Towards an Energy Lens on the Physical World with Mobile Phones

Jorge Ortiz, Jason Trager, Gavin Saldanha, David Culler, Paul Wright
Computer Science Division
University of California, Berkeley
jortiz@cs.berkeley.edu

Abstract

Despite the growing impact of climate change and energy prices, per-capita energy consumption is rising. Part of the problem is visibility. We do not have scalable means of observing our energy consumption patterns and determining how to optimize and reduce our consumption. Mobile smartphones present a unique opportunity to enable an energy view on the physical world. They can bridge the physical world, information infrastructure, and people through a rich set of sensors, ubiquitous connectivity, and highly personal user interface. With QR codes as cheap tags on items and places in the physical world, the camera becomes a portable scanner in your pocket, in addition to its traditional functions. We explore this unique triple point and re-examine classical problems of context and consistency management in mobile systems. We also examine this combination as it pertains to energy management of physical devices. In doing so, we are re-introduced to problems of apportionment and aggregation of sensor data, except with a continuously changing set of constituents. We describe our solution in a technique called *dynamic aggregation* that maintains moving aggregates as the set of data sources changes over time. We deployed our system in a 141,000 square-foot building, tagging 351 items over 139 room across 7 floors.

1 Introduction

The United States leads the world in per-capita energy consumption; our electricity use has consistently increased over the last 40 years [11]; and, other parts of the world are rising all too rapidly. With the specter of climate change and the increasing cost of energy, we must explore new ways for individuals to gain visibility and insight into their energy consumption in order to optimize and reduce it. With the increasing penetration of embedded sensors in the environment and the continued rise smartphone adoption, we see an opportunity for smartphones to bridge the physical world to our computational infrastructure and provide an 'energy lens' on the physical world.

Smartphones serve as a natural point of intersection between the physical world, information, and people. The camera accepts input from the physical world, the screen functions as the interface to the individual, and its connectivity serves to overlay virtual services on top of the physical capture to be presented to through the screen. In addition, since phones are highly personal items, they serve as a good proxy for information about the user that can be used to personalize the views and services. Moreover, as mobile devices, views can be tailored to location and other contextual cues in the environment.

The use of mobile phones presents classical, fundamental challenges related to mobility. Typically, mobility refers to the phone as the person carrying it moves from place to place. However, in the energy-tracking context, we are also refering to the movement of energy-consuming objects. Tracking their relationships to spaces and people is as important as tracking people. In our deployment, we describe how we deal with both moving people and moving objects. We show that these historically difficult problems can be addressed relatively easily, if the proper infrastructure is in place. We provide evidence that the approach is simple, incrementally deployable, and scalable.

Our system uses QR codes to tag items and spaces in the physical world. Once tagged, there are three types of interactions - registration, linking, and scanning - establish important relationships. Registration is the act of creating a virtual object to represent a physical one. Linking captures the relationship between a pair of objects. Scanning is the act of performing an item-lookup. Each of these interactions requires a set of swiping gestures. Linking requires two tag swipes while the other two actions require a single tag swipe. We incorporate these gestures in a set of applications in a deployment we did inside a building on our campus. Our system was deployed incrementally in a 7-story, 141,000

square-foot building. We tagged 351 items spread over 139 rooms throughout all floors. On this infrastructure we built an energy auditing application, a device energy viewer, and a personalized energy tracker. Based on our initial deployment experience we observe that:

- QR codes are a convenient choice for tagging items because they are customizable, cheap, easily produced, and easily replaceable.
- Smartphones equipped with a camera can download QR code scanning software freely, making it a pervasive, effective, mobile scanner.
- Network connectivity is ubiquitous. Smartphones can connect to the internet through the cellular network or WiFi.

In addition, we describe how we address the following fundmanetal issues:

- Mobility. In order to provide energy-visibility, we need to track people *and* objects.
- Consistency management. In order to know what analytics to run we need to maintain an accurate view of the physical world in our virtual representation; that is, the objects and their interrelationship.
- Apportionment and aggregation. In order to provide real-time energy analytics we need to deal with the dynamics of a changing set data sources; fundamentally linked with the virutal view of the physical world.

We address each of these through a series of gestures that give us implicit and explicit information about people and the objects around them. We also use the virtual representation of the world to manage moving aggregates of physical data. Ultimately, we hope that detailed understanding of personal energy use will induce behavioral changes that reduce overall energy consumption.

2 Related Work

- Categories: mobile computing, plug-load studies, combination
- mobile: [9, 18, 25, 15]
- plugload: [1, 21, 20]
- both: [16, 19]
- data: [24]

Prior work in this area falls into 3 main categories discussed below. The first is a set of literature on mobile systems that is concerned with various aspects of mobile computing that highlight fundamental challenges with mobility, consistency, and infrastructure setup. They also explore deeper questions about services and their delivery through the infrastructure. Recently, there has been an increase in the characterization of plug-loads in buildings, since they account for about a third of the energy-consumption envelope. These sets of studies are about methodology for data collection, organzation of data, and plug-load characterisitics. The last set of studies combine elements of these two by using mobile systems to interact directly with the physical world to learn and control electrical loads more effectively. Electrical loads are a superset in which plug-load fall into. These works explore similar issues to those that fall into mobile system in combination both methological challenges with data-collection and context-aware challenges.

The other piece of related literature is in data modeling and processing, specifically the entity-relationship and in-network aggregation. In order to provide a view of the data that make sense to the user, data organization and management is of primary importance. We discuss these in the following next sub-sections.

2.1 Mobile systems

2.2 Plug-load metering

2.3 Combination

2.4 Data modeling and management

3 System Architecture

Our architecture consists of three components: mobile phones, QR codes, and a web application stack sitting in the cloud. The goal of our architecture is to enable virtual services on the physical world and to use the mobile phone as an intrinsic element of those services. An instance of our architecture was deployed inside a building and we make the following assumptions for our archicture to deliver our goals successfully:

1. Most building occupants own a smartphone that they carry with them most of the time.
2. Network connectivity is available throughout the building.

Figure 1 shows an overview of the components of our archiecture. We place QR codes on items throughout the entire building. This includes all building entrances, floors, rooms, and energy-consuming devices. Occupants use their phone and personal printer to add new items. For items that are already registered, occupants use their phone to scan their tags to learn more information about the items and to participate in maintaining a consistent view of the building. We discuss each of the three components in further detail in the following sections.

3.1 Mobile phone

The mobile phone is the main component of our architecture. It is a unique triple-point: the point of intersection between people, physical objects, and computational infrastructure. We take advantage of the engineering that has gone into making the interaction between the phone and its owner quite natural in order to give us personal and contextual information. For example, users naturally point towards items they are interested in capturing (when a picture is taken). We use this gesture to capture physical objects and incorporate

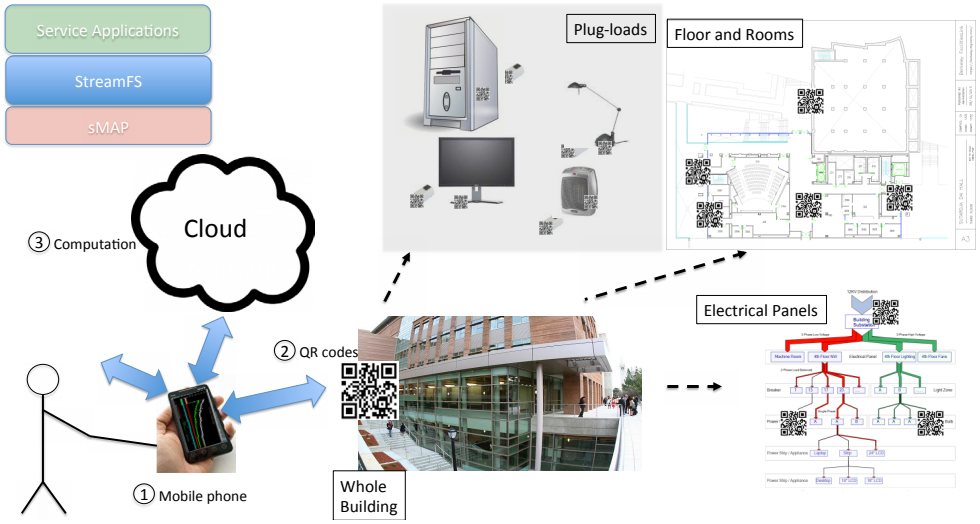


Figure 1. Above we show the system architecture. The main components are mobile phone, QR codes, and access to cloud our services. The mobile phone serves as the triple-point of the architecture: it serves as the point of intersection between the human, the physical world, and cloud services.

them in our virtual representation of the world. People use their mobile phone camera to scan QR codes in the environment. This can be used to obtain information about where they are, to obtain information about the object, or to obtain information about the relationship between objects. A simple gesture can provide the individual with a 'user inerface' for the room and the devices within it. [**?**, **?**, **?**, **?**, **?**, **?**] The screen can be used as a secondary input/output to give us/them more information about the object or context. We discuss these in more details in Section 5.

3.2 QR Codes

A QR code is a two-dimensional barcode that may encode almost 3000 bytes of data. QR code generators can be found on the internet [2, 3]. Extending the approach in [16], in our architecture the QR code contains a meaningful `URL` that a generic browser can access to provid ea human readable document with complete information about the item or space, as well as the additional information to bootstrap the smartphone to optimized access, such as native apps for interacting with the item or space. Secondly, the `URL` must be easily transformed into one that will yield a programmatic document, such as a JSON object, that apps can manipulate. And finally, the representation of the URL itself can be parsed and utilized locally by native apps, typically by lookup, to permit rapid interaction with the item or space.

Figure 2 shows an example QR code used in our deployment. Tags like these are placed on physical objects and spaces throughout the building to link between the physical world and our virtual representation of it. QR codes are cheap to produce. Any printer and some tape can be used to tag an item. This is important for scalability. With the number of physical objects and places (floors, rooms) in a typical building, **we must rely on the occupants to scale our deployment**. Because QR codes are easy to produce, we can provide occupants with a webpage that that produces them. They print them out, place them on items or places they want to interact with, register them, and provide useful information about them.

Figure 2. This is an example QR code from our deployment. This label resolves to `http://tinyurl.com/6235eyw`. QR codes like these are used as tags physical objects and spaces/locations.

Generating the right kind of QR code is important. It is trivial to encode information onto them, but it is not trivial to design them so that they encode just enough information to be useful. If too much information is encode the camera takes a long time to scan them, especially under poor lighting conditions. This can easily frustrate and drive away users, who are critical for scalability. We also want to design them for the lowest common denominator in terms of camera quality. Older phones with cheap cameras should also be able to scan the tags quickly. We ran some experiments to show how complex QR codes differ from the one we design and discuss these results in Section 4.2.

3.3 Computational infrastructure

The final piece of our architecture consists of three layered components for meter and usage data representation, data and metadata managament, and applications. Each layer in the stack is network accessible and hosted on a combination of machines we own and across two cloud-service providers. The distributed nature is a testament to the flexibility of placement for each of the pieces. However, the layering is imposed in our setup. Physical data streams, coming from wireless power meters [17], weather feeds from weather underground [4], the local building management system are directed towards the sMAP [10], above these feeds then forward their data to our context management layer, StreamFS [5], and finally, we build our applications on top of StreamFS. We describe each of these in the following sections.

3.3.1 Application layer

At the application layer we use a standard Apache web server [6]. Two of our apps – the item energy scanner and the personal energy counter – are essentially just web applications. When the phone scans a tag, it redirects to the main page which shows a list of services for that item that was scanned. It also provides an option to the user to login in to obtain a peronalized energy view. We discuss the details of these applications as well as the energy auditing application, which is a native app, in Section 5.

3.3.2 sMAP

sMAP [10] provides a uniform data access layer for sensor information. We can take any of the various sensor protocols and make it accessible through sMAP's HTTP, RESTful API. In addition, sMAP servers can live on any machine that is accessible through the web. This is convenient for our purposes, since it makes the raw sensor data stream accessible through any network. sMAP's data-forwarding facility is used extensively to link the sensor data with the context layer. When a meter is being added to the context layer, a callback is installed on the sMAP server that hosts that meter's stream. As meter data is reported to the sMAP server, it is forwarded to the appropriate node `URL` in StreamFS, setting up the context associated with that meter automatically.

3.3.3 StreamFS

We used StreamFS [5] as our data management layer. StreamFS offers various facilities to manage streaming sensor data and the associated metadata in a way that was useful for our the needs of our application. It organizes the metadata hierarchically and provides analytical facilities that are baked into it, making it easier to create energy-analytics applications. In particular, StreamFS deals well with data aggregation where the number of constituents used to calculate the aggregate changes over time. In StreamFS, this is called *dynamic aggregation* and it is described in more detail in Section 6.

Because we are fundmentally dealing with organizing data about the real-world, StreamFS is particularly useful over a relational database abstraction. StreamFS essentially constructs an entity-relationship graph [24] and exposes this graph through a object pathnames and symbolic links. It has been argued that semantic information is lost in a relational data model [14, 23]. Our applications are built on entity-relationship semantics, making StreamFS an ideal choice.

Each object in physical space is represented by a node in StreamFS. Through StreamFS' RESTful/HTTP interface, we can fetch the any node through a path rooted at the location where StreamFS is hosted. For example, if we wish to access the power meter in room 400 of building bldgXYZ, we may access it by issuing an HTTP `GET` request to `http://server.streamfs.com/bldgXYZ/rm400/pow`. The request returns a `JSON` object with attribute-value pairs that give a description of the temperature sensor. It also returned that last received value from that sensor. StreamFS provides a query interface to fetch the timeseries data collected from the sensor as well.

In addition, StreamFS support symbolic linking and this allows us to refer to nodes by multiple names. That same power meter can also be referred to through `/jortiz/meters/pow`; the meters that belong to user *jortiz*. More generally, StreamFS offers features that simplify namespace and data management. Semantically, the hierarchical node structure can be interpretted by the application. We describe the structure and interpretation in each application is section 5.

4 Gestures and QR code design

This section briefly describes the basic primitives and design decisions that are used across all our applications. The first is the set of gestures people perform with the smartphones to provide and obtain information about the physical world. The other describes our QR code design choices and why they are critical for engagement and scalability.

4.1 Gestures

Mobile phones are very well engineered for incorporating natural gestures to acquire and view information. We tried to take advantage of this by using simple gestures to obtain information about objects, context, and movement. Figure 3 show the basic gestures to perform various actions. The basic gestures fall into three categories: registration, linking, and acquisition. Registration is shown in the Figure 3(A). It consist of a single tag swipe and entering information through the screen. Linking gestures, shown is Figure 3(B), consist of two swipes and the press of button to confirm the un/linking. Finally, the acquisition swipe, shown in Figure 3(C), is a single swipe, select the data aggregate you wish to view, and viewing the data.

Each of these are natural for the user for capturing the physical world. Swipes *of* the physical world are taken to tell us about something *in* the physical world. This

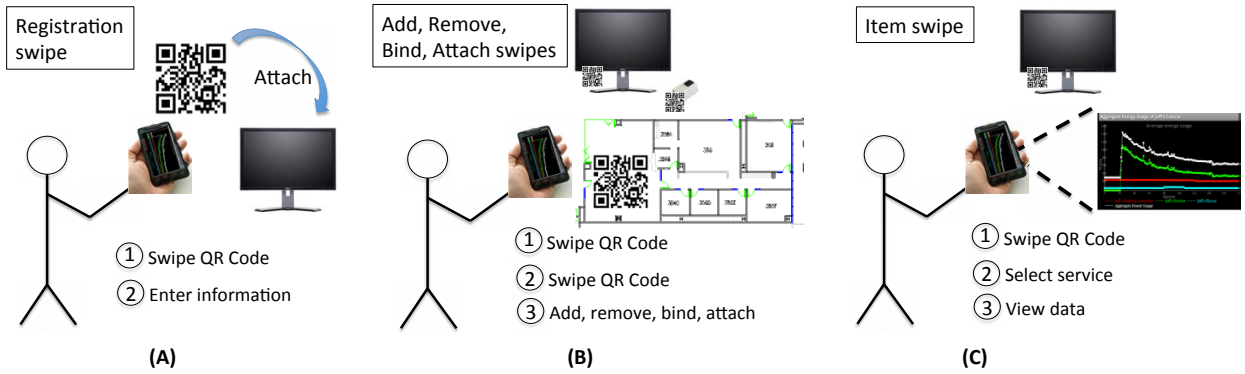


Figure 3. Gestures. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

makes the connection clear to the user about the kind of information they are informing the system about. In section 5 we will show how gestures also tell us a just enough information to infer information about context and movement. These gestures form the set of interaction primitives for all our applications.

The application logic associated with the gesture defines the form of the relationship or action that is associated with the gesture, such as `is-in`, `is-measured-by`, or `get-control-of`. This is in stark contrast to the use of gesture recognition to infer actions that the person wants to perform on a smart space. The person takes a natural action, much like flipping on a light switch, in order to perform an action or obtain information. The scope of such actions is unconstrained because they are interpreted by applications using the phone as a lens.

4.2 QR code design

An interesting observation we made in our deployments is that complex QR codes are difficult to capture and resolve quickly, especially under poor lighting conditions. The more data you encode on the QR code, the more intricate the pattern that is generated, therefore we tried to minimize the size of the `URL` encoded on the QR code. Figure 4 shows the difference between encoding a long `URL` to a short `URL` shows an example QR code that we used to tag items in one of our deployments. Notice the difference between Figures 4(a) and 4(b). The first QR code encodes 62 characters versus 26 characters for the second one.

Table 4.2 shows the results of some simple scanning experiment between the two tags shown above. We scanned each QR code under light and dark lighting conditions, off the screen of my laptop. Each experiment was run 10 times and the table shows the statistical overview of the results. Clearly, scanning the simple QR code under well-lit conditions performed the best. The complex QR code under the same condition takes about 28-36% longer to scan. On a generic QR code scanner, as used here, there is a portion of the scan time

(a) Long QR Code. (b) Minimized QR Code.

Figure 4. The QR code on the left resolves to the same `URL` at the right one, after resolution and redirection is complete. The short label resolves to `http://tinyurl.com/6235eyw`. The second encodes about half the characters as the first. We used tinyUrl to reduce the QR code image complexity and scan time.

	Average (sec)	Variance (sec)
Short,light	1.66	0.33
Short, dark	2.08	0.35
Long, light	2.26	0.71
Long, dark	2.82	0.50

Table 1. Shows the time to scan a long QR code versus a short QR code in light and dark conditions (loosely defined). Notice that short QR codes scan faster and with less variance that long ones.

that is independent of the code complexity. As these are more heavily used, this is expected to be reduced substantially and the difference is acquisition complexity will be even more pronounced. Perhaps even more important is the variance. Notice that the variance with the simple QR code is much smaller and more stable under either condition. In our experience, **large variance in scan time is a major problem for complex QR codes**. Thus we decided to re-design our codes and push more information in the lookup processes, as network access was more reliable than the focus of the camera on vari-

ous mobile devices. Tags are placed on all types of devices in all kinds of locations with varying degrees of lighting. Simple QR codes are vital for widespread use.

The design choice forced us to examine others that were related. Not being able to encode much information on our QR codes means we are more reliant on the network to provide the bulk of the information, to be very reliable, and to be widespread enough that disconnection is not problematic. Moreover, there are a number of clients that can be used to access and display the information and the tag has to be meaningful for both. In order to meet these criteria we (1) shrunk `URL`'s using tinyURL [7] as a level of indirection and (2) designed two classes of applications: *shallow* applications, and *deep-inspection* applications. Shallow applications interact with the web-application directly while deep-inspection application use the `URL` of the web application to extract a unique identifier and provide deeper inspection and update capabilities of the entity-relationship graph.

An example `URL` we used in our deployment is `http://tinyurl.com/6235eyw`. When this is resolved, we get an empty response in the body, but we use the header to identify the QR code identifier that we associate with the item. The response header looks as follows:

```
HTTP/1.1 301 Moved Permanently
X-Powered-By: PHP/5.3.8
Set-Cookie: tinyUUID=ee81f56c2c15850975b7d175;
expires=Thu, 13-Dec-2012 04:00:18 GMT; path=/; domain=.tinyurl.com
Location: http://streamfs.cs.berkeley.edu/mobile/
mobile.php?qrc=4eb460a39fcd7
X-tiny: db 0.015100002288818
Content-type: text/html
Content-Length: 0
Connection: close
Date: Wed, 14 Dec 2011 04:00:18 GMT
Server: TinyURL/1.6
```

Figure 5. The header of the response from the `tinyUrl` when resolving a QR code. The 'Location' attribute is used to extract the unique identifier for the object this QR code tags. It is also used to redirect users without the phone application to a meaningful web address for the object.

Notice the 'Location' attribute in the header. This is the location of the re-direct. *Shallow* applications use the `URL` directly. The *qrc* `URL` is unqiue identifier for the item that this tag is attached to. A shallow application can obtain mostly read-only service through our web applications. For example, we'll see how to get either item-specific data or item-aggregated data with respect to the user making the request (i.e. the total energy consumed by *my* devices). *Deep-inspection* applications are native to the phone, so we can do much more with the tag. Our energy auditing application allows you to related the item to other items by maintaining state of swipe history. This is more difficult with the web-applicaiton. We can also use the tag and item information to couple it with sensor information coming from sensors on the phone itself. For example, we could determine the direction an object is pointing by using the phone's directional sensor and negating their direction (i.e. phone is facing east, tag on item must be facing west).

5 Applications

In this section we discuss three applications deployed on top of our infrastructure. The first is an energy auditing application. The goal of the application is to quickly collect, organize, and understand plug-loads, also known as miscellaneous electrical loads (MELs), in the building. The application provides scalability of the inventory collection process where there previously was none. Various governmental agencies have conducted field studies of this sort in order to characterize these loads, but they are severely limited by the effort involved in obtaining the inventory[**?**]. And, the inventory quickly becomes stale as items are moved and changed, further frustrating longitudinal studies. Our goal is not just to facilitate the field studies but to make the audit a natural offshoot of using the appliances such that it can be used as part of an organization energy management strategy without expertise in energy auditing. A second-order result of the audit in the infrastructure it leaves deployed. We used the deployment to implement two more applications. The energy item scanner application gives historical energy information for any item that the user swipes. Items include physical objects *and* spaces. Since spaces do not directly consume energy, they embody notion of an aggregate. Similarly, personal energy metering embodies the notion of an aggregate or a family of aggregates. That is our the third application we present.

For each we discuss the fundamental challenges with respect to mobility, consistency management, or aggregation and how our use of gestures and interference were used to address these issues. We also discuss how the underlying entity-relationship graph is used to deal with a dynamic set of constituents when computing aggregates.

5.1 Energy Audit

Nationally, MELs consume about third of the energy in building in the United States [12]. However, given the nature of these electrical loads, collecting enough information quickly from hundreds of disparate loads spread throughout the building simply does not scale. In [20], they describe the major challenge in collecting device inventory for studying MELs and state the the process simply does not scale to large buildings.

The energy audit application addresses this issue. We leverage the occupants, particularly the ones with mobile phones to scale the inventory collection processes. Once they download our application and print a QR code from our site, they attach it to an item and register it using the phone. This scales to any building size, as long as the number of occupants scales with the number of devices and the size of the building.

Execution of the audit builds the entity-relationship graph. Each time a new item or space is registered a

node is added to the graph. The first objects we need to learn about are spaces and how they are inter-related. StreamFS was pre-populated with a directory to record that information. A user can add a new room if either the room has no tag or the current tag is not registered (which they can determine by swiping it). Once a room is added, objects within that room can be added. Figure 7 shows a screen shot of the auditing application. We also applied a classification hierarchy on the objects using a MELs taxonomy [21] incorporated into StreamFS as a directory.

Note, however, that in order to add an object, we need to know where you are. That is the main piece of contextual information that must be deduced or reported. Prior work has used indoor WiFi [8], learning-based approach [13], and alternative infrastructure [22]. However, we took a simpler approach:

1. The person swipes into a space, setting it as their conext for adding new objects or
2. They swipe an, whose location we already know, and we use that to infer their current location.

This approach is much simpler than prior work, but just as effective.

5.2 Inter-relationship capture

There is a special relationship between meters and items. Items are attached to meters, but more importantly, the data collected from the sensor *represents* the underlying dynamics of some physical measurement related to the item. A power meter attached to a television gives us the power profile of that television over some time period. Furthermore, if the meter is removed from the television and attached to another item, that change needs to be recorded, so that we do not attribute the power trace from the second item to the television. There are also items are that attached to each other that can affect how we aggregate feeds. For example, in our deployment, we sometimes connect meters to powerstrips, which have multiple items attached to them. The meter serves as a proxy-aggregate for the attached the power strip that's attached the meter.

Both types of relationship are interpretted by how nodes are symbolically linked. If a meter is the child of an item, it is bound to that item and can be used as a proxy for measurements pertaining to that item. If an item is a child of a meter, it is attached to the meter, but the measure is not taking measurements pertaining to the item, instead if it taking measurement of the children of the item. Figure 6 highlights the two kinds of relationships interpretted by our applications. The one of the left shows a bind relationship while the one of the right shows and attach relationship. To bind or attach the gesture is the same. You first swipe the item the swipe the meter and press a button to either un/attach or un/bind.

Figure 7 shows two screenshot of the auditing interface. The first page is a list of options and the second is the registration page. Once the page is filled out, the

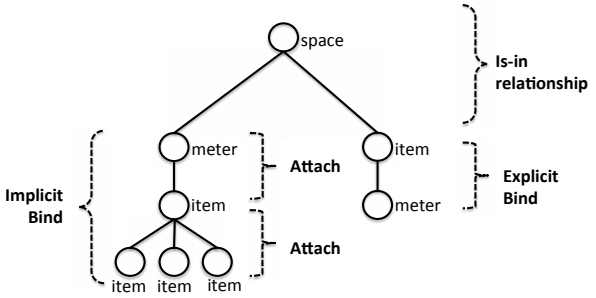


Figure 6. This diagram shows the relationship capture between the objects and locations in the building for the energy audit application. Children of a space node have an "is-in" relationship with the space. An item with another item as a child have a "attached" relationship and meters attached to items are bound to each other. Note, this is a *subset* of the relationship diagrams generated across our three applications.

user scans the QR code once again and the item is added to the inventory. In addition, the QR code tag identifier is added to the 'qrc' directory and symbolically linked to the newly added item in the inventory directory. Finally, a symbolic link is create in the directory for the room that also points to the item just added to the inventory.

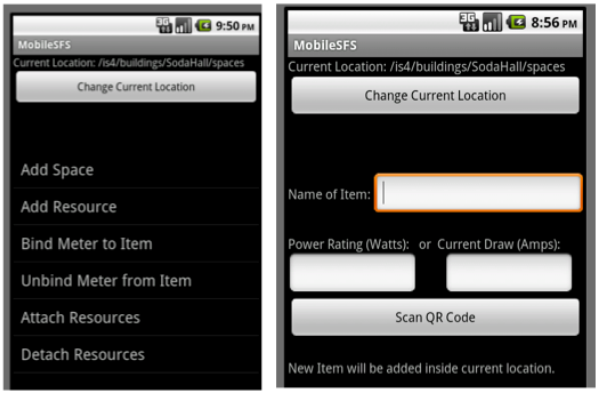


Figure 7.

Auto-classification was done using the same of the item as well. In addition to the spaces, qrc, and inventory directories there is also a taxonomony directory. Items with a given prefix were classified as being items in a particular directory of the taxonomy. A symbolic link was also created from the corresponding node in the taxonomy, that most closely matches the item, to the item node.

In our current deployment we use the link gesture to link an appliance to the wireless power meter that monitors its consumption. In the future, such capability might well be built into the appliance itself. However, other contextual information about the appliance and its relationship to the space and the people that use the space is likely to remain.

5.2.1 *Results*

This application was focused on capturing the physical, energy-consuming objects, describing them in various way and coupling that information with live streaming power data. Figure 8 shows the breakdown of the types of devices that were recorded by the energy auditing application. Since this data was collected from a office building, most of the device were actually LCD screens.

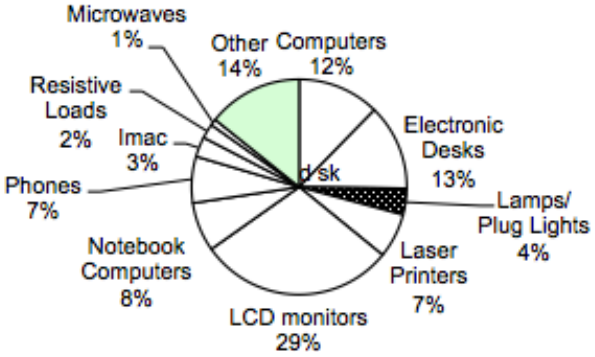


Figure 8. The percentage breakdown of the devices that were captured.

Although most of the items are static, a good number of them are moved by their owners pretty often. For example, 8% of the device were classified as notebooks. When the owner leaves the room or building, they usually take their notebook with them. This kind of information should be recorded using the auditing application. By simply swiping QR code for the notebook and clicking the 'leave' button, the item is kept in the inventory for the building, but the symbolic link from the room to the item is deleted.

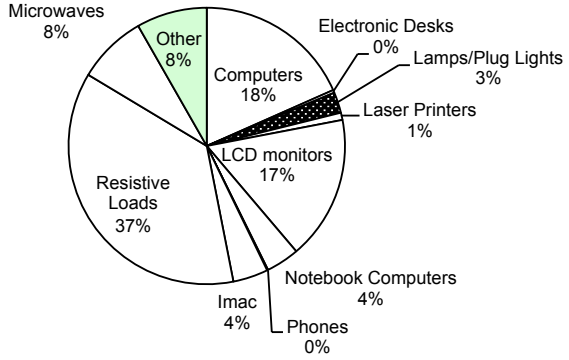


Figure 9. Device power draw in Watts.

Figure 9 shows the power-draw distribution. Interestingly, although the category of devices that fall under 'resistive loads', which includes things like space heaters and toaster ovens, make up only 2% of the items recorded, they account for a much larger fraction of the total power-draw.

Figure 10 was generated using actual live, plugload

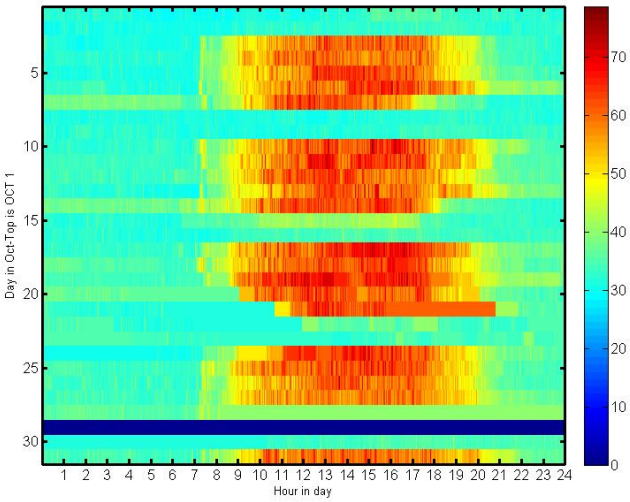


Figure 10. Power heatmap generated from the data obtained from meters attached to plug-load devices throughout the building. Red zones are locations where the highest amount of power is being consumed.

data throughout the day during the entire month of October. The x-axis shows the hour of the day, while the y-axis indicates the day in october. This data was aggregated over hourly period by summing all the plug-load data collected over that hour. Notice the busiest times between 10am and about 6pm. There's also a clear view of the weekends.

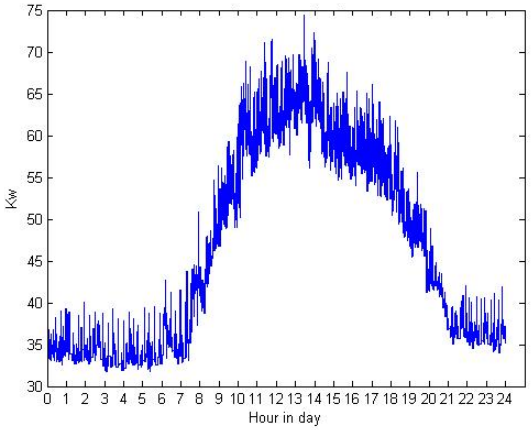


Figure 11. Whole building power draw on October 12th.

Figure 11 shows the whole building power-draw feed. This meter was obtained through the building management system and added to the audit. Notice how the peak and low times correspond to the patterns observed in the monthly plug-load heat map.

Figure 12 shows the load duration curve for each day in the month of October. The load duration curve shows the number of hours in a 24-hour day that the load was

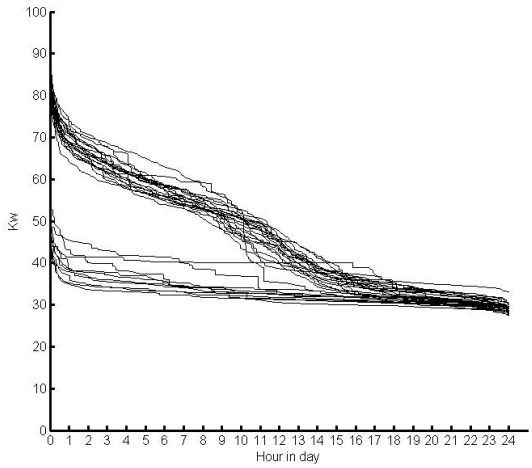


Figure 12. Load duration curves.

at or above the level indicated on the y-axis.

5.2.2 Issues

Although many of the calculations are static and in large aggregates there were some fundamental challenges that we encountered. The first challenge is maintaining *consistency* between the physical world and the virtual representation of it. About 16% of the plug-load items we registered can be moved between locations by they owner with relative frequency. 7% (laptops) move quite often. Maintaing of accurate view of what items are where is a non-trivial with the right mechanisms in place. Our approach in this case is to depend on the ubiquity of smartphones and participants. If a person moves an item from one location to another, they can swipe the item out of the current location and swipe the item back in at the new location. In addition, we take advantage of natural usage patterns. People tend to forget to swipe items out of their current location, so we leverage the second swipe (swipe-in) to imply the first operation (swipe-out). If the item was connected to a meter, we unbind the item from the old meter before swiping the item out.

We are currently working on adjusting the timestamp based on this action as well. From the trace of the meter we can see when the item was unplugged (or turned off). The sudden drop in power reading could be used to mark the point of disconnection. However, this only gets us part of the solution. If the person forgets multiple actions, it become more difficult to determine what has occurred. For example, if a lamp is plugged into a meter in room 1 and they unplug it from the meter, plug another device into it, and move the lamp to another location, the unplug time is not clear. If the new device draws power, we cannot tell the difference between the trace generated from the new device versus the old device. This is a non-intrusive load-trace classification problem and beyond the scope of the current project.

5.3 Item Scanner Application

The next two application were built on top of the deployed infrastructure from the energy audit. The item scanner application shows a power trace of an item. If the item is a space, it shows the aggregate consumption of the space over a 24-hour period. Figure 13 shows a screenshot of a trace for a room in on of the spaces we monitored.

Figure 13. Item scanner screenshot. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

The main thrust of this application revolves around aggregating feeds with respect to the spatial hierarchy that was constructed for the energy audit. A person can scan any item from the building and floor down to the room and individual device.

5.3.1 Issues

In addition to the consistency challenges from the energy audit, apportionment is non-trivial. Even with an accurate view of where items have been moved, tracking the consituents and calculating aggregate is challenging in real-time. Since meter clocks are not syncrhonized, the data must be cleaned before an aggregate can even be computed. We address this problem with dynamic aggregation and discuss the details in section 6.

5.4 Personalized Energy Tracking

A user identifies themselves with a username and password. This create a folder in StreamFS with their username. As they walk around the building, they can tag items that belong to them by swipe them and clicking 'my item' in the mobile application. The application records the item and its location when this is done. A lo-

cation folder is created in the user's directory along with a symbolic link to the item that they tagged. This allows us to aggregate both the totals by location and totals by user. Figure **??** shows the interface for tagging a device and the aggregate for the owner of the items.

5.5 Issues

In this application, the main challenge was in localization of a particular user. In order to form aggregates for the total c onsumed by the user in a particular sapce, we want to present the user with an aggregate of the items they know in that space. To do this automatically one might require indoor localization using the WiFi infrastructure. However, we leverage the tags infrastructure and user input to identify who they are and where they are. With a simple 'login' and swipe, they get personlized, location-specific, aggregate energy consumption data.

- Mobile objects: objects move and are placed in different locations
- Changing meters: meters are moved or items are disconnected and re-connected to other meters
- Aggregation that track constituents over time: just a description, talk more about it in next section

6 Dynamic Aggregation

Dynamic aggregation combines the underlying entity-relationship graph with in-network aggregation. It treats each node in the graph as a potential point of aggregation on a particular data type. For example, if we need to compute aggregates of *KW* data and we declare the node for a particular room as the point of aggregation, we accept data from all children of that node that, whose units are in *KW*, and add the streams together over pre-defined window size or pre-defined timeouts.

The scheme is hiearchical, so a node only accepts data from its children and only sends data to its parent. StreamFS checks for cycles when before node insertion and prevents double-counting errors by only allowing aggregation-points that are roots of a tree that is a subgraph of the entity-relationship graph. In our deployment, each view is a managed as an independent hierarchy. So the hierarchy of *spaces* is separate from the *inventory* hierarchy or the *taxonomy* hierarchy. This allows us to ask questions with a particular view in mind, without conflict, and is a natural fit for our aggregation scheme.

6.1 How it works

Although there are different semantics applied to different node types at the application layer, StreamFS only knows about two types of nodes: (1) default nodes and (2) stream nodes. The main difference is that *default* nodes are not explicitly associated with data coming from a sensor and *stream* nodes are. Furthermore, default nodes can have children, while stream nodes cannot. In our application, meters are represented by default nodes and each stream of data they produce is a stream node.

When an aggregation point is chosen and enabled, dynamic aggregation places a buffer at the node for the type of data that should be aggregated. If we want to aggregate *KW* data, we specify the type and send an enable-aggregation request to the node through an HTTP `POST` to the path for that node. The flow of data starts at the leaves when a stream node received data from a sensor through HTTP `POST`. As data arrives it is immediately forwarded upstream to the parent(s). If a node that receives data from its children is an aggregation point it buffers the data, otherwise it fowards it to its parent.

Ignoring the timeouts for now, lets imagine the parent is a point of aggregation and its buffer is full. At this point the parent separates data into bins for each source and cleans it for aggregation through interpolation. The main operation is to *stretch* and *fill* that data with linearly interpolated values. The *stretch* operation orders all the timestamps in increasing order and for each bin (signal) interpolates the values using the first (last) pair of data points. If there is only a single data point, the stretch uses it as the missing value. The *fill* operation find the nearest timestamps that are less-than and greater-than the missing sampling time, uses their values to determine the equation of a line between them and interpolates the missing value using that equation. Once this is done for each signal, the values are added together for each unqiue timestamp and the aggregated signal is reported to the parent, where the operation occurs recursively to the root. Figure 14 shows an illustration of its operation.

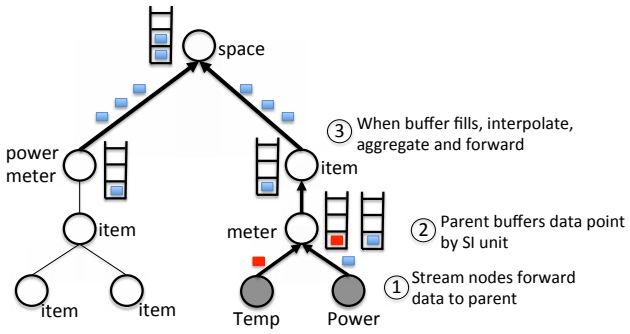


Figure 14. This shows an illustration of the aggregation tree used by *dynamic aggregation*. Data flows from the leaves to the root through user-specified aggregation points. When the local buffer is full the streams are separated by source, interpolated, and summed. The aggregated signal is foward up the tree.

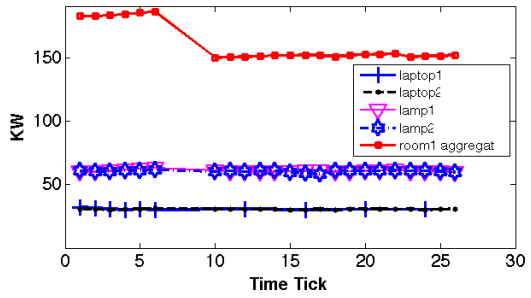
6.1.1 *Dealing with dynamics*

This approach deals with changes in the graph quite naturally. All aggregation point deal only with local data, so a node is only concerned about the children

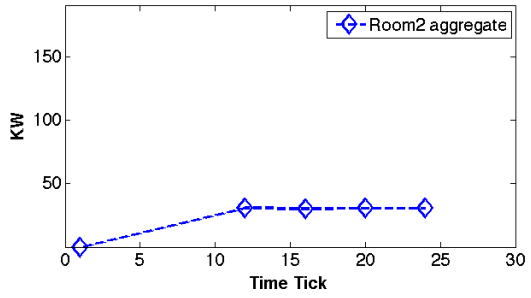
that give it data and the parent to send data to. As objects in the environment move from place to place and these changes are captured, the entity-relationship graph also changes to reflect the move. This change in aggregation constituents is naturally accounted for in the aggregate. If a child is removed, it no longer forwards data to the old parent, therefore the aggregate will reflect that change. Note, however, that changes in the entity-relationship graph are indistinguishable from energy-consuming items that have been turned off. For the purposes of aggregation, that is okay.

6.1.2 *Two scenarios*

We illustrate dynamic aggregation with a common usage scenario. Imagine there are a number of people in a building, each owning a number of plug-load appliances and a laptop. Assume that when a person is in a room their laptop is plugged in and when they leave the room they unplug their laptop and take it with them. People come and go throughout the day, changing the aggregate power consumption of the room and it happens. In addition, some of those people move to other rooms and plug their laptop in the new location. As this happens, we will assume all actions are being recorded in StreamFS.



(a) Room 1 object and aggregate streams.



(b) Room 2 aggregate.

Figure 15. The power consumes by a laptop in *room 1* is shifted to *room 2* a time t=7. Notice the aggregate drops in room 1 while it rises in room 2.

Figure 15 illustrate the aggregation results of that scenario. Notice how...

7 Lessons learned

8 Conclusion

9 Additional Authors

10 References

[1]

[2] http://qrcode.kaywa.com/.

[3] http://www.qrstuff.com/.

[4] http://www.wunderground.com/.

[5] http://streamfs.cs.berkeley.edu/.

[6] http://httpd.apache.org/.

[7] http://tinyurl.com/.

[8] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. pages 775–784, 2000.

[9] D. Caswell and P. Debaty. Creating web representations for places. In *Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing*, HUC '00, pages 114–126, London, UK, 2000. Springer-Verlag.

[10] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. smap: a simple measurement and actuation profile for physical information. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 197–210, New York, NY, USA, 2010. ACM.

[11] *Energy Balances of OECD Countries*. International Energy Agency, http://www.oecd-ilibrary.org/energy/energy-balances-of-oecd-countries-2011_energy_bal_oecd-2011-en, 2011.

[12] *Energy Outlook 2010*. Energy Information Administration, http://www.eia.doe.gov/oiaf/ieo/index.html, 2010.

[13] A. Goswami, L. E. Ortiz, and S. R. Das. Wigem : A learning-based approach for indoor localization. In *CoNext 2011*, 2011.

[14] J.-L. Hainaut and B. L. Charlier. An extensible semantic model of data base and its data language. In *IFIP Congress'74*, pages 1026–1030, 1974.

[15] T. D. Hodes, S. E. Czerwinski, B. Y. Zhao, A. D. Joseph, and R. H. Katz. An architecture for secure wide-area service discovery. *Wirel. Netw.*, 8:213–230, March 2002.

[16] J. Hsu, P. Mohan, X. Jiang, J. Ortiz, S. Shankar, S. Dawson-Haggerty, and D. Culler. Hbci: human-building-computer interaction. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 55–60, New York, NY, USA, 2010. ACM.

[17] X. Jiang, M. V. Ly, J. Taneja, P. Dutta, and D. Culler. Experiences with a high-fidelity wireless building energy auditing network.

[18] T. Kindberg and J. Barton. A web-based nomadic computing system, 2000.

[19] A. Krioukov, L. Dawson-Haggerty, Stephen Lee, O. Rehmane, and D. Culler. A living laboratory study in personalized automated lighting controls. In *Proceedings of the 3rd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '11, New York, NY, USA, 2011. ACM.

[20] S. M. Lanzisera, S. Dawson-Haggerty, X. Jiang, H. Y. Cheung, J. Teneja, J. Lai, J. Ortiz, D. Culler, and R. Brown. Wireless electricity metering of miscellaneous and electronic devices in buildings. In *2011 Future of Instrumentation International Workshop*, 2011.

[21] B. N. Nordman and M. Sanchez. Electronics come of age: A taxonomy for miscellaneous and low power products. *ACEEE Summer Study on Energy Efficiency in Buildings*.

[22] N. B. Priyantha. *The cricket indoor location system*. PhD thesis, Cambridge, MA, USA, 2005. AAI0808861.

[23] H. A. Schmid and J. R. Swenson. On the semantics of the relational data model. In *Proceedings of the 1975 ACM SIGMOD international conference on Management of data*, SIGMOD '75, pages 211–223, New York, NY, USA, 1975. ACM.

[24] P. P. shan Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1:9–36, 1976.

[25] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10:91–102, January 1992.