Systems challenges in real-time, fined-grained energy analytics through mobile phones

1 Introduction

The United States leads the world in per-capita energy consumption; our electricity use has consistently increased over the last 40 years [6]; 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 in 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.

Our work examines the challenges in setting up and deploying a real, whole-building infrastructure to support real-time, fined grained energy analytics and other physical data applications. We use mobile phones to construct an entity-relationship view of the physical world and combine it with streaming, energy-related sensor data in order to provide detailed energy-attribution analytics. In order to enable these, we must first capture the entities in the world and the relationships between them. We limit the scope of the world to a single building domain. We have designed and implemented an application, called the 'Energy Lens', that allows *building occupants* to inform us of things in the world and their inter-relationships, as well as combine the information with streaming meter data. This paper focuses on challenges faced in collecting and maitaing this information.

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-attribution 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 things and locations in the physical world. Once tagged, there are three types of interactions – registration, linking, and scanning – which establish important relationships. Registration is the act of creating a virtual object to represent a physical one. Linking captures the relationship between pairs 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. Internally, we maintain a *entity-relationship graph* of things, people, and locations, that gets updated through these sets of gestures.

In order to connect these components, we rely on having 'ubiquitous' network connectivity. However, in practice, network connectivity can be intermittent and our system must deal with the challenges of intermittency. We discuss how caching and logging are used to address these challenges. Moreoever, when connectivity is re-established, we must deal with updating the entity-relationship graph to reflect the latest state of the world. As such, we must address potential conflicts while replaying logs. Finally, certain physical-state transitions are represented as a set of updates to the entity-relationship graph that must be applied atomically. We use the notion of state-transition transactions, which is also used by our log-replay and transaction manager.

Our 'Energy Lens' system is deployed in a building on our campus. We discuss its archictecture and our design choices. As the deployment is on-going, we evaluate the transaction manager and conflict-resolution algorithms via synthetic traces – capturing conflicts that we expect to occur at scale. We also discuss novel strategies for tracking moving people/things and describe how we implement these in our system. In summary, our work makes the following contributions:

- We design and implement a system that captures and combines physical entities, their inter-relationships, and real-time sensor data in buildings.
- We observe that certain combinations of swipes give us useful information to set the location of

people and things over time. We codify this observation in our *context-tracker* and use it to maintain consistency between the entity-relationship graph and the state of the physical world. To the best of our knowledge, this is radically different from the approaches in standard localization techniques. However, we argue that it can be used to *enhance* their accuracy and overall performance.

- We implement a prefetching algorithm to obtain context-depedent information and both improve performance, as well as enable disconnected operations.
- We design and implement a log-replay and transaction manager to deal with conflicts that arise when multiple users re-establish a connection with the server and wish to execute updates to the entity-relationship graph. We describe how different conflict-resolution policies can be implemented and our rationale for the policies we chose.

In the next sections we describe a motivating scenario for this work, give some background and related work, followed by the system architecture, evaluation, and broader discussion.

2 Motivating scenario

Imagine having the ability to walk through a building and see live, detailed energy-attribution data as you point your phone at various things and locations. As you enter your work office building, you scan its tag and see the live breakdown of energy consumption traces, including HVAC, lighting, and plug-loads. You continue your walk through the building as you head to your office. When you step out of the elevator on your floor you scan the tag for the floor and observe similar figures, only this time they are in relation to that floor alone. Since there are several meeting rooms on that floor, you are curious how much is consumed by occupants versus visitors. You choose to view the total load curve co-plotted with the occupant load curve, specifically for that floor. You see that approximately half the total energy is consumed by visitors during the day. You're curious what portion of those figures can be attributed to you, so you select the personalized attribution option and you see your personal load curve plotted with the total load curve, as well as accompanying statistics, such as the percent of total over time. As you quickly examine the data on your phone, you see that you consumed energy during hours that you were not there. You choose to see a more detailed breakdown. You enter your office, scan the various items and see that your computer did not shut down properly and your light switch was set to manual. You immediately correct these.

Being able to interact with your environment and get a complete energy break-down can provide a useful tool for tracing and correcting rampant energy consumption. In buildings, having the occupants actively participate allows for localized, personal solutions to efficiency management and is crucial to scaling to large buildings. However, providing this detailed level of attribution is challenging. There's lots of data coming from various systems in the building, and integrating them in real time is difficult. Furthermore, attribution is non-trivial. At the centralized systems level, some systems service multiple locations and it is non-trivial to determine the exact break-down by location. At the plug-load level, some plug loads move from place to place throughout the building. For example, we must be able to answer: How much of the total consumed on this floor went to charging laptops? How many of those charging laptops belonged to registered occupants of this floor versus visitors?

Answering the query is relatively trivial once if the information is available, however, collecting the information is non-trivial. Historically, it has been difficult to collect plug-load information. Various studies have used wireless power meters to accomplish just this [5, 11, 14]. All previous work collected the data and performed post-processing to analyze it. We want to take the next natural set of steps: perform processing in real-time and present the occupants with live information. There are several systems challenges that must be overcome in order to achieve this vision. We need to place live metering on plug loads. We need to integrate data from the BMS with plug load and other meter data. We need to be able to approximate the attribution algorithms and tailor them for real-time processing. Most importantly, we need to deal with the systems challenges for tracking which things belong to whom, where people and things are over time, how to deal with the mobile phone as the main interactive modality, and how to do this at the scale of hundred to thousands of users and live meter data. We argue that *without addressing these systems issues, this vision cannot be achieved*. In our work, we start by deploying a network of wireless power meters and use the mobile phone to re-create a model of people, things, and locations in the building. We also use it to assist in tracking of people and things over time.

3 Related work

Our work touches on several areas from smart home research to logistics. In the building space, there has been some interest in building various kinds of energy-related visual and control applications. This work focuses on the object definition, tracking, and management component of the architecture proposed by Hsu et al. [8]. Their work stratefied the set of challenges that one could potentially face if the application were deployed at scale. Our work, in constrast, bases its design rationale on a *real deployment* that is taking place at scale in a building on our campus. We focus on solving fundamental systems challenges in dealing with intermittent connectivity and conflict resolution in tracking people and things over time. We also focus on leveraging gestures to minimize the cost of interaction for users, while maximizing the information we can attain about the state of the world.

An important aspect of the Energy Lens is determining when people and things have moved. This requires some form of indoor localization. There's a large body of literature in the area of indoor localization with mobile phones ranging from using wifi [3], to sonar [15], to ambient noise [17], and a combination of sensors on the phone [2, 12]. Dita [16] uses acoustic localization of mobile phones and also uses the infrastructure to determine gestures in free-space that are classified into predefined control actions. Each of these require relatively complex software and/or infrastrure. We take a radically different, simple approach. We use cheap, easy to re/produce tags (QR codes), place them on things in the environment over incrementally and use the natural *swiping gesture* that users make, when interacting with the Energy Lens application, to track when they have moved or when the objects around them have moved. The working principal is to attain as much information from their gesture to determine when something/one has moved. We discuss our heuristics in section **??**.

ACE [13] uses various sensors on the phone to infer a user's context. The context domain consists of a set of user activities and semantic locations. For example, if ACE can distinguish between `Running, Driving, AtHome, or InOffice`. ACE also infers the one from the others, so if the user is `AtHome` then they are not `InOffice`. Energy Lens uses inference to determine when a person or thing has moved. Certain swipe combinations give us information about whether they moved and where they moved to or whether an item moved and where it moved to. The main difference is that we only infer context when a user is actively swiping, rather than a continuous approach. Pretching is a fundamental technique used in many domains. However, the cost of a prefetch for mobile application outways the benefits if the prefetched data is not useful. Informed mobile pretching [**?**] uses cost-benefit analysis to determine when to prefetch content for the user. In the Energy Lens context, we prefetch data based on their location swipes. We also rely on pretching to anticipate loss of connectivity, not just to improve preformance.

Logistic systems focus on the tracking of objects as the move through distribution sites to warehouses, stores, shelves, and purchase. Items are tracked through bar code or RFID readers. However, the workload is very structured and well defined. The authors of [7] describe this structure and leverage it to minimize storage requirements and optimize query-processing performance. Energy Lens uses the QR codes as the tag and the phone as an active reader. As objects move, users scan those items to their new location. However, objects may belong to one or many people, they can be metered by multiple meters a day, and their history in the system is on-going. In contrast, a typical logistics workload has a start (distribution site) and end point (leaving the store after a sale). In our workload, relationship semantics are important; we need to know whether the meter is *bound-to* rather than simple *attached-to* an item. We discuss the difference later in the paper. Furthermore, we take advatange of natural gestures the user makes with the phone while scanning QR codes to extract information about the current location of the user or things.

The key idea in the HP Cooltown [18, 10] work is to web-enable 'things' in the world, grouped-by 'place', and accessed by 'people' via a standardized acquisition protocol (HTTP) and format (HTML, XML). Cooltown creates a web presence for things in the world either directly (embedded web server) or indirectly (URL-lookup tag) as a web page page that display the services it provides. Many of the core concepts in Cooltown also show up in Energy Lens. The main overlap is the use of tags in the world that contain a reference to a virtual resource, accessible via HTTP through a network connection. Cooltown, however, explicitly chooses not maintain a centralized relationship graph, it leverages the decentralized, linking structure of the web to group associated web pages together. Furthermore, things are assumed to not move. People are the main mobile entities. The kind of applications we wish to support must track where things are and their specific inter-relationships. We imposed a richer set of semantics on our, centrally maintained, relationship graph and use it to provide detailed energy information.

4 System architecture and design

The Energy Lens application aims to approximate the vision described in section 2. For our initial attempt, we tagged items in a building with QR codes and allowed users to 1) register new items and tag them with QR codes they can print from the Energy Lens site, 2) tell us which meters are attached to which items, 3) scan individual item to view their load curve over a 24-hour period.

The architecture consists of three layers: the sensing and tag layer, the data management and processing layer, and the application layer. In this section we discuss each layer and their most important components. Each layer in the architecture is carefully design to work on a real deployment with live users. In deploying the application in a real building, we ran into various issues that informed our design. For example, *QR code reading times vary substantially across phones and lighting conditions*. You must design for the least-common demoninator in terms of camera quality and lighting.

Another aspect to consider is network connectivity. Within our building deployment, connectivity is ubiquitous, connectivity can still be intermittent. Connectivity may be lost for several reasons, including disassociation from an access point due to idleness, dead spots in the building where connectivity to both wifi and 3G/4G are unavailable, multipath-induced destructive interference, and various other reason. Dealing with these throughout the data collection and update phase is especially troublesome. We discuss various mechanisms and algorithms for dealing with disconnected operation.

4.1 Sensing and tag layer

We deployed 20 ACme power meters [9] on a single floor of a building on campus. The data was made available through sMAP [4] and forwarded through our processing and data management layer, StreamFS [1]. We distributed the ACmes throughout a single floor in our building and registered various plug loads as being measured by them. We also tagged hundreds of items and locations throughout the entire building. In addition to tagging the 20 ACmes and the plug-loads they are attached to, we also tagged 351 unmetered items and 139 rooms with QR codes.

4.1.1 QR code design

Our choice to use QR codes is important, since the *only way to scale in deployment size and management complexity is to involve building occupants*. QR codes are cheap to produce. They can be printed and attached to items with tape or sticky paper. Figure 1(b) shows an example QR code used in our deployment. These are placed on physical objects and spaces throughout the building to link between the physical world and our virtual representation of it.

Since occupants are involved, we must make the application easy to use. For QR code usage, varied lighting conditions, phone-specific camera quality, and shaky hand movements can make scanning cumbersome; ultimately de-motivating continued use. QR codes must be designed to minimize scanning time. In our initial deployment and localized, controlled experiments we observed that aside from the occasional long scanning time, variance was also high. The more complex the pixel design in the QR code, the higher the average scanning time and variance. Table 1 shows some of our experimental measurements of scanning times.

We scanned each QR code under light and dark lighting conditions, off the screen of a laptop. Each experiment was run 10 times and the table shows a statistical summary. 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. Perhaps even more important is average scan time is the variance. Notice that the variance with the simple QR code is much smaller and more stable under either condition. QR code image complexity increases with the amount of information you encode on it. Therefore, it was imporant to decrease the amount of information we encoded, *placing the complexity in the lookup rather than the tag.*

This is An example `URL` we used in our deployment:`http://tinyurl.com/6235eyw`. When 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:

Notice the 'Location' attribute in the header. This is the location of the re-direct. This approach gives us flexibility in several ways:

1. It allows us to encode less information in the QR code, descreasing its visual construction, making it more robust across phones with different camera quality, poor lighting conditions, and shaky hands.
2. It allows us to use the added layer of indirection to serve two kinds of applications: the native application where users can *deeply explore* and edit the entities and their relationships or the *shallow lookup*, which re-directs the mobile to informational, read-only view of the item that was scanned.

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

Figure 1. 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.

```
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 2. 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.

4.2 Data management layer

ACme meters are accessible via IPv6 and forward their data through router that run sMAP. sMAP then

forward the incoming data to StreamFS, running in a machine in Amazon's EC2. StreamFS is a web service that organizes streaming data and metadata using a hierarchicaly naming convention. It also manages access to streaming data file via a pub/sub mechanism. We construct a cononical naming convention within StreamFS in order to express the entity relationships between thing, meters, and their associated data. Our application is built on top of StreamFS.

4.2.1 Entity-Relationships

The main aspect we want to capture is entity-relationships between the objects. Entity relationships are captured through naming and interpreted by the Energy Lens application: $/path/to/device\_or\_item$, $/path/to/qrc$, $/path/to/space$, $/path/to/taxonomy$, $/path/to/users$.

We maintain five separate namespaces and we specify a relationship between them through links between these namespaces. The relationship is also set by the type of item the path represents. The types are item, meter, location, system_device, category, and tag. The following relationships we capture are:

- **Owned-by**: When a meter/item/location is tagged as belonging to a user.
- **Bound-to**: When a meter is attached to an item and taking physical measurements associated with that device, we say that the meter is "bound-to" the device.
- **Attached-to**: When a meter/qr code/item is attached to another meter/qr code/item but NOT taking any physical measurements for that item, we say that the meter/item is "attached-to" the other meter/qr code/item. QR codes should not be attached to each other and are not accepted by the EnergyLens application.
- **Is-in**: When a meter/item is inside a location, we say that the meter/item "is-in" that location.
- **Type-of**: When an item is labeled by as a known, specific, type, we say that the item is a "type-of" thing specified by the its label.

The entity-relationship evolves over time, as items are moved from one location to another. Occupants swipe locations and things to give the system information about when things have moved and how things have changed. They also swipe items to tag things as belonging to them.

4.3 Application layer

The application consists of a web application that displays timeseries meter data and aggregate-streams calculated in StreamFS, as well as an inventory and relationship-capture application written for Android-based smart phones. The application interface is relatively simple, consisting of a menu with 4 main options. Swipe gestures manipualte a local portion of the entity-relationship graph – local with respect to a user's current location. Since each location (room, floor) have a QR code attached to them and items are associated with those locations and this relationship is identifyable by name (`/buildings/SDH/spaces/4F/toaster`), we can always deduce the location of the person who is doing the swipe. Figure 3 goes through the various sets of swipes.

The first set is called a 'registration' swipe and we use it to register new items. The user scans QR code and the item it's attached to. This creates an 'attached-to' link between them. Adding, removing, binding, and attaching items is done with a pair of swipes. A lookup is done with by swiping the QR code attached to an item.

4.3.1 Tracking people and things

We have designed a few heuristics for setting the context for an update, that piggybacks on top of the swipe gesture. The following is a list of rules for automatically setting the location of people and things:

- When a user swipes a location, they are presumed to be at that location for fixed period τ.
- If the user swipes anything that's associated with a location l at time $t \leq \tau$, and $l(t) \neq$l(τ), then we set the new location of *thing* they swiped to l(t).
- If the user swipes anything at location l at time $t \geq \tau$, we set the location of the *person* to l(t).
- If a user registers a new location, they are presumed to be at that location.

For each of these, we provide an interactive option to ask for context-change confirmation from the user. So if we think the user/item has moved but they have not, the preset action can be overriden. The guiding principal we follow in your design is to leverage the swipe gesture for as much contextual information as possible. Furthermore, we do not explicitly track users. Context is set on the phone and it represented as the root node in the ERG which you are either editing directly or adding creating/destroying a relationship link in the graph.

Because location swipes give us direct confirmation of a user's location, it can be coupled with wifi localization mechanism and supervised learning to adjust the localization algorithm as the user interacts with their environment. For future work we intend to experiment with this mechanism to enable supervised-learning-based indoor wifi localization.

4.3.2 Disconnected operation

Although connectivity is ubiquitous, network access is not. There are various reasons for this, including dead zones, idle-disconnect and failed hand-off between access points. When encountered in practice, particularly while editing deployment state, it can be quite frustrating and potentially driver users away from using the system. We designed a mechanism that does smart caching, not only to improve performance, but also to allow for disconnected operation.

The Energy Lens application captures a portion of the entity-relationship. When the user enters a new location or swipes the QR code for an item in a new location,

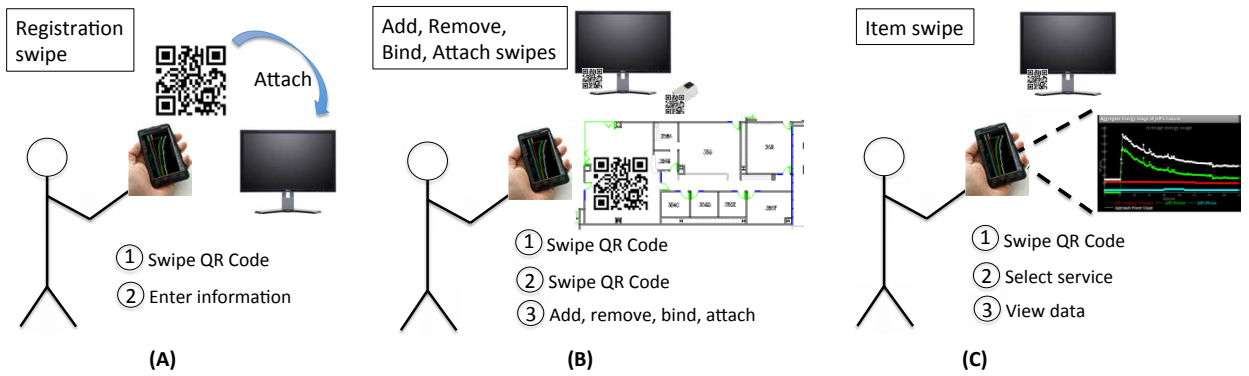

Figure 3. Swiping gestures in the mobile application that

the application fetches the portion of the graph rooted at the floor. Figure 4 shows just a small portion of that entity-relationship graph for the floor we are monitoring in our deployment. A prefetch populates the cache with all entity nodes on that floor, their inter-relationship, associated metadata, and a 30 minutes of data from the entities that represent streams. The full fetch of the 4th floor data set includes 176 nodes and about 1 MB of meter data. In total the app ends up downloading about 1.2 MB of data upon re-connection. The prefetching allows the user to interact with the application as if they were still connected, as long as they remain on the same floor.

Algorithm 1 Prefetch Loop

```
while true do
    if connected and active then
        req ← [t_{i−n}, OP_R]
        resp ← send(req)=[t_{i−k}, OP_R], ..., [t_now, OP_R];
        n >= k
        for j = 1 → size(resp) do
            op ← resp[j] = [t_{i−k+e}, OP_R]
            apply(op)
        end for
    end if
    if active then
        Sleep for 10 minutes
    else
        Sleep for 1 hour
    end if
end while
```

The application also periodically syncs with StreamFS to obtain any changes and keep a consistent view of the entity-relationship graph. Let OP_R be an operation performed on the node rooted at node R and t_i be the current time on the phone. Algorithm 1 shows the pseudocode for the prefetch process. After a set period, the phone sends the server its last performed operation and the time that operation was performed. The server responds with any operations that have take place since then. The client applies those operations internally to the cached version of the the ERG in order to maintain consistency. The *active* checks is for energy savings on the phone. If the phone application is active, the check occurs every 10 minutes, otherwise it occurs ever hour.

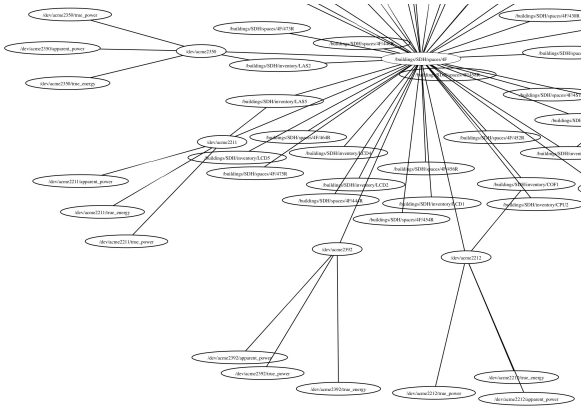

Figure 4. A portion of the prefetched entities on the a single floor in our building. This shows a snapshot of the entity-relationship graph for that floor. Each node, link, and associated content is prefetched when the user swipes the floor tag or anything on that floor.

The entire process is demonstrated in Figure 5. The components shown on the ERG cache and the operation log (OpLog). The figure also show the prefetch process. All reads go to the cache (steps 1 and 2 of the READ process). Writes go through the OpLog (steps 1 to 5 of the WRITE process). The application makes a write request (1), that request is forwarded to StreamFS (2). If StreamFS is reachable and the write is successful (3), the operation is applied to the ERG cache (4) and the success response to sent the application (5). If the operation is not successfuly step 4 is skipped. If StreamFS could not be reached, step 3 is skipped and the operation is written to the OpLog. The OpLog is flushed to the server through the prefetcher. If there are any opera-

tions that were applied to the cache that were not applied on the server, they must be applied to the server.

The OpLog contains records of operations that should be applied to StreamFS to reflect the state of the world. Some of those operations are actually mulitple operations that need to be applied atomically in order to accurately capture deployment state. For example, when a bind or attach operation takes place, we append the timestamp to the item(s) that are being connected, as well as create a link between them in the graph. The application uses both the link and the added metadata to fetch the appropriate graph for display. Therefore, the "add_link" and "update_metadata" operations must be applied atomically or not be applied at all. When the log is dumped, the global transaction manager (GTXM) that sit atop StreamFS, treats transactions accordingly.

4.3.3 Conflict resolution

When the Energy Lens application is started it makes contact with the server and attains the server's local clock time. All transactions in the log are timestamped with a rough approximation of the time the operation would have been committed on the server. When the OpLog is sent to the server, the server orders the operations in ascending order and applies them. A conflict occurs if there is any operation or transaction that was applied after the timestamp of the current operation being processed. When this happens the GTXM does a rollback operation. The rollback operation fetches all the oepration that need to be "undone", generates the negative operation, and applies them to StreamFS. StreamFS has no notion of an operation log and all state is maintained by the GTXM. After the rollback is finished, an attempt is made to apply the pending operation/transaction. If it succeeds, the "undone" operations are re-applied. If it fails, it is placed in a failure log and the undone operations are re-applied. Every successfully applied operation is logged to a success log.

Failure is silent. We make failures silent for two reasons: 1) There is nobody to contact when failure occurs. Mobile users do not often have reversably reachable addresses. 2) Failure assumes the operation was based on a false assumption about the state of the world as captured through the local ERG. However, this does not fully cover all cases. It may be the case that there the state of the world was not reflected in StreamFS or the user's phone and that it was reflected on some else's phone that has not done a log dump yet. We do not currently handle this case, however, we may be able to address it by attempting to re-apply failed operations that are logged in the failure log.

4.4 Real-time analytics

Discussion. What to measure here? Perhaps we discuss the relevant real-time analytics we run? Will there be space? For buildsys, include a half page talking about some of the analytics.

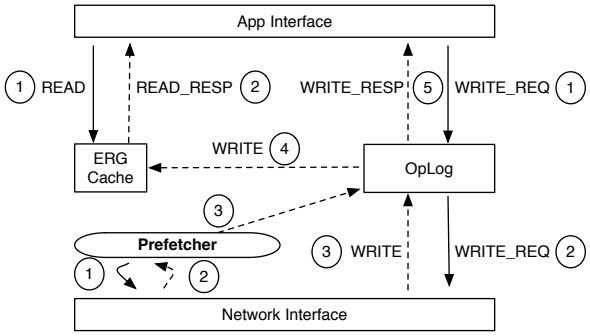

Figure 5. Standard mechanisms for consistency management on the phone. All READ request go to the local cached version of the ERG. All WRITES must go through the OpLog. They are eventually applied to the cache if successful and logged if the StreamFS is unreachable.

No. nodes	Fetch time (sec)	Std. Error (sec)
1	0.8902	0.0756
10	5.7342	1.7087
100	52.3145	14.1146

Table 2. Shows the time to fetch nodes based on the size of the fetch. The fetch time increased linearly with the number of nodes. Caching maintain fetch time near that of fetching a single node. A callback is used when cache is invalidated.

5 Evaluation

In this section we evaluate the pretch download times and discuss strategies for providing scalability. We also look at the transaction manager and evaluate conflict resolution times.

5.1 Prefetching

Prefetching occurs when a user enters a new floors, as detected by a floor scan or an item scan. Table 2 shows that the prefetch times scale linearly with the number of items (and data) to prefetch. Each node holds approximate 100 bytes or information and for a 20-node deployment of power meters, producing 100 bytes of data per stream (3) every 20 seoncds, we fetch approximately 1 MB of data.

These prefetch times are non-trivial to deal with, especially since they cause the phone application to slow down as the data is loaded into the local cache. Although we do not currently have a solution implemented, we are looking to include a callback facility, whereby the prefetch can take place one room at a time, and can be staggard over a longer time frame. This allows user to continue using the application without any frustrating waits.

5.2 Global transaction manager

Maybe we include another table that talks about the trace we ran through and the conflict times, etc.

Conflicts	Res. time (sec)	Std. Error (sec)
1	0.8902	0.0756
10	5.7342	1.7087
100	52.3145	14.1146

Table 3.

- Forwarding
- Conflict resolution

6 References

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

[2] M. Azizyan, I. Constandache, and R. Roy Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, MobiCom '09, pages 261–272, New York, NY, USA, 2009. ACM.

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

[4] 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.

[5] S. Dawson-Haggerty, S. Lanzisera, J. Taneja, R. Brown, and D. Culler. At scale: insights from a large, long-lived appliance energy wsn. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, IPSN '12, pages 37–48, New York, NY, USA, 2012. ACM.

[6] *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.

[7] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive rfid data sets. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 83–, Washington, DC, USA, 2006. IEEE Computer Society.

[8] 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.

[9] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and implementation of a high-fidelity ac metering network. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 253–264, Washington, DC, USA, 2009. IEEE Computer Society.

[10] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, and B. Serra. People, places, things: Web presence for the real world. In *In proceedings WMCSA2000. Available as http://www.cooltown.hp.com/papers/webpres/webpresence.htm*, pages 365–376, 2000.

[11] S. M. Lanzisera, S. Dawson-Haggerty, C. H. Y. Jiang, Xiaofan, J. Taneja, J. Lai, J. Ortiz, D. Culler, and R. Brown. Wireless electricity metering of miscellaneous and electronic devices in buildings.

[12] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell. Darwin phones: the evolution of sensing and inference on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 5–20, New York, NY, USA, 2010. ACM.

[13] S. Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 29–42, New York, NY, USA, 2012. ACM.

[14] J. Ortiz, J. Trage, G. Saldanha, D. E. Culler, and P. K. Wright. Live, continuous energy auditing of miscellaneous electrical loads: Methodology and results.

[15] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. pages 32–43, 2000.

[16] G. Shen, C. Peng, Y. Li, C. Shi, Y. Zhang, and S. Lu. Dita: Enabling gesture-based human-device interaction using mobile phone.

[17] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 155–168, New York, NY, USA, 2011. ACM.

[18] R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison. Bridging physical and virtual worlds with electronic tags, 1999.