Murphy Loves Potatoes
Experiences from a Pilot Sensor Network Deployment in Precision Agriculture

Koen Langendoen Aline Baggio Otto Visser

Delft University of Technology, The Netherlands
Faculty of Electrical Engineering, Mathematics, and Computer Science
{K.G.Langendoen,A.Baggio,O.W.Visser}@tudelft.nl

Abstract

We report on preliminary experiences with deploying a large-scale sensor network (about 100 nodes) for a pilot in precision agriculture. The pilot did not answer the initial research questions, but instead revealed many engineering problems typically overlooked by (computer) scientists evaluating their work by means of simulation. The deployment prompted us to rethink our development process and includes important lessons for the WSN research community as a whole.

1. Introduction

According to Murphy's law, anything that can go wrong will go wrong. And so it did. In 2004, researchers from Wageningen University, Delft University of Technology, Vertis, and Opticrop teamed up in an ambitious project to use 150 wireless sensor nodes in a three-month pilot deployment on precision agriculture. Because of lack of experience with running a sensor network application on such a scale, unattended for such a long time, and in harsh outdoor conditions, we were anticipating a considerable number of problems to emerge throughout the duration of the project (Q3'2004 - Q3'2005). We expected that most problems could be handled through simple software updates[1], that some would require labor-intensive hardware modifications, and that only a few would be impossible to resolve. Therefore, we were confident that the pilot would bring us lots of interesting data, both from the agricultural perspective, i.e. "what do microclimate measurements reveal?", and from the computer-science perspective, i.e. "how do network protocols perform in an outdoor environment?" How wrong we were! The endless stream of hardware malfunctions, programming bugs, software incompatibilities, and plain misunderstandings, combined with the time pressure of Mother Nature, has left us one year later with a meager harvest of quantitative results. One could declare the project a failure, but on the other hand, we have learned a lot – the hard way – and we would like to share our experiences with the research community to avoid a repetition of mistakes that can easily been avoided if only one is aware of their existence. Note that although experiences about previous pilots have been reported [5, 8, 10, 13], these publications in general stress technical issues like low-level network performance instead of the (basic) software-engineering problems that made running our project so difficult. In addition we identify a number of issues with the current generation of MAC and routing protocols for wireless sensor networks that need to be addressed before large-scale, long-running unattended deployments can become a reality.

[1] If need be, even during deployment through wireless reprogramming.

2. LOFAR-agro

The LOFAR-agro project[2] is the first large-scale experiment in precision agriculture in The Netherlands. This pilot project concerns the protection of a potato crop against *phytophthora*, a fungal disease that can spread easily amongst plants and destroy a complete harvest within a large region. The development and associated attack of the crop depends strongly on the climatological conditions within the field. In particular, humidity and temperature within the crop canopy are important factors in the development of the disease. To monitor these critical factors, we instrumented a potato field with wireless sensors. A close monitoring of the microclimate can reveal when the crop is at risk of developing phytophthora and

[2] http://www.lofar.org/p/Agriculture.htm

1-4244-0054-6/06/$20.00

allows the farmer to treat the field, or parts of it, with fungicide only when absolutely needed. This precise treatment saves time, reduces costs, and limits the use of environment-unfriendly substances as opposed to traditional treatment based information from a remote weather station. The objectives of the LOFAR-agro pilot are threefold:

- to obtain experience with deploying large-scale (100+ nodes) wireless sensor networks for precision agriculture;
- to gain insight into the variations of the micro-climate within a field that can be measured with commodity sensors (i.e. to determine the optimal spatial resolution for future deployments);
- to assess the feasibility of precision agriculture for future crop production in The Netherlands (i.e. cost-vs.-benefits analysis).

The participation of Delft University of Technology in the LOFAR-agro project focused on the deployment aspects. The pilot gave us a unique opportunity to experience the engineering aspects associated with deploying a real system, and to study how protocols developed by means of simulation perform in practice. In particular, our research objectives were:

- to study the influence of the environment (e.g., humidity and foliage) on radio communication (e.g., effective range and link quality) during the three-month growing season;
- to determine the accuracy of existing localization techniques in a real-world setting (through off-line simulations using traces of observed RSSI measurements);
- to evaluate the performance and robustness of our in-house developed T-MAC protocol [1], which employs an adaptive duty-cycling mechanism on the radio to conserve energy.

In practice, Delft University of Technology was responsible for transporting sensor data from the individual nodes, through a gateway server at the edge of the field, to the various partners.

3. Deployment setup

Figure 1 shows the overall organization of the LOFAR-agro deployment. The sensor nodes in the field measure relative humidity (RH) and temperature (T) once per minute. Ten samples are encoded in a single packet, which is send – directly, or through multiple hops – to a gateway at the edge of the field. The gateway uses standard Wi-Fi to forward the sensor data over the LOFAR backbone network to the Agro server, which logs the data, filters out

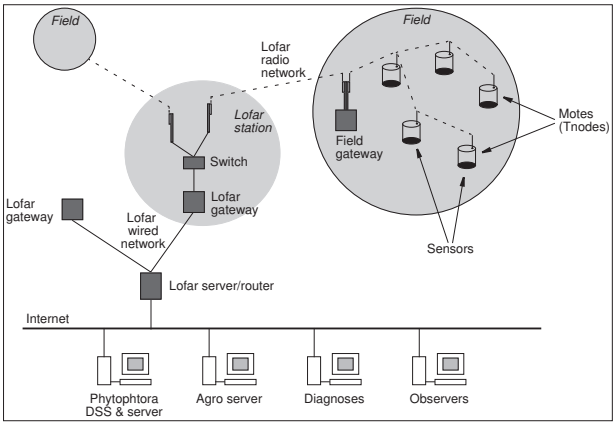

Figure 1. The LOFAR-agro setup.

erroneous readings, and hands the accumulated data to the Phytophthora decision support system (DSS) server. This decision support system combines the field data with a detailed weather forecast to determine the treatment policy, which is then provided to the farmer (one of the observers in the system).

The normal farming routine requires that a treatment advice is available every morning when the activities for the day are planned. This imposes a soft real-time constraint on the complete system. Sensor data need not be made available immediately, but can be buffered for up to one day. Due to timing and manpower constraints we have decided not to explore the loose timing requirements, but to use "standard" sensor network protocols that forward data packets when they become available. Details are presented below where we discuss the hardware and software configuration that was used within the LOFAR-agro deployment.

3.1. Hardware

At the start of the project (Q3'2004), there were no readily available sensor nodes that could be used outdoors (weatherproof casing) and could measure the microclimate (T/RH). Therefore, we had to assemble the nodes by hand out of individual components. Figure 2 shows one of the 109 nodes used in the LOFAR-agro pilot.

The sensor node is based on the TNOde platform[3], which is quite similar to the popular Mica2 platform, but with the Mica2Dot form factor. A TNOde together with a battery is packed into a waterproof PVC-based casing. The 7 cm ($\lambda/4$) antenna is placed on top

[3]Developed by TNO (the Dutch Organization for Applied Scientific Research).

CPU	ATmega128L (8-bit, 8MHz) 128KB Flash memory 4KB DRAM memory
Storage	4Mbit EEPROM memory
Radio	Chipcon CC1000 868 MHz, FSK 76.8 kbps, output power -20 to 5 dBm
I/O	RS232, 3 LEDs 5 general purpose I/O pins
Sensor	Sensirion SHT75 T $\pm 0.3\,°C$, RH $\pm 1.8\%$
Battery	Sonnenschein SL-770/T C-cell, 3.6V, 7.2Ah

Figure 2. A LOFAR-agro node (image and specs).

and the microclimate sensor is connected by a cable (I^2C) at the bottom. Experience has shown that the radio range is dramatically reduced when the potato crop is flowering and leaves cover the (ground-based) antennas [11]. Therefore the nodes are installed on poles at a height of 75 cm. The potato plants grow to about 100 cm, however, we had to include a safety margin to ensure that the nodes could not be hit by farming equipment attached to a tractor. The cost of a node is about €250.

The gateway at the edge of the field is based on a Stargate from Crossbow [9], which is equipped with a 400 MHz X-Scale processor, a 256 MB CompactFlash card for backup storage, a PCMCIA Wi-Fi card connecting to the LOFAR backbone network, and one TNOde communicating with the sensor nodes. This TNOde is connected to a five-meter-high antenna to ensure that most nodes in the field can be reached directly. The gateway is powered by a solar panel in combination with a rechargeable battery to ensure operation around the clock. The total cost of the gateway is around €1500.

3.2. Software

All TNOdes run a customized TinyOS version which handles the minor differences between the Mica2 and TNOdes hardware layout. The image for the field sensors includes the following components:

T-MAC For the medium access control layer we favored the T-MAC protocol [1] over the standard B-MAC implementation that comes with TinyOS, because T-MAC was developed in-house and we had good experiences with it in laboratory conditions. T-MAC is tightly integrated with the power management component of TinyOS and puts the radio *and* CPU to sleep whenever there is no traffic to send or receive. It uses an adaptive duty cycle, which was shown to outperform S-MAC, the third alternative MAC protocol, in previous simulation studies [1]. We configured T-MAC to run with a slot time of 610 ms, an active period of 65 ms, and a period of 6.1 s between sending out synchronization (SYNC) packets. Once every 300 slots, a node keeps on listening during the whole slot to scan for any (new) neighbor running on a different (skewed) sleep/wake-up schedule. This results in an effective duty cycle of 11 %.

MintRoute Even though we used a sensitive, long-range antenna at the gateway, we included a multihop routing protocol to overcome possible obstructions by the growing potato plants and to ensure that nodes on the far edge of the field could report their microclimate readings. We selected MintRoute since it had been successfully used in previous experiments by researchers from UC Berkeley [6, 12]. Alternate routing protocols like (Tiny)AODV seemed too complex and would use more data memory than desirable, causing the complete application to exceed the 4 KB RAM limit. MintRoute sets up a spanning tree towards the sink node – node 0 at the gateway in our setup – and builds statistics of neighbor's activity and packet loss counts. These statistics enable MintRoute to determine the best parent amongst a set of neighbors. We configured MintRoute to operate with our application data rate of 1 packet per node every 10 minutes.

Deluge Since we were lacking experience with long-running sensor applications in outdoor conditions, we decided to enable wireless reprogramming using the Deluge software [3], anticipating bug fixes and software upgrades. Deluge uses an epidemic protocol to disseminate a new code image through the network. Parts of a new image are stored in EEPROM and a node is rebooted when the image is complete. A drawback of Deluge is that it occupies quite a large fraction (50 %) of the EEPROM, and was not designed to coexist with other application components using the EEPROM[4].

LOFAR-agro application The application code running on top of MintRoute is responsible for reading the sensor data once every minute,

[4]EEPROM coexistence of Deluge is provided in the 2.0 version released August 2005.

compressing the data into a 22-byte payload once every 10 minutes, and sending it out to the gateway in a standard TinyOS message of 29 bytes including headers. Since MintRoute, in combination with T-MAC, provides a best-effort service, the application should also take care of end-to-end reliability of the data transport. We envisioned a scheme where an individual sensor node logs all data into EEPROM using a cyclic buffer. The gateway informs the nodes once a day which packets are missing from which nodes with a network-wide acknowledgment. This unified acknowledgment scheme imposes little overhead, provided that the packet error rate is rather small. Due to timing constraints we never got around to use it in the real deployment, which resulted in data not being delivered to the DSS server but remaining in EEPROM for postmortem analysis.

Note that the code image does not include any component for localizing the sensor nodes. We decided to manually determine the positions of the individual nodes when placing them in the potato field (accuracy of a few centimeters). We did, however, include some code to collect statistics about the RSSI values of messages received from a maximum of 10 neighbors. These statistics (min, avg, max) were sent to the gateway four times a day, and would be used in an offline trace-based simulation of various localization algorithms. We planned to also instrument MintRoute and T-MAC to collect and send statistics such as routing table entries and number of (lost) messages, but we never managed to get this "luxury" implemented due to lots of unforeseen factors. Nevertheless, afterwards we were able to reconstruct the routing tree to a large extend out of information on the first-hop neighbor piggybacked with data messages. Figure 3 shows a snapshot of such a reconstructed spanning tree.

4. Project history

Rome was not built in one day, and neither was the LOFAR-agro setup. Worse, we never finished it completely due to hard limitations on time and money, on the one side, and a long list of unfortunate events on the other. To put things into perspective, this section provides a short history of important events, as they took place during the LOFAR-agro project. Table 1 provides a concise overview.

July 2004 – project launch: From the first contacts between the LOFAR-agro partners in January 2003, it took more than a year to obtain funding for running a small pilot in precision agriculture with a budget of just K€300. In July 2004 we started planning for the LOFAR-agro deployment during the growing season of 2005. Around October it was decided to go with the TNOdes/TinyOS combination as outlined in Section 3. Deployment was planned to start early April 2005.

Date	Event
July 2004	Project launch
December 2004	TNOdes 1st revision
January 21st, 2005	Indoor trials commence
March 8th, 2005	TNOdes 2nd revision
March 16th, 2005	Field trial
May 4th, 2005	Field trial 2
May 16th, 2005	Potatoes planted
June 1st, 2005	Deployment start
June 5th, 2005	Batteries run out
June 22nd, 2005	Deployment restart
July 1st, 2005	Backbone connection outage
July 14th, 2005	Network dies out
August 16th, 2005	Potatoes harvested

Table 1. Major events during the LOFAR-agro project.

December 2004 – TNOdes 1st revision: Early December the first 10 TNOdes were delivered for testing to the LOFAR-agro partners. Since the TNOdes design is largely compatible with the Mica2 nodes from Crossbow, we only observed minor difficulties with the 1st revision hardware. For example, a capacitor needed to be replaced to allow for smooth operation of the radio in the 868 MHz band.

January 21st, 2005 – gateway operational: In Delft we managed to create a local loop from the gateway (inserting fake T/RH data), through an IP tunnel, to a server machine logging and converting data to XML format for processing by the Decision Support System. Uploading and getting software to work on the gateway (Stargate board) proved to be rather easy.

March 8th, 2005 – TNOdes 2nd revision: Only three weeks before deployment should commence (according to the original planning), the 150 TNOdes modules (rev. 2) were supplied. The late delivery was caused by a number of problems at the assembly factory (e.g., unavailable components and "lost" parcels). Fortunately, Mother Nature was on our side bringing snow to the Netherlands, which delayed the start of the growing season.

March 16th, 2005 – field trial: Even though we had only one week time to test out the new hardware, it

Figure 3. Reconstructed spanning tree; it is incomplete due to missing packets containing first-hop data.

was decided to go to the deployment site (a three-hour car drive from Delft) to see if we could put together a working system consisting of 5 sensor nodes and a gateway. We could not, for a number of reasons. First, the casing of the gateway was made by the partners from Wageningen, and we soon discovered that they used a different WaveLAN card for connecting to the backbone network. Our card, capable of connecting to an external directional antenna, was slightly larger than theirs and would not fit into the casing without relocating the mounting brackets of the main Stargate board. Note that we did communicate the dimensions of the WaveLAN card, but forgot to include room for the connector of the external antenna. The second problem that surfaced had to do with the sensor nodes. Four out of five nodes could not read the T/RH values. This turned out to be caused by a difference in the wiring of the cable connecting the microclimate sensor and the TNOdes board; the specification of the connector on the PCB was mirrored, so only the cable supplied by TNO was working. Finally, we discovered that the antennas were quite fragile and would come loose easily when inserting them into the plastic casing. In short, time ran out on this day and we never got a TNOde to report any data back to Delft, but we did manage to get the IP connection over Wi-Fi to work.

May 4th, 2005 – field trial 2: Since we encountered so many problems we decided to do much better testing at home, before going out to the field again. This was a wise decision since we encountered numerous problems with the T-MAC protocol when operating it continuously for more than a day (which is much longer than the typical one-hour demonstration). Apart from your regular set of bugs (e.g., storing a negative clockdrift in an unsigned variable), we discovered that putting a node's CPU really into sleep is far from trivial. For example, you have to make sure that all critical timing code is associated with the low resolution timer that remains functional during processor sleep. This is not the case in the standard TinyOS setup, and finding all exceptions requires a more than thorough understanding of the TinyOS structure and its components. By the end of April we had to decide to revert back to a version of T-MAC without power management, so we could have another field trial before the potatoes were planted.

The second field trial took place on May 4th. This was two months behind schedule, but still before the potatoes were planted (May 16th). Although we brought working hardware to the field, we again did not succeed in getting any sensor data back to Delft. This time we were hit by a lack of proper software management. The student working on T-MAC (desperately trying to get power management to work) committed a partial update into the shared CVS repository the night before deployment, which we then flashed into the sensor nodes on the next morning. It took a long time to figure out what went wrong because we set all our software to production mode, so no LEDs were blinking (to save power) and data rates were set to 1 packet every 10 minutes. Without decent debugging tools like a packet sniffer, it proved very hard to diagnose the cause of the lack of incoming data at the gateway; trial and error with opening/closing waterproof casings is not to be recommended, but wireless reprogramming (Deluge) is impossible with a malfunctioning MAC protocol.

June 1st, 2005 – deployment start: Following the planting of the potatoes on May 16th, and some subsequent farming activity, the LOFAR-agro deployment was officially started on June 1st. The placement of the sensor nodes was witnessed by the press, including four camera crews giving regional, national and even international coverage to the pilot study. Given that the software was far from stable, we were happy to see that slowly, one by one, the sensors were reporting data to the gateway. Further analysis, however, identified a number of problems that needed urgent attention. First, the network suffered from major packet loss. This was in part caused by Deluge flooding the network with a superfluous code update[5], which caused unwanted contention with ordinary data messages (T-MAC is designed for low data rate applications and collapses when load increases, see [4]). Second, MintRoute was ill-behaving and introduced (very) long paths for nodes that were in direct contact (i.e. one hop away) from the gateway. Before we could analyze what was going on exactly, the field turned silent on the evening of June 5th when heavy thunderstorms crossed the Netherlands. To our relief a site visit a few days later revealed that the nodes had simply ran out of batteries on that evening. Due to delivery problems of the high-capacity batteries, we had to resort to small batteries holding only 2200 mAh. The lifetime of only four days matches with nodes running without any duty cycling of the radio and processor. Apparently, the Deluge updates kept everybody awake all the time, and MintRoute forcing multiple hops in a single cell did not help either.

We did not rush out immediately to replace the batteries, but instead tried to determine what caused the poor behavior of our sensor network. Simulations with TOSSIM confirmed MintRoute's ill behavior, and allowed us to pinpoint an underlying problem. MintRoute maintains a fixed-sized list of neighbors, of which it selects the "best" to become its parent in the spanning tree to the route. On the field about 70 nodes form a single cell around the gateway, which forces MintRoute to make a selection since it has room for only 16 nodes in its neighbor list. When adding neighbors, the replacement policy is as follows: MintRoute picks the neighbor with the lowest statistics value and evicts it from the table. As a consequence, the gateway is *not* part of the neighbor list for the majority of nodes. A related problem is that T-MAC was maintaining a separate neighbor list with only 20 neighbors and *no* replacement policy (first come, first serve). Whenever MintRoute directs a packet to a neighbor unknown to T-MAC, that packet is dropped. This problem was aggravated by transporting the nodes together to the potato field, causing a node to pick up an arbitrary set of neighbors, some of which were not even in-reach after final placement on the field. The need for maintaining a consistent neighborhood across modules has been recognized by others [7].

[5]We conjecture that we failed to erase some nodes' EEPROM image holding a previous version of the LOFAR-agro software.

June 22nd, 2005 – deployment restart: It took another two weeks to fully analyze and remedy the neighbor problems of T-MAC and MintRoute. In the case of T-MAC we removed the legacy neighbor lists (inherited from the S-MAC implementation on TinyOS) since the protocol only needs to listen to schedules of neighboring cells and can simply transmit on the schedule of its own cell [1]. In the case of MintRoute we overrode the parent selection when overhearing a (broadcast) message from the gateway.

On June 22nd it took six people a complete day to replace all the batteries and code images of the 110 sensors on the field. From sequence numbers rewinding to 0 we inferred that the deployment was hampered by a large number of node resets. The software was configured to use a watchdog timer to guard against software problems, but for – yet unknown – reasons perfectly fine running nodes would spontaneously reset at an alarming rate once every 2–6 hours. This proved catastrophic for two reasons: 1) we rarely received any daily statistics, and 2) MintRoute was quick to delete a node from its neighbor list when observing a sequence number rollback, but very reluctant in accepting new nodes because of the activity condition effectively locking out reseted nodes (without a route you can not increase your activity!).

Another problem that surfaced after the deployment restart affected the gateway. When selecting the solar panel and rechargeable battery, the partners from Wageningen estimated the power consumption based on the assumption that the Wi-Fi card would only operate once every ten minutes. The people from Delft implementing the software on the Stargate were not aware of this and assumed that there would be a permanent wireless connection. Since the rechargeable battery had a large capacity, it took a long time before it was fully drained, but then the Stargate would lose power somewhere after midnight and wake up again a few hours after sunrise. This is unfortunate since dawn is the most critical period from the agricultural perspective as humidity is then at its peak.

July 14th, 2005 – network dies out: Around three weeks into the (restarted) deployment most sensor nodes ran out of batteries. Back-of-the-envelope

calculations reveal that either T-MAC's duty cycling has proved completely ineffective, or another effect is causing nodes to consume more energy than anticipated. We observed that remote nodes lasted about one week longer (four weeks in total). We conjecture that the difference is caused by overhearing less traffic (5 neighbors vs. 70 neighbors), but then the small (33%) difference in lifetime indicates that there must be another factor contributing significantly to the nodes' power consumption. The last message received by the gateway is dated August 8th.

August 16th, 2005 – potatoes harvested: The day before the potatoes were harvested, all sensor nodes were removed from the field. At that moment we were still hopeful that we could recover a lot of agricultural data from the logs stored in the EEPROM. Now we know that a silly programming bug caused all EEPROM writes to fail silently, even though the unit test for logging to EEPROM worked fine. Thus we are left with the limited amount of data that was streamed out of the network during the deployment. Out of 97 nodes running for three weeks generating 1 message per 10 minutes, we received only 5874 messages, which amounts to 2%. It remains to be seen if this data is rich enough to answer the agricultural concerns put forward in Section 2. Regarding the networking side of the deployment we retrieved enough RSSI data to run some localization experiments, and we got a (negative) answer as far as robustness of software is concerned.

5. Lessons learned

Looking back at the pilot deployment, which did not produce the quantitative data that we were looking for, we asked the question "why was it so hard to get the project going?". After all, we were using hardware (Mica2 clones) and software (TinyOS) that is considered standard in the research community, so we expected it to work "out of the box". This was not an unreasonable assumption since other long-running applications using a (very) similar platform have been reported about [5, 10]. Our main lesson has been that gluing (software) components together takes a lot of effort and requires in-depth knowledge of the individual components. Of course, our lack of experience with real-world deployments did not help either, but when we now factor in our steps on "the learning curve" we estimate that we would need more resources (time, money, people) to get a working deployment than we naively planned for last year. To save others from making the same mistakes as we did, we list our most valuable experiences below.

Apply software engineering principles: The fact that the complexity of software does not scale linearly with the number of lines of code, hence the number of people involved, demands a way of working that is much more restrictive than can be afforded in a single man's project. This "rigor and formality" guideline is the first of Ghezzi's software engineering principles [2]. We naively assumed that we could do without, and paid the price for it.

APIs are not enough: As always, the devil is in the details. We discovered, as many have done before, that code reuse is not as easy as advocated because of many subtle, semantic issues. Quite often only the Application Programmer's Interface (API) of a module is specified, but not the (un)intended usage and invariants that need to be obeyed. For example, MintRoute expects the MAC layer to operate in promiscuous mode so it can do link estimation, but T-MAC implements overhearing avoidance to save energy and shuts down the radio during traffic between neighbors. This mismatch was resolved, but at the expense of valuable time missing at the end of the project. We found that most TinyOS components include some implicit expectations and features that do not port easily to new configurations. This needs to be addressed to allow for rapid application development.

Design for the worst: When planning the deployment we made a number of reasonable looking assumptions about node mobility (none), failure rates (1 node per week), packet error rates (below 10%), etc. These assumptions proved to be incorrect and occasionally caused the system to collapse. In hind sight we should have made much more pessimistic assumptions, selected/coded components to be prepared for the worst to reduce the risk of failure, and avoid the costs associated with modifying components at a late stage of the development cycle.

Test, test, test: The cascading effect of the problems we encountered, combined with the time pressure by Mother Nature, forced us to go out with a system that was barely tested. We managed to create a test system with around 10 nodes, running for several days. We even stress tested that system by increasing the data rate with a factor of 10 to mimic a 100-node deployment. That test passed, but when going live we soon discovered that $10 \times 10 \neq 100$. For example, the issue with inconsistent neighbor lists did not show-up until

the real deployment. An important lesson for us is to plan more time for testing, including the development of a testbed that allows for large-scale in-house experimentation (lots of nodes, long running times, etc.).

24/7 monitoring: Debugging wireless embedded systems is very hard, especially when the network stack fails to operate correctly. With our waterproof casings it was impossible to obtain a core dump of a (stalled) system to find out what state the software was in. We advocate the manifold use of statistics. Each software component should be capable of dumping its status, and provide statistics about its internal operation in the recent past. This requires a major change to the current practice within the TinyOS community, but we really felt the need for additional information when trying to diagnose the behavior of the LOFAR-agro system.

6. Conclusions and future work

The main result from the LOFAR-agro deployment, which involved 100+ sensor nodes monitoring the humidity and temperature in a potato field, is the experience obtained during the one-year pilot. The large number of unfortunate events, ranging from silly programming bugs to subtle semantic mismatches between components, emphasizes the proper use of software engineering principles, the need for worst-case design, and the necessity for large-scale testing. We also recommend the augmentation of existing software modules with the capabilities to provide a rich set of statistics, allowing for detailed 24/7 monitoring of the sensor network.

In the second year of the LOFAR-agro project we plan to take our own lessons to practice. For example, we have started the construction of a 25+ node testbed in our computer-science lab, and plan to run extensive robustness tests (with controlled node failures, etc.). We also plan to engage outdoor tests to ensure that we are fully prepared. So, Murphy, we will be back!

7. Acknowledgments

We like to thank all members from the LOFAR-agro team, in particular Bart van Tuijl and Daan Goense with whom we spent numerous hours watching potatoes grow in sun, wind, and rain while desperately waiting for the network to come to life and start reporting temperature and humidity data. We thank Gertjan Halkes and Tom Parker for all their tuning work on the T-MAC protocol, and Winelis Kavelaars for his support in getting the TNOdes to work.

References

[1] T. v. Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *1st ACM Conf. on Embedded Networked Sensor Systems*, pages 171–180, Los Angeles, CA, Nov. 2003.

[2] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of software engineering*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 2003.

[3] J. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *2nd ACM Conf. on Embedded Networked Sensor Systems*, pages 81–94, Nov. 2004.

[4] K. Langendoen and G. Halkes. Energy-efficient medium access control. In R. Zurawski, editor, *Embedded Systems Handbook*, pages 34.1 – 34.29. CRC press, 2005.

[5] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *First ACM Int. Workshop on Wireless Sensor Networks and Application (WSNA)*, pages 88–97, Atlanta, GA, Sept. 2002.

[6] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *2nd ACM Conf. on Embedded Networked Sensor Systems*, pages 95–107, Baltimore, MD, Nov. 2004.

[7] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. Unifying link abstraction for wireless sensor networks. In *3rd ACM Conf. on Embedded Networked Sensor Systems*, San Diego, CA, Nov. 2005.

[8] T. Schmid, H. Dubois-Ferrire, and M. Vetterli. SensorScope: Experiences with a wireless building monitoring sensor network. In *Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*, pages 13–17, Stockholm, Sweden, June 2005.

[9] Crossbow Stargate. http://www.xbow.com/Products/productsdetails.aspx?sid=85.

[10] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a sensor network expedition. In *1st European Workshop on Sensor Networks (EWSN)*, pages 307–322, Berlin, Germany, Jan. 2004.

[11] J. Thelen, D. Goense, and K. Langendoen. Radio wave propagation in potato fields. In *First workshop on Wireless Network Measurements (co-located with WiOpt 2005)*, Riva del Garda, Italy, Apr. 2005.

[12] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *1st ACM Conf. on Embedded Networked Sensor Systems*, pages 14–27, Los Angeles, CA, Nov. 2003.

[13] M. Yarvis, W. Conner, L. Krishnamurthy, J. Chhabra, B. Elliott, and A. Mainwaring. Real-world experiences with an interactive ad hoc sensor network. In *Int. Conf. on Parallel Processing Workshops (ICPPW'02)*, pages 143–151, Washington, DC, USA, Aug. 2002.