An Ad-Hoc Sensor Network for Vineyard Monitoring

Nicholas Nell, Nathalie Mitton, Thomas Niesler, Riaan Wolhuter

An Ad-Hoc Sensor Network for Vineyard Monitoring

Nicholas Nell*[1],Nathalie Mitton†[2],Thomas Niesler*[3],Riaan Wolhuter*[4]
*Department of Electronic Engineering, Stellenbosch University
†INRIA, Lille, France
[1]20165374@sun.ac.za [3]trn@sun.ac.za [4]wolhuter@sun.ac.za [2]nathalie.mitton@inria.fr

***Abstract*—This paper explores the use of an ad-hoc wireless sensor network in a vineyard-based environment to increase the effective range of sensor nodes without the use of high power communication techniques. Physical sensor and gateway nodes were developed in order to test the use of an adapted AODV routing protocol in a wireless sensor network. The network was deployed to a vineyard environment where the effectiveness of the strategy was evaluated based on the ability of the network to self-configure. First tests indicate that the protocol performs sufficiently well to be used in vineyard environments for data collection. We conclude that an ad-hoc sensor network is a feasible approach to low-power sensing over wide areas, as required for precision viticulture.**

***Index Terms*—Ad-hoc Networking, Wireless Sensor Networks, Short-Range Communication, Low-Power Operation, Vineyard Monitoring, Precision Agriculture**

I. Introduction

The collection and analysis of vineyard telemetry data can significantly contribute to viticultural efficiency and productivity. Such telemetry data can, for example, be used to increase the quality and amount of grapes produced in a vineyard, while at the same time minimising the risk of wasteful operations. This can contribute to increased efficiency by reducing the cost of production and increasing environmental sustainability [1]. Thus, an improvement in the effectiveness and ease of collecting such telemetry data will have a direct impact on the efficiency of the vineyard and allow methods of precision agriculture to be implemented.

Precision agriculture refers to the use of technology to make farming as efficient as possible by means of exact measurement. Generally, this is achieved by breaking farmland into smaller manageable sections, and then obtaining precise and continuous measurement data for each section. This allows for the determination of, for example, the variability in inter and intra-field condition of the plants and soil. If the exact conditions of every square meter of soil and every plant were known, the farmer could locally apply precisely the right quantity of water or fertiliser, thereby reducing waste and improving the quality of the produce. By using technology to virtually divide large areas of land into much smaller sections, the farmer is given far greater control over the crop. In the long term, this technology will also allow the collection and subsequent analysis of large sets of data, revealing trends that can be exploited for even more efficient farming and improved decision making [2].

Since these advances depend on the collection of telemetry data from the vineyards, a means must be developed to allow such data collection to be set up quickly, easily, at manageable cost and with reliable results. Currently, farmers must operate either without such data collection methods (especially in the case of small scale farms) or must use very labour-intensive approaches, such as on-site manual measurement and data collection. Where farmers have tried to deploy more advanced data collection methods, they have frequently been frustrated by the failure of traditional communication techniques. This could, for example, be due to a lack of cellular signal for GSM enabled nodes or power usage demands which cannot be met due to a lack of grid-tied power, especially when using high power communication techniques such as satellite or GSM. Other difficulties arise in vineyards which are situated in mountainous or hilly landscapes, where direct line-of-sight between sensor nodes in a network is difficult.

On-the-go data collection is employed in situations where wireless sensor networks are not easily implemented. A relatively inexpensive data collection technique, on-the-go data collection is based on a sensor attached to pre-existing mobile farming equipment, and where the data is collected while the equipment is in use. Examples include sensors attached to tractors or irrigation systems [3]. A limitation associated with on-the-go sensors is that the equipment it is attached to has to be in use for data to be collected and that the data usually has to be manually retrieved by the farmer.

Many of these limitations can be overcome by using ad-hoc wireless sensor networks that use low power, short-range communication techniques in conjunction with in-situ environmental, soil and plant sensors. Ad-hoc networks are especially useful because it is not necessary to configure each sensor node individually. Low-power designs allow the nodes to be battery-powered, and can thus be placed anywhere in a vineyard. The use of short-range communication requires sensor nodes to communicate with their nearest neighbours, and automatically discover routes to a gateway node without manual intervention. This approach offers a practically feasible means of setting up a data collection system that can provide autonomously gathered real-time data to the farmer.

This paper will explore the design and implementation of such a low-power ad-hoc vineyard based wireless sensor network. A description of the network setup will be followed by the hardware design and implementation of the sensors. This is followed by a description of the network protocol used.

Finally, the results of the first practical test will be presented.

II. Hardware Implementation

Our sensor nodes consist of an MSP432p401r microprocessor (MCU) attached to several sensors and peripherals [4]. The use of an ultra-low-power processor allows the nodes to be battery-powered for extended periods.

Peripherals include a low-power RFM95W LoRa radio module, operating in the $868MHz$ frequency band [5]. This radio module is responsible for all inter-node communication.

Every node also includes a low-power Adafruit PA101D GPS module [6]. This module supports multiple positioning systems, including GPS, GLONASS, GALILEO and QZSS. Besides providing each node with positional data, the most important function of the GPS module is to provide a stable, shared time base to all nodes. The accurate pulse-per-second signal, which commences when a positional lock is established, allows the MCU to synchronise its onboard real-time clock (RTC). This is critical to the operation of the network, as will be discussed in Section V.

To enable data transfer to cloud storage from the sensor network, the root node includes a GSM module based on the GL865-Quad [7]. The GSM module allows the gateway sensor node to periodically establish an HTTP connection to a web server and upload all received telemetry data.

Each node also includes a low-power BME280 temperature, pressure and humidity sensor. This integrated device simplifies the hardware design by avoiding the need for multiple separate sensors. For light intensity measurement, the MAX44009 light sensor is included, and to measure the volumetric water content of the soil, the EC-5 soil probe is used.

Lithium polymer (LiPo) batteries provide power to the nodes. This type of battery offers a high energy density and reliability. To regulate the voltage provided by the battery, a DC-DC regulator circuit was designed around the TPS63001 buck-boost converter [8].

All components were housed in an IP65 rated ABS enclosure to provide environmental protection and allow continuous operation in a vineyard, shown in Figure 1.

In the current prototype, the data collected by the network is uploaded to a web server running on a Raspberry Pi using the REST API.

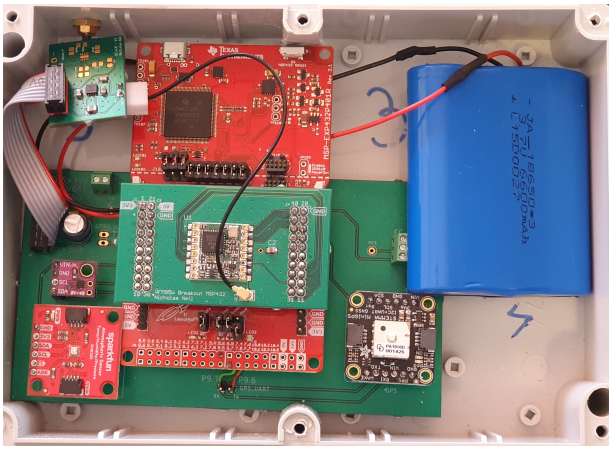

Fig. 1. Node hardware in enclosure.

III. Network Setup

Our sensor network consists of a mixture of edge nodes (nodes containing telemetry sensors) and gateway nodes (nodes that receive telemetry data from edge nodes). Edge nodes and gateway nodes have identical hardware, with the exception of the inclusion of the GSM module on the gateway nodes. The advantage of using the same underlying architecture for both node types is that it simplifies the maintenance of the hardware and makes it easy to increase the number of gateway nodes by simply adding a GSM module to an edge node.

Edge nodes primarily operate in a sleep state to conserve power. They are programmed to wake up periodically to collect telemetry data and to initiate a data transfer to a gateway node. Edge nodes also act as routing nodes when the need arises, for instance when there is no direct connection between an edge node and a gateway node. The network protocol will be discussed in greater detail in Section V.

In the current prototype, only a single gateway node is used. However, the network allows for the addition of multiple gateway nodes with no modification to the MAC or network layer protocols.

IV. MAC layer implementation

A. S-MAC

The MAC layer protocol used by our network is an adaptation of the S-MAC protocol. S-MAC is specifically designed to reduce the power consumption of a wireless sensor network. It has been shown to perform well in situations where nodes have single tasks, are deployed in an ad-hoc fashion and where nodes spend most of their time in an inactive or sleep state [9]. The implemented protocol uses the GPS pulse-per-second signal to synchronise all network node in time. The nodes can then use time slots to synchronise communication. The use of time slots reduces the chance of message collisions on a shared communication medium. However, the main reason for the use of a time-slotted approach is to reduce node energy consumption. Careful synchronisation allows nodes to

TABLE I
DATAGRAM FORMAT

Data	Header	Data	Network Statistics	Total
Size (Bytes)	10	46	17	73

TABLE II
DATAGRAM HEADER FORMAT

	Next Hop	Local Source	Network Source	Network Destination	Hop Count	Tx Slot	Current Slot	Message Type
# Bytes	1	1	1	1	1	2	2	1

bring the onboard radio module out of its sleep state only within a specified time slot.

The implemented MAC protocol borrows a few key techniques from the S-MAC protocol. Neighbouring nodes share their transmission time slots, eliminating the need for nodes to listen to the shared communication channel in every time slot. Thus nodes can remain in a sleep state while not in their own, or a neighbouring node's, data transmission time slot. Another borrowed feature is that nodes will place their radio modules in a sleep state when overhearing a data message for which they are not the intended recipient. This happens when a node listens to the shared communication channel, expecting a message, but overhears a message with a different destination. It can then conserve power by entering a sleep state for the rest of the transaction. Synchronised global transmission time slots are used to allow the nodes of the network to perform the initial routing setup when there are no known routes or neighbours.

B. Datagram Structure

Messages sent using the MAC protocol consist of a header, followed by a data section and ended with a network statistics section as seen in Table I. The message header contains information about the route the packet is on, how many hops it has gone through, information about the sending node's transmission slot and current slot as well as a message type flag. The message type flag indicates to the receiving node how to interpret the data contained in the data section of the message. Finally, the network statistics section of the message is used only to analyse the performance of the network and is not necessary for network operation.

The message header structure is illustrated in Table II. The Next Hop byte contains the ID of the next node along the route to the destination node, whose ID is contained in the Network Destination byte. The Local Source byte contains the ID of the node from which the message was most recently passed, while the Network Source byte contains the ID of the node from which the message originated. The two source ID's will be the same at the start of the lifecycle of a message, when it is generated. The Tx Slot contains two bytes that indicate the transmission slot number of the local source node, while the Current Slot contains two bytes which indicate to the receiving node in which slot the message was sent by the local source node. This information is used to ensure that the network remains synchronised.

The data section of the message can be interpreted in one of three ways, depending on the value of the Message Type byte in the message header. First, it can be interpreted as telemetry data, thus containing all the sensor data dispatched by the source node. Second, the data can be interpreted as a route request and thirdly a route reply. The route reply and route request messages will be further discussed in Section V.

V. NETWORK PROTOCOL

A. AODV

AODV (Ad-hoc On-Demand Distance Vector Routing) was used to achieve routing within our network [10]. The advantages of AODV include loop-free route discovery, quick repair of broken links, and avoiding the need for individual nodes to store information about the entire network. AODV makes use of route request (RREQ) and route reply (RREP) messages to establish paths within the network. A node wanting to find a route to another node (in this implementation a gateway node), will broadcast an RREQ packet to all its neighbouring nodes, (A neighbouring node is classified as any node that can receive a radio message from the specified node). The nodes that receive the RREQ will check to see whether they have a valid and up-to-date route to the destination node. If they do, they unicast an RREP packet back to the source node. If they do not, they rebroadcast the RREQ to all of their neighbours, and store reverse-path information indicating the way to the source node if an RREP packet is later received. Once the RREQ packet reaches its destination, or reaches a node that knows a valid route to the destination, an RREP packet is unicast back to the source node along the reverse path stored by each of the intermediate nodes along which the RREQ had travelled. Each of these nodes in turn set up forward path information to the destination node. Each node along the path from the source node to the destination node will now have a route to the destination node.

B. AODV Modifications

Some simplification to the AODV protocol is possible based on the needs of the overall system. Gateway nodes will never have to initiate a route discovery, and will only receive data from edge nodes. Edge nodes only need to store routes to gateway nodes, and not to other edge nodes, as an edge node will never be the final destination of a data message. Nodes must however store information about all of their neighbours, including their TX slots and their ID. This is because a node will not know whether it is an intermediate node along a route from another edge node to a gateway node until it receives a data message that needs to be forwarded along the route. When an edge node initiates a data transfer to a gateway node, it will look up the route to the gateway node in its routing table, and then send the message in the appropriate transmission slot. The source node will wait for an acknowledgement message from the node on the next hop along the route, and will then assume the message has successfully been sent. This eliminates the need for acknowledgement messages to be routed back from the destination gateway node. If an acknowledgement is not

received from the next-hop node, the transaction will be re-attempted in the next available global transmission slot. This is repeated until either an acknowledgement is received or the maximum number of permitted retries per message has been reached. Once this maximum is reached, a dead route is assumed and deleted from the sending nodes routing table. Then, a new route discovery sequence is initiated.

Each route recorded by an edge node has an associated expiration timer. The route will have to be checked by the edge node when the timer expires to ensure that the route is still active. The use of an expiration timer on a route allows the quick discovery of broken routes. The edge node updates its routing information by sending out a new RREQ with which to discover a new route to the gateway node.

VI. Experimental Setup

For initial testing, several statistics regarding the performance of the sensor network were collected. These include the number of times a node had to initiate a route discovery, the message overhead that is required for successful data transfer from an edge node to a gateway node, and the time it takes a node to discover a route to a gateway node.

In our experiments the nodes are configured to collect and transmit sensor data once every twenty minutes, as the environmental data does not change rapidly. The gateway node will thus attempt to upload the data collected by all nodes at the start of the twenty-minute window. This is to reduce the risk of a node sending a data message during the GSM upload. If the GSM upload fails, the gateway will store the data messages that have not been uploaded, and retry the upload halfway through the twenty-minute window. Each of the GSM upload slots is limited to a maximum of one minute, to reduce the chance of messages arriving during the upload. When the gateway is in the process of uploading GSM data, it is unable to receive any messages over the radio channel due to the processor not being able to service both modules at the same time. This can be overcome in future by implementing a dedicated co-processor for the GSM communication.

A total of five nodes were installed in a hillside vineyard in Stellenbosch, South Africa. They were arranged in a star configuration so that each node can see the gateway node. The node placement is shown in Figure 2. This was done to test the loop-free nature of the routing protocol, as it is still possible for neighbouring nodes to respond to RREQ messages even if the node doing the route request has direct contact with the gateway node.

The gateway node was installed and activated first, after which the remaining nodes were enabled one by one. After a node is powered on, it waits for a GPS lock to synchronise the onboard RTC. Once the RTC is initialised, nodes will start the route discovery process. In each global transmission slot (global transmission slots are open every 10 seconds), each node is given a 20% chance to send an RREQ, if no routes to the gateway node are known, otherwise, if at least one route to the gateway is known, the edge nodes will listen in the global transmission slot. This is to reduce congestion at the start of the network's life, or when multiple nodes are activated and start the route discovery process at the same time.

Fig. 2. Node placement in the vineyard.

After the nodes were installed this way, they were left for 48 hours to both record vineyard telemetry and to monitor the stability of the network by means of the data that was uploaded to the webserver.

VII. Experimental Results

A. Initial Observations

As expected, it was found that the nodes that were activated directly after the gateway node found routes to the gateway node quickly and with a small number of RREQ messages. For example, The first node to be activated after the gateway (Node *AB*) found a route in 21 seconds with only 1 RREQ message. The second node (node *DE*) to be activated found a route to the gateway node after 31 seconds, also using a single RREQ. The remaining two nodes experienced the situation described below.

In situations where two edge nodes can both see the gateway, but one has a much stronger link to the gateway than the other, a direct route to the gateway is not always found. For example, for nodes *AB* and *BC* in Figure 2, AB sends an RREQ that is serviced with an RREP by both the gateway node and BC. Due to the capture effect present in LoRa demodulation, the source node AB would often only hear the stronger RREP from BC. This would cause AB to set up a route to the gateway node through node BC. This situation causes unnecessary traffic on the network, since a direct route to the gateway is in fact possible. To address this, a priority system was introduced, where all nodes other than gateway nodes would perform a brief carrier sense on the channel before sending an RREP. This allows the gateway node to always send an RREP first if it is in range.

Another addition that improved the route discovery rate of the network, is the introduction of periodic HELLO messages

sent out by the gateway node. When an edge node receives a HELLO message and does not yet know a route to the gateway, it can add a route to its routing table. The edge node that received the HELLO message then does not have to initiate a route discovery. It also further prevented the situation described above, because when a node hears a HELLO message from the root, it can replace any other route to the gateway node with a direct link, if the node had stored a previously inefficient route.

Another situation that led to incorrect or inefficient routing data, was the hidden node problem. In this situation, two nodes hear an RREQ and both respond to it with an RREP. However, the two nodes are not able to hear each other. This can cause the source node to receive incorrect routing data.

B. Network Results

After the changes mentioned above, all nodes were reset. The network performed noticeably better, with all edge nodes now finding the most direct route to the gateway node. A summary of the initial route discovery statistics is shown in Table III. Each edge node required only a single RREQ to discover a route to the gateway node, except for node *BC*. This node heard a HELLO message before it began route discovery and therefore did not have to perform this sequence. The gateway node sent out three RREQ messages, further confirming that it received each of the RREQ messages.

After the initial discovery, the edge nodes sent telemetry data to the gateway node once in every twenty-minute time window. The expiration time on routes was set to sixty minutes and did not reset even when a data message was successfully sent. This was done to test the situation where a node has to rediscover a route to the gateway node. This was found to occur successfully.

After 10 hours of operation, during which edge nodes performed route discovery every hour, the edge nodes had sent 131 data messages to the gateway, and 148 messages in total. This demonstrates that even with the artificially high incidence of broken links precipitated by not resetting the expiration timers the network protocol overhead was only 12.9%. The 148 messages were sent within a single one second time slot each, with 3540 global transmission slots available in the same 10 hour period. It is clear that with only 4 edge nodes and one gateway node, the network is far from capacity.

C. GSM Upload

During the testing period, we observed that it took the gateway node required up to 46 seconds to upload the five stored messages to the webserver. This delay occurs because the GSM module has to perform one push request to the web server for each message, and response time from the webserver is dependent on multiple uncontrollable factors. Thus, the blocking nature of the GSM uploads is a key limitation in the number of edge nodes a gateway node can accommodate in the current implementation.

TABLE III
STATISTIC GATHERED DURING NETWORK INITIAL ROUTE DISCOVERY

ID	*AB*	*BC*	*CD*	*DE*
RSSI	-91	-79	-81	-65
# RREQ To Route	1	0	1	1
Time To Route	21	42	82	32

VIII. CONCLUSION AND IMPROVEMENTS

We have presented the design of an ad-hoc sensor network intended for real-time viticultural monitoring that uses an adapted AODV routing protocol to allow easy installation without prior configuration. Once placed in the vineyard and enabled, nodes independently discover routes to the gateway node quickly and without excessive routing overhead. The telemetry data that was gathered by each node was successfully sent to the gateway node and uploaded to a web server, where it could be viewed on a dashboard [11].

The upload of data using the GSM module could be improved by better scheduling to avoid the transmission slots of neighbouring nodes, or by offloading the task to a dedicated co-processor to avoid blocking any inter-node data traffic.

The network can be improved by adding additional priority levels to nodes during the route discovery process, where nodes that are closer to the gateway node (and therefore have routes with fewer hops), have a higher priority in sending RREP messages back along a path to a source node that initiated an RREQ. Nodes with a higher priority would then have to perform carrier sensing on the channel for a shorter duration of time compared to nodes with lower priority levels, allowing them to transmit RREP messages sooner.

ACKNOWLEDGEMENT

We thank Albert Strever and Talitha Venter for arranging access to the test vineyard. This research was supported by the Wine Industry Network of Expertise and Technology (Winetech). We thank Telkom South Africa for additional support.

REFERENCES

[1] A. Modawal, The Softweb Solutions Website, [Online]. Available: https://www.softwebsolutions.com/resources/vineyard-management-system.html.
[2] S. Marios and J. Georgiou, "Precision agriculture: Challenges in sensors and electronics for real-time soil and plant monitoring," in *Proc. IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2017, pp. 1–4.
[3] V. Adamchuk, J. Hummel, M. Morgan, and S. Upadhyaya, "On-the-go soil sensors for precision agriculture," *Computers and Electronics in Agriculture*, vol. 44, no. 1, pp. 71–91, 2004.
[4] "MSP432p401r Product Page," Texas Instruments, [Online]. Available: https://www.ti.com/tool/MSP-EXP432P401R.
[5] "RFM95W Datasheet," HopeRF, [Online]. Available: https://www.hoperf.com/modules/lora/RFM95.html.
[6] "Adafruit Mini GPS Module," Adafruit, [Online]. Available: https://www.adafruit.com/product/4415.

[7] "GSM-Click Product Page," MikroElektronika, [Online]. Available: https://www.mikroe.com/gsm-click.
[8] "TPS63001 Datasheet," Texas Intruments, [Online]. Available: https://www.ti.com/product/TPS63001.
[9] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proc.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, 2002, pp. 1567–1576 vol.3.
[10] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 1999, pp. 90–100.
[11] "ThingsBoard — Dashboard." [Online]. Available: http://meesters.ddns.net:8008/dashboard/a30f57f0-7d8b-11eb-9466-61843e0f4458?publicId=d7ab0d20-c43e-11eb-b1dd-0d4d650070a1

Nicholas Nell is currently a postgraduate research student at Stellenbosch University. He completed his undergraduate engineering degree at Stellenbosch University in 2019 and is currently studying the design of ad-hoc sensor networks for viticultural monitoring. His research interests include routing protocols, wireless sensor networks, and embedded software development.