The SECOAS Project—Development of a Self-Organising, Wireless Sensor Network for Environmental Monitoring

Matthew Britton and Lionel Sacks
University College London
{mbritton, lsacks}@ee.ucl.ac.uk

Abstract

In this paper, we examine the SECOAS project which aims to design and field a self-organising, wireless sensor network for environmental monitoring. We examine the background of the project and follow the design through to an implementation in field trials. We show how implementation issues led to several significant changes in the direction of the project. We explore one aspect of the project—the operating system kOS—in detail, and explain the rationale for various design decisions. The lessons learnt during this process will be of interest to other researchers and developers in the areas of wireless sensor networks.

1. Background

Currently, oceanographic studies frequently involve using large, expensive devices to log data, typically for several months at a time. During each deployment, there are substantial risks that the platform will be damaged or destroyed [2]. This approach also has the disadvantage that the sensors measure environmental phenomena at only one physical location.

An alternative is to use a network of sensors to build a spatio-temporal picture of the environment. This leads to a system that is robust even when nodes are destroyed or the network topology changes. Furthermore, nodes can be added at will or reconfigured for various purposes. The availability of increasingly low-cost microprocessors and radio devices has made this approach feasible from an economic point of view.

2. Introduction

With these developments in mind, we proposed a project that would investigate a range of novel and emerging technologies needed to create self-organising networks of microcontroller-base nodes, integrate the best ideas into a sensor network, and prove that the network can be used by scientists to meet the needs of a dynamic and challenging sensing application. We focussed on science as the first application because environmental monitoring is of immense importance in improving the science of climate change, and because this simplifies requirements in areas such as security—yet is challenging enough to adequately test the fielded system.

Our initial objectives [1] are shown below. As we will show later, practical considerations necessitated re-evaluations leading up to the first field trial.

- To develop and demonstrate decentralised algorithms to enable automated adaptation to failures, upgrades and requirement changes in a distributed network of microcontroller-based sensors
- To investigate and demonstrate novel cooperative adaptive data handling techniques
- Design lightweight, low-power, *ad-hoc* wireless communication protocols for a range of physical layer media and end-to-end guarantees for network services
- To explore methods for implementing locally intelligent sensors capable of dynamic self-configuration
- To develop and test an appropriate control interface for scientific user communities
- To demonstrate and prove the new technologies in a realistic application context
- To undertake a major re-evaluation of environmental sensing field methodologies, and design new approaches that fully exploit the new technology

The first point in the list above states a requirement for the system to be responsive to high-level user-forwarded policies. As we shall see later, this requirement proved to also be a mandatory for implementation testing purposes. In contrast to most

existing work, the proposed system was also designed to enable sharing of processing load and establishment of consensus amongst groups of devices. This "collegiate" behaviour is crucial when the devices are heterogeneous, or when the load varies across the network, both of which are expected in natural environments.

From this initial plan, the necessary components in an overall system can be designed. In Figure 1 we show the top-level view of system components. As can be seen, there is one base-station node on the land-side of the system. A small network of floating sensor nodes is placed in the ocean for sensor monitoring. We examine the components in more details below.

We initially decided to design the system to support the collection of a range of environmental parameters, including temperature, pressure, salinity, turbidity and current direction. From our project objectives above, we designed the system to forward sensor data to the shore-based node.

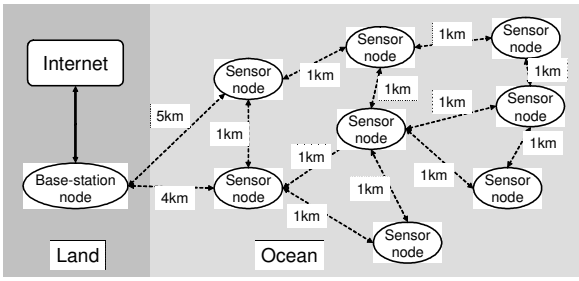

Figure 1. Top-level view of system components, showing typical distances

3. Project Organisation

The project involved three industrial and three academic partners [15]. In the first collaboration, a small company—whose experience is in sensor packages for water monitoring—built locally-intelligent sensors which adapt to local changes, such as remaining battery power. These sensors also execute a distributed application that communicates with other nodes in order to adapt to network conditions.

University College London, together with a large industrial research partner collaborated to implement various adaptation, data handling and control features. This included building a simple operating system (kOS) [3][4][12] and various applications [5][7][8][9][10] to handle sensor data and radio packets. A custom operating system was designed due to several unique features of our approach, as described in [3]. A robust, self-configuring networking layer was also designed in this collaboration [5].

A third small industrial partner, together with an academic partner collaborated to build the wireless communications infrastructure necessary for networking the nodes. As well as investigating physical layer properties, work here also focussed on the design of a self-configuring network layer, as mentioned above.

A separate academic partner provided our primary expertise in oceanography. This was necessary firstly to drive the project with reasonable user requirements, and secondly to provide domain knowledge when examining the integrity of the sensor data.

Most of the project partners began significant contributions around mid-2003, with the first field trial scheduled for summer (August) 2004. A second (full) field trial was scheduled for summer 2005, with project completion by early 2006. Both field trials were planned to be conducted at the Scroby Sands site, off the Norfolk Coast of the United Kingdom. Other minor field trials were planned to test various aspects of the system.

4. Initial research and simulation

Most of the applications designed to execute on kOS were already simulated algorithms when development began in the project (such as the gossiping/"firefly-flashing" protocol). Below we briefly examine some kOS applications and look at the research conducted into radio communications.

4.1. Information dissemination

All application objects need to communicate with (at least) their instances on radio-adjacent nodes to be most effective—that is, they are inherently *distributed applications*. For data that must be shared across the network, we have designed a gossiping/"firefly-flashing" transport protocol [10] using hash functions [13]. In this scheme, objects exchange small hash functions (data "signatures") between nodes to determine whether data stored on nodes is the same. If differences are detected, nodes increase their exchanges until data is identical. Using this scheme, whole-network traffic naturally adapts to local changes—traffic increases when differences between nodes are detected via hash function exchanges, and returns to normal after exchanges occur.

This algorithm was also simulated extensively prior to the start of the SECOAS project [10], and is also a continuing research effort.

4.2. Quorum sensing

Quorum sensing is based on the idea that significant power savings can be made by separating the network nodes into quorums (clusters), each of which nominates a node to send data on the quorum's

behalf. In our case [9], quorums are not based on arbitrary geographical or topological boundaries but are rather based on the structure of the observed parameter of interest. This results in energy savings, as nodes which observe similar parameters do not report their findings individually but use representative nodes to report parameters.

Prior to the SECOAS project (and continuing), simulation work was carried out to observe the response and limitations of this algorithm using various environmental models of the coastal environment. These models were designed to capture the wave variations that occur during various weather conditions. The cluster quality, used energy and execution time of the algorithm was then compared with other established clustering mechanisms. This algorithm is also implemented in our hardware nodes across an emulated network.

4.3. Auto-location

Auto-location is a necessary service for other applications, such as spatial data fusion and quorum sensing. It is inherently difficult to perform on sensor nodes due to limited power and processing capabilities. Therefore, our efforts have focussed on developing simple methods to estimate position, based on a ranging capability between nodes. We have simulated this algorithm [14][7], and also implemented the algorithm across various networked nodes.

4.4. Data fusion

It was important to analyse known environment data from the Scroby Sands site in order to properly dimension our solution. For example, we wished to understand the dynamic ranges, rates of change and correlations between the modalities of various parameters of the environmental parameters in order to scale the sampling functions accordingly. This also affected the ways in which we planned to spatially and temporally compress the data, as described in [14].

4.5. Adaptive Sensing

The design of our adaptive sampling policy [8] involves the use of both local and distributed information. Local information (such as battery power remaining and queue length) is used to modify local behaviour. Parameters defining this local behaviour and a related fitness measure are passed through the network. This has the effect of *spreading types of behaviour* of the sensing application, so the whole network adapts to the conditions of the environment and nodes. This work also involved significant amounts of simulation work [8] before it was successfully implemented into the sensor module of the prototype system.

4.6. Radio Communications

Several trials were conducted to gauge to characteristics of radio transmissions at sea. This was done for several sea states. Bit-error rate correlations and receive signal strength were calculated with several antenna designs and inter-node distances.

5. Initial design

In this section we briefly show the initial system design decisions.

5.1. kOS

The design of kOS was built upon several guiding principles. Note that a more thorough examination of the kOS design can be seen in [4]. Our design principles are shown below:

- Modularity of object design—we defined a simple messaging interface between both system and application objects. This decouples object *function* from *location*. For example, the adaptive sensing object could just as easily be located on the kOS board or the sensor module. This allows configuration changes to be easily made, for example from separate boards to a monolithic device (vice-versa). Development is also greatly simplified
- Simple execution model—this involved a single threaded execution model for all objects, except for occasional services of communication devices. This eliminates most context information
- Power awareness—using firefly/gossiping and adaptive scheduling, as well as other standard techniques such as using shutdown features, the kOS adapts to the lowest power usage as possible under the circumstances. We may then extend the system lifetime; keeping battery sizes small, for example
- Simple processing load control—by controlling "iterating" application objects, such as quorum sensing and auto-location, we can scale the processor load to local conditions, as explained in Section 6.2

The kOS was designed to execute on a microcontroller, as it needs access to interrupts, a USART for communication, timers, logical ports, etc, and must be debugged and re-programmable. We chose the Microchip PIC 18F452 microcontroller [6], as it (amongst other candidates) (i) has the necessary

features we wanted; (ii) was already used by one of our industrial partners and (iii) had a large user community ensuring ongoing access to timely support.

5.2. Networking

For networking, our design used a system of *inducing transmissions* to form a hierarchy of nodes, as explained in [5]. In this scheme, the base-station node sets the receive and transmit timing of all nodes in the network by forming a hierarchy of nodes with inducing transmissions—signals that force listening nodes into particular operational time-slots. Once the network topology is functional, policies flow to the network from the base-station, whilst data flows back to shore from the sensor nodes.

5.3. Radio communications system

The radio was designed to operate in the ISM band at 173 MHz, to operate in half-duplex at a 2% duty cycle. The design was for a maximum node spacing of 1km. The radio also used an RS232 communication interface for access to the kOS module.

5.4. Sensor module

The initial design for the sensor module was based on the approach used by Intelisys in past projects—that is, using PIC microcontrollers for control; a large flash memory for data storage and custom enclosures. The sensor module was also designed to use an RS232 communications interface for access to the kOS module.

6. Development

One of the most fundamental design decisions to make was how to allocate functions to physical devices. In our case, we could have located the kOS board inside the submerged sensor module. However, a substantial risk of failure existed in the communications cable breaking between the floating buoy and the submerged module. By co-locating the kOS and radio boards, the kOS board could still perform some network routing and processing of data from other nodes. This design for the first trial is shown in Figure 2. It shows the main components in a SECOAS node (we call this a "*seno*" from SECOAS node). Figure 3 shows a photo of the prototype buoy. This buoy is tethered to a submerged sensor module which logs data locally.

The sensor module occasionally passes important data to the kOS which stores and forwards it to the base-station. Various applications use this data and communicate with instances of themselves across the network. User polices and application data are forwarded by the radio module to the kOS—information is passed to the sensor module if necessary, for example and data re-transmission requests may be forwarded by the user to a particular node.

In the sections below, we show some issues that perhaps were not identified as important (if at all) in the early stages of the project; but which became increasingly important as implementation issues became apparent.

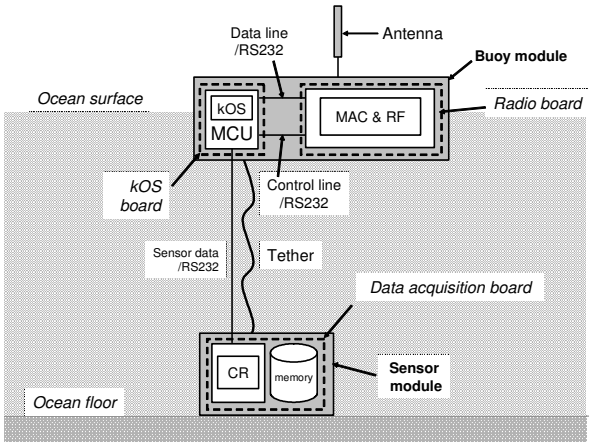

Figure 2. System configuration of SECOAS node (*Seno*)

Figure 3. Photo of prototype buoy with attached radio/kOS enclosure.

6.1. kOS module

As work on the various modules continued, it became clear that at UCL (and indeed, at other partner locations) we required our own laboratory-based facility for testing inter-node communications, especially for distributed application object messaging. More importantly, we required some kind of emulation of the radio and sensor modules. We then devised a facility which emulated most facets of the physical,

logical and application interfaces for both kOS-external modules. We configured this facility such that some boards could be "radio adjacent" to other boards—in short, an emulated network of any topology could be created. Received Signal Strength was also injected to each kOS board upon every transfer, primarily for the use of auto-location application. This, again, was a feature of the hardware radio boards.

Thus a facility was created whereby distributed applications could execute across a network as if on a real, fielded system. Similarly, we emulated the essential features of the interface to a sensor module.

With the emulation facility, testing at UCL could continue without the constant need for hardware from project partners. The development schedules for each partner was decoupled from UCL and the overall risks were reduced. This process also vindicated the design decision to use standard RS232 connectors rather than I2C, for example, in that simple, standard tools could be employed to test and interface the boards.

6.2. kOS task scheduling

Each application object, upon execution, usually decides on its next execution time—and execution times are relatively constant each instance. That is, they are designed with iterative execution in mind. Thus we can control the quality of the result an application produces simply by altering the period of its execution.

As we began to appreciate that processor CPU contention may affect the operation of kOS during development, we sought methods to manage the total processor load. By noting the simple relationship between application result and processing load by performing off-line measurements, we were able to implement a simple method for control over total processor load.

6.3. kOS robustness of operation

In SECOAS, seno robustness is very important as we are operating a remote, embedded system. By *robustness*, we mean several things: (1) that each node will operate as expected, and if not, is expected to reboot itself in an attempt to bypass any intermittent problems; (2) that applications will operate given unknown radio connectivity conditions; and (3) that applications will operate acceptably when load-controlled by the scheduler (see Section 6.2 above).

The in-built PIC Watch-Dog Timer (WDT) is the first simple step in ensuring that each node is robust as in point (1) above. This hardware feature simply resets the PIC microcontroller if the WDT is not reset within a preset time (several seconds), and is a standard method used for embedded applications.

However, after a WDT power-on reset, the problem that led to the reset may occur again. To give ourselves a significant amount of safety during field trials, we sought a method to limit or prevent the execution of particular application objects. We are currently investigating countering problems by keeping a record of the last object to execute—after a WDT power-on reset, the system examines this record and adds this object ID to a list of *prohibited objects*. The scheduler then ignores (or more closely monitors) calls to this object, bypassing this problem for that particular node.

7. Implementation issues

For the first trial, various implementation issues were discovered that changed the design. In this section we attempt to list some example items.

7.1. Power consumption

During the early stages of the project, the power efficiency of our senos was considered paramount, as the devices would have to remain *in situ* for possibly months at a time. At that time, the devices were conceived as small, possibly wallet-sized devices. As the development began, it became clear that at least for the first trial, demonstrating the essential features of the project should take precedence over device miniaturisation or an emphasis on efficiency.

The design decision was then to move towards large (60cm diameter) buoys with 18×18×10cm enclosures for the radio and kOS boards. Similarly, the sensor module was not required to be miniature. Consequently, relatively large batteries could be used to power the kOS board. This led to the design of the kOS board on an off-the-shelf prototype board from Microchip [11], with the plan to build custom boards later (with all the power efficiencies inherent in the future designs). This prototype board was suitable in that it contained all necessary features, and would allow UCL to initially focus on software development. This was important, as UCL had initially planned to only support software.

One of the tangible problems associated with an emphasis on power efficiency was found in the design of the radio module. In particular, a microcontroller was chosen for this board that would operate under adverse battery voltage conditions, extending its lifetime significantly. The downside to this was that the microcontroller had a limited number of logical ports, complicating the design of the logical communications interface between the radio and kOS boards.

7.2. Interfaces

Interfacing modules and applications proved to be one of the most difficult steps in design a functioning prototype system. The geographic distribution of project partners led to a difficulty in understanding, not of the physical interfaces, but rather of the logical nature of interfaces. Similarly, application-writers tended to produce good work in isolation; interfacing with other applications proved more challenging.

7.3. Communications

The radio and networking design was simplified for the first trial. In particular, a decision was made early to limit the size of packet payloads to 16 bytes for the 2004 summer trial. This restriction was made in order to keep the physical frame sizes small so that timing errors would be minimal—with the plan to increase frame sizes once the physical constraints were clearer. With messaging overhead, only 10 bytes could be transmitted at one time between nodes, although intra-node communication was not so limited.

A fundamental design decision, shown in Section 2, was to implement one base-station node and a number of floating sensor nodes. The idea was that the base-station node would communicate with its closest neighbours, and information from the rest of the network would gradually spread to these "edge nodes". It became clear, however, that especially for radio testing purposes we required a facility to directly query each sensor node from the shore. For this to be an active process we also required that each sensor node would respond to a reset command and take the appropriate action.

7.4. Logistics

For operations in the Scroby Sand area, we were subject to various regulatory conditions. Firstly, as we were transmitting radio power we had to apply for a suitable licence. Secondly, as we were deploying nodes in the ocean we had to apply to relevant government agencies for a *Consent to Undertake Marine Works* licence. We then had to issue a *Notice to Mariners*. Thirdly, we had to liaise with local interest groups (including fishermen) to ensure they were informed and consenting to our proposals. This elicited negative reactions from fishermen, complicated by previous disruptions caused by the construction of the Scroby wind farm.

8. Summary

Here we compare the end design solution in the first trial to the stated project objectives given in Section 2. A number of these items, by necessity, are to be addressed at a later stage—including (i) the design of the user interface design and (ii) evaluating sensing field methodologies.

A number of the items will be fulfilled by successful operation in the first trial—at the time of writing the first trial is several weeks away. An example is the demonstration of our technology in a realistic application context. We feel the remaining items have already been partially addressed:

- Designing and fielding decentralised algorithms—this has been demonstrated in an emulated network. We will support upgrades in the near future
- Designing novel cooperative adaptive data handling techniques—our gossiping/"firefly-flashing" techniques, for example, address this issue
- Implementing locally intelligent sensors capable of dynamic self-configuration—this has been implemented fully and is ready for initial testing
- Designing lightweight, low power, ad-hoc wireless communication protocols—this has been simplified for the first trial by necessity

9. Conclusions

Especially with geographically-distributed project partners, careful attention needs to be given to handling the design of physical and logical interfaces. Interfaces should be kept simple and standard, if possible. If this is done correctly, individual project partners may then concentrate on their particular functionality, in the knowledge that hardware and software modules will communicate as expected.

Standard engineering methodologies such as systems requirement analyses are also useful exercises in the early stages of a multi-partner project like SECOAS. If done adequately, this will provide a traceability of requirements from implementation decisions back to project goals (and steps in between). Conversely, this will highlight deficiencies in design reasoning. During this process, allocation of functions to hardware and software is performed. Integral to this whole process should be a weighting and prioritisation of user preferences for various system features. These steps will help ensure that a *useful* system is developed, which addresses real user requirements—a real danger in any research and development project is that self-satisfying interests prevail.

Above all, we have found that initial pragmatic solutions must be found in order to counter unexpected delays, organisational differences and kick-start the development before more sophisticated work can proceed. This is especially so in projects where there

are new operational practices combined with significant hardware and software complexity.

10. References

[1] I. Marshall, "Proposal for funding support for NWTM Project," Self-organising collegiate sensor networks (SECOS), DTI Proposal, July 2002.

[2] C. Vincent, "SECOAS: The oceanographic requirement - a concept note," SECOAS-internal note, 2003.

[3] M. Britton, L. Sacks and H. Hadaddi, "kOS—A Robust, Stateless Operating System Supporting a Self-Organising Wireless Sensor Network," *submitted to* SECON2004.

[4] L. Sacks, M. Britton, I. Wokoma, A. Marbini, T. Adebutu, I. Marshall, C. Roadknight, J. Tateson, D. Robinson and A. Gonzalez-Velazquez, "The development of a robust, autonomous sensor network platform for environmental monitoring," in *Sensors and their Applications XXII*, Limerick, Ireland, 2nd-4th September, 2003.

[5] A. Gonzalez-Velazquez, M. Britton, L. Sacks and I. Marshall, "Energy Savings in Wireless Ad-Hoc Sensor Networks as a Results of Network Synchronisation," in the *London Communications Symposium*, University College London, 8th-9th September, 2003.

[6] *PIC18FXX2 Data Sheet*, Microchip Technology Inc, Document DS39564B, 2002.

[7] T. Adebutu, L. Sacks and I. Marshall, "Simple Position Estimation for Wireless Sensor Networks," in the *London Communications Symposium*, University College London, 8th-9th September, 2003.

[8] C. Roadknight, "Sensor Networks of Intelligent Devices," 1st European Workshop on Wireless Sensor Networks (EWSN '04), Berlin, 2004.

[9] I. Wokoma, L. Sacks and I. Marshall, "Clustering in Sensor Networks using Quorum Sensing," in the *London Communications Symposium*, University College London, 8th-9th September, 2003.

[10] I. Wokoma, L. Sacks and I. Marshall, "Biologically Inspired Models for Sensor Network Design," in the *London Communications Symposium*, University College London, September, 2002.

[11] *PICDEM2 Plus User's Guide*, Microchip Technology Inc, Document DS51275A, 2000.

[12] M. Britton, "kOS v0.18.alpha Specification and User Guide," UCL internal report, August 2004.

[13] M. Castro, P. Druschel, Y. C. Hu and A. Rowstron, "Exploiting network proximity in distributed hash tables," presented at the FuDiCo 2002 International Workshop on Future Directions in Distributed Computing, Bologna, Italy, June 2002.

[14] L. Shum, I. Wokoma, T. Adebutu, A. Marbini, L. Sacks and M. Britton, "Distributed algorithm implementation and interaction in Wireless Sensor Networks," 2nd International Workshop on Sensor and Actor Network Protocols and Applications, Boston, Aug 2004 (to appear).

[15] SECOAS project web site. `www.adastral.ucl.ac.uk/sensornets/secoas`