A SELF – SYNCHRONISED SCHEME FOR AUTOMATED COMMUNICATION IN WIRELESS SENSOR NETWORKS

#Antonio Gonzalez-Velazquez[1], Ian W Marshall[2], Lionel Sacks[1]

[1]Department of Electronic and electric engineering, University College London, United Kingdom, {ae.gonzalez, lsacks }@ee.ucl.ac.uk ; [2]Department of Computer science, University of Kent, United Kingdom, i.w.marshall@kent.ac.uk,

Abstract

An algorithm for self-scheduling of node access and self-configuration of routes to data collectors in wireless sensor networks is proposed and described. The algorithm relies on the robustness and stability of the self-synchronisation of unnamed pulse coupled oscillators. Results of an initial simulation of a protocol based on the algorithm are reported. The results indicate that the protocol is resilient in the presence of low levels of mobility and noise. Plans to perform more realistic future tests including a full implementation are outlined.

I INTRODUCTION AND MOTIVATION

A typical wireless sensor network interconnects a collection of sensor nodes for the purpose of collection of data. Sensor nodes placed in unconstrained environments are vulnerable to displacement from their initial intended location. Nodes have limited computational resources in order to minimise cost, and have limited power availability in order to minimise manual intervention (changing batteries) or expensive mains feeds. This means that any communication algorithms must be resource efficient, particularly with respect to energy, state storage and processor cycles. Because of their location and additional requirements, nodes playing the role of gateways may have more resources than the rest. Since wireless devices attached to sensor nodes have a limited transmission range, multi-hop communication is needed to flexibly cover vast areas without recourse to a fixed communication infrastructure. In addition, and possibly most importantly, sensor networks must operate autonomously with no demand for high levels of operator skill in processes such as configuration and maintenance. A network of the kind described will have intermittent long latency channels to data collectors. The high probability of external interference in low power radio networks makes lightweight self-recovery from intermittent link failure crucial to achieve robustness. On the other hand the typical required bit rate is low compared to other wireless applications reducing pressure on bandwidth demands.

Four key aspects are particularly considered in our proposal. Firstly, overheads need to be minimised. These could be caused by explicit setting-up, maintaining and cancelling of communication paths to data collectors, acknowledging transmissions, and confirming message exchange. Secondly, because of the complexity of handling a large and dynamic number of active devices, communication protocols should avoid relying on addressable-endpoints. Thirdly, achieving a high probability of data arrival is considered vital for a sensor network, which is designed to collect measurements. Lastly, the network must dynamically self-reconfigure functional routes as needed.

In order to enable data collection with the characteristics described above, this proposal aims to provide self-scheduling of access for the wireless sensor nodes and self-configuration of routes to the data collectors using TDMA medium access. We are proposing a self – synchronisation algorithm simple enough to be implemented in a small MCU with properties of durability and resilience and a high sleep/wake ratio to conserve energy. The proposed algorithm is derived from previous work on self-synchronisation of discrete coupled oscillators; which indicates that given a certain set of conditions, a group of oscillators will always self-synchronise even in the presence of random perturbation [1]. Using a scheduled fashion of wireless access brings predictability and cooperation to sensor nodes. To demonstrate that our proposed algorithm possesses similar properties, a simulation is created and its results discussed.

This document is organised as follows. Section II introduces background research for the proposed method and similar proposals in the field. Section III describes in detail the characteristics of the model, including a discussion of its properties. The current implementation of this model for a wireless sensor network is described in Section IV. Section V presents initial results obtained by experimental simulation. Finally, Conclusions and future steps are mentioned in Section VI.

II RELATED RESEARCH

Properties of self-synchronisation of pulse-coupled oscillators have been extensively studied by Strogatz in [1] and [2]. Applications in different fields include the use of self-synchronisation of oscillators for performing coherent perception using discrete dynamics elements as reported by Schultz and Wechsler [3]. Li and Rus [4] use theory and methods applicable to pulse coupled oscillators for proposing a service of global time synchronisation with high levels of accuracy and robustness in ad hoc networks. Nevertheless, they focus on providing an accurate time service, and do not address other networking issues. Rhouma and Frigui [5] discuss the process of dynamic cluster formation using a population of pulse-coupled oscillators firing at a predetermined phase difference. We believe the work reported in this paper is the first application of these techniques to wireless sensor networks.

Research elsewhere indicates that reduced-state protocols can achieve interestingly high levels of resilience and performance. In [6], Vahdat and Becker discuss specific properties of a mechanism for simple propagation of routing information in ad hoc networks. They introduce valuable insights into how reduced state proposals could perform; however, their study is by no means exhaustive. The use of scheduled schemes in favour of on-demand schemes for sensor networks has recently attracted specific attention. For example in [7] El-Hoiyidi discusses the use of a non-persistent spatial TDMA technique with advantages in energy consumption for transport regular and frequent traffic. Trial experiences at the University of Western Australia indicate that CSMA mechanisms are highly prone to hidden terminal problems when transmitting sequences of messages along a path. Furthermore, the use of RTS/CTS usually proposed to address these CSMA shortcomings does not necessarily work for the high noise, asymmetric channels, and time dependent noise found in environmental sensor networks [8]. In addition the current authors have presented [9] an initial discussion of the relevance of further study on collegiate sensor networks; and mentioned the importance of establishing strong experimental experience.

III PROPOSED ALGORITHM

We assume that the sensing nodes are identical oscillators that have the ability to synchronise using externally audible radio signals emitted at a defined interval. Data collectors play the role of reference points for synchronisation by generating signals at the defined interval at all times, whereas other nodes are quiet until they hear a signal. Hop depth is defined as the number of links between a node and its associated data collector using the shortest route possible. The nodes execute the local algorithm described below and as a result the network converges to a state where groups of oscillators belonging to the same hop depth become phase synchronised with respect to groups belonging to the adjacent upper and lower hop depths. When the network has converged, the nodes at a given hop depth always listen when those one step further from the data collector are broadcasting, and always send their own packets after the further nodes have finished. When nodes finish broadcasting they then listen to the nodes one hop nearer to the data collector, to maintain synchronisation.

In detail, the oscillators are characterised by an integer variable, n, that monotonically increases until a threshold is reached. When this happens the oscillator emits its signal. The oscillator then returns the variable to its initial level and starts a new cycle. From the number of "steps" available in the oscillator only three are dedicated to network interaction with assigned activities to be performed on them. Oscillators reaching the *firing-position* (position *n*, last in each cycle) will emit a signal. When in the *collection-position* (position *n – 1*) and *checking-position* (position *0,* immediate next after *firing-position*) will look for other's signals. In the *collection-position,* the device will observe signals originating from one hop depth higher than its own. In the *checking-position,* it will observe signals originating from one hop closer to the data collector.

A discrete automaton of two states (*induced* and *not-induced*) controls the dynamics of the oscillators. Definitions are summarised in Table 1 and the pseudo-code for the general algorithm is given in Table 2a and 2b. Oscillators compare their current phase with the presence or absence of other's signals during those two states. Depending on the result they can transit from one state to the other. During the default state, *not-induced*, oscillators dedicate cycles to detecting other's signals and abstain from firing. If satisfied with a persistent signal (*inducing signal*) they will then phase their time-state to the appropriate value and transit to the *induced* state. During the induced state, the oscillators will fully perform their regular routine until they transit back as a result of failure to observe signals from elsewhere. The window of observation (forcing older observations to decay) represents the period that the oscillator is expecting to identify a consistent inducing signal and is represented by the variables failure-threshold and inducement-threshold in the pseudo-code. In our simulations failure threshold is set to be 3 (to avoid oscillation associated with transient failures), inducement threshold is set to a value of 1, and n (the number of steps) is set to be 10 to allow a maximum number of nodes in excess of 50.

```
While state = induced-state, starting at
            (frame-counter = 1 ,  slot-counter = 1 missed-signals = 0  )
 repeat:
  { (1) case of frame-counter
    {  (1.a) = firing frame
             { if slot-counter = my-slot   { transmit  packet }
               else                        {  do nothing       }
             }
       (1.b) = receiving frame
             { if packet present-at-receiver { receive-packet (slot-counter) }
             }
       (1.c) = checking frame
             { if slot-counter = 1       { presence-of-inducing-signal = false}
               if packet present at receiver { presence-of-inducing-signal = true  }
               else                          { do nothing  }
               if slot-counter = k
                 {if presence of inducing signal  = false
                   { missed-signals = missed signals + 1
                     if missed-signals > failure threshold
                           { change network state to not-induced }
                     else
                           { missed-signals = missed-signals – RAND(1)
                             //   forget singleton misses
                           }
                   }
         (1.d) else                        { do nothing  }
       }   //end of case of frame-counter = 1
  (2) slot-counter = slot-counter + 1
  (3) if slot-counter > k
             { slot-counter = 1
               frame-counter = frame-counter + 1
               if frame-counter > n  { frame-counter = 1  }
             }
 } //end of while
```

Table 2a. Pseudo-code for the induced state

```
While state = not-induced-state, starting at
             ( slot-counter = random (1..k), frame-counter= random (1..n),
               signals-received   = 0 )
 repeat:
{ (1) repeat for (n * k ) time slots
      { if packet present-at-receiver { signals-received= signals-received + 1  }
           // record a positive perception of a packet
      }
   (2) If signals-received > 0 // signal detected
       { if  signals-received >= inducement-threshold
           {  // choose a frame to synchronise to
              If activity is detected at only one frame
                     {synchronise to that frame (inducing signal) }
              else
                 { synchronise to earliest signal detected (inducing signal) }
              state = induced-state
              frame-counter = 1
              slot-counter = 1
              // internal sequence synchronises with the inducing signal
              // for the next cycle.
           }
         else
          {   signals-received = 0     }
       }
      else                  //   signal not detected
        {  do nothing  }
} // end while
```

Table 2b. Pseudo-code for the not-induced state

```
k = time slots per frame ;  n = frames per cycle
frame-counter = internal sequence of frames from [1..n]
slot counter = internal sequence of frames from [1..k]
my-slot = my self-assigned slot in the appropriate frame
signals-received = number of packets received this cycle
firing-position (firing frame)          => frame at frame-counter = n
receiving-position (receiving frame)    => frame at frame-counter =  n -1
checking-position (checking frame)      => frame at frame-counter =  1
```

Table 1. Definitions for Table 2a and 2b

The oscillators remain unaware of activity in positions different from *{collection-position, firing-position, checking-position}*; during which they merely keep track of the elapsed time. Due to the dynamics of the oscillators, the signals perceived in *checking-position* correspond to those of the *inducing signal*. The main reason of observing signals at this position is to maintain synchrony; the oscillator should remain in the induced state as long as signals are observed. A negative assessment triggers a transition to the not-induced state. This could be the result of a local error or the result of a link failure, the node does not distinguish the cause, but assumes its local clock is working correctly. If the local clock is faulty the node will fail to remain induced and after several attempts will fail into a permanent non-induced state (quiet).

IV EXTENSION OF THE ALGORITHM FOR AD HOC WIRELESS SENSOR NETWORKS

In the context of our intended application, effective communications must provide low loss of data, transfer of data to correct destination and resilience to realistic external disruptive events. In order to demonstrate that this is a reasonable objective, a stateless protocol was designed with properties of resilience, durability and operational efficiency. Because of the strong emphasis in operational stability and low cost of ownership, high utilisation and short latency are not considered.

A Assumptions

The dynamics of the algorithm presented above establishes the scheduling order and routing. Packets are transmitted in local broadcast mode. No acknowledgement of packets and no individual addressing is implemented. Data collectors are permanently in induced state, they hold improved resources than the rest of the nodes. They generate “beacon” packets needed for the phase synchronisation process. “Beacon” packets are regular packets intending to carry downstream data in future versions of the protocol. Nodes remain in *not-induced* state until a transition is triggered by the persistence of packets. The network converges to a state where all nodes within transmission range of at least another node reach *induced* state. Presence of noise, mobility and node failure represent disruptive factors affecting functional attributes of the network. Data requirements are considered low and with little demand for latency.

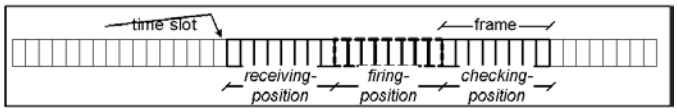

Figure 3. Group of time slots are grouped into frames. A fixed umber of frames makes a cycle. Nodes reference their respective frames accordingly to the operational cycle

B Medium access and scheduling

STDMA (Spatial Time Division Multiple Access) seems a natural choice of method for allocating access to the medium using the self-synchronisation approach. Successive sets of 8 adjoining time slots (named *frames*), where each frame corresponds to the time taken for a single step in the synchronisation algorithm, are dedicated to successive levels of hop depth. Nodes operating in the induced state schedule packet transmission to concur with the *firing-position* (last frame in the cycle); receiving packets for forwarding at the *collection-position* (penultimate frame in the cycle, just before *firing-position*); and assessing the source of the inducing signal at *checking-position* (first frame immediately after the firing-position). Nodes reference their respective frames accordingly to the operational cycle with no absolute reference.

Transmissions from nodes inside the same hop depth use time slots inside the same frame; and consequently. Nodes operating in the induced state keep wireless communications active for three frames *{collection-position, firing-position, checking-position}*; in each operating cycle. Figure 3 illustrate ths main three frames in a cycle. Nodes put the wireless interface in sleep mode for the rest of the cycle. Consequently, the operational duty cycle of the wireless interface when in induced mode is less than 3/(cycle length). Since *checking-position* is used to detect presence of transmissions, the average duty cycle can be reduced even more. Radio synchronisation and time slot dynamic allocation are implemented using proprietary algorithms that for space reasons will be discussed elsewhere. Packet redundancy is achieved by allowing more than one receiver to listen to incoming transmissions.

Packets consist of 6 bytes of a known preamble bit sequence and 4 bytes for management fields, 16 bytes of payload and 4 bytes for CRC. Having an air rate of 10 Kbps, a time slots is 50ms long, providing space for time slot alignment and synchronisation using simple radio transceivers. Frames group eight time slots for a length of 400ms. There is no explicit frame marker. The management field contains a three bit long sub-field indicating the time slot the packet is in; the rest of the management field is being reserved for debugging and future use. Receivers compute the start and end of the frames based on this. The data collector uses the time slot numbered as 0; its beacon transmission is used as a reference for the network. This reference is propagated across the network. The data collector's packets are currently used for debugging purposes. Each node locally computes an operational cycle, equivalent to 10 frames (4 seconds).

C Routing and forwarding

Packets contain no destination addressing. The payloads are assumed to carry time and location information derived from local measurements. Packets from time slots in the *collection-position* are placed in the outgoing buffer with no explicit directions. Outgoing packets are transmitted in a particular time slot inside the frame *firing-position*. Outgoing packets are released from the buffer if the presence of at least one transmitter is detected in checking-position. Our proposal uses a "store-and-forward" approach. Received packets are stored for later examination and forwarding. Nodes with no packets to transmit remain silent.

The scheduled scheme facilitates peer-to-peer interaction in addition to forwarding traffic. For activities such as location determination and cluster formation sensors will benefit from a peer-to peer interaction capability that avoids the need to communicate via the central data collector. Since the packets are broadcast at the radio level and nodes are listening, and reading packet headers, during three frames neighbouring nodes can gossip using the management field, regardless of hop depth; however, because of space reasons, this will only be discussed in detail in further publications.

V EXPERIMENTAL SIMULATION

We intend to show that the algorithm produces stable operation and efficiently schedules routing, transmission and reception of packets. The primary target is to show that an address-less, acknowledgement-less and state-less protocol can operate autonomously and stably in the context of a sensor network. A secondary target is to show that the achieved routes were reasonably efficient. Finally, and crucially, we need to show that this simple method can satisfy the purpose of efficiently collecting data from sensors.

Based in the context of project SECOAS [10], a network containing 48 nodes sampling oceanographic variables, and one data collector on shore was simulated for the equivalent of 11 hours in a single run using the agent based simulator Netlogo [11] similar to the illustration in Figure 10. The simulation area is 10.76 Km by 7.23 Km. Nodes were initially randomly placed and subsequently moved using Brownian trajectories (random direction and random displacement computed each 40ms; the maximum displacement is equivalent to 2.5 m/sec).

Nodes transmit at an air rate of 10Kbps and have a range of 1.5Km of approximately circular shape (based on preliminary measurements in the field). Noise was simulated by randomly blocking 2% of packet reception at every receiving node. *Inducement threshold* was set to 1 and *failure-threshold* was set to 3. Network density will clearly vary with time in different sections of the working area, but is normally sufficient that each node can see at least 2 others (Strogatz condition for convergence). Each device is provided with identical code and randomised initial values (between zero and the length of the cycle) for their internal time-states. The data collector remains permanently in the induced state.

During the simulation, each node generates 15.8 KB of sample data using a random sampling approach. This data is transmitted in 990 16 byte payloads. This corresponds to sampling the environment 90 times/hour; most real applications will require less than this, for example, a SECOAS node will generate around 25 16 byte packets each hour. Wireless devices can hold up to five packets in a FIFO fashion. No overflow management is performed; consequently, some received packets might be dropped with no additional notice or request.

In order to make an assessment of the resilience of the algorithms to unexpected events, when the elapsed time reaches 1 hour and 48 minutes (randomly chosen), 9 nodes from the population are randomly selected and their synchronisation settings reset.

A Evaluation method

The scheduling scheme was evaluated using several metrics;

i) by counting the number of packets, μ_p, that were not detected by neighbouring receivers. This parameter provides an indication of the link availability in the network.

ii) routing was evaluated by analysing the difference, σ_h, between the ideal hop depth of the source node of each packet and the ideal hop depth of the receiving node. A difference of one represents a successful engagement; a higher difference indicates additional hops than the minimal.

iii) by counting the number of nodes in the induced state, η_i. This is a good indicator of stability of the algorithm since transiting to the not-induced state can be seen as a failure.

iv) arrival rate of measured data (compared to measurements made by sensors)

v) arrival rate of packets (shows effect of broadcast redundancy)

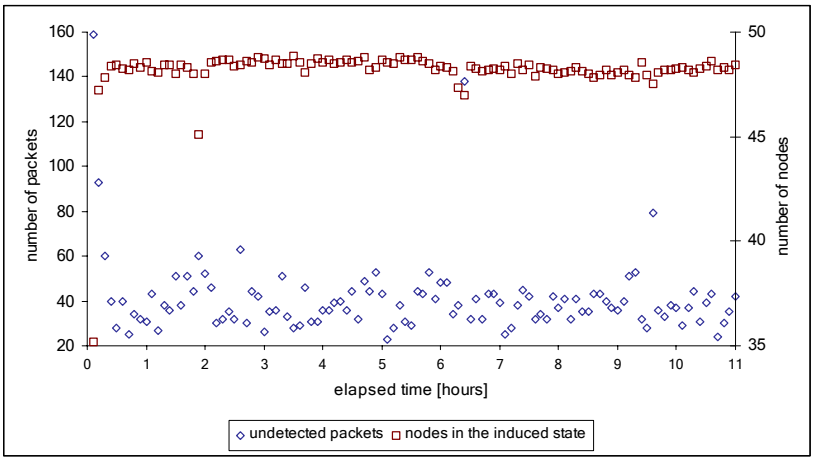

Figure 5. Results of a single run simulation in conditions of noise and low mobility. The total number of undetected packets and the average number of nodes in the induced state are plotted every ten minutes. Nine nodes were randomly selected at one hour and 48 minutes and reset their synchronisation

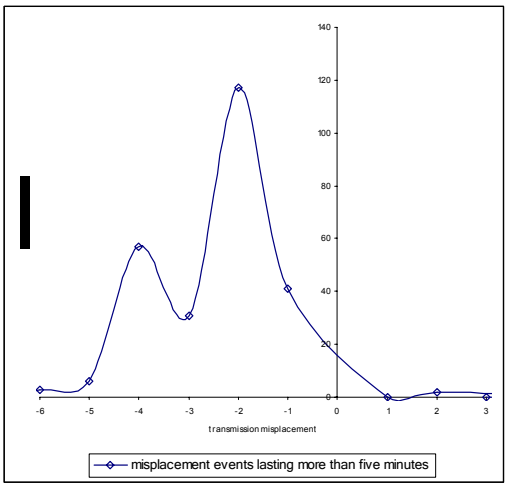

Figure 6 Results of a single run simulation. The total number of misplacements is plotted against the amount of each displacement

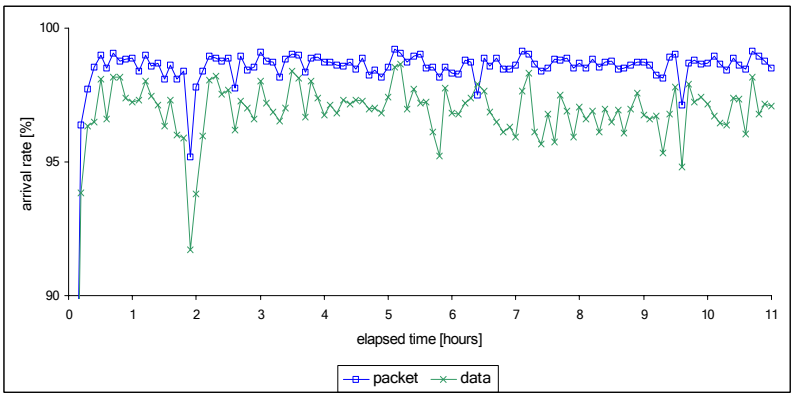

Figure 7. Results of a single run simulations of a sensor network with characteristics described in the text. Packet arrival and data arrival are plotted at the end of every period of 10 minutes. Nine nodes were randomly selected at one hour and 48 minutes and reset their synchronisation

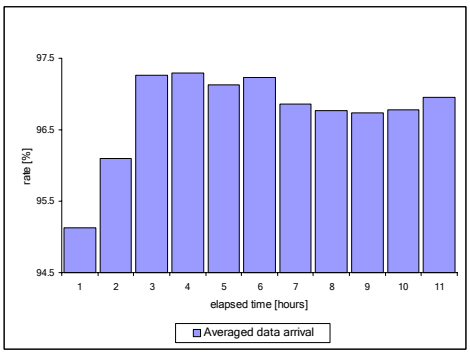

Figure 8 Results of a single run simulation. The averaged data arrival rate is plotted for each hour of the duration of the experiment.

B Initial results

Results for μ_p and η_l against simulated time are plotted in Figure 5. The left-hand Y axis of fig. 5 plots μ_p and the right-hand Y axis plots η_l. Figure 6 shows σ_h as a function of number of misplacements lasting more than 5 minutes. These graphs confirm that the oscillators can self-synchronise and form effective routing hierarchies, even in the presence of major disturbances. It must now be shown that the network is able to deliver the measured data with high efficiency. To assess this, data and packet arrival rates are plotted in Figure 7. Figure 8 shows a histogram of the averaged arrival rate for each hour of simulated time. It can be seen that the redundancy implicit in link layer radio broadcasts is sufficient to achieve high data delivery rates even during the major disturbance.

C Discussion

The network of identical unnamed oscillators converges to a stable operational state that is close to optimal (in terms of hop-count) within 1 hr. As expected the coupled oscillators naturally develop the ability to retain a successful configuration and consequently, show a natural resistance to drop the current synchronisation state in the presence of random disturbances.

From the results, it can be seen that missed pulses μ_p are present in small numbers and scattered across the experiment; in particular during the process of setting up the network. Their effect on the induced state nodes is limited due to *failure–threshold* being >1 and also to the natural redundancy of the links. It also can be seen that σ_h reflects stable and efficient routing.

In all of the plots the impact of resetting 9 nodes shortly after one hour and 48 minutes of running time can be seen. Importantly, this disruption does not seem to impact long term data recovery, so we can conclude that the five packet buffer was sufficient for the simulated network.

VI Conclusions and Future Steps

The presented results indicate that the proposed method can be used for resilient networking in wireless sensor networks. Initial convergence from random state occurs with no apparent difficulty. It can be seen that despite the presence of noise and limited mobility disturbing the connectivity of the network, the algorithms allow the network to recover gracefully from large perturbations. In addition, the algorithms are simple enough to be implemented in resource-limited devices. The device population seems to successfully self-organise the scheduled order of transmissions and reception and high data delivery efficiency is achieved. It is now considered necessary to analyse the proposal under more realistic propagation and disturbance scenarios than those used for the preparation of this document. It is also considered relevant to research further the impact of different settings and thresholds on longer operation periods.

During the next months, the proposed protocol will be fully implemented and tested as part of the project SECOAS in the Scroby sands area in the coast of Norfolk [7], as illustrated in Figure 9. The current hardware prototype (Figure 10), provided by our colleagues at the University of Essex, uses a PCB with one Microchip's microcontroller PIC16F876 run by a 8MHz crystal. The PIC16F876 is a CMOS FLASH-based 8-bit microcontroller into 28-pin package; holds 256 bytes of EEPROM data memory, 368 bytes RAM, and 14KB Flash memory. Each board has receiver and transmitter modules by Radiometrix (173.250MHZ-10 and TX1-173.250MHz-10 - VHF Narrow Band FM Transmitter and Receiver) with a dipole antenna. This hardware will be more than adequate to run our current implementation code, which requires 201 bytes of RAM (for radio handling, TDMA and network; and 6 x 16 bytes buffer) and 1200 bytes of FLASH memory (for object code and constants). At present our prototype implementation is undergoing integration tests with other components of the SECOAS system.

Acknowledgments

The authors acknowledge Ian Henning, Steve Fitz, and Taimur Khan from Department of Electronic systems engineering, University of Essex for their significant contributions in the ongoing design of the radio operation, in the synchronisation discussion and by providing the initial radio prototypes under the project SECOAS. We acknowledge Rachel Cardell-Oliver, from the Department of Computer Science & Software Engineering of the University of Western Australia, for her insights on CSMA mechanisms in sensor networks and for reviewing earlier versions of this document.

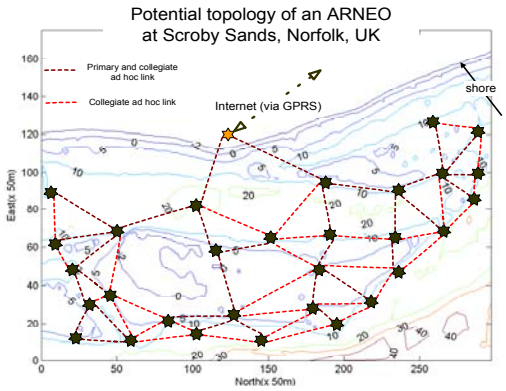

Figure 9 Example of a potential topology dynamically built for oceanographic monitoring. Annotations indicate depth in meters. Figure based on drawing provided by Dr. Chris Vincent, University of East Anglia, UK, 2004.

Figure 10. Picture of the PCB hosting the initial platform for developing the radio and network platform for SECOAS. (Picture courtesy of Steve Fitz)

References

[1] R.E Mirollo and S.H Strogatz,. 1990. Synchronization of pulse-coupled biological oscillators. SIAM J. Appl. Math. 50: 1645-1662.

[2] S. H Strogatz and I. Stewart. 1993. Coupled oscillators and biological synchronization. Scientific American 269 (6):102-09

[3] A. Schultz, H. Weschsler, A discrete Dynamics for Synchronization of pulse coupled oscillators, IEEE Transactions on neural networks, Vol. 9, No. 1, January 1998.

[4] Q. Li, D. Rus, Global clock synchronization in sensor networks, Proceedings of IEEE-Infocom 2004, ISBN: 0-7803-8356-7/04

[5] M.B.H. Rhouma, H. Frigui, Self-organisation of pulse-coupled oscillators with application to clustering, IEE Transactions on pattern analysis and machine intelligence, Vol. 23, No. 2, February 2001

[6] CA. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. 2000. Technical Report CS-200006, Duke University, April 2000.
[7] A. El-Hoiydi, Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks, Computers and Communications, 2002. Proceedings. ISCC 2002. Seventh International Symposium on, Vol., Iss., 2002, Pages: 685- 692
[8] R. Cardell-Oliver; private communication, August, 2004
[9] 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, Sensors and their Applications XXII, Limerick, Ireland, 2nd-4th September, 2003.
[10] SECOAS (Self-Organising Collegiate Sensor Networks) Project; Source: http://www.adastral.ucl.ac.uk/sensornets/secoas/
[11] U. Wilensky, (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston,IL