An Energy Efficient Environmental Sensor Network for Data Collection

by

Andrew Campbell

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Science Honours

in

The Irving K Barber School of Arts and Sciences

(Honours Computer Science Major Computer Science)

The University of British Columbia
(Okanagan)

April 2012

©Andrew Campbell, 2012

Abstract

Wireless sensor networks allow real-time, environmental data collection but are limited by unreliable communication links, node failures, and battery life. A robust network must handle failure of nodes and adapt its behavior to conserve energy by powering off nodes when not in use. This leads to the problem of node synchronization, as messages cannot be sent or received during power down mode. This research constructed a complete wireless sensor network that allowed data from any number of distributed sensor nodes to be collected and displayed.

Challenges tackled include power management, node failures, synchronization, and adaptive routing of data messages. The system uses reference broadcasts to maintain synchronization, and gradient-based routing to send data towards the sink. In neither of these protocols do nodes need to have knowledge of the network beyond the nodes they can contact directly. This allows the network to be more adaptable to changes due to nodes entering and leaving the network. The developed system will greatly reduce the cost and complexity of environmental monitoring and data collection.

1 Introduction

Grapes, and wine grapes in particular, are a highly profitable crop. They are also a very fragile crop. Under or overwatering will degrade the quality of the grapes, and subsequently impact the taste of the derived wine. A larger consideration is temperature. Grapes need a minimum number of warm days to mature properly. If they do not have enough growing days, then they are not useful for wine production. Furthermore, if grapes become too cold, they suffer frost damage and are useless. Frost damage can be averted with circulation fans, which bring warmer air into the cold regions.

However, before a problem can be fixed, it must be diagnosed. Many vineyards take tempurature data from a local weather station, and extrapolate from there to estimate the tempurature of the entire field. This is not an ideal solution. Tempurature differences of up to 33% have been recorded in two different points in one hectare of a vineyard [4]. Ideally, we are gathering data at many points in the vineyard. This should be done automatically to keep labour costs down. As well, management of the circulation fans and other processes which can be automated should be automated. To do this, we need to set up a sensor network.

For our purposes, a sensor network would consist of a few sensor nodes scattered across the vineyard. A sensor node would be equipped with every sensor we need to manage the vineyard, and would send data to one specific sink node. The sink node can forward the sensor data to a server, which can analyse the data and deal with problems automatically. The nodes would need to communicate with each other over radio. Wires in a vineyard would cause too many problems. Data can be corrupted over long travel in wires, and wires are at risk of being damaged in the day to day process of running a vineyard. A wireless solution also avoids the overhead cost of installing an entire field of wires.

Wireless networks are not without their own problems. The most pressing of these problems are routing and power management. Wireless sensor nodes are forced to run on batteries, and improper management of resources will cause the batteries to drain quickly, reducing the viability and cost effectiveness of the network. The key factor in power management is managing a sleep/wake cycle, which requires nodes to remain sychronized. Nodes also lack a physical datapath, and as such the network must specify a routing protocol to gather the data. All of this must be done in a manner sturdy enough to handle node failure, changing radio ranges caused by weather, and other events which could change the topology of the network.

While research has been done on solving these problems, very little has been implemented in hardware. In this paper, we present our findings from building and implementing a wireless sensor network.

We first give an overview of previously done work in section 2. We then detail the routing algorithm we used in section 3. Section 4 describes our approach to clock synchronization, where we introduce two algorithms: flooding, and refernce broadcast. We applied our network to a field test to determine the relative value of each algorithm, and found flooding to be more effective. Details on this test are in section 5.

2 Background

2.1 Wireless Sensor Networks

The uses of wireless sensors has expanded with the rise of inexpensive radios and microchips. One of their most popular uses is in precision agriculture. Sensor nodes deployed en mass can diagnose a host of problems which would otherwise be ignored. Baggio [2] deployed a network of 150 nodes to monitor tempurature, groundwater level and humidity in a potato crop. The data was gathered to fight the spread of phytophtora, a fungal disease. A team from Austrailia's Commonwealth Scientific and Industrial Research Organization deployed a wireless sensor network by attaching the nodes to the necks of a herd of cows [16]. Data gathered allowed the team to track the herds movements, as well as gathering data on grazing habits and the health of the pastures the cows used for grazing.

The use of wireless sensor networks in vineyards is not an entirely new concept. Noteably, Bowen et. al. ran an extended test [4] on the economic feasibility of a wireless sensor network in an Okanagan vineyard. Their test used 65 Berkeley motes across a one hectare area to measure tempurature. The results tended to indicate that sensor networks were worthwhile in vineyard applications. The network was able to identify frost pockets, and found that different areas monitored by the network had vastly different number of growing days [3]. However, Bowens network was not self-organizing. Routing paths were hardcoded before deployment, which limits the extensibility and robustness of the network. This work was aimed at developing a similarily useful self-organizing network.

2.2 Routing in Wireless Networks

Finding a routing protocol for our network was mostly a matter of assessing the problem domain, as most routing algorithms for wireless networks are designed for very specific applications. Some routing algorithms are heirarchical [7] [14] [10]: these protocols assign certain nodes to communcate with a base station, and nearby nodes communicate only with the proxy. Baggio [2], after noticing that the range of his radios was severely diminished during flowering, moved to a heirarchical routing scheme, and placed routing nodes high above the plants to facilitate this. However, grape vines are held aloft by tall poles. Antenna can be placed on these poles, and the network can remain homogenous, as radio communication is not hampered by foliage. Therefore, communication with the sink is mostly a matter of geography. While geography based routing protocols exist [17] [18], routing explicity on physical position of nodes is impractical in a self-organizing network. Given that there are no obstacles to route around, the only use of physical to position is as a metric to estimate signal strength. It would be more practical to route on signal strength itself. The routing restraints our network faced are fairly lax, so our efforts went into routing with as little message passing as possible. A survey of routing methods was performed by Al-Karaki and Kamal [1].

2.3 Clock Synchronization

As clock synchronization is one of the most prominent issues for a wireless sensor network, there has been a great deal of research done to solve the problem. An overview by Sandararaman [15] compares several techniques, and describes several classes of solutions. The first distinction we had to make was between master/slave and peer to peer networks. In a master/slave protocol, one master node determines the clock times of network, and slave nodes attempt to synchronize to the master. The Timing-Sync protocol [8] is an example of a master/slave protocol. In peer to peer protocol nodes stay synchronized with their peers. Many peer to peer protocols involve the network electing a temporary master, and as such are easily converted into master/slave protocols.

Another distinction between synchronization protocols is untethered vs clock correction. In clock correction protocols, nodes adjust their clocks in order to keep them synchronized with the rest of the network. In untethered protocols, nodes make an estimate of their neighbours drift, and adjust their wake windows to synchronize their radios. Untethered protocols were unsuitable for our purposes, due to the larger memory footprint and the possibility of a larger wake window causing additional strain on battery life. Also, as the purpose of the network is to collect time-specific data, clocks should be accurate in order for data to be considered correct.

We also considered continuous clock adjustment. Implemented by Mock [9], continuous adjustment corrects for drift by modifying the tick rate of the clock, rather than setting the internal time to a new value. This is done to avoid missing time-specific events during readjustment. However, it incurs a high computational cost. It can also lead to inaccuracy, as drift rates on the real time clock are not constant. Instead, we delay clock adjustment until no critical events fall between the old and new time. The last set of protocols worth mentioning are MAC layer protocols, a survey of which was done by Chen [5]. MAC layer protocols were not considered for use, as the sensor nodes run IEEE 802.15.4 for wireless communication, which strictly defines the MAC layer, and does not include synchronization.

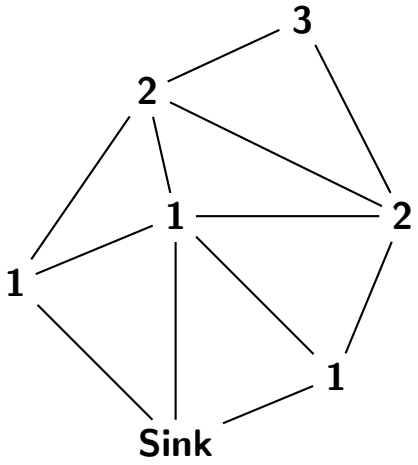
3 Routing

For a self-organizing network, the first problem to overcome is routing of nodes. In a self-organized multihop network, routes must be generated by each node on the fly. Since nodes only store data on their immediate neighbours, these routes must be generated without global knowledge of the network. In short, a node cannot choose the exact route a message takes, it only knows which node to hand the message off to. Without a routing method, messages can take an inefficient path to the sink, or can travel endlessly in a loop.

3.1 Gradient Routing Scheme

Our solution to the routing problem was to implement gradient routing, as described by Schurgers and Srivastava [11]. Every node in gradient routing is assigned a rank, which denotes the minimum number of hops a message must take to go from the source node to the sink. An example of a network divided into gradient is given in figure 1, where edges are stable lines of communication and vertices show their rank. Nodes know their own rank, and the ranks of their neighbours. When it comes time to send a message, nodes just send to a node with a lower rank then their own. If they are unable to send a message to a lower ranked node, they raise their own rank and try again.

Figure 1: A Gradient Ranked Network



The vineyard application allows the routing scheme to be extremely simple. Nodes do not move, and so the network is relatively static. This means that nodes will change their rank few times. Nodes will wake up from sleep, and their routing information will be mostly correct, as the network topology will be the same as when they went to sleep. Furthermore, the antenna can be mounted on poles used for holding the vines. The effective broadcast range for the radios will increase, which could lower dramatically the amount for hops needed to get a message to the sink. Methods used to reduce the number of messages, such as cluster heads, are therefore unnecessary.

3.2 Routing Table

Every node stores a routing table, which is a list of all nodes that the node has received a message from. The table is sorted by desirability as a routing candidate. Each time a message is received, the routing table is reordered to reflect new information in the message. The table is sorted on two parameters. First, the lowest ranked nodes go to the top of the list, and within ranks, nodes with the highest score are brought to the top. Currently, the scoring metric is the average RssI of the messages sent from that node. The score is either positive or negative. On wakeup, all scores have the same sign. If a message is received, the score of its source switches signs. When sorting the routing table, the scores are multiplied by either 1 or -1, such that the score for a node we have heard from is positive, and nodes we have not heard from have negative scores. This encourages nodes to forward messages to nodes they know to be alive.

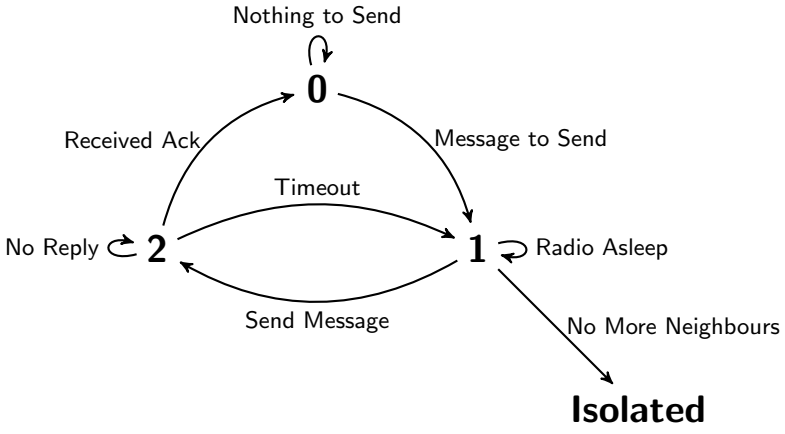


Figure 2: Send State Machine

Nodes, when sending messages, follow the state machine described in figure 2. Nodes wait until sensors have gathered enough data to send, or they receive a message which needs to be sent forward. The node then waits until the radio is awake, and sends the message to the first node on its routing table. The node then waits for an acknowledgement. If the node receives one, it is done: the message is stored at the new node. If no acknowledgement is received, we assume that the message was lost, and we send to the next node in the routing table. If we run out of nodes, we assume we are isolated, and stop sending.

3.3 Load Balancing

Sending and receiving messages has an energy cost. While we always want to route through a strong, stable link, we also want to avoid sending too many messages through one node. The overtaxed nodes batteries will quickly drain, and the node will be out of commission until the batteries are replaced. This is obviously undesirable. The routing technique must therefore include a measure of load-balancing in order to extend the longevity of the network. Currently, the nodes are scored based on their signal strength. This is more a matter of convenience than a conscious effort to expand the capabilities of the network. Routing based on signal strength is an inexpensive way to send messages down paths likely to succeed, but does not conserve energy or spread the work across the gradient.

The obvious solution would be to route based on energy, rather than signal strength. The solution is utilized in the energy-aware routing protocol [12]. In energy-aware routing, nodes maintain several paths in memory. A node picks a path randomly, adjusting to favour paths with a low energy cost. The explicit goal of this protocol is network longevity, and it provides a 44% increase in lifetime [12]. However, this protocol requires global knowledge, which is prohibitively expensive in a self-organizing network. Schurgers [11] experimented with energy based routing by having a node increase its own rank if its remaining energy fell below a certain threshold. While this balanced energy loss in the network, the gains were not as dramatic as the energy-aware protocol.

One possible strategy would be to score based on remaining energy, rather than signal strength. This would be a low cost solution, as nodes can piggyback their energy on synchronization messages, which are broadcast across the network anyway. The obvious drawback of this solution is the risk of sending across weak and unstable links. A score based on both signal strength and energy is a possibility worth researching, as it has the possibility of marrying energy demands and link stability.

4 Clock Synchronization

The largest issue in all long-term wireless sensor networks is battery life. Nodes running on batteries have a limited lifespan, which is shortened by running devices and sending messages. The CSIRO [16] team avoided this by judisious use of solar panels, but for networks set up in places less sunny than inland Austrailia, conservation of battery life is key to the longevity of the network. Battery life is conserved mostly by keeping all devices in a sleep state until used. Applying this logic to the radio creates additional complications; radios cannot receive messages while in sleep mode. The radio cannot be awake only long enough to send data, but rather needs to be awake for a short window to receive and forward messages.

In order for the network to function properly, all radios in the network should be awake at the same time. For this to be feasible, the nodes must be synchronized. Synchronized clocks are possible on a hardware level: equipping nodes with GPS transeivers, for example, could ensure a consistent

time across the network. This would, however, be prohibitively expensive, both from an economic and energy standpoint. Economically viable real-time clocks suffer from drift, where different clocks will tick at different and inconsistent rates. If such clocks are used, the network must deploy with a synchronization routine to keep nodes in line.

4.1 Algorithms Used

The most important factor used in choosing an algorithm for clock synchronization was the limitations of the hardware. If an algorithm is especially calculation or memory intensive, it would be infeasible to implement the algorithm on the sensor nodes and expect them to work properly. For example, any algorithms which required use of floating point operations were out of the question, as floating point libraries took up too much of the program space on the microcontroller. Several times during development features were dropped due to lack of available heap or stack space. As well, given the timing driven nature of the application, algorithms needed to finish quickly, with as few operations used as possible. In the end, we decided to implement two algorithms, and test them both in the field to determine their stability.

4.1.1 Global Time Flood

The first algorithm, which we refer to as the flooding algorithm, attempts to impose a global time on all nodes in the network. The flooding algorithm begins with the sink node waking from sleep, and waiting a short time to account for stragglers. The sink then broadcasts a flooding message, which contains the sinks current time. Nodes within earshot pick up the message, immediately set their internal clocks to match the time in the message, and then broadcasts a replica of the message it received. The flood message will propagate outwards from the sink until the entire network has received the message.

The flooding algorithm as implemented is a simplification of the time-diffusion protocol [13]. Unlike TDP, the flooding algorithm always begins at the sink, rather than having the network elect a root every iteration. This lowers the code space needed to implement the algorithm, and most importantly cuts down on the number of messages sent across the network. This optimization is only possible because our network already has a dedicated sink node. The algorithm is also simplified by removing the average aspect of TDP. In TDP, nodes use an average to determine how the clocks should be changed. This helps to account for uncertainty in message propagation. In lab tests, we found the system to be more stable if the time was reset instead. However, this could lead to a problem of propagation delay; since we no longer account for the time needed to propagate messages, nodes farther afield will be behind the sink node by unpredictable amounts. In our admittedly limited trails, this delay was negligible.

4.1.2 Reference Broadcasts

The second algorithm uses reference broadcasts to generate local times for each node. Rather than keeping the whole network on one time, this algorithm keeps the network synchronized by keeping every node in sync with its neighbours. At a certain predetermined point of time, every node will broadcast a reference message. Nodes within earshot will hear the message and record the time. When they have a timestamp for every node in their neighbourhood, it takes an average of all the timestamps which gives it a rough idea of how far removed its clock is from the clocks of its neighbours, and it can adjust accordingly. In this way, nodes remain sychronized with their neighbours, and the network remains in sync without the need for global knowledge.

	Reference Broadcast	Flooding
Messages Expected	186	186
Messages Received	1	172

Table 1: Messages received in 62 cycles

The reference broadcasts, as implemented, are based on Reference Broadcast Synchronization algorithm of Elson et. al [6]. In RBS, one node sends a reference broadcast, and other nodes record when the reference was heard. They then exchange this information with their neighbours in a pairwise fashion. This would require extra messages for node discovery and pairwise sharing, and would need a designated reference node. We simplified the algorithm by having nodes synchronize to a single logical time, rather than a reference signal. Rather than assuming a message was heard by all nodes at the same time, our algorithm has nodes assume all reference messages are sent at a specific time. This reduces the number of messages in the air, but fails to account for uncertainty in transmission time. Our algorithm also adjusts the clocks, rather than letting the nodes run untethered.

5 Field Test

While both algorithms are viable in theory, and in lab conditions, they still had to be tested in a field application. The purpose of the tests were to determine the relative stability of the algorithms: how reliable the algorithms were without adding any additional challenges. The basic setup of the test was a small sensor network, consisting of 3 sensor nodes and one sink node. The radios would sleep for 55 seconds, and then wake for 5 seconds, during which they ran their synchronization algorithms and sent one meaningless data message to the sink. We measured the usefulness of the algorithm by how many messages the sink received. The sink would expect 3 messages per sleep/wake cycle, and anything less would indicate a node has gone out of sync.

The test bed was set up on a vineyard in West Kelowna. The nodes were placed in such a way as to have 2 ranks, with two nodes in rank 1 and one node in rank 2. The rank 1 nodes were placed in adjacent rows to ensure that they were both in range of the rank 2 node, the sink, and each other. This setup allowed the network to be tested in a situation which required multi-hop routing and did not have a single point of failure. The hardware platform was developed in-house. The nodes were controlled by an ATMEGA644P microcontroller, communicating with XBee radios, and used a Thomson M41T81S serial access real time clock.

The results of the test are in table 1. The reference broadcasts performed very poorly. In the trial, the nodes were able to communicate during the network setup, but no messages were able to get through past this. The extent of this would suggest that reference broadcasts are too risky a proposition for use in precision agriculture sensor networks. The flooding algorithm was imperfect, but the missing messages were lost in the middle of the trial, rather than the end. This indicates that a node fell out of sync, but was able to reenter the network. The performance of the flooding algorithm, along with the proven ability for nodes to renenter, makes flooding the prefered choice of algorithm. However, 2 ranks is a trivially small network, and we still do not have data on how the algorithm would perform on a wider scale.

6 Conclusion

In this paper, we described a simple self-organizing system for a precision agriculture wireless sensor network. The key to the systems effectiveness was simplicity. Due to hardware limitations and timing-sensitive nature of the project, complex algorithms which perform better in theory often fail to meet the standards needed in a real world application. In the end, the best solution was to maintain a global time with a network flood, and to use gradients for routing. This conclusion is supported by field trials. The system lacks a load-balancing heuristic, which could be the subject of further study.

References

[1] J.N. Al-Karaki and A.E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6 – 28, dec. 2004.

[2] Aline Baggio. Wireless sensor networks in precision agriculture. *Precision Agriculture*.

[3] R. Beckwith, D. Teibel, and P. Bowen. Report from the field: results from an agricultural wireless sensor network. *29th Annual IEEE International Conference on Local Computer Networks*, pages 471–478.

[4] R. Beckwith, D. Teibel, and P. Bowen. Unwired wine: sensor networks in vineyards. *Proceedings of IEEE Sensors, 2004.*, pages 561–564, 2004.

[5] Jyh-Cheng Chen, K.M. Sivalingam, P. Agrawal, and S. Kishore. A comparison of mac protocols for wireless local networks based on battery power consumption. In *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 150 –157 vol.1, mar-2 apr 1998.

[6] Jeremy Elson, Lewis Girod, and Deborah Estrin. Fine-grained network time synchronization using reference broadcasts. *Proceedings of the 5th symposium on Operating systems design and implementation - OSDI '02*, page 147, 2002.

[7] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, page 10 pp. vol.2, jan. 2000.

[8] Ram Kumar and Mani B Srivastava. Timing-sync Protocol for Sensor Networks. *Discovery*, pages 138–149, 2003.

[9] M. Mock, R. Frings, E. Nett, and S. Trikaliotis. Continuous clock synchronization in wireless real-time applications. In *Reliable Distributed Systems, 2000. SRDS-2000. Proceedings The 19th IEEE Symposium on*, pages 125 –132, 2000.

[10] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 166–179, New York, NY, USA, 2001. ACM.

[11] C. Schurgers and M.B. Srivastava. Energy efficient routing in wireless sensor networks. *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*, 1:357–361.

[12] R.C. Shah and J.M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 350 – 355 vol.1, mar 2002.

[13] Weilian Su and Ian F Akyildiz. Time-Diffusion Synchronization Protocol for Wireless Sensor Networks. *Time*, 13(2):384–397, 2005.

[14] L. Subramanian and R.H. Katz. An architecture for building self-configurable systems. In *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pages 63 –73, 2000.

[15] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, May 2005.

[16] Tim Wark, Peter Corke, Pavan Sikka, Lasse Klingbeil, Ying Guo, Chris Crossman, Phil Valencia, Dave Swain, and Greg Bishop-Hurley. Transforming Agriculture through Pervasive Wireless Sensor Networks. *IEEE Pervasive Computing*, 6(2):50–57, April 2007.

[17] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 70–84, New York, NY, USA, 2001. ACM.

[18] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical report, 2001.