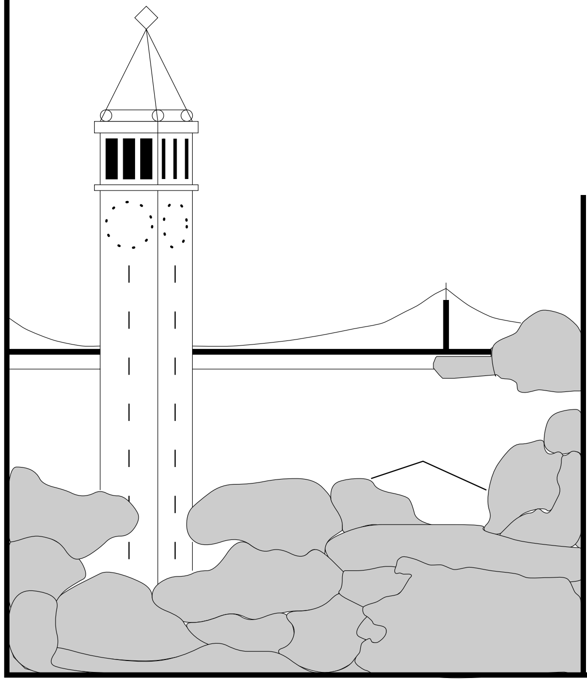
Virtual Devices: An Extensible Architecture for Bridging the Physical-Digital Divide

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



Report No. UCB/CSD-5-1375

March 2005

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

Virtual Devices: An Extensible Architecture for Bridging the Physical-Digital Divide

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 low-level, device-specific, inconsistent, and unreliable. Meanwhile, data management and query processing techniques in the *digital* world rely on understandable, consistent, and complete data. This paper aims to bridge this *physical-digital divide* by introducing *Virtual devICEs* (*VICEs*): extensible wrappers that transform the data captured from the physical world into data that can be operated upon by today's data management systems. In this sense, VICEs provide what we term *metaphysical data independence* by shielding the above data processing system from the complexity of real-world data. This paper introduces the specific challenges to implementing VICEs, the functionality required of them, and outlines the VICE architecture. To validate the VICE approach, we demonstrate three real-world use cases that illustrate the empirical benefits of using VICEs to provide metaphysical data independence.

1 Introduction

With the widespread deployment of receptor devices such as wireless sensor networks and RFID technologies, the physical world is being brought ever closer to the digital world: RFID provides enterprises with up-to-the-second information on their supply chains [18]; wireless sensor networks enable unprecedented visibility into environmental and structural processes [32, 36]; and ubiquitous computing technology is changing the way we interact with our surroundings [3, 26].

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 natural structure for a large-scale receptor-based information system is as a hierarchy with receptor devices at the leaves, and more traditional compute nodes at the higher levels [23]. For example, Figure 1 shows such a structure for a supply chain management scenario. A key challenge in the development of large-scale receptor-based systems is the need to reconcile the requirements and limitations of legacy (and even newer, stream-based) data management software at the higher levels of the system (i.e., the *digital* world), with the complexities, uncertainties, and problems that arise from the inherent imperfections of the data produced by the receptors measuring phenomena in the real (i.e., *physical*) world. This paper proposes an architectural approach for bridging the divide between these two worlds.

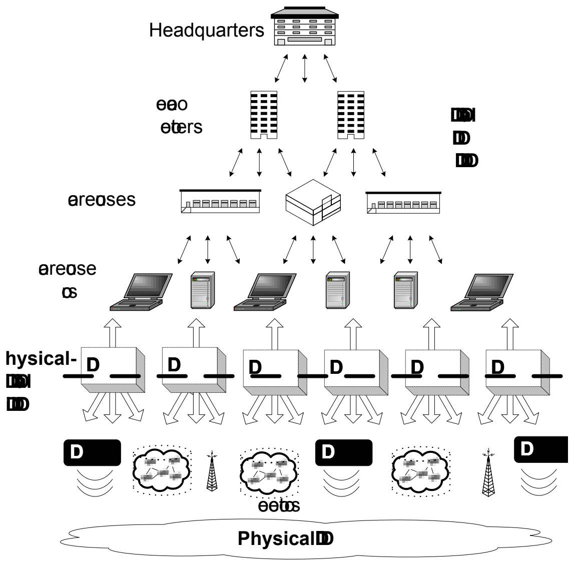


Figure 1: Supply Chain Management Scenario

1.1 The Physical-Digital Divide

The gulf between the nature of data from the physical world and the assumptions of the digital world is what we term "the physical-digital divide." This divide stems from four main causes:

1. ***Sensor Limitations***: The data acquired from receptors are typically inaccurate, may contain erroneous measurements, and may be missing values. For example, RFID readers often capture only 60-70% of the tags in their vicinity [8, 22, 30].
2. ***Semantic mismatch***: Physical sensing devices produce raw data that often has little application-level meaning and thus can seldom be used directly. For instance, industrial applications frequently employ *soft sensor* [33] technology that processes readings from one or more "hard sensors" to produce data that cannot be directly measured [41].
3. ***Data Volume***: Receptor devices can produce vast

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

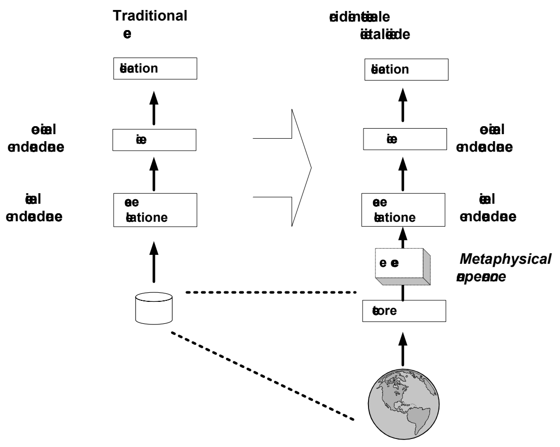


Figure 2: Metaphysical Data Independence

amounts of data in aggregate, much of it irrelevant, that threatens to overwhelm any IT infrastructure. For example, the output of large RFID infrastructures is projected to reach terabytes per day [29].

4. ***Device Complexities***: Receptor devices introduce significant complexity due to their wide variance in terms of interface, behavior, and reliability. Receptor interfaces are typically non-standard and may include support for complex interactions such as actuation [28, 40] and calibration — interactions that are not typically supported by current data management software.

These issues have been recognized to some extent, and there is a considerable amount of work that targets them individually and at various architectural levels (e.g., [12, 14, 19]). The solutions proposed to date, however, tend to be focused on single receptor types and are typically divorced from the larger data management context in which the receptor information will be used. The result is that current sensor deployments are labor-intensive to build and deploy, small-scale, brittle, and difficult to manage and operate for reasonable lengths of time.

1.2 Virtual Devices

In order to address this problem, we propose an extensible architecture called *Virtual devICEs* (VICEs). As shown in Figure 1, VICEs are placed near the edge of the network, interposed between receptors and the high-level data management system. VICEs appear to the above system to be idealized physical devices with a uniform interface. Effectively, a VICE is a "wrapper for the physical world," transforming receptor data into a form that can be processed easily by the higher levels of the system. For instance, in the Supply Chain Management scenario, VICEs can be used to make error-prone RFID readers look (to the higher levels) like idealized, accurate versions of such devices, or can be used to fuse data from groups of different types of receptors (e.g., sound sensors and RFID readers) to produce "virtual" sensors that monitor phenomena that no single physical sensor could accurately measure.

Just as traditional database systems provide *physical data independence* that fully separates the data model from the storage structures on disk, VICEs are architectural modules designed to encapsulate the processing necessary to shield the data management and query processing components above them from the complexity of the real-world data being collected below them. We refer to this new type of data independence as *metaphysical*[1] data independence (see Figure 2).

Metaphysical data independence separates receptor data from the particular sensing, error, and reporting characteristics of the physical devices below the physical-digital divide. This independence is essential for the development and scalable deployment of receptor-based systems. VICEs provide the following benefits:

- Errors and inconsistencies in device data can be removed.
- Devices can be added, be removed, or fail without requiring changes to the higher-level software.
- Data in the physical world can be accessed through a variety of access methods: physical devices of differing types (e.g., temperature may be accessed through a sensor mote or an active RFID tag), soft sensors, or through model-based processing.
- Receptor-based systems can use a uniform interface to access receptor data.

VICEs leverage the fact that many of these services are generic, can be mapped to a more meaningful, reusable architecture, and can be programmed to a great extent using declarative query interfaces.

1.3 Contributions

In this paper, we identify the physical-digital divide and introduce the concept of metaphysical data independence to hide the problems associated with crossing this divide. To directly address these issues within a VICE, we introduce a programmable set of processing stages (called ESP) designed to convert raw receptor data into data a receptor-based system can use directly (Section 3). We then demonstrate the wide applicability of the VICE approach through three detailed use cases of VICE processing built using ESP (Sections 4, 5, and 6), and describe experiments verifying that VICEs provide significant improvements over raw receptor data in terms of data quality and functionality. Finally, we outline our initial work on defining the full architectural solution for VICEs that encapsulates ESP and additional support modules necessary to provide metaphysical data independence (Section 7).

In the following section, we first place VICEs in the context of related work that has addressed issues relating to the physical-digital divide.

[1]The term "metaphysical" is a loose reference to Plato's *Divided Line* [31] philosophy on the real world and the perceived world. He conjectured that the physical world is only accessible through a person's imperfect senses and thus knowledge and reason must be used to guide any perception of the real world.

2 Related Work

Many different projects have investigated issues relating to the physical-digital divide. This work can be grouped into two categories: 1) Receptor-specific solutions, and 2) Real-world data systems. One of the goals of Virtual Devices is to provide an infrastructure and context to unify techniques from both categories in a single architecture.

2.1 Receptor-specific Solutions

A wide range of projects are addressing issues relating to the physical-digital divide for a single type of receptor.

Multiple systems provide mechanisms for abstracting and interacting with wireless sensor networks ([28, 11]). For example, TinyDB is used as a building block for acquiring data from a sensor network; however, the data received for TinyDB must be first cleaned and processed before it can be used by any application.

Another important line of work in sensor network data processing is model-driven query processing, such as in the BBQ system [17]. The main idea behind this work is to use models of reality to optimize different aspects of the data acquisition process. The architecture we propose for VICEs includes support for plugging in such approaches. The goal here is not to supersede model-driven data acquisition, but to develop an architecture where such techniques can be combined with other solutions addressing various aspects of metaphysical data independence.

Savant middleware [14] 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 does not address the problems from a high-level data processing view: it does not deal with receptors beyond RFID technology, nor does it address many of the associated data processing issues when dealing with unreliable devices.

VICE processing is analogous in some ways to the data transformations used in scientific environments to transform Level 0 (or L0) data into Level 1 (or L1) data [27]. In these applications (e.g., satellite image processing), L0 data is raw data directly captured by the instrument with little or no processing. L1 data, on the other hand, is produced by cleaning, filtering, tagging, sorting, indexing, and formatting the L0 data. Scientists typically work with L1 data to then produce L2 data (derived data or data products). VICEs are intended to provide this type of functionality, among other features, to receptor-based systems.

2.2 Real-world data systems

There are a variety of projects dealing with various aspects of real-world data processing.

Recently, there has been growing interest in the general problem of data quality and provenance, e.g., [38]. These problems are particularly acute at the physical-digital divide. Of course, there is a large body of work on data quality and cleaning techniques [13, 24, 25, 34]. The work reported in this paper does not address specific solutions to the data quality and provenance problems, but rather it provides the appropriate architectural context to incorporate such solutions.

Soft sensor [33] technology has long been a mainstay of industrial process monitoring. Soft sensors are usually machine-learning-based software modules that process input from multiple "hard sensors" to produce data that measures an unobservable variable. For instance, a beer brewer may use sensor measurements about various characteristics of yeast during fermentation, fed through a neural network, to determine the fermentation speed of the beer [35]. Similarly, *sensor fusion* [16] combines data from multiple sensors to provide a coherent view of the physical world. As with model-driven processing, VICEs provide an architectural framework for such technology.

The pervasive, mobile, and ubiquitous communities study how to tie the world in which we live with the digital world. However, their research thrust has been focused primarily on usability, expressiveness, and functionality [9, 20, 37]. In contrast, VICEs look to address issues dealing with integrating the physical world as a *data source* into digital systems.

Finally, VICEs are part of the HiFi [23] project. HiFi is a distributed, hierarchical stream processing system designed to support large-scale receptor-based networks, termed "high fan-in" systems. HiFi is built using a *uniform declarative framework* where all nodes in the system export declarative interfaces. This approach reduces the complexity of deploying such a system and provides many optimization opportunities. As was shown in Figure 1, VICEs are designed to provide metaphysical data independence at the edge of the network such that the higher levels of HiFi can use more standard declarative stream query processing.

3 VICE Processing Overview

In this section, we present ESP, the VICE data processing model.

While building the initial version of HiFi [15], we confronted many of the issues associated with the physical-digital divide. Most notably, we observed that the system was unable to produce meaningful answers to any of our application-level queries (e.g., "tell me the number of RFID tags in the area") when used directly with raw RFID data.

Our solution was to use a rudimentary pipeline of ad-hoc queries we termed "CSAVA" [23], designed to run throughout the HiFi hierarchy to convert the RFID data to application data that we could use in our demonstrations. The result of CSAVA was an improved RFID data stream.

Extensible receptor Stream Processing (*ESP*) generalizes and extends the CSAVA pipeline with a focus on processing physical device data at the edge of the network. ESP, shown in Figure 3, is the primary mechanism VICEs use to convert raw physical data into a form that can be used in traditional data processing systems by virtualizing multiple physical receptor streams into a single, improved output stream. ESP segments receptor stream processing into a logical cascade of five programmable stages (*Point*

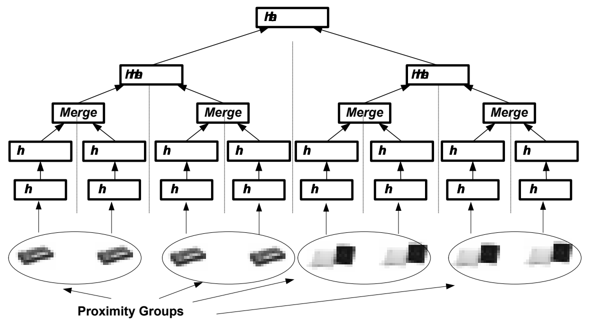


Figure 3: ESP Processing Stages

- *Smooth* - *Merge* - *Arbitrate* - *Virtualize*) that operate on successively higher levels of the data, in terms of volume, complexity, and semantics.

- Stage 1, ***Point***: This stage operates over a single value in a receptor stream. The primary purpose of this stage is to filter individual values that are not needed by the application (e.g., errant RFID tags or obvious outliers).
- Stage 2, ***Smooth***: The *Smooth* stage processes values in a sliding window over a single receptor stream, temporally aggregating and interpolating to correct for missed or dropped readings. Additional filtering of data can be done here as well, such as removing duplicate events (e.g., an item is still on the shelf).

Stages 1 and 2 are roughly analogous to the first two stages of CSAVA. The remaining stages differ more significantly from the original CSAVA approach. In general, receptor-based applications are not interested in 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, VICE processing 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. Devices in the same proximity group are spatially aggregated to produce data at the spatial granularity desired by the application.

It should be noted that proximity groups 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.

- Stage 3, ***Merge***: This stage converts from individual device readings to application-level spatial granules through spatial aggregation of proximity groups.
- Stage 4, ***Arbitrate***: This stage deals with conflicts between readings from different proximity groups.
- Stage 5, ***Virtualize***: This stage combines readings from different types of devices and different proximity groups to produce application-level data.

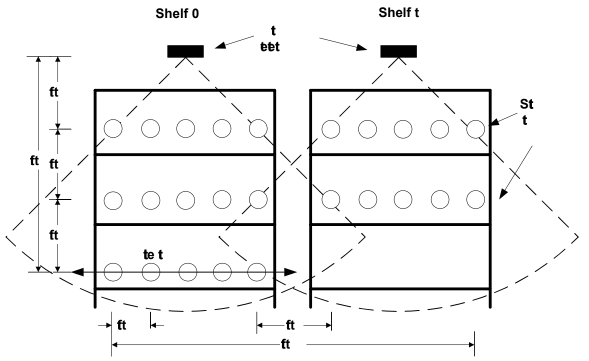


Figure 4: Shelf scenario setup with 2 shelves, each with an RFID reader and 10 tags statically placed within 6 feet of the antenna (5 tags at 3 feet, 5 tags at 6 feet). Additionally, 5 tags were relocated every 40 seconds.

These stages may be programmed through declarative stream queries [7] or through user-defined functions. As shown in the following sections, a VICE applies these processing stages in turn as data flow up the cascade.

In the next three sections, we illustrate the effectiveness of this processing model through three real-world use cases: RFID technology in a retail scenario, wireless sensor networks used for environmental monitoring, and a digital home application employing multiple types of receptors to create a "person detector" sensor. We then describe the architecture of a VICE in Section 7.

4 Use Case 1: RFID Data Processing

RFID 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. These errors are especially troublesome when RFID technology is used in a retail scenario, for example, to monitor warehouse or store shelves. In such a case, applications using raw RFID data may report that items frequently disappear and reappear, or jump to nearby shelves because of unreliable readers or other readers in close proximity.

Here we demonstrate the effectiveness of ESP when applied to a real-world retail scenario where RFID readers monitor items on shelves. In such an environment, an application would monitor the status of items on a shelf to determine when each shelf needs to be restocked (Query 1).

Query 1 *Shelf monitoring application query to determine the number of items on each shelf.*

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

Our experimental setup is depicted in Figure 4. We used two 915 MHz RFID readers [5] from Alien Technology [6], each responsible for a shelf. The readers' sample period was set at 5Hz (i.e., 5 readings per second). Each shelf was stocked with 10 items represented by 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 the performance of RFID technology as it attempts to alleviate many of the known causes of degraded readings [21]. 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. Each reader produces a stream, `rfid_data`, that consists of tuples containing a tag ID and an associated timestamp.

Figure 5(a) depicts the trace of the desired outcome of the application's query (Query 1) given our experimental setup. If the application were to use the output of the RFID readers directly, the results would be near-meaningless (Figure 5(b)): the average relative error of the output of Query 1 for the duration of the experiment was 0.83. These large variations in the count of items on the shelf would wreak havoc with any application using the raw data and cause it to report that the shelf needed to be restocked constantly. For instance, if the restock threshold is set to 5 items, then the query would demand that a shelf is in need of restocking 2.3 times per second, on average.

Obviously, this meaningless data could be cleaned in many different ways. One solution is to change the query to account for the errors in the data; however, this solution is dependent on the underlying technology and environment and is not generally applicable. Another solution is to hand-code low-level processing to correct of these errors. VICEs provide an architecture where such processing can be done in a structured manner and can be used in many different application contexts. With such an architecture, any receptor-based system can use the data stream produced by a VICE as it would a raw RFID reader, but with substantially better data quality.

To mask the poor quality of the RFID data from the application, a *ShelfVICE* can be tasked to manage multiple nearby RFID readers to provide an accurate stream of data relating to the items that are on each shelf. Note that the RFID reader already provides *Point* functionality by removing tags that fail a checksum [1]. Thus, we implement the *Smooth* and *Arbitrate* stages[2] for a ShelfVICE. We describe each processing stage with the queries, expressed in CQL [7], used to implement the ShelfVICE.

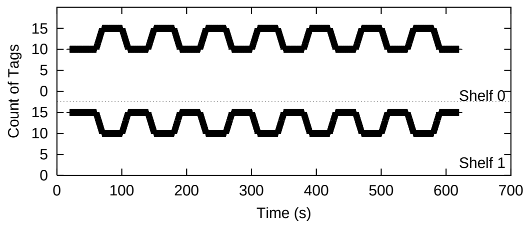
4.1 Stage 2: *Smooth*

At the *Smooth* stage (Query 2), the ShelfVICE interpolates for lost readings by temporally aggregating over a window from a single receptor stream. The VICE applies this query to each receptor stream (as illustrated in Figure 3).

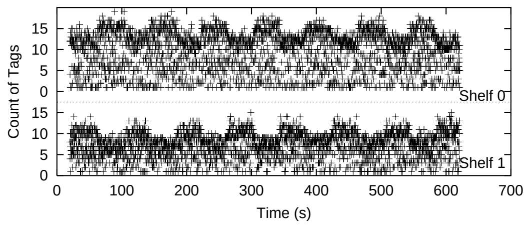
Query 2 *Interpolating for lost readings at the* Smooth *stage*

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

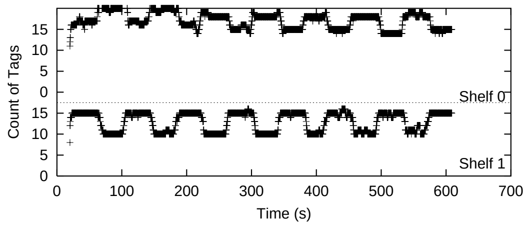
[2]We demonstrate the other processing stages in the use cases in the following two sections.



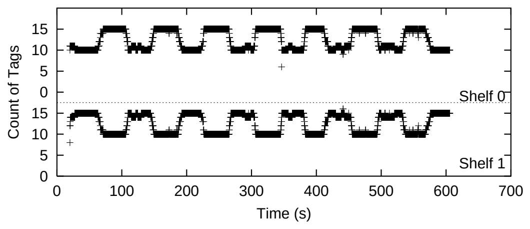
(a) Reality



(b) Query 1 results using raw RFID data



(c) Query 1 results after *Smooth* processing



(d) Query 1 results after *Arbitrate* processing

Figure 5: Query 1 results after different stages of processing

The results of Query 1 over the data produced by this stage are shown in Figure 5(c). The *Smooth* stage is able to eliminate the constant restocking alerts generated by the query run on the raw data.

The count of items per shelf, however, is still inaccurate (an average relative error of 0.50) due to discrepancies in the performance of the readers. In this experiment, the antenna for shelf 0 performed significantly better than that of shelf 1, despite being the same model; it frequently read tags that were on shelf 1. As seen in Figure 5(c), the counts reported for shelf 0 were consistently 4 to 5 items more than reality. We switched the antennae and witnessed the same behavior. 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 (Query 3) corrects for the discrepancies between antennae left unresolved by the *Smooth* stage. It does this by attributing an item to the shelf that read the item's tag the most times in a window.

Note that although the *Merge* stage is unused in this case, the VICE automatically adds a `region` attribute to each stream, corresponding to each proximity group (i.e., each shelf). The VICE runs Query 3 over the union of these streams.

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

```
SELECT region, tag_id
FROM arbitrate_input ai1 [Range By ’5 sec’]
GROUP BY region, tag_id
HAVING count(*) >= ALL(SELECT count(*)
                   FROM arbitrate_input ai2
                   [Range By ’5 sec’]
                   WHERE ai1.tag_id = ai2.tag_id
                   GROUP BY region)
```

The results of running Query 1 over the arbitrated data is shown in Figure 5(d). Observe that the VICE is able to correct for the differing performance of the two antennae and to provide a substantially more accurate picture of the items on each shelf to the application. After *Arbitrate* processing, the average relative error for the duration of the experiment is 0.13 with a variance of 0.02. This equates to an error of being off by one or two items, on average.

4.3 Discussion

As can be seen Figure 5(d), the ShelfVICE provides an accurate stream of data describing the items on a shelf. This stream can be used effectively by a receptor-based system that uses traditional query processing techniques, oblivious to the unreliable behavior beneath it. Further analysis of the ShelfVICE use case reveals a number of interesting insights.

4.3.1 Calibration Issues

Although the VICE 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 5(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: the ShelfVICE attributed a reading to the weaker antenna if the counts of the readings were equal in *Arbitrate* processing. In general, with such poor raw data, a VICE cannot correct for these errors without either calibrating the antennae or through the use of outside information, such as an inventory list. Both of these operations must be supported within a VICE.

4.3.2 Window Size Issues

The use of windowed aggregate processing is instrumental in VICE processing; however, it also causes many subtle issues. First, windowed aggregates have a "primer phase" at the very beginning of processing while the first window is initially filling. During this time, the VICE reports erroneous readings due to too few readings in the window. This can be seen as the tail at the left edge of Figure 5(d). This behavior disappears as soon as the window is filled and thus is not an issue for steady-state processing.

A more important issue concerning windowed processing involves the size of the window (i.e., its `Range By` parameter) in both the *Smooth* and *Arbitrate* phases. In order to effectively smooth, the window must be large enough to straddle any gaps in the input (i.e., it must be larger than the longest gap 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 after the *Smooth* stage (Figure 5(c)), where the periods when tags are being relocated are not as accurately captured as the other periods. This slight inaccuracy is due to the fact that the window size for *Smooth* was set to alleviate errors in the steady state and was too large to capture the rapid change during transitions.

To investigate this issue, we compared the relative errors for different window sizes at the *Smooth* stage and found that very small (less than 1 second) and very large windows (greater than 10 seconds) had large errors (a relative error of 0.7 or more), while windows that were between 1 and 10 seconds had errors of less than 0.5. Essentially, an effective window 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.

In general, setting the window size is a difficult task. In these queries, we determined the window size experimentally based on trial-and-error, guided by some knowledge the environment (known performance of the readers and anticipated rates of change). A VICE implementor cannot be expected to know these parameters in most cases. Furthermore, there is no single correct window size for a given deployment, especially over time. Ideally, windows should be used only as the *mechanism* to do VICE processing, and should not be exposed to the user. Thus, a promising area of future work is to automatically determine and adjust window size based on the observed data input and change rate. At high data rates or for rapidly changing data, the system should use a small window, while for slow data rates or low rates of change, a larger window is better.

5 Use Case 2: Environmental Monitoring

In the previous section, we demonstrated the ability of a VICE to increase the quality of RFID data streams such that an application using RFID data can be unaware of the unreliable nature of such technology. Here we present a use case where a VICE can hide well-known problems with a different technology: wireless sensor networks, where significant errors can arise due to “fail dirty” sensors and lost readings.

Wireless sensor networks are transforming the manner in which scientists monitor the physical environment [32, 36]. However, 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 [12, 17]. In a receptor-based system with a shared infrastructure used for real-time environmental monitoring, this type of conversion, calibration, and correction must be done online and must be generally applicable across applications.

5.1 Outlier Detection

Sensor motes are known to “fail dirty,” that is, sensors fail but continue to report faulty readings. Thus, in order to provide an accurate picture of the real world, a VICE can perform online outlier detection to alleviate the effects of these fail dirty motes.

To analyze the effectiveness of outlier detection in a VICE, we use a 30 day trace from a sensor network deployed in the Intel Research Lab in Berkeley. The temperature in the Intel lab is climate controlled such that the temperature never rises above 30^{o}C. We focus on three motes in the same room, assigned to the same proximity group, but where one of the motes fails by reporting increased temperatures, rising to over 100^{o}C. If an application were monitoring the average temperature in this room to adjust the air conditioning, it would wrongfully set the air-conditioning to “high.” To correct for this behavior, we implement the only *Point* and *Merge* stages of a VICE.

5.1.1 Stage 1: *Point*

Initially, the the *Point* stage (Query 4) filters any readings that are outside of its expected range; in this case, the VICE filters readings where the temperature is higher than 50^{o}C. The VICE runs this query over each sensor stream.

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

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

Note that this functionality may be able to 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 across multiple streams by throwing out individual readings that are outside of one standard deviation from the mean (Query 5). For each proximity group (in this example, there is only one such group), the VICE runs this query over the union of the output streams of the *Point* stages.

Query 5 *Outlier detection query at the* Merge *stage*

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

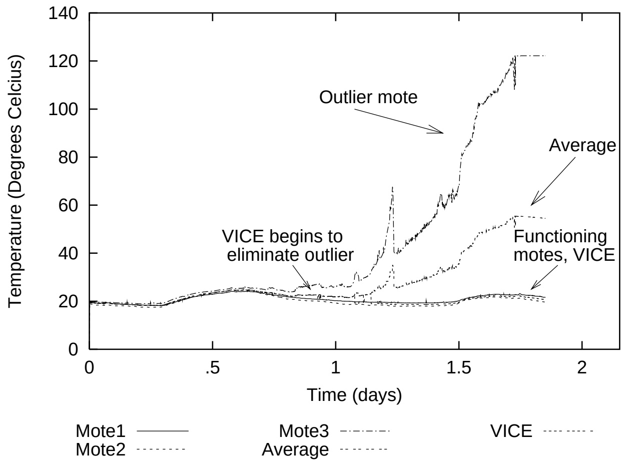


Figure 6: Outlier Detection in a VICE. The “VICE” line tracks the two functional motes’ lines

These techniques are not intended to be statistically complex, but rather demonstrate the simplicity of VICE programming. Of course, a VICE enables a user to program arbitrarily complex functionality.

Figure 6 shows a trace of the three motes’ individual readings, the average value over them without VICE processing, and the output of the VICE with outlier detection processing. Observe that the VICE 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 operation in the pipeline, *Merge* is the first stage to eliminate the outlier (at the time indicated in Figure 6). Although *Point* only begins filtering after the temperature rises above 50^{o}C, it is still useful for eliminating excess radio communication.

This fail dirty behavior of the motes is a worst-case scenario for outlier detection because it is difficult to differentiate between an outlier mote and a mote that is within normal bounds. A more common case is a single outlier value. This error correction can be easily handled by a VICE.

5.2 A RedwoodVICE

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.

A VICE for a wireless sensor network can mask the unreliability of a sensor by both temporally and spatially aggregating to correct for dropped readings. Note that although such a VICE addresses different errors characteristics than in the RFID case of Section 4, the same fundamental processing operators are used.

We demonstrate the ability of a VICE to mitigate the effects of transmission errors in a sensor network through a *RedwoodVICE*, responsible for monitoring the temperature of a redwood tree at each altitude range in the tree. A RedwoodVICE uses as inputs a stream, `sensor_data`, from each sensor. These streams are grouped into proximity groups based on similar height in the tree.

We validated the RedwoodVICE 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 done by Intel Research and UC Berkeley to better study the ecology and physiology of redwood trees [36]. 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 the same 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[3].

For our metric of success we use epoch yield, which 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%. In other words, the application only received 40% of the data it requested.

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

5.2.1 Stage 2: *Smooth*

At the *Smooth* stage, the RedwoodVICE 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[4].

5.2.2 Stage 3: *Merge*

Scientists monitoring redwood trees are interested in conditions at each altitude range of the tree. In the *Merge* stage, the VICE performs spatial aggregation across multiple sensor streams (again, in the form of a sliding window average) to further alleviate the effects of lost readings, taking advantage of motes that are in close physical proximity. This operation is performed for each proximity group.

The *Merge* stage further 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 VICE processing, biologists can get nearly complete data with a slight decrease in the accuracy.

5.3 Discussion

Through the use of online techniques to temporally and spatially aggregate sensor network readings as well as simple outlier detection, VICEs are 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.

It is interesting to note that the reliability of sensor networks can also be increased through the use of more expensive hardware or communication techniques. Such approaches are not feasible for large deployments that have power constraints. VICEs provide a mechanism to produce reliable results without expensive hardware.

As noted in the RFID use case, windowed processing plays an important role in providing an accurate stream of data from the VICE. In this case, however, the VICE's effectiveness is limited by the collection parameters of the data (recall that we used data gathered by another group for the purpose of another experiment). Samples were collected sparsely, at five minute intervals, limiting the VICE's ability to temporally aggregate. The more data in a window of a given size, the more accurate the result of the windowed aggregate. This suggests that the VICE needs to actuate the receptors during the course of processing to ensure that it is getting enough data such that it can meaningfully aggregate.

The primary source of error in the RedwoodVICE's output stream 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. This suggests that a VICE should incorporate model-driven data derivation [17] to assist in filling in these gaps. Additionally, a VICE should dynamically adjust proximity groups, perhaps through techniques such as clustering [10].

6 Use Case 3: Digital Home

In Sections 4 and 5, we demonstrated how VICEs provide a processing infrastructure to correct for a wide variety of problems associated with different physical devices. Here, we illustrate how a VICE can integrate multiple types receptors within a single architecture to produce a sensor for which no physical device exists.

Multiple projects are developing sensors and infrastructures to instrument the home to provide both a better liv-

[3] A RedwoodVICE could employ the same techniques shown in Section 5.1 to remove outliers automatically.

[4] Based on experience collaborating with biologists, an error of less than 1^oC is acceptable for trend analysis in this application.

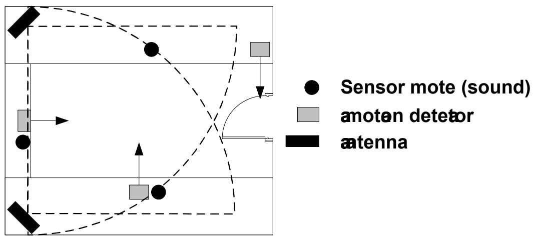


Figure 7: PersonDetectorVICE Setup

ing experience for inhabitants as well as a more efficient use of house resources [3, 26]. Such applications should not have to deal with low-level devices, but rather with high-level abstractions, such as the location of people in the house. Furthermore, such applications may employ a wide variety of sensors, frequently upgrading and changing devices (e.g., installing a webcam). This constant flux of devices must be hidden from the application behind a common application-level abstraction. In essense, digital home applications need to hide much of the complexity of such deployments using metaphysical data independence.

To provide both a means for hiding the low-level devices as well as to produce an application-level abstraction for digital home applications, such an application can employ *PersonDetectorVICEs* to manage and process the receptors in each room to provide virtual "person detector" sensors. The output of these VICEs are events describing the presence of a person in the room, or depending on the types of sensors, who is in the room. The digital home application writer can then focus on its application logic rather than deal with raw receptors.

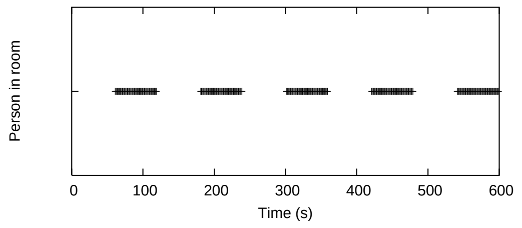
We demonstrate the PersonDetectorVICE by outfitting an office in the Computer Science building of UC Berkeley with two RFID readers, a sensor network of three motes, and three X10 motion detectors [39] tasked to determine when someone is in the office (Figure 7). During the experiment, one person, outfitted with an RFID tag, moved in and out of the office, while talking, at one minute intervals (Figure 8(a)).

6.1 RFID Readers

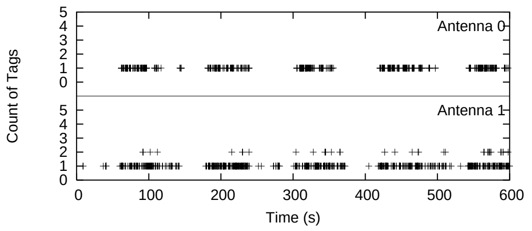
The raw readings from the RFID readers are shown in Figure 8(b). Note that antenna 1 occasionally reads an errant tag that is not part of the experiment. The *Point* stage of a PersonDetectorVICE is implemented to filter such readings through a join with a static relation containing expected tag IDs. We discuss the functionality necessary to support this in Section 7.

The programming for the RFID pipeline to process this data is similar to the ShelfVICE, but instead of implementing the *Arbitrate* stage to separate readings from different readers, we implement the *Merge* stage to union them. Here, the readers are monitoring the same area (i.e., they are in the same proximity group); thus, the *Merge* stage (Query 6) operates over the union of the streams from both of the readers.

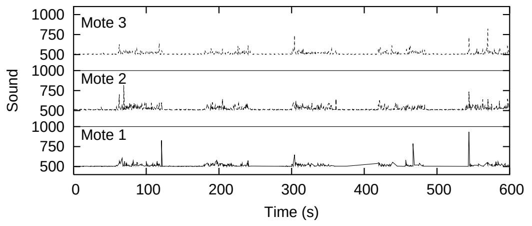
We omit the traces of the processed RFID data (as well



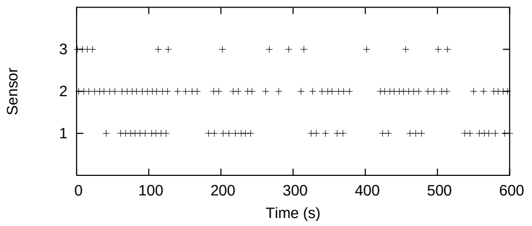
(a) Reality: one person moved in and out of a room every minute



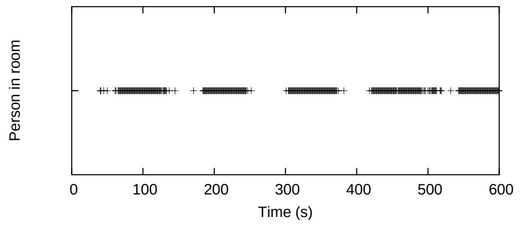
(b) Raw RFID readings from two antennas. The additional readings for antenna 1 were due to an errant RFID tag in the office



(c) Sensor network sound readings from three motes



(d) X10 Motion Detectors. A mark indicates that the device reported movement



(e) Data after VICE processing

Figure 8: A "Person Detector"

as the processed data from the other receptors) due to space considerations.

6.2 Wireless Sensor Motes

The raw sound readings from the sensors are shown in Figure 8(c). The programming for the sensor pipeline to process this data is almost identical to that of the Redwood-

Query 6 *Combining readings from the same proximity group at the* Merge *stage*

```
SELECT distinct tag_id
FROM merge_input [Range By ’5 sec’]
```

VICE, except in this case the VICE senses and processes sound readings.

6.3 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 Figure 8(d).

To eliminate these errors, the *Smooth* stage 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.

6.4 Stage 5: *Virtualize*

The main new feature of this use case (as compared to the previous two) is the use of the *Virtualize* stage. *Virtualize* contains the PersonDetectorVICE’s soft sensor logic to convert from the low-level (but cleaned, filtered, and aggregated) device readings into a “person detector.” In this case, the VICE uses a voting query (Query 7) 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 7 *“Person Detector” logic at the* Virtualize *stage*

```
SELECT ’Person-in-room’
FROM (SELECT 1 as cnt
      FROM sensors_input [Range By ’5 sec’]
      WHERE sensors.noise > 525) as sensor_count,
     (SELECT 1 as cnt
      FROM rfid_input [Range By ’5 sec’]
      HAVING count(distinct tag_id) > 1)
      as rfid_count,
     (SELECT 1 as cnt
      FROM motion_input [Range By ’5 sec’]
      WHERE value = ’ON’) as motion_count,
WHERE sensor_count.cnt +
      rfid_count.cnt +
      motion_count.cnt >= threshold
```

The output of the PersonDetectorVICE is shown in Figure 8(e). As can be seen, simple logic is capable of generally approximating reality. Here, the VICE is able to correctly indicate that a person is in the room 95% of the time with a false positive rate of 13% (using a window of 5 seconds and a threshold of 1). Adjusting the threshold of the *Virtualize* query can decrease the false positive rate at the cost of a decreased hit rate.

It should be noted that much of the effectiveness is enabled by the combination of all sensors. For instance, if the VICE were to omit the RFID input (i.e., use only sound and X10 sensors), the hit rate would drop to 69%. Similar effects can be seen by dropping out the contributions of other sensors (we omit the details for space reasons).

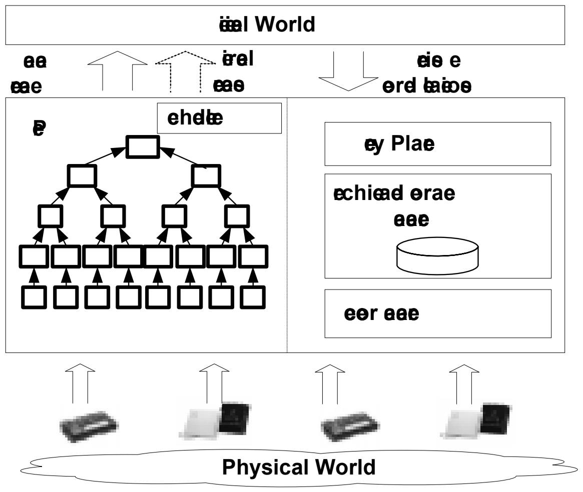


Figure 9: Logical VICE Architecture

7 VICE Architecture

In the previous sections, we introduced and validated ESP, the VICE processing model. Here, we present our initial approach to an overall VICE architecture that encapsulates this processing as well as additional support modules needed to provide metaphysical data independence.

7.1 Architecture Overview

Figure 9 shows the more important elements of the architecture of the VICE. The left side contains the hierarchical processing steps of the ESP model (as initially depicted in Figure 3). Directly associated with these stages is a *scheduler,* in charge of applying the defined processing stages to the data as they flow up the cascade, directing the traffic between the different processing elements, and allocating the necessary resources to each stage.

On the right side of Figure 9, there are a number of VICE’s support modules: the *Query planner and optimizer*, the *Data Archive and Storage Manager*, and the *Receptor Manager*. The main role of these modules is to provide the infrastructure for managing the VICE and the ESP stages. They are also used to provide a set of *virtual streams* of derived data that enhance the ability of the VICE to bridge the divide between the digital and the physical worlds.

While these three can be seen as the most basic modules, in practice, we envision an extensible architecture for the VICE where additional modules can be dynamically added. Additional modules may provide such services as automatic discovery and synchronization with receptors, maintaining group membership for sensor networks (e.g., appearance and disappearance of a sensor when the batteries are changed), and managing the life cycle of a sensor network (e.g., turning sensors on and off as needed, changing the network topology in response to data acquisition needs, incorporating new sensors, removing faulty sensors, etc.).

We describe each of the support modules in turn.

7.2 Query Planner and Optimizer

Just as TinyDB[28] provides a declarative interface for wireless sensor networks, VICEs provide a single declarative interface for querying, correlating, aggregating, and processing data across all types of receptors. The Query Planner is the module that provides this functionality. It accepts queries from clients and translates them into the necessary operations over the different processing stages. Needless to say, this translation offers numerous opportunities for optimization and, hence, the planner has to be extended with a query optimizer.

7.3 Data Archive and Storage Manager

Another basic support module for the VICE is a data archive, where receptor data is stored rather than processed as streams. There are several reasons to have such an archive in the VICE. First, we assume that the normal mode of operation will be processing over the VICE-processed data; however, often it is necessary to retain the raw receptor data, at least temporarily, for the purposes of audit tracking or lineage derivation. Second, there might be applications where the receptor data may arrive too fast for stream processing to be effective. Third, some of the queries over the data may involve archived data (e.g., for identifying composite events that process arbitrary sequences of data points).

As part of the Archive and Storage Manager, a VICE offers the opportunity to push data into the VICE for use as part of ESP processing. For instance, to decipher raw data from receptors, applications typically employ some form of augmentation, transformation, or validation (e.g., a tag list of relevant RFID tags as described in Section 6). The archive enables this push-to-the-edge functionality, where relations stored in a VICE's archive can be utilized in queries through simple joins.

7.4 Receptor Manager

Many low-level issues arise when interacting with physical receptor devices, such as actuation, calibration, and differing optimization considerations, all of which involve non-trivial interaction with the physical devices. Furthermore, interfaces for such interaction vary from type to type and even within a class of receptor. The Receptor Manager deals with issues concerning device interaction, such as actuation and receptor-specific optimization. For instance, to meet the result reporting requirements of a query (specified by its `Slide By` parameter), a VICE may need to adjust the sample period of the receptors to ensure adequate data.

Additionally, the Receptor Manager maps requests for receptor data to queries over the receptors, if the receptors support such interaction [28]. This mapping phase includes multi-query support by merging multiple queries into a single, more efficient query that shares processing and communication.

7.5 Virtual Streams

Using these support modules, a VICE can produce *virtual streams* that greatly enhance the functionality of a VICE. Virtual streams are data streams associated with the primary data stream that help a VICE's user to understand the data produced by a VICE. Virtual streams are correlated with the primary data stream through a timestamp. Note that many of these streams are application-dependent. Some of the virtual streams we are considering include:

- *Quality Stream*: A VICE should assess, track, and export *quality* of the data provided as part of the primary stream. The format of the quality stream is necessarily receptor dependent. For instance, a quality stream for sensor data would involve a timestamp, error bounds, and confidence, whereas the quality stream for RFID data would involve a timestamp, confidence, and perhaps a coverage [38].
- *Annotation stream*: To assist in debugging, VICEs may produce annotations describing operations it has done with the data.
- Partially-processed streams: To further facilitate debugging operations, VICEs may support access to partially processed streams; that is, the output of each processing stage may be individually accessible through queries.

As an example of what can be accomplished with virtual streams, a VICE can use the quality stream to incorporate model-driven query processing into a VICE. We do this by implementing the *Virtualize* stage with a BBQ-like system [17]. A VICE processes queries correlating the data stream with the quality stream (Query 8) by translating the input query into a BBQ-style query and then using BBQ to answer the query and assess the quality of that answer. Thus, VICEs can be easily extended to handle quality assessment and access with off-the-shelf components.

Query 8 *A query correlating the data stream and the quality stream to exploit model-driven capabilities*

```
SELECT d.temp, q.error, q.conf
FROM sensor_data d [Range By 'X' Slide By 'Y'],
     sensor_quality q [Range By 'X' Slide By 'Y']
WHERE d.ts = q.ts AND
      d.error <= 1 AND
      d.conf >= .95
```

8 Conclusions

In this paper, we have described the physical-digital divide and the challenges it creates. This divide appears between the physical world accessible through receptors and the digital world of receptor-based information systems. The divide arises due to limitations of physical sensing devices, semantic mismatches, the data volumes involved, and the complexity of dealing with heterogeneous receptor devices.

To directly address these issues, we introduce the concept of a Virtual Device to provide *metaphysical data independence*. Just as traditional database systems provide physical data independence, VICEs insulate higher data processing layers from the complexity of interacting with raw receptors and the data they produce.

There is a considerable amount of work addressing the issues associated with the physical-digital divide. Nevertheless, many of these efforts study isolated problems, are restricted to a single type of receptor, or apply only to ad-hoc settings. VICEs are intended to provide a coherent architectural framework for these efforts.

VICEs utilize a cascade of declarative processing stages, ESP, designed to successively clean, refine, and convert raw physical device readings into data the above system can use directly. In this paper, we showed three real world use cases demonstrating that VICEs using ESP can successfully protect receptor-based systems from many of the details of the underlying devices, in terms of data quality, complexity, and functionality. As a result, applications using receptor-based data were able to use data provided by a VICE as they would any physical device data, but without many of the associated errors and other limitations.

VICEs and the metaphysical data independence they provide are the key to building, deploying, and maintaining receptor-based systems, naturally extending the benefits of database and data management technology into 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] Alien Technology. http://www.alientechnology.com.

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

[8] Barnaby J. Feder. Despite Wal-Mart's Edict, Radio Tags Will Take Time. *New York Times*, Dec 27 2004.

[9] J. Barton *et al.*. The Challenges and Opportunities of Integrating the Physical World and Networked System. Technical Report HPL-2001-18, HP Labs, January 2001.

[10] S. Basagni. Distributed clustering for ad hoc networks. In *ISPAN '99: Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99)*, p. 310. IEEE Computer Society, 1999. ISBN 0-7695-0231-8.

[11] 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.

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

[13] S. Chaudhuri, *et al.*. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 313–324. ACM Press, 2003. ISBN 1-58113-634-X.

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

[15] O. Cooper, *et al.*. Hifi: A unified architecture for high fan-in systems.

[16] J. L. Crowley *et al.*. Principles and techniques for sensor data fusion. *Signal Process.*, 32(1-2):5–27, 1993. ISSN 0165-1684.

[17] A. Deshpande, *et al.*. Model-driven data acquisition in sensor networks. In *VLDB Conference*. 2004.

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

[19] D. Estrin, *et al.*. Next century challenges: Scalable coordination in sensor networks. In *Mobile Computing and Networking*, pp. 263–270. 1999.

[20] C. Fetzer *et al.*. Challenges in making pervasive systems dependable. In *Future Directions in Distributed Computing*, pp. 186–190. 2003.

[21] 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.

[22] 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*, number 3001 in LNCS, pp. 188–193. Springer-Verlag, Linz/Vienna, Austria, April 2004. ISBN 3-540-21835-1.

[23] M. J. Franklin, *et al.*. Design considerations for high fan-in systems: The HiFi approach. In *CIDR*. 2005.

[24] H. Galhardas, *et al.*. An extensible framework for data cleaning. In *ICDE*, p. 312. 2000.

[25] M. A. Hernandez *et al.*. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1):9–37, 1998.

[26] C. D. Kidd, *et al.*. The aware home: A living laboratory for ubiquitous computing research. In *Cooperative Buildings*, pp. 191–198. 1999.

[27] B. Kobler, *et al.*. Architecture and design of storage and data management for the nasa earth observing system data and information system (eosdis). In *MSS '95: Proceedings of the 14th IEEE Symposium on Mass Storage Systems*, p. 65. IEEE Computer Society, 1995. ISBN 0-8186-7064-9.

[28] S. Madden, *et al.*. The design of an acquisitional query processor for sensor networks. In *SIGMOD*. 2003.

[29] Manish Bhuptani and Shahram Moradpou. Emerging Trends in RFID. Feb 2005.

[30] M. Philipose, *et al.*. Mapping and localization with rfid technology. Technical Report IRS-TR-03-014, Intel Research, December 2003.

[31] Plato. *Republic*.

[32] J. Polastre, *et al.*. Analysis of wireless sensor networks for habitat monitoring. pp. 399–423, 2004.

[33] S. Qin. Neural networks for intelligent sensors and control — practical issues and some solutions, 1996.

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

[35] J. Rousu, *et al.*. Predicting the speed of beer fermentation in laboratory and industrial scale. In *IWANN (2)*, pp. 893–901. 1999.

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

[37] R. Want, *et al.*. Bridging physical and virtual worlds with electronic tags. In *CHI*, pp. 370–377. 1999.

[38] J. Widom. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In *CIDR*. 2005.

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

[40] W. Xue *et al.*. Action-oriented query processing for pervasive computing. In *CIDR*, pp. 305–316. 2005.

[41] S. Y. Yurish, *et al.*. Soft sensor for coffee beans moisture analysis. In *International Symposium on Sensor Science - I3S 2003*. 2003.