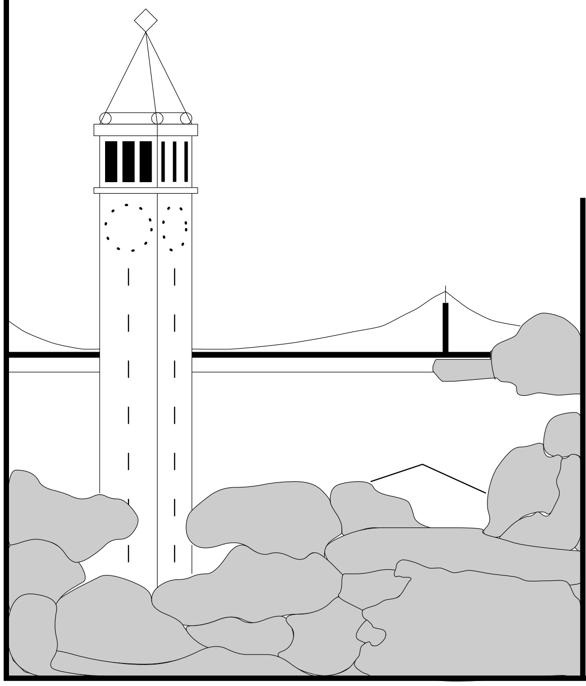
A Pipelined Framework for Online Cleaning of Sensor Data Streams

Shawn R. Jeffery *Gustavo Alonso* *Michael J. Franklin* *Wei Hong*
Jennifer Widom



Report No. UCB/CSD-5-1413

September 2005

Computer Science Division (EECS)
University of California
Berkeley, California 94720

A Pipelined Framework for Online Cleaning of Sensor Data Streams

Shawn R. Jeffery[1,3] Gustavo Alonso[2,1*] Michael J. Franklin[1] Wei Hong[3] Jennifer Widom[4]

[1]UC Berkeley {jeffery, franklin}@cs.berkeley.edu

[2]ETH Zurich alonso@inf.ethz.ch

[3]Intel Research Berkeley whong@intel-research.net

[4]Stanford University widom@cs.stanford.edu

Abstract

Data captured from the physical world through receptor devices such as wireless sensor networks and RFID readers tend to be unreliable and noisy. The data cleaning process for such data is not easily handled by standard data warehouse-oriented techniques, which do not take into account the strong temporal and spatial components of receptor data. Here we present Extensible receptor Stream Processing *(*ESP*), an extensible framework for cleaning the data streams produced by physical receptor devices. ESP is a declarative query processing tool with a pipelined design that is easy to setup and configure for each receptor deployment. We validate the ESP platform through three real-world deployments using ESP to clean receptor data streams.*

1 Introduction

Physical receptor devices such as wireless sensor networks and RFID technologies are enabling new classes of applications that utilize the devices' data streams to gain insight into the physical world. Examples include real-time supply chain management [14] and environmental monitoring [26, 28]. To support these and other emerging applications, *receptor-based systems* are being built and deployed with a focus on providing a shared infrastructure to manage and process the data produced by receptors.

A limitation of current systems is the unreliability of the data produced by physical devices. This "dirty data" manifests itself in two general forms:

- Missed readings: Receptors often employ low cost, low power hardware and wireless communication, which lead to frequently dropped messages. For example, RFID readers often capture only 60-70% of the tags in their vicinity [16, 25]. Similarly, in a one month wireless sensor network deployment at the Intel Research Lab in Berkeley [20] involving 54 motes, each sensor delivered, on average, only 42% of the data it was asked to report.
- Unreliable readings: Often, individual sensor readings are imprecise and/or unreliable. For instance, physical devices tend to "fail dirty": the sensor fails but continues to report faulty values. In a sensor network deployment in Sonoma County, CA [28], for example, 8 out of 33 temperature-sensing motes failed, but continued to report readings that slowly rose to above 100^o Celsius.

Furthermore, these error characteristics vary with the environment in which the receptors are placed. For instance, RFID readers may drop more readings in an environment with metal present [15], and sensor mote readings are affected by the ambient temperature.

Applications attempting to use such raw receptor data directly will be rendered useless. For example, we show in Section 4 that in a simple RFID-enabled shelf scenario, restock alerts (produced when the number of items on a shelf drops below a threshold) would be generated 2.3 times per second if the raw receptor data were used. In reality, no restock alerts should have been generated. Thus, the data must be appropriately cleaned to account for these errors before being used by any application. Of course, existing deployments necessarily deal with these problems to some extent, but they tend to use tedious post-processing and ad-hoc means to clean sensor data, leading to brittle receptor infrastructures and increased deployment costs.

Thus, the challenge is to 1) directly address the receptor data's particular error characteristics (i.e., missed and unreliable readings) and 2) to provide a solution that is easy to deploy and configure.

One possible solution would be to centralize the cleaning in a data warehouse. Such an approach, however, is ill-suited for receptor data. Data cleaning

* This work was done while the author was at UC Berkeley as a Stonebraker Fellow.

in warehouses is usually an offline, centralized, iterative, and sometimes interactive process that focuses on a small set of well-defined tasks [27]. In contrast, receptor-based systems and their associated applications typically are time sensitive, e.g., real-time forecasting and inventory control. This demands that the data be cleaned online before it is streamed to the application, precluding offline or interactive approaches.

More fundamentally, the nature of the errors in receptor data is not easily corrected by traditional data cleaning. Receptor data demands different techniques that address the nature of its errors (i.e., missed and unreliable readings). Receptor data tends to be strongly correlated in both time and space; the readings observed at one time instant are highly indicative of the readings observed at the next time instant, as are readings at nearby devices. We introduce the concepts of *temporal granule* and *spatial granule* to capture these correlations. These granules define the smallest unit of time and space on which an application operates. These fundamental abstractions can be used to interpolate lost readings or remove outliers through temporal and spatial aggregation.

In this paper, we propose an extensible framework for online cleaning of receptor data streams. Our platform, *Extensible receptor Stream Processing* (or *ESP*), consists of a programmable pipeline of stream-based, windowed processing stages designed to operate on-the-fly as the data are streamed through the system. These stages segment the cleaning process into five tasks, each responsible for a different logical aspect of the data:

- *Point*: Tuple-level corrections, transformation, and filters
- *Smooth*: Aggregation within a temporal granule to interpolate for missed readings and remove errant single readings
- *Merge*: Aggregation within a spatial granule to interpolate for missed readings and remove data from outlier devices
- *Arbitrate*: Conflict resolution between different readings from different receptors
- *Virtualize*: Cross-checking using multiple types of receptors and application-level cleaning

These stages are generally applicable across many types of receptors and are easy to program independently, in many cases through declarative queries.

This paper is organized as follows. In Section 2, we place ESP in the context of related work that has addressed issues relating to data cleaning and receptor-based systems. We introduce our pipelined cleaning framework, ESP, in Section 3. We then show the details of ESP processing through three detailed deployments using ESP for data cleaning (Sections 4, 5, and 6), and describe experiments verifying that ESP provides significant improvements in data quality over raw receptor data. Finally, we outline our future work and conclude in Section 7.

2 Related Work

Existing work on cleaning receptor data streams can be grouped into two categories: 1) traditional data cleaning, and 2) receptor data management.

2.1 Traditional Data Cleaning

There is extensive prior work on data cleaning in the context of data warehouses and data integration.

Traditional data cleaning tends to focus on a small set of well-defined tasks, including transformations, matchings, and duplicate elimination [27]. ESP provides a framework to utilize these techniques in an online, streaming manner, extending them to take advantage of the characteristics of physical receptor data.

The AJAX tool [18] proposes an extensible, declarative means of specifying cleaning operations in a data warehouse. This is similar in spirit to ESP in that we also define an easily deployable and configurable architecture that utilizes declarative query processing. AJAX, however, does not consider streaming data or windowed processing.

Potter's wheel [30] is an interactive tool for assisting in data cleaning operations. Its interactive nature restricts its ability to handle the streaming data produced by receptors.

Finally, many commercial efforts (e.g., [7, 19]) have addressed data cleaning issues related to enterprise data integration. These solutions provide tools for cleaning and translating data, but do not address temporal or spatial aspects of physical receptor data.

2.2 Receptor Data Management

A wide range of projects are addressing issues related to receptor data management.

Multiple systems provide mechanisms for interacting with wireless sensor networks ([23, 8]). For example, TinyDB provides a declarative means of acquiring data from a sensor network. The data received from TinyDB, however, must be first cleaned before it can be used by any application.

The work in [13, 24] addresses cleaning and error correction for wireless sensor networks. These approaches are designed only for sensor network data and the authors do not consider cleaning for other types of

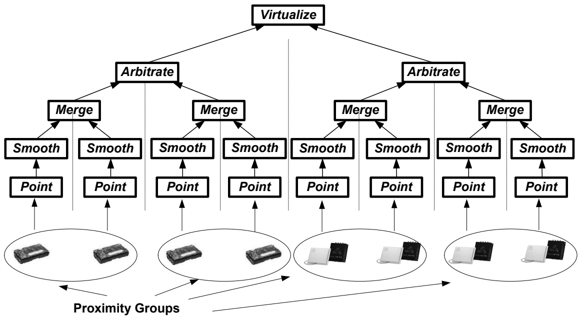


Figure 1: ESP Processing Stages

receptors (e.g., RFID). ESP can incorporate these techniques as part of a general cleaning solution applicable across all types of receptors.

Another important line of work in sensor network data processing is model-driven query processing, such as in the BBQ system [12]. While BBQ is not designed for continuous streaming data, the architecture we propose for ESP provides a platform for utilizing models to assist in data cleaning in a streaming context.

Savant middleware [10] is a configurable set of processing modules for RFID data. Savant recognizes the need for processing stages to convert raw RFID data into application-level data, but it does not deal with receptors beyond RFID technology, nor does it consider unreliable devices.

Finally, ESP is part of the HiFi [17] project. HiFi is a distributed, hierarchical stream processing system designed to support large-scale receptor-based networks, termed "high fan-in" systems. ESP is intended clean receptor streams at the edge of the HiFi network.

3 The Extensible Receptor Stream Processing Framework

In this section, we introduce ESP (Extensible receptor Stream Processing), our pipelined data processing framework for online cleaning of receptor data streams.

While building the initial version of HiFi [11], we confronted many of the issues associated with unreliable data produced by receptor devices. Most notably, we observed that the system was unable to produce meaningful answers when used directly with raw RFID data. Our solution was to use a rudimentary pipeline of ad-hoc queries we termed "CSAVA" [17], designed to run throughout the HiFi hierarchy to convert the RFID data to application data.

Extensible receptor Stream Processing (*ESP*) generalizes and extends the CSAVA pipeline with a focus on cleaning physical device data at the edge of the network. ESP cleans raw physical data by processing multiple physical receptor streams, exploiting the temporal and spatial aspects of receptor data, to produce a single, improved output stream.

Before discussing the ESP processing model, we first define our temporal and spatial abstractions that drive many of ESP's cleaning mechanisms.

3.1 Temporal and Spatial Granules

In general, receptor-based applications are not interested in individual readings in time or individual devices in space, but rather in an application-level concept of *temporal granules* and *spatial granules*. These granules define the lowest-level, atomic unit of both time and space in which an application is interested. Furthermore, a granule is a means for an application to indicate to ESP that the data within the granule should be highly correlated. Thus, ESP uses the granule concept to aggregate, sample, or detect outliers within the granule.

3.1.1 Temporal Granules

Although many receptor devices are capable of producing data continuously (or at a very high sample rate), applications are usually concerned with data from a larger time period, or *temporal granule*. For instance, a sensor network environmental monitoring application that builds models of micro-climates in a redwood tree wants data every 5 minutes as this granule captures the most dynamic variations in micro-climate [29].

To support this notion of temporal granules, ESP uses windowed processing to group readings within a granule. Within a window, readings can be aggregated or compared with each other to detect obvious outliers.

3.1.2 Spatial Granules

Similarly, receptor-based applications are not interested in the data from individual devices, but rather in an application-level notion of a *spatial granule*, such as a shelf in a retail scenario or a room in a digital home application. These spatial granules are the lowest level spatial unit on which an application operates.

To support this application-level view of spatial granules, ESP organizes receptors into *proximity groups*. A proximity group defines a set of receptors of the same type that are monitoring the same spatial granule. For instance, a set of motes monitoring the temperature in the same room may be grouped into the same proximity group, as may two RFID readers monitoring the same warehouse shelf. Readings from devices in the same proximity group can be processed in a similar manner to readings within a temporal granule.

Spatial granules and physical devices can have one-to-many, many-to-one, or many-to-many relationships and may change dynamically. These details are hidden from the application through ESP.

3.2 ESP Cleaning Stages

Having described the fundamental abstractions underlying ESP, we now describe at a high level ESP's processing stages. ESP segments receptor stream processing into a cascade of five programmable stages (shown in Figure 1): *Point* - *Smooth* - *Merge* - *Arbitrate* - *Virtualize*. These stages operate on different logical aspects of the data as they are streamed through the pipeline.

Stage 1, *Point*: The *Point* stage operates over a single value in a receptor stream. The primary purpose of this stage is to filter individual values (e.g., errant RFID tags or obvious outliers) or to convert fields within an individual tuple. In this sense, *Point* is similar to the traditional data cleaning task of transformation [27]. Note the *Point* may also be used to improve performance through early elimination of data.

Stage 2, *Smooth*: In *Smooth*, ESP uses the temporal granule defined by the application to correct for missed readings and detect outliers in a single receptor stream. It does this by processing readings in a sliding window corresponding the size of the temporal granule.

Stage 3, *Merge*: Analogous to the temporal processing in the *Smooth* stage, *Merge* uses the application's spatial granule to correct for missed readings and remove outliers spatially. *Merge* uses aggregation over receptor streams within a proximity group, filling in missed readings and eliminating non-correlated errors in individual devices.

Stage 4, *Arbitrate*: Spatial granules may not map directly to receptor's detection fields, leading to possible conflicts between the readings from different proximity groups that are physically close to one another. The *Arbitrate* stage deals with conflicts, such as duplicate readings, between data streams from different spatial granules. While this is conceptually similar to traditional duplicate elimination, the criteria used to resolve conflicts in this case are different; *Arbitrate* takes into account properties of the receptor devices and the physical world (e.g., tags closer to a reader will be read more often) to de-duplicate readings.

Stage 5, *Virtualize*: Finally, some types of data cleaning utilize application-level logic or readings from across different types of receptors to increase the confidence in the data the system reports. To provide a platform for such techniques, ESP defines a *Virtualize* stage, that combines readings from different types of devices and different proximity groups. For example, a BBQ [12]-type system used for receptor data cleaning may exploit correlations between different sensors (e.g., voltage and temperature) to provide outlier detection.

3.3 ESP Deployment

Deploying a cleaning pipeline using ESP involves implementing one or more of these stages. Stages may be implemented in a variety of ways:

- Declarative continuous queries
- User-defined functions or aggregates
- Arbitrary code

These approaches have increasing levels of functionality, but decreasing levels of flexibility. Not all stages need be implemented and multiple operations may be implemented for one stage. Additionally, the application must define the temporal and spatial granules it is interested in.

An *ESP Processor* initiates data flow from the appropriate receptors and applies each stage in a Fjord-style [22] manner as the sensor readings stream through the pipeline.

Having described ESP processing at a high level, we present in the next three sections detailed processing for each stage and demonstrate the overall effectiveness and ease of configuration of ESP through three deployments of receptor based systems.

4 RFID Data Processing

The first deployment we address using ESP is a retail scenario using RFID technology. Such technology is notoriously error-prone. Tags that exist are frequently missed while tags that are not in a reader's normal view are sometimes read. In a retail scenario, an application continuously monitors the count of items on each shelf using Query 1. In this query, the window clause indicates the temporal granule (5 seconds) and the `GROUP BY` clause denotes the spatial granule (a shelf).

Query 1 *Shelf monitoring query to determine the number of items on each shelf. The temporal granule is 5 seconds and the spatial granule is a shelf.*

```
SELECT shelf, count(distinct tag_id)
FROM rfid_data [Range By '5 sec']
GROUP BY shelf
```

To study ESP used for cleaning RFID data, we ran an experiment emulating a retail scenario. Our experimental setup is depicted in Figure 2. We used two 915 MHz RFID readers from Alien Technology [5], each

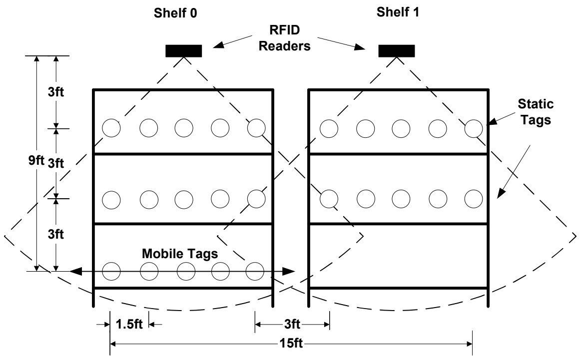


Figure 2: Shelf scenario setup with 2 shelves, each with an RFID reader and 10 tags statically placed within 6 feet of the antenna (5 tags at 3 feet, 5 tags at 6 feet). Additionally, 5 tags were relocated every 40 seconds.

responsible for one shelf and thus each forming a proximity group. The readers' sample period was set at 5Hz (i.e., 5 polls per second). Each shelf was stocked with 10 tagged items using Alien "I2" tags [4], EPC Class 1 RFID tags designed for long-range detection in a controlled environment. Tags were suspended in the same plane as the reader, spaced 1.5 feet apart from each other, and at two distances from the reader, 3 feet and 6 feet. Tags were oriented such that their antennae were directly facing the reader. Note that this setup is overly favorable to RFID technology as it attempts to alleviate many of the known causes of degraded readings [15]. Additionally, to introduce a dynamic component into the experiment, we relocated 5 items placed 9 feet from the reader between the two shelves every 40 seconds.

The metric we use to evaluate our techniques is the average relative error of the results of Query 1, which is defined as follows:

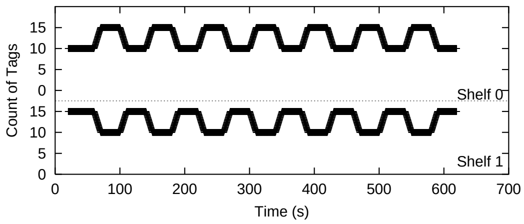
$$\sum_{i=0}^{N} \frac{\frac{|R_i - T_i|}{T_i}}{N} \quad (1)$$

Where:

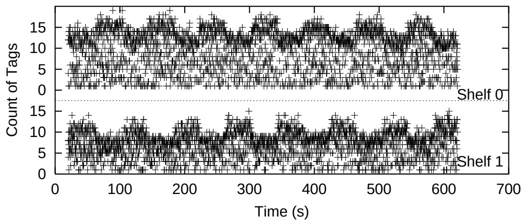
N = total number of time steps
i = time step at the granularity of the reader (5Hz)
R_i = reported count of items on a shelf at time i
T_i = true count of items on a shelf at i

This metric denotes how far off, on average, the reported count of items is from reality. We ran this experiment multiple times; all runs produced similar results.

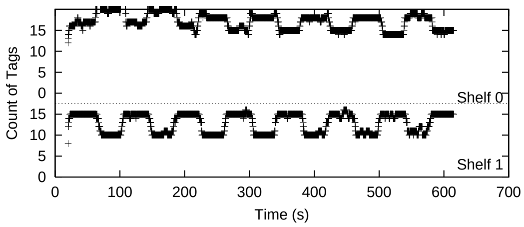
Figure 3(a) depicts the trace of the actual count of items on each shelf over the course of the experiment. If the application were to use the output of the RFID readers directly, the results would be near-meaningless (shown in Figure 3(b)): the average relative error of the output of Query 1 compared to reality for the duration of the experiment was 0.41 (i.e., the count of the number of items on each shelf was off by almost half, on average). If an application wants to be notified when the number of items on a shelf drops below 5, then the query using the raw data would report that a shelf is in need of restocking 2.3 times per second, on average.



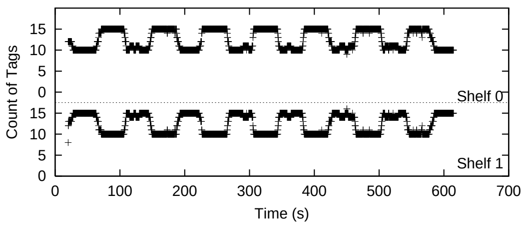
(a) Reality



(b) Query 1 results using raw RFID data



(c) Query 1 results after *Smooth* processing



(d) Query 1 results after *Arbitrate* processing

Figure 3: Query 1 results after different stages of processing

We can use an ESP pipeline to clean this data. Note that the RFID reader already provides *Point* functionality out of the box by removing tags that fail a checksum [1]. We implement the *Smooth* and *Arbitrate* stages[1] for ESP (as shown in Figure 4). As there is only one receptor per proximity group, *Merge* is not needed. We describe each processing stage with the

[1]We demonstrate the other processing stages in the deployments in the following two sections.

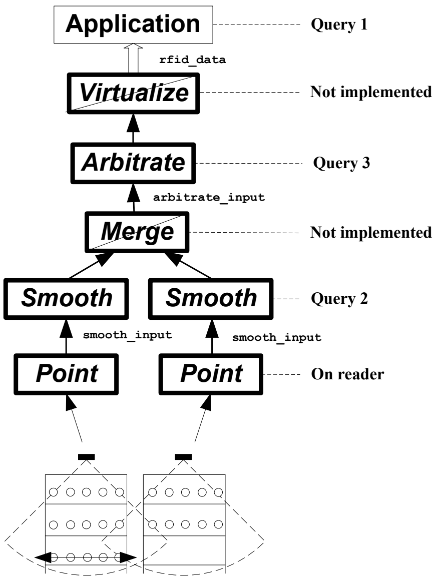


Figure 4: ESP pipeline for cleaning RFID data

declarative query (expressed in CQL [6]) with which it is implemented.

4.1 Stage 2: *Smooth*

At the *Smooth* stage (shown in Query 2), ESP interpolates for lost readings within a temporal granule. This query aggregates readings from a single reader's stream within a sliding window corresponding to the size of the temporal granule. The ESP processor applies this query to each receptor stream (as illustrated in Figure 4).

Query 2 *Interpolating for lost readings in the* Smooth *stage. The input stream,* `smooth_input`*, is provided by the ESP processor*

```
SELECT tag_id, count(*)
FROM smooth_input [Range By '5 sec']
GROUP BY tag_id
```

The results of Query 1 over the data produced by this stage are shown in Figure 3(c). The *Smooth* stage is able to eliminate the constant restocking alerts generated by the query using the raw data.

The count of items per shelf, however, is still inaccurate (an average relative error of 0.24) due to discrepancies in the performance of the readers. As seen in Figure 3(c), the antenna for shelf 0 read more tags than that of shelf 1, despite being of the same model; the counts reported for shelf 0 were consistently 4 to 5 items higher than reality. We tried different configurations of antennae and determined that this difference is likely due to known issues with the antenna ports on RFID readers [2]. Processing in the *Smooth* stage has alleviated the issues with dropped readings, but any application using this data will be misled into thinking that shelf 0 is overstocked.

4.2 Stage 4: *Arbitrate*

The *Arbitrate* stage (shown in Query 3) corrects for the discrepancies between antennae left unresolved by the *Smooth* stage. It does this by comparing, at each time step, the number of times read for tags that were read by multiple spatial granules. It then attributes the tag to the spatial granule that read the item's tag the most.

ESP runs *Arbitrate* over the union of the streams produced by Query 2[2].

Query 3 *Correcting for duplicate readings in the* Arbitrate *stage*

```
SELECT spatial_granule, tag_id
FROM arbitrate_input ai1 [Range By 'NOW']
GROUP BY spatial_granule, tag_id
HAVING count(*) >= ALL(SELECT count(*)
                 FROM arbitrate_input ai2
                 [Range By 'NOW']
                 WHERE ai1.tag_id = ai2.tag_id
                 GROUP BY spatial_granule)
```

The results of running Query 1 over the arbitrated data is shown in Figure 3(d). Observe that ESP is able to correct for the differing performance of the two antennae and to provide a substantially more accurate count of the items on each shelf to the application. After *Arbitrate* processing, the average relative error of Query 1 is 0.04. This equates to an error of being off by less than one item, on average. Thus, ESP provides significant reduction in error over the raw RFID data: compare this with the item counts using the raw data that were off by almost half.

4.2.1 Analysis of the ESP Pipeline

Much of ESP's cleaning capabilities are derived from using all applicable stages in the pipeline. To verify this claim, we analyzed the effectiveness of each stage and the particular ordering. We ran Query 1 over different configurations of the pipeline (*Smooth* only, *Arbitrate* only, *Arbitrate* followed by *Smooth*, and *Smooth* followed by *Arbitrate*) and measured the average relative error of the result. The outcome is shown in Figure 5.

Individually, *Smooth* reduces error compared to the raw data as it interpolates for many of the missing readings in RFID data. As mentioned above, *Smooth* does not, however, correct for the inconsistencies between the two antennae. *Arbitrate* individually, on the other

[2]Note that although the *Merge* stage is unused in this case, ESP automatically adds a `spatial_granule` attribute to each stream, corresponding to each proximity group (i.e., each shelf).

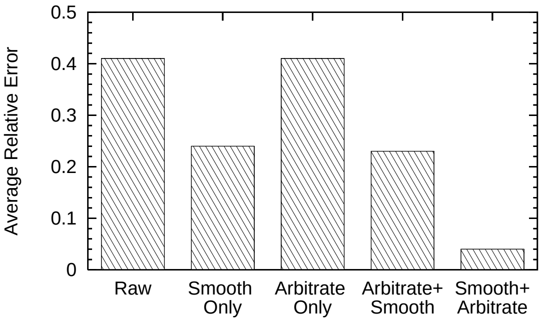


Figure 5: Average relative error for Query 1 using data produced by different configurations of the ESP pipeline

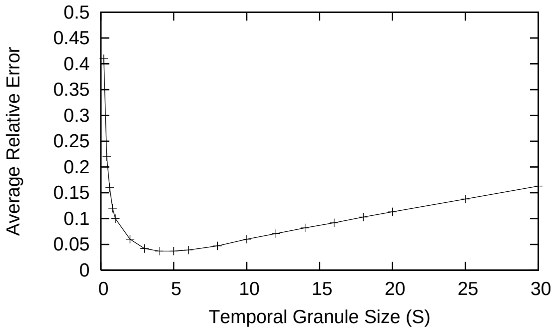


Figure 6: Average relative error for Query 1 over data produced by ESP using different size temporal granules

hand, provides little benefit beyond the raw data. This is due to the fact that *Arbitrate* cannot properly function without the missing readings filled in by *Smooth*. For the same reason, *Arbitrate* followed by *Smooth* provides little benefit beyond *Smooth* alone. Only when both *Smooth* and *Arbitrate* are used in the correct order does ESP provide significant cleaning benefit.

4.3 Discussion

Further analysis of this deployment reveals a number of interesting insights.

4.3.1 Calibration Issues

Although ESP corrected for many of the errors in the raw RFID data, it was not able to provide perfect results. These errors, seen in the uneven portions of the trace in Figure 3(d), arise because during these portions of the experiment, the reader for shelf 0 read the tags on shelf 1 more than shelf 1's reader did. We alleviated the effects of the disparities between the antennae to a certain extent through crude calibration: in *Arbitrate* processing, ESP attributed a reading to the weaker antenna if the counts of the readings were equal. In general, with such poor raw data, ESP cannot correct for these errors without either calibrating the antennae or using outside information, such as an inventory list. ESP's extensibility allows calibration functions or static table joins (e.g., for inventory lookups) to be defined and inserted in a pipeline.

4.3.2 Size of the Temporal Granule

The size of the temporal granule affects the degree at which ESP can effectively clean the data. In order to effectively smooth, the size of the temporal granule (i.e., the window size) must be large enough to straddle any gaps in the input (i.e., it must be larger than the longest period of dropped readings in the input). The window size may not be made too large, however, as its size must be balanced with the rate of change of the data values. This tension can be observed in Figure 3(c), where the periods when tags are being relocated are not as accurately captured as the stable periods.

To investigate this issue, we compared the relative errors of the output of ESP using different temporal granule sizes for the *Smooth* stage. The results are show in Figure 6. At very small and very large granules, the error is larger than for granules around 5 seconds. Essentially, an effective temporal granule size is bounded at the low end by the reliability of the devices and at the high end by the rate of change of the data.

4.3.3 RFID Summary

Through its pipelined design that utilizes temporal and spatial granules, ESP alleviates the effects of missed and unreliable readings in RFID data and provides an accurate stream of data describing the items on a shelf. This stream can be used by a receptor-based system that uses traditional query processing techniques, oblivious to the unreliable behavior beneath it.

5 Environmental Monitoring

In the previous section, we demonstrated the ability of ESP to clean RFID data streams. Here we present a use case where ESP hides unreliabilities with a different technology: wireless sensor networks.

Wireless sensor networks are transforming the manner in which scientists monitor the physical environment [26, 28]. In order to alleviate the effects of imprecise readings, calibration errors, outliers, and unreliable network communication, previous deployments involving sensor networks have had to post-process the readings, primarily by hand, to produce specialized, application data that can be used for analysis [9, 12]. In a receptor-based system used for real-time environmental monitoring, this type of conversion, calibration, and correction must be done online.

5.1 Outlier Detection

Sensor motes are known to "fail dirty," that is, sensors fail but continue to report faulty readings. To provide an accurate picture of the real world, ESP can perform online outlier detection to alleviate the effects of these fail dirty motes.

To analyze the effectiveness of outlier detection using ESP, we use a 30 day trace from a sensor network deployed in the Intel Research Lab in Berkeley. We focus on three motes in the same room, assigned to the same proximity group. In this trace, one of the motes fails by reporting increasing temperatures, rising to over 100^oC. To correct for this behavior, we use the *Point* and *Merge* stages of ESP. *Smooth* is not used because it cannot correct for extended errors within one sensor[3]. *Arbitrate* is not necessary as there is only one spatial granule.

5.1.1 Stage 1: *Point*

Initially, the *Point* stage (Query 4) filters any readings beyond its expected range; in this case, ESP filters readings where the temperature is higher than 50^oC.

Query 4 *Simple filtering at the* Point *stage*

```
SELECT *
FROM point_input
WHERE temp < 50
```

Note that this functionality could be pushed to the motes themselves, effectively eliminating network traffic from a failed sensor.

5.1.2 Stage 3: *Merge*

The *Merge* stage does outlier detection within a spatial granule by determining the average of the readings from different motes in the same proximity group and then throwing out individual readings that are outside of one standard deviation from the mean (shown in Query 5).

Query 5 *Outlier detection in the* Merge *stage*

```
SELECT spatial_granule, AVG(temp)
FROM merge_input s [Range By '5 min']
     (SELECT spatial_granule, avg(temp) as avg,
                    stdev(temp) as stdev)
      FROM merge_input [Range By '5 min']) as a
WHERE a.spatial_granule = s.spatial_granule AND
      a.avg + a.stdev < s.temp AND
      a.avg - a.stdev > s.temp
```

These techniques are not intended to be statistically complex, but rather demonstrate the simplicity of ESP

[3]It could, however, be used to correct for single outlier readings in one mote using the same mechanism presented here.

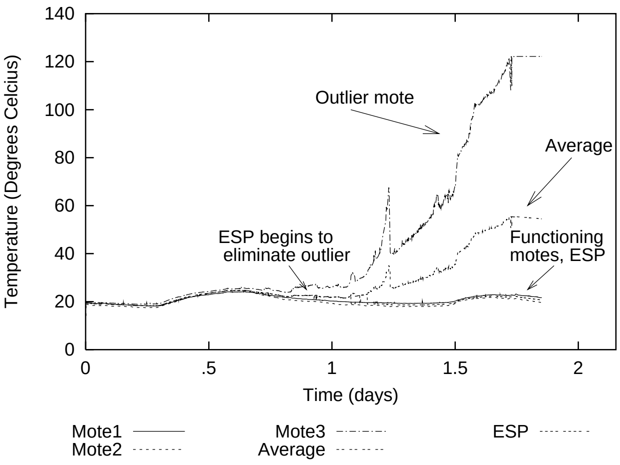


Figure 7: Outlier Detection using ESP. The "ESP" line tracks the two functional motes' lines

programming. As mentioned in Section 3.3, ESP enables a user to program more complex functionality involving UDFs, UDAs, or arbitrary code.

Figure 7 shows a trace of the three motes' individual readings, the average value over them without ESP processing, and the output of ESP with outlier detection processing. Observe that ESP is able to detect when the outlier mote begins to deviate from the other motes and then omit its reading from the average calculation. It is interesting to note that although *Point* is the first stage in the pipeline, *Merge* is the first stage to eliminate the outlier (at the time indicated in Figure 7). Although *Point* only begins filtering after the temperature rises above 50^oC, it is still useful for eliminating excess radio communication.

5.2 Sensor Network Data Cleaning using ESP

Wireless sensor networks have additional problems beyond fail dirty motes, such as frequently dropped messages. These problems are especially prevalent when sensor networks are deployed in the real world.

An ESP pipeline for a wireless sensor network can mask the unreliability of a sensor network by both temporally and spatially aggregating to correct for dropped readings. We demonstrate this cleaning through an environmental monitoring application, responsible for monitoring the temperature of a redwood tree at each altitude range in the tree. Sensors placed in the tree are grouped into proximity groups based on similar height in the tree (corresponding to the application's spatial granule).

We validated ESP processing on data collected during a month and a half period on a redwood tree in Sonoma County, CA as part of a large-scale sensor network deployment to study micro-climates of redwood trees [28]. 33 motes were placed along the trunk of the tree at varying heights. Data (e.g., temperature and humidity) were sensed at 5 minute intervals and logged to a local storage buffer (collected at the end of the experiment) and also sent over the multi-hop network. We use a roughly three and a half day trace where all motes were functioning (a software bug caused a number of motes to die partway through data collection). We grouped the motes at nearby heights into 2-node, non-overlapping proximity groups, where the distance between motes in a proximity group was less than one foot.

Note that the log data is incorrect with respect to the ground truth due to fail dirty sensors: 8 out of the 33 motes failed dirty. The readings from these motes were removed by hand shortly after data collection, but before we received the data[4].

As ESP is addressing communication errors in this case, our metric of success is the epoch yield. Epoch yield describes the number of the readings reported to the application as a fraction of the total number of readings the application requested. For the raw data, the epoch yield in this trace was 40% (ideally, the epoch yield should be 100%). In other words, the application only received 40% of the data it requested. Additionally, we measure the percent error in the readings. Based on experience collaborating with biologists, an error of less than 1^oC is acceptable for trend analysis. Therefore, the goal of ESP in this application should be to increase the epoch yield while minimizing the percent of readings with an error greater than 1^oC.

Here, we implement the *Smooth* and *Merge* stages in ESP to temporally and spatially aggregate sensor readings to increase the epoch yield of a sensor deployment.

5.2.1 Stage 2: *Smooth*

In the *Smooth* stage, ESP temporally aggregates readings from a single sensor. By running a sliding window average on each sensor stream, lost readings from a single mote are masked during the course of the window. After the *Smooth* stage, the epoch yield is increased to 77%. 99% of these readings were within 1^oC of the logged data.

Note that the collection parameters of this experiment were fixed to be the same as the temporal granule (recall that this data was collected by another group for the purpose of another experiment). In order to accumulate enough readings to smooth effectively, ESP had to expand its aggregation window to 30 minutes. Nevertheless, ESP still produced data at the temporal granule desired by the application. We discuss the implications of this type of window expansion below.

5.2.2 Stage 3: *Merge*

In the *Merge* stage, ESP performs spatial aggregation for each spatial granule (again, in the form of a windowed average) to further alleviate the effects of lost readings. The *Merge* stage increases the epoch yield to 92%. This improvement of reporting is at the slight cost of decreasing the percent of readings within 1^oC of the logged data to 94%. Thus, with ESP processing, biologists can get nearly complete data with a slight decrease in the accuracy.

5.3 Discussion

5.3.1 Receptor Actuation

As noted above, the *Smooth* stage had to expand its window in order to effectively clean the data. In this case, ESP's effectiveness was limited by the collection parameters of the data. Samples were collected sparsely, at five minute intervals, limiting ESP's ability to temporally aggregate. Ideally, ESP should be able to actuate the sensors to increase the number of readings within a temporal granule such that it can effectively smooth with a window the same size as the temporal granule.

5.3.2 Size of the Spatial Granule

Just as the size of the temporal granule affects ESP's ability to clean, so does the size of the spatial granule. The primary source of error in ESP's output stream in the redwood monitoring application occurred when there was a large change in the data values during a time when a sensor failed to report for a period, while at the same time the other sensors in the same proximity group were removed as outliers. In order to alleviate these errors, the spatial granule could be expanded in space to incorporate more devices[5]. This comes at the expense of possibly increased error. Similar to temporal granules, the size of a spatial granule must be balanced between the unreliability of the devices and the application's tolerance to error.

5.4 Environmental Monitoring Summary

Through the use of online techniques to temporally and spatially aggregate sensor network readings as well

[4] ESP could employ the techniques shown in Section 5.1 to remove these outliers automatically.

[5] Alternatively, additional physical devices could be installed to increase the granule's spatial density. As this involves a physical, offline action, we do not consider it here.

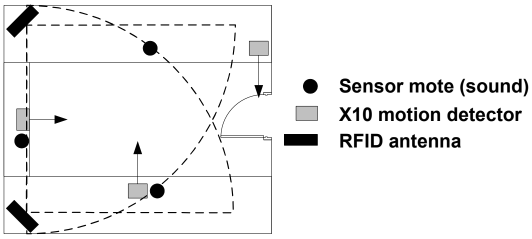


Figure 8: Digital Office Setup

as simple outlier detection, ESP is able to increase the ability of applications to make sense of the data they are getting from their receptors. Rather than have to spend time tediously post-processing the data, applications can focus on the high-level logic rather than conversion, calibration, and error correction.

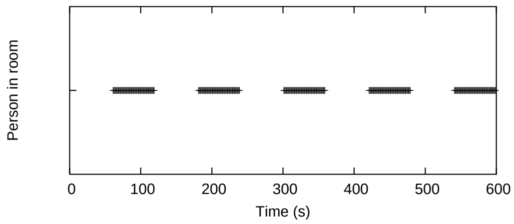
6 Digital Home

In Sections 4 and 5, we demonstrated how ESP provides a processing infrastructure to correct for a wide variety of problems associated with different physical devices. Here, we demonstrate the ease of configuration of ESP and highlight some of the advanced features of ESP through a digital home application scenario.

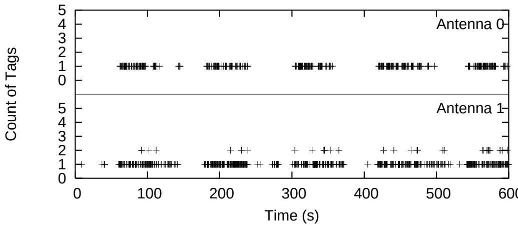
Multiple projects are developing sensors and infrastructures to instrument the home to provide both a better living experience for inhabitants as well as a more efficient use of house resources [3, 21]. Such applications use a wide variety of receptor devices providing low-level data (e.g., RFID, sensor motes, pressure sensors). ESP can provide a platform to clean the low-level device data (through pipelines similar to those defined in the previous two sections) as well as a means of supporting application-level cleaning to provide virtual "person detector" sensors. The output of ESP is a stream of events describing the presence of a person in the room.

We demonstrate ESP in a digital home scenario by outfitting an office in the Computer Science building of UC Berkeley with two RFID readers, a small sensor network of three motes, and three X10 motion detectors [31] tasked to determine when someone is in the office (Figure 8). The office corresponds to one spatial granule for the application; thus, the two RFID readers make up one proximity group, the motes constitute another, and the X10 detectors form a third. During the experiment, one person, outfitted with an RFID tag, moved in and out of the office, while talking, at one minute intervals (Figure 9(a)).

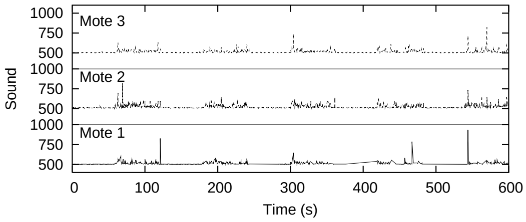
We present the ESP processing to clean the individual receptor streams and then describe how ESP utilizes these streams for application-level cleaning.



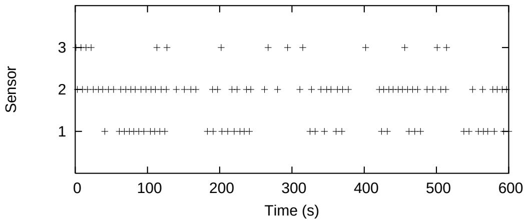
(a) Reality: one person moved in and out of a room every minute



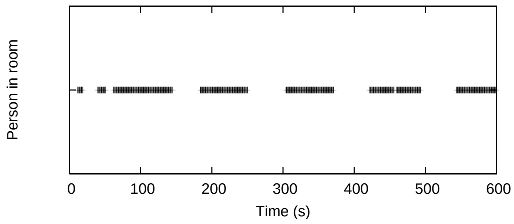
(b) Raw RFID readings from two antennas



(c) Raw sensor network sound readings from three motes



(d) Raw X10 motion detectors. A mark indicates that the device reported movement



(e) Data after ESP processing

Figure 9: A "Person Detector"

6.1 Low-Level Receptor Cleaning

RFID Readers

An advantage of ESP's pipelined approach is that changes necessary to tailor processing to each new deployment are isolated to small logical units. In this deployment, the programming for the ESP pipeline to process the RFID data is very similar to the pipeline presented in Section 4. Instead of implementing the *Arbitrate* stage to separate readings from different readers, we implement the *Merge* stage to union the streams from both of the readers (the readers are in the same proximity group). *Smooth* remains the same as the RFID scenario.

The raw readings from the RFID readers are shown in Figure 9(b). Note that antenna 1 occasionally reads an errant tag that is not part of the experiment. The *Point* stage of ESP is implemented to filter such readings through a join with a static relation containing expected tag IDs.

We omit the traces of the processed RFID data (as well as the processed data from the other receptors) due to space considerations.

Wireless Sensor Motes

The processing for the wireless sensor motes is almost identical to that of the ESP pipeline for redwood monitoring (Section 5), except in this case ESP processes sound readings rather than temperature. Note that because ESP utilizes declarative processing, this alteration involves only a small change in each query in the pipeline. The raw sound readings from the three motes are shown in Figure 9(c).

X10 Motion Detectors

X10 motion detectors provide a stream of "ON" events. These devices, however, have limited sensing capabilities and frequently fail to report or report when there is no motion in the room, as can be seen in the raw X10 traces in Figure 9(d).

To eliminate these errors, the *Smooth* stage for X10 processing interpolates "ON" events from a single detector in a similar manner as the RFID and sensor examples. The *Merge* stage combines the readings from all detectors in the room and reports motion if the number of readings exceed a threshold (e.g., if 2 out of 3 devices report motion). Once again, ESP's pipelined design assists in configuration as stages from other deployments can be reused.

6.2 Stage 5: *Virtualize*

The main new feature of this use case (as compared to the previous ones) is the use of the *Virtualize* stage. *Virtualize* allows a deployment to specify application-level cleaning to convert from the cleaned low-level device readings into application-level clean data; in this case, *Virtualize* turns the heterogeneous sensors into a "person detector." ESP uses a voting query (Query 6) that normalizes all receptor input streams to a single vote of whether it has determined that a person is in the room or not. The query then adds up the votes and registers that a person is in the room if the sum is higher than a threshold.

Query 6 *"Person Detector" logic at the* Virtualize *stage*

```
SELECT 'Person-in-room'
FROM (SELECT 1 as cnt
      FROM sensors_input [Range By 'NOW']
      WHERE sensors.noise > 525) as sensor_count,
     (SELECT 1 as cnt
      FROM rfid_input [Range By 'NOW']
      HAVING count(distinct tag_id) > 1)
      as rfid_count,
     (SELECT 1 as cnt
      FROM motion_input [Range By 'NOW']
      WHERE value = 'ON') as motion_count,
WHERE sensor_count.cnt +
      rfid_count.cnt +
      motion_count.cnt >= threshold
```

The output of the ESP pipeline is shown in Figure 9(e). As can be seen, simple, easy to deploy logic is capable of generally approximating reality. Here, ESP is able to correctly indicate that a person is in the room 92% of the time.

6.3 Discussion

6.3.1 A Platform for Advanced Processing

Although the use cases presented here all involved ESP stages defined through declarative queries, it is important to note that any stage in ESP can be implemented using user-defined code. Thus, stages can be extended to arbitrary levels of complexity. For instance, the *Virtualize* stage may be implemented with a function that incorporates machine-learning techniques to infer actions of the occupants of the digital home. Similarly, the *Virtualize* stage could also be implemented with a BBQ-like system [12]. Such a function would build models of the receptor streams to assist in cleaning the data.

6.4 Digital Home Summary

Here, we have shown that ESP is easily reconfigurable for each new application because ESP separates logically different data cleaning tasks into stages and because these stages can be programmed to a large extent with declarative queries. As observed in this deployment, we utilized already implemented stages and pipelines, dramatically speeding up deployment time.

7 Conclusions and Future Work

Data produced by physical receptor devices is notoriously dirty: readings are frequently either missed or dropped and individual readings are unreliable. Furthermore, these error characteristics vary from deployment to deployment. This leads to high costs for both data cleaning and configuration. To directly address these issues, we propose the ESP framework for online cleaning of receptor data streams. We introduce the concepts of temporal and spatial granule, which capture application-level notions of time and space. ESP utilizes these fundamental abstractions in a pipeline of programmable processing stages designed to clean receptor data as it streams through the system.

We validated the ESP platform through three real-world deployments demonstrating that ESP can successfully alleviate both missed and unreliable readings in receptor data. As a result, applications using receptor-based data were able to use data provided by ESP as they would any physical device data, but without many of the associated errors.

Through an analysis of these example deployments, we can distill general some principles for online cleaning of receptor data. First, there are many common operations across deployments. For instance, the *Smooth* stage was almost exactly the same for both RFID and sensor deployments. Second, when composing many applications, entire pipelines for processing low-level data can be reused as input to application-level cleaning. Finally, most of the changes necessary for each new deployment can be limited to a small number of stages and can be easily adjusted through declarative queries.

Therefore, we anticipate a suite of *ESP Operators*, implementing different ESP stages or entire pipelines, that can be used to configure and deploy cleaning pipelines. These operators, built using both declarative queries as shown here as well as user-defined functions, would make up a toolkit for ESP data cleaning. These operators can be extended or additional operators can be added to suit the needs of each deployment.

The ESP data cleaning framework augmented with a toolkit of ESP operators would enable rapid deployment of online data cleaning for receptor-based streams. Such a solution is necessary if we are to realize the potential of receptor-based systems to revolutionize how we view the physical world.

References

[1] Alien Technology. Nanoscanner Reader User Guide.

[2] Alien Technology. Personal correspondence.

[3] Mit house_n. Http://architecture.mit.edu/house_n/.

[4] Alien ALL-9250 I2 RFID tag. http://www.alientechnology.com/products/rfid-tags.

[5] Alien ALR-9780 915 MHz RFID Reader. http://www.alientechnology.com/products/rfid-readers/alr9780.php.

[6] A. Arasu, *et al.*. The CQL continuous query language: Semantic foundations and query execution. *VLDB Journal*, (To appear).

[7] Ascential. http://www.ascential.com/.

[8] P. Bonnet, *et al.*. Towards sensor database systems. In *Proc. Mobile Data Management*, volume 1987 of *Lecture Notes in Computer Science*. Springer, Hong Kong, January 2001.

[9] P. Buonadonna, *et al.*. Task: Sensor network in a box. In *EWSN*. 2005.

[10] S. Clark, *et al.*. Auto-id savant specification 1.0. sept 2003.

[11] O. Cooper, *et al.*. HiFi: A Unified Architecture for High Fan-in Systems.

[12] A. Deshpande, *et al.*. Model-Driven Data Acquisition in Sensor Networks. In *VLDB Conference*. 2004.

[13] E. Elnahrawy *et al.*. Cleaning and querying noisy sensors. In *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. 2003.

[14] EPCGlobal, Inc. http://www.epcglobalinc.org/.

[15] K. Fishkin, *et al.*. I Sense a Disturbance in the Force: Unobtrusive Detection of Interactions with RFID-tagged Objects. Technical Report IRS-TR-04-013, Intel Research, June 2004.

[16] C. Floerkemeier *et al.*. Issues with RFID usage in ubiquitous computing applications. In A. Ferscha *et al.*, eds., *Pervasive Computing: Second International Conference, PERVASIVE 2004*. 2004.

[17] M. J. Franklin, *et al.*. Design Considerations for High Fan-In Systems: The HiFi Approach. In *CIDR*. 2005.

[18] H. Galhardas, *et al.*. Declarative data cleaning: Language, model, and algorithms. In *VLDB*, pp. 371–380. 2001.

[19] Informatica. http://www.informatica.com/.

[20] Intel Lab Data. http://berkeley.intel-research.net/labdata/.

[21] C. D. Kidd, *et al.*. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In *Cooperative Buildings*, pp. 191–198. 1999.

[22] S. Madden *et al.*. Fjording the stream: An architecture for queries over streaming sensor data. In *ICDE*. 2002.

[23] S. Madden, *et al.*. The Design of an Acquisitional Query Processor For Sensor Networks. In *SIGMOD*. 2003.

[24] S. Mukhopadhyay, *et al.*. Data aware, low cost error correction for wireless sensor networks. In *WCNC*. 2004.

[25] M. Philipose, *et al.*. Mapping and Localization with RFID Technology. Technical Report IRS-TR-03-014, Intel Research, December 2003.

[26] J. Polastre, *et al.*. Analysis of wireless sensor networks for habitat monitoring. 2004.

[27] E. Rahm *et al.*. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[28] Sonoma Redwood Sensor Network Deployment. http://www.cs.berkeley.edu/ get/sonoma/.

[29] G. Tolle, *et al.*. A macrosope in the redwoods. In *submission*. 2005.

[30] Vijayshankar Raman and Joseph M. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In *The VLDB Journal*, pp. 381–390. 2001.

[31] X10. http://www.x10.com.