Wireless Sensor Networks for Environmental Monitoring: The SensorScope Experience

Guillermo Barrenetxea, François Ingelrest, Gunnar Schaefer, and Martin Vetterli
LCAV, I&C School, EPFL, Switzerland
Email: firstname.lastname@epfl.ch

***Abstract*—While wireless sensor networks have been extensively studied in the past few years, most results are of theoretical nature and were obtained outside of a practical context. This can be problematic for real applications, especially in the area of environmental monitoring where many factors, such as harsh weather conditions, can greatly influence the performance of such a network, while reliable delivery and high-quality measurements are required. SensorScope is an interdisciplinary project, elaborated by environmental and networking researchers, that aims at narrowing the gap between theory and practice. Several successful real-world deployments have already been undertaken in rugged environments. In this paper, we analyze the particular requirements of environmental monitoring and how these requirements have been met in the SensorScope project. We also present an application example of a deployment, undertaken in a harsh mountain environment.**

I. Introduction

Sensor nodes are small embedded devices which are mainly able to perform simple computations and to send/receive data. Their typical usage is to gather information about their environment via sensors, to potentially pre-process these data, and to finally transmit them. An autonomous set of such nodes is called a *wireless sensor network* (WSN) [1]. Because of cost and energy constraints, only one node is generally able to transmit data from the sensor network to the "outside world" by means of a longer-range connection (*e.g.*, GPRS). This node is called a *sink* since it acts as such with regards to the data stream coming from the network.

Although sensor networks have many applications, environmental monitoring is a domain in which they may have a huge impact. Recent climate change-related catastrophes have illustrated how important a detailed understanding of our environment and its evolution is for human beings. The capacity of researchers to improve this knowledge is mainly limited by current data collection techniques, which are based on very expensive stations (€ 60 000—$ 86 000—for a high-precision station) with limited embedded data loggers. Wireless sensor networks are an alternative solution, well-fitted to these problems. The SensorScope project[1] is a collaborative effort of environmental and computer science researchers to build a WSN-based measurement system, with the ability to immediately transmit gathered data to a distant server. As a result, it allows for real-time (*e.g.*, pollution) as well as long-term (*e.g.*, ice melting) monitoring of natural events in potentially large areas.

[1]http://sensorscope.epfl.ch/

In this paper, we focus on the particularities of environmental monitoring through our experience with SensorScope. Various aspects of WSNs have indeed been studied over the past few years (*e.g.*, synchronization, energy efficiency), but most of these studies are of theoretical nature and may not apply, depending of the considered application. Environmental monitoring, in particular, is very demanding due to harsh outdoor conditions that may greatly impact hardware performance. In SensorScope, we have been faced with many challenges, and we describe here, how we coped with them. As a case in point, we have already been able to deploy several networks, some of them in very harsh conditions, and we present results from such a deployment, which took place on a high mountain pass in Switzerland.

In the next section, we describe the requirements of environmental monitoring and their consequences for WSN design. Section III describes how these requirements are addressed in SensorScope, while in Section IV we detail a particular deployment in a mountain environment. We conclude in Section V.

II. Environmental Monitoring

Although some aspects in a wireless sensor networks may be generic, it is important to carefully consider the specific requirements of the application, especially when it is as demanding as environmental monitoring. Such campaigns generally consist of deploying a number of sensors in the field to periodically measure meteorological and hydrological parameters, such as wind speed and direction. Most of them change relatively slowly in time, which allows for sparse sampling (one sample every two to five minutes is most often sufficient). However, as interesting phenomena, such as rock slides or avalanches, occur seldom and are difficult to predict, deployments must last long enough to capture them.

We can translate the needed characteristics of an environmental monitoring system into the following technical requirements:

- **Autonomy**. Batteries must be able to power the weather stations during the whole deployment. Because the radio transceiver is a massive energy consumer, the network has to be energy-wise, even if a renewable source of energy is used (*e.g.*, solar power). Protocols that require the radio to be always on must be discarded.
- **Reliability**. The network has to perform simple and predictable operations, to prevent unexpected crashes.

Human maintenance should be avoided, first, because end users may not have networking knowledge, second, because areas of interest are most often remote. Achieving reliability is difficult because packet losses are more likely to happen during harsh weather conditions (*e.g.*, heavy rain, intense cold) which are at the same time the most interesting episodes for data analysis.

- **Robustness**. The network must account for a lot of problems such as poor radio connectivity (*e.g.*, in case of snow fall) or hardware failures. For instance, humidity can frequently cause short-circuits leading to unexpected reboots of stations. The use of any protocol requiring an initialization phase to be performed synchronously by all nodes is thus inconceivable.
- **Flexibility**. One must be able to quickly add, move, or remove stations at any time depending on the needs of the applications. For instance, it may turn out that the current location of the stations is not correct to gather the required data, or that new stations should be added at new points of interest. Nodes thus have to automatically detect their network neighborhood to account for such changes, and one cannot rely on *a priori* knowledge when designing the network.

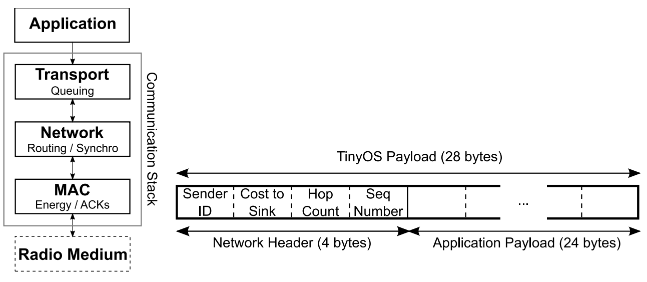
All these requirements are especially important when deploying a network in remote and difficult-to-access places. For instance, one of the SensorScope deployments occurred in high mountain, in collaboration with authorities. A helicopter was required for carrying hardware and people. Going back to the site a few days later because a battery is depleted or because a station needs to be manually rebooted is obviously inconceivable.

One way to achieve all of this is to *keep things simple*. TASK [2] is a set of WSN software and tools, designed at Berkeley, that has been used for outdoor deployments, for instance during the *Macroscope* experiments [3]. The experience of TASK's authors is of great value, and they claim that simple and application-specific approaches provide the most robust solutions for real-world usage. This is especially true for environmental monitoring, because gluing existing—and complex—components together takes a lot of time and effort for in-depth understanding of their interaction. It is sometimes, however, not worth this effort because the targeted application may not require that many features. By keeping things simple, it is possible to create a robust network, well-fitted for the intended application and outdoor usage.

III. The SensorScope Showcase

SensorScope stations are composed of an aluminium skeleton equipped with a solar panel, seven external sensors, and an hermetic box, enclosing electronic parts. We chose a Shockfish TinyNode[2], which is composed of a Texas Instruments MSP430 16-bit microcontroller, running at 8 MHz, and a Semtech XE1205 radio transceiver, operating in the 868 MHz

[2]http://www.tinynode.com/



(a) The stack. (b) Packet format.

Fig. 1: The SensorScope communication stack.

band with a transmission rate of 76 Kbps. It has 48 KB ROM and 10 KB RAM.

To meet our requirements in wireless networking, we have designed and implemented a complete communication stack for TinyOS 2.x. Our code is available under an open-source license on our website[3]. In the following, we describe the networking architecture used in SensorScope, and the ground rules that we have followed during its design.

A. Overall Architecture

Our communication stack, illustrated in Fig. 1a, closely follows the OSI model. It needs to store only four bytes per packet in the standard TinyOS payload, leaving 24 bytes for the application out of the 28 available, as illustrated in Fig. 1b. We chose to not modify the TinyOS network headers to be independent of the underlying radio drivers. There are four different layers:

- **Application**. This layer is responsible for collecting data that must be sent to the sink, such as environmental measurements and battery levels.
- **Transport**. This layer creates and receives packets, and if needed queues them, based on their type: data packets (*e.g.*, measurements) are routed toward the sink, while control ones (*e.g.*, ACKs) are sent to a particular neighbor. With overall traffic being low, there is no need for a congestion control mechanism at this time. Two fields of the header are filled: the number of hops performed by the packet and the sequence number.
- **Network**. This layer gives packets to the MAC layer after having chosen a next hop in the case of data packets. It fills the fields containing the sender identifier and the cost of the route to the sink. All routing decisions are made at this level, and implementing a new protocol only requires a new network layer. Our custom network layer is detailed below.
- **MAC**. In order to reduce energy consumption, the radio is turned off as much as possible. If a packet must be sent while the radio is off, it is kept and sent later on. Only data packets are acknowledged by sending a small packet to the previous hop. As the radio driver lacks a carrier sense, a simple back-off mechanism is used to minimize collisions.

[3]http://sensorscope.epfl.ch/network_code/

B. Networking Features

We describe here the prominent features of our communication stack, which make the system suitable for environmental monitoring. In the following, *broadcast* designates a local broadcast (*i.e.*, a packet sent to all neighbors); the distance designates the *hop-distance* to the sink, not the Euclidean one.

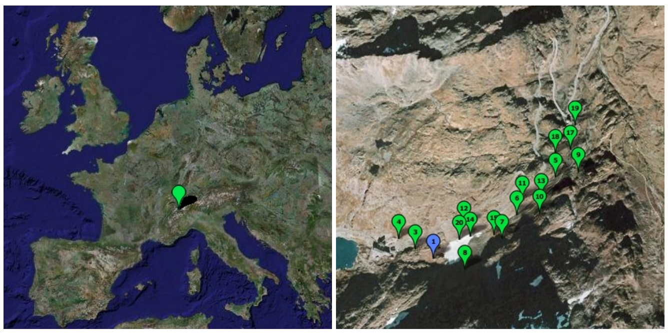
1) Synchronization: Nodes need to time-stamp their messages to allow for meaningful data analysis. Many synchronization protocols exist [4], [5], [6], and most of them contain complex algorithms to achieve a high level of accuracy. For instance, FTSP [6] compensates time drift by means of linear regression, resulting in microsecond-accurate synchronization. While this may be of theoretical interest, not all applications require such a level of accuracy. Time-stamping data messages actually only requires a precision of a few milliseconds. Moreover, the time drift compensation is specific to the chosen hardware and may not be as efficient when deployed outdoors. Temperature, for example, impacts the actual drift, and it is nearly impossible to account for such variations. From our experience, it is better to live with a drift rather than assuming there is none because a high-accuracy solution is used.

In SensorScope, `SYNC_REQUEST`/`SYNC_REPLY` messages are used to propagate the local time of the sink, chosen as the reference time. When a station wants to update its clock, it chooses a neighbor *closer than itself* to the sink, and sends it a request. This neighbor then broadcasts back the reply to its own neighbors, all of which update their clock if they are *further away* from the sink. By doing so, the reference time is propagated from the sink, even if its own clock drifts. The sink also regularly sends its time to the server which can compute the offset between this time and the absolute one and use it to convert the timestamps of incoming messages. This mechanism is simple, robust, and does not involve a huge overhead, while providing sufficient accuracy.

2) Power Management: Stations are equipped with a solar panel to allow for long-term outdoor deployments. However, the radio transceiver is a big energy consumer, and can lead to a negative energy balance. For instance, the TinyNode's power consumption increases by a factor of eight when the radio is on for receiving messages. As a result, the communication stack must absolutely turn off the radio as much as possible to provide for the stations' required autonomy.

Similarly to TASK [2], we use a synchronized duty-cycling method in SensorScope: all nodes in the network turn their radio on at the same time for a short period, and all messages are exchanged during this period. Since a synchronization mechanism between stations is already in place, it is quite easy to apply this energy-saving method. As there may be a slight time drift, nodes wait a bit before sending messages, after turning on their radio, to ensure that their neighbors are indeed "awake". This, of course, leads to a minor waste of energy, but it also keeps the whole mechanism simple and robust. It suffices for a positive energy balance.

3) Routing: To improve robustness, we have chosen not to use a routing backbone toward the sink. A backbone may in fact cause load problems when too many nodes are attached to the same next hop, as it may happen when nodes are always connected to their best parent, as done in MintRoute [7]. Moreover, maintaining a backbone involves a communication overhead due to control messages, for instance, to detect broken links.



(a) Global View. (b) Local View.

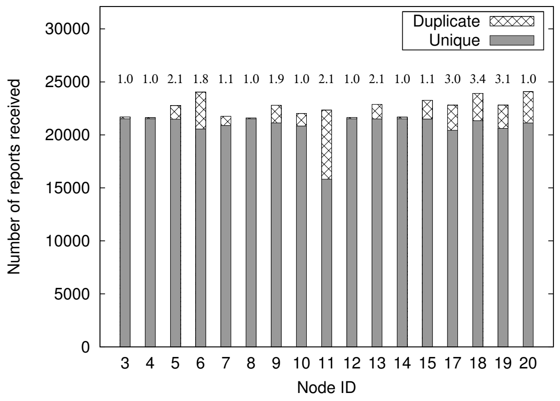
Fig. 2: The map of the Grand Saint Bernard deployment.

To avoid such problems, our stations always choose their next hop at random, greatly increasing the diversity of routes and thus the robustness. To ensure that messages always arrive at the sink, the set of next hops is composed only of neighbors closer to the sink. Since the choice is randomly performed at each step, load balancing is automatically optimal. To favor higher-quality neighbors, two thresholds are defined: when choosing a next hop, a good neighbor (in terms of QoS) is chosen, and if there is none, then a lower-quality one is used. Note that the QoS is simply computed by counting missing sequence numbers of the packets sent by neighbors. Packet losses result in missing sequence numbers, thus decreasing the QoS of the corresponding neighbors.

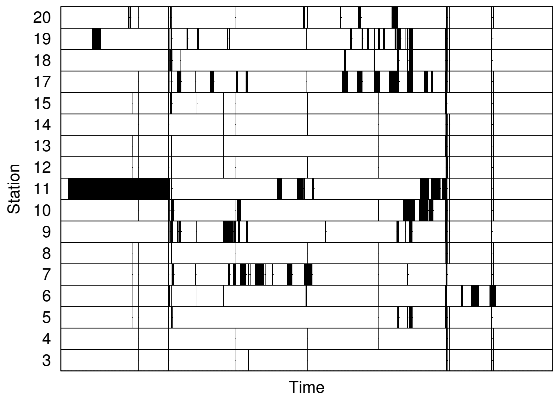
IV. Deployment Example

In 2007, we have performed six outdoor deployments, from EPFL's campus to the high Alps. For lack of space, we herein focus on one of the most important deployments which took place at the Grand Saint Bernard pass. The pass is 2 400 m (7 875 ft) high, located between Switzerland and Italy. Swiss authorities in charge of risk management asked us to deploy a network there, because they wanted to get an accurate hydrological model of the site, which could not be obtained with traditional measurement methods. The use of SensorScope allowed them to obtain spatially dense measures, and the resulting model will help in preventing avalanches and accidental deaths. This deployment began in September 2007 and lasted for one month and a half.

The 17 stations were deployed on a 900 m (2 950 ft) long line as illustrated in Fig. 2. Their location was carefully chosen with the help of our project partners, who are specialized in hydrology, in order to retrieve meaningful measurements for the model to be elaborated. The sink, located on the bottom left of the map (station 1), was equipped with a GPRS module. All system parameters used are given in Table I.



(a) Received data packets and distance of stations to the sink.



(b) Time correlation of losses per station.

Fig. 3: Sensing data statistics for a whole month of operation during the Grand Saint Bernard deployment.

Fig. 3 provides statistics on data gathering during one full month of operation, beginning on the 26th of September 2007. Above the bars, the distance of the respective stations to the sink is given. These numbers show that—the site being quite extensive—most nodes had to use multi-hopping to report their data, with routes up to 3.4 hops on average for station 18. Overall, results are satisfactory, given the harsh weather conditions on the pass.

The missing packets were mostly due to hardware failures, such as short circuits, leading to the loss of several consecutive packets. This is apparent in Fig. 3b, which shows the time correlation of packet losses per station, each black line representing a missing packet. For instance, station 11 suffered a severe short circuit, requiring on-site repair. Subsequent failures were less pronounced, and the corresponding stations were able to recover after some time. This figure also shows problems caused by the GPRS module, resulting in simultaneous losses for almost all stations. We have since determined that this was caused by a bug in the GPRS software, leading to the removal of all enqueued packets upon disconnection from the cellular network.

V. Conclusion and Future Work

Through the various deployments, we have been able to thoroughly test our communication stack in real outdoor conditions, and it has proven to be robust and well-fitted for environmental monitoring. Most of the problems we had to face were caused by hardware malfunctions and are all being addressed. One of the lessons learned is that remote management of the network is of utmost importance, and we are currently lacking such mechanism. For this reason, we are studying the possibility of incorporating Deluge [8] into our software, in order to be able to remotely reprogram the network in the event a bug is detected.

Layer	**Parameter**	**Value**
Application	Sampling	120 s
Network	Poor-quality neighbor	$\geq 70\,\%$
	Good-quality neighbor	$\geq 90\,\%$
	Synchronization frequency	1 h ($\pm$ 72 ms drift)
MAC	Sleep state length	108 s
	Active state length	12 s
	Transmission power	15 dBm

TABLE I: System parameters used during the deployment.

Acknowledgments

This work was partially financed by the Swiss National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), the European Commission under the Framework 6 IST Project "Wirelessly Accessible Sensor Populations (WASP)", and Microsoft Research.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] P. Buonadonna, D. Gay, J. Hellerstein, W. Hong, and S. Madden, "TASK: Sensor network in a box," in *Proceedings of the IEEE European Workshop on Wireless Sensor Networks and Applications (EWSN)*, Jan. 2005.

[3] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," in *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2005.

[4] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *Proceedings of the Symposium on Operating Systems Design and Implementation (OSDI)*, Dec. 2002.

[5] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[6] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.

[7] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2003.

[8] J. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at a scale," in *Proceedings of the ACM International Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2004.