E-Sketch: Gathering Large-scale Energy Consumption Data Based on Consumption Patterns

Zhichuan Huang*, Hongyao Luo†, David Skoda†, Ting Zhu* and Yu Gu‡
*University of Maryland, Baltimore County
†Binghamton University, State University of New York
‡IBM Research-Austin

***Abstract*—To reduce peak demand, many utility companies are transitioning from fixed rate pricing plans to real-time pricing plans. To apply real-time pricing plans, it is crucial to collect accurate real-time power consumption readings from individual homes. Thus, utility companies are increasing the installation of smart meters in individual homes. Smart meters can record energy related data (e.g., power consumption) every second. However, power consumption data with high time granularity needs huge data storage space and generates significant communication overhead for utility companies to gather all the data for the pricing plans. In this paper, we present *E-Sketch*, a middleware for utility companies to gather data from smart meters with much less storage and communication overhead. E-Sketch utilizes adaptive sampling to compress power consumption changes in time domain. Then frequency compression is applied to further compress the sampled data. We conducted extensive system evaluations with 30 homes' second-level power consumption data for more than 2 months. Results indicate i) our design can reduce data storage space significantly by 90% with more than 99% accuracy of second-level power consumption on average for a single home, and ii) our design can achieve even more than 99.8% accuracy on average for aggregated power consumption of 30 homes.**

***Keywords*-data compression; energy consumption; smart meter**

I. Introduction

Peak demand of power grids is the main concern for utility companies because it determines how much power utility companies need to generate. Based on the government research report [4], most of the generation and distribution infrastructures are constructed to handle some extremely rare peak demands. In 2010, Energex, a distribution network in Queensland, used 13% of its $8.8 billion infrastructures for only 100 hours of the year [2]. To reduce peak demand, many utility companies intend to introduce real-time pricing plans to encourage homes to reduce high peak demand. Presently, utility companies monitor aggregated peak demand with high time granularity (e.g., every second) [5]. However, at the individual home level, traditional meters at each home monitor energy consumption every hour. Yet, in one hour, the power grid may need notably more power than average hourly power consumption for several minutes. Thus, utility companies are transitioning from hourly pricing plans to real-time pricing plans for better regulation of peak demand. To apply real-time pricing plans, utility companies need to collect power consumption of individual homes at high time granularity.

To collect accurate power consumption information at individual homes, smart meters have been rolled out in many countries [18]. Smart meters deployed at homes usually record power consumption and other energy data every second. However, it requires huge data storage for utility companies to gather power consumption readings every second from all the homes. For example, in our experiments, 127.1TB is needed to store all second-level raw data in one day for homes in New York State (detailed discussed in §II-B). Thus, it is critical to design an architecture to gather data from all the homes with less storage and communication overhead. Besides, power consumption data at different periods of the day is not equally valuable. For example, most appliances run at stable states with relative stable energy consumption when people are not at home. Then the power consumption data at those periods is not as valuable as power consumption when you frequently turn on/off appliances at home. A possible solution for reduction of data storage is to only record appliances' usage to recover power consumption every second. However, it requires servers to know the consumption pattern of all appliances at each home for recovery of power consumption data every second. Moreover, appliances may not consume stable power all the time, which makes it difficult to recover accurate power consumption for pricing plans only with appliance usage data.

In this paper, we study the power consumption patterns of homes to compress second-level power consumption data. A middleware E-Sketch is proposed to work between smart meters at individual homes and central servers in the utility company to reduce data storage of power consumption data. In order to reduce communication overhead, E-Sketch is executed at each local smart meter. However, smart meters have limited storage and computing power, therefore the design should be simple and fast to reduce computation overhead at smart meters. Our design consists of three parts: i) adaptive sampling in time domain; ii) data compression in frequency domain; iii) encoding and decoding. We also show how compressed data can be used to recover power consumption in the central server and analyze the performance of our design. The main contributions of this paper are as follows:

- To our best knowledge, this is the first work to investigate the power consumption at different time granularity. We study the necessity to record power consumption with high time granularity and investigate the relationship be-

tween high and low time granularity power consumption data.

- With the analysis of second-level power consumption data, we propose **E-Sketch** to compress the data in both time domain and frequency domain to reduce data storage and communication overhead. Our design can guarantee that power consumption error is always under the given error bound.
- To validate our design, we evaluate our work extensively with empirical traces of 30 homes' second-level power consumption for more than 2 months. The results show that our design reduces data storage significantly by 90% with more than 99% accuracy of second-level power consumption on average for a single home. For aggregated power consumption of 30 homes, our design can achieve even more than 99.8% accuracy on average by canceling errors from different homes.

The rest of the paper is organized as follows: motivation of paper is introduced in §II; the problem is formulated in §III; the detailed design of E-Sketch for gathering consumption data is described in §IV; implementations and simulations are provided in §V; related work is discussed in §VI; finally, we conclude our paper in §VII.

II. Motivation

In this section, we firstly give an example from our empirical data to show the importance of high time granularity data at a single home and a small community; then we show the problem of gathering and utilizing data with high time granularity.

A. The Need for High Granularity Power Monitoring

In this section, we show that in a single hour there may be several minutes where a home consumes much more power than it does during the rest of given hour. That means hourly power consumption misses a lot of vital information on how homes consume energy. The missing information can be utilized for applying real-time pricing plans. Also, different substations have different capabilities, overlapping of peak demand from individual homes may cause blackouts at some substations within several seconds. Thus, it is important to investigate the relationship of second-level and hourly average power consumption. The second-level and hourly average power consumption of two different homes and aggregated power consumption of 30 homes are shown in Figure 1. For hourly average power consumption, it is defined as the average second-level power consumption within one hour.

• **Observation 1:** *Different homes with similar hourly power consumption may have significantly different second-level power consumption.* As shown in Figure 1, home 1 and home 2 have similar hourly power consumption for two hours. During some periods, second-level power consumption is much larger than hourly average power consumption. However, the detailed consumption for every second of different homes is significantly different. In Figure 1, home 1 has peak demand around $9kW$ for 20 minutes in the first hour and another peak

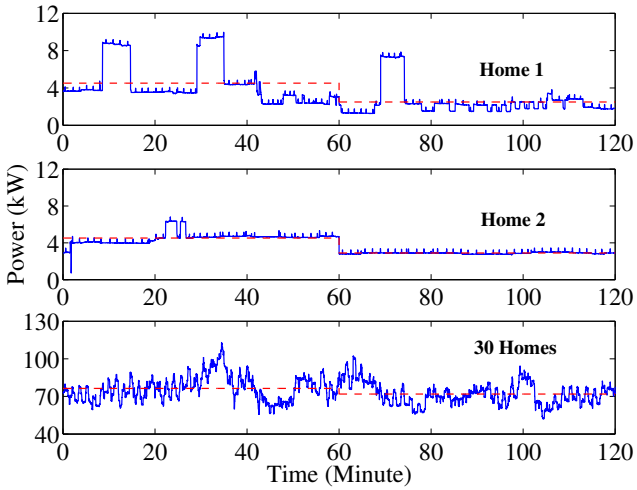

Fig. 1: Second-level and hourly average power consumption of home 1, 2 and aggregated data of 30 homes in two hours (blue solid line is hourly average power consumption; red dashed line is second-level power consumption)

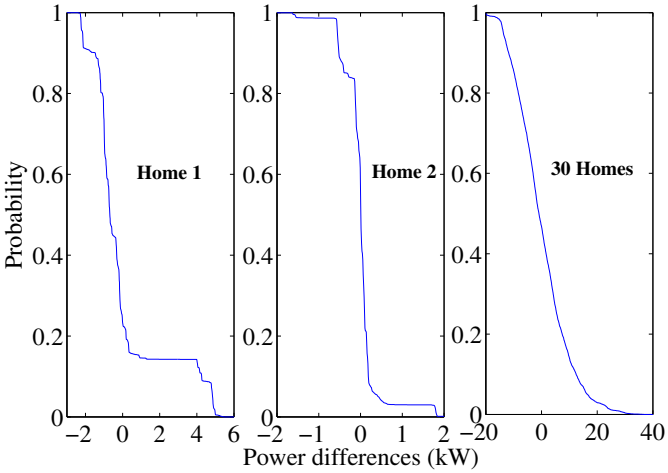

Fig. 2: Distribution of power differences between second-level and hourly consumption for a single home and 30 homes in two hours

demand around $8kW$ for 10 minutes in the next hour. At the mean time, home 2 only has peak demand around $6kW$ in the first hour and almost keeps stable consumption of $3kW$ in the next hour. The detailed distribution of power differences between consumption per second and hour for two homes is shown in Figure 2. For home 1, 10% period in an hour (6 minutes), second-level power consumption is $4.2kW$ larger than hourly average power consumption; around 30 seconds in an hour, second-level power consumption is $5kW$ larger than hourly average consumption. For home 2, 10% period in an hour (6 minutes), second-level power consumption is $0.2kW$ larger than hourly average consumption; around 30 seconds in an hour, second-level power consumption is $1.8kW$ larger than hourly average consumption. From perspective of utility company, home 2 consumes energy more reasonable than home 1. Because even they have similar hourly power consumption, home 2 has much lower second-level peak

demand and should be charged less than home 1.

• **Observation 2:** *Multiple homes can generate very high peak demand for minutes while hourly average power consumption is low.* We also show aggregated power consumption for 30 homes in Figures 1. Second-level power consumption still shows peak demand compared hourly average power consumption. In the first hour, from 30 to 35 minutes, it consumes more than $90kW$ while hourly power consumption is only $75kW$. For distribution of power differences between second-level and hourly average consumption for 30 homes (shown in Figure 2), 10% period in an hour (6 minutes), second-level power consumption is $12.4kW$ larger than hourly average consumption; around 30 seconds of an hour, second-level power consumption is $28.0kW$ larger than hourly average consumption. Thus, compared to a single home, hourly power consumption for a small community (e.g., substation) also misses detailed second-level peak demand. Moreover, peak demand of homes may overlap and create larger peak demand of $28.0kW$. Thus, it is very important to monitor second-level power consumption at individual homes for utility company designing real-time pricing plans.

B. Large Volume of Energy Data

In § II-A, we showed the necessity of recording second-level power consumption at individual homes. However, second-level data requires huge data storage and causes communication overhead. In our simulations, for one single home, 16.4MB is needed to store all second-level data related to energy for one day. The data includes date, total power consumption, power consumption of two phases, voltage of two phases and frequency of two phases. Considering the number of housing units in New York State (8,123,051 in 2012 [3]), 127.1TB is needed to store each day's power consumption data. The amount of historical data that stores for a long period for backup or study of the power grid will increase with time, which becomes a huge cost for utility companies. Furthermore, to utilize the second-level power consumption data for real-time pricing plans, transmission of such large amount of data from smart meters to the respective substations also generates huge communication overhead. Thus, it is important to design a middleware for utility companies to gather data from all the homes with less data storage and communication overhead. Besides, the computing resources at the smart meter are limited, thus the design for gathering data at individual homes should be simple and efficient.

III. Problem Formulation

In this section, we give an overview of E-Sketch, describe how power consumption is collected and present our design goal.

A. Overview of E-Sketch

The real-time data collected at the smart meters needs to be sent to utility company for billing calculation. However, we already show in § II-B that the amount of real-time data is huge, thus it is not possible for smart meters

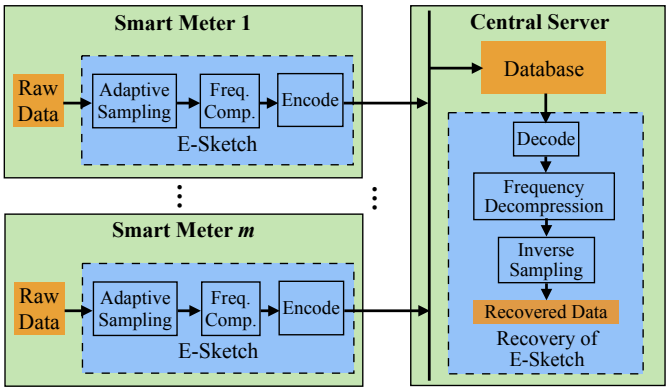

Fig. 3: Overview of system architecture

to send out the raw real-time data to the utility company. Thus, in this paper, we propose a middleware ***E-Sketch*** to work between the smart meters and the central server in the utility company (shown in Figure 3). Each home collects raw real-time power consumption data and then runs E-Sketch middleware to reduce data storage for every window size N, then the central server in the utility company decompressed the data from smart meters for billing calculation. E-Sketch includes three components: i) adaptive sampling that only samples power consumption change that is larger than certain threshold (detailed in § IV-A); ii) frequency compression that compresses adaptive sampled data in frequency domain (detailed in § IV-B); and iii) encoding that encodes compressed data (detailed in § IV-C). The compressed data is then sent to central server for every window.

In the central server, when it receives compressed data from smart meters, it first stores the data in the database directly for persistent storage. Then for every time interval (e.g., day or month), the utility company can recover the compressed data in the database to calculate billing details for consumers. The recovery of the compressed data is the inverse process of E-Sketch. It first decodes the compressed data, utilizes the frequency data to conduct frequency decompression and obtains data in time domain. Then inverse sampling is processed to get recovered data. The detailed algorithm for recovery of E-Sketch is discussed in § IV-D.

B. Design Goal

To record power consumption with high time granularity with less storage, we propose middleware E-Sketch to work between smart meters and central server in the utility companies. Because E-Sketch needs to be run at local smart meters, the computation and implementation complexity of E-Sketch should be low due to limited computation and energy resource in smart meters.To achieve these design goals, we formulate the problem as follows:

Let $\mathbf{P} = \{p(1), \cdots, p(n)\}$ be the original power consumption series. Given the specific boundaries of $e(t)$ ($|e(t)| \leq \theta$), our design goal is to minimize the communication overhead and data storage for storing power consumption data every second in the central server. Assume $\mathbf{Q} = \{q(1), \cdots, q(m)\}$ be

Notations	Definitions
N	Window size of E-Sketch
$p(t)$	Original power consumption of home at t
$\tilde{p}(t)$	Sampled power consumption of home at t
$d(t)$	Power change from t to $t+1$
$d_c(t)$	Compressed high power change from t to $t+1$
$\tilde{d}(t)$	Sampled power change from t to $t+1$
I_c	Index of compressed high power change
$\tilde{I}$	Index of sampled power change
$f(k)$	Frequency value of $\tilde{d}(t)$
$f_q(k)$	Quantization results of $f(k)$
$e_{\tilde{d}}(t)$	Error caused by quantization in time domain
$e_f(k)$	Error caused by quantization in frequency domain
$r(t)$	Recovered power consumption from E-Sketch
θ	Desired error bound
$e(t)$	Power difference between $p(t)$ and $r(t)$

TABLE I: Definitions of notations

the data stored in the central server and $\mathbf{R} = \{r(1), \cdots, r(n)\}$ be the recovered power consumption series in the central server. We can formulate our problem as

$$\begin{aligned} \min \quad & H(\mathbf{Q}) \\ s.t. \quad & g_1 : \mathbf{P} \to \mathbf{Q} && (a) \\ & g_2 : \mathbf{Q} \to \mathbf{R} && (b) \\ & e(t) = |p(t) - r(t)| \leq \theta, t \in [1, n] && (c) \end{aligned}$$

g_1 is the function that maps from $\mathbf{P}$ to $\mathbf{Q}$ and g_2 is the function that maps from $\mathbf{Q}$ to $\mathbf{R}$. $H(\mathbf{Q})$ is the information content of $\mathbf{Q}$. Then our goal is to find the functions g_1 and g_2 to minimize the information content of stored data $\mathbf{Q}$. And our proposed design E-Sketch can be considered as g_1 and recovery algorithm of Sketch can be considered as g_2. Because g_1 is run in smart meters, thus the complexity of g_1 should be relative low. Note that $m = n$ is not required, which means the data points stored in the central server and data points of original power consumption may not be equal.

IV. System Design

In this section, we introduce the main design of E-Sketch. Our design consists of three parts: i) adaptive sampling in time domain; ii) data compression in frequency domain; iii) encoding and decoding. We also show how compressed data can be recovered and analyze the performance of our design.

A. Adaptive Sampling

In this section, we introduce adaptive sampling to reduce data storage while keeping the valuable data of power consumption. To design adaptive sampling, we first analyze the distribution of power consumption. The empirical Cumulative Distribution Function (CDF) of power consumption $p(t)$ and power change $d(t)$ in a window (window size N for the experiment is 2 hours) is shown in Figure 4. And we have:

$$d(t) = p(t+1) - p(t) \tag{1}$$

In Figure 4, though power consumption varies from $1kW$ to $10kW$, power change is mostly very small. For example, 90% of power change is less than $0.05kW$, which provides great opportunity to reduce storage space by only storing the power

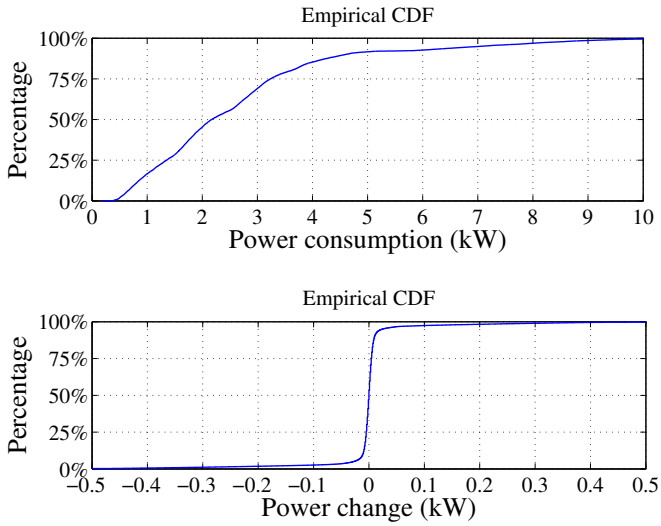

Fig. 4: Empirical CDF of power consumption and power change

consumption change that is valuable. We divide $d(t)$ into $\tilde{d}(t)$ and $\upsilon(t)$. When power consumption change at t is valuable, we have $\tilde{d}(t) = d(t)$; when power consumption change at t is not valuable, we have $\tilde{d}(t) = 0$ to save the storage. $\upsilon(t)$ is 0 when $\tilde{d}(t) = d(t)$, $\upsilon(t) = d(t)$ when $\tilde{d}(t) = 0$. Then we have

$$\mathbf{D} = \tilde{\mathbf{D}} + \mathbf{\Upsilon} \tag{2}$$

D, $\tilde{D}$ and Υ are vectors of $d(t)$, $\tilde{d(t)}$ and $\upsilon(t)$. If the data of $\tilde{D}$ is stored, then we calculate $\tilde{p}(t)$ with $\tilde{p}(t) = \tilde{p}(t-1) + \tilde{d}(t-1)$. Then at t, the error of power consumption data can be calculated as

$$p(t) - \tilde{p}(t) = \sum_{i=1}^{i=t-1} d(i) - \sum_{i=1}^{i=t-1} \tilde{d}(i) = \sum_{i=1}^{i=t-1} \upsilon(i) \tag{3}$$

Assume that $\upsilon(t)$ is a random variable with mean of zero and variance of σ^2. Then the error of power consumption data at t will be a random variable with mean of zero and variance of $s_\upsilon(t-1) \cdot \sigma^2$. $s_\upsilon(t-1)$ is the number of $\upsilon(t) > 0$ for $t \in [1, t-1]$. With the increase of time, the variance of power consumption error increases, which means the error $p(t) - \tilde{p}(t)$ can be very high when t is high. To overcome this problem, we present an adaptive sampling algorithm shown in Algorithm 1.

The basic idea of adaptive sampling is to keep track of power consumption when we decide whether power consumption change at t is valuable or not. Given the error bound θ, we iteratively check the power consumption error at $t+1$ based on $d(t)$ and power consumption error at t if ignoring power consumption change $d(t)$ (Lines 1-3). If power consumption error at $t+1$ is less the error bound θ, then we can ignore the $d(t)$ and keep track of power consumption error at $t+1$ (Lines 4-5). Otherwise, we rewrite $\tilde{d}(t) = err + d(t)$ and power consumption error at $t+1$ will be zero (Lines 6-11). Let t_i be the ith time that $\tilde{d}(t) = err + d(t)$ is rewritten. Because power consumption error is fixed by $\tilde{d}(t)$, we always have $p(t_i + 1) = \tilde{(p)}(t_i + 1)$. Then the maximum error we

Algorithm 1 Adaptive Sampling Algorithm

Input: Power consumption data $p(t)$ and desired error bound θ

Output: Compressed power consumption data.

```
err = 0, count = 0;
for t = 1 to N do
    d(t) = p(t + 1) − p(t);
    if |err + d(t)| < θ then
        err = err + d(t);
    else
        count + +;
        d_c(count) = d(t) + err, I_c(count) = t;
        err = 0;
    end if
end for
```

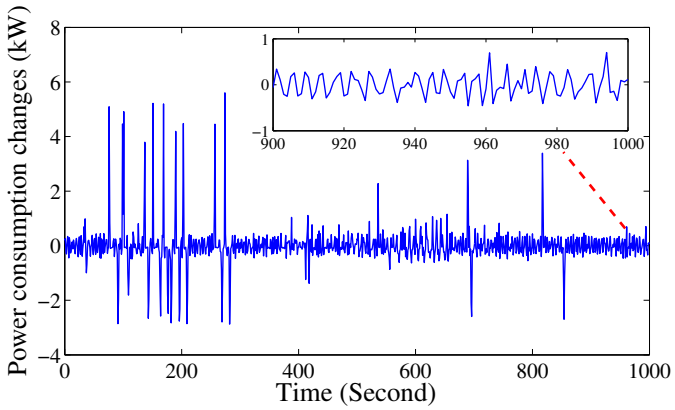

Fig. 5: Results of adaptive sampling

get is at $t = t_1, \cdots, t_i, \cdots$. The power consumption error of $\tilde{p}(t_i)$ can be calculated as

$$p(t_i) - \tilde{p}(t_i) = \sum_{j=t_{i-1}}^{j=t_i} \upsilon(j), \qquad i = 1, 2, \cdots \tag{4}$$

We know the power consumption error is always less than bound θ. We can calculate the mean of error is zero and the variance of error is $E[t_{i+1} - t_i] \cdot \sigma^2$. $E[t_{i+1} - t_i]$ is the expected mean of $t_{i+1} - t_i$. Based on the probability theory, we have

$$E[t_{i+1} - t_i] = P_1 + \sum_{j=2}^{\infty} j * P_j \prod_{k=1}^{k=j-1} (1 - P_k) \tag{5}$$

P_i is the probability that the sum of i $\upsilon(t)$ is larger than θ. With the probability distribution of $\upsilon(t)$, we can calculate P_i based on probability theory. We will not do the detailed calculation since the result is too complicated. But it can be proved that $E[t_{i+1} - t_i] < \frac{\alpha}{(1-\alpha)^2}$ and α is a constant determined by probability distribution of $\upsilon(t)$.

B. Data Compression in Frequency Domain

With adaptive sampling, we can remove most of ignorable power consumption changes. And an example of power consumption changes after adaptive sampling is shown in Figure 5. Most of the power consumption changes are still quite small (less than $0.5kW$) and few of the power consumption changes are more than $4kW$. The detailed power consumption changes from time 900 to 1000 shown in zoom-in figure is frequently fluctuated. This is because that power consumption increase caused by turning on appliances will be followed by power consumption decrease caused by turning off appliances. The highly fluctuated signal is better analyzed in the frequency domain but not time domain. We use Discrete Fourier Transform (DFT) to transfer sampled data from time domain to frequency domain. The discrete time signal $\tilde{d}(t)$ can be equivalently represented by its DFT:

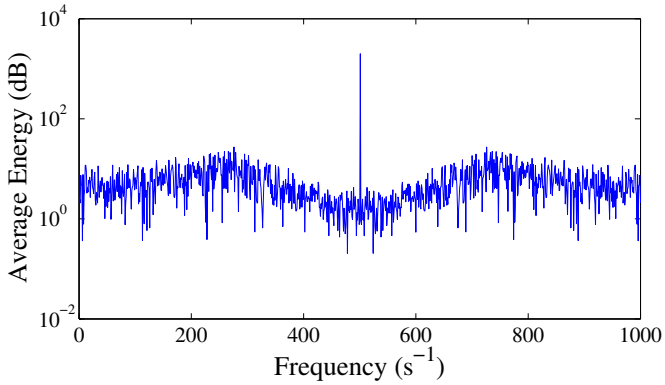

Fig. 6: Spectrum analysis of sampled data over 30 days

$$f(k) = \sum_{t=1}^{N} \tilde{d}(t) \cdot e^{-i2\pi kt/N} \tag{6}$$

For example, Figure 6 shows the frequency spectrum measured over 30 days of power consumption change after adaptive sampling. It is clear that most of the "energy" in the signal is stored in low frequency components. The mechanism to keep the signals in the time domain is through filtering and downsampling. To compress the data from frequency domain, we use Discrete Cosine Transform (DCT) instead of DFT, which outputs real number and is more suitable for data compression. The formula of DCT is as follow:

$$f(k) = \sqrt{\frac{1}{N}}\tilde{d}(1) + \sqrt{\frac{2}{N}} \sum_{t=2}^{N} \tilde{d}(t) \cdot cos\left[\frac{\pi}{N}(t + \frac{1}{2})k\right] \tag{7}$$

To simplify the calculation, we revise the parameters of standard DCT so that DCT matrix is real unitary transform. The elements of the DCT matrix $\mathbf{H} = \{H[k,t]\}$ are

$$H[k,t] = \begin{cases} \sqrt{\frac{1}{N}}, & t = 1, k \in [1, N] \\ \sqrt{\frac{2}{N}} cos\left[\frac{\pi}{N}(t + \frac{1}{2})k\right], & t \in [2, N], k \in [1, N] \end{cases} \tag{8}$$

Let $\mathbf{F}$ be $\{f(1), \cdots, f(N)\}$, then we have $\mathbf{F} = \mathbf{H}\tilde{\mathbf{D}}$. Since the DCT is a real unitary transform, $\mathbf{H}^{-1} = \mathbf{H}^T$ and the inverse DCT (IDCT) is described by $\tilde{\mathbf{D}} = \mathbf{H}^T\mathbf{F}$.

When compressing a signal, the DCT coefficients are typically quantized rather than the actual signal $\tilde{\mathbf{D}}$. The quantized DCT coefficients are denoted as $\mathbf{F}_q$, with $f_q(k) = Q[f(k)]$,

where $Q[\cdot]$ is the quantization operator. Quantization is a non-linear operation that results in a loss of information; only scalar quantization is considered here, where each element of $f(k)$ is quantized individually. Scalar quantization is a many-to-one mapping that transforms intervals of real numbers $[q_i^k, q_{i+1}^k)$ to single real numbers. The superscript "k" accounts for the possibility of different quantization intervals for different frequency coefficients, and the subscript "i" indicates the ith quantization level. Transform coefficients that are in these intervals are typically mapped to the midpoint of the interval, so that $f_q(k) = \frac{1}{2}(q_i^k + q_{i+1}^k)$, for $q_i^k \leq f(k) \leq q_{i+1}^k$.

The recovered signal $\tilde{\mathbf{D}}_q$ is obtained by performing the IDCT on the quantized frequency values, $\tilde{\mathbf{D}}_q = \mathbf{H}^T\mathbf{F}_q$. Two quantities of interest in this paper are the quantization errors in both the spatial and the frequency domains. Spatial-domain error is represented by $\mathbf{e}_{\tilde{d}} = \tilde{\mathbf{D}}_q - \tilde{\mathbf{D}}$, and frequency-domain error by $\mathbf{e}_f = \mathbf{F}_q - \mathbf{F}$. Note that the quantization error in the spatial domain can be expressed as

$$\mathbf{e}_{\tilde{d}} = \mathbf{H}^T\mathbf{e}_f = \sum_{k=1}^{N} h_k(f_q(k) - f(k)) \tag{9}$$

To compress the signal, we need to carefully select q_i^k to maximize the compression ratio while fulfilling the error bound θ. Different from traditional DCT which applies in image data compression, the high frequency component of the signal we need to compress is not ignorable and the error should be within the error bound θ. Thus, we present a frequency compression algorithm to decide quantization results of frequency data $f(k)$ that maximize the compression ratio and fulfill the error bound. The basic idea is to decide quantization results based on transferring quantization errors in frequency domain to time domain. The detailed description is in Algorithm 2. We first calculate $f(k)$ based on our defined DCT Equation (7) (Line 1). Then $f(k)$ is sorted in ascending order and initialize $f_q(k)$ and current error bound $b(t)$ (Line 2). For each $f(k)$, we first check when $f(k)$ is ignored whether the error caused by $f(k)$ still less than current error bound $b(t)$. If $|e_{\tilde{d}}(t)| < b(t)$, we can ignore $f(k)$ and set $f_q(k) = 0$ and update $b(t)$ (Lines 4-8). Otherwise, we divide $f(k)$ by λ, and check the error bound again until we find the maximum $f_s(i)$ that fulfill the error bound (Lines 10-15). Then we can update $f_q(k)$ and current error bound $b(t)$ (Lines 16-19).

To analyze the performance of our frequency compression algorithm, we calculate the variance of error after quantization. Because the quantization process is symmetry, the DCT-domain quantization errors are uncorrelated random variables. The covariance matrix $\mathbf{K}_{e_f} = E(\mathbf{F}_q - \mathbf{F})(\mathbf{F}_q - \mathbf{F})^T$ is then diagonal, with its N non-zero elements equal to the quantization noise of the individual frequency-domain coefficients, $\sigma^2_{e_f}[k]$. With $\mathbf{K}_{e_f}$, we can calculate the covariance of the error in time domain as

$$\mathbf{K}_{e_{\tilde{d}}} = E(\tilde{\mathbf{D}}_q - \tilde{\mathbf{D}})(\tilde{\mathbf{D}}_q - \tilde{\mathbf{D}})^T = \mathbf{H}^T\mathbf{K}_{e_f}\mathbf{H} \tag{10}$$

Equation 10 includes information about the correlation of the time domain error sequence, but another quantity of interest is the variance of the individual time domain errors. The variance of $e_{\tilde{d}}(t)$ is found as $\sigma^2_{e_{\tilde{d}}}[t] = \mathbf{K}_{e_{\tilde{d}}}[t,t]$, or in summation notation

$$\sigma^2_{e_{\tilde{d}}} = \sum_{k=1}^{N} H^2[k,n]K_{e_f}[k,k] = \sum_{k=1}^{N} H^2[k,n]\sigma^2_{e_f} \tag{11}$$

Algorithm 2 Frequency Compression Algorithm

Input: Power consumption changes after adaptive sampling $\tilde{d}(t)$ and desired error bound θ

Output: Quantization results $f_q(k)$ of frequency data $f(k)$ for $k \in [1, N]$.

1: Calculate $f(k)$ with $\tilde{d}(t)$ based on Equation (7);
2: Sort $f(k)$ in ascending order;
3: $b(t) = \theta$ for $t \in [1, N]$, $f_q(k) = -1$ for $k \in [1, N]$;
4: **for** i = 1 to N **do**
5: $\quad e_{\tilde{d}}(t) = \sum_{k=1}^{k=N} h_k(t) * (-f(i))$;
6: $\quad$ **if** $|e_{\tilde{d}}(t)| < b(t)$ for $t \in [1, N]$ **then**
7: $\quad\quad f_q(i) = 0$;
8: $\quad\quad b(t) = b(t) - e_{\tilde{d}}(t)$;
9: $\quad$ **else**
10: $\quad\quad f_s(i) = f(i)$;
11: $\quad\quad e_{\tilde{d}}(t) = \sum_{k=1}^{k=N} h_k(t) * (f_s(i)/\lambda)$;
12: $\quad\quad$ **while** $|e_{\tilde{d}}(t)| > b(t)$ for $t \in [1, N]$ **do**
13: $\quad\quad\quad f_s(i) = f_s(i)/\lambda$;
14: $\quad\quad\quad e_{\tilde{d}}(t) = \sum_{k=1}^{k=N} h_k(t) * (f_s(i)/\lambda)$;
15: $\quad\quad$ **end while**
16: $\quad\quad f_q(k) = f_(i) - f_s(i)$;
17: $\quad\quad b(t) = b(t) - e_{\tilde{d}}(t)$;
18: $\quad$ **end if**
19: **end for**

C. Encoding

After frequency compression, we can further reduce data storage by encoding the compressed data. Because the range of compressed data is much less than the range of original power consumption, the storage space for compressed data can be reduced.

The first step of encoding is to generate the probability distribution of compressed data. Then we can encode compressed data based on their probability to minimize the data storage. The encoding works by creating a binary tree of nodes. These nodes can be stored in a regular array, the size of which depends on the number of symbols. A node can be either a leaf node or an internal node. Initially, all nodes are leaf nodes, which contain the symbol itself, the weight (frequency of appearance) of the symbol and optionally, a link to a parent node which makes it easy to read the code (in reverse) starting from a leaf node. For example, with number of n values $[v_1, \cdots, v_n]$, we can calculate their probability to generate n nodes $[(v_1, pr_1), \cdots, (v_n, pr_n)]$. Internal nodes contain symbol weight, links to two child nodes and the optional link to a parent node. As a common convention, bit

Algorithm 3 Recovery of Power Consumption

Input: $p(1)$, N, $\mathbf{I}_c$, $\mathbf{D}_c$,$\tilde{\mathbf{I}}_c$ and $\mathbf{F}_q$
Output: Recovered power consumption $r(t)$

1: $r(1) = p(1)$;
2: $index_s = 2$, $index\_e = 2$;
3: **for** i = 1 to $|\mathbf{I}_c|$ **do**
4: $index\_e = I_c(i)$;
5: **for** t = $index\_s$ to $index\_e$ **do**
6: $r(t) = r(t-1) + d_c(i)$;
7: **end for**
8: **end for**
9: **for** t = 1 to N **do**
10: Calculate $\tilde{d}(t)$ with $f_q(k)$ and N based on IDCT;
11: **end for**
12: $index_s = 2$, $index\_e = 2$;
13: **for** i = 1 to $|\tilde{\mathbf{I}}_c|$ **do**
14: $index\_e = \tilde{I}_c(i)$;
15: **for** t = $index\_s$ to $index\_e$ **do**
16: $r(t) = r(t-1) + \tilde{d}(i)$;
17: **end for**
18: **end for**

'0' represents following the left child and bit '1' represents following the right child. The process essentially begins with the leaf nodes containing the probabilities of the symbol they represent, then a new node whose children are the two nodes with smallest probability is created, such that the new node's probability is equal to the sum of the children's probability Assume (v_1, pr_1) and (v_2, pr_2) are merged to (v_{12}, pr_{12}), then we have $pr_{12} = pr_1 + pr_2$. With the previous two nodes merged into one node (thus not considering them anymore), and with the new node being now considered, the procedure is repeated until only one node remains. Finally, based on the constructed tree, we can assign each value a code to be stored.

D. Recovery of Power Consumption

In previous sections, we present *E-Sketch* to compress power consumption with high time granularity. In this section, we show how the compressed data can be used to recover original power consumption.

Since we encode compressed data, the first step of recovery of original data is to decode by simply translating the stream of prefix codes to individual byte values. Then we have initial power consumption $p(1)$, length of power consumption data N, indexes and values of high power consumption changes $\mathbf{I}_c$ and $\mathbf{D}_c$, and indexes and frequency values of power consumption changes $\tilde{\mathbf{I}}_c$ and $\mathbf{F}_q$. Then we show how power consumption can be recovered with above data in Algorithm 3. First, we can recover the power consumption data with the indexes and values of high power consumption changes (Lines 1-8). Because we store the power consumption changes but not original power consumption, we need to calculate original power consumption based on the compressed data. Then we recover time domain data of power consumption change with quantization results in frequency domain based on IDCT (Lines 9-11). The recovered power consumption changes $\tilde{\mathbf{D}}_q$ can be obtained by performing IDCT on quantized frequency values $f_q(k)$, $\tilde{\mathbf{D}}_q = \mathbf{H}^T\mathbf{F}_q$. With the indexes and time domain values $\tilde{\mathbf{I}}_c$ and $\tilde{\mathbf{D}}$, we can also recover the power consumption data from 1 to N (Lines 12-18). Algorithm 3 can recover the original power consumption from 1 to N. If we only need power consumption of a specific time, we can also calculate the original power consumption as follows:

$$\tilde{p}(t) = p(1) + \sum_{i=1}^{I_c(i)<t} d_c(i) + \sum_{i=1}^{\tilde{I}(i)<t} \sum_{j=1}^{j=N} H[i,j] f_q(j) \quad (12)$$

Then we can reduce the computation overhead of central server and obtain original power consumption of a specific time faster.

E. Time Complexity and Storage Analysis

First, we analyze the complexity and data storage of E-Sketch at local smart meters in the following three stages.
i) *Adaptive sampling*. Basic power consumption curve is sketched in time domain. In adaptive sampling, Algorithm 1 is applied to generate sampled power consumption changes and time complexity of Algorithm 1 is $O(N)$. For window size N, we need to store the high power consumption changes and indexes in this window. Thus the storage cost should be related to the number of high power consumption changes N_h.
ii) *Frequency compression*. Sampled data of power consumption changes are transferred into frequency domain and quantize frequency data. In frequency compression, we need to conduct DCT for $\tilde{d}(t)$ and run Algorithm 2. The time complexity for DCT is $O(NlogN)$ with fast Fourier transform (FFT) and time complexity for Algorithm 2 is $O(N)$. For output of frequency compression, we need to store $f_q(k)$ and $\tilde{I}$, which is related to the number of sampled consumption changes N_s.
iii) *Encoding*. The compressed data in adaptive sampling and frequency compression is encoded to further reduce storage space. Time complexity of encoding is $O(NlogN)$ and encoding introduces no storage cost.

In total, the time complexity of our design is still $O(NlogN)$, which means our design is simple for smart meters. And the total data storage cost will be related to N_h and N_s.

Even after data compression, the amount of data in the central server is still a huge number because the number of homes is large. Thus, we also analyze the time complexity of decompression in the central server. The decompression algorithm includes IDCT calculation and two loops with complexity of $O(N)$. Thus the complexity of decompression is also $O(NlogN)$.

V. Implementation and Evaluation

In this section, we evaluate the performance of our proposed design. We deploy eGauge power meters at individual homes to collect the energy consumption related data (e.g., power,

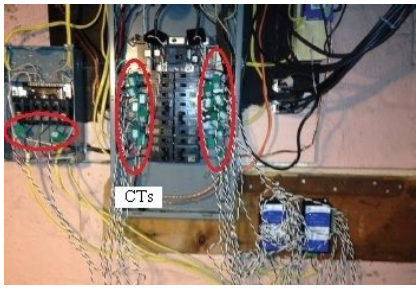

Fig. 7: eGauge deployment in a home

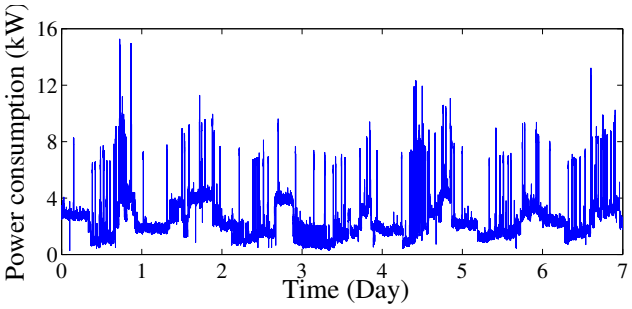

Fig. 8: Power consumption of a home in seven days

voltage, frequency, etc.) every second [1]. One of the experiment setup is shown in Figure 7. We add current transducers (CTs) around each leg of the home's split-phase power input from the grid to monitor all the circuits inside the home every second. In our simulation, we use the power consumption traces that we collected from 30 homes in eGauge website for two months. To make the figure easy to follow, we only show the power consumption of a home for seven days in Figure 8. In a day, the power consumption is mostly in the afternoon and evening. And the power consumption for different days varies significantly.

A. Evaluation Baseline and Metrics

Baseline. To verify the efficiency of our approach, we compare our design with three approaches: i) **Sparse Data**, which utilizes lower time granularity power consumption to estimate high time granularity power consumption; ii) **Polynomial Fitting**, which applies polynomial fitting to estimate the power consumption; iii) **Zip**. We realize deflate algorithm, which is most commonly used compression method for zip files [15].

Metrics. We use two metrics to evaluate the performance of our approach: i) **storage space**: the amount of storage space used to store data; ii) **average error of power consumption**: average power differences between original and recovered data.

B. Basic Evaluation Results

We evaluate the effectiveness of our proposed E-Sketch compression algorithm, which includes the storage space, error of power consumption for a single home and a community of 30 homes. All results are simulated with the two months empirical data of energy consumption. And the selection of parameters are: i) error bound $\theta = 0.05$; ii) window size

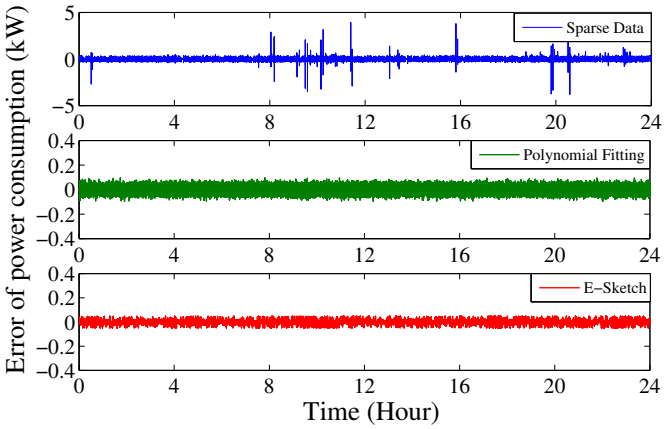

Fig. 9: Error of power consumption of different approaches compared to original power of a single home

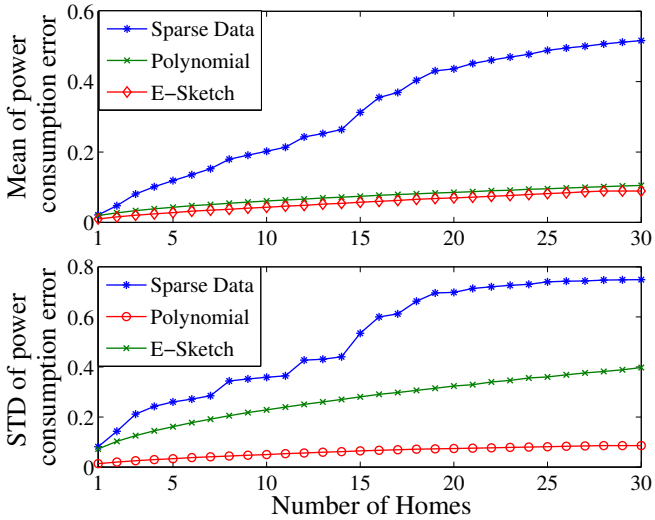

Fig. 10: Error of power consumption of different approaches compared to original aggregated power of 30 homes

$N = 3600s$. The impacts of these parameters are investigated in latter sections.

Error of power consumption for a single home. The errors of two approaches over time are shown in Figure 9. Our proposed E-Sketch can recover data with at most $0.05kW$ power error. The average power error of E-Sketch is $0.02kW$. As shown in Figure 1, the home consumes more than $3kW$ power at most time; thus E-Sketch recovers data with more than 99% accuracy. We also show the power errors of sparse data and polynomial fitting. We use average power consumption data for every 5 seconds to recover power consumption for every second. The average errors of power consumption for sparse data and polynomial fitting is also $0.02kW$. However, at some time, the power errors of sparse data and original data is more than $2kW$. And for polynomial fitting, the maximum error is around $0.1kW$, which is much better than sparse data but worse than E-Sketch. Because Zip is lossless compression, there would be no error of power consumption.

Error of power consumption for 30 homes. The high time granularity power consumption data of each home is sent to the utility companies for calculating the price of electricity. To

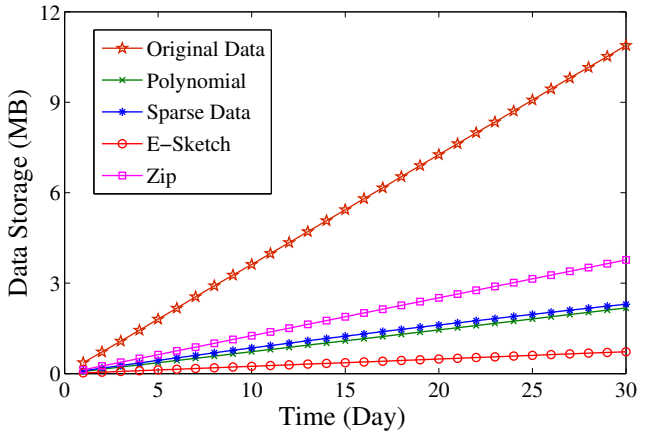

Fig. 11: Storage space of different approaches

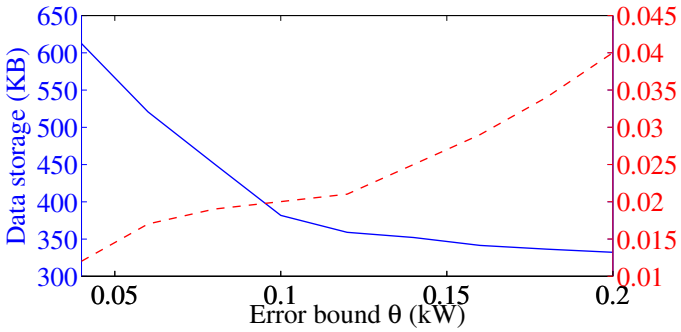

Fig. 12: Impact of error bound θ

further verify the performance of E-Sketch, we apply E-Sketch for power consumption data in 30 homes. The power errors of aggregated power consumption for 30 homes are shown in Figure 10. Compared to power errors for a single home, the average power error for 30 homes increases from $0.02kW$ to $0.1kW$. While for sparse data, average power error becomes $0.52kW$. The average power error for polynomial fitting is close to E-Sketch, however, the standard deviation (STD) of polynomial is much higher than E-Sketch. And maximum error for polynomial fitting can reach $2kW$ while maximum error for E-Sketch is only $0.3kW$. The reason that E-Sketch performs better than sparse data is E-Sketch randomizes the errors, then power errors of different homes can be cancelled.

Data storage. We show the storage space of energy consumption data for 30 days in Figure 11. Original data needs much more storage space than all the compression techniques. Because Zip is a lossless compression technique for general data, it is not optimized for power consumption data, thus it still costs more than $3MB$. The sparse data and polynomial fitting takes advantage of power consumption pattern, thus it costs around $2.2MB$. E-Sketch is the best approach for data storage, which only costs less than $1MB$ data storage.

C. Advanced Evaluation Results

In the basic evaluation results, we show E-Sketch works well for a single home and 30 homes. In this section, we investigate the impact of different parameters in E-Sketch to verify the robustness of our design.

Impact of error bound θ. The results of tradeoff of θ between storage and accuracy are shown in Figure 12. In this simulation, the setting of window size N is 1 hour. When θ increases, the storage space increases linearly and average power error decreases slowly. With higher error bound θ, adaptive sampling can sample less data and frequency compression can take advantage of more aggressive quantization, thus the data storage space decreases significantly.

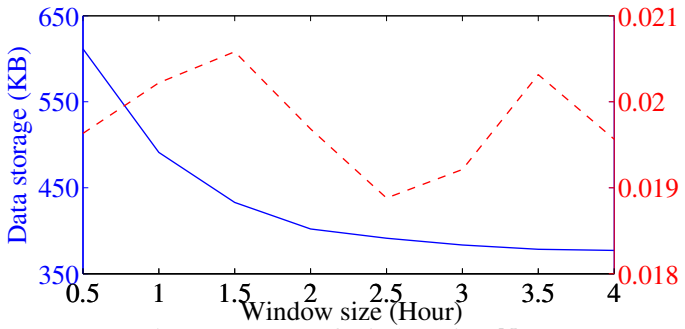

Fig. 13: Impact of window size N

Impact of window size N. We show the impact of window size N in Figure 13. In this simulation, the setting of θ is $0.05kW$. With shorter period of window size N, the adaptive sampling samples less data for a window. However, it makes frequency compression harder to compress the sampled data, which introduces more data storage. Thus, data storage decreases with larger window size. For the average power error, the adaptive sampling and frequency compression still works to keep errors under error bound, thus average power error almost stay the same.

VI. Related Work

Our work is related to three areas of previous work: peak demand and pricing models and energy data management.

Peak Demand. There are many works on modifying the elastic load components of common household appliances to reduce peak demand [17]. In [16], a novel demand response mechanism is proposed to exploits appliance elasticity to decrease peak loads, a fuzzy-logic based controller for appliances and a signal generator for the utility are designed to reduce the power consumed by appliances with elastic components. A real-time distributed deferrable load control algorithm is proposed to reduce the variance of aggregate load by shifting the power consumption of deferrable loads to periods with high renewable generation [9]. Batteries are deployed at homes to supply energy when peak demand and store energy when energy consumption is low [14]. Renewable energy is also investigated to reduce the peak demand [13] [27]. In [28], a secure energy routing mechanism for sharing renewable energy in smart microgrid is developed to reduce the peak demand. In [26], energy sharing is introduced to balance the mismatch of renewable energy generation and power consumption in buildings.

Pricing Models. In [6], Adepetu et al. analyze the hourly aggregate load data to study whether the choice of TOU parameters adequately reflects the aggregate load, and to study whether TOU pricing has actually resulted in a decrease in the mean peak-to-average ratio. In [8], Corradi et al. propose models for the dynamics of changed price response, and shows how

these dynamics can be used to control electricity consumption using a one-way price signal. In [24], a decentralized optimal load control mechanism is proposed to provide contingency reserve in the presence of sudden demand-supply mismatch. In [25], Zhong et al. develop energy sharing pricing model to incentivize energy sharing in a microgrid. In [10], a general microgrid energy sharing system architecture is designed to reduce electricity cost of houses under different TOU pricing models.

Energy Data Management. Energy data is widely used to reduce and schedule energy generation and consumption [21] [20]. In [7], Chen et al. explore energy consumption in everyday home environments to study the relationship between behavioral patterns and energy consumption and investigates how this relationship can be helpful for people to act in a more energy-efficient manner. In [19], Xiang et al. design a novel data aggregation scheme that exploits compressed sensing (CS) to achieve both recovery fidelity and energy efficiency in WSNs with arbitrary topology. In [23], a system framework of data reduction is proposed to minimize energy consumption in wireless sensor network. There are also works on energy data collection. In [22], a new and severe denial of service attack is proposed for data collection in AMI network. In [12], Time-Log Tree (TL-Tree), a novel indexing structure is proposed to consider time-series as a primary characteristic for optimizing both memory and energy constraints. In [11], Zhang et al. study timely, cost-minimizing upload of massive, dynamically-generated, geo-dispersed data into the cloud, for processing using a MapReduce-like framework.

Different from previous work that focus on data collection and aggregation, we propose to use data compression to process collected energy data. By learning from pattern of energy consumption data, we present the design to save storage space while maintain accurate data.

VII. Conclusion

In this paper, we present E-Sketch to save the space and keep high accuracy of energy data. In order to reduce communication overhead, E-Sketch is executed at each local smart meter. However, smart meters have limited storage and computing power, therefore the design should be simple and fast to reduce computation overhead at smart meter. Our design consists of three parts: i) adaptive sampling in time domain; ii) data compression in frequency domain; iii) encoding and decoding. We also illustrate how compressed data can be used to recover power consumption in the central server and analyze the performance of our design.

We conducted extensive system evaluations with 30 homes' second-level power consumption data for more than 2 months. Results indicate i) our design can reduce data storage space significantly by 90% with more than 99% accuracy of second-level power consumption on average for a single home, and ii) our design can achieve even more than 99.8% accuracy on average for aggregated power consumption of 30 homes.

VIII. Acknowledgments

This work was supported by NSF CNS-1217791.

References

[1] Egauge. http://www.egauge.net.
[2] Energex. https://www.energex.com.au.
[3] United States Census Bureau. http://www.census.gov.
[4] Ontario's Power Authority. Ontario's long term energy plan. 2010.
[5] A Smart Grid Strategy for Assuring Reliability of the Western Grid . U.S. Department of Energy. 2010.
[6] A. Adepetu, E. Rezaei, D. Lizotte, and S. Keshav. Critiquing time-of-use pricing in ontario. In *IEEE International Conference on Smart Grid Communications*, 2013.
[7] C. Chen, D. J. Cook, and A. S. Crandall. The user side of sustainability: Modeling behavior and energy usage in the home. *Pervasive and Mobile Computing*, 9(1):161 – 175, 2013.
[8] O. Corradi, H. Ochsenfeld, H. Madsen, and P. Pinson. Controlling electricity consumption by forecasting its response to varying prices. *IEEE Transactions on Power Systems*, 28(1):421–429, 2013.
[9] L. Gan, A. Wierman, U. Topcu, N. Chen, and S. H. Low. Real-time deferrable load control: Handling the uncertainties of renewable generation. In *e-Energy*, 2013.
[10] Z. Huang, T. Zhu, Y. Gu, D. Irwin, A. Mishra, and P. Shenoy. Minimizing electricity costs by sharing energy in sustainable microgrids. In *Buildsys*, 2014.
[11] C. W. L. Zhang, Z. Li and M. Chen. Online algorithms for uploading deferrable big data to the cloud. In *INFOCOM*, 2014.
[12] H. Li, D. Liang, L. Xie, G. Zhang, and K. Ramamritham. Tl-tree: Flash-optimized storage for time-series sensing data on sensor platforms. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012.
[13] A. Mishra, D. Irwin, P. Shenoy, J. Kurose, and T. Zhu. Greencharge: Managing renewable energy in smart buildings. *IEEE Journal on Selected Areas in Communications*, 31(7):1281–1293, 2013.
[14] A. Mishra, D. Irwin, P. Shenoy, and T. Zhu. Scaling distributed energy storage for grid peak reduction. In *e-Energy*, pages 3–14. ACM, 2013.
[15] D. Salomon, G. Motta, and D. Bryant. *Data Compression: The Complete Reference*. Springer, 2007.
[16] P. Srikantha, S. Keshav, and C. Rosenberg. Distributed control for reducing carbon footprint in the residential sector. In *IEEE Third International Conference on Smart Grid Communications*, 2012.
[17] P. Srikantha, C. Rosenberg, and S. Keshav. An analysis of peak demand reductions due to elasticity of domestic appliances. In *e-Energy*, 2012.
[18] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake. Leveraging smart meter data to recognize home appliances. In *IEEE International Conference on Pervasive Computing and Communications*, 2012.
[19] L. Xiang, J. Luo, and C. Rosenberg. Compressed data aggregation: Energy-efficient and high-fidelity data collection. *IEEE/ACM Transactions on Networking*, 21(6):1722–1735, 2013.
[20] P. Yi, T. Zhu, B. Jiang, B. Wang, and D. Towsley. An energy transmission and distribution network using electric vehicles. In *ICC*, 2012.
[21] P. Yi, T. Zhu, G. Lin, and Q. Zhang. Routing renewable energy using electric vehicles in mobile electrical grid. In *MASS*, 2013.
[22] P. Yi, T. Zhu, Q. Zhang, Y. Wu, and J. Li. A denial of service attack in advanced metering infrastructure network. In *ICC*, 2014.
[23] Q. Zhang, T. Zhu, Y. Ping, and Y. Gu. Cooperative data reduction in wireless sensor network. In *GLOBECOM*, 2012.
[24] C. Zhao, U. Topcu, and S. Low. Optimal load control via frequency measurement and neighborhood area communication. *IEEE Transactions on Power Systems*, 28(4):3576–3587, 2013.
[25] W. Zhong, Z. Huang, T. Zhu, Y. Gu, Q. Zhang, P. Yi, D. Jiang, and S. Xiao. ides: Incentive-driven distributed energy sharing in sustainable microgrids. In *IGCC*, 2014.
[26] T. Zhu, Z. Huang, A. Sharma, J. Su, D. Irwin, A. Mishra, D. Menasche, and P. Shenoy. Sharing renewable energy in smart microgrids. In *ICCPS*, 2013.
[27] T. Zhu, A. Mishra, D. Irwin, N. Sharma, P. Shenoy, and D. Towsley. The case for efficient renewable energy management in smart homes. In *BuildSys*, pages 67–72. ACM, 2011.
[28] T. Zhu, S. Xiao, Y. Ping, D. Towsley, and W. Gong. A secure energy routing mechanism for sharing renewable energy in smart microgrid. In *SmartGridComm*, pages 143–148. IEEE, 2011.