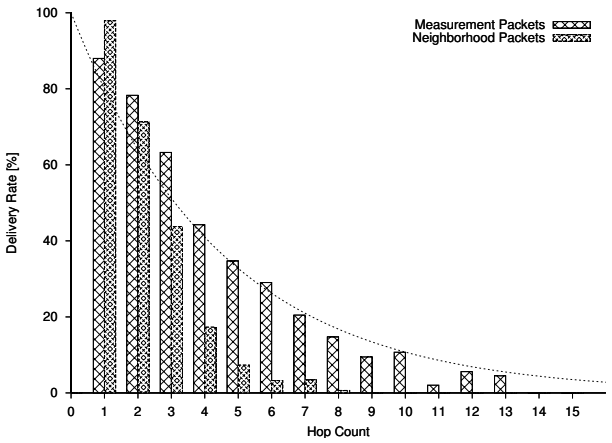

Figure 8: Transmission success in relation to hop count

The transmission success of neighborhood packets exhibits a different run of the curve. The size of a measurement packet is about 70 bytes including the packet header. Packets reporting the neighborhood list to the sink are more than 10 % larger than measurement packets. This leads to a significantly lower success rate as can be seen from Figure 8. The drop of the success rate was much higher than expected: at depth five the success rate is about 7 % compared to 35 % for the smaller packets. The main reason for this is probably not the larger packet size but is related to congestion. Neighborhood packets were sent t seconds after the measurement packets, t randomly chosen between 0 and 15. Due to the retransmission of packets this time difference became very small after a few hops. This explains the lower success rates of neighborhood packets for higher hop counts.

5. CONCLUSION

Like every scientific experiment, real deployments of experimental sensor networks need careful planning and first of all a clear definition of the goals. In sensor networks data logging is almost the only means to acquire data, consequently a logging strategy must be developed in order to collect the data to derive the intended goals. After the deployment there is usually no possibility to intervene in the logging process. The breakdown into a deployment and an application mode proved to be very useful, especially for changing the topology.

As a first conclusion it can be stated, that link quality estimation and neighborhood management are essential to reliable routing in sensor networks. The quality of individual links varies over time for no apparent reasons and unidirectional links of good quality occur more often than bidirectional links of similar quality. This observation suggests, that the concept of unit disk modeling used in many theoretical investigations is not an appropriate model at all. The following lessons can be learned from the experiment: larger packets should be broken up into smaller ones, the number of retransmissions should be modest, the transmissions should be carefully scheduled to avoid congestion, and a good understanding of the implementation is indispensable.

6. REFERENCES

[1] D. M. Doolin and N. Sitar. Wireless sensors for wildfire monitoring. *Proc. SPIE Symp. on Smart Structures & Materials/NDE 2005, San Diego*, Mar. 2005.

[2] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (TBRPF), RFC 3684, 2004.

[3] ScatterWeb GmbH. The Embedded Sensor Board. `http://www.scatterweb.net`, 2005.

[4] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6):34–40, June 2004.

[5] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *Proc. of the First European Workshop on Sensor Networks (EWSN)*, Jan. 2004.

[6] Y. H. Tsin. Some remarks on distributed depth-first search. *Inf. Process. Lett.*, 82(4):173–178, 2002.

[7] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. First Int. Conf. on Embedded Networked Sensor Systems*, pages 14–27, 2003.

SensorScope: Experiences with a Wireless Building Monitoring Sensor Network *

Thomas Schmid Henri Dubois-Ferrière Martin Vetterli†

Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland

t.schmid@ieee.org, {henri.dubois-ferriere, martin.vetterli}@epfl.ch

ABSTRACT

This paper reports on our experience with the implementation, deployment, and operation of SensorScope, an indoor environmental monitoring network. Nodes run on standard TinyOS components and use B-MAC for the MAC layer implementation. The main component on the server side is a Java application that stores sensor data in a database and can send broadcast commands to the motes.

SensorScope has now been running continuously for 6 months. The paper presents an analysis of three 2 week periods and compares them in terms of parameter settings, and their impact on data delivery and routing tree depth stability. From the data gathered, we show that network performance is greatly improved by using MAC layer retransmissions, that SensorScope is running in a none congested regime, and we find an expected mote lifetime of 61 days.

The phenomena discussed in this paper are well known. The contribution of this paper is an insight to a long running sensor network that is more realistic than a testbed with a wired back-channel, but more controllable than a long-term, remote experiment.

1. INTRODUCTION

Environmental monitoring is considered as one of the prime application fields for sensor networks today [3]. Examples include monitoring of natural habitats [4] [7], volcanic activity [8], or building structures [9]. While the number of deployed sensor networks is steadily rising, sensor networking technology is still in its infancy, and long-lived, large-scale sensor network deployments remain a challenge.

There is an inherent trade-off between realism and observability in experimental evaluation of sensor networks (Figure 1). At one extreme, simulations offer complete control and visibility into experiments, but they cannot faithfully reproduce all the parameters that affect a live system. At the other extreme, absolute realism comes with full deployments, often in remote locations, such as Great Duck Island [7]. Unfortunately deploying such a system requires significant resources; furthermore the constraint of a remote deployment means that code updates must be extremely conservative (in order to reduce the risk of system crashes), and the amount of monitoring data that nodes can report is typically kept very low in order to maximize node lifetime.

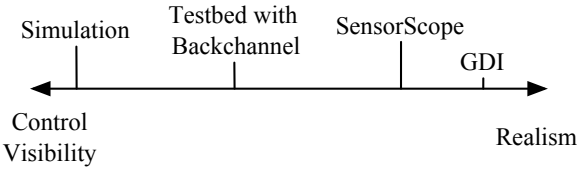

Figure 1: Trade-off between in-network visibility and realism. Long-term, remote experiments such as Great Duck Island are the most realistic. Short-term experiments on a testbed with a wired back-channel and power supply add a large degree of realism compared to simulations since they expose the system to the vagaries of real radio channels. SensorScope represents an intermediate point in the trade-off spectrum.

This paper reports on our experience with the implementation, deployment, and operation of SensorScope, an indoor monitoring sensor network. The network consists of 20 mica2 and mica2dot motes, equipped with a variety of sensors for light, temperature and sound. Motes use a multi-hop routing tree to report sensor readings and network monitoring information back to the base-station.

SensorScope represents an intermediate trade-off between realism and visibility. Unlike a powered testbed, it is a long-running (since October 2004) and dedicated deployment. Nodes are powered with batteries, even though we could have used a continuous power supply, so as to expose the network to the vagaries of node brownouts and blackouts. Unlike with a remote deployment, we can still reboot and debug motes, allowing us to be less conservative in making software updates to the network. Nodes can also send more monitoring information since battery lifetime is not as critical as in a remote system.

A second aim of SensorScope is to consider a full end-to-end system going all the way from the sensors to a user-visible front end. All information coming out of the network, such as routing tree information (Figure 2) and sensor data (Figure 3), is stored in a database, and made accessible via

*This work was supported (in part) 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.

†Also with Dept. of EECS, UC Berkeley

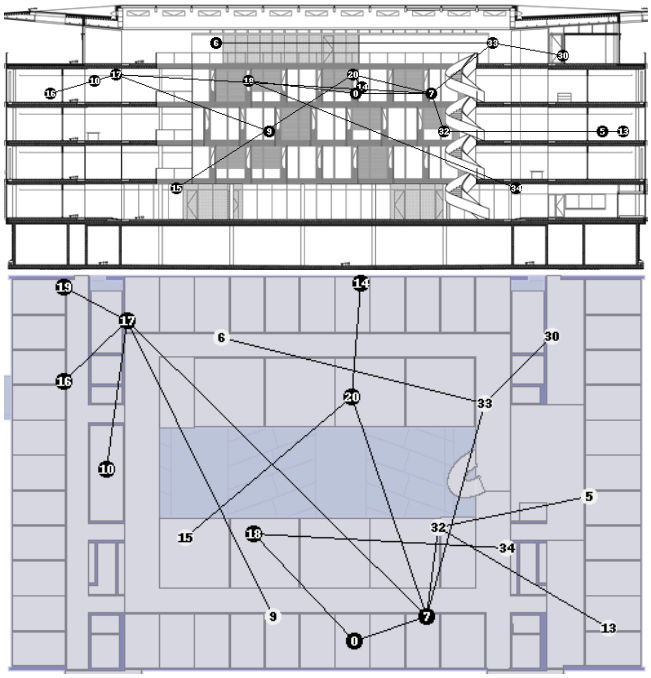

Figure 2: Network routing graph on March 16, 2005. The top image is a longitudinal section, the bottom one a floor plan of the building where SensorScope is installed. The different colors of the motes on the bottom image represent motes on the same floor (black) and motes on an other floor (white) than the floor plan shows.

a public web interface[1]. Through this website we provide to the research community a full data set (both historical archives and current data) containing both application data (sensor readings) and network monitoring data.

The rest of the paper is organized as follows: Section 2 describes the SensorScope system. Then, Section 3 discusses network performance and channel utilization. Finally, Section 4 gives some concluding remarks and an outlook for future work on SensorScope.

2. SYSTEM DESCRIPTION

2.1 Mote-Side

Nodes run a TinyOS application that was designed to be representative of simple, small-scale environmental monitoring networks. The application has two basic duties: to periodically sample sensors and route readings back to the base-station, and to interpret and disseminate command broadcasts. As a routing substrate, we use the standard (`tos/lib/Route`, later `tos/lib/MintRoute`) multi-hop routing implementation that is part of the TinyOS distribution. Protocol constants are left to their default values. Moving down in the stack, we use the B-MAC [6] MAC layer implementation and its low-power listening scheme. We also integrated the Deluge [5] network programming system into our deployment, and have since used it several times to make code updates. Reprogramming our 20-node network takes approximately 30 minutes and is reliable (though in a few instances one node was not updated and had to be manually reprogrammed).

[1]http://sensorscope.epfl.ch

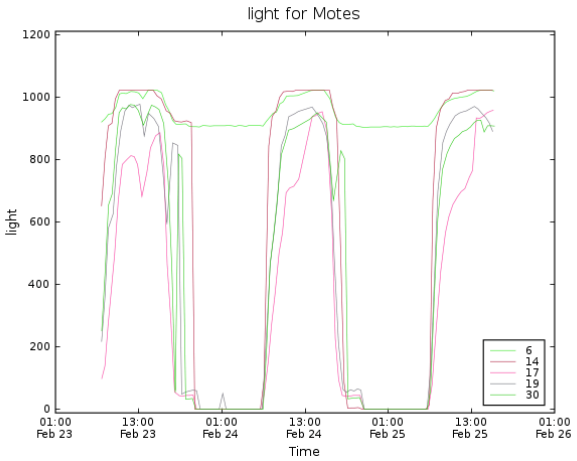

Figure 3: Sample output graph from the web interface. It shows the light sensor reading for 5 motes from Feb. 23 to Feb. 25, 2005. Note how at 21:00, the light at EPFL is automatically turned off and all readings, except one mote that is near a window with a nearby street light, are 0.

Our only significant departure from the standard TinyOS network stack is the addition of a multi-hop hybrid ARQ (MHARQ) layer between the network and link layers. With MHARQ, nodes buffer corrupt packets upon reception, and when two corrupt versions of a packet have been received, a decoding procedure attempts to recover the original packet from the corrupt copies. Unlike traditional forward error correction (FEC), MHARQ does not transmit redundant overhead on good links; and unlike adaptive coding techniques, it does not require costly channel probes to estimate the amount of redundancy required to achieve reliable communications. MHARQ also exploits the multi-node nature of a sensor network by enhancing multi-node interactions (multi-hop routing, multicast, or flooding) in a way that standard point-to-point FEC cannot, in addition to enhancing single-hop communications. This layer is also responsible for managing link-layer retransmissions. A detailed description of the MHARQ scheme is beyond the scope of this paper; for further details we refer to [1].

The application, besides periodically turning on and sampling sensors, is also responsible for parsing incoming broadcast messages and reforwarding them. Broadcast messages originate from the base-station, and either carry *query requests* or *configuration commands*. A total of 25 broadcast message types are currently defined. Queries are sent to retrieve different node configuration parameters as well as various networking-related monitoring information such as routing tables, neighbor tables, and network activity counters. Commands are sent to specify configuration parameters that may be applications-related (sensor sampling rate, which sensors to sample) or network related (max. number of retransmissions, turn on/off MHARQ, set low-power listening status, etc). A simple storage module saves configuration updates into persistent flash memory.

2.2 Server-Side

The server side of SensorScope builds upon several free software packages and libraries and glues them together as one system. A middle ware programmed in Java, makes the link

Setting	Oct.	Nov.	Dec.
Low Power Listening	4	2	4
RF Power (dBm)	-17	-14	-14
Sampling Rate (s)	120	120	120
Routing Data Rate (min)	5	5	5
Neighbor Table Rate (min)	60	60	15
Retransmission	0	0	5
MHARQ	no	no	yes
Deluge	no	no	yes

Table 1: Program parameters for the three data sets. *Low Power Listening*: mode in which the mote was. *RF Power*: the radio chips power setting. *Sampling Rate*: rate at which motes sent their sensor data. *Routing Data Rate*: rate at which routing data was sent to the base station. *Neighbor Table Rate*: rate at which neighbor tables were sent to the base station. *Retransmission*: how many times a mote tried to retransmit a packet if it was not successful. *MHARQ*: if multi-hop hybrid ARQ was enabled or not. *Deluge*: if Deluge was implemented (programming over the air)

from the sensor network to the database. This database stores sensor data, routing and neighbor tables as well as maintenance information. Additionally, it provides possibilities to send query requests and configuration commands to the motes.

The user interface is either a command line tool or a web interface programmed in PHP that uses Python CGI scripts to access the database and to generate sensor data graphs (see Figure 3 for an example). The sensor data can also be exported to Matlab files for a statistical analysis and signal processing purposes. Furthermore, the web interface visualizes routing trees and connectivity graphs (see Figure 2 for an example). Additionally, it offers a graphical interface to send queries and commands to the motes to set, for example, their radio power, sampling rate or which sensors to sample. The web interface has also facilities to help observe the network's status such that the developer can intervene if motes unexpectedly die, and is connected to a SMS messaging gateway via which it delivers a message when nodes become unresponsive.

3. PERFORMANCE

This section investigates the performance of the sensor network for three different datasets. Each dataset consists of two weeks of continuous sensor and routing table data. During these two weeks the motes were not moved and the code was not altered. Batteries were only changed, when a mote reached a voltage reading below a certain threshold. The three datasets will henceforth be called according to the month in which the data was collected: October, November, and December. See Table 1 for the configuration parameters of each set.

3.1 Network Performance

Figure 4 plots the average number of packets delivered vs. the motes depth in the routing tree. Each data point represents one mote's average number of delivered packets and average depth in the routing tree for the whole dataset. The lines correspond to the average over one hour intervals of routing tree depth and packets delivered for all motes.

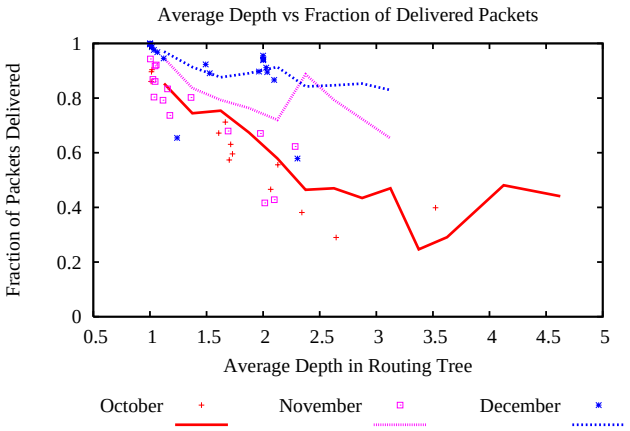

Figure 4: Average routing tree depth vs. packets delivered for each mote and dataset. Note that the data is very noisy. Therefore, the average does not decay smoothly with increasing routing tree depth. For more details, see Section 3.1.

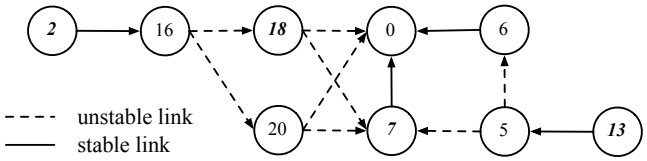

Figure 5: Network graph showing how motes can have a stable or unstable routing tree depth. See Section 3.1 for more details.

We can see that the difference in overall number of packets delivered per mote between the October and November dataset is small. The increased average fraction of packets delivered can be explained by the fact that in November the motes in the network were set to a higher RF Power, and therefore, the network had less motes with a routing tree depth greater than 2. In December, the network performed better. The major difference is the enabled MAC layer retransmission, i.e. motes now tried up to 5 times to deliver a packet to the next hop if they did not receive an ACK.

One can distinguish between motes with a stable routing tree depth, and motes with frequently changing depth. Figure 6 shows 4 different motes from the October dataset that illustrates this further. Mote 7 and 13 have a stable routing tree depth of 1 and 3, whereas mote 2 and 18 have a routing tree depth oscillating between $[1, 2]$ and $[3, 4]$ respectively. In the October dataset, one finds that 46% of the motes have a stable depth and 54% do not. This behavior can be explained by a motes parent link stability and who it can choose as parent. Figure 5 depicts the network topology for the 4 motes shown in Figure 6. The topology was reconstructed from the received parent information. Mote 0 was the sink. Mote 7 was near the sink, and therefore had a stable one hop link. Mote 13's parent was most of the time mote 5 which in turn had either mote 6 or 7 as its parent. Both of them are motes with a stable one hop link to the sink. On the other side, mote 2's parent was mote 16 which had as parent either mote 18 or 20. But both of them were motes with an unstable link to the sink and therefore chose sometimes an additional hop to deliver their messages.

Figure 7 illustrates the individual number of packets deliv-

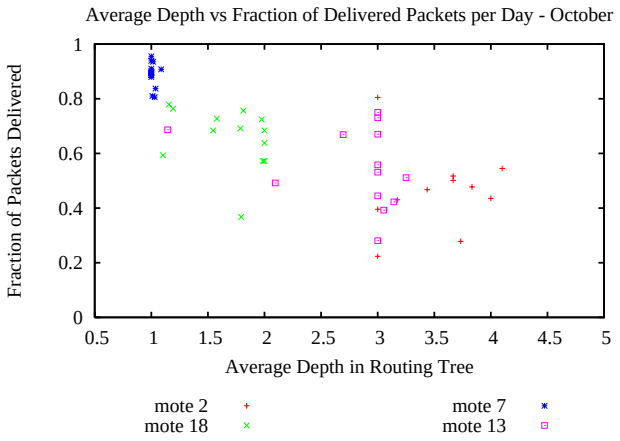

Figure 6: Average routing tree depth vs. packets delivered for 4 motes in the October dataset. Each point represents the average over one day. Note how mote 2 and 18 have an average depth between [1,2] and [3,4] whereas mote 7 and 13's depth is stable at 1 and 3 respectively.

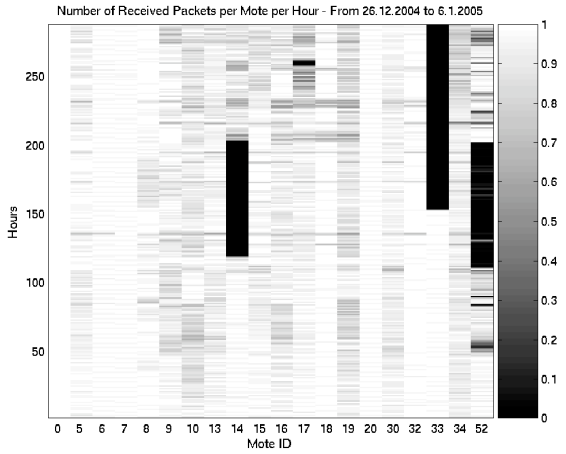

Figure 7: Fraction of packets delivered for each mote per hour for the December dataset. Each small bar represents the fraction of received sensor data packets at the sink (mote ID 0) per hour.

ered for each mote for the December dataset. One can see the difference in delivered packets between motes that are always one hop away from the sink (6, 7, 18, 20, 32, 33) and the other ones. The two black bars for mote 14 and 32 are due to battery failure. Node 52 had a persistently weak path to the sink.

3.2 Channel Utilization

SensorScope lies somewhere between wired testbed and remote deployments in terms of realism versus observability. In particular, nodes report network monitoring information with more detail and at higher rate than would be possible in a remote network where the difficulty of changing batteries requires keeping radio utilization at a strict minimum. This increased monitoring traffic could potentially cause congestion, and thus completely change network dynamics with respect to the conditions expected in a low-rate, low channel utilization deployment. It is therefore necessary to verify that even with the additional traffic, the network remains

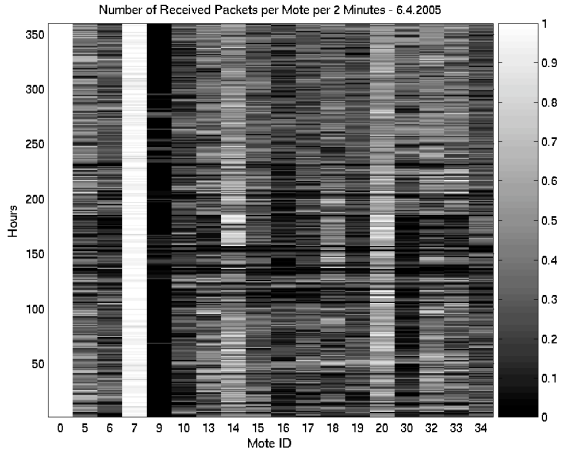

Figure 8: Fraction of packets delivered for each mote per 2 minute for a congested network. Sampling rate was set to 6.5 seconds and Low Power Listening to 4. See Section 3.2 for more details.

	November		December	
ID	packets/min	byte/min	packets/min	byte/min
5	7.2	2139.8	7.6	4048.3
14	9.0	2647.5	8.8	4711.7
16	5.2	1527.5	3.8	2039.4
20	12.1	3563.4	10.5	5619.5

Table 2: Total number of bytes transmitted for 4 motes. The data is the average for the two days 15th, 16th November and 28th, 29th December 2004

clearly in a non-congested regime.

As part of the reported network statistics, nodes send the total number of bytes transmitted over the air. We show these in Table 2 for a subset of 4 motes, for the November and December datasets. Since packet lengths are variable, the number of packets transmitted can only be approximately inferred; this approximation is however close to the expected number of packets transmitted according to the constants of Table 1. Note that there are large differences in transmission workload for different motes. This comes from the fact that the relay load varies depending on a mote's position in the routing tree. For example mote 20 frequently has two or more children in the routing tree, whereas mote 16 is nearly always a leaf node.

Under the conservative assumption that all nodes are within interference range of each other, we have a maximum network throughput equal to the radio rate of 19.2 kbps, or 144'000 bytes/min. Assuming an average transfer rate per mote of 2000 byte/min for November and 4000 byte/min for December, we would have a total network throughput of approximately 40'000 byte/min for November and 80'000 byte/min for the December dataset. This appears sufficient to establish that the network is not operating in a persistently congested regime.

To see how the network performs in a congested scenario, we ran a 24 hour experiment with a high sample rate of 6.5 seconds. All the other parameters were set to the same values as in the December dataset (Table 1). From this it follows that the estimated per node throughput without forwarding

messages is 7'150 byte/min. Under the assumption of 18 motes within interference range of each other, this amounts to a channel load of approximately 128'700 byte/min, which is lower than the theoretically possible 144'000 bytes/min.

Over the 24 hour period, only mote 7 performed well and delivered 96% of all the packets. Next was mote 20 with 54% and mote 14 with 48%. All the other motes have delivered less than or equal to 35% of all their packets. An explanation for these differences can be found in their distance and channel quality to the sink since mote 7 and 20 are closest to mote 0. All other motes are further away. Depicted in Figure 8 are the individual number of delivered packets for each mote for this dataset.

Congestion has also an effect on the transmitted bytes for each mote. Mote 20, which had most of the time mote 7 as its parent, transmitted on average 28'000 byte/min and mote 7 32'700 byte/min. In contrast, mote 14 transmitted 12'700 byte/min and mote 16 12'100 byte/min.

If we increase the congestion of the network by setting the sampling rate to 5 seconds, the delivery rate drops even further. Mote 7 still delivers 97% of its packets, mote 20 delivers 42% and mote 14 only 28%. All the others are below 20%. With our assumptions, the network would need at least 160'200 byte/min to deliver all packets. Mote 20 transmits on average 10'300 byte/min and mote 7 transmits 21'600 byte/min. In contrast, mote 14 transmits 33'500 byte/min and mote 16 37'800 byte/min, although both have a much smaller number of delivered packets. This means that collisions required them to retransmit a lot of messages before they were either sent to the next hop or dropped from the send queue.

3.3 Mote Lifetime

We analyzed mote lifetimes over the December dataset. As mentioned previously, the motes run at a Low Power Listening level of 4. This corresponds to a duty cycle of 6.1%, i.e., a maximum of 4.64 packets/sec. All our mica2dot motes are equipped with two AA Alkaline batteries, each with 1.5 Volt. If we assume that the voltage decreases linearly with the capacity of the batteries[2], we find an average power consumption of 13.1 mV/day (sample variance $s^2 = 11.0$) over the two weeks. This corresponds to an average mote lifetime of 61 days (assuming that motes die if they reach 2.2V). Further analysis of the dependence between power consumption and number of messages sent, including forwarded messages, shows that there is a significant positive correlation between these values.

4. CONCLUSION

We have presented the system architecture of SensorScope and gave an overview of the network performance for three time periods. We also showed that the network operates in a non-congested regime and what the performance is if it is congested. In total, we had 560'000 sensor data values, 94'000 routing informations and 192'000 neighbor table entries. The difficulty in analyzing this data lies in the fact that it is unreliably delivered, and as such we do not have fixed, periodic snapshots of network state. This is especially critical since the most interesting aspect of the performance is not when the network is delivering more than 90% of the data, but when there are problems, for example with interference or congestion.

Future plans for SensorScope include refining the user interface and to improve the maintenance tools. Another goal is, not surprisingly, to extend battery lifetimes. This would allow us to gather sensor data and network statistics over a longer period of time without interruptions. Finally, we intend to make the collected data more easily accessible to other scientists.

[2][2] shows that the linear assumption is good for voltages between 1.5 and 1.1 Volt.

5. REFERENCES

[1] H. Dubois-Ferriere, D. Estrin, and M. Vetterli. A multi-hop hybrid arq layer for sensor networks. In *under submission.*

[2] Energizer. Energizer en91 datasheet. http://data.energizer.com/PDFs/EN91.pdf.

[3] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Salt Lake City, Utah, May 2001.

[4] W. Hu, V. N. Tran, N. Bulusu, C. tung Chou, S. Jha, and A. Taylor. The design and evaluation of a hybrid sensor network for cane-toad monitoring. In *Proceedings of Information Processing in Sensor Networks (IPSN 2005/SPOTS 2005)*, Los Angeles, April 2005.

[5] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94. ACM Press, 2004.

[6] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of ACM Sensys*, Los Angeles, USA, April 2003.

[7] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, November 2004.

[8] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *2nd European Workshop on Wireless Sensor Networks*, Istanbul, January 2005.

[9] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, November 2004.

Session 2: Localization and location dependent services

Lost in Space
Or Positioning in Sensor Networks

Michael O'Dell
University of Zurich
Switzerland
odellm@access.unizh.ch

Regina O'Dell
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
bregina@tik.ee.ethz.ch

Mirjam Wattenhofer
Department of Computer Science
ETH Zurich, Switzerland
mirjam.wattenhofer@inf.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

ABSTRACT

We discuss the success chances of a real-world implementation of a positioning system for a wireless sensor network. While much research has been done in the area of node localization, the method of choice to verify the results has been in theory or by simulation. To realize the visions of future sensor networks, we take very basic sensor nodes (prototyping the smart dust idea) and implement a series of experiments to determine their suitability for use in positioning algorithms. Specifically, we look at the accuracy and effectiveness of direct distance measurements based on different signal strengths. We found that even though links are stable and symmetric over time, positioning in the real world is yet in its infant stage and that current theoretical models of sensor networks do not apply well in unspecialized hardware.

1. INTRODUCTION

The future of wireless sensor networks is often envisioned as a large collection of small, if not to say tiny, low-power devices. The challenges that come with this paradigm are twofold. On the one hand, engineers are trying to build hardware that conforms to this "smart dust" ideal. On the other hand, researchers need to construct models and algorithms that are suited to the very stringent requirements that these devices impose: low energy consumption, low storage and computational capabilities, and basic radio transmission hardware. With current state-of-the-art technology, such wireless sensor networks exhibit unstable links whose changes are often unpredictable.

One of the primary needs for a sensor node is to know its position so that the sensor data can be coupled to a location. Additionally, network coordinates allow for efficient routing known as geometric routing algorithms [9]. Then the problem known as *localization* (or positioning) is, given that some specially equipped nodes know their positions, for all the nodes in the network to determine their location based on the positions of the *seeds* (a.k.a. anchors) and possibly the network connectivity structure.

In this paper we focus on determining the potential of using *minimal hardware requirements* for the task of positioning in a wireless sensor network. The question is how well can a node localize itself if we only have small, low-power, low-storage devices at our disposal. The answer will allow engineers of such a network to determine whether to invest in more specialized and thus bulkier and more expensive hardware, or if the position error is tolerable in their applications.

There has been some work on investigating the physical characteristics of real-world sensor networks [5, 16]. Our findings corroberate their conclusions (as in [5]) in that the link characteristics are far from the theoretical models in use, such as the unit disk graph [4, 11] or the quasi unit disk graph [10]. However, these works are either too general in nature in that they survey the detailed link stability but not its effect on positioning ([5] looks at flooding, [16] at routing). Or they only consider two nodes at a time, whereas our findings in Section 3 indicate that there are big differences between the experimental results if only two nodes are involved versus an entire network.

There has been a significant amount of research done on the theoretical side of positioning and virtual coordinates algorithms, see [2, 3, 7, 11, 12, 14], to name a few. The underlying assumption in all of these is that the network can be represented as a static graph, usually even a unit disk graph. Physical wireless links, in contrast, are prone to instabilities. So even if the nodes are not moving, the neighborhood of a node might change rather dramatically and, more importantly, usually unpredictably, over a short period of time. This can be due to the sometimes drastic effects of intereference, scattering, or dampening of signals. If sensor networks are to be deployed in uncontrolled environments, then these effects simply cannot be ignored. We will further discuss these phenomena in Section 3 on the basis of the measurements we have taken.

Numerous of the above papers also have particular hardware requirements which assume fairly accurate measurements on the part of the nodes. Examples include measuring the time of flight (or time difference of arrival (TDoA) as in [8, 13]), the angle of arrival (AoA) [12], the received signal strength (RSS) [1]. Measuring the time of flight in reasonably-sized

ad hoc networks presupposes very accurate hardware which can detect differences in the nano second range: If we assume approximately the speed of light, the time it takes for a signal to cross the distance of a few meters amounts to some tens of nano seconds. In this paper, we have also opted to use the signal strength measured in terms of packet loss at different powers as a first indication of the distance between two nodes. However, instead of requiring a particular hardware component dedicated to the precise power measurement of incoming signals, we have even less stringent hardware requirements. This is described in more detail in Section 2.

Another line of research has been the development of hardware suited to the specific purpose of sensor localization. Papers in this area include Cricket [13], RADAR [1], or also [4, 15]. Some of these work only indoors (Cricket, RADAR), some only outdoors (GPS). Cricket has a separate ultrasonic component and RADAR was tested on a larger scale with more available computing power (laptops). With current or foreseeable technology, a node cannot support fairly sophisticated positioning hardware *in addition to* the sensor and actuators that carry out the intended purpose, all at the smallest scale. Thus, we wanted to investigate the potential of very limited hardware for positioning in sensor networks. We also do not impose any indoor/outdoor restrictions.

Our contributions are on one hand results on the stability, symmetry and distance relationship of the wireless radio links, and on the other hand we have observed that the gap in the measurements between two nodes in a lab and the nodes in a network is significant enough to render any localization attempts useless at this point. Put in another way, the distance-to-power correlation is strongly and unpredictably environment dependant.

2. HARDWARE PLATFORM

The hardware we have used for our experiments is the ESB/2 platform from the scatterweb project, now its own company [6]. The nodes are built from standard components, consisting of a chip with a 32kHz CPU, 2kB of RAM, and a low power consumption radio transceiver, along with numerous sensors and actuators such as infrared, temperature, vibration, microphone, beep, and LED.

In the current version available to us, the nodes can adjust their transmission power x from the application (for $1 \leq x \leq 100$ percent), but the transceiver is not able to directly measure the received power, only whether the signal is above a threshold. The way that the power is adjusted at the sender is via a potentiometer which controls the current to the transceiver. It has been brought to our attention that it is now possible to read out the received signal strength on the ESBs directly and this modification is part of future work in this area.

What we do instead is a "software version" of RSS by measuring the packet loss while varying the transmission power at the sender. In order to determine an approximation to the received signal strength, an anchor node writes its sending signal level into a packet, and the receiving node could then read out this value, take the minimum over all received packets, and thus determine the lowest signal level at which it can still "hear" the anchor. While this way of measuring the transmission power is certainly not the most precise way, it fulfills the natural assumption that greater perceived received signal strength means that the sender needed to use more power to reach the receiving node, thus the receiver is farther away. The exact correlation needs to be determined, but the important point we want to verify is that the same input level on the sender should reach the same distance given similar conditions.

A critical issue with these nodes is the susceptability of the radio signal strength to various outside influences. Two nodes might be as close as a couple of centimeters, but placed near a wall or close to some underground cable, or as far as a few meters, and both times the best received signal strength will be the same. Here, we build on the preliminary measurements of the transmission range as a function of the signal strength in various settings from project website [6].

3. RESULTS

We will now discuss our measurements, their results and the motivations that led from one experiment to another. In the following we use the terms sensor node and node interchangeably.

3.1 In the Lab

As discussed in Section 1 we want to implement popular positioning heuristics on real sensor nodes. Towards this goal we first need to obtain some data on the correlation between the power level received and the distance of the nodes without obstacles.

Our experimental setup is the following: In the corridor of our lab, an anchor node transmits 100 packets at each power level and a receiving node placed at a specified distance measures the number of packets it receives. This experiment is repeated for inter-node distances ranging from 1cm to 120cm, as in a first step we want to explore the accuracy of the distance-to-power relation on a small scale. The minimum power level at which a packet is received at a given distance is plotted in Figure 1. While the data points do not lie on the theoretically assumed parabola, they are almost monotone with slight deviations of at most three levels and the curve exhibits a certain regularity.

Most applications for wireless networks will, however, not be satisfied with a single packet arriving with some low probability. Therefore we furthermore test the link quality if we require that at least x percent of the packets arrives at the listening node. The results for $x = 90$ are shown in Figure 2. The graph looks similar for all other values of x (down to 50).

The deviation of the data set to the best-fit curve is in both experiments not negligible (about 10 units in the latter case), as can be seen in Figure 1 and 2. However, the data set is still well-behaved in the sense that a curve is discernable and this curve is almost monotone.

The conclusion that can be drawn from these experiments is that in a controlled environment with a clear line of sight, the distance-to-power function with a specified stability can be approximated to a certain extend by a monotonely in-

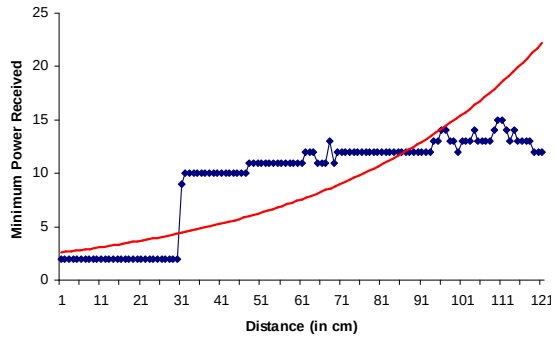

Figure 1: The minimum power level which was received at the given distance.

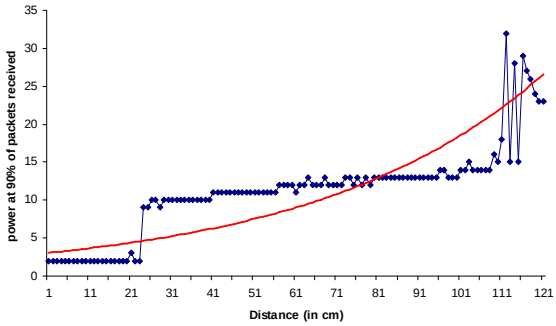

Figure 2: The lowest power level at which at least 90% of the packets were received.

creasing function thus supporting theoretical assumptions. In such a setting, a possible localization scheme could measure the packet loss for different power levels from anchor nodes (which know their position) and then use the inverse of the distance-to-level point map in Figure 2 to obtain a distance estimate.

3.2 In a Room

Any localization algorithm in the plane needs (approximate) distance measurements from at least three non-colinear anchors. Therefore, the next step in our experiments is to expose a single node in a room to several anchors and test the obtained measurements for their usability.

The experimental setup is similar to the one before. We place four anchor nodes on the corner of a rectangle in a room. A test node is then placed within that rectangle. An anchor sends out a packet at each power level from 1 up to 100, then the next anchor does the same, and so on, in a round robin fashion. Each time, the test node reports which packets it receives. Figure 3 shows how often a given power level is received by the test node from Anchor 1 over the course of the experiment. The results for the other anchors are similar and thus omitted.

This already goes to show that the link between the test

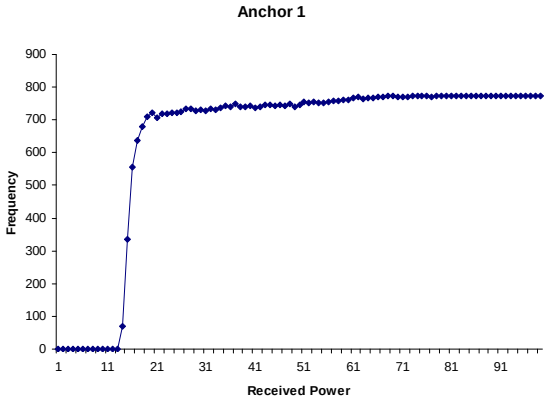

Figure 3: The number of times a given power level was received. Anchor 1 was 1.39 meters from the test node.

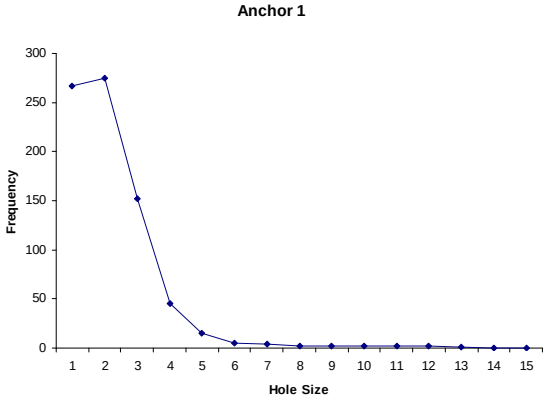

Figure 4: The size of the "holes" in the received power level progression and how often they occured.

node and the anchors is reasonably stable over time. As we already see from the earlier experiments in the lab, the *minimum* power level received, perhaps averaged over time, gives a good indication for the distance to the sending node. This is advantageous since it saves memory compared to storing all received power levels. To further strengthen this hypothesis, we also examine the size of the "holes" in the received power levels for each iteration. In other words, how big is the gap between one received power level and the next and how often does it occur. For example, if the first heard packet has level 15 and the next one heard level 18, then this results in a hole of size 3. Figure 4 shows that there are only small holes in most cases. Meaning that the minimum power level is a good estimate if the requirements are not too stringent.

In a further step, we add obstacles to our room by placing various everyday objects in the area of the rectangle. The result is that the general behavior of the link quality does not appear to be affected, seen in Figure 5. The curve has the same "shape" as before (less data points being the reason for the height difference). Astonishingly, the peak as

resulting from the experiment with obstacles is shifted to the left compared with the non-obstacles experiment. Yet, this result only completes the picture of the unpredictability of real-world sensor node behavior.

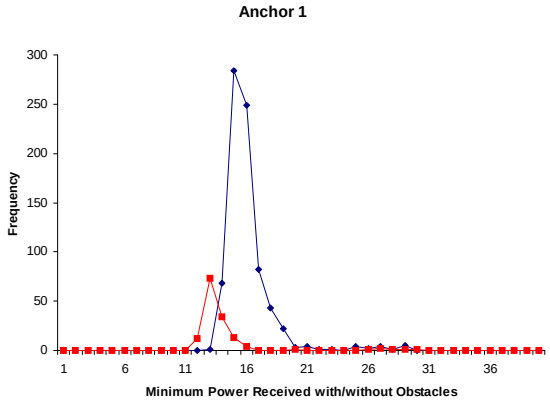

Figure 5: Number of times that a given power level was the minimum frequency received. The higher peak (blue) is from the setup without obstacles, the other one (red) with obstacles. The reason for the lower peak is that the experiment was run for less time due to external contraints.

Concluding this subsection, we can say that in an environment with several stable anchors, each sending out packets one at a time to avoid collisions, the links appear to be stable over time and exhibit a sharp peak.

3.3 Network

The results of the previous sections suggest that distance measurements based on radio power levels can be used if the accuracy constraints are not too tight. This leads us to set up a positioning experiment with four anchors at the corners of a rectangle (a table) and several nodes on the inside of the rectangle. We use a spring-based algorithm in which each node iteratively computes its new position as a function of its neighbors' positions. This approach stems from the graph drawing context and was first adapted to positioning in [14]. Since the heuristics proposed in [14] are far too power and resource consuming, we implement a simplified version which meets our purposes. For the power-to-distance conversion, the data gathered in the experiments above is used. Surprisingly, between most of the computed positions and the corresponding actual positions there is seemingly no correlation. The errors are in the order of the magnitude of the sensor field. A closer examination reveals that already the powers received do not correlate with the distances at all in the sense that a node close by needed significantly more power to communicate than some nodes far apart, even without any obstacles in the room. This finding is closer examined in the next experimental setup.

We perform the following experiment, the result of which can be seen in Figure 6. We take nine nodes placed arbitrarily in a large room. Iteratively each node sends out a series of packets, starting from level 1 to 100. In each iteration all non-sending nodes record the minimum power level

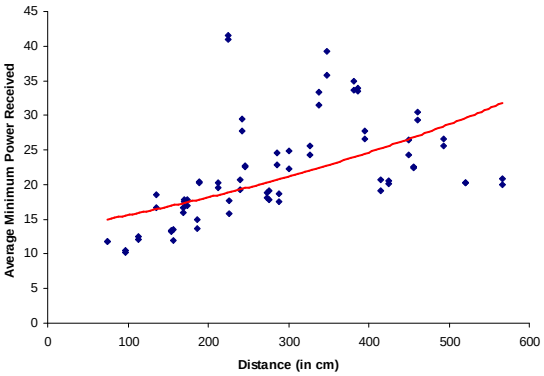

Figure 6: Measured power levels at various distances. The x-axis is distance (in cm), the y-axis is average minimum power level. A point is the measurement of a node to some other node.

received from the sending node in that iteration. As can be seen, the graph looks very different from the one in the first phase of the experiment, Figure 1. Whereas in the first phase of the experiments a curve connecting the data points is discernable, the data points measured in the current experimental setup are much more scattered. Observe also that the scale of the y-axis in both figures is different and that the deviation of the data set to the best-fit parabola in this experiments is about 30 units.

The conclusion to be drawn here is that in a two-node setting with a constant environment, the power expectedly increases approximately quadratically with distance. In a network, however, there are several nodes, unpredictable environmental conditions such as people walking around, and different positions of walls, cables, other computers and the like. So even without the effect of node transmission interference, which we excluded in this setup, the power-to-distance correlation becomes utterly useless. The data points are scattered across the graph. While *two nodes* in the network can have a quadratically increasing (theoretical) distance-to-power function, this function between *different pairs* of nodes is not necessarily related in a predictable manner.

On the positive side, this experiment allows us to conclude on the symmetry of the links between two nodes. To that end, we compare the minimum power levels between two nodes in the network. In Figure 7 the number of occurrences a power level was the lowest one received for an arbitrary pair of nodes is plotted. The peaks are sharp and coincide.

We furthermore compute the average minimum power level received for each node to all other nodes and take the difference between the node pairs. This value is rounded to the nearest integer and Figure 8 shows the number of times a difference occurs among the $\binom{9}{2} = 36$ node pairs.

4. DISCUSSION AND FUTURE WORK

In this paper, we presented an experimental study of the link quality in real-world sensor networks. If one expects the sensor network to be in place for an extended period, then we

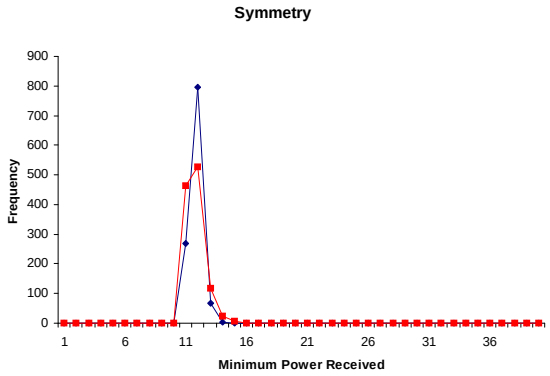

Figure 7: Number of times that a given power level was the minimum frequency received. The blue curve is from node 75 to node 65 and the red curve the other way around.

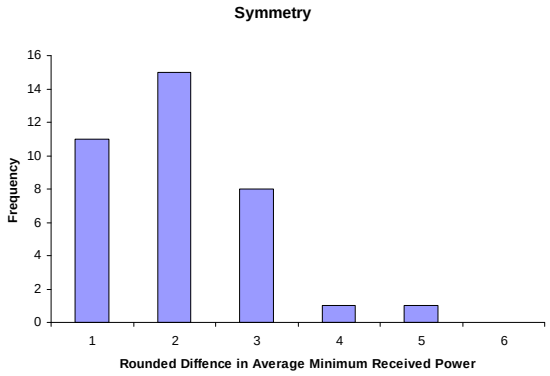

Figure 8: Rounded differences between average minimum received power.

do not necessarily require an unvarying connection between two nodes, but that the link characteristics should be *steady over time*. The minimum power at which a node receives a packet from a sender node was shown to be stable, sharp and symmetric over time in all experimental setups. Whereas this power level is a good indication for the distance between nodes if there are only two nodes in a static experimental setup, this correlation does not apply to all nodes pairs in a larger arbitrary network. Our experiments illustrate that in larger networks the power levels measured in lab conditions are not correlated with distances in the real world. Thus drastically new models are required for sensor networks if theoretical constructs are ever to be applied with a realistic chance.

Despite the pessimistic results, the work in this paper opens up a number of interesting directions meriting further investigation. First of all, these tests should be performed on different hardware to see whether this is a general problem. Second, with the new version of the ESBs, we can perform actual received signal strength measurements which would refine the results observed in this paper. Third, it would be interesting to investigate whether there is a qualitative difference between short and long range measurements.

We would like to thank the anonymous reviewers for their helpful suggestions.

5. REFERENCES

[1] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *Proc. of Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, 2000.

[2] R. Bischoff and R. Wattenhofer. Analyzing Connectivity-based, Multi-hop Ad hoc Positioning. In *Proc. of the IEEE Intl. Conf. on Pervasive Computing and Communications (PerCom)*, 2004.

[3] P. Biswas and Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. In *Proc. of Intl. Symp. on Information Processing in Sensor Networks (IPSN)*, 2004.

[4] E. Dijk, K. van Berkel, R. Aarts, and E. van Loenen. A 3-D Indoor Positioning Method using a Single Compact Base Station. In *Proc. of the IEEE Intl. Conf. on Pervasive Computing and Communications (PerCom)*, 2004.

[5] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. Technical report, UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013, 2002.

[6] ScatterWeb GmbH. `http://www.scatterweb.net`.

[7] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large Scale Sensor Networks. In *Proc. of Mobile Computing and Networking (MobiCom)*, 2003.

[8] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning Systems: Theory and Practice.* Springer, 5th edition, 2001.

[9] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proc. of Symp. on Principles of Distributed Computing (PODC)*, 2003.

[10] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-Hoc Networks Beyond Unit Disk Graphs. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 2003.

[11] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer. Virtual Coordinates for Ad hoc and Sensor Networks. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 2004.

[12] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS). In *Proc. of IEEE Global Communications (GLOBECOM)*, 2001.

[13] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. of Mobile Computing and Networking (MobiCom)*, 2000.

[14] A. Rao, C. Papadimitriou, S. Ratnasamy, S. Shenker, and I. Stoica. Geographic Routing without Location Information. In *Proc. of Mobile Computing and Networking (MobiCom)*, 2003.

[15] K. Römer. The Lighthouse Location System for Smart Dust. In *Proc. of ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.

[16] D. Son, B. Krishnamachari, and J. Heidemann. Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks. In *Proc. of the First IEEE Conference on Sensor and Adhoc Communication and Networks*, 2004.

Improving Location Accuracy by Combining WLAN Positioning and Sensor Technology

Paul Hii
Monash University
900 Dandenong Road
Melbourne, Australia
+61 3 9903 2556
Paul.Hii@infotech.monash.edu

Arkady Zaslavsky
Monash University
900 Dandenong Road
Melbourne, Australia
+61 3 9903 2479
Arkady.Zaslavsky@infotech.monash.edu

ABSTRACT
This paper presents a heterogeneous location-aware system that combines two technologies so that their properties can be aggregated to produce more accurate location data. This paper proposes and demonstrates a location system that uses a sensor mounted on a robot to determine its location acoustically and complement the data with location from the existing Ekahau Positioning Engine (EPE). It is shown that the acoustic localization is capable of verifying errors that exist in the location information provided by the EPE. The heterogeneous system resulted in location information that gives 100% accuracy in spatial space information which the EPE was unable to provide all the time. The sensor fusion by combining the acoustic location system with the EPE helped improve the accuracy of location sensing by verifying the location information and increases location data availability.

Categories and Subject Descriptors
I.2.9 [**Artificial Intelligence**]: Robotics - a*pplications, sensors*

General Terms
Measurement, Performance, Experimentation, Verification.

Keywords
Location-awareness, wireless sensor networks.

1. INTRODUCTION
Application pervasiveness can be achieved by providing intelligence in the form of contextual information. If computers can have the same perception as human beings about their vicinity, applications can re-configure their behavior based on the model and state of the environment. With location information, applications can adapt their interface and features in response to the current geological space.

There are already commercial deployments of location-aware applications especially for use outdoors. An example is the Global Positioning System (GPS) used in navigating cars on land and ships in the oceans [5]. Depending on the receiver used, the accuracy of GPS may be about 30 meters. Another technology for outdoor positioning is the telecommunication infrastructure that measures and computes line-of-sight and time-of-flight of radio signals for locating mobile phones. These technologies are not suitable for use in the indoor environment.

Indoor location information has to have high precision and accuracy to at least a couple of meters or less. Most pervasive applications require at least the knowledge of the spatial space the client is in or the relative location of the clients with respect to certain landmarks. Other location-aware applications, such as robot localization, require numerical coordinate location information. The busy indoor environment requires specialized sensors to discover location information accurately and robustly. A single location system is required to provide location information on the tracked client but it will not always give accurate location information and this information cannot be verified. To overcome the single technology location system's drawback, a heterogeneous location system is proposed so that location information from two location-aware systems can be compared to improve accuracy. The use of multiple location systems with different technologies to form hierarchical and overlapping levels of sensing is known as sensor fusion [8]. Multiple location systems can aggregate the properties that are not available in a single technology system.

Section 2 will detail some of the existing positioning technologies. Section 3 will discuss the benefits of having a heterogeneous system. Section 4 will discuss our proposed solution of integrating our existing system with an acoustic location system. Performance of the existing location system will be compared with the proposed solution in Section 5. This paper will then conclude in Section 6.

2. LOCATION-AWARE SYSTEMS
A number of wireless physical medias have been used in devising means for obtaining indoor location information. These include the use of infra-red in Active Badge [13] and PARCTAB [12], radio frequency in RADAR [1], SpotOn [7] and FLARE [3] and ultrasound in Active Bat [6] and Cricket [11]. Infra-red (IR) is a line-of-sight technology and it suffers adversely under the influence of external radiation. Radio frequency (RF) produces location information that is not accurate enough for use in location-aware system because of the variations in the received signal strength due to multi-path propagation especially in the highly cluttered and varying indoor environment as observed in the Ekahau Positioning Engine [4]. RF propagates at a very high velocity that makes precision timing very difficult. Ultrasound and acoustic sound were later adopted as the preferred medias.

2.1 Ekahau Positioning Engine

The Ekahau Positioning Engine (EPE) is completely a software-only solution that utilizes the existing wireless LAN infrastructure [4]. The EPE uses its predictive capabilities to determine location based on the process of site calibration that builds a signal strength model of the environment.

Site calibration involves taking signal samples at different known locations to construct a signal strength model based on signal strengths from various radio channels on the floor. The model is stored in the Positioning Engine. Ekahau clients will retrieve received signal strength values and return the values to the engine for location calculations. The dynamic indoor environment ensures that the radio signal propagation is susceptible to multi-path effect thus making the generated signal strength model inconsistent with the actual signal propagation in the environment. Site calibration creates the empirical model of the signal strength in the environment. Therefore modifying the existing wireless LAN layout after calibration will result in the positioning model to be inconsistent with the new environment. Signal strength at different areas will fluctuate in time due to movement of objects or new installations. This will require re-calibration of the EPE to re-produce a new positioning model.

The EPE exposes location information in the form of (x, y, floor) and logical area. The accuracy of the system is about one meter if there is a minimum of seven access points in sight but in a long term the accuracy decreases due to fluctuations in signal strength. The error in EPE can range from 1.5 meters when no one is in the room to 3.0 meters when the room is half filled with people. This is one of the problems noted in RADAR [1].

Applications that obtain location information from the EPE do not know whether the location information is accurate as there is no form of location verification. EPE only knows the signal strength model of the environment but unaware of walls and obstacles. Therefore if a device is near a wall, the EPE may see the device on the other side of the wall. The location given by EPE has greater inaccuracy when a device is near a wall then when it is in the middle of the room. This could be due to radio signal distortion and multi-path effects caused by the wall.

The EPE has been applied as the source of location information for a mobile robot device. With the EPE, the robot is aware of its location and all the locations of other tracked wireless devices. Since the EPE is not precise, the problem propagates to the robot and makes the robot behave indiscriminately. In order to verify the location information, another means of localization is required. All of the existing location-aware systems are homogeneous systems as each focuses on a single technology. The single system design means that the problems they face are inherent to the system and the technology used.

3. HETEROGENEOUS LOCATION SYSTEM

Two very different and independent location systems that use different technologies should be able to complement and compensate each other for more accuracy and better location-awareness. Hightower and Borriello said that "The more independent the techniques, the more effectively they can be combined" [8]. The two sources of location information can validate each other to extrapolate accurate location information.

A research in combining various location systems to form a heterogeneous location system by [2] lacks precise numerical location information. [2] used Session Initiation Protocol (SIP) that had three means of providing location information. They are location profile provided by Bluetooth devices, IR/RF badges that send unique identifiers to access points of known location and an extended DHCP with information about jack locations. The downside of this system is the lack of coordinate information.

Research in robot localization also showed that a single localization method is insufficient for high accuracy localization [10]. As a result [10] proposed using sensor fusion consisting of ultrasonic tracking and image target analysis. Image analysis for localization is not popular but it was also used in the TRIP project by [9] for their sentient computing project. This sensor fusion is an example how combining two independent systems can improve the location accuracy.

This project will combine the EPE with acoustic localization that is supported by a wireless sensor network. The proposed location system to complement the Ekahau Positioning Engine (EPE) uses a wireless sensor network that is designed to apply acoustic sound to approximate distances for localization. Unlike the EPE that tracks all wireless devices from a centralized server, this design in contrast is a decentralized design like the Cricket Location System [11] whereby any devices can track only themselves by performing the location calculations themselves. The acoustic localization will provide location information to complement with the EPE to form the Multi Location Test Verifier so that location coordinates from the EPE can be verified and validated to be accurate enough for use by the robot or any mobile device. The sensor fusion by combining two location-aware systems to produce accurate location information is a heterogeneous design that ensures redundancy of location information to be available all the time and anywhere as long as one system is working.

4. ACOUSTIC LOCALIZATION

The robot will have a Mica2 as a base mote (Figure 3) that broadcasts a radio signal to other fixed Mica2 motes whose locations are known. Concurrently, the robot's base mote also produces a 4 kHz tone. Since radio signals propagate faster than sound, the fixed listening motes will receive the radio signals first. The listening mote will receive the radio signal that instructs the mote to start counting until it detects the 4 kHz tone. Upon hearing the tone, the mote will send back a radio message to the robot the time difference of arrival of the radio and tone signals. The time difference of arrival can then be used to infer distance between the robot and the fixed mote. This is illustrated in Figure 1.

If the base mote knows its distances relative to three listening motes whose locations are known, the location coordinate of the base mote can be calculated by hyperbolic tri-lateration (Figure 2). Each listening mote has a locus of possible positions for the base mote described by $(x-x_0)^2 + (y-y_0)^2 = r^2$ where (x_0, y_0) is the known location of the listening mote and r is the distance between the listening mote and the base mote. The intersection of three loci will give the location of the base mote. Therefore this localization design cannot calculate location if less than three

distances to listening motes are available. Thus it is very crucial that the listening motes must be installed in an unobtrusive location to increase the chance of the tone being heard by the listening motes.

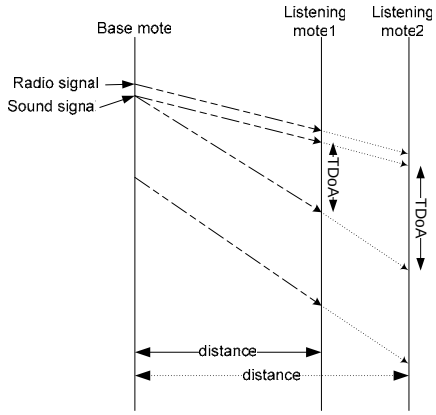

Figure 1. Inferring distance from Time Difference of Arrival

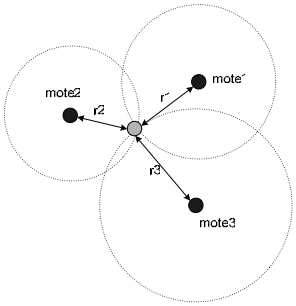

Figure 2. Hyperbolic tri-lateration

Acoustic localization is more applicable to use on a robot than on human beings or stationary items. It constantly emits an audible tone which can be an irritation to the wearer of the Mica2 mote. In the robot's case, it only needs to beep when the robot is on the move which is usual as a safety precaution.

The acoustic localization can work as a separate entity to complement the EPE to determine the most accurate and sensible location information. Furthermore the 4 kHz tone does not penetrate walls. This solves the problem if the EPE should provide the wrong spatial space information.

4.1 Hardware

The wireless sensor network is made up of nodes which are small Mica2 motes (Figure 3). A base mote on a robot will listen for time of flight information from static motes positioned at known locations. The Mica2 motes have a microphone and sounder on each of them. The sounder is use to generate a 4 kHz tone and the microphone has a tone detector that turns the analog microphone into a digital high or low level output when a 4 kHz tone is detected. These two components will be used in conjunction with radio communication to determine distances.

Figure 3. Mica2 mote on ER1[1] robot

4.2 Design Issues

Acoustic was chosen over radio signal strength as the media to infer distances because of several reasons. Radio signal travels at about 300,000 kilometres per second while sound travels at about 340.29 meters per second which is almost a million times slower than radio signals. Accurate time of flight measurement of radio signal on a small scale is very difficult. Furthermore, a one microsecond error will result in 300 meters error but for sound, a one millisecond error will result in only 34.03 centimetres error. It is also very difficult to detect whether the radio signal is suffering from multi-path effects since the received signal might not be original and may have been a delayed one. In the case of sound, the first instant of received sound is always the original since echoes are significantly slower and have lower amplitudes.

There must be no body or obstacles that can absorb or distort sound waves that can adversely influence the tone signal propagation. The listening motes must be installed in high enough places so that there is nothing that can hinder it from listening for incoming tone signals. The spacing of listening motes must about 2 meters apart so that at least three motes can hear the incoming tone. The base mote must always have the sounder facing upwards so that sound energy is propagated upwards to the listening motes.

Using acoustics to determine distances is not a robust solution as the tone detector is vulnerable to detect all kinds of 4 kHz noises. Loud noises could also mask out the 4 kHz tone. Therefore to ensure that the mote is actually listening for the correct tone from the base mote, a low pass filter is programmed into the mote for limited robustness.

5. RESULTS DISCUSSIONS AND COMPARISON OF PERFORMANCES

An application called Multi Location Test Verifier (MLTV) has been developed to concurrently obtain location information from the Ekahau Positioning Engine (EPE) and the acoustic localization system. MLTV is a visualization client that maps the location information onto the floor plan of the room where the

[1] http://www.evolution.com

experiment was carried out to give a visual perspective of the locations as seen by the two different location systems.

5.1 Results Discussions

The first Multi Location Test Verifier results on the acoustic localization and EPE Positioning showed that the acoustic localization is a promising location system as shown in Figure 4.

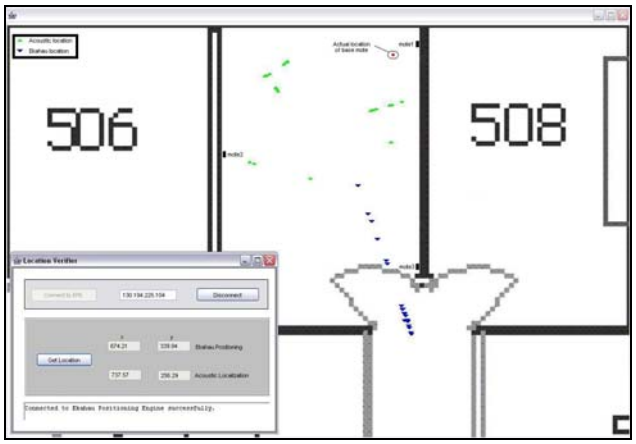

Figure 4. Tracking view when tracked device is near mote1

The first test had the mobile device (base mote) situated nearer to mote1. This location was chosen because there is greater error from EPE when the device is nearer to walls. Most of the acoustically determined locations are between mote1 and mote2; and between mote1 and mote3. This indicates that the tracked device is somewhere nearer to mote1 which is true. Of all the samples taken, 80% indicate that the location is nearer to mote1 and 100% were located within the room. On the other hand, locations provided by the EPE had greater errors. EPE located the device to be nearer to mote3 which is on the other end of the room. Of all the samples taken, a 45% of the samples indicate that the tracked device is in the room. This shows that the acoustically determined locations are more accurate than the EPE. This is useful in validating useful and accurate location information from EPE when a device wants to know its location.

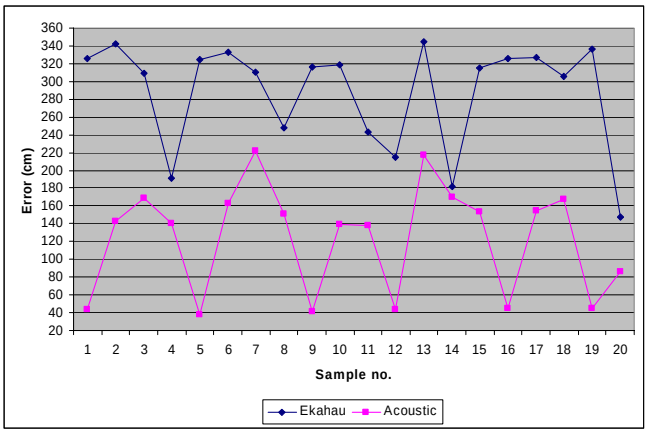

Figure 5. Location errors for acoustic localization and Ekahau Positioning Engine

The amount of errors for each sample is shown in Figure 5. Each sample was taken at about ten seconds one after the other. The acoustic localization method has an error that ranged from 38.10cm to 221.85cm with an average error of 123.40cm for this first test. The EPE produced an error that range from 146.97cm to 343.31cm with an average error of 287.98cm.

Figure 5 also shows that the characteristics of the EPE and acoustic localization have an almost similar pattern in their performance. This could be due to the varying environment factors such as temperature, speed and direction of air flow and so on. Since the locations from the EPE and acoustic localization were taken concurrently, the two location systems are influenced by the same environmental variables. In this first test the acoustic localization system has better accuracy than the EPE as the EPE suffers from greater error when the tracked device is near a wall.

A second test was carried out at the other end of the room by situating the tracked device near mote3 (Figure 4). At this location, the acoustic localization successfully determined the location to be around mote3 at 90% of the time. Although the error distance for the EPE is not very big, it provided wrong spatial space information. The locations EPE provided indicated that the tracked device is outside of the room in which it is actually in. This is misleading especially for mobile application such as robot localization. This is an example where another location system is useful in verifying the accuracy of the EPE. Thus the acoustic localization system can verify the EPE's location information since it has better accuracy.

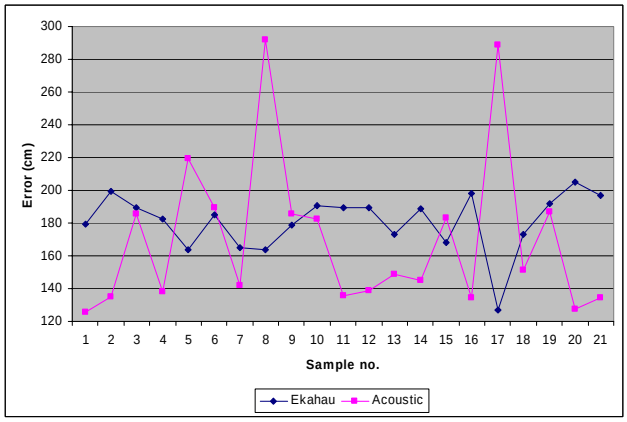

Figure 6. Location errors for acoustic localization and Ekahau Positioning Engine

Figure 6 shows the distance error from the actual location of the tracked device. The acoustic localization method has an error that range from 125.59cm to 291.58cm with an average error of 189.97cm for this first test. The EPE produced an error that range from 163.50cm to 205.03cm with an average error of 180.94cm. Errors in the acoustically determined locations have bigger variation because it is near the door where wind draft can be flowing in and out of the room. The error margin in the previous location is smaller because it was previously located in the room's corner where atmospheric condition is more stable than near the door.

The accuracy of the acoustic location system is about 2 meters error. The localization rate is about 90% with at least three nodes that can audibly hear the tone beep.

5.2 EPE vs Acoustic Localisation

The designs of the two systems are different. The EPE has a centralized design where the Positioning Engine computes and provides location information of all tracked clients. The acoustic localization system is decentralized as the client is tracking itself thus providing the location information to itself only.

The decentralized design of the acoustic localization also ensures privacy of the tracked client. Location information is provided directly to the tracked client only. Thus not everyone can see the tracked client's location unless the client decides to publicize its location details. On the other hand, the EPE is a centralized server which knows location of all tracked clients. Anyone who can access the EPE can list all tracked devices and locations. The information may be exploited and privacy may be breached.

Time taken to retrieve location coordinate is significantly faster in acoustic localization than the EPE. The EPE has a delay of about 2-5 seconds when tracking devices. The acoustic location system is capable of obtaining location in just about 1 second which is significantly faster than EPE. This is because the EPE suffers from network delays but the Mica2 base mote only needs to communicate with nearby listening motes. Tracked clients need to relay information of received signal strength to the EPE through the wireless infrastructure. The rate at which tracked client devices take signal strength samples is also another factor that determines how fast the EPE can provide the location.

The advantage for the EPE is that it uses existing 802.11 infrastructures and no additional proprietary hardware. It is a Java-based software only technology that is easy to deploy and cheaper than most other location-aware systems. The acoustic localization system requires a wireless sensor network to communicate and listen for the 4 kHz tone. The cost of setting up the using the Mica2 motes is proportional to the scale of coverage. Listening motes need to be installed in many places. This is expensive in cost and management.

The drawback to the acoustic localization system is that it cannot return a location if less than three receivers return the time difference of arrival information of the radio and tone signals. The EPE can still provide location information even if there are less than three access points but the error increases with less access points.

Overall the acoustic location system was harder to use as the motes need to be properly placed and oriented and the room has to be free of noise. On the other hand, the EPE was easier to use.

6. CONCLUSION AND FUTURE WORK

This paper demonstrates improving location-awareness by using WLAN positioning and sensor technology. The experimental results are encouraging as location error is small. This suggests that location-aware services can be enhanced with technologies complementing each other. The existing Ekahau Positioning Engine (EPE) and the acoustic location system complement and compensate each other in verifying and choosing the more accurate location information. The Multi Location Test Verifier (MLTV) showed that the acoustic location system can discover incorrect location information provided by the EPE. This shows the benefit of having a heterogeneous location-aware system. One might suggest that one very accurate location system is sufficient but the homogeneous design inhibits the system to a certain capability such as limited coverage and objects tracked. Therefore, the aggregation of the properties of different location systems can give a location system that has a larger tracking area and tracks more things with better accuracy.

Future work on the location-aware sensor networks should focus on improving the accuracy of the time measurements. A larger scale sensor network should be setup to prove the ability of the location system to track mobile clients. The acoustic location system can also be extended to track clients quietly and robustly by using ultrasound that will make the system robust.

7. REFERENCES

[1] Bahl, P. and Padmanabhan, V. N. (2000), 'RADAR: An In-Building RF-based User Location and Tracking System', *IEEE INFOCOM*, Mar 2000, Tel-Aviv, Israel, 775-784.

[2] Berger, S., Schulzrinne, J., Sidiroglou, S., and Wu, X. (2003), 'Ubiquitous Computing Using SIP', *13th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, June 2003, ACM Press, 82-89.

[3] Christ, R. and Lavigne, R. (2000), 'Radio Frequency-based Personnel Location Systems', *IEEE 34th Annual Int. Carnahan Conf. on Security Technology*, Oct 2000, Ottawa, Canada, 141-150.

[4] Ekahau (2002), "Ekahau Positioning Engine 2.0: 802.11-based Wireless LAN positioning system", An Ekahau Technology Document (http://www.ekahau.com), Nov 2002.

[5] Getting, I.A. (1993), "Perspective/Navigation – The Global Positioning System", IEEE Spectrum, vol. 30, no. 12, Dec 1993, 36-38.

[6] Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P. (1999), 'The Anatomy of a Context-Aware Application', *5th Annual ACM/IEEE Int. Conference on Mobile Computing and Networking, MOBICOM'99*, Seattle USA, Aug 1999, 59-68.

[7] Hightower, J., Borriello, G., Want, R. (2000), "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength", UW CSE Technical Report #2000-02-02.

[8] Hightower, J. and Borriello, G. (2001), "Location Systems for Ubiquitous Computing", *IEEE Computer*, vol. 34, no. 8, Aug 2001, 57-66.

[9] Ipina, D.L., Mendoca, P.R.S. and Hopper, A. (2002), "TRIP: A Low-Cost Vision-Based Location System for Ubiquitous Computing", *Personal and Ubiquitous Computing*, vol. 6, no. 3, May 2002, 206-219.

[10] Jarvis, R.A. (1992), 'Autonomous Robot Localization by Sensor Fusion', *IEEE Int. Workshop on Emerging Technologies and Factory Automation*, Aug 1992, 446-450.

[11] Priyantha, N.B., Chakraborty, A. and Balakrishnan, H. (2000), 'The Cricket Location-Support System', *6th Int. Conf. on Mobile Computing and Networking*, Boston, USA, Aug 2000, 32-43.

[12] Schilit, B.N., Adams, N., Gold, R., Tso, M.M. and Want, R. (1993), 'The PARCTAB Mobile Computing System', *4th Workshop on Workstation Operating Systems*, Oct 1993, Napa, USA, 34-39.

[13] Want, R., Hopper, A., Falcao, V. and Gibbons, J. (1992), 'The Active Badge Location System', *ACM Transactions on Information Systems*, vol. 10, no. 1, Jan 1992, 91-102.

Using Wireless Sensors as Selection Devices for a Multimedia Guidebook Scenario

Montserrat Ros†, Matthew D'Souza*, Michael Chan*, Konstanty Białkowski*, Adam Postuła*, Neil Bergmann*, Andras Toth‡

* School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD, Australia 4072

{dsouza, mchan, ksb, adam, n.bergmann}@itee.uq.edu.au

† School of Mathematics, Statistics and Computer Science, University of New England, Armidale, NSW, Australia 2351

ros@mcs.une.edu.au

‡ Ericsson AB, Torshamnsgatan 23 164 80 Stockholm Sweden

andras.toth@ericsson.com

ABSTRACT

This paper describes the implementation of a wireless sensor network for a multimedia guidebook scenario incorporating a pointer-sensor system for the selection of locality-aware information. An Information Point Station Network (IPSN) was developed and consists of several Information Point Stations (IPSs) placed at locations of significance, with access to information items on a centralized server. In the multimedia guidebook scenario, a user selects a particular information item to view, either by way of a menu system appearing on their mobile computing device (MCD) or a more intuitive pointer-sensor system as described in this paper. Laser sensors are placed next to prominent or relevant objects, and can be either directly connected to an IPS, or function as isolated sensor nodes. The pointer is attached to the MCD by way of a serial port and the user points the pointer at the laser sensor next to the object for which they require information. The information is then sent to the MCD via Bluetooth. The implementation was found to be successful and was tested with multiple users accessing information items from a given IPS as well as multiple IPSs attached to the centralized server. Still, there is further work to be done on the isolated sensor nodes.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Wireless communication*.

General Terms

Design, Experimentation

Keywords

Wireless Sensors, Bluetooth, Multimedia Guidebook.

1. INTRODUCTION

Wireless Sensor Networks have many uses. This paper introduces a wireless sensor network that was used for the purpose of selection for a multimedia guidebook application. The Multimedia Guidebooks scenario allows the user to access specific information related to their immediate surroundings and can be used in museums and other tourist facilities. This paper also presents the development of an interactive pointer and wireless isolated sensor node architecture used for the intuitive selection of information items sent to the guidebook.

The multimedia guidebook operates by allowing the user to select information to be viewed by aiming a pointer device at a laser sensor located near an object of interest, upon which information is sought. The requested information is then 'beamed' to the user's mobile computing device (MCD) via Bluetooth for the user to view. Bluetooth was selected as the medium of choice due to its prevalence on most modern mobile computing devices such as Personal Digital Assistants and mobile (cell) phones. This fact, coupled with the nature of Bluetooth communications lacking the requirement for line-of-sight meant users could retrieve location-aware information at the "locality" level. The user can also select the information by using a menu on their mobile computing device.

An Information Point Station Network (IPSN) was developed for use in the Kelvin Grove Urban Village located in Brisbane, Australia [2]. The network consists of information point stations (IPS) placed at specific locations of interest. A point of interest can be a monument, a building or any article of significance. Each information point station has access to media information relevant to its location and consists of a Bluetooth transceiver and system of laser sensors. The Bluetooth transceiver is used to transfer the requested information to the user's mobile computing device. The laser sensors are used to detect when the user is selecting information by using the interactive pointer device. As well as laser sensors connected to information point stations, the guidebook scenario provides for isolated sensor nodes. The isolated node is similar to the IPS-connected laser sensors in that it contains a laser sensor. However, the isolated sensor node also contains an infrared transceiver that it uses to communicate with the user's MCD.

This paper is organized into 6 sections. Section 2 presents a review of related work. Section 3 describes a typical user scenario. Section 4 discusses the implementation of the pointer and sensor multimedia guidebook network infrastructure. Future areas of investigation are discussed in section 5 and conclusions are drawn in section 6.

Figure 1 – a) Menu Browser on the IPAQ – b) Pointer prototype – c) Sensor prototype

2. RELATED WORK

Multimedia Guidebooks wireless networks are found in museums and other tourist attractions. There are two main types of multimedia guidebook wireless networks. One type involves the information being transferred to the guidebook on request. The second type already has the information stored on the mobile computing device. Examples are the Exploratorium [7] and the Cyberguide guidebooks [3]. These guidebook wireless networks have been developed for older PDA platforms. No guidebooks have yet been developed for the latest smart mobile phones using interactive pointers. Existing guidebook wireless networks tend to use older short-range wireless protocols for information transfer such as infrared communications. This is due to the widespread usage of infrared transceivers on older PDA platforms. Infrared is not as widely integrated into mobile computing devices as Bluetooth [4] has become. Infrared has been surpassed by Bluetooth in many applications because it does not require line of sight, supports ad-hoc networking and has more robust data communications.

The Exploratorium guidebook provides the user with information about exhibits in a museum. The Exploratorium guidebook deploys Radio Frequency ID (RFID) nodes, 802.11b wireless LAN and HP Jornada PDAs. When a user with a PDA comes within range of an exhibit that has a RFID node, the node's ID is sent to the PDA via infrared. Information about the exhibit is requested by the PDA and the result is returned via the 802.11b LAN. The Cyberguide is a map guide that allows the user to find their location on the map of a venue. It uses customized infrared sensors to determine the user's location and PDAs. The infrared sensors are at known locations on the map of the venue. When the user comes with in range of an infrared sensor, their location can be displayed on their PDA's map.

Interactive pointer devices are more commonly known as remote controls and are used in everyday life to control devices such as television sets. Pointers can be classified in terms on the communication medium used. Most commonly used pointers use infrared. There are projects to develop pointers that use laser light. One such project described in [8] developed a laser pointer linked to a PDA so that it can control devices remotely. One of the reasons for using laser rather then infrared is that it provides the user with a better visual mechanism to point at a device as suggested by [5]. So far pointers have not been used with guidebook wireless networks or other similar wireless network applications that require the user to select information.

3. WIRELESS SENSOR NETWORK SCENARIO

The scenario of a Multimedia Guidebook gives the user the opportunity to experience locality-aware information on demand in various multimedia formats. The user can choose to interact with the Information Point Station Network (IPSN) and request their chosen information item in a number of ways. Other than by selecting from a menu (Figure 1a), the user can also intuitively point for the information they desire, without the need of connecting to a particular Information Point Station (IPS) and searching through a menu. As the user roams around the Kelvin Grove Urban Village site, he/she may choose to use the pointer device attached to their mobile computing device (MCD) (Figure 1b) when approaching an object of interest with a laser sensor or isolated node (Figure 1c).

In the case of an IPS-attached laser sensor, only one-way communication between the pointer and the laser sensor is required. When the pointer is connected to the serial port of the MCD and the user points to the laser sensor, the pointer transmits the MCD's identification information (currently the Bluetooth address of the device). Upon reception of this information, the laser sensor transmits a command message consisting of both the user's identification and the sensor's own identification to the IPS. This is so that the IPS can send the requested item for the corresponding object of significance to the MCD.

In the case of an isolated sensor node, two-way communication is required between the pointer and the sensor. Hence, these isolated nodes require an infrared transceiver to facilitate the return communication. When the pointer is connected to the serial port of the MCD and the user points to the laser sensor, the pointer transmits the MCD's identification information as per an IPS-connected sensor. However, upon reception of this information, the isolated sensor node transmits the command message back to the MCD using infrared. Once equipped with the sensor's identification, the MCD software automatically connects to a nearby IPS to request for the information item associated with that sensor node's identification details. Although the isolated sensor node requires line-of-sight to communicate its identification details via infrared, this is not a problem, as the user is already pointing the pointer directly at the sensor.

The user's experience of an information item once it is downloaded via Bluetooth is the same regardless of how the user selected it – whether it be from an onscreen menu, or by selecting a sensor or node with the laser pointer. The goal of offering locality-aware information is achieved by ensuring that retrievable information is the information suited to the object of interest for which the sensor is a selector.

4. WIRELESS SENSOR NETWORK IMPLEMENTATION

The Multimedia Guidebook wireless network consists of information point stations (IPS) placed at specific locations of interest. Each IPS forms part of an Information Point Station Network (IPSN). The IPSN is shown in Figure 2. The IPSN consists of a central server, information point stations, isolated sensor nodes, mobile computing devices and pointer devices. The server controls the IPSN. It communicates to each information point station (IPS) via Ethernet. Each IPS contains a wireless Bluetooth transceiver and laser sensors. The Bluetooth transceiver is used to communicate to the mobile computing device. The mobile computing device (MCD) can be a Personal Digital Assistant or a mobile phone. The MCD is used by the operator to select and view information from the IPS using either the menu or the interactive pointer by aiming it at a laser sensor.

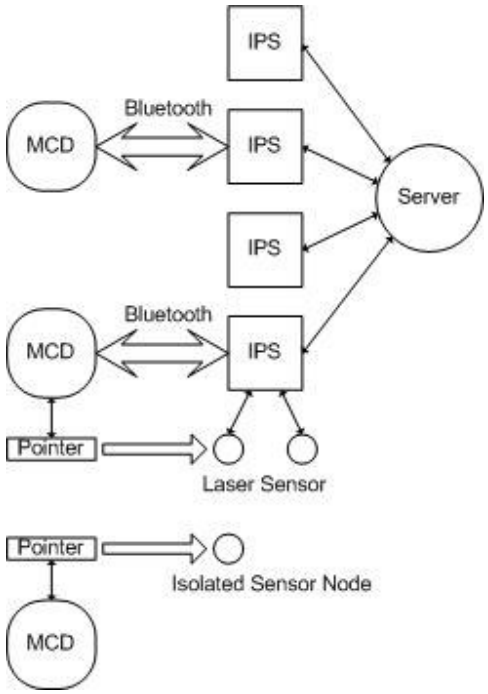

Figure 2 – IP Station Network.
(MCD = Mobile Computing Device, IPS = IP Station)

Sensor nodes can also be used to by the user to select information. When a sensor node is selected by the interactive pointer, it responds with the sensor node's identification which is transmitted by infrared to the pointer. The pointer then relays the sensor node's identification to the MCD which will connect to a nearby IPS to request for the information item associated with that sensor node's identification details.

4.1 Communications

Figure 3 and Figure 4 show the different communication protocol channels used for IPS-attached sensors and isolated sensor nodes, respectively. For the Bluetooth connection between the IPS and the MCD, the Bluetooth serial port (SPP) and Object Exchange File transfer (OBEX-FTP) profiles are used. The Bluetooth serial port profile is used as a control channel where a specialized control protocol was designed and implemented to request information from the IPS [6]. The OBEX-FTP profile is used to transfer information items in the form of files from the IPS to the MCD.

The IPS communicates with the server via a standard Ethernet connection and with attached laser sensor/s via conventional RS232. As shown in Figure 3, the pointer communicates to the IPS-attached sensor using the laser optical channel and the IPS must be connected to the laser sensor for this to occur. For an isolated sensor node, the pointer communicates with the node using the laser optical channel and the node responds to the pointer using the infrared optical channel. Figure 4 shows these communication channels.

4.2 Network Infrastructure: Server and Information Point Stations

The backbone of the IPSN infrastructure consists of the centralized server and the collection of Information Point Stations. The main functions of the server are to maintain the information items database, monitor each IPS and control the user identifier database. The information items database contains all information items associated with each IPS as well as individual sensors and sensor nodes. The identifier database contains the user Identifiers (UID) and preferences of all registered users. The server can monitor the status of the IPSN which includes displaying statistics such as the number of active users or the number of requests for a particular item. The server consists of a Linux computer with an Ethernet hub. Communication to each IPS is done using the TCP/IP networking protocol.

The function of the IPS is to provide a Bluetooth access point to the IPSN. The IPS will authenticate the user and supply the requested information items. Statistics such as the number of active users or the number of requests are recorded and sent to the server. The current implementation of the IPS is a Linux computer with a wireless Bluetooth USB. The open source Linux Bluez Bluetooth stack[1] is used to facilitate the Bluetooth

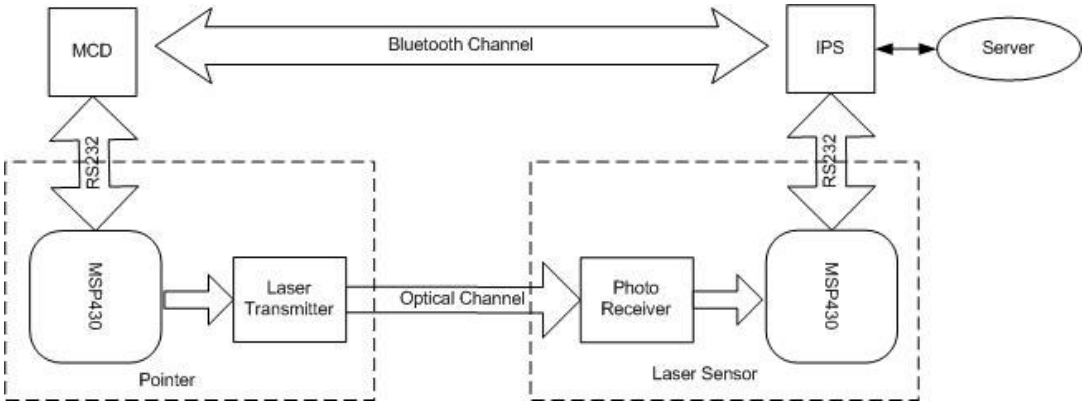

Figure 3 - Communication channels of IPSN with IPS, Laser Sensor, Pointer and MCD

connectivity to the mobile computing device. Using the Bluez Bluetooth stack was ideal for this project because it was easily customized to suit the project requirements.

4.3 Network Infrastructure: Pointer and Sensor/Node Implementation

The purpose of the interactive pointer device is to request data from an IPS. When the pointer is activated, it continually transmits the user identifier of the MCD, as shown in Figure 3. The pointer transmitter circuit is realized with a laser diode and associated driving circuit. A laser diode was chosen for its exceptional range, and directionality. Because binary signaling is used, the power output of the laser transmitter doesn't need close regulation, which simplifies the design and reduces power consumption of the driver circuit. The MSP430 [12] micro controller from Texas Instruments encodes the data stream using Manchester encoding, and manages communication with the MCD via RS232. This micro-controller was chosen for its ultra-low power consumption.

4.3.1 IPS-attached Laser Sensors

The laser sensor receives the modulated data stream via two OPT101 [11] monolithic photo-diode receiver circuits. This integrated circuit consumes minimal power, and includes an internal amplifier, leading to a simple receiver design. Two receiver circuits are employed to increase the effective viewing angle of the sensor. The MSP430 is used here to demodulate the waveform, and to forward requests containing the MCD's identification and the sensor's own identification to the IPS via RS232.

4.3.2 Isolated Sensor Nodes

The wireless sensor node is a standalone entity. It consists of the laser detector and an infrared transceiver. This can be seen in Figure 4. The laser detector design used is the same as that described in section 4.3.1. The infrared transceiver is used to convey identification information about the node to the user's MCD via the pointer device. The pointer device also contains an infrared transceiver.

4.4 Mobile Computing Devices

In the scenario of a guidebook application, the mobile computing device (MCD) functions primarily as an item requester. The guidebook software was implemented on a HP IPAQ 5550 Personal Digital Assistant running the Pocket PC 2003 operating system; and a SonyEricsson P910i SmartPhone [9] running the

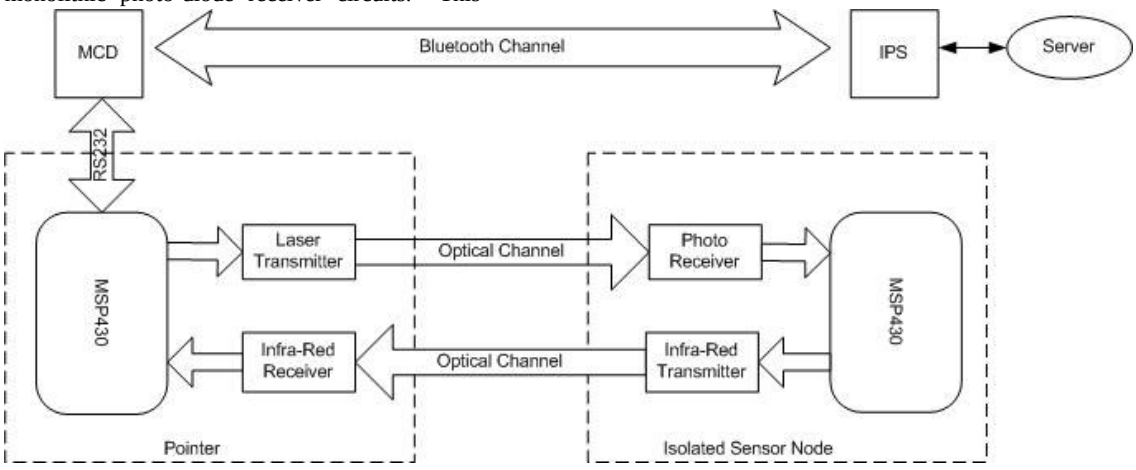

Figure 4 - Communication channels of IPS, Isolated Sensor Node, Pointer and MCD

Symbian operating system with Java 2 Microedition (J2ME) compatibility [10]. The software on the HP IPAQ was created using the C# language from Microsoft Visual Studio .Net for Pocket PC 2003.

The IPAQ software was integrated with the pointer and IPS-attached sensor system. This was done by physically connecting the pointer to the MCD using an available RS232 serial port and sending the device's Bluetooth address to the pointer. This is the user identification that the pointer then uses to send to the sensor for item requests.

5. FUTURE WORK

Although functional prototypes were created of all components of the Multimedia Guidebook wireless network, some areas are highlighted for future work. The information point stations will be converted to an embedded processor platform solution that runs the uClinux operating system. Another area of future work is the further development of the isolated sensor nodes. The functionality of the isolated sensor node will also be expanded to include other sensors. The sensor node will also be redesigned to be low powered and to use remote powering techniques such as Radio Frequency powering. This will allow the sensor node to be powered or charged without a physical connection, thus enabling it to be placed in unreachable positions.

6. CONCLUSION

A Multimedia Guidebook wireless network system was created to allow users to access specific information related to their immediate surroundings using their mobile computing devices. The Multimedia Guidebook was designed to provide information about the Kelvin Grove Urban Village in Brisbane Australia. Users can request to view information by using either an interactive pointer device or by software menu selection. The Multimedia Guidebook wireless network consists of information point stations and isolated sensor nodes placed at locations of significance. The information point stations consist of a Bluetooth transceiver and a series of laser sensors. The Bluetooth transceiver is used to transfer the requested information to the user's mobile computing device. The isolated sensor node consists of a laser sensor and an infrared transceiver. The infrared transceiver is used transmit the sensor nodes identifier to the interactive pointer device. The user can use the interactive pointer to select information items to view by aiming it at a laser sensor on either the information point station or the isolated sensor node. Future areas of investigation include creating an embedded platform for the information point station and investigating remote powering mechanisms for the isolated sensor node.

7. ACKNOWLEDGEMENTS

This work is supported by the ACID CRC Project and is also a part of the EU Project RUNES.

8. REFERENCES

[1] "Bluetooth Z - Official Linux Bluetooth Protocol Stack", http://www.bluez.org

[2] "Kelvin Grove Urban Village", http://www.kgurbanvillage.com.au/

[3] G. D. Abowd, C. G. Atkeson, et al., "Cyberguide: a mobile context-aware tour guide," *Wirel. Netw.*, vol. 3, pp. 421-433, 1997.

[4] Bluetooth SIG, "Bluetooth Specification Documents", https://www.bluetooth.org/spec

[5] D. Cavens, F. Vogt, S. Fels, and M. Meitner, "Interacting with the big screen: pointers to ponder," in *CHI '02 extended abstracts on Human factors in computing systems*. Minneapolis, Minnesota, USA: ACM Press, 2002, pp. 678-679.

[6] M. D'Souza, A. Postula, N. Bergmann, and M. Ros, "A Bluetooth Based Protocol for Multimedia Guidebooks on Mobile Computing Devices (*submitted*)", *2nd International Conference on E-Business and Telecommunication Networks*, Reading, United Kingdom, 2005.

[7] S. Hsi, "The Electronic Guidebook: A study of user experiences using mobile web content in a museum setting", *IEEE International Workshop on Wireless and Mobile Technologies in Education*, 2002.

[8] M. Ringwald, "Spontaneous Interaction with Everyday Devices Using a PDA", *Workshop on Supporting Spontaneous Interaction in Ubiquitous Computing Settings at Ubicomp 2002*, Gothenburg, , Sweden, 2002.

[9] Sony Ericsson, "Developers Guidelines: Java J2ME for P900 and P910 series mobile phones", http://developer.sonyericsson.com/site/global/docstools/java/p_java.jsp

[10] Sun MicroSystems, "Java 2 Platform, Micro Edition", http://java.sun.com/j2me/

[11] Texas Instruments, "Monolithic Photodiode and Single-Supply Transimpedance Amplifier", http://focus.ti.com/lit/ds/symlink/opt101.pdf

[12] Texas Instruments, "MSP430 Ultra-Low-Power MCUs", http://focus.ti.com/lit/ml/slab034h/slab034h.pdf

Session 3: Software development for sensor nodes

Timber as an RTOS for Small Embedded Devices

Martin Kero*, Per Lindgren, Johan Nordlander
Luleå University of Technology
Department of Computer Science and Electrical Engineering
EISLAB
SE-97187 Luleå, Sweden
www.csee.ltu.se

ABSTRACT

Software development for small, real-time and resource constrained, embedded systems is becoming increasingly complex. To be able to guarantee robustness and reliability, the underlying infrastructure should not be based upon ad hoc solutions. In this paper we identify three key features of a minimalistic Real-Time Operating System (RTOS), and presents the run-time system of Timber, a reactive deadline-driven programming language. We scrutinize the functionalities of the run-time system in the light of real-time requirements, and emphasize the importance of integrating an adequate notion of time, both semantically in the programming interface as well as part of the run-time system.

1. BACKGROUND

Software development for small, real-time and resource constrained, embedded systems is becoming increasingly complex. In order to guarantee robustness and reliability, the underlying infrastructure should not be based upon ad hoc solutions. For this reason real-time operating system (RTOS) features for small embedded systems have gained recent interest.

We have identified a set of desired key features. A minimalistic RTOS should at least:

- supply sufficient infrastructure for reactive concurrent programming,
- preserve state integrity, and
- realize real-time constraints.

In addition, it is beneficial if the RTOS and its programming interface provides the ability of formal reasoning about system properties, which would be useful towards safe and minimal system dimensioning. It is also desirable if this can be accomplished without limiting the expressive power of the programming interface.

RTOS's in general are too heavy weighted for small embedded devices, but a few proposals have been presented. Among them are systems like Contiki[10], PicoOS[3], FreeRTOS[2], Nucleus RTOS[1], and TinyOS[12, 11, 5]. All of these systems have their unique characteristics, but we will, in short, only present the last one. TinyOS is a minimal operating system suitable for the smallest embedded devices. It is based upon a component structure and a reactive event-based concurrency model. Its core footprint is only about 500 bytes. Robustness is partly achieved by a static analyzer for data race detection. The major problem that still remains unsolved in TinyOS, as well as in all the others, is the realization of real-time constraints in run-time. To be able to semantically guarantee some degree of robustness, TinyOS has restrictions in terms of expressive power. For instance, dynamic storage allocation is not allowed.

In this paper, we will present the run-time system of Timber [8, 9], a reactive deadline-driven language for embedded programming. The language it self will only be presented briefly and we will focus on the run-time system features. Readers interested in learning more about the language should read [8, 9] or visit [4]. Space does not allow a detailed side-by-side comparison of features in this paper, although such an evaluation is forthcoming. We will show that Timber offers a systematic approach to deal with real-time issues in embedded programming, unique in that the language semantics is fully reflected in the run-time system - in fact the run-time system and the application are one! This allows the executable to be fully tailored to the problem at hand, which is hard (or even impossible) to achieve under the tradional paradigm that separates the application from the operating system.

2. TIMBER - A SHORT INTRODUCTION

Timber, *TIme - eMBEdded - Reactive*, is a reactive, real-time, concurrent, object-oriented, functional programming language. It is based upon O'Haskell which in turn is an extension to Haskell [16, 15]. The development of the language is a joint effort by Luleå University of Technology, Chalmers University of Technology, and Oregon Health and Science University.

In brief, the language is based upon concurrently executing reactive objects [17]. The inter-object communication is message-based by means of synchronous and asynchronous message sends. A message send is equivalent to invoking a method of the recipient object.

Even though Timber is a general purpose language [9], it is primarily designed to target embedded systems, and we will discuss the aspects of the language in the context of embedded programming.

*e-mail: Martin.Kero@csee.ltu.se

```
1   sonar (port,alarm) =
2      template
3         t := baseline
4         ping = before (50*us) action
5            port.write(beepOn)
6            t := baseline
7            after (2*ms) stop
8            after (1*s) ping
9         stop = action
10           port.write(beepOff)
11        echo = before (5*ms) action
12           distance = k*(baseline-t)
13           if (distance < limit) then
14              alarm.on
15        return{
16           sonar = echo
17           start = ping
18        }
19  main regs =
20     template
21        s <- sonar ((regs!0xac00) a)
22        a <- alarm (regs!0xa3f0)
23        return {
24           reset = s.start
25           irqvector = [
26              (sonarIRQ, s.sonar),
27              (buttonIRQ, a.off)
28              ]
29        }
```

Figure 1: Example Timber program, A *Sonar*

2.1 Records and Objects

Besides primitive data types such as integers, floating point numbers, etc., Timber includes user-defined records and primitive types to support object-orientation. Records can either be used to define immutable data or to describe interfaces to objects.

Timber objects basically consists of two parts, an internal state and a communication interface. An object is instantiated by a template construct, which in turn defines the initial state of the object and its communication interface. The template construct can be seen as a module, offering an input interface and requiring an output interface, when instantiated into an object.

The primitive object-oriented types are `Action`, `Request`, and `Template`, which all are subtypes of `Cmd`. The meaning of the `Action` and `Request` types are asynchronous and synchronous message sends, respectively. These actions and requests are collectively called methods. `Template` is the type defining the template command from which objects are created.

A Timber program has to include a specific main template. The communication interface of this template is system dependent. For embedded devices the input interface usually contains a reset method and bindings from interrupts to actions. We will discuss this interface more thoroughly later on. The output interface is the environment in which the Timber program will operate. In the context of embedded devices, it shall at least provide methods to read from and write to ports.

At system startup, the main template command will be executed, creating an instance of the object main and then executing the reset method. The system will supply the main object with its environment.

2.2 Methods

A method is invoked by a message send command, either an asynchronous action or a synchronous request. A Timber program running on an embedded device can be seen as a set of concurrent objects, all awaiting external stimuli initially caused by interrupts. A method can basically do three things, it can update the state of the object, create new objects, and invoke methods of other objects. After an external stimulus, the chain of reactions will eventually fade out and the system will return to the state of waiting for new stimuli. We will refer to the time when the whole system is inactive and passive as the *idle state*, or *idle time*.

Each message in Timber has a corresponding baseline (earliest release time) and deadline attached to it. By default, a message inherits *baseline* and *deadline* from its sender but for asynchronous messages both can be adjusted by `after` and `before` constructs.

2.3 Objects as concurrent reactive processes

Objects in Timber has its own execution context, or thread of control. Inter-object communication is achieved by message-passing. Only one method within an object can be active at a time, and the object state is only accessible through its methods. This results in mutual exclusion of state mutations, usually referred to as state integrity. Furthermore, a method cannot block the object thread indefinitely which leads to a controllable responsiveness of each object.

The input interface of the main template is, as mentioned earlier, a reset method and an interrupt vector. The interrupt vector is a vector of tuples, connecting interrupt numbers to actions. This is how the environment will have the ability to trigger the reactive Timber program.

3. THE RUN-TIME SYSTEM OF TIMBER

The functionalities of the run-time system is directly reflected by the semantics of the language. The key features that needs to be facilitated in terms of functionalities of the run-time system are as follows:

Scheduling: The fundamental functionality of the run-time system to achieve concurrency between Timber objects, with scheduling based on the baselines and deadlines of their methods.

Message-passing: Supplying sufficient infrastructure for the inter-object communication.

Threading: Facilitating the unique execution contexts for Timber objects.

Time: Ability to supply sufficient time information to make baselines and deadlines meaningful.

Interrupt handling: Functionality for receiving and distributing interrupts throughout the system.

Environment interface: Implementation of the interface to the environment.

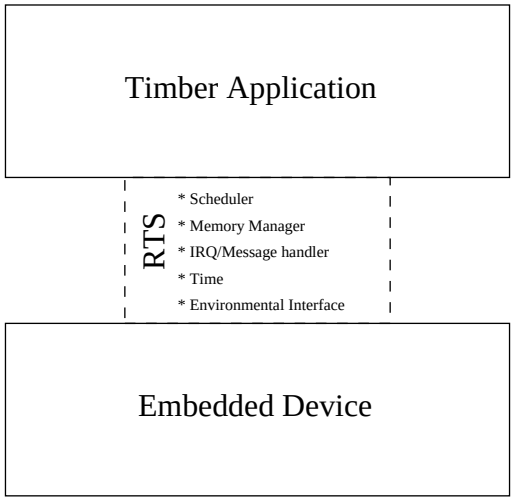

Figure 2: Timber Run-Time System overview

Automatic memory management: Timber does not rely on explicit allocations and deallocations of dynamic data and needs an automatic memory manager to serve with garbage collection.

The semantics of Timber actually does not imply a specific scheduling algorithm. It rather states the following: *Every method invoked by a message send has to be finished within the specified time-line (between its baseline and its deadline).*

The language also requires two levels of scheduling, one for messages (method invocations) within an object, and one between objects. The intra-object scheduling is non-preemptive due to state integrity, and priority inversion is solved by priority inheritance. The inter-object scheduling is preemptive, though, and realized in the run-time system by strict EDF, where the current deadline of an object is equivalent to the deadline of its most urgent message.

The message-passing mechanism in the run-time system is facilitated by message queues. Each object holds a queue of messages sorted by EDF. In addition to the local queues of each object, the run-time system also holds a queue of timed messages. A message send will insert the message into the queue of the recipient object. The corresponding method will be executed when the object has that message first in queue and is scheduled to run. Messages with a baseline ahead in time will be stored in a global queue of timed messages. This queue is sorted by earliest baseline first.

To accomplish concurrency between objects, each object needs to have its own execution context. At least they need a reference to their current execution point in the code. We will further base the context on a non-shared stack environment, adding the current stack-pointer to each context. The ability to use a shared stack environment is under investigation, mostly based on the work by Baker presented in [6, 7]. The context switching and storage is accomplished by means of non-local goto, storing the code- and stack-pointers in jump-buffers attached to each object.

A notion of time is essential for the run-time system in accomplishing correct scheduling. Baselines and deadlines are naturally expressed relatively to the actual occurrence of a message send. A straight-forward way to supply the necessary scheduling information is to calculate an absolute time from a global baseline for all message sends. This will greatly simplify the comparisons needed for scheduling, only dealing with notions of when a method has to be finished and making the time when a message send occurred necessary only locally and temporarily.

The main task of the interrupt handling mechanism is to serve as an interface between the interrupts and the message-passing mechanism. This is solved by a generic interrupt handler for hardware interrupts and a timer interrupt handler. Both of these handlers translate the interrupts into messages and post them as any other message sent by the application. This makes the handling of interrupts transparent to the rest of the system, which treats them as any other message.

The run-time system will allow context-switches to occur at three occasions. First of all, a context-switch may occur after a method is finished. This means that when the method finishes, the object will return the control back to the scheduler, which in turn will do the eventual context switch. The second case is after an interrupt has occurred and the corresponding message is sent. Both the first and the second case may cause a context-switch, but will not necessarily result in that. The third and final case will always cause a context-switch, which occurs on a synchronous request. To be able to receive the requested value, the calling object has to let the recipient object execute the method, and thus a forced context-switch will occur.

The environment interface includes the low-level implementation of time, hardware initializations of interrupts, and environment methods used by the main template. When a Timber program is in its idle state (all objects are inactive) the device is put into proper sleep-mode to lower the power-consumption. The low-level implementation of achieving this is also included in the environment interface.

The semantics of Timber is highly dependent on the ability to allocate memory storage dynamically. Furthermore, as mentioned, the language does not include any explicit allocation/deallocation commands. It is thus crucial to include a garbage collector in the run-time system.

A prototype of a reference counting garbage collector has been implemented for the run-time system of Timber [14] and currently a copying collector is under development. The work has so far shown a set of collector characteristics but due to space limitations they are only addressed in short. A thorough description of the collector is forthcoming.
1 The memory manager will reclaim garbage memory when the system is in its idle state.
2 The time needed for bookkeeping during execution of any method of the program is kept constant and small.
3 The time it takes to perform a whole garbage collection cycle will at worst be proportional to the maximum amount of simultaneously live memory.
4 The amount of contiguous free heap storage has to be

at least the maximum amount of simultaneously live memory. This however should not be misunderstood as a need for twice as much memory than a system based on explicit allocations/deallocations. It is a rather trivial fact that, in a multi threaded system, avoidance of deallocating semantically live data will most of the times result in a lot of semantically dead data to be kept alive. This is almost impossible to measure but if such a measurement would have been accomplished, it would not be surprising if it showed a factor much greater than two between the amount of data kept alive and the semantically live data for a typical embedded application.

5 The collector is incremental, that means, the application can preempt the collector with a very fine time granularity. The needed atomic operations of the collector is constant and small and no extra bookkeeping is needed during a collection cycle.

4. TIMBER IN A REAL-TIME CONTEXT

Timber objects and TinyOS Components are very similar in terms of their design motivations. Both hold the dynamic property of a state, that will be repeatedly updated during the lifetime of the system. The updates will be made by procedures associated to the object or component. Timber objects include methods whereas TinyOS components includes tasks and command/event handlers. Notions of concurrency and reactivity are also supported on both platforms. TinyOS achieves concurrency through enabling event handlers to interrupt the current executing thread of control, either a task or another handler. Timber facilitates concurrency by enabling the scheduler to intervene the execution of a method at the occurrence of an interrupt.

4.1 Semantic Characteristics

Timber is, in contrast to TinyOS, an object-oriented language that not only allows, but rather demands the ability of dynamic storage allocation due to the extensive use of immutable data. Timber allows objects to be created dynamically with very little restrictions. An object may even be created by executing a locally defined template command within a method. TinyOS does not allow dynamic storage allocation at all.

The most unique and significant feature of Timber is its direct real-time support. Timing constraints can be expressed in the source code and will migrate into meaningful scheduling data used by the scheduler at run-time.

Timber eliminates the risk of data races by means of mutual exclusion between methods that may access common mutable data. Instead of relying on the programmer to avoid data races, this protection is induced by the language semantics.

4.2 Run-Time Characteristics

The presence of time is the major characteristic that permeates the entire run-time system. Every scheduling decision is based upon timing constraints originally expressed in the source code.

Another major part of the run-time system is the memory manager. The real-time characteristics of the memory manager is mainly due to the common memory usage behavior of Timber applications. The allocator is totally predictable (incrementing a free pointer) and the collector is transparent due to the fact that it only runs during idle time. The collector is furthermore incremental with a very fine granularity and no extra bookkeeping is needed during a collection cycle.

In contrast to TinyOS, the scheduler can perform scheduling decisions at the occurrence of any interrupt, if desirable. However, in cases where the deadline of an interrupt-handler is extremely short, the run-time system may also be configured to run the handler directly, without passing through the scheduler. The semantics of the language guarantees that only the ability to meet deadlines, not the meaning of the program itself, may be affected by such a choice.

4.3 Analysis and System Dimensioning

We have not discussed how Timber enables analyses for proper system dimensioning and verification. An example of this is shown in [18] by Svensson et al., where a WCET analysis for the schedulable units of Timber programs is presented. This is the first fundamental step in performing whole system schedulability analysis. It has also been shown that the fundamental tools in WCET analysis can be applied in other analyses, such as memory usage analysis [19].

5. CONCLUSION

We have shown how the run-time system of Timber facilitates the main infrastructure for reactive concurrent real-time software development. The integration of time is consistent and meaningful throughout the whole system, from the system specifications in the source code into each scheduling decision made in run-time. The combination of reactive objects and controlled use of mutable state eliminates the risk of data races and preserves state integrity. Dynamic memory storage allocation and garbage collection relieves the programmer from the error-prone task of manual memory management. The graph of objects may also, due to the ability to create objects dynamically, change rapidly over time, resulting in a more agile system than the case where the run-time structure is static. In contrast to TinyOS, Timber does not require any limitations in expressive power to avoid race conditions, this is instead guaranteed by the language semantics. Furthermore, the core of the run-time system has a greater potency (scheduling, memory management, etc.) than the corresponding parts of TinyOS, without introducing any unsafe real-time characteristics..

Timber mainly lacks two things in comparison to other minimalistic RTOS's such as TinyOS. First of all, TinyOS is well adopted, well tested, and well understood by practitioners. It has been available for several years and many practitioners have joined in. Second of all, the heritage of TinyOS is a lot more known by general practitioners due to the well-known imperative language C [13]. We cannot even start to compare C with Haskell in terms of how many well-skilled practitioners each programming paradigm has.

The language Timber includes more features than we have been able to cover within the scope of this paper. Features like polymorphism, sub-typing, inheritance, partial application, etc. may not be fundamental for embedded real-time

programming, but may prove useful also in this field. We have throughout this paper discussed the characteristics of the run-time system of Timber in the light of the common idea of how an RTOS should look like. In contrast to this point of view, the paper also points towards a rather new paradigm, to wit the importance of the programming language metaphor. The language Timber, with its concurrent object model, can actually be seen as the most fundamental part of the RTOS presented in this paper, due to the fact that all features of the run-time system is directly induced by the language semantics. This furthermore means that the run-time system is as complex (or simple) as the application needs. The programming language and the run-time system is thus not separate parts.

5.1 Future Work

Due to the fact that both the language Timber and its run-time system is still in the stage of development, many features are still missing. Garbage collection is still not fully implemented and integrated into the run-time system. Many features in the run-time system are still cumbersome and lack some functionalities. Even though the multiple message queues (one for each object) is a straightforward solution to accomplish the two level scheduling, it may not be the most efficient solution. It will be interesting to see if the two level scheduling can be accomplished by one queue solely. Another optimization that will increase the efficiency significantly in terms of memory usage is the use of a shared stack environment [6, 7], either completely shared or a hybrid solution. These optimizations will make the run-time system less unwieldy.

Due to the characteristics of the language, Timber provides ample opportunities to system analysis. It has been shown that WCET analysis can be performed on the schedulable units in Timber [18], which lays the foundation for whole system schedulability analysis. The knowledge of how WCET analysis can be performed will also be useful for the realization of memory usage analysis [19]. It will furthermore be interesting to see how the programming interface and the run-time system characteristics can render possibilities for an analysis framework including a definition of important behavioral attributes of embedded real-time software systems, and realization of tools to measure them.

6. REFERENCES

[1] Accelerated Technology official homepage, http://www.acceleratedtechnology.com/, 2005.

[2] FreeRTOS official homepage, http://www.freertos.org/, 2005.

[3] PicoOS official homepage, http://picoos.sourceforge.net/, 2005.

[4] Timber website at Luleå University of Technology, http://www.csee.ltu.se/index.php?subject=timber, 2005.

[5] TinyOS official homepage, http://www.tinyos.net/, 2005.

[6] T. P. Baker. A stack-based resource allocation policy for realtime processes. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 191–200, 1990.

[7] T. P. Baker. Stack-based scheduling of real-time processes. *Advances in Real-Time Systems*, pages 64–96, 1993.

[8] A. Black, M. Carlsson, M. Jones, R. Kieburtz, and J. Nordlander. Timber: A programming language for real-time embedded systems, 2002.

[9] M. Carlsson, J. Nordlander, and D. Kieburtz. The semantic layers of timber. *The First Asian Symposium on Programming Languages and Systems (APLAS), Beijing, 2003. C Springer-Verlag.*, 2003.

[10] 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, Tampa, Florida, USA*, 2004.

[11] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *In ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2003.

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[13] B. W. Kernighan and D. M. Ritchie. *The C programming language*. Prentice-Hall, Inc., 1978.

[14] J. Mattsson. Garbage collection with hard real-time requirements. Master's thesis, Luleå University of Technology, 2004.

[15] J. Nordlander. *Reactive Objects and Functional Programming*. PhD thesis, Chalmers University of Technology, 1999.

[16] J. Nordlander and M. Carlsson. Reactive objects in a functional language – an escape from the evil, 1997.

[17] J. Nordlander, M. Carlsson, M. Jones, and J. Jonsson. Programming with time-constrained reactions, 2005.

[18] L. Svensson, J. Eriksson, P. Lindgren, and J. Nordlander. Language-based WCET analysis of reactive programs, 2005.

[19] L. Unnikrishnan, S. D. Stoller, and Y. A. Liu. Optimized live heap bound analysis. In *Proc. 4th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 2575 of *Lecture Notes in Computer Science*, pages 70–85. Springer-Verlag, jan 2003.

Using Protothreads for Sensor Node Programming

Adam Dunkels
Swedish Institute of Computer Science
adam@sics.se

Oliver Schmidt
oliver@jantzer schmidt.de

Thiemo Voigt
Swedish Institute of Computer Science
thiemo@sics.se

ABSTRACT

Wireless sensor networks consist of tiny devices that usually have severe resource constraints in terms of energy, processing power and memory. In order to work efficiently within the constrained memory, many operating systems for such devices are based on an event-driven model rather than on multi-threading. While event-driven systems allow for reduced memory usage, they require programs to be developed as explicit state machines. Since implementing programs as explicit state machines is hard, developing, maintaining, and debugging programs for event-driven systems is difficult.

In this paper, we introduce protothreads, a programming abstraction for event-driven sensor network systems. Protothreads simplify implementation of high-level functionality on top of event-driven systems, without significantly increasing the memory requirements. The memory requirement of a protothread is that of an unsigned integer.

1. INTRODUCTION

Wireless sensor networks consist of tiny devices that usually have severe resource constraints in terms of energy, processing power and memory. Most programming environments for wireless sensor network nodes today are based on an event-triggered programming model rather than traditional multi-threading. In TinyOS [7], the event-triggered model was chosen over a multi-threaded model because of the memory overhead of threads. According to Hill et al. [7]:

> "In TinyOS, we have chosen an event model so that high levels of concurrency can be handled in a very small amount of space. A stack-based threaded approach would require that stack space be reserved for each execution context."

While the event-driven model and the threaded model can be shown to be equivalent [9], programs written in the two models typically display differing characteristics [1]. The advantages and disadvantages of the two models are a debated topic [11, 14].

In event-triggered systems, programs are implemented as *event handlers*. Event handlers are invoked in response to external or internal events, and run to completion. An event handler typically is a programming language procedure or function that performs an action, and makes an explicit return to the caller. Because of the run-to-completion semantics, an event-handler cannot execute a *blocking wait*. With run-to-completion semantics, the system can utilize a single, shared stack. This reduces the memory overhead over a multi-threaded system, where memory must be allocated for a stack for each running program.

The run-to-completion semantics of event-triggered systems makes implementing certain high-level operations a complex task. When an operation cannot complete immediately, the operation must be split across multiple invocations of the event handler. Levis et al. [10] refer to this as a split-phase operation. In the words of Levis et al.:

> "This approach is natural for reactive processing and for interfacing with hardware, but complicates sequencing high-level operations, as a logically blocking sequence must be written in a state-machine style."

In this paper, we introduce the notion of using *protothreads* [3, 6] as a method to reduce the complexity of high-level programs in event-triggered sensor node systems. We argue that protothreads can reduce the number of explicit state machines required to implement typical high-level sensor node programs. We believe this reduction leads to programs that are easier to develop, debug, and maintain, based on extensive experience with developing software for the event-driven uIP TCP/IP stack [4] and Contiki operating system [5].

The main contribution of this paper is the protothread programming abstraction. We show that protothreads reduce the complexity of programming sensor nodes. Further, we demonstrate that protothreads can be implemented in the C programming language, using only standard C language constructs and without any architecture-specific machine code.

The rest of this paper is structured as follows. Section 2 presents a motivating example and Section 3 introduces the notion of protothreads. Section 4 discusses related work, and the paper is concluded in Section 5.

2. MOTIVATION

To illustrate how high-level functionality is implemented using state machines, we consider a hypothetical energy-conservation mechanism for wireless sensor nodes. The mechanism switches the radio on and off at regular intervals. The mechanism works as follows:

```
enum {
  ON,
  WAITING,
  OFF
} state;

void radio_wake_eventhandler() {
  switch(state) {

  case OFF:
    if(timer_expired(&timer)) {
      radio_on();
      state = ON;
      timer_set(&timer, T_AWAKE);
    }
    break;

  case ON:
    if(timer_expired(&timer)) {
      timer_set(&timer, T_SLEEP);
      if(!communication_complete()) {
        state = WAITING;
      } else {
        radio_off();
        state = OFF;
      }
    }
    break;

  case WAITING:
    if(communication_complete()
       || timer_expired(&timer)) {
      state = ON;
      timer_set(&timer, T_AWAKE);
    } else {
      radio_off();
      state = OFF;
    }
    break;
  }
}
```

Figure 1: The radio sleep cycle implemented with events.

1. Turn radio on.
2. Wait for t_{awake} milliseconds.
3. Turn radio off, but only if all communication has completed.
4. If communication has not completed, wait until it has completed. Then turn off the radio.
5. Wait for t_{sleep} milliseconds. If the radio could not be turned off before t_{sleep} milliseconds because of remaining communication, do not turn the radio off at all.
6. Repeat from step 1.

To implement this protocol in an event-driven model, we first need to identify a set of states around which the state machine can be designed. For this protocol, we can see three states: *on* – the radio is turned on, *waiting* – waiting for any remaining communication to complete, and *off* – the radio is off. Figure 3 shows the resulting state machine, including the state transitions.

```
PT_THREAD(radio_wake_thread(struct pt *pt)) {
  PT_BEGIN(pt);

  while(1) {
    radio_on();
    timer_set(&timer, T_AWAKE);
    PT_WAIT_UNTIL(pt, timer_expired(&timer));

    timer_set(&timer, T_SLEEP);
    if(!communication_complete()) {
      PT_WAIT_UNTIL(pt, communication_complete()
                        || timer_expired(&timer));
    }

    if(!timer_expired(&timer)) {
      radio_off();
      PT_WAIT_UNTIL(pt, timer_expired(&timer));
    }
  }

  PT_END(pt);
}
```

Figure 2: The radio sleep cycle implemented with protothreads.

To implement this state machine in C, we use an explicit state variable, `state`, that can take on the values OFF, ON, and WAITING. We use a C switch statement to perform different actions depending on the `state` variable. The code is placed in an event handler function that is called whenever an event occurs. Possible events in this case are that a timer expires and that communication completes. The resulting C code is shown in Figure 1.

We note that this simple mechanism results in a fairly large amount of C code. The structure of the mechanism, as it is described by the six steps above, is not immediately evident from the C code.

3. PROTOTHREADS

Protothreads [6] are an extremely lightweight stackless type of threads, designed for severely memory constrained systems. Protothreads provide *conditional blocking* on top of an event-driven system, without the overhead of per-thread stacks.

We developed protothreads in order to deal with the com-

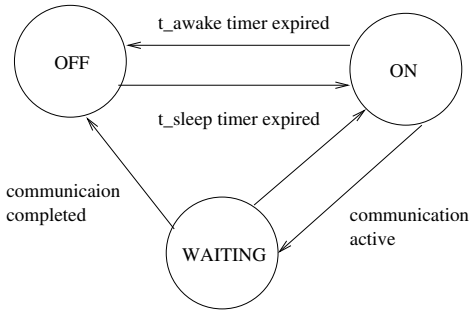

Figure 3: State machine realization of the radio sleep cycle protocol.

plexity of explicit state machines in the event-driven uIP TCP/IP stack [4]. For uIP, we were able to substantially reduce the number of state machines and explicit states used in the implementations of a number of application level communication protocols. For example, the uIP FTP client could be simplified by completely removing the explicit state machine, and thereby reducing the number of explicit states from 20 to one.

3.1 Protothreads versus events

Programs written for an event-driven model typically have to be implemented as explicit state machines. In contrast, with protothreads programs can be written in a sequential fashion without having to design explicit state machines. To illustrate this, we return to the radio sleep cycle example from the previous section.

Figure 2 shows how the radio sleep cycle mechanism is implemented with protothreads. Comparing Figure 2 and Figure 1, we see that the protothreads-based implementation not only is shorter, but also more closely follows the specification of the radio sleep mechanism. Due to the linear code flow of this implementation, the overall logic of the sleep cycle mechanism is visible in the C code. Also, in the protothreads-based implementation we are able to make use of regular C control flow mechanisms such as while loops and if statements.

3.2 Protothreads versus threads

The main advantage of protothreads over traditional threads is that protothreads are very lightweight: a protothread does not require its own stack. Rather, all protothreads run on the same stack and context switching is done by stack rewinding. This is advantageous in memory constrained systems, where a stack for a thread might use a large part of the available memory. In comparison, the memory requirements of a protothread that of an unsigned integer. No additional stack is needed for the protothread.

Unlike a thread, a protothread runs only within a single C function and cannot span over other functions. A protothread may call normal C functions, but cannot block inside a called function. Blocking inside nested function calls is instead implemented by spawning a separate protothread for each potentially blocking function. Unlike threads, protothreads makes blocking explicit: the programmer knows exactly which functions that potentially may yield.

3.3 Comparison

Feature	Events	Threads	**Protothreads**
Control structures	No	Yes	**Yes**
Debug stack retained	No	Yes	**Yes**
Implicit locking	Yes	No	**Yes**
Preemption	No	Yes	**No**
Automatic variables	No	Yes	**No**

Table 1: Qualitative comparison between events, threads and protothreads

Table 1 summarizes the features of protothreads and compares them with the features of events and threads. The

```
void radio_wake_thread(struct pt *pt) {
  switch(pt->lc) {
  case 0:

  while(1) {
    radio_on();
    timer_set(&timer, T_AWAKE);

    pt->lc = 8;
  case 8:
    if(!timer_expired(&timer)) {
      return;
    }

    timer_set(&timer, T_SLEEP);
    if(!communication_complete()) {

      pt->lc = 13;
  case 13:
      if(!(communication_complete() ||
           timer_expired(&timer))) {
        return;
      }

    }

    if(!timer_expired(&timer)) {
      radio_off();

      pt->lc = 18;
  case 18:
      if(!timer_expired(&timer)) {
        return;
      }

    }
  }
  }
}
```

Figure 4: C switch statement expansion of the protothreads code in Figure 2

names of the features are from [1].

Control structures. One of the advantages of threads over events is that threads allow programs to make full use of the control structures (e.g., *if* conditionals and *while* loops) provided by the programming language. In the event-driven model, control structures must be broken down into two or more pieces in order to implement continuations [1]. In contrast, both threads and protothreads allow blocking statements to be used together with control structures.

Debug stack retained. Because the manual stack management and the free flow of control in the event-driven model, debugging is difficult as the sequence of calls is not saved on the stack [1]. With both threads and protothreads, the full call stack is available for debugging.

Implicit locking. With manual stack management, as in the event-driven model, all yield points are immediately visible in the code. This makes it evident to the programmer whether or not a structure needs to be locked. In the threaded model, it is not as evident that

a particular function call yields. Using protothreads, however, potentially blocking statements are explicitly implemented with a `PT_WAIT` statement. Program code between such statements never yields.

Preemption. The semantics of the threaded model allows for preemption of a running thread: the thread's stack is saved, and execution of another thread can be continued. Because both the event-driven model and protothreads use a single stack, preemption is not possible within either of these models.

Automatic variables. Since the threaded model allocates a stack for each thread, automatic variables—variables with function local scope automatically allocated on the stack—are retained even when the thread blocks. Both the event-driven model and protothreads use a single shared stack for all active programs, and rewind the stack every time a program blocks. Therefore, with protothreads, automatic variables are not saved across a blocking wait. This is discussed in more detail below.

3.4 Limitations

While protothreads allow programs to take advantage of a number of benefits of the threaded programming model, protothreads also impose some of the limitations from the event-driven model. The most evident limitation from the event-driven model is that automatic variables—variables with function-local scope that are automatically allocated on the stack—are not saved across a blocking wait. While automatic variables can still be used inside a protothread, the contents of the variables must be explicitly saved before executing a wait statement. The reason for this is that protothreads rewind the stack at every blocking statement, and therefore potentially destroy the contents of variables on the stack.

Many optimizing C compilers, including gcc, are able to detect if an automatic variable is unsafely used after a blocking statement. Typically a warning is produced, stating that the variable in question "might be used uninitialized in this function". While it may not be immediately apparent for the programmer that this warning is related to the use of automatic variables across a blocking protothreads statement, it does provide an indication that there is a problem with the program. Also, the warning indicates the line number of the problem which assists the programmer in identifying the problem.

The limitation on the use of automatic variables can be handled by using an explicit *state object*, much in the same way as is done in the event-driven model. The state object is a chunk of memory that holds the contents of all automatic variables that need to be saved across a blocking statement. It is, however, the responsibility of the programmer to allocate and maintain such a state object.

It should also be noted that protothreads do not limit the use of *static local* variables. Static local variables are variables that are local in scope but allocated in the data section. Since these are not placed on the stack, they are not affected by the use of blocking protothreads statements. For functions that do not need to be re-entrant, using static local variables instead of automatic variables can be an acceptable solution to the problem.

3.5 Implementation

Protothreads are based on a low-level mechanism that we call *local continuations* [6]. A local continuation is similar to ordinary continuations [12], but does not capture the program stack. Local continuations can be implemented in a variety of ways, including using architecture specific machine code, C-compiler extensions, and a non-obvious use of the C *switch* statement. In this paper, we concentrate on the method based on the C switch statement.

A local continuation supports two operations; it can be either *set* or *resumed*. When a local continuation is set, the state of the function—all CPU registers including the program counter but excluding the stack—is captured. When the same local continuation is resumed, the state of the function is reset to what it was when the local continuation was set.

A protothread consists of a C function and a single local continuation. The protothread's local continuation is *set* before each conditional blocking wait. If the condition is true and the wait is to be performed, the protothread executes an explicit return statement, thus returning to the caller. The next time the protothread is invoked, the protothread *resumes* the local continuation that was previously set. This will effectively cause the program to jump to the conditional blocking wait statement. The condition is re-evaluated and, once the condition is false, the protothread continues to execute the function.

```
#define RESUME(lc) switch(lc) { case 0:
```

```
#define SET(lc)    lc = __LINE__; case __LINE__:
```

Figure 5: The local continuation *resume* and *set* operations implemented using the C switch statement.

Local continuations can be implemented using standard C language constructs and a non-obvious use of the C switch statement. With this technique, the local continuation is represented by an unsigned integer. The resume operation is implemented as an open switch statement, and the set operation is implemented as an assignment of the local continuation and a case statement, as shown in Figure 5. Each set operation sets the local continuation to a value that is unique within each function, and the resume operation's switch statement jumps to the corresponding case statement. The case 0: statement in the implementation of the resume operation ensures that the resume statement does nothing if is the local continuation is zero.

Figure 4 shows the example radio sleep cycle mechanism from Section 2 with the protothreads statements expanded using the C switch implementation of local continuations. We see how each `PT_WAIT_UNTIL` statement has been replaced with a case statement, and how the `PT_BEGIN` statement has been replaced with a switch statement. Finally, the `PT_END` statement has been replaced with a single right curly bracket, which closes the switch block that was opened by the `PT_BEGIN` statement. We also note the similarity between

Figure 4 and the event-based implementation in Figure 1. While the resulting C code is very similar in the two cases, the process of arriving at the code is different. With the event-driven model, the programmer must explicitly design and implement a state machine. With protothreads, the state machine is automatically generated.

The non-obviousness of the C switch implementation of local continuations is that the technique appears to cause problems when a conditional blocking statement is used inside a nested C control statement. For example, the case 13: statement in Figure 4 appears inside an if block, while the corresponding switch statement is located at a higher block. However, this is a valid use of the C switch statement: case statements may be located anywhere inside a switch block. They do not need to be in the same level of nesting, but can be located anywhere, even inside nested if or for blocks. This use of the switch statement is likely to first have been publicly described by Duff [2]. The same technique has later been used by Tatham to implement coroutines in C [13].

The implementation of protothreads using the C switch statements imposes a restriction on programs using protothreads: programs cannot utilize switch statements together with protothreads. If a switch statement is used by the program using protothreads, the C compiler will is some cases emit an error, but in most cases the error is not detected by the compiler. This is troublesome as it may lead to unexpected run-time behavior which is hard to trace back to an erroneous mixture of one particular implementation of protothreads and switch statements. We have not yet found a suitable solution for this problem.

4. RELATED WORK

Kasten and Römer [8] have also identified the need for new abstractions for managing the complexity of event-triggered programming. They introduce OSM, a state machine programming model based on Harel's StateCharts. The model reduces both the complexity of the implementations and the memory usage. Their work is different from protothreads in that OSM requires support from an external OSM compiler to produce the resulting C code, whereas protothreads only make use of the regular C preprocessor.

5. CONCLUSIONS

Many operating systems for wireless sensor network nodes are based on an event-triggered programming model. In order to implement high-level operations under this model, programs have to be written as explicit state machines. Software implemented using explicit state machines is often hard to understand, debug, and maintain.

We have presented *protothreads* as a programming abstraction that reduces the complexity of implementations of high-level functionality for event-triggered systems. With protothreads, programs can perform *conditional blocking* on top of event-triggered systems with run-to-completion semantics, without the overhead of full multi-threading.

Acknowledgments

This work was partly financed by VINNOVA, the Swedish Agency for Innovation Systems, and the European Commission under contract IST-004536-RUNES.

6. REFERENCES

[1] A. Adya, J. Howell, M. Theimer, W. J. Bolosky, and J. R. Douceur. Cooperative Task Management Without Manual Stack Management. In *Proceedings of the USENIX Annual Technical Conference*, pages 289–302, 2002.

[2] T. Duff. Re: Explanation please! Usenet news article, Message-ID: <8144@alice.UUCP>, August 1988.

[3] A. Dunkels. Protothreads web site. Web page. Visited 2005-03-18. http://www.sics.se/~adam/pt/

[4] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS '03)*, May 2003.

[5] 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*, Tampa, Florida, USA, November 2004.

[6] A. Dunkels and O. Schmidt. Protothreads – Lightweight Stackless Threads in C. Technical Report T2005:05, Swedish Institute of Computer Science.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000.

[8] O. Kasten and K. Römer. Beyond event handlers: Programming wireless sensors with attributed state machines. In *The Fourth International Conference on Information Processing in Sensor Networks (IPSN)*, Los Angeles, USA, April 2005.

[9] H. C. Lauer and R. M. Needham. On the duality of operating systems structures. In *Proc. Second International Symposium on Operating Systems*, October 1978.

[10] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The Emergence of Networking Abstractions and Techniques in TinyOS. In *Proc. NSDI'04*, March 2004.

[11] J. K. Ousterhout. Why threads are a bad idea (for most purposes). Invited Talk at the 1996 USENIX Technical Conference, 1996.

[12] J. C. Reynolds. The discoveries of continuations. *Lisp Symbol. Comput.*, 6(3):233–247, 1993.

[13] S. Tatham. Coroutines in C. Web page, 2000. http://www.chiark.greenend.org.uk/~sgtatham/coroutines.html

[14] R. von Behren, J. Condit, and E. Brewer. Why events are a bad idea (for high-concurrency servers). In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems*, May 2003.

Driving Forces behind Middleware Concepts for Wireless Sensor Networks

Kirsten Terfloth and Jochen Schiller
Freie Universität Berlin
Takustr. 9
14195 Berlin
+49 (0) 30 838 75211
{terfloth, schiller}@inf.fu-berlin.de

ABSTRACT
In the first days of the emergence of Wireless Sensor Networks (WSN) programming software involved expertise in both hardware and networking. Since then, various middleware architectures have been proposed to achieve a suitable abstraction from distribution and management tasks of sensor devices. This allows users to focus on the application development.
The approaches suggested so far differ in concept, functionality and envisioned abstraction. This paper surveys some representative approaches and states that only three different flavors of middleware exist, each of them addressing some characteristic abstraction middleware architectures are eager to provide. For future middleware implementations it will be beneficial to carefully balance the components of each class to obtain a mature design.

Keywords
Wireless Sensor Networks, middleware, classification.

1. INTRODUCTION
Wireless Sensor Networks (WSN) have recently received a lot of attention within the research community since they demand for new solutions in distributed networking. A common scenario associated with these networks is that tiny nodes, equipped with several sensors and hardware for wireless communication, are deployed randomly and in large numbers within a certain area. In order to report the data they gather in their proximity to an interested application or user, nodes connect to their neighbors and send valuable information on a multi-hop path to its destination.
New questions arise from the architecture and purposes of the network as well as from the embedded nature of the sensor nodes. The resources of single nodes, especially in terms of energy and memory, are very limited, thus system-oriented issues cannot be ignored in the software development process. Failure of nodes may be common if we assume the network to consist of huge numbers of cheap, error-prone nodes that, once deployed, shall never be touched again. The devices may simply run out of energy, loose connectivity due to shadowing or death of surrounding nodes.
Unlike other ad-hoc networks, nodes participating do not have their own objectives for being a member but rather serve a global effort or task introduced by a user. This leads to a shift from a local view of a node, towards a more sophisticated, data-centric understanding of the network as a global component. An application programmer wants to treat the network as an entity, to get information about certain data streams or events, and not to be concerned with single nodes. This perspective introduces new design and optimization goals: Prolonging lifetime and connectivity of the whole network becomes crucial while fairness or motivation issues do not have to be taken into account.
Also, sensor nodes deployed close to one another are likely to sense similar values within the same time interval. This spatial and temporal correlation can be exploited to optimize the network behavior.
To be able to provide a high-level interface to programmers, software is needed in between the system-oriented sensor node and the application. The following sections analyze and discuss the nature of such a middleware abstraction layer.

2. MOTIVATION AND BACKGROUND
Striving for a middleware layer in the domain of restricted devices like sensor nodes may seem inadequate at first. Aiming at the most efficient software possible, draining the last bits of performance and energy-savings seems to interfere with an architecture that allows customers with possibly limited programming skills to employ and use wireless sensor networks.
But while middleware certainly introduces some overhead compared to tailored software, a valuable abstraction can encapsulate functionalities very different applications depend on. This effectively reduces memory consumption due to reusability of components and also simplifies application development.
Looking at a functional level of a middleware, two areas of problems have to be dealt with properly: the embedded nature of the networked sensors and the distribution of a large amount of nodes. The different strategies to what extend these challenges are met within current approaches are discussed in Section 3.
On a conceptual level, middleware offering a well-known programming paradigm to the programmer is advantageous, since it establishes a specific view on the network. Users are able to use familiar structures, programming styles and patterns to implement the tasks they have in mind.
The challenges for middleware design have already been studied with focus on programming paradigms and general design principles. Römer [8] identifies several requirements that have to be fulfilled to efficiently support applications for WSN. Furthermore, he examines different programming paradigms used in existing middleware abstractions and discusses their advantages and disadvantages. Krishnamachari et al. [4] go a step further and develop a two-layer-architecture depending on their analysis. They combine a cluster layer to organize groups of nodes and a resource management layer, which is in charge of allocation and adaptation of resources depending on the current

state of the network. A definition of a set of common services a middleware should provide, including a standardized service interface to support several applications with possibilities to specify the quality of service needed, is suggested. As has been proposed in [1], they also emphasize on the importance of a lightweight and data-centric design which relies on localized algorithms.

When looking at existing middleware approaches for WSN, some interesting similarities can be found throughout the different programming paradigms and approaches. The major contribution of this paper is to identify and give an understanding of the driving forces in this area, and to point out the strengths of the different concepts applied.

3. REPRESENTATIVE APPROACHES

The following examples have been carefully selected from a huge pool of existent approaches to give an overview of the spectrum of concepts as well as functions existing platforms and suggested architectures for middleware supply. All projects are introduced in a short manner with focus on the services they feature from an application programmers' perspective.

3.1 Hood

Whitehouse et al. propose a middleware architecture called Hood [11] that provides an interface to a subset of the sensor nodes, called a neighborhood. Based on criteria for choosing neighbors and a set of attributes to be shared, a user can specify different kinds of neighborhoods. Hence a neighborhood may e.g. be formed by those nodes within a one-hop distance that are able to provide temperature readings. Hood basically handles any management issues arising, like supervision of neighborhood lists, data caching and sharing among nodes and the definition of messaging protocols.

Communication within a neighborhood is based on a broadcast/filter mechanism. If a node wants to share one of its attributes, it simply broadcasts the value. Incoming packets are filtered, and nodes can determine whether or not the received attribute is of interest and cache it. There is no feedback to the node that sent the value, so in contrast to concepts building upon reliable networking, Hood only needs asymmetric links between nodes.

Looking at the programming part of the approach, a neighborhood becomes a programming primitive. To create a new neighborhood and allow its individual parameterization, a code generation tool has to be invoked by the developer. The system itself builds upon TinyOS [3], an operating system widely deployed on sensor nodes. Several interfaces offered by Hood provide handles to access neighborhood attributes and define sharing and updating strategies. Furthermore, values of neighbors stored locally on a node can be annotated with so called scribbles. These simply note extra information, for example the quality of the link to the mirrored node

Overall, a programmer who builds applications upon Hood is given the possibility to address and control functional parts of the network together, instead of issuing single nodes. Thus, this project alleviates effort for maintenance and low-level concerns, and takes the burden of dealing with distribution from the programmer.

3.2 Maté

Reprogramming of nodes usually involves flashing a complete binary image. If this is done when the network has already been deployed, the complete software stack, which includes firmware, network protocols, middleware and application, will need to be distributed on a multi-hop path to every single node. Dependent on the size of the image, this procedure consumes a tremendous amount of energy, a resource that most of the times cannot be re-charged after node deployment. To tackle this problem Lewis et al [5] have developed a virtual machine, Maté, and a specific byte-code it can interpret. This way, software updates may solely implicate transmission of the new application.

Using Maté on sensor nodes presumes that applications have to be expressed in special Maté instructions. The design of a suit-able language is therefore crucial for the ability to express applications the network is supposed to accomplish, and thus for the success of the virtual machine. Mandatory goals include the need for concise instructions and dense byte-code to save energy on transmission, and an expressive but simple language to enable the envisioned wide spectrum of possible applications. The authors decided on a stack-based architecture and an instruction set programming style. Since Maté is highly dependent on TinyOS and makes use of its messaging infrastructure, the complete system architecture is optimized for a symbiotic relationship. The sizes of Maté instructions are for example customized to perfectly fit into TinyOS packages. Programs can thus be segmented into equal sized chunks, so called capsules, which is advantageous for on-the-fly software installations.

The instruction set of Maté combines low and high-level instructions, and allows three possible operand types to be used: values, sensor readings and messages. Besides basic instructions for arithmetic computations, halting and branches, sensor network specific commands are available and offer a convenient abstraction for an application programmer. A build-in routing algorithm can e.g. be called by issuing a single instruction, which is in charge of sending the specified package to its destination. Also, packets can forward themselves to install new applications in the network within a single instruction call. Furthermore, the instruction set includes eight instructions that may be implemented by the application programmer, and is thus tailored to the needs of a special application scenario.

A safe execution environment as provided by a virtual machine hides the complexity of the hardware or, in this case, TinyOS's complex, asynchronous execution model, and prevents system crashes. The instruction set design is especially targeted to the sensor network domain.

3.3 Generic Role Assignment

An approach to offer a configuration environment for nodes in WSN is suggested by Römer et al. [9] with the General Role Assignment (GRA) project. The large scale of a wireless sensor network lets configuration issues get intricate after deployment of nodes. A direct interaction between users and single nodes is not viable, which leads to the fact that automation is necessary in this context. Nodes will have to evaluate their own status within the network, compare it to surrounding nodes and then tune their behavior themselves.

To accomplish such self-organizing networks, the nodes agree on specific functions or roles each of them will take within the network. GRA acts as a framework that organizes the distribution, communication and evaluation issues involved in role constitution. Four key elements have to be defined to support such actions: First of all, characteristic properties of individual nodes have to be accessible to determine the state of an individual node.

These attributes may include static hardware elements like available sensors and their resolution, as well as dynamic features like remaining battery charge or eventually physical location. Any element of interest is maintained in a property directory on each node in a name-value list, and can be questioned using a provided interface.
Based on this directory, a programmer can depict roles for nodes to implement the envisioned task of the network. A role specification states the behavior of a node within its network context with regard to its local attributes and network properties. Each role is associated with an identifier and a set of conditions or rules that have to be met to assign it to a node. This evaluation, or role assignment algorithm, usually not only depends on the attributes of one, but on a group of spatially neighboring nodes, and thus builds upon distributed, localized interaction. The authors presume a similar software state of all participating nodes. Any change of condition may result in re-evaluation of roles, and therefore starts a reassignment process.
The last element this approach requires is a set of basic services. They encapsulate diverse functions the role assignment process or nodes can call at runtime. Routing, time management or other common features will be part of this library.
Application programmers are offered a new way of specifying the task a network has to accomplish. Instead of writing one application, a set of roles can be implemented to grasp the behavior of the network. The role specification is a configuration tool, once again helping a user to abstract from distributed management concerns.

3.4 TinyDB

TinyDB [7], among other distributed database approaches [2], is a very popular suggestion to alleviate network programming from application programmers. Although it has been discussed in depth before [8], it will be presented here briefly since it incorporates many interesting features. The idea of a middleware using a database abstraction is to enable the utilization of a well-known, declarative programming language upon the distributed nodes without having to deal with network issues. Therefore, the network established by the sensor nodes is understood as a distributed database which can be queried using a subset of SQL. The authors added some essential key-words specific to the sensor network domain to enhance the language respectively.
In this concept, each node contributes one row to a single, virtual table, and each column represents one of the attributes that can be queried. A query processor is run on every node to handle and possibly aggregate the sensor data questioned by the query specification. Thus to obtain values from the network, a user issues a query, which is then automatically routed to all nodes. TinyDB maintains a spanning tree from the node where the request has been initialized, so that resulting data can be sent back in reverse direction. Queries may be marked to be evaluated periodically, or values to be aggregated, summed up or grouped on their way back through the network. Any maintenance concerning bootstrapping or failure of nodes or routing issues will be handled by TinyDB without any interaction with the programmer.
TinyDB contributes with its SQL-style programming manner an interface that is already widely accepted. Since distribution issues are transparent to the user, even unexperienced users are able to task the network appropriately.

3.5 Impala

The last middleware component introduced here is Impala [4], an architecture implemented within the ZebraNet project [6]. The primary design goal has been to build a modular, lightweight run-time environment for applications that manages both devices and occurring events. Hence, Impala splits the field of duty into two layers, one to encapsulate the application protocols and programs for ZebraNet, and an underlying layer that contains functions for application updates, adaptation and event filtering.
Application programming follows an event-based programming paradigm, thus any application deployed upon the nodes has to implement a set of event- and data-handlers to respond to different types of events, including timer, packet, data and device events. Besides supplying event filter mechanisms, Impala emphasizes the need for integrating adaptation and updates of applications at runtime within the system architecture.
Adaptation of an application or an application-level protocol can become necessary due to changes of the system, e.g. failure of certain sensors or low battery level, as well as application specific modifications, e.g. a sudden drop of successfully delivered packages. A middleware agent, the Application Adapter, checks the overall state of the system on a regular basis and selects the most suitable configuration according to the present circumstances. Dynamic software updates may be mandatory during execution, but since the devices are not re-programmable at the same time (ZebraNet equips wildlife animals with sensor nodes, which results in a highly dynamic topology, so software can be received in incomplete bundles of packets) the Application Updater serves as a management component for available versions and code bundles. In contrast to Maté, Impala does not use a special instruction set and byte-code, but supplies compiled binaries that are linked whenever a complete new version is available. The proposed approach can be interpreted as a combination of operating system functionalities, a resource manager, a configuration tool and an event filter. Programs depending on Impala have a clear interface to a set of events. The middleware offers a management framework for applications beyond operating system issues.

4. CLASSIFICATION

All of the approaches presented in the last section can be categorized on behalf of the focus they are taking. We identify three different conceptual groups of middleware implementations, with each contributing a solution to one substantial problem for application development in WSNs. The scope of this classification is certainly not restricted to the examples, but holds true for almost all middleware architectures present. The affiliation to one group is neither exclusive nor discrete, so an approach may feature some elements of one group and totally incorporate attributes of another. Table 1 provides a subsumption of the approaches discussed into the recommended classes.

4.1 Group Abstraction

Several research teams have created or suggested middleware implementations that organize nodes of a network into groups sharing certain characteristics. The most common attribute to qualify nodes for membership in a subset of the network is network proximity. Clusters may for example include all nodes reachable in a two-hop neighborhood. But although proximity is one essential criterion, it fails to aid an application in search of sensor data of a specific kind. In this case, it will be more

valuable to use the sensor hardware to come to a decision on a group infrastructure. Any other characteristics may be applied as well, leading to a very widely applicable tool for node organization.

The group abstraction tackles those problems that arise for tasks due to the scale of a network and the distribution of nodes. Sensor networks are envisioned to contain nodes in the magnitude of hundreds or maybe thousands, thus there is no efficient way to obtain control manually. Rather, the network itself has to be able to cope with common issues that arise from using small, cheap and thus error-prone hardware. Likewise, lifetime of a network depends on the behavior of single node, and cooperation strategies are certainly indispensable to maximize it. The keyword commonly used in this context is localized algorithms, meaning that the state of the network has to be controlled by using algorithms that do not span over the whole network, but are executed within a defined spread of nodes, a group.

Other scenarios the group abstraction provides powerful tools for are setups that feature heterogenic hardware. For example nodes may not all be equipped with the same sensors, the same amount of memory or energy. In the Scatterweb network [10] some nodes posses a solar cell, and in case of sunshine these nodes are preferably chosen to route information. A middleware should be able to react to such environmental changes and man-age to configure the system accordingly.

Hood and GRA are two projects that explicitly provide a group abstraction. While Hood servers as a management entity for data sharing and networking issues, GRA emphasizes upon configuration of the network and leaves basic services, e.g. management of software updates or neighborhood discovery, up to an underlying software layer. Groups are built by nodes acting similar when executing the same role. Opposed to this, Hood implements an interface for applications to use the specified attributes of different groups.

The only contribution to group abstraction Maté fosters is the incorporation of TinyOS's Active Messages and their capability to address logically separate sensor networks. Impala does not recognize the need for grouping of sensors and assumes these functionalities to be implemented within the application software.

TinyDB supports the SQL statement GROUP BY. This statement is misleading, if we assume TinyDB will build a somehow clustered infrastructure when sending a query that contains this keyword. Instead, TinyDB maintains a global spanning tree for queries, and floods the request to all nodes. The result of a query will be presented in a structured manner, grouped by the chosen attribute, but no optimization with regard to networking issues is involved. Grouping is therefore possible at application level, but not implemented at network level.

4.2 Virtual Machine Abstraction

Middleware approaches that employ a virtual machine (VM) upon sensor nodes aim at improving the ease of use for the application programmer. More generally speaking, a virtual machine addresses the complexity originating from the underlying software, which can be within the range of pure firmware up to a complete operating system. In both circumstances a VM serves as a safe execution environment for application code, thus providing means to prohibit an application from crashing a node completely.

The impact of the chosen language interpreted on each device is important: The more sophisticated the language is, the more experienced a user usually has to be to program a task. On the other hand, if the VM supports a script language, application development will be a lot easier for non-experts, but may also lack the ability to express essential functionality, and may therefore in the worst case become obsolete. Besides this tradeoff, it also has to be taken into account whether to support a language already established, which is advantageous for rapid application development, or to provide a domain-specific solution. Due to the encapsulation of basic functionalities within the language and definition of WSN specific data types, applications may be expressed in a more natural and concise way in the latter case.

The developers of Maté provide a language that already holds commands for sending and receiving packets and a default routing algorithm. These high-level commands take the responsibility from the application programmer to specify the correct header information and wrap the data being sent in packages. Furthermore, values can be typed as sensor readings, supporting an intuitive way to handle data. But while the Maté instruction set offers a programming tool that partially abstracts from network management issues, the language itself requires solid knowledge of assembler programming and stack architectures. Inexperienced users are probably not the intended audience for Maté.

TinyDB also incorporates a VM abstraction in its implementation. In contrast to Maté, the language has been chosen with regard to rapid development goals. SQL, enhanced by domain-specific keywords, provides a fair and rather easy to use programming abstraction to deal with data collection within WSNs. At the downside, any user-defined optimization that goes beyond this scope, for example querying only a selected part of the network, cannot be expressed within the language.

The concept of Impala explicitly rejects a VM abstraction in favor of a runtime environment that supports update and adaptation management. The authors motivate their choice with the expected infrequent software updates and argue that compilation and linking will fit their envisioned target application better.

To specify the conditions a node has to satisfy for choosing a role, the developers of GRA suggest a framework that evaluates rules, Boolean expressions, eventually containing predicates over nodes properties. This concept is rather easy to understand and, since functions can be called by a rule, an expressive means to program. Whether the rules are really interpreted upon the nodes, or precompiled and linked is left up to the implementation.

Hood itself does not provide a runtime environment of any kind, but serves as a functional layer in between the operating system and the application.

4.3 Modular Service Architecture

Classical middleware implementations like CORBA or DCOM share the goal to hide complexity due to heterogeneity, distribution or communication from applications. The common way to implement an infrastructure supporting this goal is to define a set of basic services, implement them and offer a standardized interface, applications are free to invoke. Furthermore, a modular design is advantageous: As long as interfaces are being kept the same, the implementation of services can be exchanged without touching any software relying on the middleware.

This design is favorable, since it allows easier adaptation of software components than a monolithic design. Other than the group and VM abstraction, approaches of this category do not provide a solution to an inherent problem of the WSN domain, but rather emphasize the software development process. If modularity

can be obtained in a way that even parts of the middleware are exchangeable at runtime, this will of course be beneficial with regard to energy-consumption and adaptation.

When discussing services and a middleware layer, the question what service should be incorporated within this layer arises, and whether a general middleware abstraction layer exists. The examples above demonstrate that this question is still a topic of active research, if not even superfluous due to their application driven nature.

TinyDB supplies a framework that implements a default routing strategy as well as node discovery, scheduling and power management support. The newest version allows the user to integrate new routing and aggregation algorithms, thus the design has been changed towards a modularized architecture since its first release.

The virtual machine Maté serves as a runtime environment for applications, but also implements basic routing and software update strategies that can be invoked. In case a programmer identifies the need for more basic services, Maté offers eight instructions that may encapsulate their implementation. Services are mapped onto the instruction set level.

GRA can be interpreted as a single service offered to the application layer: the control of network structure by assignment of roles, thus behavior of nodes, within their network context. Since routing and management issues are left to an underlying service layer, these have to be incorporated on demand.

Services in Impala include application scheduling, adaptation and linking of software updates and device management, classical operating systems functionalities but also middleware features with its event-handling notion. To enable on-the-fly reconfigurations and linking of code at runtime, the design has to be modular, otherwise changes at runtime would not be possible.

Hood allows a user to specify and alter any TinyOS components and modules generated. The basic service it provides for programmers is an infrastructure handling data sharing and caching issues with regard to membership of nodes to groups.

As can be derived from the examples above, there is no clear perception of the nature of middleware services. They may range from very high-level support to basic operating system functions.

Table 1: Middleware approaches and their affiliation to the three categories group abstraction, virtual machine abstraction and service-oriented middleware.

Approach	Group	VM	MSO
Hood	++	0	0
Maté	0	++	+
Impala	0	0	++
GRA	+	+/++	n/a
TinyDB	0/+	++	+

5. CONCLUSION

In this paper we discussed five middleware architectures for Wireless Sensor Networks. Although they are very different in terms of provided services and interface, programming paradigm or architecture, they all share some conceptual views upon WSNs. We believe, that approaches designed to tackle common problems in programming for this application domain usually make use of at least one of the following techniques: Abstraction by providing a way of grouping nodes, abstraction by providing a virtual machine or usage of a modular, service-oriented architecture. While the first two abstractions deal with the distribution and embedded nature of the sensor network devices, the third concept contributes to the software development process. Evaluation of possible means to implement features of all three concepts may help in the development process of future middleware approaches.

6. REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, Wireless Sensor Networks: A Survey. In *Computer Networks*, 38(4): 393-422, March 2002.

[2] P. Bonnet, J. E. Gehrke, and P. Seshadri. Querying the Physical World. In *IEEE Personal Communications*, 7(5):10–15, 2000.

[3] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer and D. Culler. The nesC language: A holistic approach to networked embedded systems. In *Programming Language Design and Implementation (PLDI)*, June 2003.

[4] P. Juang, H. Oki, Y. Wang et al. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *ASPLOS-X*, San Jose, USA, October 2002.

[5] P. Lewis and D. Culler. Maté: A Tiny Virtual Machine for Sensor Networks. In *ASPLOS-X*, San Jose, USA, October 2002.

[6] T. Liu and M. Martonosi. Impala: A Middleware System for Managing Autonomic Parallel Sensor Systems. In *ACM SIGPLAN*, San Diego, USA, June 2003.

[7] S. R. Madden, M. J. Franklin, J. M. Hellerstein and W. Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *OSDI 2002*, Boston, USA, December 2002.

[8] K. Römer. Programming Paradigms and Middleware for Sensor Networks, *GI/ITG Workshop on Sensor Networks*, pp. 49-54, Karlsruhe, Germany, February 2004.

[9] K. Römer, C. Frank P. M. Marron and C. Becker. Generic Role Assignment for Wireless Sensor Networks. In *ACM SIGOPS European Workshop 2004*, Leuven, Belgium, 2004.

[10] J. Schiller, A. Liers, H. Ritter, R. Winter and T. Voigt. ScatterWeb - Low Power Sensor Nodes and Energy Aware Routing. In *HICSS 2005*, Big Island, January 2005.

[11] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: A Neighborhood Abstraction for Sensor Networks. In *ACM MobiSys 2004*, Boston, USA, June 2004.

[12] Y. Yu, B. Krishnamachari, and V.K. Prasanna. Issues in Designing Middleware for Wireless Sensor Networks. In *IEEE Network Magazin*, 2003.

Session 4: Dealing with limited resources

Processor Choice For Wireless Sensor Networks

Ciarán Lynch
Centre for Adaptive Wireless Systems
Cork Institute of Technology
Cork
Ireland
ciaranlynch@cit.ie

Fergus O' Reilly
Centre for Adaptive Wireless Systems
Cork Institute of Technology
Cork
Ireland
foreilly@cit.ie

ABSTRACT

Networking and power management of wireless sensor networks is an important area of current research. The choice of microcontroller to control such devices is critical to their performance. We discuss some of the features that affect the performance of a microcontroller in sensing applications, with particular reference to power. Consideration is given to the particular demands of TinyOS, the most prevalent sensor operating system. We survey some popular existing devices, and suggest how the performance of a new design might be optimised.

1. INTRODUCTION

Wireless sensor networking is one of the most exciting technologies to emerge in recent years. Advances in miniaturisation and MEMS-based sensing technologies offer increases by orders of magnitude in the integration of electronic networks into everyday applications. Traditional microcontroller design strategies have not reached the best possible power consumption, especially for the specialised application set of sensing networks.

Power efficiency is a prime concern in wireless sensors, whether powered by a battery or an energy-scavenging module. Trends in miniaturisation suggest that the size of wireless sensors will continue to drop, however there has not been a corresponding drop in battery sizes. For example, a standard 3V CR2450 lithium coin cell has an energy density of 240mAh/cm^3 at 3V [4]. – a sensing application requiring 4mAh per day with a twelve month deployment would require 6.1cm^3 of battery. In one test deployment of the Mica2 motes, an environmental monitoring application had a daily energy budget of 8.14mAh [8].

In this paper we consider the microcontroller and its effect on the power consumption of the sensor node. We review the factors influencing the power consumption and calculate the expected performance of some existing hardware in sensor applications.

Section 2 describes TinyOS, Section 3 describes some of the significant hardware factors affecting the performance of the microcontroller. Section 4 discusses existing microcontroller hardware, Section 5 the power used by these devices and Section 6 concludes the paper.

2. TINYOS

TinyOS, originally developed at the University of California, Berkeley [6], has emerged as the leading operating system in research relating to control of wireless sensor networks. Its modularity and C-based syntax aim to provide a shallow learning curve for an experienced programmer.

All of the controlling software runs on the microcontroller contained in the wireless sensor node. TinyOS was originally designed for AVR-based microcontrollers. Recently, a port for the Texas Instruments MSP430 has been carried out, and previous work [7] demonstrated how the basic operating system and a subset of the libraries can even be written for such a limited device as the Microchip PIC16.

TinyOS uses a two-level scheduler – tasks run in a FIFO queue, with the system sleeping when no task is scheduled. Events run asynchronously, usually from interrupt handlers, and will interrupt any task that is running, or wake the system from sleep. It also includes a wide range of library modules.

3. HARDWARE

Power conservation is almost always the principle factor in TinyOS application design. Some of the factors affecting the power conservation of a particular device are considered here.

3.1 Wakeup Time

A widely used goal for sensor applications is an average duty cycle of 1%. This is achieved by scheduling events for some time in the future, and then sleeping. Most microcontrollers implement an asynchronous timer which is clocked while the core and other peripherals are powered off. An interrupt is triggered when this timer rolls over to wake the system from sleep.

Interfacing with external hardware often results in waiting times in the core. A typical sequence consists of setting some bits, waiting a number of milliseconds, changing the settings, waiting some more milliseconds and so on. If the wakeup is sufficiently fast, the core may sleep while waiting.

The primary factor determining this is how quickly the main oscillator can start up and settle into a stable cycle. Depending on how timing-sensitive the application is, it may not be necessary for the cycle to have completely settled. It may be useful to start execution before the oscillator has completely settled, and provide a flag indicating when it has settled to

within a certain tolerance of the nominal value which can be checked before particularly sensitive operations.

Some architectures may allow the oscillator to keep running but not be connected to the microcontroller core, allowing the core to start up within a few clocks of an interrupt. For long periods of sleep the extra power drawn by the oscillator is likely to dominate the system power dissipation but if it is known in advance that the sleep is short (for example from a timer) the fast wakeup time may allow it to sleep where it would not otherwise be possible.

Recently, some architectures have been proposed which use entirely asynchronous logic [3] eliminating oscillator startup delay. However these architectures are still experimental, and some work is required before functioning silicon will be available.

3.2 Clock Scaling

Some microcontrollers can dynamically switch their operating frequency, either by using a divider on the primary system oscillator or a low-frequency oscillator with a controllable multiplier. The first approach runs the oscillator at full speed, dissipating maximum power, but the system logic operates at a lower speed and lower power. The multiplier has its own power overhead, and could require time to settle if the multiplying factor is changed, whereas the divider will be able to switch frequencies instantly. The multiplier-based approach saves power by running a low-speed oscillator. There is usually a 32kHz oscillator for timing purposes, so there is no power overhead apart from that of the multiplier itself.

It is important to consider the overhead of the switch – depending on how the clock division is carried out it may require a number of cycles to settle before execution continues. This may result in "dead time" where the microcontroller is powered on but not executing anything, wasting power.

3.3 Memory Architecture

When servicing an interrupt, the handler will often have to save a number of registers and flags before executing anything. Since an interrupt handler will generally execute very little code, this can also come to dominate the time for which the core is active.

In the case of TinyOS, event handlers may not pre-empt each other. This means that the interrupt handler executes with interrupts disabled, and nested interrupts are not possible. It is therefore not necessary to store the values on a stack, since there will only ever be one stored state. The most efficient solution would be to have two completely seperate sets of registers, and switch between them using a banked access, allowing the context switch to take place in a single instruction.

A compiled stack, as discussed in Section 3.6 seperates the two stacks completely, however this may introduce overhead in other areas. It is addressed efficiently in any architecture, since the memory access is always a constant address. Accessing local variables on a dynamic stack is typically done using an indirect address of the form *base register + constant offset* [12] – this should be done in a single instruction. Protection of the stack from overflow and underflow will enhance the reliability of the application.

A segmented memory architecture, with registers, stack and static data all accessed differently can produce very efficient code, but may result in complications when using pointers as it is difficult to define a generic pointer.

3.4 I/O Issues

Since the microcontroller is required to control a number of other devices, it must have a large number of digital I/O pins available for general use. For example, the Nordic nRF2401 [11], a reasonably advanced modern radio transciever operating at 2.4GHz, needs nine I/O pins connected to the microcontroller (six if only one data channel is being used). Digital sensors will generally have a serial interface, but this may still require up to four or five pins per sensor, as will an external FLASH memory.

Equally important is the logic behind them. Serial interface logic can allow the shifting out of digital data to be carried out in the background, or while the microcontroller core is powered down. Many digital sensors use standard serial bus protocols such as I^2C or SPI, hardware support will allow these to be used more efficiently.

Almost all microcontrollers contain a hardware UART, which can be used to control a standard serial port, using an external level-shifting chip. Although this is far too power-hungry for a battery-powered device, it is common for some of the nodes to function as wired "base-stations". The UART may also be used to control data clocking at the radio interface. Some UARTs may continue to run while the microcontroller core is asleep, waking it up with an interrupt whenever attention is required.

3.5 Instruction Set Complexity

The real power of sensor networks lies in the possibility of the network itself processing the data and taking action, or at least minimising the amount of data transmitted over the radio, saving power. Current microcontroller architectures generally do not lend themselves well to complex data processing tasks. The word size is eight bits, and general-purpose registers are usually scarce. Floating point calculations are inefficient.

The addition an *Add with carry* instruction makes carrying out sixteen or thirty-two bit arithmetic much easier on an eight-bit architecture. A hardware multiplier allows floating-point arithmetic to be carried out, although this will always be inefficient in an eight-bit architecture. More advanced instructions to carry out matrix operations, such as those found on some DSP processors allow digital filtering to be carried out at the sensor.

3.6 Software Issues

A processor with a large working set of registers will usually have a more flexible compiler than a processor with a load/store architecture.

Embedded compilers generally use one of two methods to assign function auto variables – a dynamic stack or a "compiled stack". A dynamic stack is the traditional LIFO queue,

which is accessed by *push* or *pop* instructions. A compiled stack determines the data locations at compile time, using a function call graph to determine which locations can safely be overlapped.

The advantage of a dynamic stack is that any configuration of functions can be supported, and functions can (if desired) be reentrant. The disadvantages are that some embedded architectures may not have dedicated stack access instructions. In a compiled stack all memory accesses are static, since the address is determined at compile time, result in efficient memory access.

4. REVIEW OF EXISTING HARDWARE

The existing platforms with a full TinyOS implementation are the ATMEL AtMega128L, the AT90LS8535, and the Texas Instruments MSP430. Many other platforms are in development, or have partial implementations. These and some other common microcontrollers are compared under some of the headings discussed in Section 3.

The microcontrollers considered are the ATMEL ATMega128L [2], the Microchip PIC16F877 [9], the TI MSP430C1351 [13], the Analog Devices ADuC845 [1] which contains an 8051 core, and the Microchip PIC18F4525 [10]. This represents a wide selection of the microcontrollers currently available "off the shelf". The MSP430 has a sixteen bit word size; all of the others use eight.

4.1 Power Saving

The MSP430 has six different power modes, ranging from fully active, to not clocking the core, to keeping the digital oscillator running to generate the clock but disabling the loop control to save power to fully powered down (with peripherals seperately enabled or disabled). Due to the use of the digital oscillator, wakeup time can be as low as 6μs.

The ATMega128L also has a variety of power down modes. The CPU clock can be stopped, leaving the peripheral clocks running. The CPU oscillator can be kept running – allowing it to restart in under one microsecond, or stopped. A special power-down mode stops all peripherals but the asynchronous counter and oscillator, which runs off an external 32kHz crystal.

The PIC18 has two power-saving modes – One runs the peripherals but not the core, the other powers down both (but allows the asynchronous timer to run). The 8051 device has similiar power-saving modes. The PIC16 is even simpler – it only has one power-saving mode. This allows the asynchronous timer to run with everything else powered off, or the ADC (allowing a more accurate conversion due to lower system noise).

4.2 Clock Scaling

The PIC18, the ATMega128L and the MSP430 all implement software-controlled clock scaling.

The MSP430 uses a low-frequency analog oscillator to generate its base clock (at 32kHz) and then multiplies this using a digital frequency-locked loop (FLL). The multiplier factor can be set in software, allowing the clock to be set to any value from 32kHz up to 6Mhz (at 3V). The FLL can also be disabled, decreasing the accuracy of the generated clock (since it is now functioning as an open-loop multiplier) but saving power.

The 8051 uses a similiar approach – using a phase locked loop (PLL) to generate a 12MHz clock and then dividing this down to give the system clock, from 6.3MHz (at 3V) down to 98kHz.

The ATMega128L implements a seven bit digital clock divider on the incoming clock signal. This affects the peripherals as well as the core. The core clock frequency must be at least four times the asynchronous clock frequency – giving a minumum core frequency of 128kHz.

The PIC18 can use an external crystal for precise timing, or an internal 8MHz RC oscillator. This can be divided down to any of eight set frequencies between 32kHz and the 4MHz. It can also be multiplied by four using a PLL to generate frequencies of up to 32MHz (although at 3V the maximum is 20MHz). The core can also use the same clock source as the asynchronous timer, running at 32khz. Switching clocks results in a delay of two cycles of the old clock plus three cycles of the new clock. The PIC18 has a "two-speed startup" option, where the clock is provided by the internal RC oscillator immediately when the device wakes up, and then transitions to the selected clock when it has stabilised.

The PIC16 does not support any clock scaling. The clock frequency must remain constant, between DC and 10MHz (at 3V).

4.3 Memory Architecture

The MSP430 has a single (Von Neumann) address space, with data RAM and program ROM all accessed by a single 16-bit pointer. It supports a variety of addressing modes and has dedicated stack instructions, and a stack pointer register.

The ATMega128L uses a Harvard architecture. It has a dedicated stack pointer and three dedicated (sixteen bit) indirect memory access registers, accessed directly or using a constant offset.

The PIC18 also uses a Harvard architecture. It contains a dedicated, 32-level function call stack, and a "fast save" area – the interrupt logic saves the three most commonly used registers here. The data memory is divided into 256 byte banks. The current bank is accessed in one instruction, otherwise the bank must be switched. The first half of bank zero and the second half of bank fifteen (containing peripheral access registers) are mapped into a special "access bank" which is always accessible in a single instruction. Included in this are three dedicated indirect access registers. Accesses using a base and an offset must be done explicitly.

The PIC16 uses a simplified version of the PIC18 architecture. It has a maximum of 512 bytes of accessible RAM, divided into four 128 bytes banks, of which 386 bytes are general-purpose. 16 bytes of this are mirrored across every bank. Apart from one indirect access register, memory accesses must be within the current bank. The function call

stack is also fixed in hardware, and only contains eight levels. It has one accumulator which is targetted by most of arithmetic instructions.

The 8051 supports up to 2048 bytes of Extended RAM, and 256 bytes of normal RAM. However, only the first 128 bytes of normal RAM is accessible directly, and this overlaps the register space. The register space consists of four banks of eight registers. The next 16 bytes consists of bit-addressable memory. This leaves 80 bytes of general-purpose RAM in this bank. The remaining 128 bytes consists of SFRs, and is only accessible indirectly, as is the ERAM. The 8051 contains stack pointer registers and stack instructions.

Both the 8051 and the PIC16 were not designed with C code in mind, and C code compiled for these architectures tends to produce large program images, although previous work [7] suggests that the compiled data stack of the PIC C compiler deals well with this for simple applications.

4.4 Instruction Set

Both the ATMega128L and MSP430 have rich instruction sets – a wide range of arithmetic instructions, dedicated multiply hardware and many addressing modes. The PIC18 and 8051 are more limited. One important factor for the PIC16 is the omission of a dedicated "add with carry" instruction, which makes even simple arithmetic on values greater than eight bits inefficient.

4.5 I/O

All of the microcontrollers feature a number of digital I/O ports. These are multiplexed with peripheral hardware, such as UARTs and external timers, so in a particular application, not all of them will be available.

The MSP430 features 40 digital pins, the ATMega128L 53, the PIC18 36, the PIC16 33, and the 8051 37.

All of the devices contain a hardware UART. The ATMega128L has hardware supporting I^2C and SPI, four timers and a ten bit, eight channel ADC. The MSP family has an extensive set of I/O options ranging from low-power devices with little I/O to devices with multiple ADCs, serial I/O support and even DMA controllers, although the particular device considered here does not have any of these. The PIC18 has hardware which can either carry out I^2C or SPI, four timers and a thirteen channel, ten bit ADC. The PIC16 is the same, but only three timers and eight ADC channels. The 8051 has three timers, SPI support in hardware, and two independent 24-bit ADCs.

4.6 Compiler

The MSP430 and ATMega128L are supported by a free port of gcc. The PIC16 and PIC18 have their own commercial compilers available, although the limitations of the load/store architecture make these relatively inflexible. The 8051 is supported by a number of C compilers, including SDCC, a free compiler. The existing compilers for the PIC architectures use a compiled stack, all others use a dynamic stack.

Commercial compilers for many of these architectures are available, however at the time of writing the TinyOS build system is focused on a gcc-based compiler.

5. RESULTS

The current consumption figures of the various states of each microcontroller are listed in Table 1, taken from the appropriate datasheet. For the purposes of comparison, all of the figures are given at operating frequencies of 8MHz and 1MHz, with no peripherals enabled in active mode, and at the same frequencies with only an asynchronous timer enabled for power-down mode. Where the chip is capable of dynamically switching to a 32kHz clock in active mode this is given. The ATMega128L must operate at at least 128kHz, but the value at 32kHz is still given here for comparison. Similiarly, the 8051 can only operate from 98kHz to 6.3MHz but values are extrapolated outside this range. All values are taken at an operating voltage of 3V. This does not give the energy used since different operations take different times on the various architectures.

The wakeup time is the time from an interrupt being signalled to the beginning of the interrupt handler – this does not include software overhead from saving registers etc. (see Section 3). The wakeup time for the PIC18 assumes two-speed startup is enabled – the initial part of the interrupt handler will execute with an RC oscillator, not the crystal oscillator if enabled.

Average current consumptions for a TDMA application are given in Table 2. The "Rx" application listens for a synchronisation signal once per second, "Rx+Tx" does this and also transmits a reply once per second, in the correct timeslot. The values used were measured on the PIC16 and AVR and estimated for the other architectures. Assuming all devices are powered by a 3V source, this is directly proportional to energy consumption since it takes into account the time the processor is actually active in each device.

A packet-level interface is assumed, so the microcontroller spends very little time in the active state. The most common operations are waking from sleep, setting an output pin, restarting a timer and returning to sleep, and using the SPI bus to shift out a packet to the radio chip. Of these operations, the only time the microcontroller is active is while restarting the timer. The power consumption is dominated by the wakeup time. These values are calculated with an operating frequency of 8MHz. The "Rx + Tx" application contains twice as many "wake + sleep" operations per second as the "Rx" application.

Table 3 shows the "Rx+Tx" application with the introduction of encryption, using values measured by P. Ganesan et al. [5]. The process of encryption consists of repeating simple mathematical operations and gives a good indication of the relative power consumption of the active modes of the various platforms. It is assumed that the 16-byte payload of the transmitted and received data must be encrypted and decrypted respectively. Two different algorithms are considered, RC5 and IDEA – the primary difference is that bitwise shifts are used in RC5 while multiplication is used in IDEA.

6. CONCLUSIONS

	AVR	PIC16	PIC18	MSP	8051
Word Size	8 bit	8 bit	8 bit	16 bit	8 bit
Max F at 3V	8Mhz	10MHz	20MHz	6MHz	6.3MHz
Power Down	8μA	20μA	2.6μA	1.8μA	21μA
Idle (1MHz)	0.5mA	220μA	120μA	55μA	n/a
Idle (8MHz)	4mA	1.5mA	843μA	440μA	n/a
Active (32k)	88μA	n/a	35μA	19.2μA	2.78mA
Active (1M)	2mA	220μA	480μA	240μA	4.05mA
Active (8M)	8mA	1.5mA	2.4mA	1.9mA	13.3mA
Wakeup	2ms	102μs	10μS	6μs	20μs

Table 1: Current consumption information

	AVR	PIC16	PIC18	MSP	8051
Rx	76.5μA	22.0μA	3.81μA	2.40μA	30.9μA
Rx + Tx	138.0μA	23.3μA	4.66μA	2.83μA	34.9μA

Table 2: TDMA current consumptions

Whle the microcontroller is a central part of any wireless sensor node design, little consideration has gone into choice of device in the past. There are a wide range of microcontrollers currently on the market – all of them offer broadly similiar features. Node lifetime is determined by battery life, so power conservation is the prime concern.

The type of memory and memory access instructions determine how efficiently code will execute. An overhead of a few instructions per memory access can significantly influence the executing duty cycle, and power dissipation. More complex instruction sets can also speed up processing tasks, allowing the system spend more time sleeping. A useful feature is the ability to self-program. In order to generate efficient code, a well-optimised version of gcc or a similiarly flexible, optimising compiler for the target architecture is a major advantage. However, due to the design of gcc it does not deal well with accumulator-based architectures such as the PIC.

Clock scaling can also be used to save power. Depending on the application, it may be more useful to run continuously at 32kHz and switch bits in the output without sleeping at all than run at 8MHz, waking up and sleeping. Clock flexibility tends to reduce power consumption.

In determining the power used by a sensing application (particularly using TinyOS) it is important to examine the most common operations. The "Timer- based state machine" approach is widely used to control external devices. While the system is only executing "useful" code for a single instruction, depending on the hardware it may require several milliseconds to power on and carry out the extra code required to handle an interrupt correctly.

In a typical application it can be seen that wakeup time can dominate power consumption. When the number of wakeups was doubled the power consumption of the AVR, with the slowest wakeup time, almost doubles while for the MSP, with the fastest wakeup, it only increases by 18%.

	AVR	PIC16	PIC18	MSP	8051
RC5	151μA	26.4μA	8.99μA	4.55μA	62.8μA
IDEA	148μA	27.8μA	8.26μA	7.56μA	75.1μA

Table 3: Encryption current consumptions

The MSP family of microcontrollers from Texas Instruments have recently seen wider use for sensor networking applications and it is expected that the MSP will out-perform the older architectures in use until now.

7. REFERENCES

[1] Analog Devices. *ADuC845/ADuC847/ADuC848 Datasheet, Rev. B*, 2005.

[2] Atmel Corporation. *ATMega128 Datasheet, Rev. 2467I-09/03*, 2003.

[3] A. J. M. et al. The Lutonium: A sub-nanojoule asynchronous 8051 microcontroller. In *proc. 9th IEEE Symposium on Asynchronous Circuits and Systems*, May 2003.

[4] Eveready Battery Co. *Energizer No. 2450 Engineering Datasheet*.

[5] P. Ganesan et al. Analyzing and modeling encryption overhead for sensor network nodes. In *WSNA '03: Proc. of the 2nd ACM int. conference on Wireless sensor networks and applications*, pages 151–159. ACM Press, 2003.

[6] J. Hill. A software architecture supporting networked sensors. Master's thesis, University of California, Berkeley, 2000.

[7] C. Lynch and F. O. Reilly. Pic-based TinyOS implementation. In *proc. 2nd European Workshop on Wireless Sensor Networks, EWSN 2005, Istanbul*, pages 378–385, Feb. 2005.

[8] A. Mainwaring et al. Wireless sensor networks for habitat monitoring. In *Proc. ACM Int. Workshop on Wireless Sensor Networks and Applications*, pages 88–97, Sept. 2002.

[9] Microchip Technology Inc. *PIC16F877 Datasheet, Revision C*, 2000.

[10] Microchip Technology Inc. *PIC18F2525/2620/4525/4620 Datasheet, Revision B*, 2004.

[11] Nordic VLSI ASA. *Nordic nRF2401 Datasheet, Revision 1.1*, June 2004.

[12] D. A. Patterson and J. L. Hennessy. *Computer Organization & Design, The Hardware / Software Interface*. Morgan Kauffman, 2nd edition, 1998.

[13] Texas Instruments. *MSP430C13x1 Datasheet, revised Sept. 2004*, 2004.

Power Characterization of a Bluetooth-Equipped Sensor Node.

M. Lundberg, J. Eliasson, J. Allan, J. Johansson, P. Lindgren.
EISLAB, Dept. of Computer Science and Electrical Engineering
Luleå University of Technology
SE-971 87 Luleå, Sweden
maglun@csee.ltu.se

ABSTRACT

Wireless Sensor Networks (WSNs) consist of small, autonomous devices with wireless networking capabilities. In order to further increase the applicability of WSNs in real world applications, minimizing energy consumption and size are important research topics. A WSN node itself is a complex system consisting of numerous components, and the energy consumption of the node depends heavily on the interaction between its components and their respective operation modes. To develop a power consumption model, we have investigated the power characteristics of a Bluetooth(BT)-equipped node based on COTS (commercial off-the-shelf) components running standardized protocols for communication. The characterization captures the transient behavior of the individual components as well as the dynamic behavior of the system as a whole. Although the parameters of the model are derived for a specific node, the model and our conclusions can be applied to WSN nodes in general. Based on our model the estimated lifetime of a battery powered BT-equipped node can range from a couple of days to several months depending on battery and usage. This result indicates that COTS based sensor nodes can be used in a wide range of applications.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) provide unique opportunities in environmental monitoring, industrial, health care, and military applications. WSNs are networks of several small, autonomous devices equipped with wireless communication. Over the years, WSNs has evolved from tiny data gathering networks [12] to functionally rich distributed systems [8]. Comprehensive surveys on WSNs can be found in [4, 21] where different aspects of sensor networking are discussed.

Minimizing energy consumption and size are important research topics in order to make WSNs deployable. As most WSN nodes are battery powered, their lifetime is highly dependant on their energy consumption [17]. In cases with hundreds of nodes, changing batteries can be an almost unachievable task due to the sheer number of nodes. Due to the low cost of an individual node, it is perhaps more cost effective to replace the entire node than to locate the node and replace or recharge its battery supply. In other scenarios the location of the node make battery changes infeasible; the node might be physically inaccessible (as embedded into the hosting equipment), the node may be located in an environment where human intervention is undesirable (such as a bird nest [14]), the node is situated in a dangerous environment (such as a chemical plant), or the node resides in rugged unaccessible terrain.

An effective solution to prolong the operational lifetime of a node is to apply energy scavenging methods such as solar technology [20], or vibrations [1][2]. Another approach is to apply a local RF field to temporarily power an individual node in order to retrieve data [10].

The work presented in this paper is based on COTS components and standardized protocols for communication, in our case the microcontroller based platform MULLE [11] featuring Bluetooth and TCP/IP communication. COTS based solutions provides valuable insight of the underlying problems and serve as a basis for experimental work and development of real world applications.

As wireless communication is one of the key issues for WSN nodes, the problem of energy-aware routing has been the focus of recent research [18, 5]. We however, investigate the power characteristics of an individual node in a single-hop network in order to develop a power consumption model. The node itself is a complex system consisting of numerous components. As the energy consumption of the node depends heavily on the interaction between its components and their respective context (operation mode), it is necessary to regard the transient behavior of the individual components as well as the dynamic behavior of the system as a whole. To formally analyze a WSN node would be intractable using today's methodologies and tools. Component characteristics are often publicly unavailable due to proprietary issues. Even if we could obtain such information, there would still be a lack of formal methods for complex system analysis. So we ask ourselves, given the complexity of a node, is it at all possible to derive a model of its energy consumption. If so, will the model be robust, intuitive, and applicable?

To address this problem, we undertake an applied approach, based on real life measurements conducted for representative contexts (operation modes for the node as a whole). In this paper we extend the general design methodology previously proposed in [13], by developing a model for the MULLE node. Although the parameters of the model are derived for this specific node, the model and our conclusions can be applied to WSN nodes in general.

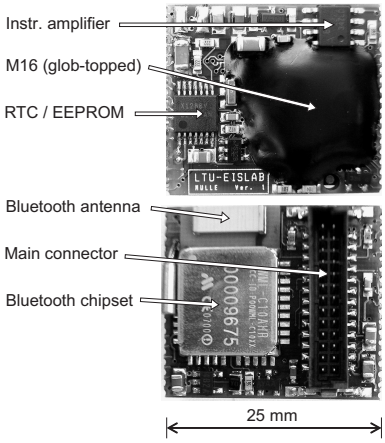

Figure 1: MULLE: front and back.

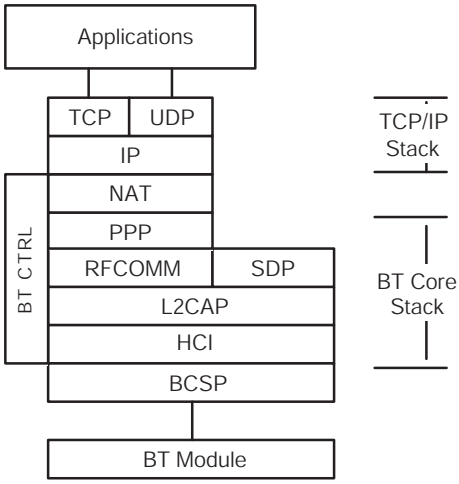

Figure 2: The MULLE software stack.

The outline of the paper is as follows; chapter 1 gives an introduction, chapter 2 describes the system setup, chapter 3 gives information on measurement setup, chapter 4 is a description of the characterization, chapter 5 explains the model, and the paper is concluded with discussion and results.

2. SYSTEM SETUP

2.1 WSN Architecture

A WSN node consists of a sensor, a microcontroller (MCU) and a communication device. The MCU acquires data from the sensor and transmits it using the communication device either raw or preprocessed.

2.2 Hardware

The MULLE, shown in Figure 1, is a wireless sensor node developed at EISLAB [3], Luleå University of Technology. The physical size of the sensor node is 25x23x5 mm. The batteries used are a selection of lithium or lithium-ion with capacity ranging from 120 mAh to 2200 mAh. It holds a Renesas M16 [15] microcontroller with 20 kB of RAM, mounted as a bare die to save space on the PCB. The wireless communication device chosen is a Bluetooth module [6]. The MULLE has a 3.0 V linear regulator controlling the power supply. A real-time clock (RTC) is also integrated and serves several purposes, it; provides the MCU with a sub clock, generates timer interrupts, and serves as non-volatile storage. A 26 pin connector allows a multitude of sensors to be connected to the MULLE using both analog and digital I/O's.

2.3 Software

To give the node the possibility to use available communication infrastructure, such as cellular networks and the Internet, all communication is performed using standardized protocols. Contained in the highest layer of the software hierarchy (Figure 2) are the applications; a generic sensor application which acquires and communicates data from the sensor, and a web server. The web server provides the user interface for the sensor application in the form of a Java applet. TCP/IP communications are handled by a lightweight stack lwIP [9] and Bluetooth communications are handled by lwBT [16]. No realtime operating system is used, all low level functionality is handled by an in-house hardware abstraction layer. In [19] several data delivery models are presented. The data delivery model used by the MULLE in this work is continuous, this means that data is sent at a pre-specified rate. Between transmissions, the MCU and the BT-module sleep and only wake up to send data or store data for later transmission.

3. MEASUREMENT SETUP

To capture the power consumption, a digital oscilloscope Tektronics TDS7254 was set up to measure the voltage over a series resistor. The oscilloscope has a vertical resolution of 8 bits with a gain accuracy of 2%. During measurement the oscilloscope has been setup to use as much as possible of the available resolution. The resistor was chosen to 10 Ohm to get a reasonably large signal to measure while keeping the voltage fluctuation over the MULLE low. The steady state current measurements were made with a digital Sourcemeter Keithley 2400. As the voltage provided to the MULLE is held constant we can calculate the power consumption. The Sourcemeter has a accuracy of 0.012% and a resolution of 5 1/2 digits. In order to validate the derived model (presented in chapter 5) real world life time tests were made. To give the MULLE a lifetime in the range of minutes, convenient for conducting repeated experiments, a capacitor was used as power supply in a similar manner to [18].

4. CHARACTERIZATION

To make a complete characterization of the MULLE. The measurements were made on three representative operating modes; sleep mode, data acquisition, and data transmission. These modes should apply to WSN nodes in general.

4.1 Sleep mode

In order to reduce the power consumption, the MULLE utilizes the sleep mode whenever possible. To investigate how much power individual components consume, measurements began with only the MCU, and the basic components it requires to operate, such as the RTC and voltage regulator mounted. The next step was to measure with the voltage references mounted, the voltage references are used by the MCU internal AD converter. The last component mounted

Components	Power
MCU + Regulator	270 μW
Voltage References	470 μW
Instrumentation Amplifier	450 μW
Bluetooth module	270 μW
Total Sum	1460 μW

Table 1: Power consumption of different parts on MULLE in sleep mode.

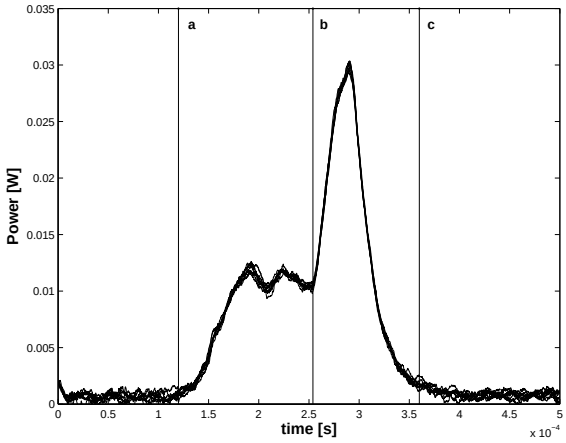

Figure 3: Ten overlayed measurements of the MULLE power consumption when acquiring one sample. At time *a* an interrupt is generated by the RTC and the MCU wakes up from sleep mode and starts initialization. At time *b* the MCU switches from 1.25 MHz to 10 MHz and acquires the sample. The MCU returns to sleep mode at time *c*.

was the instrumentation amplifier. The instrumentation amplifier is used to give MULLE an analog differential input. The results of the measurements are shown in Table 1. This characterization shows that for the MULLE, the sleep mode power consumption could be reduced from 1460 μW to 270 μW by introducing switches to completely power down the BT-module, voltage references, and the instrumentation amplifier when they are not used.

4.2 Data Acquisition

To characterize the MULLE during data acquisition, measurements were taken when the MULLE was reading analog values from a temperature sensor at a specific periodic rate controlled by interrupts generated by the RTC. Figure 3 shows ten overlayed measured activations. Each activation consists of the RTC generating an interrupt, the MCU initiating and acquiring one sample. During characterization tests where made with the MCU running at different clock frequencies. A comparison between 1.25 MHz and 10 MHz is shown in Figure 4. The tests showed that running the MCU at full speed (10 MHz) was most energy efficient. To fully take advantage of the reduced frequency an accompanying reduction in supply voltage would be required. As the supply voltage could not be adjusted the faster clock speed was preferred, allowing the MULLE to return to the low power sleep mode earlier.

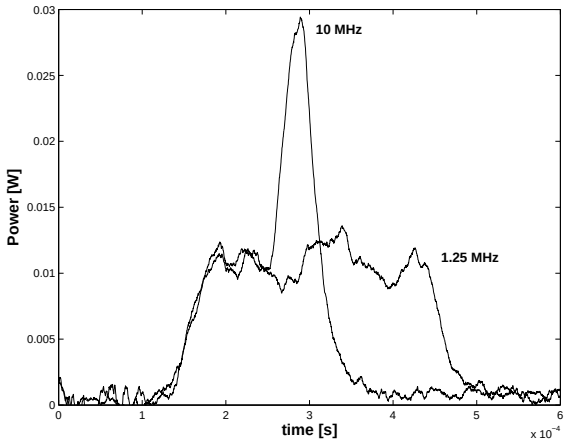

Figure 4: MULLE power consumption while running at 10 MHz vs 1.25 MHz. The total energy consumption for these cases are 2.8 and 3.4 μJ respectively, obtained by integrating the above curves.

4.3 Data Transmission

To be able to characterize the data transmission, measurements were made when the MULLE created a Bluetooth connection to a LAN-access point and sent 16 bytes of sampled data. The by far most energy consuming component of the MULLE is the Bluetooth-module, where most of the energy is spent during the connection phase. Sending of the actual measurement data only corresponds to about 1% of the energy used by the Bluetooth-module during transmission.

In order to decrease the time required to make the Bluetooth connection, a fixed Bluetooth address was used. By using this approach, there is no need to make a time consuming *Inquiry Scan* before each connection. The recommended time for scans is set to 10.24 s by the Generic Access Profile [7], so by reusing the previously used address, the connection setup time is reduced and made more predictable.

In many cases, such as when using a mobile phone or LAN Access Point as the Internet provider, the Bluetooth address will in fact remain constant over time. The only occasion it is necessary to make an *Inquiry scan* is when the provider is no longer operational, in which case the MULLE must find a new Internet providing device within its vicinity. A Bluetooth connection starts by synchronizing the Master and Client clocks. The next step is to set up an ACL radio link. When the ACL link is established, the power consumption initially fluctuates, leveling out at around 0.17 W, as seen in Figure 5.

The next steps involve setting up L2CAP, RFCOMM and PPP connections. All of these steps are considered to only be data traffic on the ACL link and do not cause any major variations on the power consumption.

When a PPP connection is established between the Client and the Master, TCP traffic flows from the Master to a server on the Internet. Once all TCP data is sent, the Master shuts down the ACL link causing all other layers to abruptly disconnect.

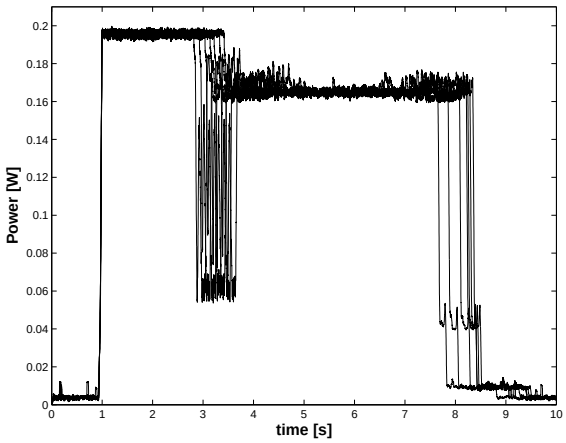

Figure 5: Ten overlayed measurements of the MULLE power consumption during a Bluetooth connection

Variable	Description	Unit
P_{sleep}	Power Consumption while sleeping	W
E_{acq}	Energy Consumption while acquiring data	J
E_{con}	Energy Consumption for a connection	J
f_{acq}	Frequency of acquisition	Hz
f_{con}	Frequency of connection	Hz

Table 2: Expression description

5. ENERGY CONSUMPTION MODEL

During characterization, the energy consumption for MULLE was measured in three representative operating modes, based on these measurement the model was formulated. The expression is given in Equation 1 with a description of the variables given in Table 2.

$$E = T \times (P_{sleep} + f_{acq} \times E_{acq} + f_{con} \times E_{con}) \qquad (1)$$

This model describes the total energy consumption for the MULLE platform given periodic data acquisition and transmission. It captures the complex interaction between the hardware and software components and incorporates operation mode transitions. In this way, the model provides the system designer with a simple tool that conceals the complex system implementation when dealing with application development and system dimensioning, e.g. battery capacity, sampling frequency and data aggregation.

Table 3 shows the average power consumption for MULLE depending on time in between activations/connections. It is obvious from the results that the communication interval has by far the most influence on energy consumption. To validate the model real world life time tests were made. Preliminary results indicate a good agreement with the model.

Time between Activations	Transmission interval			
	10 sec	1 min	1 h	24h
1 ms	129 mW	25 mW	4.5 mW	4.1 mW
1 s	127 mW	22 mW	1.8 mW	1.5 mW
> 1 s	127 mW	22 mW	1.8 mW	1.5 mW

Table 3: Average power consumption depending on time between activations and transmissions.

6. CONCLUSIONS

Wireless Sensor Networks are gaining increasing interest, and can be useful for example in; environmental monitoring, industrial, health care, and military applications. As most of the nodes are battery operated the dominant constraint for WSNs is power consumption. Due to the complexity of a WSN node, it is a difficult task to make a formal analysis of the energy consumption. Instead, we derive a model of the energy consumption by characterizing measurements, and we show that the model is robust, intuitive and applicable by comparison between expression and real lifetime experiments.

The model is useful to target operational lifetime and modes of operations, and to make design decisions to minimize energy consumption. The model can be used as a design tool e.g. to;

- Estimate battery operating life, given activation and connection frequency.
- Decide the number of activations or connections for a certain battery capacity.
- Make decisions on battery capacity, given a requested lifetime.

The derived model has been validated experimentally with promising results, but the internal resistance of the capacitance gives rise to voltage drops. This can cause the MULLE to reset, so further refinements to the measurements are needed. The characterization of the MULLE indicates possible improvements, for example by switching off the voltage references for AD conversion and power down the instrumentation amplifier and the BT-module. Such modifications are projected for the next revision of the MULLE. Future work also includes extending the model with support for routing costs in ad-hoc sensor networks.

7. REFERENCES

[1] Energy harvester by Ferro Solutions. http://www.ferrosi.com/, March 2005.

[2] Ipower energy harvesters by Continuum Control Corp. http://www.powerofmotion.com/, March 2005.

[3] Luleå University of Technology, Embedded Internet System Laboratory, Jan 2005. http://www.csee.ltu.se/eislabfo.

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38:393–422, 2002.

[5] Juan Alonso, Adam Dunkels, and Thiemo Voigt. Bounds on the energy consumption of routings in wireless sensor networks. In *Proceedings of the 2nd WiOpt, Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Cambridge, UK, March 2004.

[6] Bluetooth Module WML-C10 Class 2. MITSUMI ELECTRIC CO., LTD. http://www.mitsumi.com, February 2005.

[7] Bluetooth Specification v1.2 Core Specification, Version 1.2. November 2002. https://www.bluetooth.org/, Mar 2005.

[8] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[9] A. Dunkels. lwIP - A Lightweight TCP/IP Stack. http://www.sics.se/~adam/lwip/, February 2005.

[10] D. Friedman, H. Heinrich, and D.-W. Duan. A low-power cmos integrated circuit for field-powered radio frequency identification tags. In *Digest of Technical Papers 44th ISSCC Solid-State Circuits Conference*, pages 294 – 295. IEEE, Feb 1997.

[11] J. Johansson, M. Völker, J. Eliasson, Å. Östmark, P. Lindgren, and J. Delsing. Mulle: A minimal sensor networking device - implementation and manufacturing challenges. In *IMAPS Nordic 2004*, pages 265–271, 2004.

[12] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: mobile networking for 'smart dust'. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278. ACM Press, 1999.

[13] M. Lundberg, J. Eliasson, L. Svensson, and P. Lindgren. Context aware power optimizations of wireless embedded internet systems. In *Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference, IMTC 04*, volume 1, pages 91 – 95, May 2004.

[14] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, September 2002.

[15] Microcontroller M16C/62M. Renesas Technology Corporation. http://www.renesas.com/eng/, June 2004.

[16] C. Öhult. lwBT - A Lightweight Bluetooth Stack. http://www.csee.ltu.se/~conny/lwBT/, February 2005.

[17] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. IEEE Signal Processing Magazine, vol. 19, iss. 2, pp. 40–50, March 2002.

[18] Hartmut Ritter, Jochen Schiller, Thiemo Voigt, Adam Dunkels, and Juan Alonso. Experimental Evaluation of Lifetime Bounds for Wireless Sensor Networks. In *Proceedings of the Second European Workshop on Sensor Networks (EWSN2005)*, Istanbul, Turkey, January 2005.

[19] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless microsensor network models. In *ACM Mobile Computing and Communications Review* (MC2R 2002).

[20] Thiemo Voigt, Hartmut Ritter, and Jochen Schiller. Utilizing solar power in wireless sensor networks. In *The 28th Annual IEEE Conference on Local Computer Networks*, October 2003.

[21] A.L.A.P. Zuquim, L.F.M. Vieira, M.A. Vieira, A.B. Vieira, H.S. Carvalho, J.A. Nacif, Jr. Coelho, C.N., Jr. da Silva, D.C., A.O. Fernandes, and A.A.F. Loureiro. Efficient power management in real-time embedded systems. In *Emerging Technologies and Factory Automation, EFTA '03*, volume 1, pages 496 – 505, Sept 2003.

Realizing Robust User Authentication in Sensor Networks

Zinaida Benenson*
Chair of Computer Science 4
RWTH Aachen University
zina@i4.informatik.rwth-aachen.de

Nils Gedicke*
Chair of Computer Science 4
RWTH Aachen University
nils.gedicke@post.rwth-aachen.de

Ossi Raivio
Chair of Wireless Networks
RWTH Aachen University
ora@mobnets.rwth-aachen.de

ABSTRACT

We investigate how to organize access control to the WSN data in such a way that an unauthorized entity (*the adversary*) cannot make arbitrary queries to the WSN. We call this problem *authenticated querying*. Roughly, this means that whenever the sensor nodes process a query, they should be able to verify that the query comes from a legitimate user. Authenticated querying is especially challenging if the adversary can gain full control over some sensor nodes through physical access (*node capture* attack). We propose first solution to this problem and present our experiments with an implementation of the first step of this solution.

1. INTRODUCTION

Wireless sensor networks (WSN) gather environmental data, process and store them, and finally give these data to the user, either on demand or upon event detection. This means that sensor networks offer services to the users.

Consider a sensor network spread over a large geographic area. The maintainer of the sensor network offers services to a large number of mobile users. For example, sensor readings could be used for precision agriculture. Farmers can subscribe to the services and remotely query sensors on their fields using some mobile device like a PDA. In this case, only the queries of a legitimate user should be answered by the network. However, existing query processing systems (for an overview, see e.g. [18]) are not concerned with this issue. Meanwhile, this problem becomes especially difficult in presence of node capture attacks.

Node capture means gaining full control over a sensor node by a physical attack, e.g., opening the sensor's cover and appending some wires somewhere. This type of attack is fundamentally different from gaining control over a sensor remotely through some software bug, e.g., a buffer overflow. As all sensors are usually assumed to run the same software, in particular, the same operating system, finding an appropriate bug would allow the adversary to control the whole sensor network. In contrast, a node capture attack can only be mounted on a small portion of a sufficiently large network.

In presence of node captures, critical tasks such as posting queries to the WSN, cannot rely on a single sensor, as otherwise a captured sensor could make arbitrary queries on behalf of the adversary, or give fake data to a legitimate user. In previous work [2] we identified and formalized data access issues in this adversary model.

Contributions

We identify a new security issue for WSNs — the *authenticated querying* problem. We propose first, to our knowledge, solution to this problem in presence of node capture attacks.

We report our experience with an implementation of the first step of this solution, a robust user authentication mechanism. The user needs to communicate with several sensors concurrently before he receives access to the WSN data. We investigated the feasibility of this approach for WSNs consisting of Telos Revision B motes [11] which run TinyOS [16].

Our user authentication protocol is based on public key cryptography which is said to be feasible for WSNs even without special hardware support [17, 9]. To verify these claims, we implemented digital signatures with the EccM library for elliptic curve cryptography by D. Malan [9]. From the logical point of view, public key crypto is more appropriate for our task than symmetric key cryptography, see Section 2.3 for the corresponding reasoning. To our knowledge, this is a first attempt to implement an advanced authentication protocol using the EccM library.

Roadmap

In Section 2, we present the authenticated querying problem. After outlining our system model (Section 2.1), we define authenticated querying (Section 2.2) and outline a possible solution to this problem (Section 2.3). In Section 3 we present our initial implementation of this solution. We first give the protocol specification (Section 3.1), then the experimental results (Section 3.2). Analysis of the protocol is presented in Section 4. We refer to related work in Section 5 and conclude in Section 6.

2. AUTHENTICATED QUERYING

2.1 System and Adversary Model

We assume a large static sensor network. The users can access the network using some mobile device. The users are mobile, but during a particular querying process they have to remain in place. On average, there are n sensors

*Zinaida Benenson and Nils Gedicke were supported by German Research Foundation (DFG) as part of the Graduate School "Software for Mobile Communication Systems" at RWTH Aachen University.

in the communication range of the user. Of these, t sensors are allowed to fail or to be malicious, meaning that they are captured and run programs which are different from the expected ones. For example, they can falsely authenticate illegitimate users, deny access to legitimate ones, or alter data which they are supposed to send to the user. Therefore, the user can rely for communication on at most $n-t$ sensors in his communication range.

2.2 Problem Statement

In general, whenever sensor nodes process a query, they should be able to verify that the query comes from a legitimate user. We call this problem *authenticated querying*. More formally, a WSN enables authenticated querying if it satisfies the following properties (perhaps, with some large probability):

- *Safety*: If a sensor s processes the query q, then q was posted by a legitimate user U.
- *Liveness*: Any query q posted by a legitimate user U is processed by at least all sensors $s \in S_q$, where S_q is the set of sensors which must process the query in order to give the required answer to the user.

Next section outlines one of possible solutions to authenticated querying in presence node capture attacks.

2.3 An Idea for Authenticated Querying

If the number of users is large, the natural method to use for authentication is public key cryptography because of its scalability. On the other hand, public key cryptography is power-hungry, so the sensors should communicate with each other using symmetric cryptography. Our concept is to let the sensors in the communication range of the user serve as interpreters (or a gateway) between the "public key crypto world" of the user and the "symmetric crypto world" of the WSN. The user talks to sensors in his communication range using public key cryptography, and these sensors then talk to the remainder of the sensor network on behalf of the user using symmetric cryptography. This happens in authenticated fashion:

1. *Robust secure channel setup between the user and the WSN*: The user executes a mutually authenticated key establishment protocol [10] using public key cryptography with a specified number of sensors in his communication range. The protocol results in establishment of symmetric session keys between the user and each correct sensor which participated in a protocol run.
2. *Authenticated query forwarding*: After the successful secure channel setup, the sensors in user's proximity forward user's query into the sensor network and append to it some additional information which enables the other sensors to verify the legitimacy of the query.

We briefly outline a possible solution to authenticated querying assuming that the query is addressed to a single sensor s. The user first sends his query to the surrounding sensors using secure channels. Then the sensors in user's communication range compute symmetric keys which they share with the sensor s [8]. Each sensor computes message authentication code (MAC) on the query. These MACs are sent back to the user who appends them to the query. The sensor s answers the query only if enough MACs are appended.

Note that in this solution, no coordination between the sensors in user's proximity is required. They do not need to know n or m. In case an entity is unable to authenticate to m sensors, not enough MACs are appended to the query, and therefore, the query is rejected by sensor s. Solutions where no coordination between the sensors is needed should generally be preferred. Otherwise, coordination requires additional messages, and therefore, additional resource consumption.

The above solution explains the idea and shows the feasibility of our approach. However, it has many drawbacks. For example, the legitimacy of the query cannot be verified by intermediate sensor nodes which route the query to the sensor s. This feature would enable early rejection of illegitimate queries. We consider more practical and sophisticated solutions in a concurrent work [1].

3. A PROTOCOL FOR ROBUST USER AUTHENTICATION

To verify the feasibility of the above idea for authenticated querying, we partially implemented the first stage of the solution. In our implementation, the user unilaterally authenticates to the sensors in his proximity using public key cryptography. We call this part of solution ***robust user authentication***.

Mutual authentication and session key establishment are left to the future work.

3.1 Specification

3.1.1 Choice of a Public Key Cryptosystem

RSA with small public exponent ([7, 17]) and Rabin public key cryptosystems ([5]) have fast algorithms for encryption and digital signature verification. However, decryption and signature generation are slow and resource-demanding. Therefore, these cryptosystems can be used in sensor networks only if the sensors are not required to decrypt or to sign messages.

In contrast, elliptic curve cryptosystems (ECC) [7, 9] require more overhead for encryption and signature verification than for decryption and signing. Nevertheless, with ECC, not only encryption and signature verification, but also decryption and signing are feasible for sensor nodes. As our full solution targets mutual authentication, we chose ECC for implementation of robust user authentication, even though the sensors in this case have to execute relatively expensive for ECC signature verification.

3.1.2 System Setup and Assumptions

We consider a public key infrastructure (PKI) for elliptic curve cryptography. There is a certification authority CA in the system. This could be the base station. The CA has a private/public key pair ($priv\_key_{CA}$, pub_key_{CA}) and gives to

each legitimate user a certificate. The certificate is the user's public key signed by the CA: $cert_U = sign_{CA}(pub\_key_U)$.

Each sensor has the public key of the CA preloaded. With this public key, each sensor can verify the certificate. We used the EccM library [9] to implement a digital signature method with message recovery called the Nyberg-Rueppel scheme [14]. "Message recovery" means that upon signature verification, the verifier also gets the signed message. Therefore, the certificate verification process also extracts the public key of the user from the certificate.

Standard certificates include, in addition to a public key signed by the certification authority, also at least user's name and the expiration date [10]. However, we propose a different approach for sensor networks. As time synchronization is a difficult issue in WSNs, and Telos motes do not have a real-time clock, we propose a different certificate management strategy. The certificate consists only of a signed public key of the user. The public key of the certification authority is changed periodically and distributed to all sensors in the WSN. This can be done, e.g., daily or monthly, depending on the application. Any entity which possesses a public key signed with the current key of the certification authority, can access the network. The users could refresh their keys at the web site of the WSN maintainer.

3.1.3 Parameters

- n sensors (on average) in the communication range of the user, denoted as $s_1, \ldots, s_n$
- $t < n$ sensors can be captured and run programs which are different from the correct, authorized one.
- $t < m \leq n - t$ is the number of sensors which must successfully authenticate the legitimate user, and deny access to illegitimate one. W.l.o.g. we denote these sensors as $s_1, \ldots, s_m$. For example, if $n = 10$ and $t = 3$, we could set $m = 6$.
- For the calculations concerning the elliptic curves we use the parameters hardcoded in the original EccM library by David Malan. With these parameters, it is possible to manage 163 bit keys which is a sufficiently large value for a secure authentication algorithm.

It is worth mentioning that 163 bits is the size of a private key. Public keys consist of two 163-bit numbers, and Nyberg-Rueppel digital signature on a 163-bit message also consists of two 163-bit numbers. Therefore, a certificate, which is basically a signed by a certification authority public key, consists of four 163-bit numbers.

In the following, we describe our protocol. It is based on a standard challenge-response protocol with digital signatures [10].

3.1.4 Protocol

1. $U \Rightarrow WSN$: $(U, cert_U)$

   The user stands somewhere in the sensor field and starts the protocol. He first broadcasts his identity and certificate to the sensors in his communication range.

2. $\forall\, 1 \leq i \leq m$: $s_i \rightarrow U$: $(s_i, nonce_i)$

   Upon receiving user's first message, each sensor s_i saves it and sends his identity and a random challenge $nonce_i$ to the user. "Nonce" means "number used only once" and is a standard term from the cryptography.

   Because the first message is broadcast, all the sensors hear it practically simultaneously. If all the sensors reply immediately, this will result in collisions. Therefore, a random backoff is needed.

   The medium access control (MAC) layer in Telos motes [12] is practically the same as B-MAC [13] which has been part of TinyOS already for some time. When there is data to be sent, medium is detected. If the medium is free, random backoff of $(0, 320, \ldots, 4800)$ µs is used. After this period, the medium is detected again. If it is free, packet is sent. If the medium is busy, congestion backoff of $(0, 320, \ldots, 20160)$ µs is used. There are at most eight retries.

   In our tests the backoffs provided by the operating system were not enough and at worst all the packets were lost due to collision. Because of this, we use a random backoff of $(0, 1, \ldots, 255)$ ms before trying to send the response to the broadcast.

3. $\forall\, 1 \leq i \leq m$: $U \rightarrow s_i$: $sign_U(h(U, s_i, nonce_i))$

   The user answers the challenge to each sensor. He first constructs a hash $h(U, s_i, nonce_i)$ using the hash function h. In our implementation, we used the SHA-1 hash function [15] which provides 160-bit hash values. Here, the identity of the user U, the identity of the sensor s_i and the nonce are concatenated before hashing.

4. s_i: `verify`$(cert_U) := pub\_key_U$

   s_i: `verify`$(sign_U(h(U, s_i, nonce_{s_i})))$

   Each sensor s_i extracts user's public key pub_key_U from the certificate $cert_U$. It then uses the extracted public key to extract the content of the third protocol message which is supposed to be $h(U, s_i, nonce_i)$. Then the sensor independently computes the hash value and compares it to the extracted one. The user is successfully authenticated if and only if the values are equal.

3.2 Experimental Results

3.2.1 Implementation details

We implemented our protocol on Telos Rev. B motes [11] using the EccM library for elliptic curve arithmetic [9]. The algorithms to sign a message and to verify a signature were implemented by combining and modifying the functions of EccM. Nevertheless, some more functionality was needed like a modular multiplication function and the procedure to generate the SHA1 hash. The whole program uses 45.5 kB of ROM and 2.0 kB of RAM.

3.2.2 Protocol execution with one mote

The charge and time consumption for one mote is shown in Table 1. It is based on the measured execution time of the implementation and nominal figures for power consumption from [11]. Absolute majority of overall time and sensor's energy are consumed by its calculations. The user's energy

Step	No. of packets	Time [s] U	Time [s] s	Charge [mC] U	Charge [mC] s
1	9	0.011	0.011	0.24	0.24
2	1	0.001	0.001	0.03	0.03
3	3	65	0.004	117	3.6
4	1	0.001	375	20	675

Table 1: Time and charge consumption in the case of a user (U) and one sensor (s). Number of packets contains both the received and sent data packets and acknowledgments.

is mainly consumed in signing operation. Radio communication takes only a minute part of the whole.

One authentication takes approximately 440 s and consumes 140 mC on user side and 680 mC on sensor side. A good battery contains a charge of 1500 mAh. If not doing anything else, a sensor can run our authentication cycle around 8 000 times and user 40 000 times before exhausting their energy sources.

As our user also was a sensor, instead of being a more powerful device like a PDA or a mobile phone, execution time and energy consumption in step 3 (user signs a nonce) would be much lower in real life.

3.2.3 Protocol execution with several motes

We performed several test runs with several combinations of $1 \leq n \leq 5$ and $1 \leq m \leq 5$, where n is the number of sensors in user's communication range, and m is the amount of sensors required to authenticate the user. Because our conditions were near-ideal the system behaved as it theoretically should have behaved: no packet losses or deadlocks were encountered, legitimate user was never denied access in the presence of enough sensors. Thus, the initial results seem promising. However, more exhaustive testing with e.g. radio interference, longer distances, absence of line of sight, and specifically tailored bogus messages are needed in order to be assured of the protocol's robustness.

4. PROTOCOL ANALYSIS

4.1 Impersonation Attacks

As in our implementation the sensor nodes do not authenticate to the user, a single captured sensor node could impersonate $m - 1$ valid sensor nodes and authenticate the adversary. However, in a fully implemented solution to authenticated querying, sensor nodes also authenticate to the user, and append some information to the query, such that other sensors can verify the legitimacy of the query. Therefore, care should be taken that the proof of query legitimacy cannot be produced by less than m sensors (perhaps, with some large probability).

For example, in the solution outlined at the end of Section 2.3, enough MACs should be appended to the query. An adversary which captured less than m sensors, would not be able to produce m valid MACs needed for authentication information.

4.2 Denial-of-Service Attacks

Denial-of-service (DoS) attacks are most difficult to mitigate, as they target the inherent resource constraints in WSNs [19].

The easiest DoS attack on wireless communication is jamming. Besides, as in Steps 2 and 3 the user communicates with at least m sensors concurrently, DoS by deliberate collisions on MAC layer should be taken into account. Some defenses are presented in [19].

As sensors save user's certificate, there is a possibility for DoS attacks by broadcasting several bogus certificates in Step 1 of the protocol. The sensor then would have to save them (and the nonces in Step 2) and thus sensor's memory could be exhausted. Therefore, we do not allow a sensor to run authentication protocols with multiple users concurrently. This does not seem to be a severe restriction in our setting, as we do not expect several users to be in the same location anyway.

However, there is a very severe DoS possibility against which no strong defense is known. If an adversary sends a bogus certificate in Step 1 and then a bogus signature in Step 3, the only possibility to discover this attack for a sensor is to verify both the certificate and the signed challenge. As the signature verification takes up very much energy and time, the sensor nodes would run out of energy rather quickly.

To mitigate this attack, some redundancy could be added to user's certificate, such that the sensor could discover the attack immediately after certificate verification. However, this is a very weak defense: The adversary could eavesdrop on the first protocol step of a legitimate user and then replay the valid certificate to sensors in an arbitrary part of the WSN.

Such an attack would deny access to a legitimate user, but would not give the adversary unauthorized data access. Perhaps the most promising procedure for defense would be by combining technical and procedural means. The gain to the adversary in service denial to legitimate users should be reduced, e.g., by data replication techniques. On the other hand, measures for quick response should be developed, such as quick deployment of additional sensors in the affected area and notification of the WSN about the occurred attacks, such that new attacks could be quickly discovered by the forewarned sensors and reported to the base station.

5. RELATED WORK

In the last few years, quite a number of papers considered implementing efficient public key cryptography on small devices. Most relevant for the WSNs are [7, 9, 17]. Most related to our work is the EccM library [9] which we used in our implementation.

As for advanced cryptographic protocols, D. Malan et al. implemented Diffie-Hellman key exchange with elliptic curve cryptography on MICA2 motes [9]. R. Watro et al. [17] implemented an RSA-based authentication protocol on MICA2 motes where the user authenticates to each sensor node separately in order to access sensor's local readings. And finally, after our paper was submitted, a great breakthrough

was reported by Gupta et al. [6]. They implemented the SSL protocol on MICA2 motes using elliptic curve cryptography. This confirms choice of ECC for our implementation and the feasibility of mutually authenticated key establishment on sensor nodes, as their SSL handshake takes less than 4 s.

However, neither of above papers considers building an access control mechanism which can withstand node capture. To our knowledge, we are the first to consider authenticated querying in WSNs in presence of node capture attacks.

6. CONCLUSIONS AND OUTLOOK

We implemented a user authentication protocol which is robust against node capture attacks. The protocol is based on elliptic curve cryptography and utilizes redundancy to withstand node capture. This protocol is our first step towards realizing authenticated querying in WSNs. This means that whenever the WSN processes a query, sensor nodes should be able to verify that the query comes from a legitimate user. As there are potentially many users in the WSN, we use public key cryptography for user authentication, as it scales much better than symmetric cryptography approaches. The basic idea is to use sensors in user's proximity as interpreter between the "public key crypto world" of the user and the "symmetric crypto world" of the WSN.

The protocol execution time of several minutes is disappointing, which is mainly due to long execution times of elliptic curve cryptography routines in EccM. EccM needs 34 sec for one point multiplication on MICA2 motes [4], which accounts for long execution time for signature generation and verification in our protocol. However, in [6] execution time of less than 4 s for full SSL handshake is reported (in assembly language). Also in *nesC* more efficient execution times are now being implemented according to [9] and to personal communication with E.-O. Blaß (see also [3]). Therefore, our ideas are becoming applicable.

Acknowledgments

We thank one of the anonymous reviewers for detailed analysis of our approach which greatly helped to improve the paper.

7. REFERENCES

[1] Z. Benenson. Authenticated queries in sensor networks. (in submission).

[2] Z. Benenson, F. C. Gärtner, and D. Kesdogan. An algorithmic framework for robust access control in wireless sensor networks. In *Second European Workshop on Wireless Sensor Networks (EWSN)*, January 2005.

[3] E.-O. Blaß and M. Zitterbart. Towards acceptable public-key encryption in sensor networks. In *The 2nd International Workshop on Ubiquitous Computing*, May 2005. to appear.

[4] Crossbow, Inc. MICA2 data sheet. Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[5] G. Gaubatz, J.-P. Kaps, and B. Sunar. Public key cryptography in sensor networks - revisited. In *ESAS*, pages 2–18, 2004.

[6] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. C. Shantz. Sizzle: A standards-based end-to-end security architecture for the embedded internet. In *Third IEEE International Conference on Pervasive Computing and Communication (PerCom 2005)*, Kauai, March 2005.

[7] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *CHES2004*, volume 3156 of *LNCS*, 2004.

[8] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61. ACM Press, 2003.

[9] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *First IEEE International Conference on Sensor and Ad Hoc Communications and Network*, Santa Clara, California, October 2004.

[10] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, FL, 1997.

[11] Moteiv, Inc. Telos revision B datasheet. Available at http://www.moteiv.com/products/docs/telos-revb-datasheet.pdf.

[12] J. Polastre and A. Broad. Module for Chipcon 2420 series radio in TinyOS. [Source code.] Available at http://www.tinyos.net/tinyos-1.x/tos/lib/CC2420Radio/.

[13] 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*, pages 95–107, New York, NY, USA, 2004. ACM Press.

[14] M. Rosing. *Implementing elliptic curve cryptography*. Manning Publications Co., Greenwich, CT, USA, 1999.

[15] Secure Hash Standard. FIPS PUB 180-1, April 1995. Available at http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[16] TinyOS web site, http://www.tinyos.net/.

[17] R. Watro, D. Kong, S. fen Cuti, C. Gardiner, C. Lynn, and P. Kruus. TinyPK: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 59–64. ACM Press, 2004.

[18] A. Woo, S. Madden, and R. Govindan. Networking support for query processing in sensor networks. *Commun. ACM*, 47(6):47–52, 2004.

[19] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.

Session 5: Short presentations

Using REWARD to detect team black-hole attacks in wireless sensor networks

Zdravko Karakehayov
University of Southern Denmark
Mads Clausen Institute
Grundtvigs Alle 150, DK-6400
Sønderborg, Denmark
Tel. +45 6550 1696
E-mail: zdravko@mci.sdu.dk

ABSTRACT
This paper describes REWARD, a novel routing algorithm for wireless sensor networks. The algorithm is adjustable and can wage counter attacks against either single black holes or teams of malicious nodes. The proposed routing technique is suitable for network nodes that can tune their transmit power. REWARD forwards packets using geographic routing. The algorithm utilizes two types of broadcast messages, MISS and SAMBA, to organize a distributed data base for detected black hole attacks. MISS helps to identify malicious nodes working in the ID space. SAMBA is related to the physical space and provides locations of detected black hole attacks. If a malicious node acts on behalf of another node, SAMBA messages will decline its efficiency. Inevitably, security overhead requires additional energy drawn from the batteries. REWARD allows to strike the balance between security capability and lifetime performance. The method has different levels of security which can be set according to the local conditions. Finally, the paper analyzes the energy overhead associated with different REWARD modifications.

Categories and Subject Descriptors
C.2.0 [**General**]: Security and protection. C.2.1 [**Network Architecture and Design**]: Wireless communication. C.2.2 [**Network Protocols**]: Routing protocols.

General Terms
Design, Security.

Keywords
Distributed sensor networks, Secure routing, Low-power design

1. INTRODUCTION
A special class networks, termed distributed sensor networks (DSN), are characterized by requirements for small size of the nodes and sufficient lifetime performance. Distributed sensor networks can be alternatively labeled mobile ad-hoc networks (MANET). While the term DSN is associated with data acquisition applications, MANET emphasizes mobility and the lack of infrastructure. The interaction between the nodes is based on wireless communication. Wireless sensor networks (WSN) is yet another synonym. To improve the power efficiency, WSNs process data within the network wherever possible. Also, the energy consumption is reduced by multihop communication.

2. FUNCTIONALITY AND SECURITY
Wireless sensor networks are integrated part of numerous applications which demand security capability. Monitoring and management of troops and weapons, surveillance, protection, urban warfare and rescue missions are fairly common security and defence applications [1].

Availability emerges as a top-priority security requirement. A proper implementation has two parts: a prompt deployment and a constant ability to sense the environment and forward traffic. In the traditional computer security, secrecy is associated with controlling who gets to read information. In the field of distributed sensor networks, the network itself may act as an intruder. In this case, the size of the nodes becomes an important design metric. Short range, multihop communication is the typical course of action.

Since the sensor readings must be bound to known areas, the first task after deployment is location. For applications, such as event tracking, each node must be aware of a set of neighbor locations. The location data base allows all nodes to be turned off, but the nodes in the close vicinity of the event [2]. This method may provide a substantial power reduction for a large sensor field. However, if nodes that line the perimeter around the event, misbehave and declare a transition, it will force several other nodes to wake up and waste energy [3]. The false alarm is a form of a battery attack. Since the chance for battery replacement is practically non existent, the attack emerge as a significant threat.

When data is gathered from numerous sensors in a dense network, there is a high probability for redundancy. Data redundancy will result in unnecessary and replicated transmissions. Aggregation, based on correlated data of neighboring nodes, helps to reduce the total volume to be routed [4]. This approach utilizes nodes to receive two or more data streams and then aggregate them into a single stream. A drawback of aggregating data is that the network becomes more vulnerable. Nodes that route the aggregated stream are in a good position to wage a black hole attack. They can simply consume the packets [5, 6, 7].

The free movement of nodes results in a dynamic topology. The routing protocols for wireless sensor networks must have a sufficient capacity to adapt to changing conditions [8, 9, 10]. The protocols can be broken down into three styles: topology-based, position-based and hybrid. The topology-based algorithms can be further split into table-driven and demand-driven. The main idea behind the table-driven protocols is to create a clear picture of all

available routes from each node to every other node in the network. In contrast, the demand-driven algorithms create routes only when a necessity arises. The actual routing take place after a route discovery procedure.

Another axis along which routing protocols are classified relates to node positions. While in this case additional information is required, the physical position of the nodes, an essential advantage is the possibility to forward packets without pre-established routes. Finally, hybrid schemes, such as Grid, employ routing in both ID and physical space [11, 12]. Grid's functionality includes three major tasks: selecting location servers, providing geographic positions and actual routing. Each node recruits a small set of nodes to keep track of its location. These nodes are called location servers. The density of location servers is decreased with the distance. To achieve this, the algorithm selects a server from squares of increasing size. When a node must be accessed, it will be sufficient to reach one of its location servers and to obtain the required geographic position.

3. RADIO COMMUNICATION

There are two possibilities for communication in the field of mobile ad-hoc networks: radio-frequency radiation and optical communication [13, 14 15]. Currently, radio communication dominates the realm of wireless sensor networks. At the radio level all packets are broadcast. This feature of the inter-node radio behavior can be used to take advantage in two directions. First, routing can be improved if instead of choosing a single route ahead of time, the path through the network is determined based on which nodes receive each transmission [16]. Second, security can be improved if nodes listen to their neighbor transmissions to detect black-hole attacks [17].

4. REWARD

REWARD is a routing method that allows a wireless ad-hoc network to organize a distributed data base for detected black hole attacks. The data base keeps records for suspicious nodes and areas. Routing in a dense network would allow alternative paths to avoid suspicious nodes and areas. The algorithm utilizes two types of broadcast messages, MISS and SAMBA. The MISS message is not related to a specific protocol and can be used after any route discovery procedure. In contrast, nodes are capable of providing suspicious locations via SAMBA messages only if they apply REWARD as a routing algorithm.

4.1 MISS message

Assume that a demand-driven protocol performs a route discovery procedure. When the destination receives the query, it sends its location back and waits for a packet. If the packet does not arrive within a specified period of time, the destination node broadcasts a MISS (material for intersection of suspicious sets) message. The destination copies the list of all involved nodes from the query to the MISS message. Since the reason for not receiving the packet is most likely a black-hole attack, all nodes listed in the MISS message are under suspicion. Nodes collect MISS messages and intersect them to detect misbehaving participants in the routes. Another reason may be a collision of packets, however if a single central station controls a sensor network, a proper organization will alleviate this problem. In a dense network, the suspicious nodes can be avoided. A more gradual approach is to introduce ratings for the nodes and calculate a path metric by averaging the node ratings in the path [17]. If there are multiple paths to the same destination, the path with the highest metric is selected.

4.2 REWARD with replication

The initial idea for REWARD (receive, watch, redirect) was associated with replication. Figure 1 shows the sequence of multihop transmissions under REWARD.

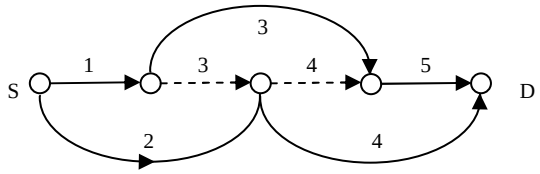

Figure 1. Two identical packets are sent to the destination.

After five transmissions the destination receives two packets with identical data. Each node's transmissions are directed to both immediate neighbors, one node forward and one node backward. If a node attempts a black-hole attack and drops a package, it will be detected by the next node in the path. The watcher waits for a predefined time period, transmits the packet changing the path and broadcasts a SAMBA (suspicious area, mark a black-hole attack) message. The SAMBA message provides the location of the black-hole attack. In order to limit the flooding to the nodes located in a close vicinity of the malicious node, SAMBA has a counter which is decremented at each node before retransmission. When the counter expires, the retransmission is terminated. Consequently, a shell of nodes around the malicious node will either avoid the area or use REWARD to go through.

4.3 REWARD against a single malicious node

The simplest version of REWARD is very close to standard routing. Figure 2 shows an example. Each node tunes the transmit power to reach both immediate neighbors. The nodes transmit packets and watch if the packets are forwarded. If a malicious node does not act as a forwarder, the previous node in the path will broadcast a SAMBA message.

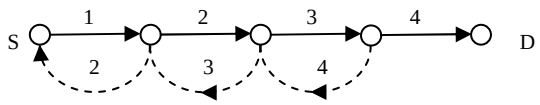

Figure 2. Transmissions must be received by both neighbors.

4.4 REWARD against a team of black holes

REWARD is a scalable method capable of waging counter attacks against a different number of black holes. Figure 3 shows an example routing with the assumption that a team of two malicious nodes would attempt a black hole attack. In this case the algorithm requires the nodes to listen for two retransmissions. Since each transmission must be received by two nodes backward in the path, the transmit power must be increased. In a general setting of nodes, the transmit power is more likely to be increased as well.

Figure 4 indicates the exact positions of the black holes in the path. The first malicious node forwards the packet using the required transmit power to deceive two nodes backward. The second malicious node drops the packet, however the attack is detected by the last node before the black holes. The missing transmission is shown by a dot line in Figure 4. An extra black hole in the path would mask the attack.

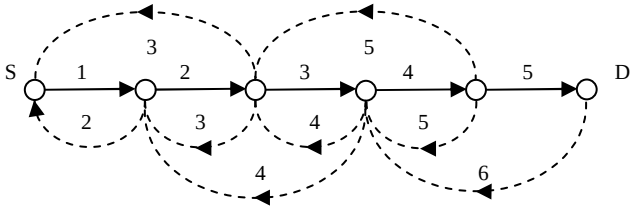

Figure 3. REWARD against two black holes.

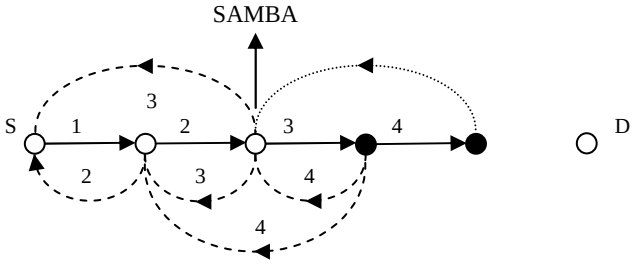

Figure 4. REWARD detects the second black hole.

4.5 Routing through a suspicious area

The nodes that line the parameter around a suspicious area decide if a packet should bypass the area or cross it. For example, if a destination is located in that particular area, the route can not bypass it.

Figure 5 shows how REWARD adapts to the current conditions. Based on the next hop location, a security level is set. The source sends the packet with L0 security level. REWARD is not active. The next node sets the security level to L1 which activates REWARD to detect single black holes. The following node increases the level to L2. As a result, REWARD is capable of detecting a team of two malicious nodes. Leaving the suspicious area, the routing method declines the security level gradually.

5. ENERGY OVERHEAD

REWARD declines the network's vulnerability at the expense of more energy drawn from the batteries of the involved nodes. Communication between two nodes requires creating a physical link between two radios. The energy used to send a bit over a distance d may be written as

$$E = A \times d^n \qquad (1)$$

where A is a proportionality constant and n depends on the environment [18, 19, 20, 21].

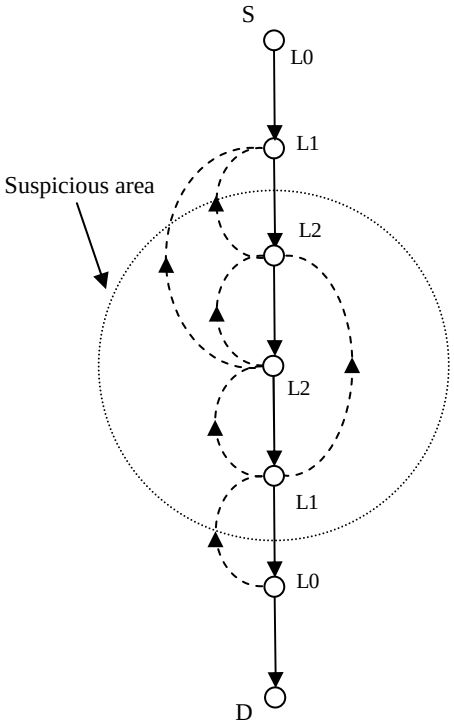

Figure 5. Using different levels of security capability.

The greater-than-linear relationship between energy and distance promises to reduce the energy cost when the link is partitioned. Rewrite Equation (1) for NH number of hops [21]. Also, include the energy for receiving E_R and energy for computation E_C.

$$E = A(\frac{d}{NH})^n NH + (E_R + E_C)NH \qquad (2)$$

Under REWARD nodes must receive more than one copy of a single packet which increases the receiver consumption. The energy for computation depends also of the processing required to compare a sequence of received packets.

The energy has a minimum for

$$NH_{OPT} = d \times \sqrt[n]{\frac{A(n-1)}{E_R + E_C}} \qquad (3)$$

When a destination receives a query, it checks if the route includes nodes under suspicion. These nodes are excluded from the route if suitable replacements are available in the destination's data base. Then the destination checks if it can minimize the energy associated with the route. The node compares the number of hops in the query with the optimal number given by Equation (3). In a dense network, the destination should be able to add or remove nodes to obtain a close match to the optimal number of hops.

Each bit routed by a node under REWARD with replication requires energy

$$E = A(2d)^n + 2E_R + E_C \qquad (4)$$

The routing per bit expense under REWARD-1 (L1 security level) is characterized by

$$E = A \times d^n + 2E_R + E_C \qquad (5)$$

REWARD capable of detecting attacks organized by M black holes requires energy described by the following equation.

$$E = A(M \times d)^n + (M+1)E_R + M \times E_C \qquad (6)$$

Figure 6 shows plots for the energy per bit for different distances between neighbor nodes. REWARD-1 indicates the energy when the routing is set to L1 level. REWARD-2 demonstrates the energy cost when teams of two malicious nodes are expected. The calculations are based on A=0.2 fJ/m^4, n=4, E_R = 50 pJ and E_C = 100 pJ.

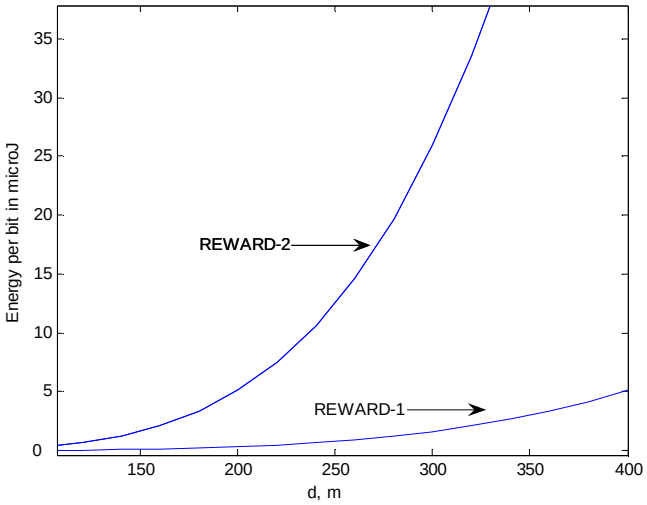

Figure 6. Plots for the energy per bit, levels L1 and L2.

Figure 7 shows a mesh plot for the energy per bit, security level between L1 and L5.

REWARD is more suitable for dense networks where it is easier to find neighbor nodes scaling the distance gradually.

6. CONCLUSION

In this paper we presented REWARD, a routing algorithm for wireless sensor networks. REWARD take advantage of the broadcast inter-radio behavior to watch neighbor transmissions and detect black hole attacks. As soon as network nodes misbehave, the method begins to create a distributed data base which includes suspicious nodes and areas. The sets of suspicious nodes emerge locally by intersecting MISS messages. The locations of detected black hole attacks are broadcast via SAMBA messages. Using both the ID space and the physical space, the algorithm makes more difficult a malicious node to act on behalf of an uncompromised node.

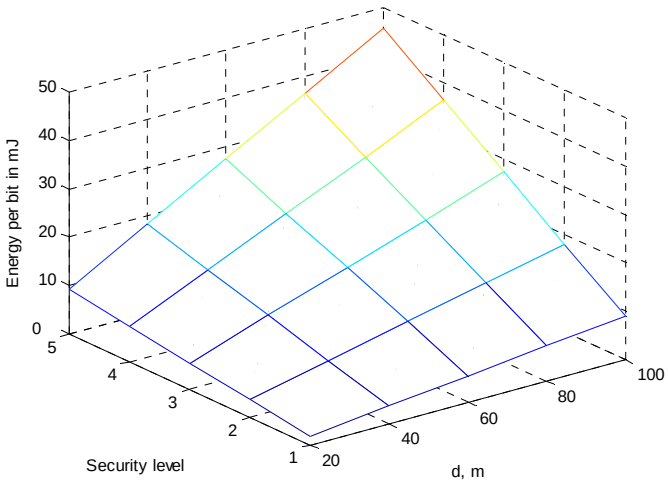

Figure 7. Mesh plot for the energy per bit.

REWARD allows to strike the balance between security capability and lifetime performance. The method has different levels of security which can be set according to the local conditions. Finally, the paper analyzes the energy overhead associated with different REWARD modifications.

7. REFERENCES

[1] Haenggi M. Opportunities and challenges in wireless sensor networks, in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, edited by Mohammad Ilyas and Imad Mahgoub, CRC Press LLC, 2005.

[2] Liu J., Cheung P., Guibas L. and Zhao, F. A dual-space approach to tracking and sensor management in wireless sensor networks, Palo Alto Research Center Technical Report P2002-10077, 2002. Available at www2.parc.com/spl/projects/cosense/pub/dualspace.pdf.

[3] Karakehayov Z. Design of distributed sensor networks for security and defense, In *Proceedings of the NATO Advanced Research Workshop on Cyberspace Security and Defense: Research Issues*, (Gdansk, Poland, September 6-9, 2004). Springer, 2005.

[4] Karakehayov Z. Low-power communication for wireless ad hoc networks, *Proceedings ELECTRONICS'2003 International Conference*, Sozopol, 2003, 77-82.

[5] Deng H., Li W. and Agrawal D. P. Routing security in wireless ad hoc networks, *IEEE Communications Magazine*, October, 2002, 70-75.

[6] Hu Y. C. and Perrig, A. A survey of secure wireless ad hoc routing, *IEEE Security & Privacy*, May/June, 2004, 28-39.

[7] Wood A. D. and Stankovic J. A. "A taxonomy for denial-of-service attacks in wireless sensor networks", in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, edited by Mohammad Ilyas and Imad Mahgoub, CRC Press LLC, 2005.

[8] Royer E. M. and Toh C. K., A review of current routing protocols for ad hoc mobile wireless networks, *IEEE Personal Communications*, April, 1999, 46-55.

[9] Mauve M. and Widmer J. A survey on position-based routing in mobile ad hoc networks, *IEEE Network*, November/December, 2001, 30-39.

[10] Hong X., Xu K. and Gerla M. Scalable routing protocols for mobile ad hoc networks, *IEEE Network*, July/August, 2002, 11-21.

[11] Li J., Jannotti J., De Couto D. S. J., Karger D. R. and Morris R. A scalable location service for geographic ad hoc routing, *Proc. ACM/IEEE MobiCom*, August 2000, 120-130.

[12] Chen B., Jamieson K., Balakrishnan H. and Morris R. An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, *ACM Wireless Networks Journal*, vol. 8, Number 5, 2002, 481-494.

[13] Warneke B., Last M., Liebowitz B., and Pister K. S. J., Smart Dust: communicating with a cubic-millimeter computer, *IEEE Computer*, January, 2001, 44-51.

[14] Warneke B. Miniaturizing sensor networks with MEMS, in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, edited by Mohammad Ilyas and Imad Mahgoub, CRC Press LLC, 2005.

[15] Karakehayov Z. Zero-power design for Smart Dust networks, *Proceedings 1st IEEE International Conference on Intelligent Systems*, Varna, 2002, 302-305.

[16] Biswas S. and Morris R. Opportunistic routing in multi-hop wireless networks, *ACM SIGCOMM Computer Communication Review*, Vol. 34, Issue 1, January, 2004, 69-74.

[17] Marti S., Giuli T. J., Lai K. and Baker M. Mitigating routing misbehavior in mobile ad hoc networks, In *Proceedings 6th Int. Conference Mobile Computing Networking (MOBICOM-00)*, New York, August, 2000, ACM Press, 255-265.

[18] Sohrabi K., *On Low Power Self Organizing Sensor Networks*, Ph. D. theses, University of California, Los Angeles, 2000.

[19] Gao J. L. *Energy Efficient Routing for Wireless Sensor Networks*, Ph. D. theses, University of California, Los Angeles, 2000.

[20] Rabaey J. M., Ammer M. J., Silva J. L., Patel D. and Roundy S. PicoRadio supports ad hoc ultra-low power wireless networking, *IEEE Computer*, Vol. 33, July, 2000, 42-48.

[21] Karakehayov Z. Low-power design for Smart Dust networks, in *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*, edited by Mohammad Ilyas and Imad Mahgoub, CRC Press LLC, 2005.

Sensor Networks and the Food Industry

Martin Connolly
CEO & Technical Director,
Sykoinia Ltd
Cork, Ireland
Tel: +353-87-2224378
martin_connolly@sykoinia.com

Fergus O'Reilly
Centre for Adaptive Wireless Systems
Cork Institute of Technology
Cork, Ireland
Tel: +353-21-4326342
foreilly@cit.ie

ABSTRACT
Over the past decade there have been a myriad of scandals and incidents involving contamination of the Food Supply Chain which has drained consumer confidence and has had untold financial, political and health costs. Despite this, much of the processes in the Food Industry are still manual. This paper examines the role that Sensor Networks can play in providing the industry with an automated software solution that will help prevent repeats of the contamination of the food supply chain that is occurring on a seemingly regular basis.

1. THE FOOD INDUSTRY AND ITS SCANDALS
Over the past decade there have been several incidents that have shaken consumers' faith in and perception of the Food Industry. In Europe, BSE is probably the most well known scandal to affect consumer confidence. Unfortunately however, this is one of many incidents that have occurred. Each incident has impacts in terms of health, financial and even political costs.

European Union countries have suffered several high profile scandals in recent years. The presence of cancer-causing dioxins in farm animal feed led to the banning of pork, beef, chicken and egg products by the Belgian authorities in 1999 [1]. This scandal cost the country's farm industry €1 billion and contributed to the electoral defeat of the outgoing government. In 2002, an Irish Oyster Farm was forced to close after consumers in Hong Kong suffered from the Norwalk-like virus [2]. This was traced to the oysters feeding on a sewage outflow containing industrial waste from a local hospital that had suffered an outbreak of what is commonly termed the 'Winter Vomiting Bug' (a loose term embracing viruses classified in the "Norwalk-like virus" genus of the family Caliciviridae).

These incidents are not limited to food producers. Retailers and distributors have also been indicted in a number of incidents. In 2002 NorgesGruppen, the leading Norwegian food service conglomerate, attracted unwanted publicity when it was found that 5 of its Meny supermarkets were selling meat well past the expiration date. Indeed, one of these shops was expelled from the chain after health inspectors found that staff was making forcemeat from expired products.

These and other incidents have had a critical influence upon consumers.

Table 1: Food Industry Scandals

Year	Incident	Loc.	Industry
2005	160 ill after eating in a UK restaurant.	UK	Retail
2005	Sudan 1 Cancer-linked colorant in Food Chain.	UK	Food Proc.
2004	Fake infant formula causes baby deaths	China	Food Proc.
2004	Raw meat found to be injected with water and additives to retain moisture.	UK	Food Proc.
2003	Fish found to be laced with mercury and PCBs	USA	Fishing
2002	Cheap and standard meats mixed and sold at premium prices.	Japan	Food Proc.
2002	Altered expiry dates on butter products.	Japan	Food Proc.
2002	Wheat containing cancer-linked nitrofen herbicide fed to chickens in organic farms	Germ.	Farming
2002	Eggs contaminated with salmonella	UK	Farming
2002	Oysters contaminated, 'Norwalk-like' virus.	Ireland	Fishing
2002	Expired Meat Products sold to consumers.	Norw.	Retail
2001	GM corn found in Taco Bell products.	USA	Food Proc.
2000	Mass food poisoning caused by bacteria-ridden milk produced in unhygienic conditions	Japan	Food Proc.
2000	Kellogg plant closes down, as it could not source corn guaranteed to be GM free grain.	USA	Food Proc.
1999	Illegal steroid hormones found in animal feed, meat and soft drinks.	Europe	Farming Food Proc.
1999	Dioxins in feed for poultry & pigs.	Belg.	Farming
1991	Food poisoning outbreak from consumption in a Fast Food Chain linked to kidney failure.	UK	Retail
1990	BSE/CJD ('Mad Cow Disease')	Europe	Farming
1985	Wine laced with antifreeze	Aus.	Wine

Table 1 illustrates that the examples cited above are just a sample of the many events that have shaken consumer confidence.The consumer group is now much more conscious of the product they purchase and consume and even companies and restaurants not directly implicated by scandal have been forced to adjust to reflect the new reality of the health-conscious consumer. Witness the recent marketing campaign by McDonalds to re-brand their image as a provider of 'healthy' meals.

Food Authorities throughout the world have responded to the growing concern among consumers and lack of legal framework for the Food Industry. The European Union has introduced much legislation in recent years regarding food product traceability, the maximum permitted levels of colorants and dioxins and food hygiene[3]. It should be noted that many of these directives apply throughout the food chain i.e. growing, production, transports and distribution.

However, there is still much work to be done as evidenced by the recent recall of products containing the carcinogenic Sudan Red 1 food colorant[4]. From the perspective of those in the food industry there is a requirement for the timely supply of information. This information includes temperature, the presence of agents and airflow. This paper addresses the potential for Sensor Networks to aid in the provision of a software solution.

While the incidents in Table 1 can be attributed to a variety of factors including human error, carelessness, irresponsibility and fraud, the incidents could n future be prevented or managed through the judicious use of wireless intelligent sensor technology. For those affected by the careless or irresponsible actions of participants in the food chain the deployment of a sensors that can detect pathogens, bacteria and other potentially harmful agents. On the other hand, those who act less than ethically can be policed more effectively by health and food safety inspectors through the ad-hoc set-up of an intelligent sensor network during site visits.

2. TECHNOLOGY IN THE FOOD INDUSTRY

Woodworth[5] outlines how the Food industry is generally receptive to the use of information systems. Traditionally the principal use of Information Technology by Food Processing companies has been in Enterprise Resource Planning (ERP) Software. However, in recent times companies have started using mobile devices such as PDAs and wireless technology such as 802.11 to provide solutions in areas as diverse as traceability and logistics. Indeed, the incidents outlined in the previous section have led to increasing interest in Product Life Cycle Management Software which, in conjunction with RFID (Radio Frequency Identification Technology), is being mooted as a solution for addressing food safety [6].

As pointed out in Friend[7], ERP software is designed for managing operations within the enterprise only. However, given a typical food processor's dependencies on external suppliers and the external environment this is no longer adequate for today's food industry. Indeed, many of the key tasks in food production are still carried out manually, For example, growers still manually measure temperature and food scientists often have to manually carry out tests to detect ingredient composition and check for the presence of contaminants. Mobile technology is one factor in providing an automated software solution for the food industry but the solutions offered today only cater for the tagging and labelling of products. Granted, these are key components of a good traceability system and are vital for the post-auditing of food contaminations but they don't prevent the actual problem. It is in addressing that intelligent sensors can provide a vital role.

3. SMART SENSOR NETWORKS

The potential applications for Smart Sensor Networks is diverse with current uses ranging from vineyards to equipment maintenance. The communication potential for Smart Sensor Networks with external systems is cited in Hac[8] where Bluetooth is proposed as a possible communication standard between Sensors. Given the protocol's suitability for smaller electronics products such as mobile phones and digital camera one would expect it to be a feasible option. The Zigbee Wireless Technology has also been proposed for intra-sensor communication.

External interfaces such as those provided by Crossbow [10] enable the Sensors to communicate with electronic devices such as PCs, Laptops and PDAs. Once the data reaches these devices it can literally be transmitted to any network. Support is provided for detecting this data transmission through the Java API provided by the TinyOS open source forum [11].

Sensor Networks are recognised as one of the fastest growing segments in the technology industry. Part of the reason for this is their wide applicability. Good[12] illustrates the many uses of sensor networks for security purposes while Harbor Research estimates that the market size could be US$1 billion by 2009[13].

4. SMART SENSOR NETWORKS & THE FOOD INDUSTRY

Good[12] raises an interesting point in its illustration of the use of sensors for detecting biological and chemical agents. While security is certainly a pressing issue and an obvious application for sensor networks the technology can also be put to other uses, in particular the detection of contaminants in the Food Supply Chain. The 1999 Belgian crisis could certainly have been averted if technology existed for the detection of dioxins in the production of animal feed.

As is clear from a perusal of EU sources[3], the EU has imposed heavy duties upon food manufacturers. One key requirement for manufacturers is Regulation (EC) No. 178/2002 requiring manufacturers to keep detailed records of their supply chain. Despite the fact that this legislation has been effective since the 1st January 2005 the manual processes still practiced by many companies has made it difficult for them to comply with this legislation.

Among the requirements are labelling and barcodes for product batches so that 'problem' batches can be easily recalled. Not only is this good for the consumer it is also good for the producer and retailer as products can be recalled in a more timely fashion. This is a key point the legislation does not mandate the publicising of product recalls if this process occurs before the product enters the consumer market so companies can avoid the attendant adverse publicity and damage to their reputation.

However, there are issues with labels and barcodes, which can reduce their effectiveness. Barcodes and labels are line of sight only and are unreadable if they are damaged or soiled. The emerging solution to this problem is the use of Radio Frequency Identification (RFID) tags. RFID tags are essentially sensors that contain unique ID for a product or product batch.

The tags are long range and since they are a radio technology can be embedded in plastic. Of course, since they are Sensors they can form a part of an overall Sensor Network. Roberti[14] demonstrates how the US Navy used RFID as part of an overall Sensor Network to monitor pressure, temperature and humidity in shipping containers for aircraft parts.

The point raised in Roberti[14] can also be applied to the Food Industry. Many products require constant monitoring throughout their supply chain. This is not only for compliance purposes but also to ensure that the basic quality assurance requirements to retailers are met. For example, the chill chain requires that chilled products are stored at a constant temperature throughout the delivery and storage of the product. Typically each product batch is examined by the retailer on arrival and is rejected if it does not meet the required temperature. Up to now, the only solution to this problem for many smaller companies was to take periodic manual temperature readings. With the advent of Sensor Networks it is now possible to devise a solution whereby the producer is notified virtually instantaneously if the temperature failed outside acceptable parameters.

5. SENSOR NETWORKS & FOOD GROWERS

Of course, food processing companies are only one segment of the Food Industry. Growers also have requirements that can be met by sensor networks. The most well know example is that of the 'smart vineyard'. The Australian Cooperative Research Centre (CRC) for micro technology and Motorola have devised a Sensor Network solution for the Australian Wine Industry[15]. The devices used in this project can measure wind speed and direction, temperature, light, humidity, soil moisture and leaf wetness. Given that weather conditions and the anticipation of same is a key factor in the production of quality wine it is obvious that the use of a Sensor Network can be of great benefit to vine growers.

Such a system also has the potential to give vine growers better information for monitoring the crop and can ultimately play a role in anticipating problems such as the presence of pests such as phylloxera, an insect that sucks fluid from the grape vine which results in the rotting of the plant.

The ubiquitous nature of the radio transmission of Sensor Networks is also important here. By its nature agricultural land can vary widely and the same applies to vineyards. The Sensors can be deployed virtually anywhere in a vineyard as long as they are in requisite range of each other. The other key advantage here is cost. Traditionally climate sensors are expensive which precludes their use from most growers and even then only on a limited scale.

The above characteristics of sensor networks are not just useful to vine growers. Any fruit grower will have similar requirements, as indeed will growers of cereals, vegetables and sugar beet. Once again the potential use of Sensor Networks is wide ranging.

Monitoring of crops is at present performed manually and on a somewhat ad-hoc basis. At times, this is a very onerous task given the very specific requirements of a crop. Mushrooms growers, for example, have very specific requirements in terms of light, temperature and airflow and have to manually measure these attributes. The deployment of a Smart Sensor Network would greatly increase the efficiency of this process.

6. CASE STUDIES

6.1 The Wine Industry

One of the most popular early deployments of Sensor Networks is in the Wine Production Industry[15]. However, there are other successful examples in different parts of the world, such as how Pickberry Vineyard in Sonoma, California was able to manage its crop growing problems through the use of a Sensor Network provided by Accenture Labs[16]. From the perspective of vine growing sensors can be used to monitor soil moisture, rainfall, wind velocity and direction, and air and soil temperature. The monitoring of these factors can be critical in the cost management for a vineyard. For example, detecting frost in a timely manner can prevent loss of a crop, all the more important since, to take the Pickberry example, each ton of grapes can be worth US$1000 to US$4000.

However, while all the focus is currently on using Sensor Networks for the growing of grapes this is merely one of the stages in wine production for which intelligent sensors can play a role. For example, temperature must be strictly controlled during the vinification process (i.e. the conversion of grape juice into wine). This is so that yeasts attached to the grape can feed on the sugar to produce alcohol – for white wine the temperature must be between 15°C to 20°C for fermentation, for red wine fermentation requires a temperature of between 25°C and 30°C. Furthermore, for this controlled fermentation process the presence of Sulphur Dioxide must be added but this is monitored in accordance with strict European Union guidelines.

Intelligent Sensors can clearly play a role in the fermentation process whether it is in temperature or Sulphur Dioxide monitoring. However, these two examples are just one of the many potential uses for Sensor Networks in vinification – intelligent sensors play a role in everything from detecting the presence of malic acid to tannins.

The final step in the production process of course is storage. Once again sensors can be used here to detect levels of humidity and temperature in a cellar. Randomly deployed sensors could also be potentially be used to detect what is commonly termed 'corkage' (air contact with the wine due to inadequate corking) in wine bottles or batches.

6.2 The Chill Chain

One of the greatest challenges for the Food Industry is for the manufacturers of frozen foods, in particular meat and poultry. Temperature must be maintained at a constant level from initial processing to final display by a retailer. However, given the steps from production to sale this has up to now been an

onerous though essential requirement for frozen goods producers. This term coined for this issue is the Chill Chain.

As implied above the Chill Chain consists of a number of steps. Primary chilling relates to removing the heat from the carcass before it can be further processed or shipped. However, there can be startling differences in the surface temperature of a carcass and its deep temperature[17]. Once a previously chilled produced has been cut, minced, wrapped or cooked secondary chilling must take place. This is vital for ensuring that a product remains at a constant temperature during transportation.

Studies have shown that primary and secondary chilling are vital for maintaining potential shelf life. In studies[17] it was found that vacuum packaged beef on average achieved only 25% of its potential shelf life, 2 weeks instead of 8 weeks. This is due to a number of factors including loading the meat for transport at too high a temperature and the incomplete cooling of boxed meat due to weak air movement.

There is an obvious opportunity for sensor networks in primary and secondary chilling, not only to monitor temperature but also to monitor airflow. Currently, despite the best intentions of manufacturers the possibility of error is very high. Not only is this a risk for the consumer but there is also much wastage if shelf life is being reduced to a quarter of its potential.

Primary and secondary chilling take place in plant but these are only two steps in the Chain. Refrigeration when transporting the goods much provide an adequate air distribution. It should be noted that the refrigeration unit does not cool the goods. Rather it maintains them at the temperature at which they were loaded. There are similar requirements for Off-site Cold Storages, Central Distribution Centres and Wholesalers and, of course, Retailers.

While primary and secondary chilling are the responsibility of the processor, essentially manufacturers have no control once the goods leave their premises particularly if they do not have their only transportation. Ultimately, any fault in the chill chain will damage a frozen goods manufacturer's reputation even if this fault is caused by a third party. However mandating the use of a sensor network to monitor temperature and air flow in the other steps in the process can play a critical role in the monitoring of food quality and can help prevent spoiled goods reaching the end consumer.

7. FOOD INDUSTRY SYSTEM REQUIREMENTS

The Food Industry involves complex processes and detailed regulation. By contrast, the typical Information Systems department wants software that is easy to use [5]. Any Sensor Network System that is targeted for the Food Industry will have to be easy to deploy. Food Processors are uninterested in configuring sensors, their interfaces and the attendant software for analysing the data. The system will literally have to work from 'out of the box'. While this is admittedly a difficult goal for any emerging technology it is critical to the success of Sensor Network Systems in the Food Industry.

However, while deployment is a challenge, smart sensors already monitor the actual measurements required. Critical measurements such a temperature, wind speed and humidity are already catered for and there is little that is new. One open research topic is the development of Sensors that can measure not only the presence of a particular ingredient or characteristic but also the quantity or percentage of that characteristic. For example, it would be useful not only to whether there is limestone in the soil but also the percentage composition of the soil that is limestone. Similarly, while the presence of dioxins can be tested with current sensor technology the *level* of dioxins in products such as milk cannot currently be tested using smart sensors. Such information is important for quality assurance programs and has to be carried out manually for the most part.

The other point of interest regarding the Food Industry is its use of relatively few and relatively generic software systems. The output from any Sensor Network System must be easily transferred to third party software such as MS Excel and SAP. Thus, any system must be open and have easy to use or even automated export functionality to other software.

8. FAILINGS OF SENSOR NETWORKS FOR THE FOOD INDUSTRY

From the authors' experience, there are a number of factors that act as serious barriers to entry for the use of Sensor Networks in the Food Industry:

- Reliability
  A recent deployment of Sensor Networks in two redwood trees in Sonoma, California found that 65% of the nodes never returned data [18]. Similar problems were recorded when using Sensor Networks for habitat monitoring[19]. The former statistic would be unacceptable in any commercially deployed food monitoring network and it would be very difficult to make a business case for a network where a significant proportion of the nodes never work.
- Ease of Use
  Food processing organisations tend to be technology agnostic. Microsoft based technology prevails as it is easy to use and install. For the non-technical user Sensor motes are intimidating to deploy and use. Indeed, implementing a Sensor Network lies outside the range of the average Food Industry Information Systems Professional.
- Data gathering and Data Interpretation
  While Sensor Networks undoubtedly can play a significant role in improving the availability of update data on a physical environment there is currently no standard way for gathering and interpreting the data. The Food Industry relies heavily on standard reporting tools such as Crystal Reports and Brio and would be loathe to use non-standard Java-based reporting GUIs.
- Organising Networks and Clustering

There is at present no mechanism by which end users can organise their sensors for reporting purposes. Granted, group IDs are available but there can be limited – for example, two sensor groups might be monitoring the same area and would be effectively the same sensor group from a reporting perspective. What is required is a means by which users can organise their groups into clusters that make sense to them.

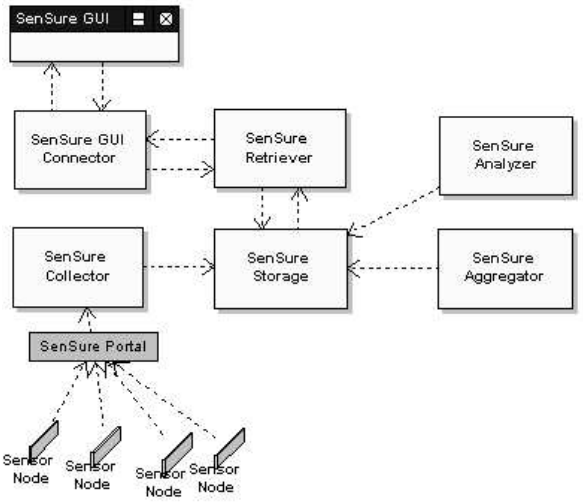

Figure 1: Sensure Sensor Portal

9. THE SYKOINIA SOLUTION

The above issues effectively preclude any commercial use of Sensor Networks in the Food Industry. Sykoinia's SenSure solution aims to bridge this significant gap in Sensor Network technology, by providing a means for end users to organise their networks, providing integrated Crystal Reports functionality and enabling users to add different types of sensors easily and quickly.

Data produced by the sensors is collated by the SenSure portal Figure 1, and is then forwarded to the server system for analysis and storage. The server system can be accessed through a user interface that can organise the Sensor Groups and can also request and schedule reports.

10. CONCLUSION

Sensor Networks have a myriad of potential uses. The Food Industry provides ample opportunities for both hardware and software providers given its need for monitoring software – a niche that a sensor network consisting of diverse smart sensors can fulfil.

However there are a number of caveats – a complex and hard to use system will not succeed in this industry nor will a 'closed' system where data is difficult to export, interpret and analyse.

11. About the Authors

Martin Connolly is the CEO and Technical Director of Sykoinia, a provider of integrated Information Systems solutions for Sensor Networks. Sykoinia's solution has been deployed in orchards, vineyards and food processing halls. For further information visit http://www.sykoinia.com.

Fergus O'Reilly is a lecturer in the Department of Electronic Engineering in Cork Institute of Technology and a Principle Investigator in the Centre for Adaptive Wireless Systems, Cork Institute of Technology. (http://www.aws.cit.ie)

12. REFERENCES

[1] van Larebeke, Nik, Hens, Luc, Schepens, Paul, Covaci, Adrian, Baeyens, Jan, Everaert, Kim, Bernheim, Jan L, Vlietinck, Robert, and De Poorter, Geert, "The Belgian PCB and Dioxin Incident of January-June 1999: Exposure Data and Potential Impact on Health, *Environmental Health Perspectives Volume 109, Number 3 March 2001*

[2] Cassidy, Eddie, "Norwalk-like Virus Outbreak in Hong Kong Forces Closure of Irish Oyster Farm*", The Irish Examiner 26/03/2002*

[3] *Official Journal of the European Union*

[4] Food Safety Authority of Ireland, FSAI Alert Notifications, http://www.fsai.ie/alerts/fa/index.asp

[5] Woodworth, Simon. "Track and Trace: An opportunity for Information Systems Deployment and Supply Chain integration", Food Traceability Seminar, NUI Cork, February 2005

[6] Sullivan, Laurie, "PLM Software Has A Role In Food Safety", *InformationWeek, Jan. 12, 2004*

[7] Friend, Bill and Thompson, Olin, "Managing Software: Is the food industry ready for Product Life Cycle Management?", *Food Industry Magazine March 2003*

[8] Hac, Anna, "Wireless Sensor Network Designs",Wiley Press 2003

[9] The ZigBee Alliance http://www.zigbee.org

[10] Crossbow Technology Inc. http://www.xbow.com

[11] Tiny OS http://www.tinyos.net/

[12] Good, Barbara G, "Sensors for Security", *Sensors Magazine, July 2004*

[13] Harbor Research, Inc http://www.harborresearch.com/

[14] Roberti, Mark, "Navy Revs Up RFID Sensors", *RFID Journal June 2004*

[15] "MEMS Come to Oz Wine Industry", *Electronic News June 2004*

[16] "Accenture Prototype helps Pickberry Vineyard improve Crop Management", http://www.accenture.com/xdoc/en/services/technology/case/pickberry.pdf

[17] Managing your Chill Chain--The Undiscovered Country, *Meat & Poultry, February 1999*

[18] Tolle, Gilman & Culler, David, Design of an Application-Cooperative Management System for Wireless Sensor Networks, *2ndEuropean Workshop on Wireless Sensor Networks (EWSN) Istanbul, Jan 31-Feb2, 2005.*

[19] Szewczyk, Robert et al., "Lessons from a Sensor Network Expedition", *1st European Workshop on Wireless Sensor Networks, Berlin, Jan 19-21, 2004.*

Experimental construction of a meeting model for smart office environments

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

ABSTRACT

The simulation of mobile networks requires mobility models that reproduce the movement of nodes in a realistic way. Although many such models exist, they are not well suited for model office scenarios in which movements are mostly caused by people meeting in known locations to discuss some issues. We present first steps towards a meeting model for office environments which is based on the real world movement data gathered after performing a one-week sensor network deployment in our department. We describe details about the processing of the acquired data, the construction, and the execution of the movement model. Finally, we discuss some of the lessons learned throughout the experiment.

1. INTRODUCTION

Many research laboratories are experimenting with sensor networks, but only a few of them have been deployed in real settings. Most such networks are not easily accessible and are deployed on islands [7], vineyards [1], or even on zebras [4]. Errors may arise due to environmental factors which are not observed by the scientist and are therefore unexplainable and irreproducible. For this reason, the department of Distributed Systems at the University of Stuttgart decided to build an in-house Smart Environment for our office using wireless sensor network technology, which is a controlled environment where experiments can be performed.

Before deploying a network, simulations are used to test the application. The quality of the simulations in mobile environments is dependent on the mobility model. However, we have found that existing mobility models do not provide realistic movements for our application. Therefore, we set up an experiment to record the movement and meeting patterns of employees in our department in order to derive information about the duration and composition of meetings. This data was used in the creation of a meeting-based movement model, which is the focus of this document.

The rest of the paper is organized as follows: In the next section we describe 'Sense-R-Us', our Smart Environment application. Section 3 presents the experiment, the processing of the data, and the design of the group meeting model. Some experiences we have gained during the experiment are also described here. Finally, section 4 concludes this paper.

2. SENSE-R-US

The 'Sense-R-Us' application aims at building a Smart Environment in our department using Mica2 motes from Crossbow Technology Inc. We decided to use Mica2 motes since they are readily available and easy to deploy. Additionally they have a low power consumption, are small enough to be carried around, and run TinyOS, so that a powerful development platform is available for testing. The application is divided into mote and PC components. Using a PC-based GUI, the position and status ('isInMeeting', e.g.) of persons can be queried. The location estimation does not have to be very accurate, so that room information is sufficient for our purposes. Other queries the system is expected to answer are "What is the temperature in room X?", "Which rooms are available?", or "How many meetings has person P already had today?".

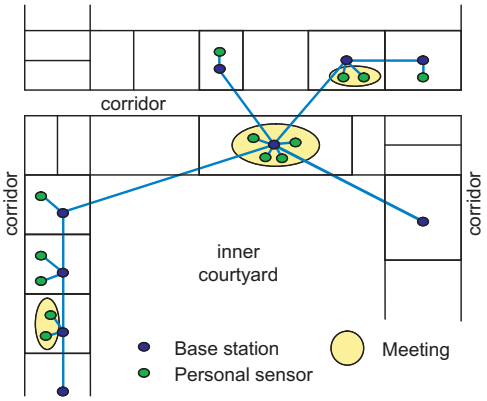

Figure 1: Floor plan with deployed Sense-R-Us

Two different types of sensor nodes are used in Sense-R-Us: static 'base stations' and mobile 'personal sensors'. Base stations are installed in all office rooms and known locations, e.g., meeting room, lab, and break area. 'Personal sensors' are carried around by the employees. Figure 1 shows the floor plan of our department. The office rooms are situated along three corridors. In the middle of the department, there is an inner courtyard. Since we share the floor with another department, not all of our office rooms are adjoining.

Base stations send out location beacons with their room id all the time. They are plugged to the power outlet and, thus, have no energy constraints. Some of the base stations are connected to the serial port of a nearby PC and — using a small Java application — to the wired LAN. Messages from the GUI running on any PC in the department are routed via LAN to the base station PCs and via radio from base station to base station following a tree-based algorithm.

Personal sensors receive the location beacons and determine their position by selecting the base station with the highest signal strength. They also send beacons which are received by other personal sensors to update their neighborhood list. This list and microphone data is used to detect the occurence of meetings. This detection runs periodically and not just when a query is to be answered since it may happen that nobody is speaking when a 'isInMeeting'-query is received. Messages received by the personal sensors, like queries from the PC-based GUI, are flooded to other personal sensors to ensure that all of these mobile nodes are reached.

Mobility Models for Simulation

During the development process of Sense-R-Us, the application was tested using TOSSIM, enhanced by a simple graph-based mobility model. Soon, we observed that the resulting movements are far from reality due to the following reasons:

Entity mobility models [2] such as Random-Waypoint regard all nodes as independent. Movements in office environments have such an independent component, for example going to the printer or to the coffee maker, but many movements are caused by meetings with colleagues, either in their office rooms or in separate conference rooms. With entity mobility models, these meetings only happen occasionally. [2] describes also group mobility models which focus on moving an entire group but not on the formation process of a group.

Secondly, the number of people involved in a meeting varies widely. There are a lot of two-party meetings, but also group meetings of 3 or more persons, and meetings in which all employees of the department are involved. Just as these meetings are planned in reality so that all persons have time to attend the meetings, it is necessary to plan them in simulation. Normal mobility models do not support such planning.

Additionally, the structure of organizations has to be reflected in the simulation. A department has a head and is divided into several groups which in turn have a group leader. Meetings between people of the same group happen more often than meetings between different groups. It is also obvious that the head of the department meets more often with the group leaders than with the other members of a group. In [3], a social mobility model is presented, but it allows the definition of pairwise relationships only. The model of [6] allows the definition of a complete "Interaction Matrix" between all people. But this matrix is not suitable to define organizational structures and meetings in these structures.

Finally, the duration of meetings varies widely and is additionally dependent on the number of people in a meeting and the composition of the meeting concerning organizational structure.

Although we had a feeling for the nature of meetings, we did not have a quantitative specification for group size, group composition, and duration of meetings. These were gained by performing the experiment explained below.

3. EXPERIMENTAL EVALUATION

The goal of the experiment was to record the locations and meetings of several persons. It should give answers to the question "who meets with whom at which location at what time for how long?".

3.1 Setup

We started by placing 10 base stations in the rooms depicted in figure 1. Although similar in topology, the difference to Sense-R-Us is that the base stations do not work as gateways to the wired LAN, but send beacons with their room ID every 10 seconds. 11 employees were involved in the experiment, carrying around personal sensors. These sensors send ID beacons every 10 seconds and receive ID beacons from base stations and other personal sensors. They do not have query processing capabilities, but save the received beacons with their signal strength and a timestamp to the serial flash. The Matchbox file system was used to provide an easy way to store the data during the experiment and to retrieve the data when the experiment was over. To be able to correlate the meeting data of different sensors, a simple time synchronization mechanism based on a master time sensor was used.

The experiment started on a Tuesday afternoon and ran one week until the next Tuesday afternoon. The personal sensors were handed out and collected in a department meeting. Afterwards, all logger data was read out from the personal sensors. Since the experiment period was short, all persons participated in the experiment without interruption.

3.2 Data Processing

Logger data from the serial flash was processed in several steps to extract the relevant information. The four processing steps to clean the data, i.e., to detect and remove false meetings, are described in the following paragraphs.

In step 1, we computed the time a node (both base stations and personal sensors) was heard from a single personal sensor. To avoid counting beacons which were recorded accidentally as meetings when two persons passed each other in the corridor, beacons had to be heard for at least 30 seconds in order to be considered part of a meeting. On the other hand, due to possible transmission errors it was necessary to tolerate missing beacons: the beacons of a node had to be missing for at least 60 seconds to be regarded as lost. The result of this processing step for one day is shown in figure 2. The bars (grey and black) indicate the intervals during which mote 11 has seen the base stations (Base 1–10) or other personal sensors (Mote 1–10).

Analysis of the intervals for base stations 1 and 2 reveals that most of the time both base stations were heard when the personal sensor was located in the room of base 1. Base 2 was actually the base station of the neighbor room. Therefore, in step 2 the strongest base station had to be selected. The intervals of other base stations that overlap with the interval of the strongest node were deleted or shortened. The results are shown as black bars in the lower part of figure 2 — the intervals of the personal sensors were not changed in this step. The grey bars indicate the intervals that were removed during this step. Base 1 is always stronger than base 2 and base 3 almost always than base 4. Therefore, all bars of base 2 have been deleted which is correct since the employee was not in the room of base 2 that day. Also all the bars of base 4 between 207351s and 213021s have been deleted since the employee actually was in the room

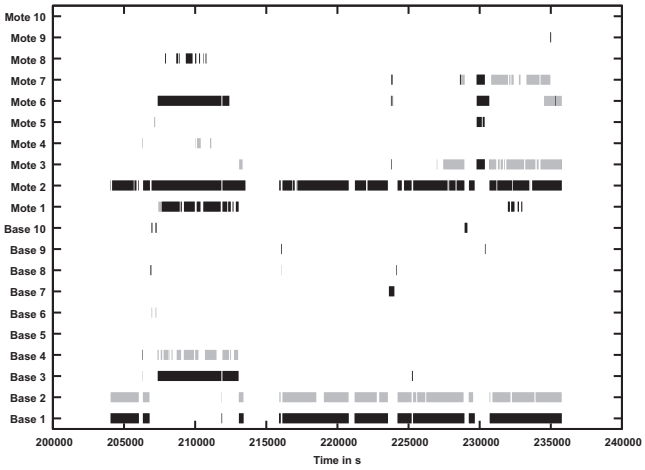

Figure 2: Results after processing step 1 (all bars) and step 4 (black bars only) for personal sensor 11

of base 3 at that time. This step was performed for each personal sensor separately. After this step, the position for each personal sensor at any point of time is unambiguous.

In a meeting of two persons, person A has to hear person B and vice versa. Likewise, based on the logger data, a meeting is only a meeting when mote A hears mote B and at the same time mote B hears mote A. Therefore, in step 3 such mutual meetings were looked for. To perform this step, it was necessary to combine the data of all personal sensors.

Step 4 is similar to step 2 for personal sensors. Since a personal sensor can receive beacons from a base station in the neighbor room, it can receive beacons from a personal sensor in the neighbor room, too. For each meeting of motes A and B, the positions of motes A and B during the meeting interval are determined. The parts of the meeting interval with contradictory positions are deleted. Without this step, a meeting of two persons would be detected although they are in different rooms. The results of steps 3 and 4 are shown as black bars in the upper part of figure 2 — the intervals of the base stations were not changed during these steps. The grey bars in the upper part were removed during this step. For example, the bars for motes 3 and 7 between 230808s and 234698s have been deleted which is consistent with reality since both employees had a meeting in the neighbor room, but not with the employee carrying mote 11.

It is not possible to proof the correctness of the described processing steps. Thereto, all the employees involved in the experiment would have had to record their movements manually which is a high effort reducing normal working time. Moreover, it is very likely that also the manual records are incomplete since sometimes the employee forgets to write the movement down. As indicated in the above paragraphs, we have checked the data processing with samples. For some meetings, we know for sure that they have or have not happened. At least for these meetings, the data processing works correctly, and it is very likely that it does for other meetings as well.

Based on consistent two-pair meetings, it is now possible to extract group meeting information. In a group meeting, all

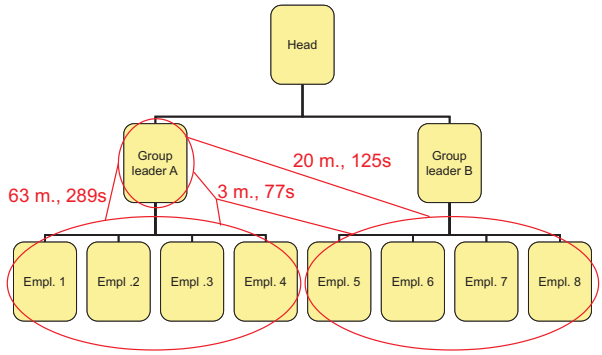

Figure 3: Meetings in organizational structure

involved persons hear each other. This can be detected by constructing a graph of all mutual meetings at a time and finding all cliques (fully connected subgraphs) in this graph. When a node joins (or leaves) an existing meeting, a new meeting starts with one more node (or one less node).

3.3 Model Construction

Using the data gathered during the experiment, we have created a model that contains the following parts: the composition of a meeting, its frequency, its duration, and its location. We show how these parts are identified from the experiment and how they are used in the model.

3.3.1 *Composition*

In companies, apart from personal relations, most meetings are due to work, and some groups cooperate more closely than other groups. These groups are defined by the organizational structure of the company. Figure 3 depicts the structure of the employees involved in the experiment. Employees 1–4 form one group, employees 5–8 another one. Group leader A and B and the head of the department are treated as separate 'one-man' groups.

For all meetings we have analyzed how many persons from which group took part in the meeting. As result, in the model 20 different compositions of meetings were defined.

3.3.2 *Frequency*

Depending on the composition of a meeting, its frequency varies. Figure 3 shows as an example the number of meetings observed during the experiment (abbreviated by 'm.') of group leader A with his own group (which means 'at least one person of group A'), with group B, and with people from both groups. As expected, most of the meetings happen within his own group. Meetings with the other group or with both groups also happen, but are less frequent. In the model, every group combination can have its own frequency to be used in the simulation. The frequencies are given in absolute numbers relative to a common time period during which the indicated number of meetings are expected to happen.

3.3.3 *Duration*

Just as the frequency depends on the composition, the duration also varies from meeting to meeting. The average duration of meetings of group leader A is annotated in figure 3. For example, meetings with his own group are longer than meetings with group B or both groups together.

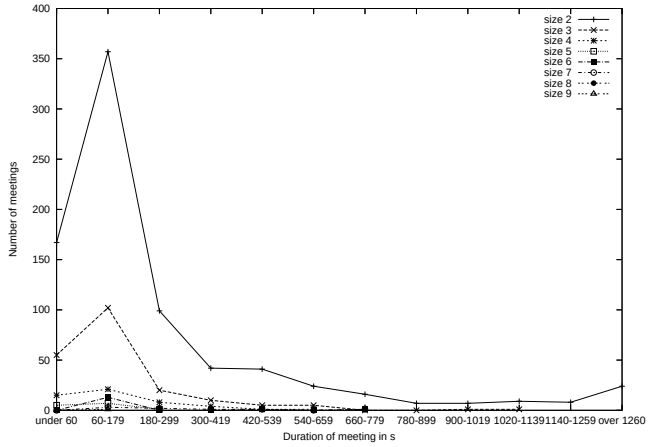

Figure 4: Histogram of duration of group meetings

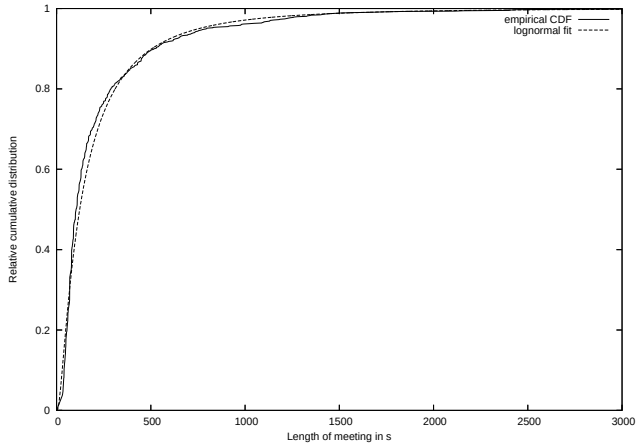

Figure 5: Empirical and calculated distribution

The duration of meetings is neither equally nor normally distributed, therefore the mere use of an average value would be misleading. In figure 4, the meetings are broken down into meeting size and duration of meeting. All meeting sizes show similar duration characteristics: Most meetings take between 1 and 3 minutes, but 19% of all meetings are longer than 5 minutes. The histogram of the duration has a high spike and a long tail to the right. The log-normal distribution provides such a density function. The average value and standard deviation can be calculated from the measured durations and are used to calibrate the distribution function.

When simulating the model, a duration has to be assigned to each meeting. Therefore, a normal distributed random number is calculated and converted to a lognormal distributed random number using the formula $random\_lognormal = e^{\sigma' random\_normal + \mu'}$ with $\sigma' = ln(\sigma)$ and $\mu' = ln(\mu)$. The result is directly used as the meeting duration. For each of the 20 combinations of meetings used, μ and σ are defined as derived from the experiment.

In figure 5, the empirical distribution of all meetings and a lognormal distribution function with $\mu' = 4.8086$ and $\sigma' = 1.1048$ are shown. The good fit is also indicated by a root mean square error of 0.7328.

3.3.4 Location

The location of a meeting depends both on the size of the meeting and on the composition. For example, meetings in which more than a certain number of persons are involved usually take place in the conference room or are informal meetings in the coffee corner. Smaller meetings with the head of the department mostly happen in his room. Usually, two-party meetings happen in the room of one of the participating persons. We do not show a detailed evaluation and the realization in the model due to space restrictions.

3.3.5 Model Execution

To schedule the meetings, the simulation time is divided into smaller sections. The length of a section is chosen so that a meeting with the same characteristics cannot start twice in a section. The meetings that happen in each section are selected randomly, but based on the frequency of the meeting composition. For each meeting, free candidates in each group participating in the meeting are selected. If there are no free candidates in each group, the meeting is postponed to the next section. Otherwise, a starting time and random persons of each group are selected for the meeting. The duration is calculated using the lognormal distributed random number described above.

Then, a location is assigned to each person for all free (i.e., 'non-meeting') intervals. Thereby, the own office has a higher probability. This assignment reflects the individual behavior of persons when they do not have a meeting. This step is necessary before assigning locations to the meetings because a meeting can only be scheduled to the office of one of the participants if the colleagues sharing the office are not there that time. Finally, the location of the meetings are set to one of the rooms of the participants, to one of the common rooms, or to an unknown location. In the last case, the movement algorithm will move the participants to a location outside of the department where no one else is meeting already. In all other cases, the algorithm will move them to the location the meeting is taking place.

3.3.6 Model Evaluation and Adaptation

When setting the simulation time to the time period during which our measurements were taken, the model generates exactly the same number of meetings for each composition. For compositions with lower frequencies, the medium duration deviates from μ given in the model due to the lognormal distribution, but when considering all meetings together, the distribution measured in the experiment is approximated.

The model is general and can be adapted to other, bigger scenarios. The values given above reflect our measurements with 11 employees. When applying to other organizations, the meetings compositions have to be derived theoretically or measured by an experiment. For each composition, the frequency and duration parameters have to be figured out as well. Finally, the office rooms of the persons and meeting locations have to be defined.

3.4 Discussion

During the analysis, we discovered several points which directly influenced the accuracy of our data.

Signal strength was sometimes inaccurate. The signal of a base station placed right behind the wall in the neigh-

bor room was sometimes stronger than the signal of a base station at the opposite corner of the same room. Thus, a personal sensor was assigned to the wrong room. Moreover, the actual meeting this person might have had has not been detected, and phantom meetings might have been added. A possible solution includes a good placement of base stations and their calibration. A base station should be installed in the center of an office room, if possible, but not at walls to adjacent rooms. The transmitting power of each installed base station should be adjusted during a calibration phase. For several positions in the office room, the minimal power is determined which is necessary to receive a personal sensor. The maximum of the minima can be used as new transmitting power of the base station.

Base stations were not installed in every room. If an employee stayed in a neighbor room without a base station, he was located in the room with the base station. Walls attenuate the signal, but as described in the last paragraph, this effect is too inaccurate to serve as a distinguishing feature. The calibration described in the last paragraph could alleviate this problem if the best solution — the installation of base stations in every room — is not feasible.

Not all employees took part in the experiment. Only two of four groups of our department were equipped with personal sensors. But certainly there have been meetings between employees with sensors and employees without sensors. These meetings have not been recorded. But since every employee of the two groups carried a sensor, we believe that the meetings between them have been detected quite accurately. Generally, the short time frame of the experiment and the small number of participants do not limit the idea of the model, but reduces only the accuracy of the calibration parameters.

The granularity of the beacon interval (10 seconds) can lead to shorter meeting intervals of almost 20 seconds. If two sensor nodes meet just after the nodes have sent the beacons the meeting interval will start at the next two beacons. If the two sensors part just before the nodes send the next beacon, the meeting interval will end at the last two beacons. This effect has a greater influence on shorter meetings, but can be ignored for long meetings. Shortening the beacon interval would result in a higher energy consumption and, therefore, a shorter lifetime of the personal sensors. In the experiment it would have been necessary to read out the logger data several times during the experiment since otherwise the data flash would have run out of space.

According to the recorded data, one employee was in his own room only 5.3% of his working time which did not correspond to reality. We discovered that he had bent the antenna to make the mote fit into a cellular phone case. This way, the mote was neither able to receive nor to send any beacons. Clear instructions for the users of new devices help to avoid such problems. A 90 degree rotation of the antenna, so that it can be folded parallel to the long side of the mote, would make the Mica2 mote more practical.

Several of the processing steps we have presented in section 3.2 have been improved or developed during the data processing phase since the problems were unknown in advance. Therefore, it would have been almost impossible to develop an application with full in-network processing of the beacon packets from scratch. We believe that it is always necessary for unknown environments to deploy a test network. Analysis of the acquired sample data gives hints how to improve the quality of data. The application has to be changed accordingly and deployed again. An alternative is a system like TinyCubus [5] that allows for the replacement of the data processing component with an improved one.

4. CONCLUSION

In this paper we have described an experiment to gather real world movement data of employees in an office environment. We have shown the necessary steps to clear up the data and to extract the relevant information. These were used to build a meeting driven movement model for office scenarios. Finally, some problems and other issues we encountered during the experiment and the evaluation were discussed.

We have also presented a Smart Environment application based on Mica2 motes. The sensor network is hybrid, consisting of static and mobile nodes and multiple gateways to the wired network.

Currently, the application is about to be finished and will be deployed soon. Then, we will be able to assess how realistic our model is simulating the movements in our department.

5. REFERENCES

[1] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive Computing*, 3(1):38–45, 2004.

[2] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[3] K. Herrmann. Modeling the sociological aspects of mobility in ad hoc networks. In *Proc. of the 6th ACM intl. workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 128–129, 2003.

[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. In *Proc. of the 10th Intl. Conf. on Architectural Support for Programming Languages and Oper. Systems*, pages 96–107, 2002.

[5] P. J. Marrón, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. Rothermel. TinyCubus: A Flexible and Adaptive Framework for Sensor Networks. In *Proc. of the 2nd Europ. Workshop on Wireless Sensor Networks*, pages 278–289, 2005.

[6] M. Musolesi, S. Hailes, and C. Mascolo. An ad hoc mobility model founded on social network theory. In *Proc. of the 7th ACM intl. symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 20–24, 2004.

[7] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Comm. of the ACM*, 47(6):34–40, 2004.

TinyREST – a Protocol for Integrating Sensor Networks into the Internet

Thomas Luckenbach, Peter Gober, Stefan Arbanowski
Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
Contact: thomas.luckenbach@fokus.fraunhofer.de
Andreas Kotsopoulos*
Dept. of Electrical Engineering & Computer Technology, University of Patras, Greece
Contact: akotsopoul@upnet.gr
Kyle Kim
Samsung Advanced Institute of Technology SAIT, P. 0. Box 111, Suwon 440-600, Korea
Contact: kyle22.kim@samsung.com

Abstract

In the last couple of years, huge research effort has been spent focusing on the realization of wireless sensor networks, especially in the area of radio interface efficiency, power consumption, ad-hoc networking, routing etc. In this paper we would like to focus on how sensors/ actuators and sensor networks can be integrated into the Internet and therefore supporting the development of IP-based applications in different domains.

The work, which will be introduced in the following, represents intermediate results of a joint R&D project between Samsung Advanced Institute of Technology and Fraunhofer FOKUS. The main outcome of the project is a Home Services Framework with respective functions and protocols, based on a middleware layer that integrates different sensor networking technologies, home automation technologies and consumer electronics into the Internet.

Keywords: sensor networks, home automation, middleware, pervasive computing, context-aware applications, consumer electronics

1. Introduction & Motivation

It is generally accepted that sensors and sensor networks[1] will become an important part of the future Internet. Sensors are being used to monitor the "real world" by measuring and detecting environmental information like temperature, brightness, speed, velocity, etc. By providing this information to applications being able to control real world objects like e.g. HVAC (heating, ventilation, air conditioning) systems or electronic home appliances or by being able to feed in information via real world objects like loudspeakers or information displays the Internet is not any more just a network of computers but a "network of things" - also called "the tangible Internet".

[1] The term sensor network is being used throughout this paper in it's general sense referring to a network of sensors and actuators

*) from 10/04 till 3/05 working at FOKUS

A world of seamless intelligent context-aware networking is approaching rapidly. The deployment of tiny communicating low-cost sensors/actuators appears to be a promising solution for the realization of such networks, not only in large factory environments (e.g. industrial building monitoring), automotive networks (e.g. danger on-the-road avoidance, traffic control, etc.) and social service support (e.g. earthquake warnings/readings, patient monitoring in hospitals, context-aware support on emergency situations, etc.), but also in future smart home networking environments.

The idea of IPfication of everything is being researched since several years and also the authors have not only worked out and developed prototype solutions based on so-called smart-IP technology [6] but also brought the idea to standardization fora like Object Management Group - OMG [1].

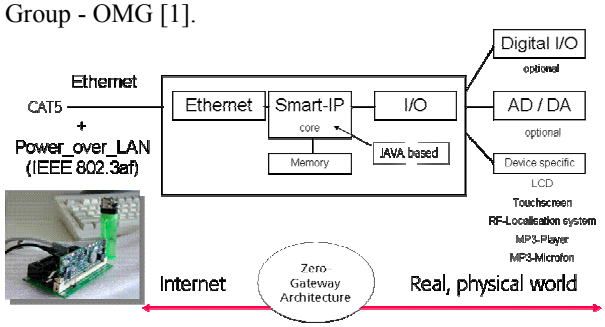

Figure 1: Generic Smart-IP architecture

While the smart-IP approach was based on the usage of a general purpose microcontroller [2] running a device communication protocol called I-net [7] and mainly dedicated to the IPfication of components being used in home&building automation like IP enabled doorplates, loudspeakers, microphones and power plugs, the current paper describes a more general approach by integrating various kinds of sensors and actuators into the Internet.

The sensor boards used for the work described in this paper are based on the results of the "smart dust" project being finished in 2001 at Berkeley University and nowa-

days available as MICA motes from Crossbow technologies [3].

Unfortunately the software coming with these motes does not include any IP networking functionality but just a non-standard messaging mechanism using the TinyOS (see chapt.2). Therefore in order to provide an efficient way to integrate those MICA based sensor networks into the Internet a suitable communication protocol had to be selected and implemented. Due to its inherent support of IP based applications the decision was to use an http-based mechanism for the exchange of information between the sensor network and the Internet.

We use the http based REST architecture to obtain/change the state of the sensors/actuators (see chapter 3). Using the REST architecture and implementing the basic mapping between a set of http-requests and TinyOS messages, guarantees a very efficient way for accessing the MICA motes and the respective sensor network from any Internet client and any Internet based application, respectively.

The current implementation of the TinyREST gateway could be considered as a sensor enhanced middleware for Internet based access to a variety of sensors and actuators supporting completely different application domains like home automation, health care service, facility management or even vehicular area networks. For proof of concept in home automation and facility management, we moved one step forward to the integration of TinyREST into our Smart Environments test bed, which enables the development of a unified user-friendly access-from-everywhere scheme for the entire Smart Environments system (see chapter 6).

2. Sensor Network Platform

As of today there is a large number of different sensor network hardware on the market to choose from for research and prototype purposes [5]. Common to most of them is a simple microcontroller, a few 10 to 100 Kbytes of RAM, a simple radio transmitter operating in an ISM band and some analogue and digital input/output pins. Some sensor nodes already contain some sensors, e.g. light sensors. The size of those nodes is in the order of a few cm^3, their battery lifetime with e. g. two AA size batteries in the order of days to months. Sensor nodes typically come with some kind of basic operating system and a software development kit based on the C programming language.

By far the most widespread sensor network platform is the MICA architecture utilizing the TinyOS operating system [9], both of them being developed by the UC Berkeley. It is programmed in a C variant called nesC and features an event-driven concurrency model. For our application that aims to result in a commercial deployment, it was especially important that the MICA hard- and software design is open source and very widespread, so that one can expect to be able to work with it for a long period of time in the future.

The MICA nodes feature an ATmega microcontroller and are powered by two AA batteries. A large variety of plug-in sensor boards with different sensors are available. We use one featuring a photo diode, a thermostat, a microphone, an accelerator, a magnetometer and a sounder (MTS310 sensor board).

Due to the inherent differences in their messaging mechanism (IP versus Active Messages), there is no full IP stack or even low footprint IP stack like uIP [10] available for TinyOS. Further, communication via full IP packets would be devastating for the usually limited power resources of sensors/actuators. The TinyOS network stack provides active message communication, which supports single hop unicast and broadcast communication. The packet payload length is limited to 29 bytes.

We choose the latest “MICAz” nodes. While earlier MICA nodes used narrowband modulation in the 433/868/915 MHz bands, the MICAz nodes are based on the IEEE 802.15.4 standard, employing a spread spectrum modulation in the 2.4 GHz band. In our lab, the MICAz nodes exhibited a much better radio transmission robustness compared to the older nodes despite the presence of various WLAN equipment competing in the same 2.4 GHz band.

Recently, the first specification of the industry standard ZigBee 1.0 [11] has been released. ZigBee addresses the application to the network layer and also relies on IEEE 802.15.4 as its physical and data link layer. The standard is specifically targeted at low power networked sensors and transducers with a low to medium number of nodes. Currently, the industry consortium is in the process of specifying so-called “profiles” that prescribe the syntax and semantics of messages for some applications like building air conditioning equipment. ZigBee is designed to be used self-contained, without connection to the Internet: Sensors and transducers can be assigned to each other by a process called “binding”. Only recently, the ZigBee consortium addresses the problem of a “gateway” to the Internet. ZigBee is still in its infancy. Currently, only prototype products are available. In the future, ZigBee will probably play an important role in wireless sensor mass markets. However, lack of interoperability with the Internet is still a drawback and gave reason for this work.

3. REST Concept & Basics

The Representational State Transfer (REST), proposed by Fielding [12], applies the uniform interfaces, self-descriptive data, stateless communication, cache components, and code-on-demand constraints on a simple client-server starting architecture, resulting in an architectural style suitable for distributed networked applications that emphasizes on scalability, generality of interfaces, inde-

pendent/self-organised deployment of components and intermediary components.

REST introduces *resources* as the primary abstraction of information and functionality. Resources in REST are addressed via Uniform Resource Locators (URL) using the Hypertext Transfer Protocol (HTTP) and its methods for accessing them(e.g. http://sensor or http://sensor/light).

The basic idea behind the concept of REST is that the state of a resource, e.g. a device, can be get or set by the GET or POST http methods respectively. In our opinion this approach has the following three basic advantages:

1. Conforms with the most widespread internet standard which is HTTP.
2. Tunneling over firewalls is possible
3. The representation of data/information is done in ASCII format which is human-readable, enhancing human-device interaction.

Therefore we decided to make use of the REST's architectural style concept as a basis for the proposed protocol, providing an architecture format with the desired characteristics.

4. Implementation

For the implementation and proof of concept of the proposed protocol, a network of eight 802.15.4 compliant MICAz motes was deployed. The MICAz, developed by the UC Berkeley and Crossbow, operate in the 2.4GHz frequency band, which is the only one worldwide available, providing transmission rates of up to 250Kbps and supporting the IEEE 802.15.4 PHY/MAC layer of the ZigBee standard (Figure 2).

	Channels	Band	Coverage	Data Rate
2.4 GHz	16	ISM	Worldwide	250 kbps
868 MHz	1	ISM	Europe	20 kbps
915 MHz	10	ISM	Americas	40 kbps

Figure 2: ZigBee bands and data rates world-wide

The TinyREST implementation provides any client connected to the Internet with the ability to issue http-like POST, GET and SUBSCRIBE requests, and thus making use of the MICAz not only as sensors but as actuators as well. With POST users are able to command an actuator to take some action (e.g. *switch on a light* or *fire the alarm*), while with GET users can obtain current values by each sensor the MICAz provide[2]. Finally, with SUBSCRIBE clients are able to register their interest to specific services that the sensors/actuators provide, and further with various personalized parameters depending on each client's needs. Each subscribed client will automatically be notified accordingly with a NOTIFY message if and each time a desired event has been sensed (e.g. a temperature value passing a specified threshold).

The necessary interface between the clients and the sensors/actuators is provided by a multithreaded lightweight HTTP-2-TinyREST gateway, able to manipulate more than one connection from clients concurrently. The HTTP-2-TinyREST gateway is responsible for establishment, handling, and dropping communication to/from the client side and the sensor/actuator side, including all the necessary validity checks and message format mappings.

In fact, the gateway utilizes the *DCP*[3] *Device Adapter* inside the overall architecture of our Smart Environment Lab which is set up in Fraunhofer FOKUS (Figure 5), and thus translates DCP elements into TinyREST specific protocol elements.

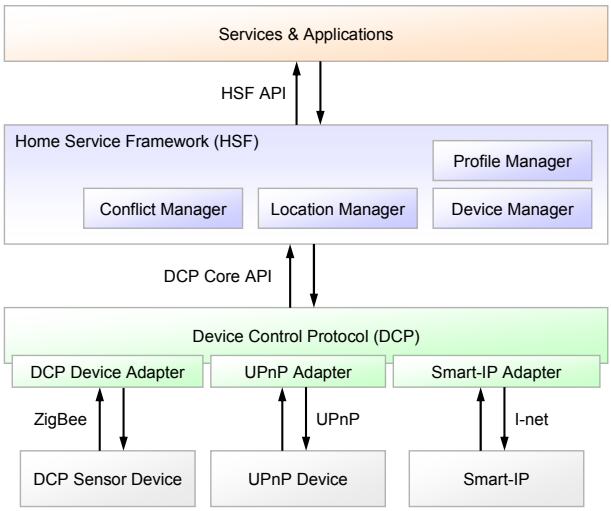

Figure 3: Overall Architecture

Additionally, through an addressing mechanism, all requests, POST, GET, SUBSCRIBE can be forwarded by the gateway to a specific sensor/actuator, or to a group of sensors/actuators, or broadcasted to all motes in the network. The addressing is based either on the mote_id or on the group_id, which are mapped to locations on the *Location Manager* of the *Home Services Framework* which is hosted on the *Home Server* (Figure 4), so that the users can easily and in a human understandable manner specify where their requests should be directed, without having to remember mote/group ids which could also change.

[2] The current implementation uses MTS310 sensor boards attached on top of the MICAz (processor/radio), each of which comprises light, temperature, accelerometer, magnetometer and sound sensors, as well as a 4 KHz tone generator.

[3] Device Control Protocol is an abstractive higher level framework developed in FOKUS enabling interconnection and communication between different networking protocols such as UPnP, Smart-IP, etc.

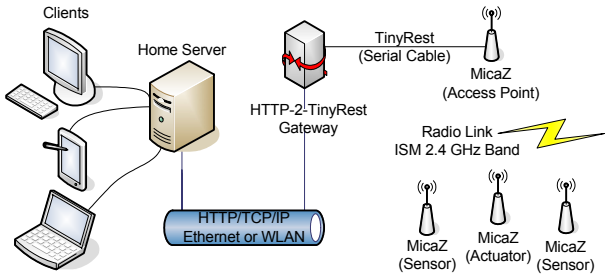

Figure 4: TinyREST Achitecture

The requests issued have the following format [8]:

"*METHOD URL*"

where *METHOD* combined with *URL* is the actual command to a sensor/actuator. The messages are transmitted as plain ASCII with the mechanisms provided by TinyOS.

The following table gives some examples how to construct a TinyREST request:

<METHOD>	**<URL>**
POST	/gatewayIP/bathsensor/devices/lheater/on remote setting of bathroom heater
GET	/gatewayIP/baby_sensor/temperature remote checking of body temperature via body area sensor
SUBSCRIBE	/gatewayIP/entire_home/light automatic detection of presence in the house in absence of resident
SUBSCRIBE	/gatewayIP/floor1/sound automatic detection of noise on the 1st floor for intruders. Alarm may be posted upon notification.
SUBSCRIBE	/gatewayIP/entrance/motion-sensor/change automatic motion detection

Table 1: TinyREST message examples

Upon start-up, the *HTTP-2-TinyREST gateway* publishes its current IP address and PORT number to the always up-and-running *Home Services Framework* (HSF) which resides on the *Home Server*. Further, each sensor/actuator will broadcast an '*ANNOUNCE(new) /mote_id/voltage'* message upon initialization, registering on the network and providing its remaining battery power. Since this announcement is broadcasted and since no multi hop is supported in our current implementation (it is consequently assumed that every MICAz has the base station in range), the message will be dropped as unknown by every other mote except the base station. The latter will forward it to the gateway and the gateway will forward in turn to the HSF. Thus, the HSF and consequently the clients are aware of where they should/can send their requests.

Suppose a request from an application is sent via the HSF to the *HTTP-2-TinyREST-Gateway*, which is continuously listening for incoming connections. This part of communication is done using TCP/IP between the application and the gateway. Once the gateway receives the HTTP request, it makes the necessary validity checks, constructs a TinyREST message with the actual command, encapsulates it in the data field of the message format understood by the MICAz motes and sends it through a serial cable to the MICAz mote acting as a base station. The base-station forwards the message over radio to the MICAz sensor by broadcasting. The message will be accepted by those motes, whose mote_id/group_id matches the one specified in the request or by all motes if the broadcast address was specified in the request.

Upon reception of the request and depending on the type of the request:

- the sensor either responds with an ACK, and continues with the necessary actions the request demands,
- or executes the command and informs the gateway with an ACK that the command was executed.

In this way the gateway either receives an ACK or receives nothing (assuming that the request has not been serviced) and notifies the application accordingly (request serviced/request time-out: resend).

Currently, scenarios in which the sensor(s) has received the request but the ACK(s) was lost have not been addressed, but will be dealt with in the future to increase the system's reliability. The most common solution for this is adding TCP/IP-like sequence numbers in our message format. Finally, *keep-alive and PING* mechanisms have been implemented. Every mote on the network broadcasts an '*ANNOUNCE(alive) /mote_id/voltage'* message every 30sec. These messages are forwarded by the base station to the gateway which keeps track of alive motes and informs the HSF in case a mote seems to be out of the network or goes very low on battery. Additionally, any client is able to issue the *'PING'* command to a mote or to a group of motes, or to all motes and receive an immediate response accordingly.

5. Experiments & Measurements

As already mentioned the MICAz proved to have a far better performance regarding not only radio efficiency and speed but also power awareness and robustness against interference.

We tested the power efficiency and stability of our implementation by letting 3 MICAz working with eight different subscriptions stored on each mote, such that oblige the motes to transmit NOTIFY messages frequently (approx. every 2min. during the day, and every 1hour or less during the night according to the indoor environmental conditions i.e. light, temperature, movement). The timestamps of the messages sent by each mote where stored in a log file for analyzing the frequency of transmission and the stability of the motes. The results showed that all three motes lasted about seven days, each starting powered with two fully charged AA batteries. It must be

referred that no sleep modes where used for power saving. Most probably, the motes would have lasted much more if a power-awareness scheme was applied.

Further, the MICAz proved to be robust against interference, even with the co-existing WLAN networks operating in the same frequency band. We noticed negligible packet loses even in distances (between the base station and the sensors/actuators) approaching 25 meters.

Finally, the round-trip time, from the time the user issues e.g. a "GET /mote_id/light" request until the actual light value appears on the screen of the user, was measured to be approximately 40-50msec. The overheads included in this time period can be analyzed as follows, starting from the point when the request arrives at the gateway:

$RTT = 2 \times (t_{java} + t_{serial} + t_{air}) + t_{processing}$

where

RTT: Round Trip Time

t_{java}: Java gateway overhead

t_{serial}: Gateway-base station communication via serial interface at 115Kbps

t_{air}: base station-sensor/actuator over-the-air communication at 256Kbps

$t_{processing}$: request service time by sensor/actuator

assuming that the lengths of messages travelling to and from the sensors/actuators are more or less equal.

6. Test Bed Integration

The development of the TinyREST middleware has been seamlessly integrated into the Smart Environment test lab at FOKUS being shown in figure 5.

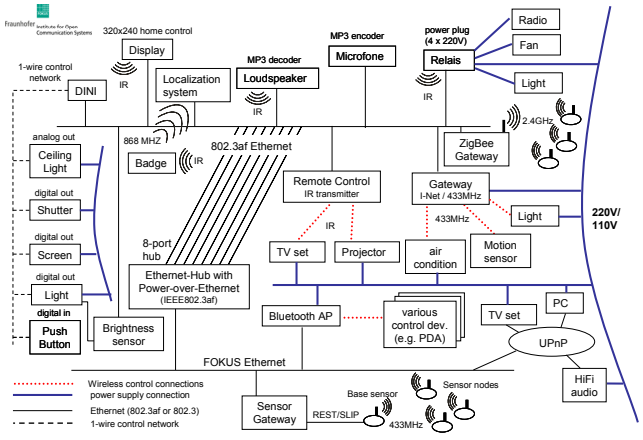

Figure 5: Smart Environment Test Lab

The demo lab comprises a variety of technologies being evaluated and including different wired and wireless technologies like Power-over-Ethernet based on 802.3af, 433/868MHz RF, 2.4GHz WLAN, Bluetooth and Infrared. From a functional point of view it includes a local positioning system (LPS) based on a combination of IR transmitters sending permanently their own (fixed) position and active badges receiving this information when they are in reach (about 1-2 meters), combining the received position information with their own badge/user-id and transmitting the combined values via 868MHz RF to the LPS base station (max. distance about 20m). By this mechanism it is possible to implement location based and personalized services in indoor environments [4].

Beside this LPS the lab includes a UPnP network, a network of electronic home appliances including IP-enabled loudspeakers, displays and microphones as well as a 1-wire control network for facility management functions. The described integration of the TinyREST middleware for sensor networks allows us to build applications making use of all the available components in a unified and Internet based manner.

7. Summary & Outlook

The work described resulted in a first gateway prototype running on a notebook PC with a "base station" MICAz mote connected to the serial interface. The TinyREST gateway allows direct access from the Internet via http to the sensor network. Based on this work the development of application scenarios making use of this Internet-integrated sensor networking environment has already started.

8. References

[1] OMG SDO DSIG. "PIM and PSM for Super Distributed Objects". RFP. OMG Document: sdo/02-01-04

[2] TINI, Tiny InterNet Interface, http://www.maxim-ic.com/TINIplatform.cfm

[3] MICAz data sheet, www.crossbow.com

[4] T. Pfeifer, D. Elias: Commercial Hybrid IR/RF Local Positioning System, KIVS 2003, Feb. 26-28 2003, Leipzig, Germany

[5] P. Gober: IP-managed networking – Sensors, ZigBee,…, e/home fair, Sept. 1-3 2004, Berlin, Germany

[6] T. Luckenbach: Smart-IP in (home) automation: the Infranet on the Internet, e/home fair, Sept. 1-3 2004, Berlin, Germany

[7] I-net, Individual-centric networking, final project report, June 2004 (in german), available through the authors

[8] W. Drytkiewicz, I.Radusch, S. Arbanowski, R. Zeletin: pREST: A REST-based Protocol for Pervasive Systems, 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Oct. 24-27 2004, Ft. Lauderdale, USA

[9] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer and D. Culler: The Emergence of Networking Abstractions and Techniques in TinyOS, NSDI 2004.

[10] Adam Dunkels, Juan Alonso, and Thiemo Voigt, "Making TCP/IP Viable for Wireless Sensor Networks", the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session.

[11] ZigBee homepage, http://www.zigbee.org

[12] Fielding, Roy Thomas. Architectural Styles and the Designs of Network-based Software Architectures. Phd, University of California, Irvine, 2000

Real World Issues in Deploying a Wireless Sensor Network for Oceanography

Jane Tateson,
Christopher Roadknight,
Antonio Gonzalez
BT
Adastral Park
Ipswich IP5 3RE
01473 640710,
01473 644763,
01473 663704
jane.tateson@bt.com,
christopher.roadknight@bt.com,
ae.gonzalez@ee.ucl.ac.uk

Taimur Khan,
Steve Fitz,
Ian Henning
Essex University
Wivenhoe Park
Colchester CO4 3SQ
01206 872894,
01206 872865,
01206 872431
khant@essex.ac.uk,
stephenf@essex.ac.uk,
idhenn@essex.ac.uk

Nathan Boyd[1]
Intelisys Ltd
Chris Vincent[2]
University of East Anglia
Ian Marshall[3]
University of Kent
nboyd@intelisys.co.uk,
c.vincent@uea.ac.uk,
i.w.marshall@kent.ac.uk

ABSTRACT
In this paper we describe some of the practical issues involved in designing, building and deploying a sensor network for oceanographic monitoring. The paper explains some of the design decisions and their consequences, and some of the lessons learned from a first sensor network trial at sea.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based systems**]: - *microprocessor/microcomputer applications.*

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design - *wireless communication*.

General Terms
Design, Experimentation, Algorithms

Keywords
Wireless sensor networks, environmental monitoring, design, deployment

1. INTRODUCTION
Oceanography is the study of processes that govern the complex interplay of tides, currents, waves, and seabed and coastal modelling. Oceanography can tell us about coastal deposition and erosion and consequently about flooding and sea defences. An area of particular interest for this work has been Scroby Sands, a sandbank off the coast of Great Yarmouth where a windfarm has recently been constructed. The sandbank is interesting oceanographically, providing a sheltered coastal region, but also a raised feature in the seabed which exhibits a highly dynamic topography[4]. Scroby Sands therefore offers the opportunity to study not only active sedimentation and wave processes in the area, but also how this may be affected by the building of the windfarm, which consists of 30 large (100m tower height) turbines.

Sensor networks offer a new paradigm for oceanography, and many other scientific, commercial, agricultural and industrial applications. Deployments of sensor networks for environmental monitoring include work on Duck Island, Maine[7] and work in Western Australia on water balance[1]. Our aim was to enable oceanographers, represented by UEA in our consortium, to collect spatially distributed data points, by providing a spatially dispersed measurement array that was at least one order of magnitude cheaper than conventional oceanographic kit. This should enable a much richer characterisation of complex oceanographic processes, especially near-shore, where large-scale macro models do not hold.

As well as exploiting cheap sensors, the sensor network has been designed to operate autonomously, and adapt its rate of taking measurements, data processing and network communication, to local conditions that are not known a priori.

Power management has also been central to the design, with the autonomous AI used to control sensor node operation on the basis of available resources, in particular: communication bandwidth and battery power[5].

Prior to prototype development, device intelligence and networking techniques were devised and simulated and have

[1]Intelisys Ltd, Manchester M13 9PL, 0161 2768366

[2]UEA, Norwich NR4 7TJ, 01603 592529

[3]University of Kent, Canterbury CT2 7NF, 01227 827753

generated novel and exciting new approaches. Primarily, these approaches consist of, firstly, self-organising time division multiplex[2] and, secondly, a layered approach to autonomous AI for distributed devices, which combines lightweight management of optimal sampling rates and data aggregation, with differential qOS and forwarding control.

However, the main subject of this paper is the more practical issues involved in going from the ideas phase to prototype deployment. This work was carried out in the SECOAS project, which was part-funded by the Department of Trade and Industry, as part of their Next Wave Technologies and Markets initiative, and involved collaborative work amongst BT, Intelisys, University College London, Kent University, Essex University and the University of East Anglia.

2. PROTOTYPE DESIGN

The sensor package was designed as a waterproof cylinder (approx 50cm long) containing a sensor section, a data-logger and a microprocessor (PIC) running the lightweight device control algorithm, designed to control the measurement rates, data processing, queue management, data aggregation and data forwarding, together with 2 alkaline D-cells. The sensor package is designed to last several months using these batteries. Attached to this cylinder is a 'dongle' which is able to move in the current, and thus provide current velocities. The sensor section incorporates a temperature sensor, a water-pressure sensor, from which wave-height can be derived, an optical backscatter sensor that measures turbidity, and an electrical conductivity sensor, which is used as a surrogate for salinity. The sensor package is suspended within a pyramidal-shaped cage, designed to remain fixed on the seabed, thereby giving a consistent reference orientation for current velocities and clearance above the sea bed for the optical sensor.

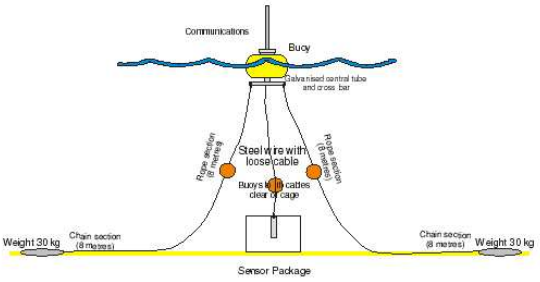

Figure 1. Sensor node mechanical design.

As part of prototype design, a decision was taken to modify a mooring buoy, to house the radio and antenna, rather than having a bespoke buoyancy housing made. This was a good decision in terms of cost, but also had the advantage that we could get advice on appropriate moorings for the buoy, and the buoy was bright yellow, meaning that it was an approved colour that would be visible and recognised by sea-going traffic. Metalworking was needed to give a platform on to which a waterproof box could be attached, to house the radio electronics and batteries. We were also required by marine authorities to put a light on top of the buoy. The total cost of manufacture of one node, including sensor package, cage, cables, ropes, chains, radio buoy, machining, electronics housing, radio, batteries and light was approximately one thousand pounds. This is an upper limit for this node design, representing small-scale manufacture, without the efficiencies that can be achieved with scale. Already this is easily one order of magnitude cheaper than typical oceanographic kit, with the prospect of another order of magnitude (down to a few hundred pounds) for large-scale manufacture. These cost advantages are also supported by technological advantages of adaptive sampling, robustness through node redundancy, and autonomous operation and data collection, plus the fact that our devices can measure current velocity (speed and direction) which is not typically available to oceanographers using conventional equipment.

The radio frequency was selected from the range of unlicensed ISM frequency bands. The propagation characteristics of a number of frequencies were investigated at sea and of those 173.25 MHz was found to offer a good compromise between antenna size, and range. Results indicated that using +10dBm RF power, a theoretical link length of 1.5 km at a data rate of 10 kbps was possible in calm conditions, using sea level antennae. Higher rates were possible for shorter distances. Key to our approach has been the ability to increase the rate of measurements, in order to gain greater resolution of phenomena when these are of greatest interest, i.e. changing rapidly and unpredictably. At these times, the bandwidth is not sufficient to send all these measurements immediately. However, the node management algorithms enable data storage and data processing, in order that the most 'interesting' measurements take priority, in terms of available bandwidth [5]. Wanting to keep the profile of the node small, and in particular the torque on the buoy, during windy conditions, a half-wave monopole antenna was chosen for a frequency of 173.25 MHz, that being around 50 cm in length.

3. PRETRIALS

3.1 Mechanical Trial

The first sea trial off Great Yarmouth consisted of a week-long deployment of a sensor package and cage with software running, attached by chain to a buoy. This was essentially a mechanical trial, to verify the efficacy of the moorings, and to check that the box on the buoy, designed to house the radio, was waterproof. A significant part of the experiment also concerned physically deploying the system and recovering it. We used two weights to moor the buoy. The idea was that opposing moorings would prevent the buoy from twisting, which would be important to prevent twisting of the cable to the sensor package. The line to the cable package was designed to remain slack, with the mooring ropes under tension. Attached between the mooring ropes and mooring weights was heavy chain, designed to lie on the seabed, and prevent the weights from being dragged. We were aware that if the weights once lifted, the whole equipment would most likely be carried out to sea. Our week-long trial with sensor package at about 8m depth showed that the moorings were stable, at least for these moderate sea conditions. We were able to recover sensor data and verify that the node intelligence algorithms were stable and reliable[5].

Figure 2. Photo of deployed radio buoy.

At the start of the mechanical trial, we held a public meeting with local fishing groups, and informed them of the location of our deployed buoy, which had been positioned outside the shipping lanes and away from known fishing areas. This meeting was part of our application to the Marine Consents and Environment Unit (MCEU), to be allowed to carry out sensor network trials at sea in this area. Ultimately we were granted a licence by the MCEU to carry out our two network trial deployments, the first, which is the main subject of this paper, to be carried out with 6 nodes, for one week, in October 2004. A radio licence was also applied for and granted by OFCOM to enable radio experimentation at 173.25 MHz, within specified power limits and in the deployment zone. A licence was also purchased from Crown Estate to allow equipment to be placed on the coastal seabed.

3.2 Radio Trial

The second mini-trial was essentially to test the range and visibility of the radio buoy antennas, in open sea. We stationed ourselves on the shore, in a caravan park, just above the dunes (5-8m above sea level). A prototype radio buoy was constructed, taken out in a boat, and set to transmit diagnostic packets from a number of points at various ranges from the shore. Positions were logged from the boat with GPS. With a YAGI antenna in our hands, attached to a receiver radio and laptop, we listened to the diagnostic packets being sent from the radio buoy at sea, and were able to assess the radio visibility of the buoy. In spite of the small size of the antenna (0.5 m), in relation to typical wave heights that can reach several metres in storm conditions, good signal strength was received, in light to moderate sea conditions, with acceptable bit error rates, for ranges of up to 3km.

4. FIRST NETWORK TRIAL

4.1 Integration

The first system trial took place over 2 weeks in Oct 2004. This trial was the first time the sensor package had been linked to the radio buoy in field conditions. The sensor controller acted as the master, polling the radio and sending data to the buoy when the radio controller indicated availability in response. This data transfer took place over a 12m cable linking the seabed package with the radio buoy via RS232 ports.

4.2 Networking

The longer term intention of the work is to create a large scale network incorporating hop-hop communications, and exploiting self-organizing clustering. However, for the first network trial it was decided to experiment with a simple single cluster as a representative of one element of the target system. This experiment was designed particularly to produce an accurate characterization of network settings for the oceanic environment. The focus was on single hop link behaviour as the appropriate first step in understanding the challenges that sea conditions create for networking. A ‘star’ type of network was deployed, with a cluster-head. See Figure 3. The experiment created the data to enable parameters of sea-to-sea and sea-to-shore packet exchange to be deduced, over time and at different sea states. Tools, instruments and programs were deployed to record received signal strength information, bit error rates, synchronisation parameters and specific ARNEO (Autonomous Resilient Network for Resilient Environmental Observation) parameters, such as induction sequencing and persistence[3].

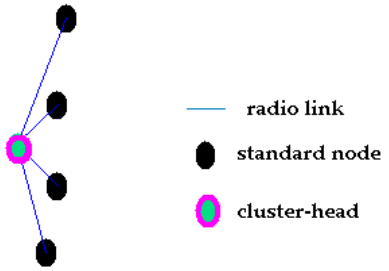

Figure 3. Network topology.

4.3 Node Assembly

We chose a venue for assembly in Great Yarmouth, as it made sense to have a short trip to sea for the nodes which, though very lightweight in terms of oceanography, were bulky to handle. The moorings were loaded on to an ex lifeboat vessel, and moorings were attached to each buoy, prior to launch. For water-tightness, the radio box was packed with resin. The radios themselves were first put into plastic bags, with the hope that the resin would not prevent them from being probed and possibly re-used after the trial. All holes and channels through the buoy, through which the data cable passed were also squirted with resin, as was the entrance to the sensor module. Unfortunately, immediately prior to expected launch, some of the sensor modules appeared not to be sending any data along the cable. This led to the sensors being cut off to be fixed. This was caused by a last-minute software change which had had an unexpected fatal side-effect. The software was soon put right, but this resulted in having to attempt a water-tight junction to re-join the cut data cable. Despite our best efforts in filling this junction box with resin, this was inevitably a weak point, and probably caused the failure of one of the sensor modules, due to water leakage, at deployment.

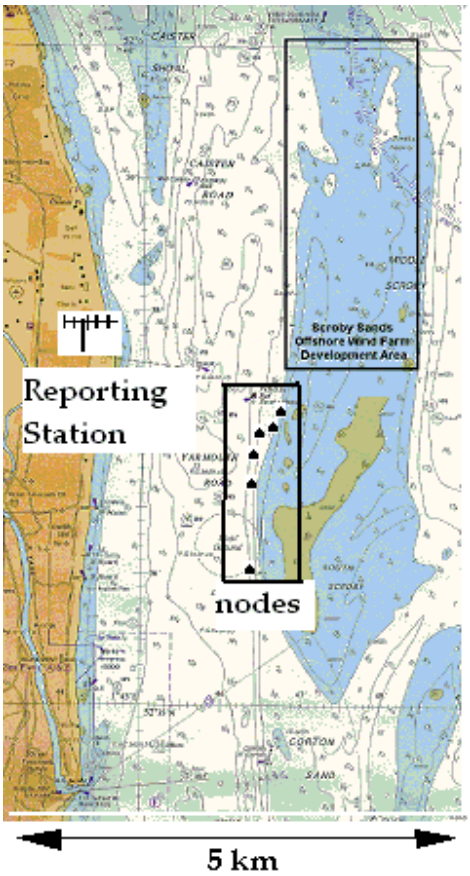

Figure 4. Map[4] showing positions of nodes and report station.

4.4 DEPLOYMENT

Following the mechanical trial, we expected to be able to deploy 6 nodes in one day, weather permitting. Luckily, it was close to flat calm. Deployment was carried out parallel to the direction of the tidal current, and at high tide, when the tidal current is weak, and the sensor modules are at maximum depth. Locations were chosen to be close to the sandbank, in shallow water, immediately south of the windfarm. See Figure 4. Engines were cut to prevent interference of the propellers with the ropes and chains, and the first mooring was dropped overboard. The tidal current caused the boat to drift parallel to the shore. Waiting for the rope to go tight, the sensor module was then dropped overboard, closely followed by the radio buoy, and then – as the second mooring rope began to tension - the second mooring weight was thrown overboard. It was useful to have an additional short loop of rope on the top of the buoy to help to prevent the buoy becoming submerged. Although, where capsize did occur, we found that the buoys righted themselves, and were stable with the antenna out of the water. However, the deployment of the sixth node showed that this stability depended on the strength of current acting on the buoy and mooring system. The first five sensor nodes were deployed at depths of around 6m, where the tidal range would vary this by +- 1m. The sixth buoy, placed the deepest, at around 10-12 m, was also in a more exposed position, in relation to the sandbank, and the current here was much stronger, and tending to pull the buoy under water.

4.5 Reporting Station

A reporting station was set up at a caravan facing the sea front. The reporting station included a radio snooping device, two PCs redundantly logging radio events, and a RSSI logger. The radio snooping device was implemented with a seven element yagi antenna and a radio board running on batteries. The two PCs logged the radio events being produced by specific networking diagnostic software running on the radio snooping device. The RSSI logger sampled the RSSI output PIN available from the radio receiver of the radio snooping device. The detailed reasons for this design will be published subsequently.

4.6 Weather

From an oceanographic point of view, and an experimental point of view, we had ideal weather for the trial. Initially it was flat calm, but, over the next 24 hours, the winds became much stronger, reaching gale force after two or three days. Our concurrent observations from a fishing association said 'S/W gale force overnight and into the morning. Seas moderate to rough. Waves up to 3m high on sand banks. Not suitable for fishing. Clarity of water very poor. Current speed approx 2-2.5 knots.' At this point, small fishing vessels were unable to put to sea, and we were able to gauge the performance of the radio communication and the mechanical design of the nodes in adverse conditions. In spite of strong winds and heavy rain, information derived from the networking diagnostics received up to 3km away on the shore, indicated that network activity was still taking place. Further details will be included in a future paper.

4.7 Node Recovery

Due to the change in the weather, we could not recover the buoys on the expected day, but had to leave them in the sea for about 10 days in total. When we came to look for them, the moorings had held well, even in the faster moving current. Knowing that it had been arduous to lift the mooring weights manually back into the boat, following the first mechanical trial, we had also experimented with anchors instead of weights, for 3 out of the 6 nodes deployed. The anchors did an equally good job, and were marginally easier to pull in. Recovery was certainly harder than deployment, taking more effort and more time. The method of deployment had ensured tight moorings, which therefore offered little flexibility for hooking out the buoys. In the most dramatic buoy recovery, the whole node, including sensor cage and moorings became entangled around one of the boat's propellers and had to be removed in a dry dock. However, all the sensor modules and radio buoys were ultimately recovered intact.

4.8 Results

One of the sensor modules failed almost immediately, on deployment. We suspect that this was because of water ingress along the data cable from a cable junction. Another sensor module exhibited faulty behaviour after a couple of days of deployment, which we can also attribute to water ingress.

---

[4]Permission to reproduce chart granted from UK Hydrographic Office.

However, in general the sensor modules behaved as expected. The independence of sensor and radio modules enabled the radio networking side of the experiment to be unaffected by the sensor module failure.

In all, about one and a half million remote radio/network transactions were recorded, producing a large multivariate log for further analysis. More than 50 million samples of locally collected high resolution RSSI data were recorded. We have started analysing the results, and the viability of the mechanics, node management software, radio and antenna has been demonstrated.

In addition to the data samples transmitted, a comprehensive data set was retrieved from the deployed data loggers. This is being used as an ideal reference with which to test the sampling and aggregation strategies. An example of unprocessed data of water pressure recorded at 3 nodes over a 10 day period is shown in Figure 5, from which the tidal cycles are clearly visible. At finer granularity, wave activity can be observed.

In terms of node intelligence, we found that a rule based approach (sliding window averaging) was successful in reducing the amount of data to be transferred, while minimising the loss of information [5]. This type of systematic approach lacks adaptive behaviour, but benefited from local learning feedback that modified probabilities of certain actions based on internal and external conditions [5]. Making these probabilities evolvable and transferable between devices has been shown to increase robustness by optimising parameters in real time [6].

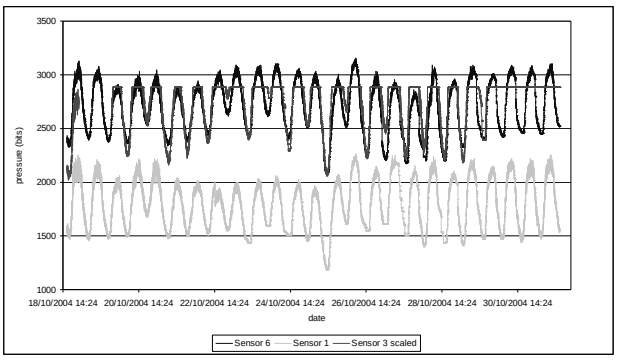

Figure 5. Unprocessed pressure sensor data from 3 nodes.

From a sensor point of view, the pressure sensor calibration was not ideal for the deeper node depth of around 12m, but this is easy to adjust. The turbidity sensor gave occasional aberrant readings, but this was probably due to a piece of weed being introduced into the sensor chamber. Prevention of such large debris entering the chamber is relatively easy. A second longer trial, in autumn 2005, will provide a better characterisation of the oceanography, as well as an improved network functionality.

5. LESSONS LEARNED

Early analysis indicates that sea-to-sea communication can produce tremendous stresses on networking that could lead to its collapse, supporting the need for specific approaches that include properties of autonomy and resilience. (Full results will be published elsewhere.) The experiment was also a good indicator of the suitability of the algorithms for computing resource constrained platforms[3,6]. Improved versions of the radio devices are in the process of being designed and tested at Essex University, which will deliver improved performance, with considerably less power consumption.

The basic node design was reasonable for the conditions to which it was exposed. Even the node at the southern-most point of the sandbank remained moored. We were able to show that the node management software was stable, with analysis of its performance being published elsewhere [6]. However, clearly there was at least one design omission, in not having a manual node re-set. One of the early premises of our work has been reprogrammability of our nodes, remotely, via the network. This feature would also have helped if we had had time to implement it for this trial. It certainly remains an important design goal.

6. REFERENCES

[1] Cardell-Oliver, R., Smettem, K., Kranz, M., and Mayer, K. Field Testing a Wireless Sensor Network for Reactive Environmental Monitoring. In *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '04)* (Melbourne, Australia, 5-8 December 2004).

[2] Gonzalez, A., Marshall, I.W., Sacks, L., Henning, I., Fitz, S., and Kahn, T. Self-Synchronised Scheme for Communication in Wireless Sensor Networks. In *Proceedings of the London Communications Symposium (LCS '04)*(London, UK, September 14, 2004).

[3] Gonzalez, A., Marshall, I.W., and Sacks, L. A Self-Synchronised Scheme for Automated Communication in Wireless Sensor Networks. In *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '04)* (Melbourne, Australia, 5-8 December 2004).

[4] Reeve, D.E., B. Li, and Thurston N. Eigenfunction Analysis of Decadal Fluctuations in Sandbank Morphology at Great Yarmouth, *Journal of Coastal Research*, 17 (2), 371-382, 2001

[5] Roadknight, C., Parrott, L., Boyd, N., and Marshall, I.W. A Layered Approach to in situ Data Management on a Wireless Sensor Network. In *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '04)* (Melbourne, Australia, 5-8 December 2004).

[6] Roadknight, C., Parrott, L., Boult, S., and Marshall, I.W. An Intelligent Sensor Network for Oceanographic Data Acquisition. Submitted to *SECON 2005*.

[7] Szewczyk R. et al. Lessons from a Sensor Network Expedition. In *Proceedings of the 1st European Workshop Wireless Sensor Networks (*EWSN 04), LNCS 2920, Springer, 2004, pp 307-322

Poster abstracts

Embedding a Microchip PIC18F452 based commercial platform into TinyOS

Hans-Jörg Körber† hj.koerber@hsu-hh.de
Housam Wattar† wattar@hsu-hh.de
Gerd Scholl† gerd.scholl@hsu-hh.de
Wolfgang Heller‡ Wolfgang.Heller@enocean.com

† Helmut-Schmidt-University
Electrical Measurement Engineering
Holstenhofweg 85
22043 Hamburg, Germany
+49 4065412638

‡ EnOcean GmbH
Kolpingring 18a
82041 Oberhaching, Germany
+49 89673468934

1. INTRODUCTION

For embedding radio communication modules into low power devices unconstrained by expensive or obstructive wiring an efficient network operating system working under stringent energy conditions is vital. TinyOS is specially designed for wireless sensor networks (WSNs) and thus faces many of the design criteria of WSN such as severe memory, minimized code size requirements, fine-grained power management and flexible event handling and task scheduling. As both the interest in the programming environment TinyOS/nesC [1] and its user community are rapidly increasing a steady rise of the number of supported platforms (motes) can be simultaneously observed. The most recent platforms which have been implemented into the TinyOS source tree [4] are the Telos and Eyes motes [2], which are both based on a TI MSP430 microcontroller (MCU).

Comparable with the MSP430 Microchip offers with the PIC18F family a bundle of MCUs especially designed for low-power devices. To tap the full potential of PIC-based low-power wireless sensors and actuators already on the market we implemented the PIC18F into TinyOS. However, at present there is no GCC tool chain available, which is required by TinyOS. Although there is an open source project [7] with the goal to make GCC available for the PIC the GCC support has not reached his maturity. Instead an alternative method had to be found to introduce the PIC into TinyOS.

2. EnOcean TRANSCEIVERMODUL

The respective hardware platform embedded into TinyOS is the transceiver module TCM 120 developed by EnOcean [6]. The TCM 120 was originally intended to operate as a wireless modem with the capability of processing and relaying all of the different types of EnOcean radio messages generated by self-powered wireless modules like radio switches and sensors. The TCM 120 is controlled by a Microchip PIC18F452 MCU (32 KB ROM, 1.5 KB RAM) which offers all of the common features like UART, ADC, Timer, SPI, I^2C, etc. The MCU is driven by a 10 MHz crystal. For clock rate adjustment a phase-locked loop (PLL) is employed leading to effective 10 Mega instructions per second (Mips). This enables fast clocking of the transmit data into the radio part so that a radio data rate of 120 kbps is achieved. The radio part which is split into a discrete very low-power transmitter and an Infineon TDA 5200 receiver operates at a center frequency of 868.3 MHz. The modulation type is ASK, the maximum transmission power is 10 dBm.

The performance characteristics of the TCM 120 in comparison to alternative hardware platforms are given in "Table 1". If the energy per bit and mW radio output power is considered this comparison clearly shows that the TCM 120 has a very good performance. Moreover the high radio data rate minimizes both the radio message length and energy consumption. This is advantageous if especially network traffic and network lifetime have to be considered.

Table 1. Platform comparison

Parameter	EnOcean TCM	Crossbow Mica 2	Telos	EyesIFX
MCU	PIC 18F452	ATMEL 128 L	TI MSP 430	TI MSP 430
Current Active Mode (mA)	13.40	8.90	1.70	1.40
Current Sleep Mode (µA)	8.00	27.70	3.30	2.90
Tx Current (mA)	9.90	15.50	19.40	11.90
Rx Current (mA)	15.80	10.70	21.10	14.80
Supply Voltage (V)	4.75	2.70	1.80	2.10
Data rate (kbps)	120.00	38.40	250.00	64.00
Energy per bit - Tx (µJ)	0.92	1.72	0.15	0.44
Energy per bit and mW (µJ/(bit*mW))	0.09	0.52	0.15	0.31
Tx Output Power (dBm)	10.00	5.00	0.00	1.50
Tx Input Power / Tx Output Power	4.70	13.23	34.92	17.67
Energy per bit - Rx (µJ)	1.16	1.38	0.16	0.53
Rx Sensitivity (dBm)	-95.00	-98.00	-94.00	-95.00

3. EMBEDDING PROCESS

In the case of hardware platforms like Mica 2 [8] or Telos the standard TinyOS compilation process is as follows: the nesC-compiler NCC, which is an extension to GCC, is started and generates an executable for the respective platform as well as a huge C-file called "app.c" in which all nesC source files are preprocessed and collected. Right after that a binary form of the executable is written into the program memory of the respective mote. Hereby the code is pushed through a programming interface like JTAG into the program memory of the MCU. Each type of MCU requires its own GCC-compiler libraries, which are to be embedded into the TinyOS compilation process.

Due the PIC's missing GCC option an alternative way had to be found in order to make the TinyOS nesC source files running on the PIC18F452 based EnOcean platform.

Like in the Wisenet project [3] our method of making the TinyOS nesC-files to match with the Microchip PIC18F452 and the

associated Microchip C18 C-compiler is build upon the "app.c"-output-file of the nesC-compiler (see "Figure 1"). We had to include the PIC into the appropriate sections of the TinyOS- make process. Additionally we created a platform directory for all PIC specific nesC source files. In that way we were able to invoke NCC with our platform as target. After this step we used a Perl-script to modify the "app.c" such that it can be compiled by the PIC specific C18 C-compiler. Hereby the respective Perl-script of the Wisenet project served as template. In contrast to [9] we did not have to reduce the TinyOS inherent function call depth since the 31-level call stack of the PIC18F452 is sufficient.

To test the feasibility of this approach we first implemented the PIC specific interrupt service routine (ISR) with its appropriate syntax in the "app.c"-file of the very simple TinyOS "Blink"-application which periodically toggles a LED.

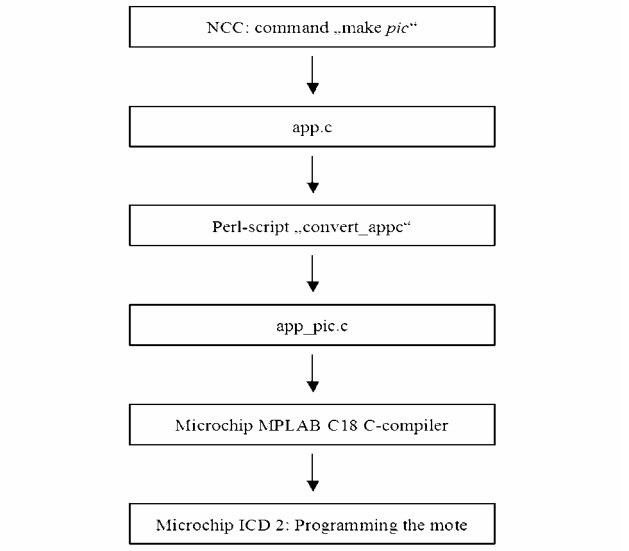

Figure 1. Compilation process

4. TINYOS IMPLEMENTATION

By porting TinyOS to the EnOcean TCM 120 it was our intention to expose most of the peripheral modules offered by the MCU. Thereby it was our goal - in order to maintain compatibility with the existing TinyOS applications - just to modify the hardware abstraction layers (HPL) and the lower presentation layers as much as necessary. Our TinyOS software package for the standard TCM120 hardware, which is available for download on the TinyOS sourceforge site [5], contains the following modules:

- Radio Interface module, which handles both the radio transmitting and receiving and provides a received signal strength indicator (rssi)
- Timer module, which is driven by the main oscillator and generally offers a resolution of 1.024 milliseconds with a precision of +/- 25.6 µs
- UART module, which runs with a default serial data rate of 57.6 kbps
- ADC module (10-bit successive approximation a/d)
- Power Management, which forces the mote into low power sleep mode with a wake-up time of 2 ms plus 1024 clock cycles and the MCU internal watch dog timer serving as wake-up source

5. CONCLUSION

The presented embedding of the commercial Microchip PIC18F452 based platform EnOcean TCM 120 into TinyOS showed the feasibility of porting TinyOS to a MCU without GCC support. Our platform shows very good performance when compared with the other TinyOS platforms. In the current design state our platform can serve as gateway between self-powered wire-and batteryless ambient energy sensors form EnOcean, which are already available on the market, and the steadily increasing TinyOS world.

In order to reduce power consumption to a level such that the energy for the transceiver modules can be entirely sourced from the primary process or the environment we will replace the PIC18F452 by the new PIC18F4620 with *nanoWatt Technology* which will simultaneously extend the RAM (4 KB) and ROM (64 KB) resources significantly. A first test showed that power consumption of the new PIC in sleep mode will be reduced by a factor of at least 2.5 which will bring the TCM 120 to the level of the MSP430 platforms.

Additionally we will reorganize the routing and pinning to make both SPI and I^2C useable and an asynchronous timer option available, which is required for enabling a controlled sleep mode while maintaining the system wide time-reference. Moreover we will develop a base platform on which the modified TCM will be mounted. This base platform will contain additionally hardware such as a 32 kHz crystal, a low power reference voltage supply, and an EEPROM. Beside it offers slots for a sensor and a power module, in which the latter is currently developed within a master thesis.

6. REFERENCES

[1] Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., and Culler, D. The nesc language: A holistic approach to networked embedded systems. In Proceedings of the ACM SIGPLAN 2003, pages 1-11, 2003.

[2] Handziski, V., Polastre, J., Hauer, J.-H., and Sharp, C., "Poster Abstract: Flexible Hardware Abstraction of the TI MSP430 Microcontroller in TinyOS," presented at the SenSys'04, Baltimore, Maryland, USA, 2004

[3] http://cegt201.bradley.edu/projects/proj2003/wisenet/index.html

[4] http://cvs.sourceforge.net/viewcvs.py/tinyos/

[5] http://cvs.sourceforge.net/viewcvs.py/tinyos/ tinyos-1.x/contrib/hsu/

[6] http://www.enocean.com

[7] http://www.gnupic.org

[8] http://xbow.com

[9] Lynch, C., O'Reilly, F., PIC-based TinyOS Implementation, In proc. 2nd European Workshop on Sensor Networks, Istanbul, Feb 2005, pp. 378-385, 2005

ZigBee-ready modules for sensor networking

Johan Lönn
Department of Science and Technology
Linköping University
SE-60174 Norrköping, Sweden
+46 11 363445
johlo@itn.liu.se

Jonas Olsson
Department of Science and Technology
Linköping University
SE-60174 Norrköping, Sweden
+46 11 363446
jonol@itn.liu.se

Shaofang Gong
Department of Science and Technology
Linköping University
SE-60174 Norrköping, Sweden
+46 11 363459
shago@itn.liu.se

ABSTRACT
Fully functional ZigBee-ready modules have been designed, implemented, and tested. One of the modules consists of both the IEEE 802.15.4 PHY/MAC and ZigBee networking stack layers, while the other module consists of only the IEEE 802.15.4 PHY/MAC stack layers. Furthermore, a ZigBee based test-application for temperature sensor networking has been evaluated by utilizing the developed ZigBee-ready modules. The modules can have a line-of-sight radio link up to 180 m, when having only 1 mW output power. It is shown that ZigBee is suitable for sensor networking where long battery life, large networks, and fast network establishment are the requirements.

Categories and Subject Descriptors
B. [**Hardware**]

General Terms
Design, experimentation and verification.

Keywords
ZigBee, modules, networks, sensors.

1. INTRODUCTION
The past several years have witnessed a rapid growth of wireless networking. However, up to now wireless networking has been mainly focused on high-speed communication applications such as the IEEE 802.11 wireless local area network (WLAN) standard. Another well-known standard focusing on low-rate wireless personal area networks (WPAN) is Bluetooth, but it has limited capacity for networking since a Bluetooth piconet only can have up to 8 active nodes. There are many wireless monitoring and control applications in the industrial and home environments that require less complexity and lower data rates than those from existing standards. For such wireless applications, a new standard called ZigBee based upon the IEEE 802.15.4 specification has been specified by the ZigBee alliance with members of Freescale, Philips, Atmel, Siemens, Samsung, Analog Devices and Chipcon, etc. Figure 1 shows the entire ZigBee stack. The ZigBee standard has the characteristics of large network capability (up to 65,000 nodes) , long battery lifetime (up to a few years), short link establishment time (15-30 ms), but low data rate (up to 250 kbps) [1] [2].

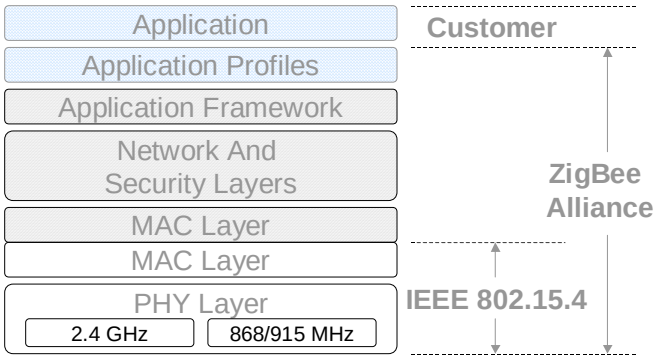

Figure 1. ZigBee stack.

2. DESIGN
The main task was to develop fully functional ZigBee modules. It was decided to develop two different modules. One module consists of both a radio frequency part (RF-part) and a microcontroller unit (MCU), which is called the "RF+MCU module". The other module only consists of the RF-part of the system, hereafter called the "RF module". This RF module can be used in an embedded system with an integrated microcontroller. The RF+MCU module is preferably used when the user wants to quickly develop a new ZigBee application. The RF-part is identical in the two modules.

2.1 RF+MCU Module
The RF part of the RF+MCU module utilizes the CC2420 chip [3] from Chipcon, while the microcontroller is the ATmega128L chip [4] from Atmel. The ATmega128L is fully capable of operating the ZigBee software stack, since it contains 128kB of Flash memory. The modules are designed to support in-system programming via the JTAG interface. The RF+MCU module has a wide range of external data interfaces, which allows the microcontroller to communicate with a wide range of different external-devices. This RF+MCU module is thus a ZigBee-ready module.

2.2 RF Module
The RF module utilizes the same RF chip from Chipcon as in the RF+MCU module, but without the microcontroller chip. In order to convert the single-ended signal from the antenna to the differential inputs of the RF-chip, a BalUn circuit is used. The BalUn circuit also matches the impedance between the antenna

and the RF-chip inputs. The connector used is the same as that in the RF+MCU module. This is an IEEE 802.15.4-compatible RF module.

2.3 ZigBee Application

Since our RF+MCU module has both a microcontroller and a connector, it is very easy to develop various ZigBee-based applications. A suitable application is to form wireless sensor networks. A sensor can be connected to the RF+MCU module. In this work, a temperature sensor has been chosen. The measured temperature is then transferred to the network coordinator that presents the value onto an LCD display. Similarly, other types of sensors can also be connected to the RF+MCU module. A larger network can then be established.

3. IMPLEMENTATION

The PCB of the two modules utilizes a standard 4-layer FR-4 board. The board has a thickness of 0.9 mm. The two internal layers are used for power and ground, while the top and bottom layers are used for routing. A copper pour connected to ground is used on both the top and bottom layers.

After processing the PCB, all components including the antenna, the RF and microcontroller chips, passive components, and the connector are assembled on the board. Figure 2 shows both sides of the assembled modules, the RF+MCU module (right) having a size of 23x40 mm, and the RF module (left) having a size of 18x25mm.

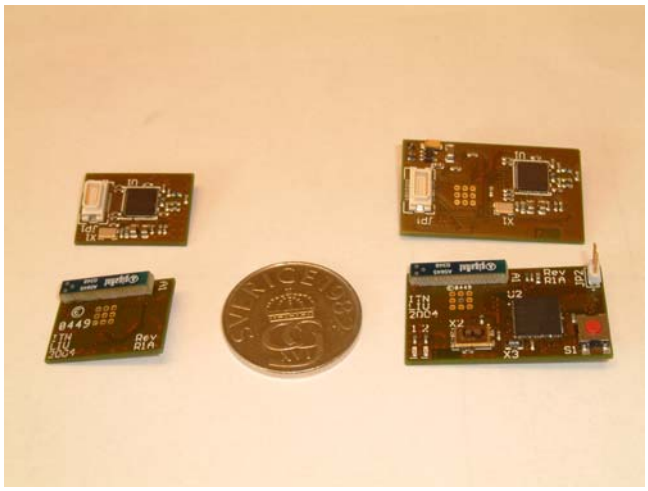

Figure 2. Assembled modules.

3.1 ZigBee Temperature Sensor Network

To evaluate the developed ZigBee-ready module, a test application with a temperature sensor and a display has been developed. A ZigBee module connected to a display, seen in Figure 3, is programmed as the network coordinator. Three other ZigBee modules, also shown in Figure 3, are connected to temperature sensors, respectively. The three ZigBee-ready modules are wirelessly connected to the coordinator, and the temperature values from the three different sensor modules are transmitted every 5 seconds to the coordinator. The values are presented on the display connected to the coordinator.

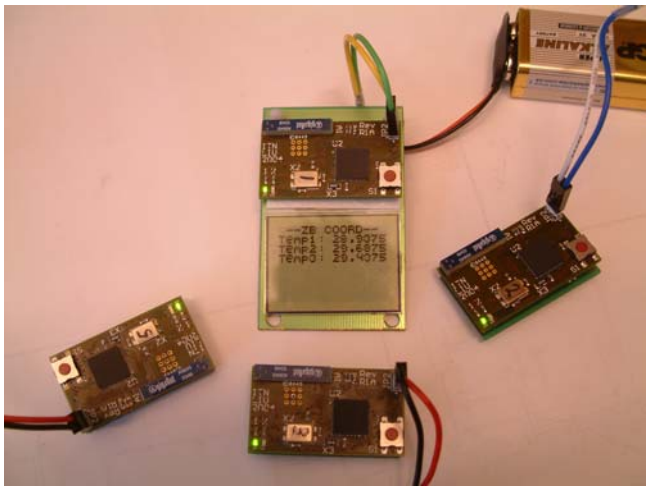

Figure 3. Test application.

4. RESULTS

This work has resulted in two different types of modules, i.e., a ZigBee-ready RF+MCU module and an IEEE 802.15.4-compatible RF module. Moreover, a test application of a ZigBee-based temperature sensor with a display has been developed. The test results show that both the RF+MCU module and the RF module work properly, and the line-of-sight (LOS) radio link can reach up to 180 m when the output power from the transmitter is only 1 mW. Some main features of these developed modules and a test application are summarized below.

- ***RF+MCU module:*** 2 USARTs, SPI, TWI (I2C) and JTAG interface. 5 digital I/O ports, and 5 pieces of 10bit ADC ports.
- ***RF module:*** IEEE 802.15.4 compatible, SPI interface.
- ***Test Application of Sensor Networking:*** ZigBee-based coordinator with a display. The coordinator initiates the network. ZigBee-based temperature sensors.

5. CONCLUSIONS

- A ZigBee-ready module consisting of RF+MCU has been developed. The module can be used for quick prototyping of ZigBee-based applications.
- An IEEE 802.15.4-compatible module has also been developed. This module can be used in embedded systems where a different microcontroller is preferred.
- A ZigBee-based test application of temperature sensor networking has been evaluated, showing the success to use our ZigBee-ready modules for quickly development of new applications.

6. REFERENCES

[1] IEEE P802.15.4, “Low Rate Wireless Personal Area Networks,” Oct. 2003, ISBN 0-7381-3677-5 SS95127.

[2] The official website of the ZigBee Alliance, http://www.zigbee.org, 2005-03.

[3] Chipcon, http://www.chipcon.com, 2005-03.

[4] Atmel Corporation, http://www.atmel.com, 2005-03.

Use of Wireless Sensor Networks for Fluorescent Lighting Control with Daylight Substitution

Fergus O'Reilly
Dept. of Electronic Engineering,
Cork Institute of Technology,
Cork, Ireland,
Tel: +353-21-4326342
foreilly@cit.ie

Joe Buckley
Dept. of Electrical Engineering,
Cork Institute of Technology,
Cork, Ireland.
Tel: +353-21-4326206
jbuckley@cit.ie

ABSTRACT
This paper examines the use of Wireless Sensor Networks interfaced with Dimmable Fluorescent light fittings to allow for daylight substitution techniques to reduce energy usage in existing buildings. This creates a wire free dimmable fluorescent system for existing buildings with minimal disruption and cost.

1. INTRODUCTION

Dimmable fluorescent fittings, using modern electronic ballast dimmers[1] are widely fitted to new buildings, to allow for the accurate dimming and control of office lighting. Factoring in natural incident daylight, allows a reduction in the artificial light (daylight substitution), giving savings between 10% and 40%. The DALI[2] light control interface provides a two wire low voltage control bus to allow the addressing and control of individual light fittings. Unfortunately, for existing buildings, the requirements to install/cable in photo sensors and the lighting control bus, makes this approach impractical, unless a full renovation is in progress. In the US, some 6 Billion square meters of work space is economically unattractive for DALI[3] for this reason.

Wireless Sensor Networks can provide work plane light measurements, which can be forwarded to a standard building monitoring system, which can through the same wireless network control the dimmable ballast elements, allowing the retrofitting of existing installations without the need to re-cable and with minimal disruption.

2. LIGHTING REQUIREMENTS

The specifications and variations required for work plane lighting, for some sample areas are shown in Table 1, full specifications are in the CIBSE Lighting Guides[4]. Individual work plane light levels are typically read and forwarded to a facilities management system which can issue control signals to the lighting elements. Replacing the standard wired approach with a wireless interface is shown in Figure 1.

Table 1: Work Plane Light Requirements

Filing - Office Work	300 lux
General Office (writing, typing)	500 lux
Fine Painting (Industry)	750 lux
Precision Assembly (Industry)	1000 lux

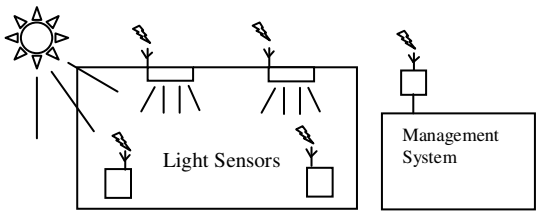

Figure 1: Wireless Daylight Substitution

3. WIRELESS SENSOR N/WS

The small unobtrusive size, achievable with sensor nodes and the ability to wirelessly network them makes them suitable to sensing light ambient levels in an office environment, and networking throughout the office for the communication of this information.

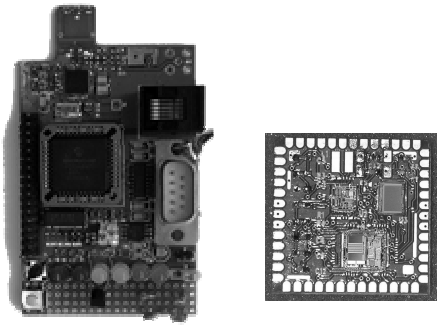

Figure 2: Sensor Test/Debug Platform, COB

3.1. CIT Sensor Architecture

The sensor node developed at CIT is shown in . It is controlled by a Microchip PIC16F877 microcontroller [5], uses a scalable clock from DC to 20MHz and a Nordic nRF903 868MHz FSK transceiver operating at up to 76.8kbit/s. Attached to it are a LDR and a thermistor.

4. DALI CONTROL INTERFACE AND SMART BALLASTS

DALI is a master/slave digital communication system that sends digital signals over a two wire bus network to provide full dimming and switching control of fluorescent lights down to the individual ballast level. The DALI forward communication protocol is 16 bits, containing both the device address and command, as shown in Figure 3. This

allows control of up to 64 ballasts individually, which can be arranged as 16 groups or as an entire network. DALI ballast reference designs and kits are available from a number of manufacturers[6][7], all have a common element of a Microcontroller driving a ballast with a T5 or T8 fluorescent tube. The micro-controller interfaces with the DALI bus and interprets the on/off/dimming commands.

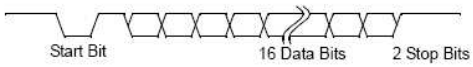

Figure 3: Dali Forward Frame

Using the micro-controller already in the DALI ballast and replacing the physical bus wiring with a radio transceiver provides an efficient extension of the ballast facilities, as shown in Figure 3. Parsing and interpretation of the DALI protocol on the PIC uses just under 2K of program memory and 71 bytes of data memory.[8]

The additional program memory available allows TDMA and hopping radio schemes back to the management system. Within a typical office environment, other fittings will be within range and the ambient light sensors will be within range of a fitment. Standard DALI allows a maximum of 64 ballasts, but extending the wireless packet format allows the additional addressing information, creating zones, using an extra byte for this purpose with 4 bits for zone identification and 4 bits for routing flags, can extend DALI to 1024 ballasts and use the wireless addressing for zone creation.

5. POTENTIAL SAVINGS

Globally lighting uses more than 2000 terawatt hours(TWh) per annum.[9] (Equivalent to 1000 power stations, producing 2.9 billion tonnes of $C0_2$ per annum.) The amount of power consumed in lighting varies widely from industrialised countries. e.g Netherlands - 5%, Israel - 19%, to developing countries, Tanzania - 86%. 64% of lighting electricity use is in the Service or Industrial Sector which are dominated by fluorescents.

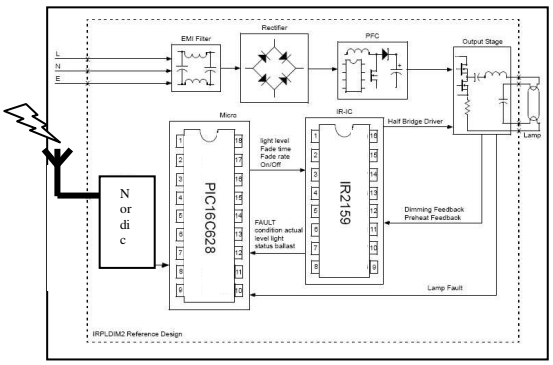

Figure 4: Block Diagram for Radio Interface DALI Dimming Ballast

If we take a typical office installation, using 1000 fittings each with two 58W tubes, used from 8am to 7pm, 2750 hours pa. With a conservative daylight substitution saving of 30% and a per Kwh cost of 10c, a savings of €9,570 per annum or €9.57 per fitting is achieved. The approx. 40% increase in tube lifespan saves another €200 on labour and €360 on materials using a wholesale tube cost of €0.90 in bulk quantities.

Other studies have shown payback in less than 3 years[10]. Installation costs of Daylight Substitution systems are typically dominated by labour costs, often in excess of 60% of the total cost. This also will be dramatically reduced with wireless networking.

6. CONCLUSION

This paper has proposed and examined a system to use wireless sensor networks to control dimmable lighting, using hopping networking strategies to transfer data to/from the photo sensors and fittings.

Integration of the wireless network elements with dimmable ballast controllers has been examined and a solution designed.

This wireless approach provides the ability to retro-fit existing lighting installations, giving significant cost savings with the corresponding environmental benefits.

7. REFERENCES

1 T. Ribbarich, J. Ribarich, "A New Control Method for Dimmable High Frequency Electronic Ballast," IEEE-IAS Con. Rec 1998.

2 Digital Addressable Lighting Interface, http://www.dali-ag.org

3 2004 Buildings Energy Databook, http://buildingsdatabook.eren.doe.gov/

4 CIBSE Code for Interior Lighting, ISBN 0900953640, http://www.cibse.org

5 Microchip Technology Inc. PIC16F877 Datasheet, Revision C, 2000.

6 International Rectifier DALI Ref. Design, http://www.irf.com/forms/eltdk.html

7 Freescale Reference Designs, http://www.freescale.com/webapp/sps/site/application.jsp? nodeId=02430ZQT84P04C

8 "Digitally Addressable DALI Dimming Ballast", AN809, http:// www.microchip.com,

9 Electrolink Magazine, Feb April 2001/ IAEEL Newsletter 1-2/00

10 IAEEL Newsletter 3-4/95

Wireless sensor networks in precision agriculture

Aline Baggio
Delft University of Technology – The Netherlands
A.Baggio@ewi.tudelft.nl

ABSTRACT

We present the initial setup of the Lofar Agro project that concentrates on monitoring micro-climates in a crop field. In addition to the agronomic experiment, Lofar Agro aims at gathering statistics on the wireless sensor network itself. These statistics will form the basis for simulations of algorithms in wireless sensor networks and will be distributed.

1. PRECISION AGRICULTURE

In the past few years, new trends have emerged in the agricultural sector. Thanks to developments in the field of wireless sensor networks as well as miniaturization of the sensor boards, *precision agriculture* started emerging. Precision agriculture concentrates on providing the means for observing, assessing and controlling agricultural practices. It covers a wide range of agricultural concerns from daily herd management through horticulture to field crop production [1, 2, 7]. It concerns as well pre- and post-production aspects of agricultural enterprises.

A facet of precision agriculture concentrates on site-specific crop management. This encompasses different aspects, such as monitoring soil, crop and climate in a field; generalizing the results to a complete parcel; providing a decision-support system (DSS) for delivering insight into possible treatments, field-wide or for specific parts of a field; and the means for taking differential action, for example, varying in real-time an operation such as fertilizer, lime and pesticide application, tillage, or sowing rate.

This article describes an experiment in field crop production, referred to as *Lofar Agro*[1]. The example application being deployed deals with fighting phytophtora in a potato field. Phytophtora is a fungal disease which can enter a field through a variety of sources. The development and associated attack of the crop depends strongly on the climatological conditions within the field. Humidity is an important factor in the development of the disease. Both temperature and whether or not the leaves are wet are also important indicators. To monitor these three critical factors, we instrumented a potato field with wireless sensors. The main goal of monitoring is to reveal when the crop is at risk of developing the disease and let the farmer treat the field or parts of it with fungicide only when absolutely needed.

In addition to the agronomic experiment, we wish to gather data and statistics on the behavior of a wireless sensor network in a real-world experiment. Furthermore, we wish to test the robustness of the energy-efficient T-MAC protocol that was developed by our group [5]. The remainder of this article describes our experimental setup and the statistics we plan to collect.

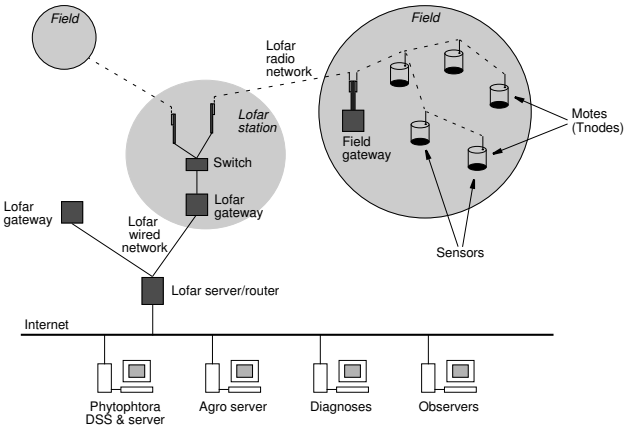

Figure 1: Lofar Agro setup

2. EXPERIMENTAL SETUP

Figure 1 shows the organization of the Lofar Agro experiment. A total of 150 sensor boards, namely TNOdes (see Figure 2), very similar to the Mica2 motes from Crossbow[2] are installed in a parcel for monitoring the crop. The nodes are manually localized so that a map of the parcel can be created. The TNOdes are equipped with sensors for registering the temperature and relative humidity. Earlier deployments have shown that the radio range is dramatically reduced when the potato crop is flowering [4]. To maintain sufficient network connectivity, 30 sensorless TNOdes act as communication relays. To further improve communication, the nodes are installed at a height of 75cm while the sensors are installed at a height of 20, 40 or 60cm. In addition to the TNOdes, the field is equipped with a weather station registering the luminosity, air pressure, precipitation, wind strength and direction. The humidity of the soil is a major factor in the development of the micro climate. A number of sensors that measure soil humidity are thus also deployed in the field. Finally, an extra sensor measures the height of the groundwater table.

[1] http://www.lofar.org/p/Agriculture.htm

[2] http://www.xbow.com/

Figure 2: A Lofar mote

A TNOde records the temperature and relative humidity every minute. For energy-efficiency considerations, the nodes are reporting data only once per ten minutes. To further save energy, the data sent over the wireless links is minimized by using delta encoding. In addition, we are currently investigating ways of compressing the data to save more energy on radio communication [3]. The T-MAC protocol cares for energy efficiency as well and impose the radio a duty cycle of 7%. The TNOdes use TinyOS as operating system. Data is thus sent using the multihop routing protocol MintRoute available within TinyOS [6]. In addition, the nodes are reprogrammable over the air using Deluge.

The data collected by the TNOdes is gathered at the edge of the field by a so-called *field gateway* and further transferred via WiFi to a simple PC for data logging, the Lofar gateway in Figure 1. The Lofar gateway is connected via wire to the Internet and data is uploaded to a Lofar server and further distributed to a couple of other servers under XML format.

3. STATISTICS COLLECTION

Beside agronomic purposes, the Lofar Agro experiment also aims at collecting live data about the behavior of the wireless sensor network itself. The goal is twofold. First, there is a need for monitoring the network while the experiment is running. Second, the data we collect should enable us to run simulations of algorithms for wireless sensor networks, such as localization algorithms.

The collected data and statistics allow us to rebuild the spanning tree used in the wireless sensor network and watch its evolution throughout the duration of the experiment. This spanning tree provides us with sensible input for simulating a wireless sensor network. Statistics also allow us to discover the dead links and nodes, check for vanishing nodes due to broken radio link because of foliage growth for example. We plan to collect information about the various neighbors of a node, parts of its MintRoute routing table, signal strength and link quality measurements, message rate and error rate in reception and transmission, and battery level. These statistics will be made publicly available.

4. RELIABILITY CONSIDERATIONS

The reliability of the wireless sensor network is of great importance as we want to prevent the loss of data and statistics. We implemented several mechanisms to make sure that all data and statistics are eventually delivered to the Lofar gateway. First, each node is logging both data and statistics in EEPROM and overwrites it only once acknowledged. Second, the Lofar gateway is checking once a day that all the expected data and statistics packets were received correctly. It uses a network-wide acknowledgment to signal the nodes which portion of their log can be safely reused.

Lost packets are handled in two ways. First, the T-MAC layer cares for up to three retransmissions of a packet. Second, dedicated software running on the Lofar gateway requests all packets that are still missing once a day. Missing packets are detected using sequence numbers. For data packets, we use a unique interval identifier composed of the node identifier, the day and a block number, which identifies a ten-minute block in a given day. For statistics packets, whose total number per day is not known in advance, we use a global sequence number per node.

The MintRoute multihop routing protocol [6] also plays a role in maintaining the reliability of the wireless sensor network. More precisely, it keeps routes estimates for each (active) neighbor and ensures that bad-quality links or malfunctioning nodes are evicted from a node's routing table and by-passed. In our deployment, we decided to let a node refresh its route estimates once per hour and switch parent if necessary.

5. STATUS AND ON-GOING WORK

At the moment, we are setting up the various connections and configuring the wired and wireless networks shown in Figure 1. A dozen of sensors are deployed in the experimentation field for test purposes. Mid-April, the potato crop was planted and the main of the sensors will soon be installed in the field. In June, we expect to have initial measurements data and statistics. The data collection will go on until September, time at which the potatoes are lifted.

6. REFERENCES

[1] J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing: sensor networks in agricultural production. *IEEE Pervasive Computing*, 3(1):38–45, Jan-Mar 2004.

[2] K. Mayer, K. Taylor, and K. Ellis. Cattle health monitoring using wireless sensor networks. In *Second IASTED International Conference on Communication and Computer Networks*, Cambridge, Massachusetts, USA, Nov. 2004.

[3] T. Schoellhammer, B. Greenstein, E. Osterweil, M. Wimbrow, and D. Estrin. Lightweight temporal compression of microclimate datasets. In *First IEEE Workshop on Embedded Networked Sensors (EmNetS-I)*, Tampa, Florida, USA, Nov. 2004.

[4] J. Thelen, D. Goense, and K. Langendoen. Radio wave propagation in potato fields. In *First workshop on Wireless Network Measurements (co-located with WiOpt 2005)*, Riva del Garda, Italy, Apr. 2005.

[5] T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *First ACM international conference on Embedded Networked Sensor Systems (SenSys'03)*, Los Angeles, CA, USA, Nov. 2003.

[6] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *First ACM international conference on Embedded Networked Sensor Systems (SenSys'03)*, Los Angeles, CA, USA, Nov. 2003.

[7] W. Zhang, G. Kantor, and S. Singh. Integrated wireless sensor/actuator networks in an agricultural applications. In *Second ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, page 317, Baltimore, Maryland, USA, Nov. 2004.

Intelligent Sensor Networks - an Agent-Oriented Approach

Björn Karlsson
Communication Research Labs Sweden AB
Varvsholmen
SE-39230 Kalmar, Sweden
+46 (0)480 423947
bjorn.karlsson@crl.se

Oscar Bäckström
Communication Research Labs Sweden AB
Varvsholmen
SE-39230 Kalmar, Sweden
+46 (0)480 423947
oscar.backstrom@crl.se

Wlodek Kulesza
University of Kalmar
SE-39182 Kalmar, Sweden
+46 (0)70 237 56 75
wlodek.kulesza@hik.se

Leif Axelsson
Ericsson Microwave Systems AB
SE-43184 Molndal, Sweden
+46 (0)70 987 63 48
leif.l.axelsson@ericsson.com

ABSTRACT

This paper describes the agent oriented programming paradigm for development of intelligent sensor networks. The case study was performed by implementing a testbed using JADE, **J**ava **A**gent **DE**velopment **F**ramework, as a basis of testing this proposed approach, and making implemental experiences of the agent paradigms maturity for dynamic wireless sensor networks. The application of the specific implementation is an **U**nattended **G**round **S**ensor **N**etwork, UGSN, for surveillance of moving targets. An intelligent sensor network in this case should consist of autonomous sensor nodes, which exchange information, reason and collaborate with each other. The UGSN should preserve energy resources and work as one unit when delivering fused and compiled sensor information to the end user.

Keywords: Unattended Ground Sensor Network, Agent Programming**,** Distributed Data Fusion, Ad hoc networks

1. INTRODUCTION

In a surveillance application scenario the situation picture from the sensor network should in many cases preferably consist of target tracks and target identities based on fused sensor information, rather than sending single and non-fused sensor readings. Since the network is decentralized, the data fusion process has to take place without the need of a central fusion node.

Decisions have to be made by the network where the fusion process spatially has to be executed. Because the targets are moving through the network and sensor information must not be broadcasted over all network nodes, the data fusion should be processed in a distributed manner. To reduce energy consumption, information in an intelligent sensor network should only be shared to nodes that can be of benefit by that specific information. This is not a trivial problem and a communication strategy must therefore exist for information dissemination between network nodes.

A sensor network with intelligent behavior is a system that can adapt to the situation, present information that is relevant for the moment and a system that has reasoning parts that are designed to function with low-level rules and work together to accomplish a high-level goal.

Focus of the intelligent sensor network has not been put on the explicit fusion algorithm but on developing a communication architecture that is suitable for distributed data fusion. With sensor data correlation and fusion, the UGSN detects targets, track movements in the area and also possibly classifies and types targets. Sophisticated and advanced sensors, in the sense of performing some degree of data processing, can with advantage be used together with more simple and low-cost sensors to achieve both target tracking and target type identification. The distributed *intelligence* of the network and fusion process must be able to easily be adjusted for new types of sensors and data entities.

2. AGENT PROGRAMMING PARADIGM

The main characteristics of an **A**gent **O**riented, AO, design are that it consists of autonomous problem-solving entities that interact with other agents using high-level communication. Regular objects communicate through simple lower level method invocations or method calls with limited range of variation in parameters. Agents communicate through a thorough declarative high-level agent communication language. Those speech acts can be informing, requesting, offering, accepting, believing, rejecting, competing, assisting and so on.

In the **O**bject **O**riented **P**rogramming, OOP, paradigm, the programming object is specified by the real object it represents by its state and the actions that can be performed by, or on, it. Similar, an agent is formally specified by its mental state consisting of its beliefs, goals, actions and ongoing interaction with other agents and its environment.

There exist numerous of different naming and terms for the properties describing an agent. Autonomy, pro-activeness, collaboration and mobility are the most important agent properties for the intelligent sensor network. *Autonomy* means that the agents have control over their own internal state and behavior. *Pro-activeness* denotes that the agent has self-starting behaviors and ability to initiate actions. The *collaborative* (or *cooperative*) behaviour means that an agent perceives its environment and other agents, and has the ability to communicate and interact/cooperate with those. *Mobility* of an agent requires the ability to migrate in a self-directed way from one host platform to another.

3. IMPLEMENTATION AND RESULTS

An intelligent ground sensor network with reasoning agents has been developed. The network is correlating sensor information and autonomously tracking targets and fusing target information. The information fusion process is made possible due to reasoning and mobile software agents and results in a higher confidence level of target entities, such as type-class and type-identification. The fusion algorithm is based on a majority voting ordinary technique, merely implemented to show the possibilities and benefits of distributed information fusion in wireless sensor networks. The network autonomously decides

when to update external applications and with what information, which results in greater situation awareness for the end user.

JADE, [5] is used as a tool for the sensor network implementation and agent runtime environment. JADE is FIPA, [3], specification compliance, open source under LGPL license and implemented in Java. The benefit of using an agent development framework results in a more mature and functional system. JADE handles the agent lifecycle, the internal agent execution and the agent communication and message handling. Due to the interaction protocols provided by the development framework, dynamic behaviours of the UGSN, such as correlation and fusion, can easily be adjusted by small means. The framework also provides all common agent tasks as message encoding/decoding, tracing, monitoring and agent thread handling. JADE makes it possible to focus even more on the business logic instead of being forced to deal with the lower communication layer and framework implementation issues. Regarding low performance systems, JADE has in projects been adjusted to be compatible with J2ME CLDC/MIDP1.0. In this configuration the memory footprint was about 100kB, [1].

The UGSN performs both decentralized and distributed data fusion. At each ground sensor, the sensor data is fused into target information corresponding to decentralized data fusion.

The collaborative and mobile properties of software agents has resulted in a mobile agent, called **T**arget**A**gent, TA. The TA is created on the host where the first sensor trig occurs. The TA is then reasoning with its neighboring nodes and correlating their sensor data in order to decide where to migrate next. The TA migrates between nodes in order to always be located close to the target it tracks. After each migration the TA is performing the fusion process in a distributed manner with available sensor information from the new hosting node. The supported interaction protocols in JADE have also made it possible for the TA to delegate sub-tasks. The TA could query other agents for assistance, e.g. ask a camera node to take a snapshot on a target and further find another sensor node to assist the camera node as a trig sensor for better timing when taking the snapshot.

4. REQUIRED IMPROVEMENTS FOR AD HOC ENVIRONMENTS

Currently the FIPA specifications have not yet been fully suitable for agents in ad hoc networks. Current FIPA abstract architecture specifies two mandatory components of an **a**gent **p**latform, AP. The first is the **A**gent **M**anagement **S**ystem, AMS, which represents the management authority of an AP and is responsible for e.g. agent life-cycle management, agent creation and deletion. The second mandatory component is the **d**irectory **f**acilitator, DF, which acts as a yellow pages directory for agent lookups. FIPA specifications do not contain descriptions for mechanisms to implement agent migration. However, JADE enables migration but only within one distributed AP.

The node hosting the AMS and DF will in a distributed AP represent a single point of failure. Using a global AMS and DF implies a heavy load on the network resources in order to maintain an updated directory of all agents.

Therefore, FIPA has to this date specified a preliminary specification of a modified AP reference model for discovery in ad hoc networks in the specifications of an Agent Discovery Service and a Discovery Middleware Specification, [2]. Each node contains a complete AP, including the AMS, meaning a node is not dependent on any other node for its agent management functionality. The AP contains an **A**gent **D**iscovery **S**ervice, ADS, by which the local agents register their services they wish to make public in the ad hoc network. The ADS searches for and/or publishes services to the network by using one or several **d**iscovery **m**iddleware, DM, available in the ad hoc network. SLP, Gnutella, Bluetooth, UPnP, Salutation and JXTA are examples on different discovery technologies. A service on a peer is found by another peer, either by a service pushing information about its presence over the network, or by a peer sending a search request through the DM. The inconsistency of found agents/services is a specific difficulty for ad hoc environments. A solution is that each registered service has a leasing time.

5. CONCLUSION

The development of an intelligent sensor network faces numerous objectives and challenges. This work has evaluated the benefits of using the agent-oriented view of software development in distributed information fusion systems.

Agents are specified by mental states such as beliefs and intentions. This way of modeling makes AO systems more capable of handling dynamic relationships, which often can be found in distributed sensor networks, than e.g. the OOP paradigm. The agent paradigm and the agent's high-level communication language together with an agent development framework, like JADE, makes it possible to realize complex distributed tasks, such as agent reasoning and sub-task delegation, by use of small means.

The agent oriented software paradigm has approved to be highly suitable in development of complex and distributed systems. The current architecture specifications of agent systems are not mature or efficient enough for ad hoc environments. However, FIPA has presented a draft for a specification of how to use discovery middleware such as JXTA in ad hoc environments to solve the need of central lookup- and management services.

6. REFERENCES

[1] Bellifemine F., Caire G., Proggi A., G. *Rimassa JADE – A White Paper* EXP – Volume 3 .n.3 –September 2003, http://exp.telecomitalialab.com/upload/articoli/V03N03Art03.pdf

[2] FIPA Agent Discovery Service Specification Version 1.2e FIPA TC Ad Hoc – 2003/10/20, http://fipa.org

[3] FIPA Foundation for Intelligent Physical Agents - Home Page, 2004, http://www.fipa.org

[4] Gorodetski V., Karasayey O., Samoilov V. *Multi-agent Data Fusion Systems: Design and Implementation Issues* St. Petersburg Institute for Informatics and Automation

[5] JADE "Java Agent DEvelopment Framework –" Home page for the JADE, http://jade.tilab.com/

[6] Labrou Yannis, Tiim Finin, Yun Peng *The current landscape of Agent Communication Languages* Laboratory for Advanced Information Technology, University of Maryland, Baltimore County – March 1999 http://www.csee.umbc.edu/~jklabrou/publications/ieeeIntelligentSystems1999.pdf

[7] Nicholas R. Jennings, Michael Wooldridge *Agent-Oriented Software Engineering* Queen Mary & Westfield College University of London, UK http://www.ecs.soton.ac.uk/~nrj/download-files/agt-handbook.pdf

[8] Shoham Y. *Software Agent - chapter: ' An overview of Agent-Oriented Programming'"* by Jeffrey M.

A Tree-Based Approach for Secure Key Distribution in Wireless Sensor Networks

Erik-Oliver Blaß
Institute of Telematics
University of Karlsruhe
Germany
blass@tm.uka.de

Michael Conrad
Institute of Telematics
University of Karlsruhe
Germany
conrad@tm.uka.de

Martina Zitterbart
Institute of Telematics
University of Karlsruhe
Germany
zit@tm.uka.de

ABSTRACT

Providing security for wireless sensor networks is a challenging task. Besides physical restrictions like limited computing abilities and available memory storage, a major problem results from the lack of any infrastructure components in a sensor network. Central components for security like eg. key-distribution centers or Central Authorities (CA) are impossible to realize. It is therefore difficult to distribute encryption keys necessary for secure communication among sensors. This paper presents a novel approach to securely distribute keys to sensors joining the network. The approach is self organizing and minimizes memory consumptions as well as radio transmissions.

1. INTRODUCTION

As wireless sensor networks are becoming more and more popular, questions about their security arise. There is a large number of scenarios where the data exchanged between sensor nodes is critical eg. in health applications. Microcontroller-based tiny hardware sensors are not capable of performing complex security protocols known from the PC/Internet world. However the major problem for sensor networks is the absence of central infrastructure components. In this context, key distribution becomes a delicate problem. A malicious node can learn or even forge plaintext keys sent from one node to the other and is thus able to decrypt encrypted messages or forge new messages. Due to the lack of infrastructures key-distribution centers or CAs are not feasible in sensor networks. This paper presents the basic idea for a new key distribution mechanism in sensor networks. It uses the fact that communication in sensor networks follows a certain tree-like scheme, *aggregation*, allowing memory efficient and energy saving but still secure key distribution. New sensor nodes joining the network are able to autonomously share the keys they need to accomplish their mission.

2. RELATED WORK

Most work dealing with key distribution in wireless sensor networks assumes that there is the need for every sensor node to be able to securely communicate with arbitrary other nodes from the network. This is a very strong assumption that might not be realistic in a real world sensor scenario (cf. section 3). A typical representative for this class of distribution schemes is presented in [1]. Every sensor node receives a huge subset of an even larger set of *pool-keys* from the user. With a certain probability it is now possible that two nodes willing to communicate have at least one common key in their subset. In such a way however a lot of memory is wasted, which is especially critical for low memory sensor devices with eg. only 4KByte RAM. Other work focuses on the use of a base station to distribute keys, eg. [2]. Depending on a base station for every key exchange is an unrealistic assumption, as in a real world sensor network this base station might not be available at all times, especially not for each and every key exchange.

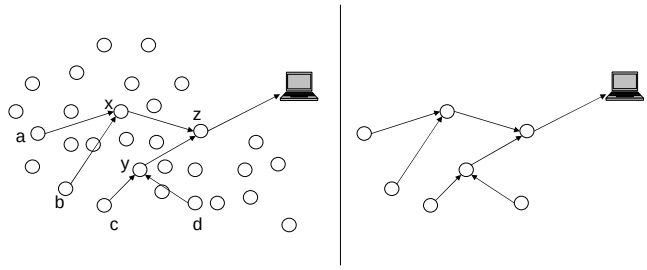

Figure 1: a) Example data aggregation in a sensor network b) Aggregation as a tree-like hierarchy

3. AGGREGATION

Typically sensors report measured data, eg. temperature, towards a *data sink*. On the way to the sink data can be aggregated by so called *aggregation nodes*. These nodes are able to collect data from other sensors nodes and process them, for example computing a mean value and forward the *aggregate* to the sink. Figure 1 a) shows an example network. Sensor nodes a and b measure the temperature in room 1 at different positions, eg. at the top and the floor, and nodes c and d measure the temperature in room 2 respectively. Represented as a laptop, the sink is however only interested in the mean temperature of the complete building. Therefore a tree-like scheme has to be established for sensor communication. Aggregation node x collects temperature measurements from nodes a and b and computes their mean value forwarding this to aggregation node z. Aggregation node y does the same for node c and d. Finally node z computes the mean temperature for the whole building, i.e. two rooms, and reports it to the sink. As shown in figure 1 b) this communication scheme forms a hierarchy, sensor nodes (vertexes) and communications paths (edges) form a graph, more precisely a tree. The question whether communication within this tree is overlay communication or

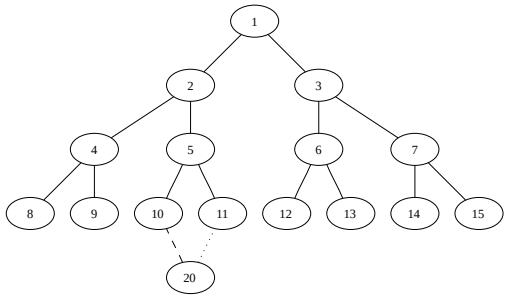

Figure 2: A sample aggregation tree

not, i.e. if there are multiple *hops* necessary to reach x from a, is negligible at this point.
Contrary to assumptions made in related work, the major observation here is that in this typical scenario, sensor nodes do not have to communicate arbitrarily with each other node. Nodes talk to each other only by a communication scheme, i.e. an aggregation tree. Sensor node a for example has to exchange data only with node x, therefore a needs a shared key with x, aggregation node y needs a key with node z. On the other hand communication is unlikely to happen between nodes from other categories like eg. communications between a temperature sensor and a light sensor. Also communication between nodes within the same category will never happen. Nodes a and b as well as c or d will most likely never exchange data among each other. Again: They might *transport* or *forward* data in multi-hop situations but there is no need for an applied *end-to-end* communication. However to ensure authenticity it will also be necessary for eg. the sink to verify that certain received aggregated data has been aggregated in a correct way, whatever that means – think of data origin verification. As an example, to verify aggregated data from z the sink will have to talk to x and y. To check whether x aggregated correctly, the sink has to talk to a and b. It is not in the scope of this work to introduce an efficient algorithm for authentic data aggregation, but important to point out that therefore secure communication has to be established not only between the vertexes in the aggregation respectively communication graph: Keys are also necessary between a vertex and its grandfather, great-grandfather and so on. So eg. sensor a will need a shared key with x but also different shared keys with z and the sink to enable possible additional authenticity verifications.

4. TREE-BASED KEY DISTRIBUTION

Let us assume for now that aggregation in sensor networks forms a complete binary tree without loss of generality. Before a new sensor joins the network, it must be paired by the user or a *MasterDevice*. The pairing is essential for the node to obtain its new position inside the aggregation tree, to identify the parent i.e. the first aggregation node. The user knows node positions because of their duty or use within the network. The distribution scheme is defined inductively. Shown in figure 2, a complete binary tree exists before a new node starts to join. This node is now paired by the user. The user identifies, that this node will have to communicate with aggregation node 10 because of its use. For this the user assigns the new node a new id ID_{node}. As the new node will be a child of 10, ID_{node} could be 20 or 21 to comply with the binary tree scheme. As the new node is the first child of 10, it becomes 20. Now 10 is the *primary parent* P_1 of 20. Furthermore the user computes a *secondary parent* P_2 for 20 using: $P_2 = \begin{cases} ID_{node}/2+1 & \text{if } ID_{node} \text{ is even} \\ (ID_{node}+1)/2 & \text{if } ID_{node} \text{ is odd} \end{cases}$
P_2 for 20 is therefore node 11. The user can now handout two *tickets* (including keys) to 20 that allow secure communication for 20 with 10 and 11. This can be done efficiently as each node in the network might share a pairwise different key with the user – thus allowing the user to securely access distinct nodes. As 20 can now establish secure channels to parents 10 and 11, it will ask them, which other aggregation nodes are on the way to the sink 1. Even in the presence of one cheater 20 will come to know 5, 2 and 1. The next step is to build secure channels to these nodes by securely exchanging shared secrets with them, first of all with 5. The main idea is, that 20 generates a new key K and splits it into two parts K_1 and K_2, these parts are encrypted with the key for 10 or 11 respectively and sent to 10 and 11 to forward them towards 5. As K_1 is encrypted from 20 with the key shared with 10 only 10 can decrypt it. Then 10 encrypts K_1 with the secret key 10 shares with 5 and sends the result to 5. On the other hand 11 does the same with K_2. Finally 5 can decrypt both transmissions from 10 and 11 and restore K. As neither 10 nor 11 comes to know the other part of K, K is finally secretly transmitted from 20 to 5 even in the presence of one malicious node. Also changing K_1 or K_2 maliciously would not help any node as this would only deny communication between 20 and 5, but impersonation attacks are not possible. To secure communication between 20 and 2 or 1, the same procedure can be repeated. 20 sends one half of the encryption key K to 10, the other one to 11. As the aggregation tree was build inductively both have already a secure channel to node 2 and can transfer their part of K to node 2 directly (10s parents were 5 and 6).
An analysis of this mechanisms complexity shows, that each setup of a secure communication channel only needs 4 symmetric encryptions and 4 communication steps inside aggregation *overlay*, completely independent from the position of the new node inside the tree or the depth of the tree. Also each node has to store only keys from other nodes, which are absolutely necessary because of its mission.

5. CONCLUSION

In contrast to related publications in wireless sensor networks, this work concludes that there is no need to distribute keys between *arbitrary* sensor nodes. Instead this paper states that such need is an unnecessary strong and superfluous assumption, as real-world sensor networks communication is often a tree-like aggregation towards the sink. This paper then presents first steps to an efficient algorithm for a secure key distribution within this aggregation tree. The algorithm uses only a constant number of symmetric encryptions and stores only keys a sensor node needs anyhow because of its mission.

6. REFERENCES

[1] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. In *ACM CCS*, 2002.

[2] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *Mobile Computing and Networking*, 2001.

Poster Abstract:Wireless Sensor Network-Based Tunnel Monitoring*

Sivaram Cheekiralla †
Massachusetts Institute of Technology
sivaram@mit.edu

ABSTRACT

In this paper we describe the development and deployment of a wireless sensor network (WSN) to monitor a train tunnel during adjacent construction activity. The tunnel in question is a part of the London Underground system. Construction of tunnels beneath the existing tunnel is expected to cause deformations. The expected deformation values were determined by a detailed geotechnical analysis. A real-time monitoring system, comprising of 18 sensing units and a base-station, was installed along the critical zone of the tunnel to measure the deformations. The sensing units report their data to the base-station at periodic intervals. The system was used for making continuous measurements for a period of 72 days. This window of time covered the period during which the tunnel boring machine (TBM) was active near the critical zone. The deployed WSN provided accurate data for measuring the displacements and this is corroborated from the tunnel contractor's data.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Wireless Communication; C.3 [**Special-Purpose and Application-Based Systems**]: Embedded Systems

General Terms

Design, Deployment

1. INTRODUCTION

The existing state of infrastructure needs to be known to apply any repair techniques. Monitoring systems are commonly used to know the existing state of the infrastructure. Nearly all monitoring systems in Civil Engineering are wire-based systems. The primary purpose of the wires is for data communication, and may also be used for powering the sensors. Wire-based systems are expensive to install and maintain, and instrumenting an existing structure with a wire-based system has some practical difficulties. Wireless monitoring systems, to a great extent are devoid of these problems. Being modular in nature, these systems can be moved to new locations as needed. The availability of low-powered and cheap computing power (microcontrollers, DSP chips etc.), radio frequency (RF) integrated circuits, and the development of new wireless standards has fueled interest in wireless sensor systems. Wireless sensor technology is one of the promising technologies of the future [5]. Active research is being done in the use of wireless sensor systems for structural and environmental monitoring. In this paper, we discuss a wireless sensor network (WSN) based tunnel monitoring application.

*This material is based upon the work supported by The Cambridge MIT Institution (CMI):New Technologies for Condition Assessment and Monitoring of Ageing Infrastructure

†Sivaram Cheekiralla is a Graduate Research Assistant at the AutoID labs, MIT.

2. TUNNEL MONITORING APPLICATION

2.1 Background

The wireless sensor system developed was deployed in a section of the London Underground tunnel system near Highbury & Islington Station. Adjacent construction activity was expected to cause deformations in the existing tunnel and a potential disruption of the tunnel services. The construction activity comprised of construction of two tunnels (dia 8.1 m) using a tunnel boring machine (TBM) beneath the existing tunnel. These tunnels were a part of the Channel Tunnel Rail Link (CTRL) project and run from Stratford station to King's Cross station.

The most likely deformation that the tunnel could undergo was sagging with the most vertical displacement directly above the construction activity. The other possible deformation being the ovalization of the tunnel's critical cross-section. The critical cross-section was directly above the construction activity. The expected vertical displacements, obtained by a geotechnical analysis, were between 10-30 mm depending upon the soil volume loss. The expected deformation due to ovalization of the critical cross-section was between 1-3 mm.

2.2 Goals of the monitoring activity

The goal of this endeavor was to use a wireless sensing system to measure vertical displacements and hence evaluate the potential of WSN for infrastructure monitoring. Electrolevels are commonly used to measure the vertical displacements in a tunnel. These are sensitive instruments that measure slope. Slope is integrated over length to get displacement. The electrolevel system is not redundant and is expensive to install. Moreover, the errors in slope measurement get accumulated in calculating the displacements. On the other hand, a wireless sensing system would be much cheaper and easier to install. The main challenge of developing a sensing system was the limited "design" and "test" time available. The whole system had to be developed and tested before a "strict" deadline.

3. DEVELOPMENT AND DEPLOYMENT DETAILS

In order to monitor the vertical displacements, a wireless sensing system was developed. One of the main requirements on the technology was non-obtrusiveness as there is little room in the tunnel

outside the dynamic envelope of the train. This ruled out the possibility of using existing technologies like laser-based surveying instruments for measuring displacements. Commercial off-the-shelf (COTS) wireless sensing systems that were considered are: Mica [4] motes and i-Beans [3]. Each of them had limitations which made us to develop our own wireless sensing system. The detailed design and development of the sensing system can be found in [2, 1].

The wireless sensing system consisted of 18 sensing units (size: 15 cm * 10 cm * 7.5 cm), a base-station and a hydraulic reference line; each sensing unit consists of a pressure transducer, microcontroller, RF transmitter and signal and power conditioning hardware. Each unit was powered by 6 C-cell alkaline batteries. The sensing units were connected to the hydraulic reference line that was laid out along the critical zone. The sensing units measured the absolute pressure at various points on the hydraulic reference line. The vertical displacements were calculated using the relative change in pressure. One of the sensing units (one farthest from the construction activity) was used as the reference for calculating the pressure changes. The base-station was a RF receiver unit connected to a laptop, which was shelved in an access shaft near the tunnel. Since, the train tunnel was operational, the system was deployed during the engineering hours of the tunnel. The engineering hours are a four hour window (1 am to 5 am) every night when the trains are not operating. It took four days and two persons to install the whole system. The system was operational from October 17, 2003 to December 29, 2003.

Each sensor unit collected a pressure reading from the transducer every 10 seconds, temperature and voltage supply to the unit every minute, and transmitted these readings to the base-station. The data was packeted using a pre-defined format and was sent to the base-station using the RF transmitter. The base-station laptop stored the data on its hard-disk and also transmitted this data to a remote server using a wireless modem. The base-station performed the calculations to measure the displacements from the pressure readings. These readings were not corrected for temperature effects. Post-processing was done to calculate the displacements [1].

4. RESULTS

The first TBM (up line tunnel) passed the site on 29^{th} October, 2003. The centerline of this tunnel passed below approximately the 53 meter mark of our horizontal alignment scale. The majority of the sensors reported a downward movement over the course of a week, followed by stabilizing. Most sensors showed a downward movement between 5 and 15 mm which is a reasonable observation. The second TBM (down line tunnel) passed beneath roughly the 33 meter mark on our horizontal scale on December 8. To calculate the displacements, the sensor data was re-zeroed just before the second tunneling took place. This removed the effects of movement after the first tunneling. Any long-term drift in the measurements of the sensor modules was also removed by the re-zeroing. The second passing of a TBM caused sensors to report a settlement between 5 and 10 mm. This is quite similar to the results of the first passing. Good correlation with the data from the tunnel contractor is found but the exact values are not publicly available.

4.1 Performance of the system

Of the 18 sensors installed, only 11 of them worked without problems. Four of the sensors reported unreliable data and the base station could receive only intermittent messages from two of the sensors. It was observed that many of the readings were lost because of the collisions that occur when two transmitters try to transmit data at the same time. This could have been avoided by using an appropriate medium access control (MAC) scheme. Because of the time constraint in designing the whole system, this was not done. In order to avoid a cyclic pattern of collisions, the sensing units were programmed so that they send data at a random interval within a window of 0.5 milliseconds after the microcontroller collects data from the pressure transducers. This decreased the likelihood of two transmitters sending the data at the same time. But this was not sufficient as is evident from the data loss. The data during nighttime hours, after being corrected for temperature effects and pressure spikes, provided results which were steady to within a millimeter over a period of several days. Pressure changes in the tunnel caused noise of several millimeters in the readings during the hours that the trains were operating. On the longer term, there was a drift of several millimeters. For the present application, where the system had to monitor a sudden event of TBM passing, an accuracy of 1 millimeter was sufficient. The six C-cell alkaline batteries ran each sensor for approximately two months before the voltage dropped too low to provide five volts to the circuit board. This was sufficient to cover the two events [1].

5. CONCLUSIONS

A wireless sensing system to monitor displacements was designed and deployed that provided good correlation with the contractor's data. It was simple to install and no maintenance of the system was required during the monitoring period. The system can be improved by using an RF transceiver instead of the RF transmitter. Alternatively, the Mica motes with sufficient modifications would improve the communication and power-management of the wireless sensing unit. For future monitoring applications, these improvements will provide more fine-grained data. This endeavor shows that wireless sensor networks have a great potential for infrastructure monitoring applications.

6. ACKNOWLEDGMENTS

The author would like to thank Jim Moriarty from LUL for allowing to install the wireless sensing system in the tunnel. Thanks also goes to all the members of the CMI team, specifically to James Brooks, Prof. Andrew Whittle and Prof. Robert Mair. Finally, thanks to the reviewers for their comments and suggestions.

7. REFERENCES

[1] J. R. Brooks. Rapid development of a wireless infrastructure monitoring system. Master's thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, June 2004.

[2] S. M. Cheekiralla. Development of a wireless sensing unit for tunnel monitoring. Master's thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, `http://web.mit.edu/sivaram/www/Sivaram-MS-thesis.pdf`, February 2004.

[3] *i-Bean brochure*. `http://www.millennial.net/pdf/MN_Brochure.pdf`.

[4] *Mica datasheet*. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf`.

[5] Technology Review. *10 Emerging Technologies That Will Change the World*, February 2003.

Simulation of Real Home Healthcare Sensor Networks Utilizing IEEE802.11g Biomedical Network-on-Chip

Iyad Al Khatib
Dept. of Microelectronics & IT
Royal Institute of Technology
Stockholm, Sweden
iyad@imit.kth.se

Axel Jantsch
Dept. of Microelectronics & IT
Royal Institute of Technology
Stockholm, Sweden
axel@imit.kth.se

Mohammad Saleh
Dept. of Microelectronics & IT
Royal Institute of Technology
Stockholm, Sweden
mohsaleh@kth.se

ABSTRACT

This paper presents a wireless biomedical Network-on-Chip (NoC) platform utilizing 54Mbps IEEE802.11g inter-NoC links for healthcare applications in home-care sensor-networks. NoC is a nano-technology microelectronic chip that consists of several processing-cores and sensors that are interconnected to form an ultra-fast on-chip network of distributed computing systems. BioNoC is a biomedical NoC designed to serve medical applications. A BioNoC is a basic block for future medical sensor networks. Patient mobility adds the need for a wireless BioNoC link in home healthcare. We concentrate on the wireless BioNoC for chronic diseases and sleep disorders as a component in a healthcare sensor-network utilizing the fastest wireless local-area-network technology available in the market, namely IEEE802.11g. One of the challenges is to enable the wireless BioNoC sensor network to converge to medical decision and warning within an acceptable time limit for critical life cases, especially when more than one BoiNoC are interconnected. We use a defined mechanism to enable many BioNoCs to interact together. The advantages of this mechanism are: the ability to connect different wireless BioNoCs in a scalable manner, increasing biomedical computational capabilities, and wireless BioNoC interoperability. We simulate a specific application of wireless interconnected BioNoCs utilizing IEEE802.11g in a home care medical sensor-network for patients suffering from apnea. Our simulations show a millisecond convergence time for the protocol for biomedical applications in case of a medical warning.

Categories and Subject Descriptors

C.2.1 **Network Architecture and Design**

General Terms

Algorithms, Performance, Design, Experimentation, Human Factors.

Keywords

Wireless Sensor Networks, biomedical application, healthcare, IEEE802.11g, Network-on-Chip, BioNoC.

1. INTRODUCTION

The demand for higher computational speeds, lower power consumption, and patient mobility in healthcare applications are continuously increasing. The NoC technology is a solution [1] for this demand mainly when used as a component in a sensor-network. NoC is made of a large set of on-chip microelectronic processing-cores and sensors that are interconnected to form an ultra-fast nano-technology network of distributed computing systems [2]. Biomedical NoCs or BioNoCs provide a solution to a growing healthcare set of needs. In order for many BioNoCs to communicate (whether with a wire or wireless protocol), we use the definition of NoC as an autonomous system (NAS) in a sensor network [3]. In healthcare, there is an increase in the number of biomedical equipment used at home, but current biomedical devices lack the ability to provide large-scale analysis, simulations and computations at the patient's location. Today's home healthcare progression is becoming a predominant form of healthcare delivery [3]. In this paper we focus on the wireless BioNoC based platform utilizing the fastest market-available wireless-local-area-network protocol (IEEE 802.11g) for one medical scenario of patients suffering from apnea, which is the most prevalent of all sleep disorders and a life-threatening condition. The monitoring of sleep apnea patients includes measuring the apnea-hypopnea index (AHI), breathing frequency, as well as the oxygen low point or destruction index (DI), which are biomedical parameters that are looked upon in this paper as just numerical values of interest to simulations. We simulate a home-care environment for apnea-patient monitoring and care using an IEEE 802.11g wireless BioNoC sensor network.

2. OVERVIEW OF WIRELESS BIONOCS

A Wireless BioNoC is a Network-on-Chip with medical sensors and processing cores that support wire and wireless communications. In this investigation we use the IEEE 802.11g as the wireless protocol that is used by the wireless BioNoCs for a home-care sensor network.

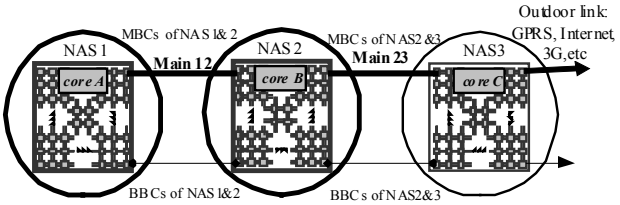

Figure 1. Interconnected BioNoCs. The MBC is Main Border Core. The BBC is a Backup Border Core. The Main and Backup links are wireless links utilizing the IEEE 802.11g. NAS stands for NoC Autonomous System, representing a self-governed-and-administered BioNoC.

The reason for choosing IEEE 802.11g is the relatively high speed that this technology provides. We use the 54 Mbps bit rate as the transmission rate in the home healthcare sensor-network of BioNoCs.

3. ENVIRONMENT SIMULATIONS

We ran a NAS topology simulation (using C++) for a biomedical home healthcare scenario, which we borrowed from biomedical scenarios using wireless Main and Backup links (with µs delays for IEEE802.11g transmissions at 54Mbps). The scenario for the apnea patient is shown in Figure 2. The patient wears two BioNoCs (1 & 2) that are interconnected via an IEEE802.11g channel, and BioNoC 2 interconnects to the Wall BioNoC (decision maker for warning) via a different channel.

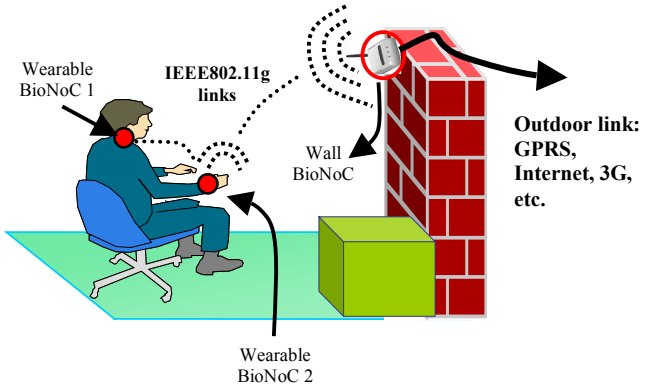

Figure 2. Apneic patient with three wireless BioNoCs: one on the head (Wearable BioNoC1), one on the hand (Wearable BioNoC2), and the Wall BioNoC, which has a wall power connection and an outdoor communication link.

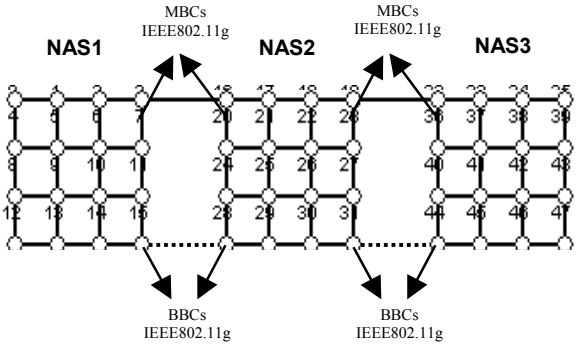

Figure 3. Wireless BioNoCs simulation environment. The numbers over the nodes reveal the hardware address of each core of the 16 cores of the BioNoC. The MBCs and the BBCs run IEEE802.11g at 54Mbps with OFDM modulation.

The Wall BioNoC is responsible for warning decisions since decision calculations consume a lot of power, and it is the only BioNoC in the scenario that is not self-powered (power connected to the wall). The Wall BioNoC is connected to the outside (to a care-center or hospital) via an outdoor communications link (Internet, 3G, or GPRS, etc). The topology of our simulated scenario of BioNoCs is shown in Figure 3. We run the scenario of interest to our project on apneic patients treatment looking for only two biomedical parameters set in the BioNoC simulator *parameters-file*: AHI and DI. AHI and DI are considered as random numerical values in the simulations by taking into consideration their physical limits. For example a value of DI below 70 would require an alarm to be sent to the hospital as it shows a severe lack of oxygen. These two parameters are sensed every hour for patients with high level of suspicion and the average is calculated every 24 hours. Accordingly the simulation is Run with three BioNoCs (Figure 3). Our simulations showed that the NoC discovery time is in milliseconds for the wireless link example of 54Mbps IEEE802.11g as shown in Table 1. The response time of the system was satisfactory from a medical point of view (Table 1).

Table 1. BioNoC simulation results for apneic patient with AHI and DI sensors of the environment shown in Figure 2.

Convergence time (ms)	Information units (# of messages)	Node of Alarm signal
41.71	1220608	1; BioNoC1
40.66	1220608	12; BioNoC1
41.30	1220608	13; BioNoC1
31.35	1220608	17; BioNoC2
30.57	1220608	30; BioNoC2

4. CONCLUSION

The paper presented an IEEE802.11g BioNoC sensor-network for healthcare with remote monitoring of patients suffering from apnea as a life threatening sleep disorder. We simulate a platform of two wearable wireless BioNoCs and a wall BioNoC. Simulations show a convergence-to-warning time that is relatively short (ms). The convergence time is satisfactory to take counter apnea measures and give audio and light warning signals to the patient.

5. REFERENCES

[1] Deb, B., Bhatnagar, S. and Nath, B., ReInforM: Reliable Information Forwarding Using Multiple Paths in Sensor Networks, *The 28th Annual IEEE Conference on Local Computer Networks (LCN),* October 2003.

[2] Tao Ye, T., Benini, L. and De Micheli, G., Packetization and routing analysis of on-chip multiprocessor networks, *Journal of Systems Architecture, vol. 50*, 2004, 81-104.

[3] Al Khatib, I., Jantsch, A., Kayal, B., Nabiev, R. and Jonsson, S., Wireless Network-on-Chip as an Autonomous System: A Novel Solution for Biomedical Healthcare and Space Exploration Sensor-Networks, *IEEE INFOCOM 05 Student Workshop, Miami, Florida, USA*, March 2005.