# Smart Power Plugs for Efficient Online Classification and Tracking of Appliance Behavior

Muhammad Aftab
Khalifa University of Science and Technology, Masdar Institute
muhaftab@masdar.ac.ae

Chi-Kin Chau
Khalifa University of Science and Technology, Masdar Institute
ckchau@masdar.ac.ae

## ABSTRACT

Smart power plugs are notable cyber physical systems that can track and control appliance behavior. Traditional smart plugs provide limited functions, often only capable of recording consumption data, without advanced automation features. In future smart homes, sophisticated functions are expected to be provided by smart plugs, such as online learning, classification, and diagnosis of appliance behavior. Instead of relying on centralized cloud servers for providing advanced functions, the paradigms of edge and fog computing are increasingly populated by the applications of Internet-of-things, which aim to provide timely intelligent processing using small local memory space in a standalone manner. In this paper, we develop standalone smart plugs that are capable of providing efficient online classification and tracking functions on the continuous power signals of appliances. We built and implemented our smart plug systems using low-cost Arduino platform with a small amount of memory space. In general, our standalone online classification and tracking systems can be applied to a variety of smart sensors for wearable, biomedical, and environmental monitoring applications.

## CCS CONCEPTS

• **Computer systems organization** → **Embedded hardware**; **Embedded software**; • **Hardware** → *Sensor devices and platforms*;

## KEYWORDS

Smart power plugs, cyber-physical systems, edge/fog computing, Internet-of-things, online information processing

**ACM Reference format:**
Muhammad Aftab and Chi-Kin Chau. 2017. Smart Power Plugs for Efficient Online Classification and Tracking of Appliance Behavior. In *Proceedings of APSys '17, Mumbai, India, September 2, 2017,* 7 pages.
https://doi.org/10.1145/3124680.3124735

## 1 INTRODUCTION

The rise of cyber-physical systems enables diverse applications of pervasive intelligence, such as smart sensors for smart homes, wearable, biomedical, and environmental monitoring. These systems are usually implemented in small-footprint low-cost embedded systems, which only allow efficient computations and small memory space. They can operate in standalone mode or connected mode (if external communications are allowed for information exchanges). Recently, instead of relying on centralized cloud servers for intelligent information processing, the paradigms of edge and fog computing are increasingly populated by the applications of Internet-of-things, which aim to provide timely intelligent processing using small local memory space in a standalone manner. Edge computing pushes the computational intelligence and data analytics of information processing directly into smart end nodes (e.g., smart sensors, tags), whereas fog computing pushes the computational intelligence down to the local-area networks in fog nodes or gateways. Moreover, because of portability and limitation of battery life, reduced external communication overhead to remote cloud servers is often desirable.

To realize the potentials of edge and fog computing, online information processing tasks are required to be implemented directly in smart sensors. One particular example is streaming data processing systems, which are a class of information processing systems for a sequential stream of input data using a small amount of local memory space. These systems are usually required for cyber-physical systems with constrained local memory space and limited external communications. Traditional settings of streaming data processing often consider discrete digital input data, such as data objects carrying unique digital identifiers. However, the paradigm of Internet-of-things, which bolsters close interactions with physical environments, has been increasingly applied to diverse scenarios of data sensing in terms of continuous signals. For these scenarios, streaming data processing systems are required to take explicit consideration of input data as continuous signals.

As a primary example studied in this paper, we consider smart power plugs (or simply smart plugs), which are computing devices augmented to power plugs to monitor and track energy consumption signals, as well as performing inference and diagnosis tasks for the connected appliances. Smart plugs and sensors may share data with each other if efficient information dissemination is provided without exacerbating their battery life. We note that a similar setting occurs for body area or biomedical sensors that track and infer biological signals with constrained local memory space and limited external communications. In general, the framework of Internet-of-things requires distributed systems to perform sensing, tracking and inference tasks using small local memory space.

In general context, one of the major tasks of cyber-physical systems is to record and recognize the occurrences of events associated with certain continuous signals over time. Furthermore, multiple sensors may need to correlate their records temporally to identify a suitable context of events among the sensors. For example, one

*APSys '17, September 2, 2017, Mumbai, India*
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5197-3/17/09…$15.00
https://doi.org/10.1145/3124680.3124735

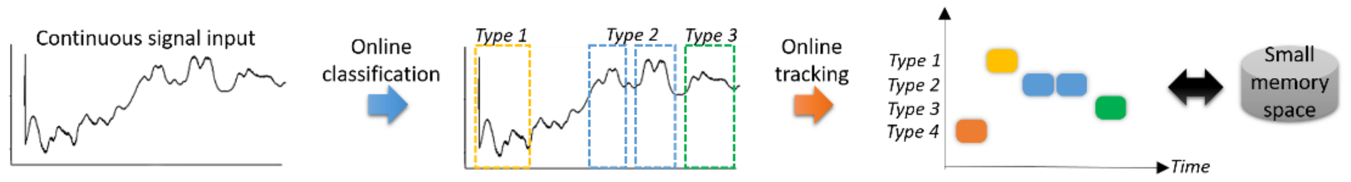


**Figure 1: Basic framework of streaming data processing systems for classifying and tracking occurrences of continuous signal inputs, using small local memory space.**

has to recognize the activities of occupants by multiple sensors in smart home applications. The basic framework of streaming data processing systems is presented in Fig. 1.

The overall scope of our research is to study and extend streaming data processing systems, where we explicitly consider continuous signal inputs to support the following functions:

(1) Learning and classifying the events associated with the continuous signal input in an online manner.
(2) Tracking the occurrences of events using small local memory space.
(3) Tracking the temporal correlations among multiple sensors to identify the context of events.

This paper studies the problem of learning and classifying sensor events, and provides a high-level introduction to our approach for tracking temporal occurrences of events. We develop standalone smart plugs that are capable of providing efficient online classification and tracking functions on the continuous power signals of appliances. Building upon our previous work [1, 2], we built and implemented our smart plug systems using low-cost Arduino platform with a small amount of memory space. In general, our standalone online classification and tracking systems can be applied to a variety of smart sensors for wearable, biomedical, and environmental monitoring applications.

## 2 BACKGROUND AND RELATED WORK

There is a large body of work for streaming data algorithms. The analysis of streaming data algorithms focuses on the worst-case inputs, by which the adversary can select arbitrary inputs to induce the worst-case performance of the algorithms, as compared to the offline setting with known future inputs. The basic idea of streaming data algorithms is to make use of randomized data structures which are able to amortize the worst-case inputs with high probability. Clustering is often employed for the analysis of data streams, whereby the data stream is partitioned into clusters such that elements within a cluster are similar to each other and different from events in other clusters. A large number of algorithms have been proposed for clustering of data streams, which can be divided into two classes: (i) algorithms that summarize that data stream in an online manner, then generate clusters from the summary data in offline mode, (ii) algorithms that are able to cluster the data stream entirely in online manner without the need for the offline processing. An extensive survey of both types of algorithms can be found in [16]. The authors in [10] put forth a streaming algorithm that balances clustering quality with time complexity using a small amount of memory. An online variant of the k-Means clustering algorithm is proposed in [5], which groups parallel streams of continuously evolving time series data in constant time. Another study [18] proposed a technique to yield meaningful clusters based on structural features instead of distance metrics to reduce dimensionality and decrease sensitivity to missing data. A comprehensive review of the various experimental methods employed for data stream clustering and its applications can be found in [9, 13].

The inference and diagnosis tasks for the electric appliances by monitoring their energy consumption signals can be achieved using either non-intrusive methods or intrusive methods. Non-intrusive methods involve measuring the energy consumption of the entire house and disaggregating it to identify individual appliances and track their behaviors. Intrusive methods, on the other hand, require measuring the energy consumption of individual appliances in a home using some kind of energy meters or smart plugs. The authors in [4] present an experimental method where they monitor the power consumption data of a large number of electric appliances and extract a small number of common characteristics, which are then used to derive a small number of model types that describe the power usage pattern of the majority of commonly used appliances. A related work [11], apply statistical and machine-learning methods for automatic modeling of appliances from their power consumption data. Their method is able to automatically model both the appliance type and its usage pattern. Their approach is to fit a curve or a distribution onto the power consumption data that best describes the appliance behavior.

The are several differences between our study and the existing studies, which characterize the novelty of our solution: (i) With one or two exceptions (such as [3]), most of the proposed non-intrusive, as well as intrusive methods, do not perform real-time appliance identification and instead work with pre-recorded data in an offline manner, where all the available data is employed for identification. Our smart plug system, on the other hand, is able to achieve real-time monitoring and classification of appliances using only the data seen thus far, without any knowledge of the future data; (ii) Our smart plug is developed using low-cost embedded systems, and performs all processing tasks locally in an online manner using small memory space, which entails extremely efficient implementation of the classification and clustering algorithms; (iii) The existing classification and clustering algorithms are not designed with storage and/or computation limitations in mind. Thus, we make non-trivial modifications to the existing algorithms so that our smart plug can execute them for classification and clustering using the available processing power and memory.

## 3 PROBLEM AND FORMULATIONS

We first formulate the problem considering a general setting of data sensing with multiple sensors. We then also provide specific examples regarding the applications of smart homes.

### 3.1 Event Classification

First, we consider the single-sensor scenario. We denote by $x(t)$ a stream of continuous-time signal observed by a sensor. The signal may be associated with multiple events at different times that are not revealed to the sensor. Given the continuous nature of signals, the sensor needs to interpret and identify the states and transitions embodied by the signal.

***Known Event Types***: The signal may be triggered by a finite number of events (e.g., consumption of a particular appliance), which may be known by the sensor. In particular, there are several common models suitable for describing the energy consumption of appliances (see Section 4.2). One may assume that the signal is generated by one of these models, and the sensor attempts to match and classify according to a set of known class of events.

***Unknown Event Types***: Alternatively, one can employ a clustering algorithm to learn the events and associate with the segments signal in a way to minimize the distance between events. Let $x(t_1, t_2)$ be the segment of signal from time $t_1$ to $t_2$, and $\mathcal{S}$ be a non-overlapping segmentation of time, such that $(t_1, t_2), (t_2, t_3) \in \mathcal{S}$, then $x(t_1, t_2)$ and $x(t_2, t_3)$ are non-overlapping segments of $x$. We denote the set of events of signal be $\mathcal{E}$. For each event $i \in \mathcal{E}$, denote $c_i$ be a canonical signal. Let $\mathrm{d}\big(x(t_1, t_2), c_i\big)$ be a distance metric given $x$. One example of distance metric is dynamic time warping (DTW). We aim to find suitable $\mathcal{S}$ and $\mathcal{E}$, such that

$$\min_{\mathcal{S},\mathcal{E}} \; \rho|\mathcal{E}| + \sum_{i \in \mathcal{E}} \sum_{(t_1,t_2)\in\mathcal{S}_i} \mathrm{d}\big(x(t_1, t_2), c_i\big) \tag{1}$$

where $\rho$ is a weight assigned to balance the two objectives, and

$$\mathcal{S}_i \triangleq \Big\{(t_1, t_2) \in \mathcal{S} \mid \mathrm{d}\big(x(t_1, t_2), c_i\big) \geq \mathrm{d}\big(x(t_1, t_2), c_j\big), \forall j \in \mathcal{E}\backslash\{i\}\Big\}$$

### 3.2 Occurrence Tracking

Suppose that the sensor can infer the events $\mathcal{E}$ and the respective segmentation $\mathcal{S}$. The occurrences of each event will be recorded. In general, there is a stream of items with multiple occurrences. We want to record the items with the most prominent occurrences when observing the stream continuously. Note that we do not know the number of distinct items in advance, and the available storage space is much less than the number of distinct items in the stream.

Let $X_i$ be the total occurrence of event $i \in \mathcal{E}$ with a certain an epoch of time $\mathcal{T}$. Our objective is that given a memory space constraint c in an epoch of time $\mathcal{T}$, one can identify the occurrences of the common events $X_i$, while only using a storage space of size c. The difficulty is that $|\mathcal{T}|$ may be a large number, and the memory space constraints c cannot grow at the same pace as $|\mathcal{E}|$, and $\mathcal{E}$ is supposed to be observed continuously.

Hence, we focus on the task of *approximate* tracking, by obtaining the estimated occurrence $\widehat{X}_i$ at the end of $\mathcal{T}$. We aim at a proven bound of error probability for accuracy guarantee, defined as an $(\epsilon, \delta)$-accurate estimation, which satisfies:

$$\mathbb{P}\Big(|\widehat{X}_i - X_i| \geq \epsilon \cdot |\mathcal{T}|\Big) < \delta$$

$\epsilon$ and $\delta$ are the parameters for controlling the trade-off between accuracy and memory space.

## 4 EVENT CLASSIFICATION RESULTS

This section presents the algorithms and results for learning and classifying the events associated with the continuous signals in an online manner. As a case study, we consider the continuous power consumption signals of electrical appliances.

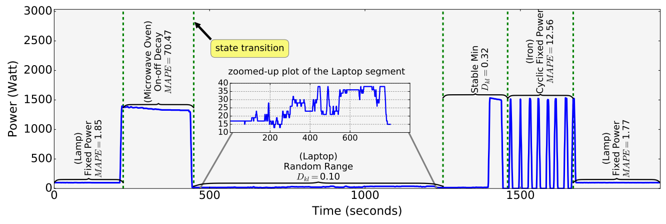
### 4.1 Event Detection

The first step is the detection of the events embodied by the appliance power consumption signal. An appliance may transition through several operating states, where the power consumption pattern usually varies from one state to the next. A state transition may signify a switch from one basic load to another, where the basic loads together constitute the overall load of the appliance. For example, a dishwasher contains a motor and a heating element. The motor powers the pump to propel and spray the hot water on the dirty dishes. The heating element is responsible for heating the water to wash the dirty dishes or heating the air to dry the clean dishes off. A state transition may also occur among the different active states of the same load. For instance, electric clothes Irons are equipped with temperature control dials, which allows a user to select the Iron's operating temperature. Changing the position of the control dial causes the Iron to change from one active power state to another. We treat all such state transitions as events and aim to interpret and identify these events. Specifically, we propose Algorithm 1 to detect the appliance state transition events by analyzing the appliance power consumption data stream. The algorithm is an online variant of the *Energy-specific Change Point Detection* algorithm originally proposed in [11].

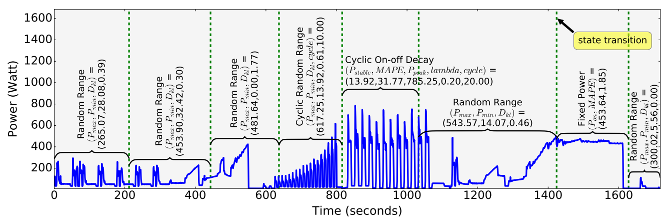
**Algorithm 1:** Online detection of appliance state transitions

```
   Initialization: $H \leftarrow \emptyset$, edge $\leftarrow$ False, $t_{\text{prev\_event}} \leftarrow 1$
   Input: $X,\ t_{\text{now}},\ \phi,\ \delta, \eta < \delta$
1  if $|H| < \eta$ then
2      $H \leftarrow H \cup$ ApEn($X[t_{\text{now}} - \phi : t_{\text{now}}]$)
3  else
4      $H \leftarrow H - H[1]$                         ▹discard oldest element
5      $H \leftarrow H \cup$ ApEn($X[t_{\text{now}} - \phi : t_{\text{now}}]$)   ▹add new element
6      edge $\leftarrow$ CannyEdge1D($H$)               ▹returns True if detects edge
7  end
8  if edge = True and $t_{\text{now}} - t_{\text{prev\_event}} > \delta$ then
9      $t_{\text{prev\_event}} \leftarrow t_{\text{now}}$
10     return True                  ▹state transition event detected at $t_{\text{now}}$
11 else
12     return False                 ▹no state transition at $t_{\text{now}}$
13 end
```

The algorithm is based on the notion of *Approximate Entropy* (ApEn), which is an algorithm for quantifying the repeatability or predictability within time series data [15]. Our algorithm operates over a sliding look back window of the appliance power consumption data stream. For each element of the data stream arriving at time $t_{\text{now}}$, it computes the approximate entropy over a look back window of length $\phi$, from $t_{\text{now}} - \phi$ to $t_{\text{now}}$. The algorithm, then, executes the one-dimensional variant of the Canny edge algorithm CannyEdge1D [11], which analyzes a sliding look back window of the most recent $\eta$ approximate entropy values and identifies sudden and significant changes in entropy as edges. If an edge is detected and at least $\delta$ time has elapsed since the time of the previous event $t_{\text{prev\_event}}$, then the algorithm has detected a state transition event at $t_{\text{now}}$. The time constraint $\delta$ is enforced because in practice an appliance stays in a given state for at least some time.



**(a) Composite trace of multiple appliances**



**(b) Washing machine**

**Figure 2: Results of online appliance state transition event detection using Algorithm 1.**

The results of Algorithm 1 are illustrated in Fig. 2. Each vertical dashed line in the figures indicates a state transition event that is detected by Algorithm 1. During the experiment, $\phi$, $\delta$, and $\eta$ were set to 60, 200, and 7, respectively. In addition, approximate entropy calculation requires specifying two parameters: the sequence length $M$ and filtering level $r$. Following [11], $M = \phi/4$ and $r = .2.\sigma(X[t_{\text{now}} - \phi : t_{\text{now}}])$ were used during the experiment. In particular, Fig. 2a shows the results of state transition detection for the power trace of multiple appliances (Lamp, Microwave Oven, Laptop Computer, and Iron), where at any given time only one appliance was connected to the smart plug. As can be seen, Algorithm 1 is able to partition the power trace into segments corresponding to different appliances through accurate detection of state transition events in most cases. We also obtained results of state transition detection for the power consumption trace of a washing machine as shown in Fig. 2b. A washing machine is a composite load with several active power states, each corresponding to a constituent basic load. Again, Algorithm 1 is able to partition the trace into segments of active power states as indicated by the vertical dashed lines. We note that the figures contain additional information, which will be described in the next two subsections, where we classify the events detected by Algorithm 1.

## 4.2 Classification of Known Events

An event may belong to a finite number of events (e.g., consumption of a particular appliance), which may be known by the smart plug. Every electrical appliance possesses its own power signature which can be used to automatically identify the specific appliance. There are a few basic models suitable for describing the power consumption of appliances [4]:

(1) *On-off model:* Appliances belonging to this model draw fixed power $\mathtt{P}_{\mathtt{on}}$ when active.
(2) *On-off decay model:* In this case the appliance power consumption follows an exponential decay curve, dropping from initial surge power $\mathtt{P}_{\mathtt{peak}}$ to a stable power $\mathtt{P}_{\mathtt{active}}$ at a decay rate $\lambda$.
(3) *On-off growth model:* The power consumption of these appliances follows a logarithmic growth curve, starting with a power level $\mathtt{P}_{\mathtt{base}}$ and a growth rate $\lambda$.
(4) *Stable min-max model:* These appliances draw stable power $\mathtt{P}_{\mathtt{stable}}$ with random upward or downward spikes $\mathtt{P}_{\mathtt{spike}}$. The magnitude of random spikes is uniformly distributed between $\mathtt{P}_{\mathtt{stable}}$ and $\mathtt{P}_{\mathtt{spike}}$. The interarrival time of spikes follows exponential distribution with mean $\lambda$.
(5) *Random range model:* The power consumption of these appliances is similar to a random walk between a maximum power $\mathtt{P}_{\mathtt{max}}$ and a minimum power $\mathtt{P}_{\mathtt{min}}$, where the power consumption randomly varies within these bounds.
(6) *Cyclic model:* These appliances exhibit repeating power consumption patterns.

Common resistive and inductive appliances exhibit on-off, on-off decay, or on-off growth behavior, whereas appliances with non-linear power consumption exhibit stable-min, stable-max, or random-walk behavior. In addition, many appliances (such as washing machine, air-conditioner etc) are composed of a combination of the above described resistive, inductive, and non-linear constituent loads, due to which these appliances exhibit complex power usage patterns.

Once the smart plug detects an event, it classifies the power data segment for the event into one of the above six types of models. For a given appliance, the smart plug chooses the model that best explains the observed power consumption. For the first three models, the smart plug uses ordinary linear least squares method to fit a straight line, an exponential decay curve, or a logarithmic growth curve onto the data. To fit exponential curve, we employ a special technique proposed in [12], since fitting an exponential decay curve involves inferring three parameters (i.e. $\mathtt{P}_{\mathtt{peak}}$, $\mathtt{P}_{\mathtt{active}}$, $\lambda$), which can not be accomplished using ordinary linear least squares method. For the remaining three models, which are exhibited by appliances with non-linear power consumption, the smart plug fits a probability distribution following [4].

The classification results for trace segments partitioned by Algorithm 1 are shown in Figure 2. In particular, Figure 2a provides the fitted model along with the model error for each segment, whereas Figure 2b additionally provides parameters of the fitted model. We determine the curve fitting error for linear segments using Mean Absolute Percentage Error (MAPE) [11], where a lower MAPE indicates a better fit. Commenting on MAPE in Figure 2, we get reasonably low MAPE values in all cases except for Microwave Oven because the particular segment contains data from the previous as well as the next state of the appliance due to slightly imprecise detection of the state transition event by Algorithm 1. Similarly, to evaluate distribution fitting for non-linear segments, we use Kullback-Leibler (KL) Divergence [8], which indicates the information loss incurred by fitting a specific probability distribution to the data. Similar to MAPE, a lower KL divergence ($D_{kl}$) value implies a better fit. Looking at the $D_{kl}$ values in Figure 2, we can see that the fitted probability distribution is a good approximation of the true probability distribution for each non-linear segment.

## 4.3 Classification of Unknown Events

In the previous section, we assumed that an event may belong to a finite number of events (e.g., consumption of a particular appliance), which may be known by the smart plug. In this section, we assume that events are unknown to the smart plug. For the case of unknown events, our approach is to employ online clustering to group the events detected by Algorithm 1 into clusters such that events within a cluster are similar to each other and different from events in other clusters. We note that the number of data elements in one event may be different from another event since appliance state transitions do not necessarily occur at fixed time intervals. Thus, we use the following five summary statistics to represent each event in order to facilitate uniform clustering of the events:

- Measures of central tendency: (i) *arithmetic mean*, (ii) *median*, and (iii) *mode*.
- Measures of statistical dispersion: (iv) *standard deviation*, and (v) *range*.

Our clustering algorithm, which is based on *The Doubling Algorithm* [7] and *Online k-Means Clustering with Discounted Updating Rule* [14], is provided in Algorithm 2. The algorithm aims to find a suitable solution to the problem posed in Equation 1. It consists of two stages: the first is the update stage in which the algorithm adds each event either to an existing cluster or puts it in a new cluster. This stage continues until the number of clusters exceeds $k$, at which point the algorithm moves to the second stage. In the second stage, the algorithm reduces the number of clusters by simply merging the two nearest clusters. The merging stage guarantees that no more than $k$ clusters are used to cluster the events in the data stream even if the stream arrives forever. What follows is a detailed description of the pseudo code provided in Algorithm 2.

*(a)* **Initialization:** The algorithm starts by initializing the first $k+1$ events as separate clusters, where $\mathcal{E}$ denotes the set of clusters and $C$ represents the set of cluster centers. At this point, each event itself is the cluster center since each cluster has only one event. The minimum inter cluster distance in $\mathcal{E}$ is denoted by $d*$.

*(b)* **Updating Clusters:** Upon receiving a new event $z_t$, the algorithm finds the cluster whose center is nearest to $z_t$ (line 2). If the distance between the nearest cluster center and $z_t$ is less than 2d*, then $z_t$ is added to that cluster (line 4) and the cluster center is shifted proportionally (line 5). If, however, the distance is greater than 2d*, then a new cluster is created and event $z_t$ is added to it (line 7). Notably, we use the discounted updating rule shown in line 5 instead of standard online k-Means updating rule ($c_i \leftarrow c_i + \frac{1}{|e_i|}(z_t - c_i)$) because the discounted update has been shown to work comparatively better when the data lends itself to clustering and the cluster centers are shifting over time [14]. The variable $\alpha \in (0, 1)$ determines the relative weightage of the new event $z_t$ and has the effect of exponential smoothing.

*(a)* **Merging Clusters:** Whenever the total number of cluster exceeds $k$, the merging step is invoked to reduce the total number of clusters back to $k$. In this step, the algorithm first finds and merges the two closest clusters (lines 10-12). Then, it adds the newly merged cluster to $\mathcal{E}$ and removes both old clusters from $\mathcal{E}$ (line 13). Similarly, the center of the new cluster is added to $C$ and the old centers are deleted (line 14). Finally, the cost of creating new clusters $d*$ is doubled (line 15). The effect of doubling the cost is that eventually it will become prohibitively expensive to create new clusters and the algorithm will simply assign new events to one of the existing $k$ clusters. The intuition behind doubling the cost is that most electrical appliances have a limited number of distinct power states, and the algorithm is likely to encounter the different power states of an appliance within a finite time period.

**Algorithm 2:** Online event clustering using k-Means

```
Input: Event: $z_t$, Max clusters: $k$, Weight: $\alpha$
Initialization: $\mathcal{E} \leftarrow \{\{z_i\} : i \in \{1, \ldots, k+1\}\}$ $C \leftarrow \{z_i : i \in \{1, \ldots, k+1\}\}$
    $d^* \leftarrow \min_{(c_1, c_2 | c_1 \neq c_2) \in C} \|c_1, c_2\|_2^2$
1  if $|\mathcal{E}| <= k$ then                          ▹CLUSTERS UPDATING STEP
2      $i \leftarrow \arg\min_{i \in \{1,\ldots,k\}} \|z_t, c_i\|_2^2$      ▹get the nearest cluster to $z_t$
3      if $\|z_t, c_i\|_2^2 <= 2d^*$ then
4          $e_i \leftarrow e_i \cup z_t$                    ▹add event $z_t$ to the nearest cluster
5          $c_i \leftarrow c_i + \alpha(z_t - c_i)$         ▹update the cluster center
6      else
7          $\mathcal{E} \leftarrow \mathcal{E} \cup \{z_t\}$     ▹create new cluster comprising of $z_t$
8      end
9  else                                                 ▹CLUSTERS MERGING STEP
10     $(i, j) \leftarrow \arg\min_{(i,j|i \neq j) \in |C|} \|c_i, c_j\|_2^2$   ▹find two closest clusters
11     $e' \leftarrow \{e_i \cup e_j\}$                    ▹merge the clusters
12     $c' \leftarrow \frac{1}{|e'|} \sum_{y \in e'} y$     ▹center of the merged cluster
13     $\mathcal{E} \leftarrow \mathcal{E} \cup e' - \{e_i, e_j\}$   ▹update clusters
14     $C \leftarrow C \cup c' - \{c_i, c_j\}$             ▹update cluster centers
15     $d^* \leftarrow 2d^*$                               ▹double of the cost of creating new clusters
16 end
```

Figure 3 illustrates the results of Algorithm 2 applied to cluster the segments in Figure 2b. The number of clusters $k$ was set to 4, while $\alpha$ was set to 0.5 during the experiment. There is a total of 4 subfigures, each representing a separate cluster. The plotted curves in the subfigures represent segments of the washing machine power trace (Figure 2b) that are clustered together by Algorithm 2. It can be seen that similar segments are clustered together. For instance, all curves in Figure 3b have Random Range model.

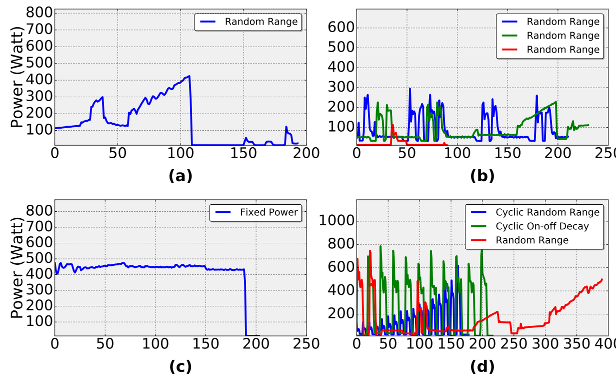


**Figure 3: Classification of unknown events using online clustering. X-axis shows time in seconds.**

To evaluate the clustering performance without reference to external information, we use within-cluster sum of squares error (WSSE) and between-clusters sum of squares error (BSSE), calculated using the following formulas:

$$WSSE = \sum_{i}^{k} \sum_{z \in e_i} (z - c_i)^2, \; BSSE = \sum_{i}^{k} |e_i|(c - c_i)^2$$

where $z$ is an event, $e_i$ is the $i_{th}$ cluster, $c_i$ is the center of cluster $i$, $|e_i|$ is the size of cluster $i$, and $c$ is the center of all clusters. WSSE measures measures how closely related are the events in a cluster (i.e., cohesion), while BSSE indicates how well-separated the clusters are from each other (i.e., separation). A good value for $k$ is the one that minimizes $CH = \frac{BSSE/(k-1)}{WSSE/(n-k)}$, where $n$ is total number of events, and CH is Calinski-Harabasz Index [6]. Table 1 lists the WSSE, BSSE, and CH for different values of $k$ when clustering the segments in Figure 2b using Algorithm 2. The CH row in the table indicates that the clustering performance will improve as we increase $k$.

| Number of clusters ($k$) | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| Cohesion (WSSE) | 2942862 | 3751363 | 4223219 | 4650565 |
| Separation (BSSE) | 2133919 | 1785765 | 2243008 | 2707623 |
| Calinski-Harabasz Index (CH) | 5.8 | 1.66 | 1.06 | 0.72 |

**Table 1: Performance evaluation of online event clustering.**

## 5 OCCURRENCE TRACKING

We briefly describe our approach for tracking of the occurrences of events. Recall from Section 3.2 that we want to keep track of the most frequently occurring events in the data stream using small memory space. The approach we use for occurrence tracking is called *count-min sketch*. A count-min sketch uses $k$ hash functions to track the events (or items in general) with the help of $m$ counters organized in a 2-dimensional ($k \times \frac{m}{k}$) array referred to as a sketch. Each event is mapped by the $k$ hash functions, where the $i_{th}$ hash function $h_i$ maps the event into one of the $\frac{m}{k}$ counters in the $i_{th}$ row of the sketch, and increments the particular counter. Thus, for each event $z$ in the stream, a counter is incremented in each of the $k$ rows of the sketch. The number of occurrences of an event $z$ at a given time is the minimum of the all the counters mapped by the hash functions for the event $z$.

$$N(z) = min\{Counter_{i,h_i(z)} : i = i, \ldots, k\} \tag{2}$$

N(z) is of course an overestimate of the true number of occurrences, because multiple items can be mapped to the same counter by a hash function. However, N(z) is not far from the true value.

## 6 TESTBED AND HARDWARE

In this section, we present the implementation details of the smart plug platform. The smart plug needs to have sufficiently powerful hardware to carry out real-time computation of the various tasks involved in event detection, classification, and clustering such as *Approximate Entropy, Edge Detection, Smoothing, Linear Regression, Least Square Fitting, Autocorrelation, and k-Means Clustering*. With this in mind, we decided to develop the smart plug platform based on the Arduino-compatible WiFi-enabled *ESP-12s* hardware platform [17]. Unlike the Arduino-family micro-controller boards (such as Arduino Uno, Nano, Micro, and Pro), ESP-12s has the processing power and memory capacity to compute the above tasks in a real-time. Figure 4 shows the hardware components of the smart plug. The smart plug contains a relay to control the power to the attached appliance and sensors to measure the instantaneous AC voltage and current. The smart plug performs the following additional functions besides online detection, classification, and tracking.

*(i)* **Accurate power measurement.** The ability to measure instantaneous voltage and current enables the computation of active power, reactive power, power factor, and more advanced power quality parameters, which can be helpful in power quality monitoring, and detection of malfunctions, etc.
*(ii)* **Remote control.** The smart plug supports remote control of the appliances locally or over the internet through an intermediate server. It provides RESTful APIs for controlling the attached appliance and querying its status using smartphones or web clients.

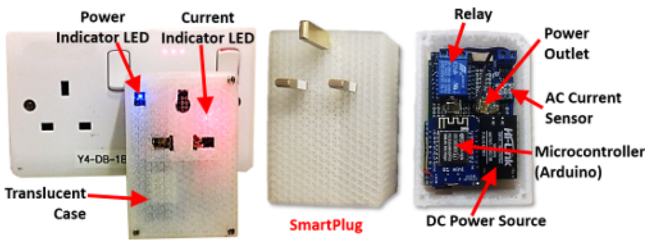


**Figure 4: Hardware components of the smart plug.**

A key component of the smart plug system is the computational overhead, which is important to evaluate whether the technique can be extrapolated to other data streams. Table 2 lists three of the computationally demanding tasks (first column) performed by our smart plug. For each task, the table provides the time it takes for the smart plug to execute the task in increasing order of complexity. In case of model fitting, for instance, the time taken by the smart plug grows sub-linearly with data length, whereas, for the remaining two tasks (Autocorrelation and Approximate Entropy), the growth is more than linear. Notably, the last column represents the maximum complexity of the given task such that if we increase the task complexity any further, the smart plug will not be able to perform appliance monitoring and classification in real-time.

| | | | | | |
|---|---|---|---|---|---|
| Autocorrelation | Lag | 50 | 100 | 200 | 400 |
| | Time Taken (milliseconds) | 24 | 88 | 344 | 1351 |
| Model Fitting | Data Length | 200 | 400 | 800 | 1600 |
| | Time Taken (milliseconds) | 529 | 711 | 1085 | 1564 |
| Approximate Entropy | Sliding Window Length | 60 | 120 | 240 | 480 |
| | Time Taken (milliseconds) | 38 | 126 | 616 | 2010 |

**Table 2: Computational overhead by the smart plug.**

## 7 CONCLUSION AND FUTURE WORK

This work presented a smart plug system that can perform sophisticated monitoring of the appliance power consumption behavior such as real-time detection, classification, and clustering of the appliance events. These monitoring features offer many benefits to researchers, e.g., automated demand response, appliance localization, detection of anomalous appliance behavior, and effective power allocation. The smart plug was developed using the low-cost Arduino open hardware platform. In future, we plan to use the smart plug system for implementing the functionality to track the temporal occurrences of events and identify the context of events by finding correlations among events from multiple smart plugs.

## REFERENCES

[1] Muhammad Aftab, Chi-Kin Chau, and Majid Khonji. 2017. Real-time Appliance Identification Using Smart Plugs: Demo Abstract. In *Proceedings of the Eighth International Conference on Future Energy Systems (e-Energy)*. ACM.

[2] Muhammad Aftab, Amalfi Darusman, Israa Al Qassem, Majid Khonji, and Sid Chi-Kin Chau. 2015. OS|Plug: Open Platform for Smart Plugs.. In *Proceedings of the Sixth International Conference on Future Energy Systems (e-Energy)*. ACM.

[3] L. P. R. Ambati and D. Irwin. 2016. AutoPlug: An Automated Metadata Service for Smart Outlets. In *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*. 1–8. https://doi.org/10.1109/IGCC.2016.7892604

[4] Sean Barker, Sandeep Kalra, David Irwin, and Prashant Shenoy. 2013. Empirical Characterization and Modeling of Electrical Loads in Smart Homes. In *Green Computing Conference (IGCC), 2013 International*. IEEE, 1–10.

[5] JÃijrgen Beringer and Eyke HÃijllermeier. 2006. Online Clustering of Parallel Data Streams. *Data & Knowledge Engineering* 58, 2 (2006), 180 – 204. https://doi.org/10.1016/j.datak.2005.05.009

[6] Tadeusz Caliński and Jerzy Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* 3, 1 (1974), 1–27.

[7] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. 1997. Incremental Clustering and Dynamic Information Retrieval. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (STOC '97)*. ACM, New York, NY, USA, 626–635. https://doi.org/10.1145/258533.258657

[8] Thomas M Cover and Joy A Thomas. 2012. *Elements of Information Theory*. John Wiley & Sons.

[9] Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. 2016. State-of-the-art on Clustering Data Streams. *Big Data Analytics* 1, 1 (2016), 13.

[10] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. 2003. Clustering Data Streams: Theory and Practice. *IEEE transactions on knowledge and data engineering* 15, 3 (2003), 515–528.

[11] Srinivasan Iyengar, David Irwin, and Prashant Shenoy. 2016. Non-intrusive Model Derivation: Automated Modeling of Residential Electrical Loads. In *Proceedings of the Seventh International Conference on Future Energy Systems (e-Energy '16)*. ACM, New York, NY, USA, Article 2, 11 pages. https://doi.org/10.1145/2934328.2934330

[12] Jean Jacquelin. 2009. Regressions and Integral Equations. (2009), 16–17 pages. www.scribd.com/doc/14674814/Regressions-et-equations-integrales.

[13] V. Kavitha and M. Punithavalli. 2010. Clustering Time Series Data Stream - A Literature Survey. *CoRR* abs/1005.4270 (2010). http://arxiv.org/abs/1005.4270

[14] Angie King. 2012. Online k-Means Clustering of Nonstationary Data. (2012).

[15] Steven M Pincus. 1991. Approximate Entropy as a Measure of System Complexity. *Proceedings of the National Academy of Sciences* 88, 6 (1991), 2297–2301.

[16] Jonathan A. Silva, Elaine R. Faria, Rodrigo C. Barros, Eduardo R. Hruschka, André C. P. L. F. de Carvalho, and João Gama. 2013. Data Stream Clustering: A Survey. *Comput. Surveys* 46, 1 (2013), 13:1–13:31.

[17] Espressif Systems. 2017. ESP8266: Low-power, Highly-integrated WiFi Solution. (2017). https://espressif.com/en/products/hardware/esp8266ex/overview

[18] Xiaozhe Wang, Kate Smith, and Rob Hyndman. 2006. Characteristic-based Clustering for Time Series Data. *Data mining and knowledge Discovery* 13, 3 (2006), 335–364.