A DEPLOYED SENSOR NETWORK TO MONITOR A FLOODPLAIN

R. Roddis and E.L. Kuan
Multiple Access Communications Limited
Delta House, Chilworth Science Park
Southampton SO16 7NS
United Kingdom
Email: robert.roddis@macltd.com
Fax: +44 (0)23 8076 0602
http://www.macltd.com

1 Introduction

One project employing sensor networks is the FloodNet project [1], which involves a number of organisations and is partially funded by the Department of Trade and Industry (DTI) Next Wave Technologies and Markets programme [2]. The organisations involved in this project include Multiple Access Communications Limited (MAC Ltd), IBM, Halcrow, Associated British Ports and the University of Southampton. In the following sections, we will discuss some of the issues related to the deployment of this sensor network, its purpose and some of the algorithms that have been developed as a result of this project.

2 Sensor network

The FloodNet project involves the deployment of a wireless sensor network on a floodplain in Brandy Hole on the River Crouch in the UK. The purpose of this network is to monitor the flooding situation in the floodplain and to collect data from the flood events that occur naturally in the area.

This sensor network consists of 12 nodes deployed over an area with the dimensions of approximately 2 km by 1.5 km. Figure 1 shows an aerial photograph of the floodplain in Brandy Hole and the locations of the 12 nodes are indicated by arrows.

Each node is assembled from commercial off-the-shelf (COTS) components including a water depth sensor, a single-board computer, an IEEE 802.11b [3] wireless network interface card, an omni-directional antenna, a solar panel and a rechargeable battery. All of the nodes are identical in configuration except for the gateway node. The gateway node contains the same components as an ordinary node but also has a General Packet Radio Service (GPRS) modem that allows it to communicate with a mobile communications network. This provides a portal for communications between the sensor network and external networks.

The water depth sensor is a piezoelectric water level pressure sensor manufactured by GE Druck. It generates a small-signal scale voltage in proportion to the water depth at its

particular location and a sensor reading module converts this voltage to a depth value that is stored by the processing unit. The processing unit implements various functions including simple statistical analysis of the sensor data, and provides the networking and application layers of the ad hoc network. The sensor data collected by the processor are stored until they can be delivered to the gateway node at appropriate intervals. The gateway node forwards the data gathered from all the nodes in the field to an external network for further processing by a hydrological model and they are stored for future data mining.

Figure 1: An aerial photograph of the floodplain in Brandy Hole showing the locations of the sensor nodes and the viable communications links between them.

As these nodes are located in a remote and largely inaccessible area, the nodes are powered by solar energy using the solar panel and the rechargeable battery within the node. The one exception is the gateway node that is mains powered. This node is considered the most critical node within the network and it is important that it is not subjected to the vagaries of solar energy and is able to remain switched on all of the time.

Sensor nodes communicate with each other by wireless means, using the IEEE 802.11b wireless technology. The deployed network has typical inter-nodal distances of 400m, and the furthest reliable communications range we have achieved in the field is about 600m, therefore nodes at the edges of the network must use nodes between themselves and the gateway as data relays when communicating with the gateway. To fulfil this requirement, the nodes form an ad hoc network, where each node aims to form as many communications links to other nodes in the network, where possible. The main restrictions on a viable communications link between any two nodes is the transmit power of each node, the minimum received signal strength and interference within the area. The nodes are configured so that they are permitted to communicate with each other, but only the gateway node may connect to networks outside of the sensor network, and it will not forward data packets from an external network to the sensor network.

The IEEE 802.11b communications protocol provides communications between two neighbouring nodes; therefore, the sensor network requires a packet routing mechanism that enables nodes to distribute the sensor data amongst themselves and to deliver the data to the gateway node. The dynamic source routing (DSR) protocol [4] is a well-known routing protocol and has been implemented to provide the packet routing mechanism of the network. The DSR algorithm is based on a working document that has been submitted to the Internet Engineering Task Force (IETF) Mobile Ad Hoc Network (MANET) working group. The DSR algorithm is a routing protocol for mobile ad hoc networks that enables nodes to discover routes to other nodes within the network and to recover from broken links or node failures. The basic DSR protocol consists of two main mechanisms, route discovery and route maintenance. The route discovery process enables each node to discover routes for delivering packets to destination nodes within the network, whilst the route maintenance process enables each node to update the status of known routes and to eliminate broken routes and initiate the discovery process when required.

Although our nodes are static within the sensor network, the topology of the network changes over time as nodes turn on and off according to their sensor measurement reporting requirements and remaining battery charge. This changing network topology allied to the self-configuring and self-healing properties of the DSR algorithm implies that the nodes do not need to be pre-configured with routes or routing tables as these are learnt through the route discovery process. The algorithm also enables the nodes to cope with situations of node failure or excessive interference, allowing a node to find alternative routes, when needed. In this environment where the nodes are static, following the initial route-finding, the routing overhead should be minimal.

3 Node architecture

The block schematic diagram of a node is presented in Figure 2. The processor board hosts the majority of the components and the interfaces to the sensors, power supply system and communications components. The processor board and 802.11b wireless networking adaptor form the core of the node and is implemented as an Arm Linux-based single board computer (SBC) with a PCMCIA wireless network card. The SBC is connected to a 256 MByte Compact Flash (CF) memory card that provides writable data storage for the node.

The SBC used in the node has 64MBytes of RAM for working memory and 32MBytes of FLASH RAM for persistent storage that contains the file system for the node. This SBC uses an Intel XScale PXA250 microprocessor running at 400 MHz. At the beginning of the project, the performance and capabilities of this device were comparable to that of a high-end personal digital assistant (PDA) available at the time. The SBC was configured to run the Linux kernel and use a typical embedded Linux environment.

Wireless networking was provided by the PCMCIA-based IEEE 802.11b network interface card that used the Intersil PRISM II chipset. This chipset was well supported under Linux and can operate in all three 802.11 modes, ie, as an access point, as an access point client, and in ad hoc mode. Although the data requirements of the FloodNet network were quite low, of the order of a few tens of kilobits per second, the considerably larger usable bandwidth of 11

Mbits/s offered by the IEEE 802.11b technology was particularly useful for supporting system diagnostics and remote administration. The FloodNet applications were written to run over an IPv4 network, therefore the choice of the lower network protocol layers did not affect the design and implementation of these applications. Considering these requirements, the 802.11b WiFi standard was chosen primarily because of its ubiquity and the existing driver support built into the Linux kernel.

The water depth sensor module used an RS232 serial port interface that implemented a simple command/response sequence to perform depth readings on demand. This was later integrated into a power monitoring microcontroller that is discussed in the next section.

In the process of deploying the sensor network, various issues were encountered. In the following sections, we will describe some of these issues and how they were resolved.

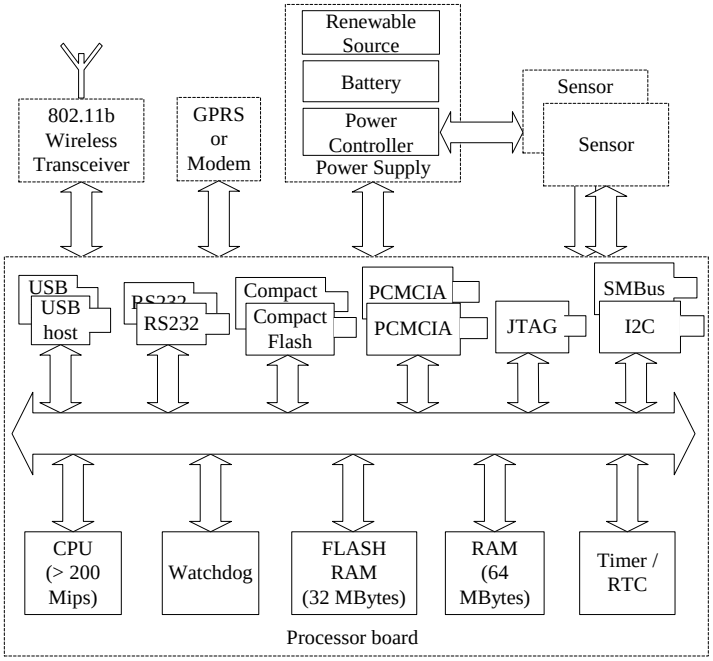


Figure 2: The schematic of a sensor node, where the grouping of components into units, indicated by the dashed outlines, denotes a sub-assembly.

4 Power control

One of the main issues with a node was that of power consumption. It was recognised from the start of the project that the power consumption of an active SBC would be considerably higher than the average available solar power, so it was necessary for the node to conserve battery power by running in a low power mode for the majority of the time. The SBC was required to support a low-power sleep mode, and be able to wake itself from sleep using a time source. Unfortunately, the sleep mode on the SBC used did not operate at a sufficiently low power, so a power control module had to be implemented to provide a programmable time-controlled power-switch to the SBC.

The power control module was implemented as a separate unit from the SBC and used a microcontroller as a smart electronic switch that controlled the power supply to the SBC. The requirement to completely shut down the SBC raised a number of problems. Firstly, the SBC did not contain a battery-backed real time clock (RTC). Therefore, the microcontroller had to implement one to keep time for the SBC while it was switched off. Secondly, the water depth sensor had to be sampled at a rate higher than the maximum rate that a node could be switched on and off without draining the battery too quickly. Thirdly, the microcontroller had to prevent the SBC from causing a deep discharge of the lead acid battery by monitoring the battery voltage and using this information to determine whether it was safe to turn on the SBC, and whether it was safe to leave it on.

The main function of the power control module was to switch the SBC on at regular intervals and switch the SBC off again until it was required. The wake and sleep cycles were dependent on the frequency of sensor data samples required by the flood monitoring analysts. Whilst the SBC was switched on, the recorded sensor data were collected into data packets that were then delivered to the gateway node, which were then sent to an external database for further processing.

Although the wake and sleep cycles were pre-determined, they were also dependent on the available charge in the battery. At times of bad weather, especially in the winter months, there was insufficient solar energy to charge the battery. Therefore, before switching on the SBC, the microcontroller monitored the battery voltage in order to ascertain whether the battery was sufficiently charged and capable of sustaining the activities required of the SBC whilst it was switched on. A hysteresis model was used to regulate the wake and sleep cycles of the SBC. The microcontroller continuously monitored the battery voltage. If the battery voltage dropped below a level of v_1, the SBC was switched off and would not be switched on again until the battery voltage exceeded the level, v_2, where $v_2 > v_1$.

As explained earlier, the target sampling rate of the water depth sensor was higher than the maximum wake-sleep rate of the SBC. Therefore, the SBC could not be depended on to obtain regular and frequent samples of the sensor data. The microcontroller was adapted to read the voltage output of the depth sensor, convert the voltage to a sensor value and to store the sensor data until the next wake time of the SBC for delivery to the gateway node.

5 Synchronisation

As the sensor nodes could not remain switched on all of the time, we had to ensure that the active periods of all nodes were synchronised with each other so that there were enough nodes switched on together to act as relay nodes for the delivery of data packets from the edge of the network to the gateway node. Therefore, a simple synchronisation protocol was implemented to ensure that the nodes would turn on at the same time of day. As the gateway node was mains powered it was always in the active mode. It also had a permanent connection to the Internet, and thus the network time protocol (NTP) was used to set its clock. This, in turn, was used as the reference time source to set the clocks of all other network nodes. Whenever a node connected to the gateway it would resynchronise its clock to the gateway, and then examine its sampling manifest to determine how long it should

remain switched on for. In the initial deployment, the entire network switched on at the same time for a period of five minutes every three hours to transmit data back to the gateway, except at 15:00 when the entire network switched on for 30 minutes to provide a maintenance window, and allow re-synchronisation for nodes that have lost contact with the rest of the network.

As the non-gateway nodes were switched on and off in time with one another, a node that could not communicate directly with the gateway would be unable to communicate with the gateway if, while it was turned on, sufficient intermediate nodes in the routes to the gateway remained switched off to prevent establishing any route to the gateway. This could happen, for example, because an intermediate node was forced to remain switched off until its battery was charged to a safe operating voltage. Therefore, it was necessary for the nodes to have a mechanism to detect that they were out of step with the network, and then resynchronise to the gateway node, even if the local clock within a node was reset. This was achieved as follows.

If a node failed to connect to the gateway five times in succession, it decided that it was no longer synchronised with the rest of the network. It would conclude that its clock was inaccurate and enter a search mode to resynchronise with the gateway. The node achieved resynchronisation by searching for the maintenance period, attempting to contact the gateway every 25 minutes, and remaining switched on for 3 minutes. Figure 3 shows the timing relationship between the normal node activity cycle (shown as a solid line) and the searching cycle (shown as a dashed line). Because the period of the search pattern, 28 minutes, is less than the duration of the maintenance window, 30 minutes, there would always be at least one occurrence of a node in the search pattern turning on at the same time as the maintenance period in the rest of the network. Once the node made contact with the gateway it reset its clock and determined that it had successfully resynchronised with the network. Using this simple algorithm it could take up to one day per hop from the gateway to achieve resynchronisation.

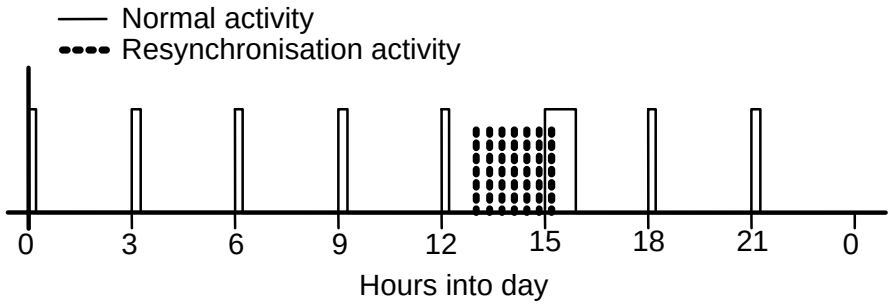
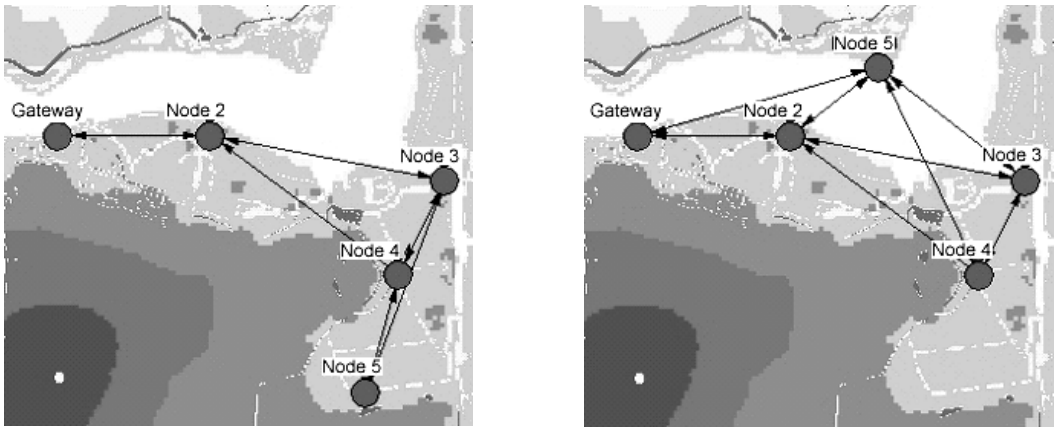


Figure 3: The timing relationship between the resynchronisation search pattern (dashed), in which the node is turned on for three minutes every 25 minutes, and the normal activity scheme (solid), in which the node turns on for five minutes every three hours, except at 15:00hrs when it remains on for 30 minutes.

6 Deployment

Choosing the network deployment sites was supported by a proprietary 802.11b WiFi simulator that we have developed to model the 802.11b physical layer. The simulator incorporates digital terrain maps of the deployment area and takes into account the impact of terrain on radio propagation. This allowed us to investigate the suitability of a particular network deployment from a communications standpoint before physically installing the nodes.

The results from this simulator, shown in Figure 4, encouraged us to change the initial deployment sites of the original five nodes in the first phase of deployment to improve the available network connectivity. In Figure 4 a, the simulator predicted a unidirectional link between Node 4 and Node 2, which would not work with the inherently bidirectional 802.11b network protocol. This effectively made the proposed network a single line of nodes, where if any one node failed, all nodes beyond that point would lose contact with the gateway. By re-positioning Node 5, as shown in Figure 4 b, we re-organised the layout of the network such that any one node could always communicate with at least two other nodes in case of failure.



a) Simulation of original node layout shows a unidirectional link between Node 4 and Node 2

b) Revised node layout where all nodes have links to at least two other nodes.

Figure 4: Results from the simulator showing possible communication links between the nodes.

7 Conclusion

In this paper, we have presented the issues related to the deployment of a practical wireless sensor network for collecting floodplain data on a tidal floodplain in Brandy Hole on the River Crouch in the UK. A number of issues that were encountered during deployment were discussed, including energy conservation and network synchronisation. The solutions that were implemented were presented.

The convenience of using an IP network of commercial components has been offset by the stringent demands of the power consumption of the nodes. As the SBCs did not support the initial requirement of a low power sleep mode, we have developed an approach with a microcontroller to control the power supply to the SBC that this has allowed us to collect sensor data at a higher rate than the activity cycle of the SBC would otherwise allow.

As our sensor network nodes must be turned on to communicate with one another, the activity cycles of the nodes must be synchronised to one another, and we have shown a simple mechanism that allows a node to determine whether it is synchronised to the network and how to regain synchronisation once it is lost.

Since the start of this project, better suited low-power SBCs that support embedded Linux are now available to us, and we can use these to reduce the period between network activity cycles, and therefore reduce the delay between taking a sensor reading and transmitting it to the data consumers via the gateway node.

In the second phase of deployment, we are using SBCs that support a functioning low power sleep mode that allows a node to turn on and off in little more than a second, rather than the minute it takes the original SBC to power-on. With a working sleep mode, nodes can also retain state information in memory while asleep, which allows us to pursue a data-driven sampling scheme and synchronisation model where nodes can communicate with one-another to determine how best to fulfil the sampling requirements of the data consumers while maximising the available battery power in a node.

Acknowledgements

The authors thank the DTI UK for funding the NextWave Technologies and Markets programme, their colleagues at Multiple Access Communications Ltd for their invaluable advice and support, as well as their fellow collaborators within the FloodNet consortium.

References

[1] http://envisense.org/floodnet.htm
[2] http://www.nextwave.org.uk
[3] ANSI/IEEE Std 802.11: Wireless LAN medium access control (MAC) and physical (PHY) layer specifications, 1999.
[4] D. B. Johnson, D. A. Maltz, Y.-C. Hu, “The dynamic source routing protocol for mobile ad hoc networks (DSR)”, Internet Draft, MANET Working Group, draft-ietf-manet-dsr-09.txt, April 2003, work in progress.