

# SensorScope: Experiences with a Wireless Building Monitoring Sensor Network \*

Thomas Schmid

Henri Dubois-Ferrière

Martin Vetterli†

Ecole Polytechnique Fédérale de Lausanne (EPFL)  
School of Computer and Communication Sciences  
CH-1015 Lausanne, Switzerland

*t.schmid@ieee.org, {henri.dubois-ferriere, martin.vetterli}@epfl.ch*

## ABSTRACT

This paper reports on our experience with the implementation, deployment, and operation of SensorScope, an indoor environmental monitoring network. Nodes run on standard TinyOS components and use B-MAC for the MAC layer implementation. The main component on the server side is a Java application that stores sensor data in a database and can send broadcast commands to the nodes.

SensorScope has now been running continuously for 6 months. The paper presents an analysis of three 2 week periods and compares them in terms of parameter settings, and their impact on data delivery and routing tree depth stability. From the data gathered, we show that network performance is greatly improved by using MAC layer retransmissions, that SensorScope is running in a non congested regime, and we find an expected mote lifetime of 61 days.

The phenomena discussed in this paper are well known. The contribution of this paper is an insight to a long running sensor network that is more realistic than a testbed with a wired back-channel, but more controllable than a long-term, remote experiment.

## 1. INTRODUCTION

Environmental monitoring is considered as one of the prime application fields for sensor networks today [3]. Examples include monitoring of natural habitats [4] [7], volcanic activity [8], or building structures [9]. While the number of deployed sensor networks is steadily rising, sensor networking technology is still in its infancy, and long-lived, large-scale sensor network deployments remain a challenge.

There is an inherent trade-off between realism and observability in experimental evaluation of sensor networks (Figure 1). At one extreme, simulations offer complete control and visibility into experiments, but they cannot faithfully reproduce all the parameters that affect a live system. At the other extreme, absolute realism comes with full deployments, often in remote locations, such as Great Duck Island [7]. Unfortunately deploying such a system requires signif-

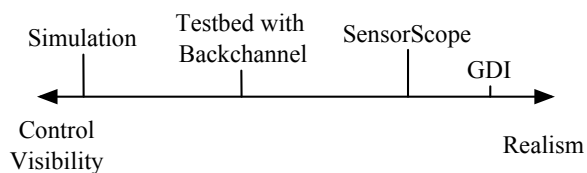


Figure 1: Trade-off between in-network visibility and realism. Long-term, remote experiments such as Great Duck Island are the most realistic. Short-term experiments on a testbed with a wired back-channel and power supply add a large degree of realism compared to simulations since they expose the system to the vagaries of real radio channels. SensorScope represents an intermediate point in the trade-off spectrum.

icant resources; furthermore the constraint of a remote deployment means that code updates must be extremely conservative (in order to reduce the risk of system crashes), and the amount of monitoring data that nodes can report is typically kept very low in order to maximize node lifetime.

This paper reports on our experience with the implementation, deployment, and operation of SensorScope, an indoor monitoring sensor network. The network consists of 20 mica2 and mica2dot motes, equipped with a variety of sensors for light, temperature and sound. Motes use a multi-hop routing tree to report sensor readings and network monitoring information back to the base-station.

SensorScope represents an intermediate trade-off between realism and visibility. Unlike a powered testbed, it is a long-running (since October 2004) and dedicated deployment. Nodes are powered with batteries, even though we could have used a continuous power supply, so as to expose the network to the vagaries of node brownouts and blackouts. Unlike with a remote deployment, we can still reboot and debug motes, allowing us to be less conservative in making software updates to the network. Nodes can also send more monitoring information since battery lifetime is not as critical as in a remote system.

A second aim of SensorScope is to consider a full end-to-end system going all the way from the sensors to a user-visible front end. All information coming out of the network, such as routing tree information (Figure 2) and sensor data (Figure 3), is stored in a database, and made accessible via

\*This work was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

†Also with Dept. of EECS, UC Berkeley

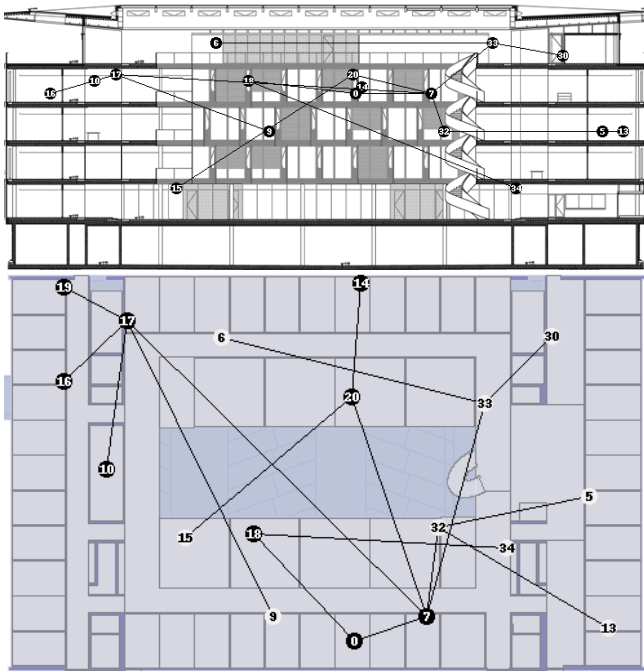


Figure 2: Network routing graph on March 16, 2005. The top image is a longitudinal section, the bottom one a floor plan of the building where SensorScope is installed. The different colors of the motes on the bottom image represent motes on the same floor (black) and motes on an other floor (white) than the floor plan shows.

a public web interface<sup>1</sup>. Through this website we provide to the research community a full data set (both historical archives and current data) containing both application data (sensor readings) and network monitoring data.

The rest of the paper is organized as follows: Section 2 describes the SensorScope system. Then, Section 3 discusses network performance and channel utilization. Finally, Section 4 gives some concluding remarks and an outlook for future work on SensorScope.

## 2. SYSTEM DESCRIPTION

### 2.1 Mote-Side

Nodes run a TinyOS application that was designed to be representative of simple, small-scale environmental monitoring networks. The application has two basic duties: to periodically sample sensors and route readings back to the base-station, and to interpret and disseminate command broadcasts. As a routing substrate, we use the standard (`tos/lib/Route`, later `tos/lib/MintRoute`) multi-hop routing implementation that is part of the TinyOS distribution. Protocol constants are left to their default values. Moving down in the stack, we use the B-MAC [6] MAC layer implementation and its low-power listening scheme. We also integrated the Deluge [5] network programming system into our deployment, and have since used it several times to make code updates. Reprogramming our 20-node network takes approximately 30 minutes and is reliable (though in a few instances one node was not updated and had to be manually reprogrammed).

<sup>1</sup><http://sensornscope.epfl.ch>

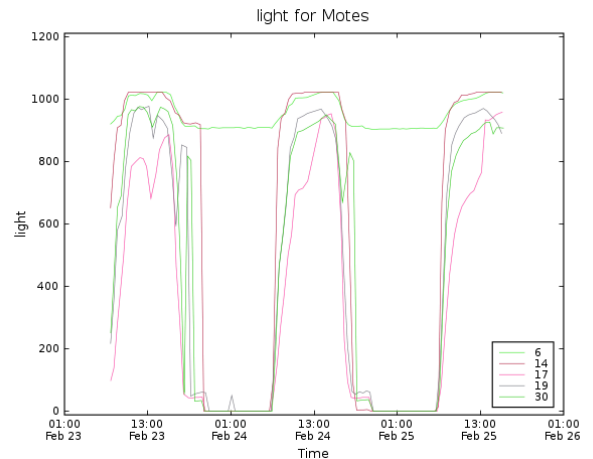


Figure 3: Sample output graph from the web interface. It shows the light sensor reading for 5 motes from Feb. 23 to Feb. 25, 2005. Note how at 21:00, the light at EPFL is automatically turned off and all readings, except one mote that is near a window with a nearby street light, are 0.

Our only significant departure from the standard TinyOS network stack is the addition of a multi-hop hybrid ARQ (MHARQ) layer between the network and link layers. With MHARQ, nodes buffer corrupt packets upon reception, and when two corrupt versions of a packet have been received, a decoding procedure attempts to recover the original packet from the corrupt copies. Unlike traditional forward error correction (FEC), MHARQ does not transmit redundant overhead on good links; and unlike adaptive coding techniques, it does not require costly channel probes to estimate the amount of redundancy required to achieve reliable communications. MHARQ also exploits the multi-node nature of a sensor network by enhancing multi-node interactions (multi-hop routing, multicast, or flooding) in a way that standard point-to-point FEC cannot, in addition to enhancing single-hop communications. This layer is also responsible for managing link-layer retransmissions. A detailed description of the MHARQ scheme is beyond the scope of this paper; for further details we refer to [1].

The application, besides periodically turning on and sampling sensors, is also responsible for parsing incoming broadcast messages and reforwarding them. Broadcast messages originate from the base-station, and either carry *query requests* or *configuration commands*. A total of 25 broadcast message types are currently defined. Queries are sent to retrieve different node configuration parameters as well as various networking-related monitoring information such as routing tables, neighbor tables, and network activity counters. Commands are sent to specify configuration parameters that may be applications-related (sensor sampling rate, which sensors to sample) or network related (max. number of retransmissions, turn on/off MHARQ, set low-power listening status, etc). A simple storage module saves configuration updates into persistent flash memory.

### 2.2 Server-Side

The server side of SensorScope builds upon several free software packages and libraries and glues them together as one system. A middle ware programmed in Java, makes the link

Setting	Oct.	Nov.	Dec.
Low Power Listening	4	2	4
RF Power (dBm)	-17	-14	-14
Sampling Rate (s)	120	120	120
Routing Data Rate (min)	5	5	5
Neighbor Table Rate (min)	60	60	15
Retransmission	0	0	5
MHARQ	no	no	yes
Deluge	no	no	yes

**Table 1: Program parameters for the three data sets.** *Low Power Listening*: mode in which the mote was. *RF Power*: the radio chips power setting. *Sampling Rate*: rate at which motes sent their sensor data. *Routing Data Rate*: rate at which routing data was sent to the base station. *Neighbor Table Rate*: rate at which neighbor tables were sent to the base station. *Retransmission*: how many times a mote tried to retransmit a packet if it was not successful. *MHARQ*: if multi-hop hybrid ARQ was enabled or not. *Deluge*: if Deluge was implemented (programming over the air)

from the sensor network to the database. This database stores sensor data, routing and neighbor tables as well as maintenance information. Additionally, it provides possibilities to send query requests and configuration commands to the motes.

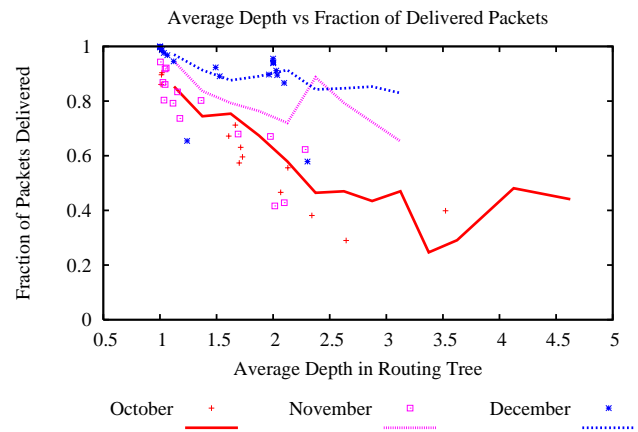
The user interface is either a command line tool or a web interface programmed in PHP that uses Python CGI scripts to access the database and to generate sensor data graphs (see Figure 3 for an example). The sensor data can also be exported to Matlab files for a statistical analysis and signal processing purposes. Furthermore, the web interface visualizes routing trees and connectivity graphs (see Figure 2 for an example). Additionally, it offers a graphical interface to send queries and commands to the motes to set, for example, their radio power, sampling rate or which sensors to sample. The web interface has also facilities to help observe the network’s status such that the developer can intervene if motes unexpectedly die, and is connected to a SMS messaging gateway via which it delivers a message when nodes become unresponsive.

### 3. PERFORMANCE

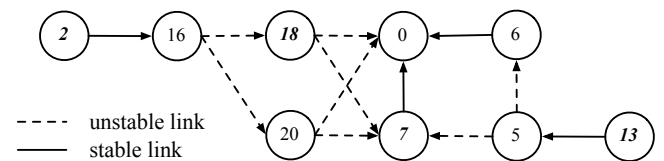
This section investigates the performance of the sensor network for three different datasets. Each dataset consists of two weeks of continuous sensor and routing table data. During these two weeks the motes were not moved and the code was not altered. Batteries were only changed, when a mote reached a voltage reading below a certain threshold. The three datasets will henceforth be called according to the month in which the data was collected: October, November, and December. See Table 1 for the configuration parameters of each set.

#### 3.1 Network Performance

Figure 4 plots the average number of packets delivered vs. the motes depth in the routing tree. Each data point represents one mote’s average number of delivered packets and average depth in the routing tree for the whole dataset. The lines correspond to the average over one hour intervals of routing tree depth and packets delivered for all motes.



**Figure 4: Average routing tree depth vs. packets delivered for each mote and dataset.** Note that the data is very noisy. Therefore, the average does not decay smoothly with increasing routing tree depth. For more details, see Section 3.1.

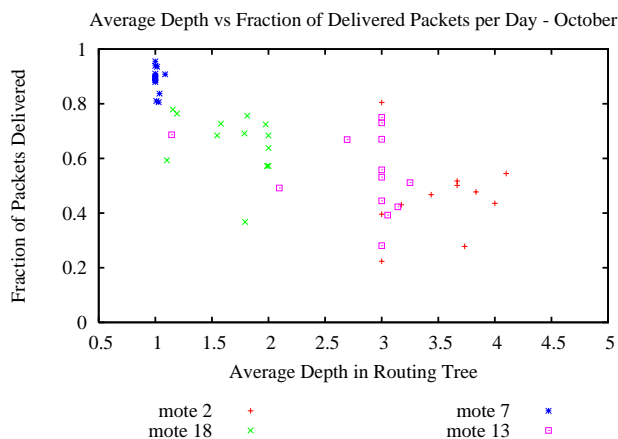


**Figure 5: Network graph showing how motes can have a stable or unstable routing tree depth.** See Section 3.1 for more details.

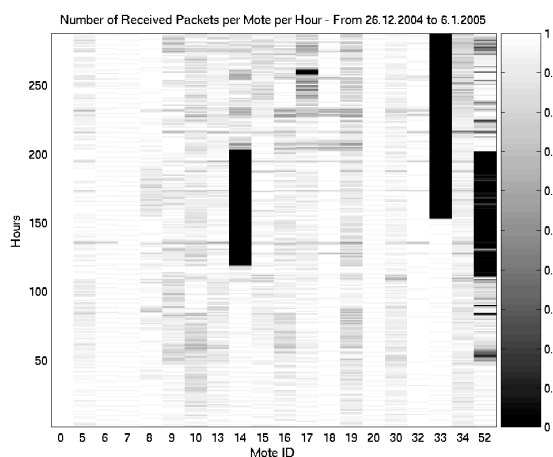
We can see that the difference in overall number of packets delivered per mote between the October and November dataset is small. The increased average fraction of packets delivered can be explained by the fact that in November the motes in the network were set to a higher RF Power, and therefore, the network had less motes with a routing tree depth greater than 2. In December, the network performed better. The major difference is the enabled MAC layer retransmission, i.e. motes now tried up to 5 times to deliver a packet to the next hop if they did not receive an ACK.

One can distinguish between motes with a stable routing tree depth, and motes with frequently changing depth. Figure 6 shows 4 different motes from the October dataset that illustrates this further. Mote 7 and 13 have a stable routing tree depth of 1 and 3, whereas mote 2 and 18 have a routing tree depth oscillating between [1, 2] and [3, 4] respectively. In the October dataset, one finds that 46% of the motes have a stable depth and 54% do not. This behavior can be explained by a motes parent link stability and who it can choose as parent. Figure 5 depicts the network topology for the 4 motes shown in Figure 6. The topology was reconstructed from the received parent information. Mote 0 was the sink. Mote 7 was near the sink, and therefore had a stable one hop link. Mote 13’s parent was most of the time mote 5 which in turn had either mote 6 or 7 as its parent. Both of them are motes with a stable one hop link to the sink. On the other side, mote 2’s parent was mote 16 which had as parent either mote 18 or 20. But both of them were motes with an unstable link to the sink and therefore chose sometimes an additional hop to deliver their messages.

Figure 7 illustrates the individual number of packets deliv-



**Figure 6: Average routing tree depth vs. packets delivered for 4 motes in the October dataset. Each point represents the average over one day. Note how mote 2 and 18 have an average depth between [1,2] and [3,4] whereas mote 7 and 13’s depth is stable at 1 and 3 respectively.**

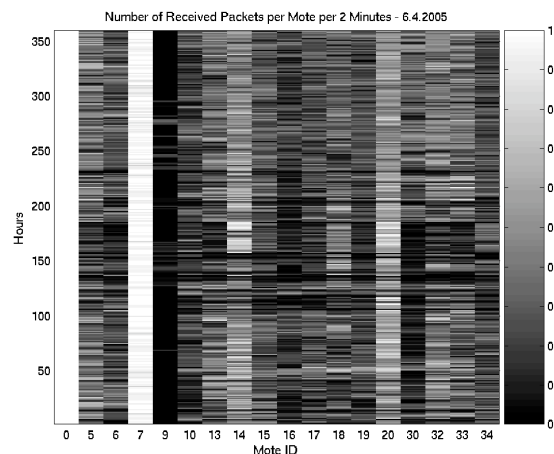


**Figure 7: Fraction of packets delivered for each mote per hour for the December dataset. Each small bar represents the fraction of received sensor data packets at the sink (mote ID 0) per hour.**

ered for each mote for the December dataset. One can see the difference in delivered packets between motes that are always one hop away from the sink (6, 7, 18, 20, 32, 33) and the other ones. The two black bars for mote 14 and 32 are due to battery failure. Node 52 had a persistently weak path to the sink.

### 3.2 Channel Utilization

SensorScope lies somewhere between wired testbed and remote deployments in terms of realism versus observability. In particular, nodes report network monitoring information with more detail and at higher rate than would be possible in a remote network where the difficulty of changing batteries requires keeping radio utilization at a strict minimum. This increased monitoring traffic could potentially cause congestion, and thus completely change network dynamics with respect to the conditions expected in a low-rate, low channel utilization deployment. It is therefore necessary to verify that even with the additional traffic, the network remains



**Figure 8: Fraction of packets delivered for each mote per 2 minute for a congested network. Sampling rate was set to 6.5 seconds and Low Power Listening to 4. See Section 3.2 for more details.**

ID	November		December	
	packets/min	byte/min	packets/min	byte/min
5	7.2	2139.8	7.6	4048.3
14	9.0	2647.5	8.8	4711.7
16	5.2	1527.5	3.8	2039.4
20	12.1	3563.4	10.5	5619.5

**Table 2: Total number of bytes transmitted for 4 motes. The data is the average for the two days 15th, 16th November and 28th, 29th December 2004**

clearly in a non-congested regime.

As part of the reported network statistics, nodes send the total number of bytes transmitted over the air. We show these in Table 2 for a subset of 4 motes, for the November and December datasets. Since packet lengths are variable, the number of packets transmitted can only be approximately inferred; this approximation is however close to the expected number of packets transmitted according to the constants of Table 1. Note that there are large differences in transmission workload for different motes. This comes from the fact that the relay load varies depending on a mote’s position in the routing tree. For example mote 20 frequently has two or more children in the routing tree, whereas mote 16 is nearly always a leaf node.

Under the conservative assumption that all nodes are within interference range of each other, we have a maximum network throughput equal to the radio rate of 19.2 kbps, or 144’000 bytes/min. Assuming an average transfer rate per mote of 2000 byte/min for November and 4000 byte/min for December, we would have a total network throughput of approximately 40’000 byte/min for November and 80’000 byte/min for the December dataset. This appears sufficient to establish that the network is not operating in a persistently congested regime.

To see how the network performs in a congested scenario, we ran a 24 hour experiment with a high sample rate of 6.5 seconds. All the other parameters were set to the same values as in the December dataset (Table 1). From this it follows that the estimated per node throughput without forwarding

messages is 7'150 byte/min. Under the assumption of 18 motes within interference range of each other, this amounts to a channel load of approximately 128'700 byte/min, which is lower than the theoretically possible 144'000 bytes/min.

Over the 24 hour period, only mote 7 performed well and delivered 96% of all the packets. Next was mote 20 with 54% and mote 14 with 48%. All the other motes have delivered less than or equal to 35% of all their packets. An explanation for these differences can be found in their distance and channel quality to the sink since mote 7 and 20 are closest to mote 0. All other motes are further away. Depicted in Figure 8 are the individual number of delivered packets for each mote for this dataset.

Congestion has also an effect on the transmitted bytes for each mote. Mote 20, which had most of the time mote 7 as its parent, transmitted on average 28'000 byte/min and mote 7 32'700 byte/min. In contrast, mote 14 transmitted 12'700 byte/min and mote 16 12'100 byte/min.

If we increase the congestion of the network by setting the sampling rate to 5 seconds, the delivery rate drops even further. Mote 7 still delivers 97% of its packets, mote 20 delivers 42% and mote 14 only 28%. All the others are below 20%. With our assumptions, the network would need at least 160'200 byte/min to deliver all packets. Mote 20 transmits on average 10'300 byte/min and mote 7 transmits 21'600 byte/min. In contrast, mote 14 transmits 33'500 byte/min and mote 16 37'800 byte/min, although both have a much smaller number of delivered packets. This means that collisions required them to retransmit a lot of messages before they were either sent to the next hop or dropped from the send queue.

### 3.3 Mote Lifetime

We analyzed mote lifetimes over the December dataset. As mentioned previously, the motes run at a Low Power Listening level of 4. This corresponds to a duty cycle of 6.1%, i.e., a maximum of 4.64 packets/sec. All our mica2dot motes are equipped with two AA Alkaline batteries, each with 1.5 Volt. If we assume that the voltage decreases linearly with the capacity of the batteries<sup>2</sup>, we find an average power consumption of 13.1 mV/day (sample variance  $s^2 = 11.0$ ) over the two weeks. This corresponds to an average mote lifetime of 61 days (assuming that motes die if they reach 2.2V). Further analysis of the dependence between power consumption and number of messages sent, including forwarded messages, shows that there is a significant positive correlation between these values.

## 4. CONCLUSION

We have presented the system architecture of SensorScope and gave an overview of the network performance for three time periods. We also showed that the network operates in a non-congested regime and what the performance is if it is congested. In total, we had 560'000 sensor data values, 94'000 routing informations and 192'000 neighbor table entries. The difficulty in analyzing this data lies in the fact that it is unreliably delivered, and as such we do not have fixed, periodic snapshots of network state. This is especially

critical since the most interesting aspect of the performance is not when the network is delivering more than 90% of the data, but when there are problems, for example with interference or congestion.

Future plans for SensorScope include refining the user interface and to improve the maintenance tools. Another goal is, not surprisingly, to extend battery lifetimes. This would allow us to gather sensor data and network statistics over a longer period of time without interruptions. Finally, we intend to make the collected data more easily accessible to other scientists.

## 5. REFERENCES

- [1] H. Dubois-Ferriere, D. Estrin, and M. Vetterli. A multi-hop hybrid arq layer for sensor networks. *Under submission*.
- [2] Energizer. Energizer en91 datasheet. <http://data.energizer.com/PDFs/EN91.pdf>.
- [3] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Salt Lake City, Utah, May 2001.
- [4] W. Hu, V. N. Tran, N. Bulusu, C. tung Chou, S. Jha, and A. Taylor. The design and evaluation of a hybrid sensor network for cane-toad monitoring. In *Proceedings of Information Processing in Sensor Networks (IPSN 2005/SPOTS 2005)*, Los Angeles, April 2005.
- [5] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94. ACM Press, 2004.
- [6] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of ACM Sensys*, Los Angeles, USA, April 2003.
- [7] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, November 2004.
- [8] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *2nd European Workshop on Wireless Sensor Networks*, Istanbul, January 2005.
- [9] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, November 2004.

<sup>2</sup>[2] shows that the linear assumption is good for voltages between 1.5 and 1.1 Volt.