Sensor Networking in Aquatic Environments - Experiences and New Challenges

Thiemo Voigt*, Fredrik Österlind*, Niclas Finne*, Nicolas Tsiftes*, Zhitao He*, Joakim Eriksson*, Adam Dunkels*, Ulf Båmstedt†, Jochen Schiller‡, Klas Hjort§
*Swedish Institute of Computer Science, Sweden
Email: {thiemo,fros,nfi,nvt,zhitao,joakime, adam}@sics.se
†Umeå Marine Sciences Centre, Sweden
Email: ulf.bamstedt@emg.umu.se
‡Freie Universität Berlin, Germany
Email: jochen.schiller@inf.fu-berlin.de
§Uppsala University, Sweden
Email: Klas.Hjort@angstrom.uu.se

***Abstract*—In this paper we present the design and implementation of a small-scale marine sensor network. The network monitors the temperature in the Baltic Sea on different heights from the water surface down to the bottom. Unlike many other wireless sensor networks, this network contains both a wired and a wireless part. One of the major challenges is that the network is hard to access after its deployment and hence both hard- and software must be robust and reliable. We also present the design of an advanced buoy system featuring a diving unit that achieves a better vertical resolution and discuss remaining challenges of sensor networking in aquatic environments.**

I. Introduction

Marine research requires studies in remote environments such as the Baltic Sea. However, today's marine environmental monitoring is expensive, with a daily cost of operation at sea of at least 10.000-15.000 Euro when using a large research vessel. In addition, there are costs of laboratory technicians, analytical instrumentation and all the logistics around that. Because of these costs, current monitoring of the environment has a low resolution both in time and space. For example, in the whole Bothnian Bay (from land to Haparanda) there are nine fixed stations, visited eight times per year, and the situation is similar in the whole Baltic proper, in the combined Kategatt and Skagerrak.

Sensor networks consist of computing devices with sensing capabilities that collect data and transport it to a base station often using wireless technologies. For marine research, deploying a sensor network could dramatically improve access to real-time data covering long periods and large geographical areas. The two main challenges are the marine environment that requires water-proof components and that such a network, once it is deployed is hard to access. Therefore, both the hardware and the software for such a network must be carefully designed.

In this paper, we report about the design and implementation of a sensor network designed to collect temperature data in the Baltic Sea. We present the specifically designed hardware, that enables cost-efficient marine data collection. We discuss our software design, in particular the robust and energy-efficient synchronization protocol. We also demonstrate how the process structures on top of the event-driven Contiki operating system [1] enable the design of robust software. Due to a bug discovered during the calibration phase, the network has not been deployed yet. One of the messages of the paper is that we confirm Langendoen et al.'s experience about real-world deployments that is "Test,test,test" [2].

We also present the design of an advanced buoy system for the next generation of buoy systems. The described system features a diving unit that achieves a high vertical resolution using only one set of sensors. Furthermore, the design implies that the diving unit is parked in a garage while not sensing which inherently eliminates fouling, i.e. the accumulation of material on hard surfaces in aquatic environments.

The rest of the paper is outlined as follows: After presenting the hardware in the next section, we focus on our software design in Section III. Section IV discusses our experiences, initial results and lessons learned. In Section V we present an advanced buoy system and discuss future challenges we see in this exiting field of marine sensor networking. Before concluding we describe some related work in Section VI.

II. Hardware for Wired and Wireless Communication

In this section we describe the hardware entities of the small-scale marine sensor network. The network consists of four different entities: slave sensor nodes that measure temperature using an on-board temperature sensor, chain masters that collect data from the slave sensors and send it to an ESB node [3] functioning as a data collector and a GPRS module that transfers data onshore to a server. The first three entities are developed by FU Berlin, both the slave sensor nodes and the chain masters specifically for our project.

A. Hardware Entities

a) Slave sensor nodes.: The slave sensor nodes (see Figure 1) feature a Texas Instrument MSP 430F1232 low-power

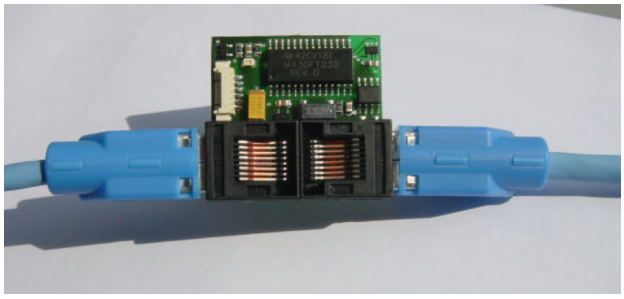

Fig. 1. Slave sensor node

microcontroller with 256 Bytes RAM and 8 KB Flash memory. On-board is an LM 71 temperature sensor and two RS485 connectors. As shown in the figure, the RS485 connectors are used to chain together a number of slave sensors.

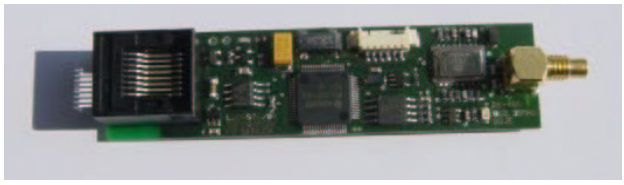

Fig. 2. Chain master

b) Chain masters.: Similar to the standard ESB nodes, the chain masters feature a TI MSP430F149 with 2 KB RAM and 60 KB Flash memory. Furthermore, the chain master has an RS485 connector used to attach the slave chain. The chain master can control the power supply to the slaves on the chain via this RS485 connector.

c) ESB embedded sensor board: The ESB nodes from FU Berlin, now sold by FU Berlin's spin-off Scatterweb GmbH, are used by many universities and research institutes in particular in Europe. Besides the low-power TI MSP430F149 micro-controller the nodes have a large number of on-board sensor including a passive infrared sensor for motion detection, vibration sensor and microphone. ESB's also have a serial interface.

d) GPRS module: In order to transfer data onshore we use a GPRS module. The Wavecom Fasttreck GPRS module includes a TCP/IP stack and features several low power modes as well as a serial interface. Using this serial interface, an ESB node can control the GPRS module and e.g. put it into a low power mode using the available AT commands.

B. System Architecture

The small-scale network shown in Figure 3 consists of two buoys. On the first buoy called *simple buoy*, we have one chain master and a chain of slave sensors. The second buoy called *king's buoy* is responsible for transferring data onshore so that the marine scientists can access it. Besides the chain master and its associated slave sensors, this buoy also contains an ESB node *GPRS connector* and the GPRS module. The latter two

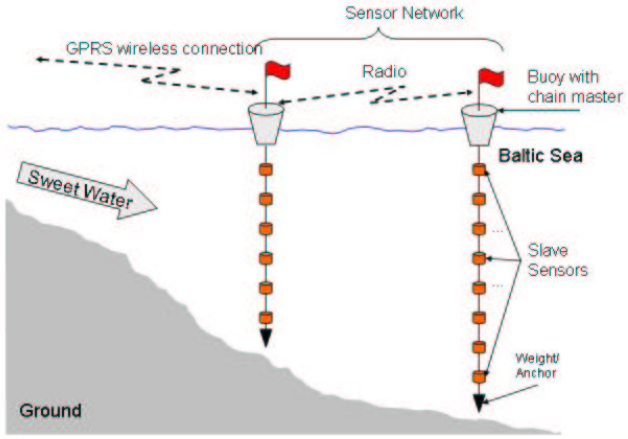

Fig. 3. System Architecture

entities are connected via their serial interfaces. By sending a series of AT commands to the GPRS module, the ESB nodes makes the GPRS module send the sensor data onshore and puts its into sleep mode.

III. Software

This section describes the software used in the sensor system. The challenging problem is that the sensors are not (or only with major difficulty) accessible after deployment. Therefore, the software must be as robust as possible. Our system is required to send data onshore about every hour.

e) Contiki: All sensor nodes run the Contiki operating system [1] developed at SICS. Contiki is an advanced operating system for small sensor nodes that has e.g. support for loadable modules and contains a TCP/IP stack. Furthermore, Contiki makes use of protothreads, extremely lightweight stackless threads designed for severely memory constrained systems [4]. Protothreads provide linear code execution on top of the event-driven Contiki OS. Through the use of protothreads, Contiki provides software processes similar to the standard process abstraction used in traditional operating systems such as Linux. Contiki also puts the processor into sleep mode when there are no runnable processes.

f) Application design: The basic application design of our system is shown in Figure 4.

The application running on the chain master on the simple buoy is implemented as one process that performs an infinite loop collecting the data from the slave sensors, transmitting it to the chain master on the other buoy and on reception of the acknowledgement goes to sleep for one hour. Since Contiki contains a TCP/IP stack we use the UDP protocol for communication between the two chain masters. Erroneous packets are discarded by the TCP/IP stack and need to be retransmitted as described in the next section.

The application on the king's buoy's chain master consists of two processes. The first process' task is to collect data from the slave sensors and to receive data from the other buoy while the other process' task is to send the data collected by both

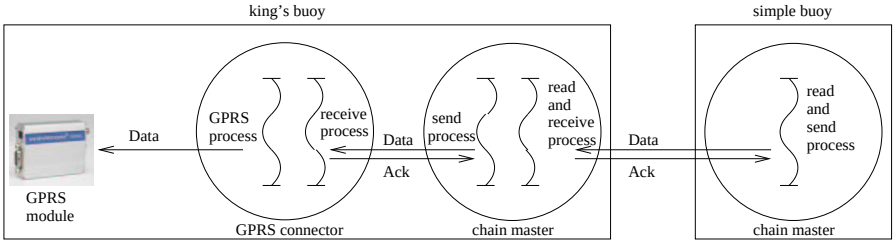

Fig. 4. Application design

chain masters to the GPRS connector. For the communication between chain master and GPRS connector we also use UDP but on a different UDP port. One of the nice features of using Contiki and IP is that processes only receive an event when a packet arrives on a port they opened themselves. For example, the chain master's send process does not receive an event when a packet from the simple buoy arrives. This feature of Contiki has proven valuable since it simplifies event handling enabling easier design of robust software.

The GPRS connector is implemented using two processes, one to receive data from the chain master and one to connect to the GPRS module. The receiver process turns its radio on for one minute or until it receives the data. After receiving the data or when it turns off the radio after one minute, the receiver process posts an event to the connector process. On reception of this event, the connector process instructs the GPRS module to send the data or an error message in case no data was received. After having posted the event to the connector process, the receiver process turns off the radio for one hour.

g) Robust Time Synchronization: All nodes in our system are battery-driven and hence we need to save energy. Since for typical sensor nodes the radio is the major energy consumer [5], we need to turn off the radio as often and long as possible. However, nodes need to turn on their radio when their peer is sending data. As the clocks on the nodes may drift and nodes may temporarily fail, there is a risk that the nodes get unsynchronized. Hence, a simple but energy-efficient synchronization protocol is required.

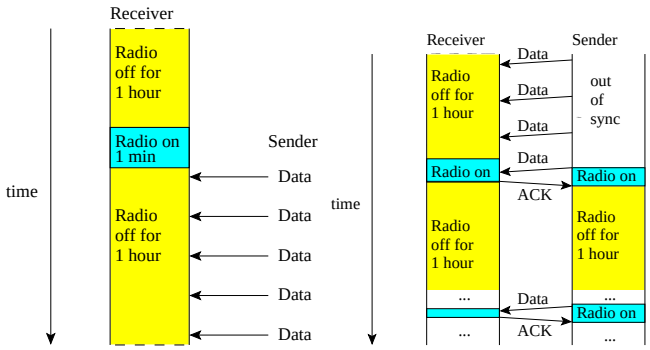

Fig. 5. Synchronization principle (left) and resync (right)

The basic idea behind our synchronization algorithm is depicted in Figure 5. We assume that nodes sleep for one hour and then turn on their radio for little more than one minute or until they receive the expected data, whatever happens first. A sender that does not receive a reply and therefore considers itself out of sync resends its data every minute. Since the receiver has its radio on for more than one minute, the sender will eventually become synchronized again.

When the receiver receives data it acknowledges it and turns off the radio for one hour. On reception of the acknowledgement, the sender can turn off its radio. As shown on the right part in Figure 5, when the peers are synchronized the radios of both peers need to be on for a short time only.

Since the chain master on the king's buoy has to wait for an acknowledgement of the ESB node, it cannot turn off the radio immediately after having sent an acknowledgement to the chain master on the simple buoy. This means that in reality, the chain master on the simple buoy goes to sleep earlier than the chain master on the king's buoy and its first data transmission after waking up does not get acknowledged. Hence, it does not wait one minute until it sends another data packet but sends data packets in short intervals after waking up. Currently five within a short interval of one second and then another five within an interval of 20 seconds if it does not receive an acknowledgement earlier. Note that sending more than one packet is important in case a packet might be lost. For the same reason, the receiver of a data packet sends more than one acknowledgement before going to sleep for one hour.

IV. Experience, Initial Results and Lessons Learned

Two slave sensor chains with five sensors each have been infused at Umeå Marine Sciences Centre, who also encapsulated the other entities as well as the batteries. Figure 6 shows the encapsulated simple buoy and the infused sensors. Figure 7 depicts a complete buoy with floats.

The system was calibrated in water with controlled temperature. During calibration we could confirm that the system was indeed water-proof and functioned correctly in warmer water. However, it turned out that the system did not produce any data when the temperature fell below 10 degrees Celsius. When we previously tested the system during summer 2006, the temperature had never been below 10 degrees and hence this bug had not occurred during our tests. Debugging revealed that the bug was due to `sprintf` not working correctly on the slave sensor nodes when the `temp` parameter in `sprintf(buf,`

Fig. 6. Encapsulated node and simple buoy

Fig. 7. Complete buoy

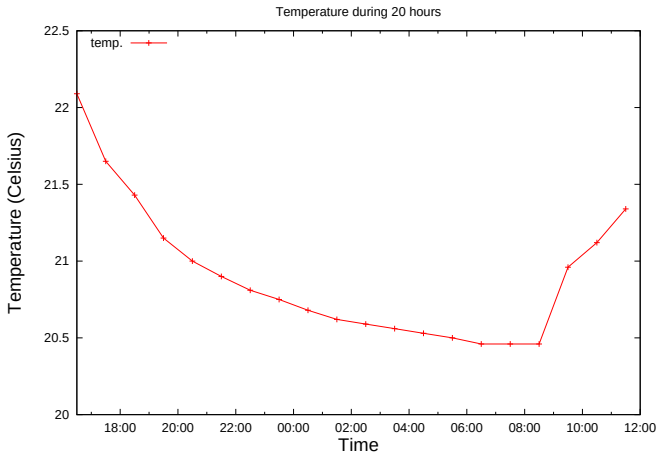

Fig. 8. Measured temperature in an office

`¨%d:%04d¨, node_id, temp)` started with 0, i.e. when the temperature was below 10 degrees.

Fortunately, we had not infused all available sensors. Therefore, we still had enough slave sensor nodes for debugging since we could not reprogram the slave sensors that had been infused. After correcting the bug we have tested a small sensor chain in the deep-freezer. The reported temperature was minus 15 Celsius, a reasonable temperature for a deep-freezer. We also put a temperature sensor outside the office's window and saw that the slave sensor node actually delivered values below 10 degrees confirming that we had removed the bug.

Furthermore, we ran a 20 hour test in an office at the Swedish Institute of Computer Science with the complete setup, including the GPRS module. The results taken from one temperature sensor show that after the researcher had left the office, the temperature decreased as expected during the night while raising again when the person returned to the office the next day.

After these new tests, we are comfortable that the system after being infused will send initial data from the Baltic Sea soon. But we can confirm by own experience that Langendoen's learning "Test, test, test" [2] must not be ignored. Furthermore, we have learned that it is of utmost importance to design for several iterations of the system. In our case this means to infuse in a way that the connectors used for programming sensors are relatively easy to access. If you have hardware specifically developed for your project do not use all your hardware in the first iteration but make sure you have enough pieces for future iterations.

V. Design of an advanced buoy system and future challenges

In this section we first discuss the design of an advanced low-cost buoy system. Then we discuss some future challenges for sensor networking in aquatic environments.

A. Design of an advanced low-cost buoy system

We are currently designing a more advanced buoy system shown in Figure 9. One of the main design choices is that this system will feature a so-called diving unit, i.e. a sensor carrier consisting of several sensors connected to one sensor node that moves up and down along an anchor line. The diving unit enables vertical profiling, i.e. taking measurements at different depths, with only one set of sensors. We will use a simple ballast system to drive the sensor carrier up and down along the fixed anchoring line. Currently, it is unclear if we need to stop the diving unit for taking measurements or if a slow sink speed is sufficient. This depends mainly on the response time of the sensors. A simple pressure sensor can determine at which depth a measurement is taken.

The buoy features a so-called garage that hosts the diving units when it is not outside taking measurements. We hope that parking the diving unit in the garage while not sensing inherently eliminates fouling, i.e. the accumulation of material on hard surfaces in aquatic environments. When the diving units returns to its garage it transfers the collected measurements to another node on top of the garage.

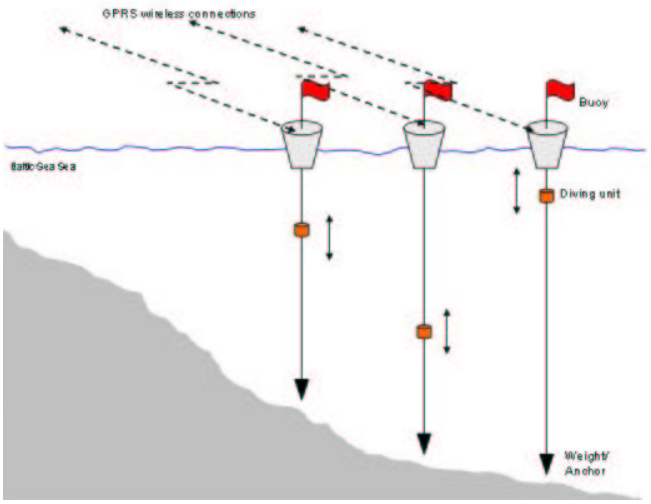

Fig. 9. Advanced buoy design

While multi-hop communication between different buoys using short-range communication is feasible as shown in the previous sections, there are some efforts to develop sensor boards with on-board GPRS modules. Therefore, we decided that we will equip each buoy with a GPRS module.

Unfortunately, there is a lack of sensors that are real-time capable. Many interesting water-proof sensors are merely data loggers that need to be programmed to collect measurements at fixed intervals, rather than reprogrammable sensing devices.

B. Remaining challenges

Besides fouling and a lack of real-time capable sensors there are other remaining challenges. On-line calibration of aquatic sensors and methods to reduce sensor drift are of high importance for remote sensors. Existing sensors as well as communication using long-ranges wireless communication such as GPRS are power-hungry. Therefore, energy harvesting is a must. However, so far energy harvesting in aquatic environments has mainly been restricted to solar cells. However, in spring and autumn the efficiency of solar cells is not very high in Northern countries. Therefore, we envision a harvesting unit that tandems a wave generator and solar cells. In order to keep track of the available energy, sensor nodes must be able to track the available energy. We have developed and evaluated a software-based on-line energy estimation mechanism and implemented in the Contiki operating system [6]. While this mechanism is able to measure to estimate the energy consumption of a sensor node with sufficient precision, further work is required to include harvested energy in the model. Once an extended mechanism is in place, an intelligent energy managing unit can be developed that is able to decide which tasks to execute when there is not enough energy available to execute all desired tasks.

Furthermore, compression of sensor data will be of high importance. While some algorithms exist that are feasible to implement on resource-constrained sensor nodes [7] more work is required. For more intensive monitoring of aquatic areas we envision that a large number of buoys will be used. Managing these remote via long-range communication networks will be an interesting challenge. Further issues are anti-theft alarm and theft prevention. For example, a network described by Ong et al. [8] was in the lake for two days only since the developer's were afraid of theft. Here we envision a combination of GPS and other mechanisms. Another real-world requirement are more advanced methods for collision warning to avoid collisions with ships.

VI. Related Work

Habitat and environmental monitoring have been one of the most popular applications for wireless sensor networks. Therefore, we can only mention a subset of these efforts here. The network on Great Duck Island was one of the first and most well-known sensor networks deployed to learn more about the behaviour of a specific bird [9]. Another prominent environmental monitoring system is Glacsweb deployed to monitor glacier behaviour in Norway [10]. Pendock et al. describe a wireless sensor node for monitoring the movement of bats between their nesting boxes [11].

Werner-Allen et al. have deployed a sensor network to monitor a volcano in South America [12]. They encountered several bugs in TinyOS after the deployment. For example, one bug caused a three day outage of the entire network, other bugs made nodes lose time synchronization. Besides the already mentioned project by Langendoen that encountered severe problems in deployment [2] other deployments had severe problems: in a surveillance application called "A line in the sand" [13], a subset of the nodes would detect false events exhausting their batteries early. Lakshman et al. report that a node with a hardware problem caused highly spatially correlated failures [14].

Our effort is not the first application of wireless sensor networks in water. Tateson et al. [15] have deployed a wireless sensor network at a sandbank off the coast off Great Yarmouth to study active sedimentation and wave processes. While their solution features a number of different sensors, our system provides higher vertical resolution by deploying sensors at different depths. Another aquatic sensor network deployed in water is presented by Ong et al. [8]. Also this network did not have a high vertical resolution. Neither Ong's nor Tateson's network did harvest wave energy.

A related emerging research field are underwater sensor networks with acoustic communication [16], [17]. One of the main advantages of these is that underwater communication makes it possible to comprise moving objects. So far there is, however, only limited practical experience with these networks.

VII. Conclusions

In this paper we have reported on a small scale marine sensor network designed to measure temperature in the Baltic Sea. The network consists of a set of slave sensors that are chained together under the water surface and powered by a

chain master sensor node deployed in a buoy above the sea level. We have described the implementation of robust software on top of the Contiki operating system in particular the simple energy-efficient synchronization algorithm. Just before deployment during calibration of the sensors, a previously unknown bug was discovered that had not been observed during testing. After correcting the bug and new tests under different conditions we expect that the system will deliver initial data without further problems.

We have also presented the design of a new marine buoy system featuring a diving unit equipped with a set of sensors and discussed some challenges around marine sensor systems.

ACKNOWLEDGEMENTS

This work was financed by VINNOVA, the Swedish Agency for Innovation Systems.

REFERENCES

[1] 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, Nov. 2004.

[2] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture," in *14th Int. Workshop on Parallel and Distributed Real-Time Systems*, Rhodos, Greece, 2006.

[3] J. H. Schiller, A. Liers, H. Ritter, R. Winter, and T. Voigt, "Scatterweb - low power sensor nodes and energy aware routing." in *HICSS*, 2005.

[4] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali, "Protothreads: Simplifying event-driven programming of memory-constrained embedded systems," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys 2006*, Boulder, Colorado, USA, 2006.

[5] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava, "Energy aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 40–50, Mar. 2002.

[6] A. Dunkels, F. Österlind, and N. Tsiftes, "Software-based on-line energy estimation for sensor nodes," in *Fourth Workshop on Embedded Networked Sensors*, Cork, Ireland, Jun. 2007.

[7] C. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks," in *Proceedings of ACM SenSys 2006*, Nov. 2006.

[8] K. O. et al., "A wireless sensor network for long-term monitoring of aquatic environments: Design and implementation," *Sensor Letters*, vol. 2, no. 1, 2004.

[9] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *First ACM Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, Atlanta, GA, USA, Sep. 2002.

[10] P. Padhy, K. K. Martinez, A. Riddoch, H. Ong, and J. Hart, "Glacial environment monitoring using sensor networks," in *Proc. of the Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*, Stockholm, Sweden, Jun. 2005.

[11] G. Pendock, L. Evans, and G.Coulson, "Wireless sensor nodes for monitoring bats," in *Proc. of the ACM Workshop on Real-World Wireless Sensor Networks (ACM REALWSN'06)*, Uppsala, Sweden, Jun. 2006.

[12] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, USA, 2006.

[13] A. A. et al., "A line in the sand: a wireless sensor network for target detection, classification, and tracking." *Computer Networks*, vol. 46, no. 5, pp. 605–634, 2004.

[14] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. D. Yarvis, "Design and deployment of industrial sensor networks: experiences from a semiconductor plant in the north sea." in *ACM SenSys*, 2005, pp. 64–75.

[15] J. Tateson, C. Roadknight, A. Gonzalez, T. Khan, S. Fitz, I. Henning, N. Boyd, C. Vincent, and I. Marshall, "Real world issues in deploying a wireless sensor network for oceanography," in *Proc. of the Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*, Stockholm, Sweden, Jun. 2005.

[16] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li, "Research challenges and applications for underwater sensor networking," in *Proceedings of the IEEE Wireless Communications and Networking Conference*. Las Vegas, Nevada, USA: IEEE, April 2006, pp. 228–235.

[17] R. Jurdak, C. Lopes, and P. Baldi, "Software acoustic modems for short range mote-based underwater sensor networks," in *Proceedings of IEEE Oceans Asia Pacific*, Singapore, 2006.