Abstract—We present the ColoredPower algorithm, which is designed to provide collaborative electricity demand shaping for residential and small-business customers. Demand shaping for this market sector is an important and challenging problem, since the vast number of such customers collectively account for a large fraction of total electricity consumption, yet each individual’s consumption is small. Under the PACEM system, customers participate by “coloring” their appliances with a qualitative priority such as “can be shut off at peak power.” Demand shaping for this system must be scalable to millions of appliances, operate quickly and fairly across customers, and act on any given appliance infrequently. This last constraint is particularly challenging: if an appliance that switches on or off must not be switched again for many minutes, then at any instant, a large fraction of appliances may not be controllable. The ColoredPower algorithm addresses these challenges using randomized local actions. When the action distribution is adjusted to compensate for currently uncontrollable appliances, standard feedback controllers can be used to produce local actions that combine to create the desired global effect. Experiments in simulation verify that the algorithm provides fair control that is fast, scalable, and robust enough to be realistically deployable.

Keywords—electricity demand management; demand shaping; peak-shaving; distributed algorithms; spatial computing; amorphous computing;

I. INTRODUCTION

Demand shaping is a pressing problem in energy delivery. Not only is there a general environmental motivation for reducing energy consumption, but peaks in demand are extremely costly, requiring generation capacity that goes largely unused, and can cause blackouts or brownouts. Residential and small business consumers make up a large fraction of total electricity consumption, and studies have shown that their energy needs are fairly flexible (e.g. [1]), implying a large potential for demand shaping.

However, demand shaping for this market sector is extremely challenging. The number of customers is extremely high, often in the millions, so a system must be highly scalable. The energy consumption per customer is low, however, so the per-customer cost of deployment and operation must also be extremely low. Moreover, small customers are not typically available or motivated to devote significant effort to managing their energy use, so the system must require negligible ongoing attention from the customer. As a result, existing demand shaping solutions of the sort already being deployed for for large electricity consumers like manufacturers and municipal stadiums (e.g. EnerNOC) do not scale to the small customer market.

The Proto/Amorphous Cooperative Energy Management (PACEM) system[2] aims to address the challenges of small customer demand shaping by means of distributed control and a simple qualitative preference interface. In the PACEM system, customers participate by “coloring” their appliances with a qualitative priority such as “can be shut off at peak power.” Participating appliances communicate with one another and with a gateway device that connects to other households, forming a regional demand management network. The participating appliances then self-organize over this network to perform collective control, shutting off some appliances in order to adjust their aggregate energy consumption to match a global demand shaping command.

Besides scaling to millions of appliances, demand shaping for this system must operate quickly enough to be useful—on the order of a few minutes. Also, the set of appliances shut off must be distributed fairly across customers over time. Finally, if a given appliance is switched on or off, it must not be switched again for many minutes—both to avoid damaging the appliance and to avoid “flickering” that irritates a customer. This low-frequency switching constraint is particularly challenging, for it means that at any instant a large fraction of appliances may not be controllable.

We have developed the ColoredPower algorithm as a controller for the PACEM system, addressing these challenges using randomized local actions. When the action distribution is adjusted to compensate for currently uncontrollable appliances, standard feedback controllers can be used to produce local actions that combine to create the desired global effect. In this paper, we first give an overview of the PACEM system and describe the ColoredPower algorithm in detail. We then analyze the algorithm to determine that it satisfies its design goals. Finally, we verify our analysis in simulation, demonstrating that the algorithm provides fair control that is fast, scalable, and robust enough to be realistically deployable.
II. PACEM System Overview

PACEM [2] proposes a network that automatically matches a grid control authority’s requests for demand regulation with ways customers are willing to decrease demand (Figure 1). The control authority (e.g., a utility or a municipal government) sends supply data and regulation requests to a network of demand controllers. Each controller talks to a network of smart outlets and appliances in the household or business that it regulates. A distributed algorithm running on the network then harvests the potential demand decreases offered by appliances in order to meet the request from the control authority as best it can given the available resources.

Customers indicate their demand flexibility by choosing what to plug into smart outlets and setting when an outlet is allowed to be regulated. The initial design for PACEM envisions four categories of flexibility, each associated with a color: anytime (“green”), peak power (“yellow”), emergency only (“red”), and never (“black”) (Figure 2). The only other way that customer needs to interact with the system is to push a 1-hour override button if they want to use an appliance that they normally allow to be cut off. The total available flexibility is thus the sum of all of the individual consumers’ flexibility colorings. When the control authority requests a demand decrease, the system must select a set of devices to shut off such that the aggregate consumption drops to the target level.

More information on PACEM’s incentivization, feasibility of deployment, and variation by category of appliance can be found in [2] and [3]: in this paper, we focus on the problem of actually controlling devices in response to commands from the control authority.

A. Control Algorithm Requirements

We consider PACEM’s demand response as a problem of feedback control: we need to closely follow (“track”) the global target for energy consumption, which we will call $Q_t$, and which generally needs to match the available supply of power. We want our measured global power consumption $Q_m$ to follow the global target $Q_t$ as closely as possible. The total supplied power has many different factors that go into how it changes over time, including economics of the power companies supplying the power, the capacities and overhead of the generators producing the power, government regulations, and of course the global demand for power $Q_d$. Note that when the supply and demand are not restricted, $Q_m = Q_d$.

Any algorithm to control the distributed network of small customer electrical devices must satisfy the following requirements:
1) **Demand Flexibility:** At any given point in time, the demand for power should have as much flexibility as possible—either to shut down devices that are currently on, or to relax and turn back on devices that were shut down for demand response.

2) **Dynamic Response:** The algorithm must be able to control the measured global power consumption $Q_m$ such that it tracks a changing global target $Q_t$ quickly and reliably. For the current electrical grid, this means a significant response on the order of minutes.

3) **Fairness:** Because PACEM depends on weakly incentivized participation, we do not want users of the system to perceive it as unfair, or else they may stop participating. For example, a user may get upset if his air conditioner gets shut off more than his neighbor’s. To satisfy this, we require that over a sufficiently long period of time, the expected total power consumption by two identically colored devices should be the same, and that if at any given time two devices have the same state, they should have an equal chance of deviating from their state.

4) **Privacy:** Fine-grained power consumption data is a significant privacy concern, so the data about different users and their devices should remain private. We thus require that global computations operate on many-consumer aggregates (which are by nature anonymized), and that no single device should ever have information about a large number of other individual devices.

5) **Scalability:** The algorithm must be scalable to very large numbers of devices. For instance, a large city grid might have tens of millions of devices.

6) **Non-intrusiveness:** The devices running the algorithm should only switch on or off occasionally, with many minutes between switchings. A user should always be able to “override” the system on a particular device at any time.

### B. Related Projects

PACEM draws inspiration from other, smaller scale energy demand regulation projects. One such example is Hewlett-Packard’s “Smart Cooling” project, which includes temperature control through the spatial distribution of processes[4]. Another is the market-based time-shifting of refrigerator cooling decisions envisioned by Ogston et al[5]. However, such market-based planning approaches:

1) may not be able to react quickly enough to unexpected situations like a power generator failure,
2) may not scale to the complexity of highly heterogeneous systems. For example, Ogston et al. point out the difficulty of computing optimal solutions even in their limited planning scope.
3) require reliable design of a complex artificial market, which is an open problem for domains of this complexity, particularly given the interaction with irrational human users.

A more similar approach to PACEM is the EnviroGrid controller, which uses non-market control to locally desynchronize periodic loads[6], but which does not change overall demand.

There is much active commercial research and development in the area of demand control as well, largely focused around either large consumers, as in the case of EnerNOC[7], or centralized solutions, as in Google’s PowerMeter or the Tendril platform. A brief survey of existing demand response programs, as of 2006, can be found in the FERC report in Chapter VI[8]. A fully developed PACEM would be able to support many of these programs that are currently executed manually.

### III. THE ColoredPower ALGORITHM

We have designed the ColoredPower algorithm to fulfill these requirements via distributed probabilistic control. The reasons for choosing a distributed probabilistic approach are threefold: speed, robustness, and privacy.

The basic idea is this: rather than attempt to gather fine-grained data back to a central point, the ColoredPower algorithm maintains an aggregate model of global system state, which is shared with all devices. When the target consumption $Q_t$ changes, each device computes from this model what percentage of devices that should change state overall, then flips a coin to determine whether it is one of those devices. Although random variance and consumer heterogeneity make it extremely unlikely that this control will immediately succeed, the aggregate consumption is likely to be much closer to the target, and with feedback can quickly arrive. What is more, by the law of large numbers, the more consumers that participate in the system, the better that probabilistic control is expected to perform. Note that we will never consider the amount of power consumed by an individual device in determining whether to switch that device off; this is done to help satisfy the perceived fairness constraint, because it means that a user’s experience will not be significantly affected by their coloring choices.

Decentralized probabilistic control also provides robustness, since it does not require critical points in the network where a small number of failing devices can cripple the system. Finally, since control is local, data can be aggressively aggregated to preserve privacy.

We will now explain the ColoredPower algorithm using a step by step build up, starting with the algorithm for the simplest system and adding refinements to produce the full ColoredPower algorithm.

#### A. Base Local and Global State

Let us begin by defining the base information that we assume is available for the network of devices. For now, as we begin with the simpler control algorithms, we will

1) The algorithm must be scalable to very large numbers of devices. For instance, a large city grid might have tens of millions of devices.

2) The algorithm must be able to quickly arrive. What is more, by the law of large numbers, the more consumers that participate in the system, the better that probabilistic control is expected to perform. Note that we will never consider the amount of power consumed by an individual device in determining whether to switch that device off; this is done to help satisfy the perceived fairness constraint, because it means that a user’s experience will not be significantly affected by their coloring choices.

3) The algorithm must be able to quickly arrive. What is more, by the law of large numbers, the more consumers that participate in the system, the better that probabilistic control is expected to perform. Note that we will never consider the amount of power consumed by an individual device in determining whether to switch that device off; this is done to help satisfy the perceived fairness constraint, because it means that a user’s experience will not be significantly affected by their coloring choices.

4) The algorithm must be able to quickly arrive. What is more, by the law of large numbers, the more consumers that participate in the system, the better that probabilistic control is expected to perform. Note that we will never consider the amount of power consumed by an individual device in determining whether to switch that device off; this is done to help satisfy the perceived fairness constraint, because it means that a user’s experience will not be significantly affected by their coloring choices.

5) The algorithm must be able to quickly arrive. What is more, by the law of large numbers, the more consumers that participate in the system, the better that probabilistic control is expected to perform. Note that we will never consider the amount of power consumed by an individual device in determining whether to switch that device off; this is done to help satisfy the perceived fairness constraint, because it means that a user’s experience will not be significantly affected by their coloring choices.

6) The algorithm must be able to quickly arrive. What is more, by the law of large numbers, the more consumers that participate in the system, the better that probabilistic control is expected to perform. Note that we will never consider the amount of power consumed by an individual device in determining whether to switch that device off; this is done to help satisfy the perceived fairness constraint, because it means that a user’s experience will not be significantly affected by their coloring choices.
ignore power coloring and consider all consumption to be in a single category. This means that each device is an actual electrical appliance.

Each device $i$ holds the following state information:

- $n$, the total number of devices on the network
- $Q_t$, the current global target (i.e., total supplied power).
- $Q_m$, the total measured power consumption on the network (which we wish to control to equal $Q_t$)
- $Q_d$, the total power demand from all the devices on the network
- $d_i$, the device’s own measured power demand
- $m_i$, the device’s own measured power consumption (we assume it zero when off, $d_i$ when on)
- $t_{flip}$, the time remaining until the device is next allowed to flip a coin to decide whether to change state.

Each device is also assumed to have a clock that measures elapsed time with no more than a small error, and to evaluate its control algorithm frequently. Whenever $t_{flip}$ reaches zero, a device will execute its probabilistic control step, then reset $t_{flip}$ to an expected value of $T_{flip}$ (see Section III-C for more on how the value is chosen).

The global state ($n$, $Q_t$, $Q_m$, and $Q_d$) is assumed to be provided by a distributed aggregation algorithm, and is therefore delivered at a lag. This lag cannot be less than $\Omega(diameter/c)$, where $diameter$ is the number of hops across the network and $c$ is the maximum speed of information flow per hop. In the ColoredPower implementation for this paper, we use a distance-based spanning tree as our aggregator. We chose this aggregator for simplicity and its $\Theta(diameter/c)$ lag, but it is not robust to network changes and we expect that much more robust aggregator is both possible and necessary for a real deployment.

**B. Simple Local Probabilistic Control**

The simplest probabilistic control for $Q_m$ to track $Q_t$ is to have each device $i$ flip a coin with probability $p_{simple} = \frac{Q_t}{Q_d}$ of turning heads. If the coin falls heads, the device chooses to turn on and consume $d_i$ power, if not, it chooses to turn off and consume 0 power. If each device does this the expected total consumption will be

$$E[m_i] = p_{simple} \times d_i$$
$$E[m_i] = \frac{Q_t}{Q_d} \times d_i$$

$$E[Q_m] = E[\sum_i m_i]$$
$$E[Q_m] = \sum_i E[m_i]$$
$$E[Q_m] = \sum_i \frac{Q_t}{Q_d} \times d_i$$
$$E[Q_m] = \frac{Q_t}{Q_d} \times Q_d$$
$$E[Q_m] = Q_t$$

For example, consider 100 devices, each consuming 1 unit of energy (thus the global demand is 100), and the global target is 70. If each device turns on with 70% probability then our expected global power consumption is equal to the global target.

There are two major problems with this design:

1) From iteration to iteration of the local control, there is nothing that prevents an individual electrical device from switching on and off very rapidly; this is not
an acceptable solution since the rapid oscillation of a single device is undesirable.

2) This simple probabilistic control does not account for the variance that comes with randomization or the fact that different devices consumer different amounts of power. It is thus unlikely that the global consumption actually hits exactly $Q_t$.

C. Timed Local Probabilistic Control

To address the first problem we add timers to every device that ensure that once a device turns on or off, it stays that way for a period of time. We thus introduce the following new state for each device:

- $t_{\text{fall}}$ the time remaining until the device is allowed to decrease its power consumption $m_i$
- $t_{\text{rise}}$ the time remaining until the device is allowed to increase its power consumption $m_i$

Every time a device increases power consumption, $t_{\text{fall}}$ gets reset to an expected value of $T_{\text{fall}}$. Similarly, if a device decreases consumption, $t_{\text{rise}}$ gets reset to an expected value of $T_{\text{rise}}$. Those devices that have recently changed state are thus “timed out” and cannot change state again in the opposite direction soon.

When a timer $t_x$ is reset to an expected value of $T_X$, it is important that there be a large amount of variance in the value it is reset to. This effectively desynchronizes devices form one another, ensuring that in the expected case, there are always some devices that are allowed to change their state, and therefore some demand flexibility. Therefore, at each reset of a timer $t_x$, its new value is selected from a uniform random distribution on the interval $\left[\frac{T_x}{2}, \frac{3T_x}{2}\right]$.

With the addition of these timers, our prior simple probabilistic control will no longer operate correctly, since timed-out devices are capable of changing state. We thus need to adjust $p_{\text{simple}}$ in some way that will depend on the number of devices that are not-timed-out, in order to maintain the accuracy of our expected global power consumption. To do this we aggregate new global state information about the state of the network. Each device is classified into exactly one of three states (Figure 5):

- **1-fixed devices** are devices unable to fall at that instant (i.e. recently turned on). The total demand for these devices is denoted by $Q_1$
- **0-fixed devices** are devices unable to rise at that instant (i.e. recently turned off). The total demand for these is denoted by $Q_0$
- **flippable devices** are the remainder: those that are available for local probabilistic control. The total demand for these is denoted by $Q_f$, and is a measure of the current demand flexibility of the system.

The 1-fixed and 0-fixed terminology comes from the status of the devices as on(1) or off(0). Note that by definition, $Q_1 + Q_0 + Q_f = Q_d$

As opposed to the Simple Local Probabilistic Control, where the demand flexibility is $Q_a$, the demand flexibility is $Q_f$, reflecting the fact that the control itself temporarily impinges on flexibility. Further, $Q_1$ demand is already fixed as on, which means that $Q_1$ power is already being consumed regardless of the control at that moment. In order for the expected consumption to be $p_{\text{simple}} \times Q_d$, the devices modified local probabilistic control:

$$p_{\text{timed}} = \frac{(p_{\text{simple}} \times Q_d) - Q_1}{Q_f}$$

Each device which is not timed out flips a coin with probability $p_{\text{timed}}$. If the coin falls heads, the device turns on and consumes $d_i$ power; if not, it turns off and consumes 0 power. It is easy to see that if $Q_f = Q_d$, i.e. all devices are flippable, then $p_{\text{timed}} = p_{\text{simple}}$. Note also that if there is not enough demand flexibility to achieve the target, $p_{\text{timed}}$ will be outside of [0, 1]; in this case, we clip it to [0, 1] to get as close as possible to the target.

In general, we have:

$$E[m_{i, \text{flippable}}] = p_{\text{timed}} \times d_i$$
$$E[m_{i, \text{1-fixed}}] = d_i$$
$$E[m_{i, \text{0-fixed}}] = 0$$
$$E[Q_m] = Q_1 + \sum_{i \in \text{flippable}} E[m_i] = Q_1 + \sum_{i \in \text{flippable}} p_{\text{timed}} \times d_i = Q_1 + p_{\text{timed}} \times Q_f = Q_t$$

This timer dependent and census-adjusted local probabilistic control give us the desired expected global power consumption, while neatly allowing each device to be switched between on and off at a non-intrusively low frequency.
D. Timed Local Probabilistic Feedback Control

We still need to address the problem of variance. We will do this with feedback control based on the global consumption \( Q_m \). For this paper, we have chosen to use a simple PID controller. This long-established generic controller, which incorporates a (P)roportional term to address instantaneous error, an (I)ntegral term to address accumulating “past” error, and a (D)erivative term to predict likely “future” error, is a simple and well-understood starting point for adding feedback control to a system (though we shall see in Section IV-D that a more sophisticated controller will eventually be needed).

At any point in time, the error in tracking is given by \( \Delta(Q) = Q_t - Q_m \). Using a PID controller, the desired error correction is:

\[
\Delta_{PID}(Q) = G_P \cdot \Delta(Q) + G_I \int_0^t \Delta(Q) \, dt + G_D \cdot \frac{d}{dt} \Delta(Q)
\]

This can be converted into a local probability of change in much the same way as before: \( p_{feedback} = \frac{\Delta_{PID}(Q)}{Q_t} \). The expected new value after an expected set of flips (from time \( t_0 \) to time \( t_1 = t_0 + T_f \)) is thus:

\[
E[Q_m(t_1)] = Q_m(t_0) + E[\sum_{i \in f lippable} p_{feedback} \cdot d_i] = Q_m(t_0) + p_{feedback} \cdot Q_f(t_0) = Q_m(t_0) + \Delta(Q)
\]

If the gains for the PID controller are stable with respect to the delay in obtaining the aggregate state variables, then may be expected to converge to \( Q_t \). Unusual in the design of a controller, however, it is important that the control be significantly overdamped. This is because “timed out” devices generally make the system very slow to recover from overshoots. Thus the controller must be overdamped enough that it approaches the target in a series of steps, adjusting the flipping probability using the census as well as the error at every step, and where the probability of random variance causing a significant overshoot on any step is small.

E. Adapting to a four color system

With Timed Local Probabilistic Feedback Control, we now have an algorithm that can control power for a single PACEM “color.” All that remains is to extend it to a multiple-color system. Note that while we discuss this algorithm in terms of the four colors in the PACEM proposal, it generalizes trivially to a \( k \)-color algorithm.

Each appliance is set to exactly one color. Each household is represented by a demand controller device holding the aggregate information of the different colored energy demand of all the appliances within.

To generalize from one to multiple colors, we introduce the concept of Range. The Range is always a real number between 0 (black) and 3 (green), and serves as a numerical relation between an amount of power and the total power demand, which is pre-divided into the four colors. Let \( Q_d = Q_d^1 + Q_d^2 + Q_d^3 + Q_d^4 \) denoting the division of the total demand into the four colors, green, yellow, red, and black, respectively. Each household demand controller device similarly controls four different demands \( d_i = d_i^0 + d_i^1 + d_i^2 + d_i^3 \), and has four different kinds of local power consumption \( m_i = m_i^0 + m_i^1 + m_i^2 + m_i^3 \). Note that each \( m_i^c \) is a discrete block of power, i.e. \( m_i^c \in \{d_i^c, 0\} \). The maximum \( i \) for which \( m_i^c = d_i^c \) is the color \( c \) of the device, e.g. \( c = 2 \) would indicate the color “yellow.”

When a power quantity \( Q_x \) has a range of \( r_x \) this means that it includes all of the power “below” it:

\[
Q_x = (r_x - [r_x]) \times Q_d^{[r_x]} + \sum_{r \leq [r_x]} Q_d^r
\]

For example, a range of 1.3 would mean that \( Q_x \) contains all the power in the “red” and “black” blocks and 30% of the power from the “yellow” block.

The algorithm uses two ranges: the target range \( r_t \) corresponding to \( Q_t \) and the measured range corresponding to \( Q_m \) (see Figure 2). With regards to control, the fractional and integer portions are handled separately. The integer portion is simple: when \( [r_t] \) changes, every device in the entire block of power changes to be on or off (as appropriate) as soon as \( t_{fall} \) or \( t_{rise} \) allow the device to. This portion of control is naturally quite fast in achieving its goal.

Let’s look at tracking the fractional part. There is a \( Q_d^{[r_t]} \) which we need to track using only the \( m_i^{[r_m]} \) portion of power, since our integer tracking is already working to make sure that \( [r_t] = [r_m] \). The demand is \( Q_d^{[r_t]} \) and there is already some \( Q_m^{[r_t]} \) which is the power consumption within that block. We just need to use some local probabilistic control which will push \( Q_d^{[r_t]} \) toward \( Q_d^{[r_t]} \). This is exactly the problem that we solved using the Timed Local Probabilistic Feedback Control. Instead of \( \Delta(Q) \) we will introduce the

Figure 7: The distributed control algorithm, given the global desired and current consumption, produces a global command for the new level. This command must then be translated into a weighted coin-flip, to occur independently at each device, scaled to take into account the fact that the switching-frequency limitation means that some devices not currently controllable.
corresponding error in range,
\[ \Delta(r) = (r_t - \lfloor r_t \rfloor) - (r_m - \lfloor r_m \rfloor) \]
This can be plugged into the PID controller as before to produce a \( P_{\text{feedback}} \) which, when combined with integer control, completes the feedback controller.

The last detail to be filled in, handling of user overrides, is simple: When a user presses the override button on a device, it can never be controlled by the algorithm until the user stops the override. The demand from this device is then recolored as “black,” which is soon thereafter reflected in aggregate state variables.

Figure 8 summarizes the \textit{ColoredPower} algorithm. Each device receives aggregated data in the form of the global target, the global demand, and the global consumption, along with a census of demand flexibility. The device now infers the target range, measured range, and range-error using this input. The device goes through decision-tree based on a state table (Figure 9) that takes into account its local parameters: the timers, the local demand, the local measured consumption, etc. The integer part of the range tells the device what its minimum color should be, and the fractional part is converted into a probability with which it should turn on the color above the minimum. Finally, each device supplies the new local energy consumption and device state (which of the three census categories it falls into) into the aggregator, leading to an eventual update of the global state variables.

Figure 9: State table for \textit{ColoredPower}, based on the target range, measured range, and timers.

IV. EXPERIMENTAL VERIFICATION

In this section, we describe a series of experiments by which we verify that \textit{ColoredPower} behaves as desired. We have implemented \textit{ColoredPower} in Proto[9], a high level language for distributed algorithms, where programs are described in continuous regions of space and time, rather than individual devices. Proto depends on the \textit{amorphous medium abstraction}, which views a network of devices as an approximation of a computational material with a processor at every point. This continuous abstraction makes programs in Proto highly scalable: if a program works for a neighborhood, it is likely to work for an entire metropolitan area.

A. Experimental Setup and Parameters

Except where indicated, the following experiments are conducted using the following parameters:

- The network contains \( n = 100 \) devices. These devices are distributed randomly in a \( 100 \times 100 \) unit square. Each device has a communication radius of 50 units. Thus, the expected diameter of the network is 3.
- Each device has a demand profile of \((d^3_d, d^2_d, d^1_d, d^0_d) = (3, 6, 7, 4)\) units of power demand in the green, yellow, red and black blocks respectively. The total possible consumption in the base system is therefore \( Q_d = 100 \times (3 + 6 + 7 + 4) = 2000 \) units. This also means that \((Q^3_d, Q^2_d, Q^1_d, Q^0_d) = (300, 600, 700, 400)\).
We choose $T_{flip}$ randomly in the interval of $[2,8]$ seconds, with $E[T_{flip}] = 5$ seconds.

- We choose $T_{rise}$ and $T_{fall}$ randomly in the interval $[500,1500]$ seconds with $E[T_{rise}] = E[T_{fall}] = 1000$ seconds.

- The PID controller uses two sets of gains: $\{0.5,0.08,0.3\}$ and $\{0.4,0.1,0.4\}$, the two best performing values found via a heuristic parameter search.

- To prevent over-impact from accumulated error, integral error is given a window is 50 seconds, and an exponential backoff filter of coefficient 0.5.

- System state is sampled once every 10 seconds.

These parameters are not intended to reflect actual demand models, but to characterize controller performance; the choice of times for parameters and response goals, however, is guided by [10].

### B. Homogeneous Demand

We begin by verifying that the algorithm works correctly under homogeneous demand conditions. We examine behavior using two target profiles: square wave and sinusoidal. The square wave shows us the impulse response of the system and gives an estimate of behavior in worst case conditions of the energy grid, e.g. if a power plant suddenly fails, or a major transmission network failure causes effective demand to suddenly drop. The sinusoidal case shows the system’s response to smoother, incremental changes.

We tested impulse response using a square wave with a period of 8000 seconds, with one experiment for steps between every possible pair of colors except black (since the consumption cannot fall below red), using the following values for $Q_i$: 2200, 1800, 1400, and 500. Impulse response graphs for each pair are shown in Figure 11. The overall convergence times are shown in Table I, where we defined convergence time as the first time after which the measured consumption stays within 3% of the target for more than 300 seconds, choosing 3% because any smaller percentage would allow only a single device to be wrong in some situations. As can be seen, fall times are generally significantly better than rise times (due to an intentional bias in the construction of the feedback control), but in all cases the system begins responding rapidly and is nearly complete within 20 minutes.

We tested incremental tracking using sine waves with periods 100 to 4000, scaled and offset such that the peak is at 2000 ($Q_d$) and the trough is at 400 ($Q_d^1$). Each sine wave was run for 40000 seconds so that we get at least 10 periods worth of response data. Figure 12 shows a typical long-period response: good tracking on the falling curve and a long delay on the rising curve.

We further measure performance by the phase lag between the measured consumption and the target, determined by minimizing root mean squared error between the measured consumption and a sine wave with the target’s frequency and amplitude (Figure 13). At long periods, the system tracks well, improving for longer periods; below period 2000, when the half-wave period is shorter than the convergence time, tracking begins to break down, eventually failing completely at high frequencies.

### C. Heterogeneous Demand and Overriding

In the next set of experiments, we move closer to a real-world situation, in which users have different demand profiles and a small but changing percentage override the system, in order to verify that the simplifying assumptions used in the design of ColoredPower are not disrupted by a more general case.

To model heterogeneous demand, we change the demand profile from being fixed at $(3,6,7,4)$, to use $(d^3, d^2, d^1, d^0)$ such that each $d^i$ is an integer chosen at random between 0 and 10 (inclusive). We measure impulse response using a square wave as before, over 10 different randomly generated demand profiles. Results are shown in Figure 14(a) and Table II: we find that convergence times are comparable to...
For override, we model a small fraction over overriding by having each device make occasional independent decision of whether to override each color \( d^i \) (effectively adding them to “black”). The likelihood of override is fixed at 5% and the device decides on average every \( T_{\text{override}} \) seconds, where \( T_{\text{override}} \) is distributed identically to \( T_{\text{fall}} \) and \( T_{\text{rise}} \). Results are shown in Figure 14(b) and Table III: as can be seen, the mean behavior is the same as without override, but the worse case is higher, likely due to occasionally small perturbations.

D. Diameter Variance

Finally, we verify that the algorithm is scalable by increasing both the diameter of the network and the number of devices. For larger networks with increasing diameters, we expect that the performance ColoredPower will be better in terms of convergence time and accuracy for small steps in the global target (due to higher demand flexibility) but the lag time for a fast changing global target (like the sinusoidal family) will be progressively worse.

The experimental setup uses rectangular boxes of increasing area, with a fixed communication radius of 20. We use a fixed width \( x = 20 \) for these experiments, and a varying length \( y \) starting at 100. The number of devices on the network is equal to \( y \) so as to maintain a dense distribution.

Since \( x \) is small compared to \( y \) we can use an approximation of the true network diameter as the number of hops required to cover the length \( y \) of the box (density is high enough that the stretch from indirect travel is only a few percent [11]).

As can be seen from Table IV, performance improves significantly for larger numbers of devices, but falls again as lag rises. We expect that the degraded performance may be partly due to the PID gain parameters being unable to scale to arbitrary lag, and that a better controller may correct this.

Table II: Convergence Times for Heterogeneous Demand

<table>
<thead>
<tr>
<th>P.I.D</th>
<th>Fall Convergence Time Mean ± Std.Dev.</th>
<th>Worst</th>
<th>Rise Convergence Time Mean ± Std.Dev.</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5,0.08,0.3</td>
<td>1240 ± 300 1690</td>
<td>1300 ± 520 1830</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4,0.1,0.4</td>
<td>1220 ± 420 1780</td>
<td>1300 ± 480 1780</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table III: Convergence times for heterogeneous demand with overrides

Table IV: Convergence times for varying diameter

<table>
<thead>
<tr>
<th>Diameter</th>
<th>Fall Convergence Time Mean ± Std.Dev.</th>
<th>Worst</th>
<th>Rise Convergence Time Mean ± Std.Dev.</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>450 ± 104 590</td>
<td>915 ± 45 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>400 ± 55 450</td>
<td>932 ± 54 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>388 ± 88 540</td>
<td>928 ± 38 970</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>792 ± 382 1120</td>
<td>910 ± 139 1130</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1138 ± 25 1170</td>
<td>865 ± 45 900</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 14: Comparison of homogeneous demand (top) and heterogeneous demand response. The graph showing heterogeneous demand (center) also has the different demand values marked with the appropriate colors. The graph showing heterogeneous demand with overrides allowed (bottom) includes the mean and std. dev. of the global demand values.

Figure 15: Lag times vs. period for sine wave response with heterogeneous demand and PID gains of 0.5, 0.08 and 0.3.

V. CONTRIBUTIONS

The ColoredPower algorithm provides a scalable mechanism for distributed shaping of small customer energy demand. By maintaining a globally distributed summary model of current system state, the algorithm allows managed devices to take independent probabilistic actions that rapidly adjust the total system consumption to match a desired level. This algorithm thus addresses one of the key obstacles to the deployment of small-customer energy demand management systems.

There are many areas in which the algorithm might be further improved. The feedback control is one obvious place: the gains might be tuned better, and a more sophisticated adaptive control could be substituted for the simple PID controller to provide better scaling with network diameter and to avoid variance-driven overshoots. We expect that the fairness model can also be adjusted to allow faster response by assuring that a smaller percentage of the devices are waiting for timeouts at any given time. More robust aggregation protocols are possible as well, though our previous investigations [12] have ruled out some apparently attractive options. Finally, an obvious next step toward the realization of PACEM is to deploy the algorithm on prototype devices, validating that it behaves as expected in a real network environment before moving toward test deployments with actual consumers.

ACKNOWLEDGMENTS

This work was partially funded by the MITEI Seed Fund.

REFERENCES


