

Design Tools for Sensor-Based Science

Randal Burns, Andreas Terzis
Computer Science Department
Johns Hopkins University
{randal,terzis}@cs.jhu.edu

Michael Franklin
Computer Science Department
University of California, Berkeley
franklin@cs.berkeley.edu

Abstract

We describe the architecture of software tools that aid scientists in designing and programming wireless sensor networks (WSNs). The goals of our system are (1) to reduce the complexity of deployment by automatically programming a sensor network and database based on a visual description of a site and experiment and (2) to provide a simple, intuitive network design tool that explores trade-offs in cost, reliability, and network lifetime. We motivate the need for such tools based on our experience in deploying a sensor network used to study soil ecology. We also present techniques for the development of site-specific transmission models, the placement of motes and gateways, and data processing in hierarchical networks.

1 Introduction

Currently, developing and deploying a science experiment using sensor networks is an onerous task executed with primitive techniques. It requires a great deal of expertise in computer science. For data management, this includes programming motes for data extraction, often using SQL variants for sensor networks, *e.g.* TinyDB [5]. While declarative languages for sensor networks have made this process simpler, it demands sophistication and is a time-consuming task. For network design, the scientist must understand the transmission properties of network nodes to ensure that they can communicate and that redundant paths exist so that the network survives failures.

Furthermore, many environments of interest are fragile on intrusion and hostile to sensor hardware – a bad combination. Hostile environments result in unpredictable performance and frequent failures [9]. This, in turn, leads to frequent and damaging site incursions: trial-and-error configuration and repair of the network. Additionally, high-variance in performance factors, such as radio range, makes it difficult to over-provision a network reliably and at reasonable cost.

From this description, we conclude that domain scientists today lack the intuitive, high-level tools necessary to design and deploy field experiments that utilize sensor networks. To address this need, we are developing network and data design tools that simplify dramatically the deployment of a sensor network and the collection and compilation of data. These tools economically provision the topology of a hierarchical sensor network, program the motes, manage data flows from motes, through gateways, to a back-end database, analyze, clean and calibrate data, and provide Web-services interfaces to the data.

Using these tools, scientists will be able to purchase

motes and a database server and, using an electronic map and specific knowledge of the site of the experiment, construct an end-to-end data collection and analysis application with no programming—in the traditional sense. Furthermore, no manual tuning of the placement and characteristics (*e.g.* transmission range and frequency) of network nodes is necessary. These tools serve as a “design wizard,” hiding complexity and obviating programming effort. Also, they achieve a solution rapidly in both time and number of iterations, which reduces costs and minimizes impact on fragile sites.

2 A Sensor Network Experiment

Recent experience deploying a sensor network at Johns Hopkins University highlights the poor state of tools for designing and conducting experiments using sensor networks. The goal was to build an experimental platform to study soil ecology at a meso-scale in collaboration with Dr. Szlavecz from the Earth and Planetary Sciences Department at JHU [4]. We deployed a small grid – 10 motes total – with sensors every 3 meters. The grid covers the hill of a woodland site with three layers of foliage and intermittent surface water, *i.e.* around heavy rainfall. The site is semi-urban; it is located near a campus building and roads. Thus, it is subject to interference from buildings and other man-made structures (*e.g.* fences), as well as RF noise from nearby WiFi networks.

The development and deployment were labor intensive and presented major hurdles in network design, calibration, and programming. All told, eight team members contributed to this installation, including four faculty members in three disciplines – Physics, Computer Science, and Earth and Planetary Sciences. The combined effort, including hardware and software design as well as management of the deployed network, took more than 400 person hours, including 80 faculty-person hours.

Through this process, we identified two major hurdles that must be overcome. (1) Programming sensor networks is complex owing to the network’s distributed nature and the heterogeneity of components. We spent the majority of our efforts programming the collection of motes and processing the data at multiple levels. This includes the sampling discipline, the coordinated transmission of collected samples to a base station, the injection of data into a database, calibration routines, and Web-services interfaces for remote access. (2) The vagaries of the wireless medium severely hampered the data collection process. Figure 1 represents the average link quality for all wireless links in our network for the duration of our experiment. It is evident from this graph

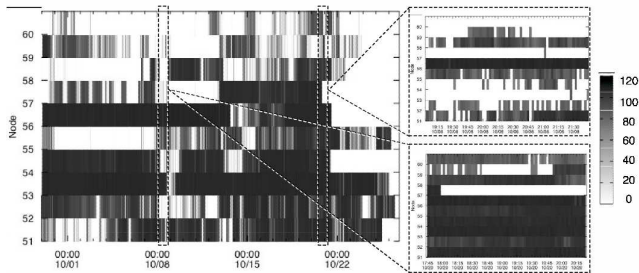


Figure 1: Average Link Quality Indicators (LQI) for the wireless links between the base station and ten motes (51-60) over one month. Darker links have higher quality and lower loss rate.

that link quality is highly variable both spatially and temporally, resulting in observed loss rates up to 60%. To counter these negative effects, we had to manually measure transmission properties, adjust mote locations to improve signal reception, and fine tune a custom reliable transfer protocol to minimize power consumption. Similar experiences were reported for other sensor networks used for environmental monitoring [9, 10].

Based on the collective experience from previous deployments, we conclude that a high-level *experiment design toolkit* is needed critically. While individual system components have the necessary capabilities, the challenges lie in integrating and automating these capabilities to present a unified, simple view of the whole system to scientists.

3 Network Design

Unpredictability and variance in performance complicate the deployment of a functioning sensor network. The MicaZ motes we use have a rated transmission range of 100m and they easily achieve that in a nearby parking lot [7]. However, four out of ten motes in our grid could barely communicate with a base station 10m away and 5m up. The “out-of-service” motes did not follow a trivial pattern, *e.g.* farthest away or greatest angle to base station. Rather, they were sensitive to refraction, scattering, and reflections from foliage and other site obstacles as well as interference from nearby WiFi networks. Furthermore, reception quality changed daily and seasonally.

We conclude that for network design, one must capture the baseline transmission properties of the site through a custom signal propagation model. This model estimates connectivity among network nodes and bounds this connectivity variation over time. As a network is deployed, performance measurements refine the propagation model. We posit that this information allows the construction of practical propagation models that accurately describe the actual performance through instrumentation. We use the propagation model to position additional *relay points* (*i.e.* network devices that act solely as data forwarders) wherever predicted network quality is low.

Evolving models may be used for subsequent installations as well as for network tuning and repair. This leads to a deployment strategy in which a small network may be deployed to “learn” a site prior to building a large, longer-lived, and more expensive deployment. This strategy will result in a more accurate and inexpensive configuration, *i.e.* a mini-

mally over-provisioned network. It also realizes a network solution quickly, by not placing hardware until transmission models have been learned. Finally, this approach minimizes site degradation associated with the trial and error topology configuration process, because the limited deployment is less invasive.

Cartographic Interface The derivation of the propagation model through analysis and network measurements is hidden from the scientists designing the sensor network. Instead, their interface to experimental design is a map, overlaid with site-specific signal propagation properties. Figure 2 presents such a map of the authors’ experimental site. The grayed regions of the map describe the estimated signal attenuation for radio signals from motes placed on the ground. The tool derives information for the site-specific, propagation model from many sources. In this case, a model generated using collected measurements from the existing network at Patch A provides an accurate and more detailed model, when compared with new deployments at Patches B and C. For Patches B and C and other regions, we derive estimates of signal attenuation based on site morphology (*e.g.* density of foliage and elevation difference among different motes). Finally, the scientist may annotate the map with additional knowledge. In our case, we select the region occupied by the Olin Hall building and mark it as a hard barrier to radio transmissions.

Using a sensor tool from the toolbar shown in Fig. 2, the scientist selects locations at which to collect data. Having selected locations, she requests the tool to construct the best network topology (§5.1), subject to constraints, such as the number of additional gateways and relays or minimum availability at each mote. The tool places relays and gateways and draws a routing graph, which it annotates with link characteristics such as average link quality.

By manipulating data objects on the map, the scientist specifies the data to be collected and the analysis against that data. A menu (Patch B), associated with each mote on the map, shows the capabilities of that mote. In this menu, one selects the quantities to be measured and the sampling discipline. Sampled data may be transferred to the back-end database or they may be manipulated, filtered, and analyzed within the network. The scientist builds expressions against that data to compute aggregates or correlations. For example, in Patch C we compute a regional mean temperature across the patch. Because temperature has small variation over these scales, it may be computed within the network to save bandwidth (§5.2).

4 Data Management

Based on the scientists’ inputs, our design tools develop a data management application that runs across motes, gateways, and a back-end database (Figure 3). For data handling, we divide the architecture into two major software components: (1) **Declarative Mid-Tier Processing** based on the High Fan-In system (HiFi) [1] and (2) **Data Services**, which stores and analyzes data and makes it available to users and applications.

The Mid-Tier processing layer virtualizes the hierarchical network of motes and gateways, presenting a stable view of in-network data to the database. The Mid-Tier layer populates tables within a database. The Data Services layer manages the database of collected data, presenting the informa-

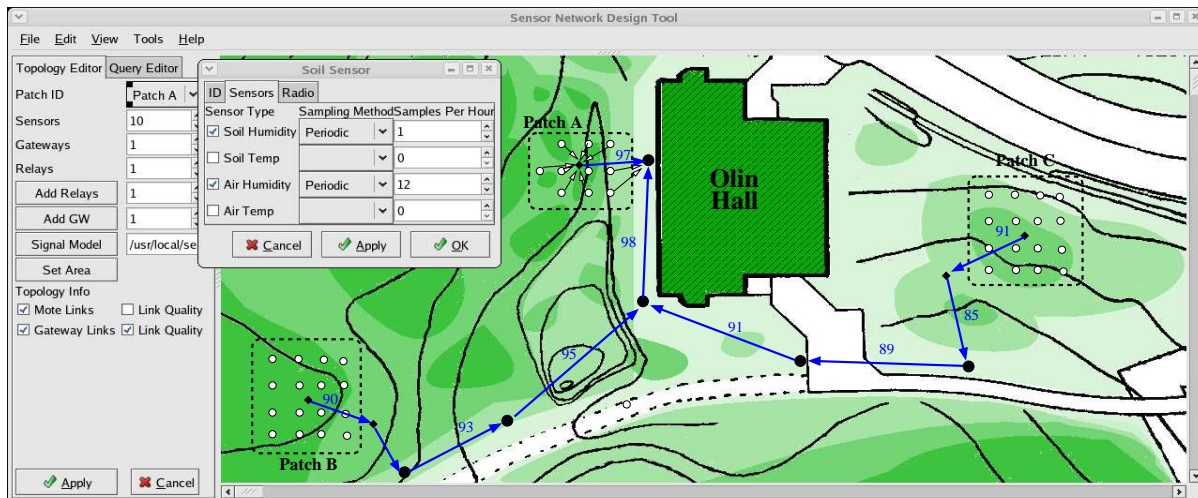


Figure 2: Design tool interfaces demonstrated on the authors' soil monitoring field site.

tion as requested by the scientist. It also performs analyses, often in the form of OLAP datacubes that permit multi-dimensional data to be correlated and compared. Once analyzed and stored, data are made available to users and Web applications through several interfaces (including the emerging SensorML standard [2]).

While similar functions may be performed in both the Mid-Tier and Data Services layers, the archival nature of the database differentiates their roles. The Data Services layer preserves all data in the database. An aggregate computed in the database, *e.g.* a mean value over a sensor patch, preserves the original data. Whereas the same aggregate computed in the Mid-Tier layer, reduces the sampled data to the mean value, discarding the component inputs. The benefit of computing in the Mid-Tier lies in data reduction, which leads to power savings, increased network lifetime, reduced storage, and bandwidth conservation. While scientists often wish to retain “all the data,” this might be impossible due to sensor network limitations [6].

The data design process configures and programs both of these systems so that they act in concert. Users describe the data to be gathered and computations against that data using sensor management and query-building tools. We divide each user query into two sub-queries: one in the Declarative Mid-Tier system that converts inputs into the data to be stored at the database and one that analyzes or transforms the data in the database. For example, in our soil ecology experiment, we track the light intensity and soil moisture at each mote along with the mean soil temperature at each patch and the ambient air temperature. Mid-Tier processing samples the data, computes the aggregate temperature at each patch, integrates external climatological data (for ambient temperature), and populates the database with the results. The Data Services layer stores and analyzes the data and provides users access to both raw (collected) and analyzed (processed) data. The proposed tools make very few assumptions about the code that runs on the devices of the deployed network. Specifically, we assume that mote-class devices can deliver raw measurements over multi-hop wireless paths to gateway-class devices, which run the HiFi queries.

5 Techniques

In this section, we sketch some of the specific technologies that support the design tools. These include the iterative refinement of network models, operator placement for in-network processing, and sensor virtualization.

5.1 Network Topology Design

The network design process builds a model based on information about the deployment site and the properties of sensors and gateways. Site information includes the dimensions of the site, the site type, *e.g.* indoor vs. outdoor, and the location and characteristics of obstacles in the deployment area. The network owner inputs this information using a graphical interface, using a floor plan if the deployment is within a building or a topographical map for outdoor deployments. Furthermore, the user annotates obstacles on the map of the deployment site based on a library of different obstacle categories included in the network design tool.

Based on these inputs, the design tool estimates the quality of a sensor network and proposes the location of additional relay points and sensor gateways to improve this quality. In this context, we define quality as the percentage of sensor measurements retrieved by the back-end servers. In calculating this metric, we estimate the underlying loss rate of all network links in order to determine the percentage of measurements delivered to the root of the “best” (*i.e.* least lossy) tree. If the calculated percentage of retrieved measurements lies below a user-defined threshold, the tool determines the location of additional relay points that maximize quality improvement.

As Figure 4 illustrates, the first stage of the network design tool is the Modeler. This stage generates *site-specific* signal propagation models that are variants of previously proposed models [8], customized for the site and the hardware used. These models estimate the power of the signal received at each network node when one of the other nodes transmits at a given frequency and power. The goal of the Physical-layer simulator (PHY for short) is to calculate the bit error rate for every transmitter-receiver pair in the network. The Optimizer determines the quality of the sensor network as originally laid out on the deployment map, based

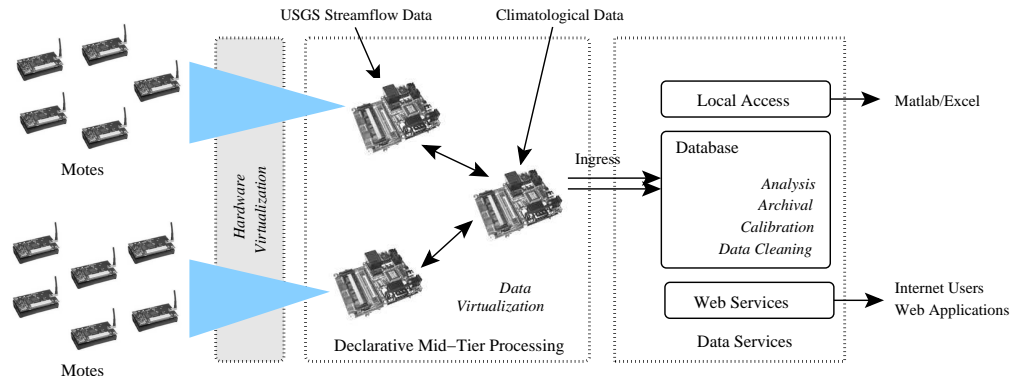


Figure 3: Inputs and Elements of the Design Tools.

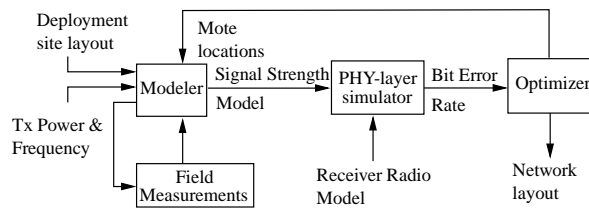


Figure 4: Block diagram of the Network Design tool.

on the link qualities predicted by the PHY-layer simulator. If the network quality is below a threshold, the Optimizer suggests the location of additional relay points. Furthermore, the Optimizer computes the locations of sensor gateways within each “sensor patch” (cf. Figure 2) to minimize power consumption and maximize the percentage of measurements delivered to the gateways.

As noted previously, the outcome of this process acts only as a *first level approximation* of the production network performance and guides a small, initial deployment. Measurements collected from that pilot deployment further improve the accuracy of the network model.

5.2 Declarative Mid-Tier Processing

A key aspect of our architecture is the declarative mid-tier level that sits between the sensor devices and the Data Services layer. This level consists of data stream processing software that runs on gateway devices. By declarative, we mean that data manipulation is specified using a high-level, set-based language similar to query languages developed for relational database systems. We adapt it to processing over continuous streams of sensor readings. The advantages of the declarative approach include: ease of initial deployment, ability to incorporate a wide range of data sources and devices, automatic and dynamic optimization of data processing operators, and integration with external data streams and repositories.

Our approach to building this middle tier is to leverage the HiFi sensor data processing system [1]. HiFi was originally targeted at large-scale, widely-distributed enterprise applications, such as RFID and sensor-enabled supply chains. Instead, we focus on the self-managing and ease-of-use aspects of the system for highly error-prone environmental sensing systems.

Operator Placement and Optimization: Dynamic approaches for placing application functionality within the network present a fundamental challenge. This includes identifying and exploiting commonality among concurrent tasks running within the system.

A key consideration is the placement of queries and data across the nodes of the system. Given a query with a set of operators, the query planner determines where in the hierarchy to place each operator. This decision attempts to reduce overall system bandwidth usage by pushing operators down the hierarchy. Some data streams (or static relations) may not be visible at lower levels of the hierarchy. For example, processing that requires the correlation of outputs from multiple devices can not be placed on any one of those devices, but must be placed at a node (typically a gateway node) that can read the streams from all of the devices. Thus, the query planner tends to push operators to the lowest level at which the streams and relations over which they operate are visible.

Furthermore, when adding a new query to a running sensor network, the query planner considers data flows and queries that are executing already in order to exploit shared processing. For instance, if multiple operators from different queries process the same underlying data stream, then it may be advantageous to pull the operators up. Alternatively, it may be possible to improve the visibility of some queries by pushing external streams or static data down toward the edge of the network. Downward dataflow incurs initial bandwidth costs and complexity due to replication, but may improve parallelism and resource utilization, and can provide overall bandwidth savings. Caching can also improve performance, but this raises additional issues, because query and data placement in a cache-based system are inherently inter-dependent [3].

Virtual Device Interface: The need to seamlessly integrate the physical world with the digital world presents the most unique challenge in data-intensive sensor deployments. Real world data comprise an infinite collection of unbounded continuous streams with inherent ambiguities and inconsistencies, whereas the digital world is inherently discrete with strict semantics. Furthermore, data collection techniques are imperfect at best. Physical receptor devices introduce complexity, owing to a wide variance in interface, behavior, and reliability. Thus, sensor-based systems must bridge these disparate worlds in a manner that enables users to both trust and make sense of the data provided by the system.

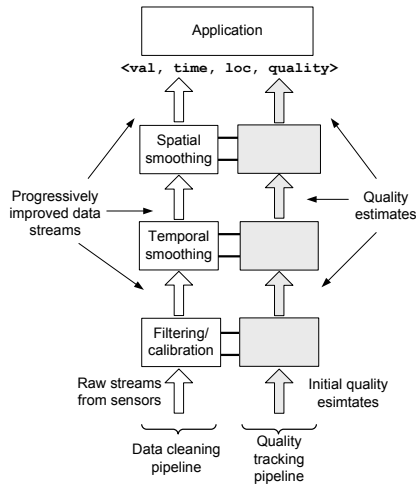


Figure 5: A sensor data cleaning pipeline and its parallel quality assessment pipeline.

Toward this end, we provide a virtual device interface to encapsulate points of interaction with the physical world. A virtual device comprises groups of sensors, processing and fusing their streams to produce more useful, higher-quality data. It does so by incorporating data-cleaning techniques, conversion and calibration, virtualization, lineage tracking, and quality assessment. A virtual device combines declarative and non-declarative processing in a pipelined fashion.

Figure 5 gives an example of such a virtual device with two parallel pipelines: the main sensor stream processing pipeline on the left and a parallel, “shadow” pipeline on the right. The shadow pipeline monitors the data streams as they are processed, and calculates estimates of the quality of the data. The outputs of these pipelines are merged to produce readings of the form $\langle \text{val}, \text{time}, \text{location}, \text{quality} \rangle$, which are consumed by the Data Services layer. This example demonstrates three stages in the data cleaning pipeline and the three corresponding stages in the shadow quality pipeline. The lowest stage is Filtering and Calibration, which applies typically to individual readings from the sensors, one-at-a-time. Above this basic processing level, we place different types of *smoothing* stages, used to boost the quality of the readings.

Virtual devices provide support for notions of answer quality. The virtual device augments raw sensor data with error estimates and confidence intervals, using the shadow pipeline (Figure 5). For example, a virtual device for a sensor network can use model-based techniques for determining data quality based on probability distributions and/or sampling according to the type of processing being done at each pipeline stage. These error estimates are preserved and adjusted as the readings are processed through the cleaning pipeline. Applications use these estimates for thresholding, to provide overall quality estimates in experiments, or to monitor the effectiveness of the sensor deployment over time.

6 Concluding Remarks

Research and development in WSN infrastructure – hardware, networking, operating systems, and middleware – have made sensor networks a powerful tool for science,

changing the scale, resolution, and cost at which data may be collected. We have had just this experience with our field deployment for soil ecology. Rather than collecting samples manually every month, our network samples temperature, light, and soil moisture data every minute, allowing us to observe transient phenomena and episodic, punctuated events.

However, the scientific community lacks the management tools to make sensors networks widely useful. Current deployment techniques require manual tuning of the network and low-level programming, and thus the expertise to perform these tasks.

Our efforts in developing design tools aim to remove these barriers by providing high-level, intuitive tools with which a scientist can manage the lifetime of an experiment. This includes application development, deployment, tuning, monitoring, and repair.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments that helped improve the quality of this paper.

Andreas Terzis is partially supported by NSF CAREER grant CNS-0546648 and research funds from Microsoft Research and the Gordon Bell Foundation. Franklin’s work was funded in part by NSF under ITR grants IIS-0086057 and SI-0122599, and by research funds from Intel and the UC MICRO program.

References

- [1] M. Franklin, S. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, E. Wu, O. Cooper, A. Edakkunni, and W. Hong. Design considerations for high fan-in systems: The HiFi approach. In *Proceedings of the Conference on Innovative Data Systems Research*, 2005.
- [2] O. G. C. Inc. Sensor Model Language (SensorML) for In-site and Remote Sensors. Available at: http://vast.nsstc.uah.edu/SensorML/SensorML_04-019_1.0_beta.pdf, Nov. 2004.
- [3] D. Kossmann, M. J. Franklin, G. Drasch, and W. Ag. Cache investment: integrating query optimization and distributed data placement. *ACM TODS*, 25(4):517–558, 2000.
- [4] Life Under Your Feet. Available at <http://www.lifeunderyourfeet.org/>.
- [5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM TODS*, 30(1):122–173, 2005.
- [6] S. Madden and J. Gehrke. Query Processing in Sensor Networks. In *Proceedings of the International Conference on Pervasive Computing*, 2004.
- [7] MICAz Specifications. Available at http://www.xbow.com/Support/Support\pdf_files/MPR-MIB_Series_Users_Manual.pdf.
- [8] T. S. Rappaport. *Wireless Communications: Principles & Practices*. Prentice Hall, 1996.
- [9] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *Proceedings of SenSys 2004*, Nov. 2004.
- [10] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, P. Buonadonna, S. Burgess, D. Gay, W. Hong, T. Dawson, and D. Culler. A Macroscopic in the Redwoods. In *Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov. 2005.