Exploiting Home Automation Protocols for Load Monitoring in Smart Buildings

David Irwin, Anthony Wu[†], Sean Barker, Aditya Mishra, Prashant Shenoy, and Jeannie Albrecht[‡]
University of Massachusetts Amherst
Amherst College[†]
Williams College[‡]
{irwin,sbarker,adityam,shenoy}@cs.umass.edu, awu13@amherst.edu, jeannie@cs.williams.edu

Abstract

Monitoring and controlling electrical loads is crucial for demand-side energy management in smart grids. Home automation (HA) protocols, such as X10 and Insteon, have provided programmatic load control for many years, and are being widely deployed in early smart grid field trials. While HA protocols include basic monitoring functions, extreme bandwidth limitations (<180bps) have prevented their use in load monitoring. In this paper, we highlight challenges in designing AutoMeter, a system for exploiting HA for accurate load monitoring at scale. We quantify Insteon's limitations to query device status—once every 10 seconds to achieve less than 5% loss rate—and then evaluate techniques to disaggregate coarse HA data from fine-grained building-wide power data. In particular, our techniques learn switched load power using on-off-dim events, and tag fine-grained building-wide power data using readings from plug meters every 5 minutes.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; K.6.2 [**Management of Computing and Information Systems**]: Installation Management—*Performance and usage measurement*; J.7 [**Computer Applications**]: Computers in Other Systems—*Command and control*

General Terms

Design, Measurement, Management

Keywords

Home Automation, Load Monitoring, Smart Grid

1 Introduction

Recently, smart grid initiatives have focused research attention on improving the electric grid's efficiency.[1] Smart grids are important due to a long-term upward trend in energy costs and a growing consensus that carbon emissions contribute to climate change. Since home and office buildings represent a major fraction (73%) of grid consumption [7], the design of smarter buildings that interact with the smart grid to reduce consumption is also important. Smart buildings use *demand-side energy management* to self-regulate their energy footprint to reduce overall energy consumption and peak power usage, while better aligning consumption with renewable generation [17]. Demand-side management requires buildings to 1) *continuously monitor* the power consumption of electrical loads and 2) *remotely control* when and how much power each load consumes, e.g., to enable automated demand response. Although load monitoring and control are closely coupled in demand-side management, two disjoint sets of technologies have evolved to perform these tasks in smart homes.

On the monitoring front, researchers have developed numerous techniques to enable fine-grain tracking of electric usage at various spatial and temporal dimensions, e.g., [8, 9, 11, 12, 15]. Several past efforts focus on outlet-level monitoring, and in some cases control, of electric loads using wireless technologies, such as 802.15. These technologies are still expensive for typical homes, which may contain several tens to hundreds of outlets. For instance, while not commercially available, ACme meter components cost $85 plus time for assembly and calibration [10, 13], while Tweet-a-watt components cost $60 per meter plus time for assembly [5]. Similarly, the commercial PloggSE plug meter costs $215 per outlet [4]. A less expensive option is to deploy a single sensor at a home's electric panel to monitor aggregate usage, and use NILM techniques to disaggregate individual loads. While many inexpensive building-wide meters exist, e.g. the TED 5000 costs $200, accurate disaggregation of individual loads requires higher resolution data, e.g., 1000s of samples per second, than these meters typically support.

On the control front, Home Automation (HA) protocols, such as X10 and Insteon, were designed explicitly for remote load control. The protocols enable programatic actuation of outlets and switches hard-wired into a building and controlled from a central server, via command-line or remote web/smartphone interfaces, using the building's powerlines for communication. HA protocols are also mature standards: X10 was introduced in 1975 and Insteon in 2005. Since the protocols are embedded into "normal" power outlets and switches, they are inexpensive, with X10 and Insteon versions available for under $15 and $40, respectively. Once designed for hobbyists, HA has now become mainstream and

[1]This work is supported by NSF grants CNS-1143655, CNS-0916577, CNS-0855128, CNS-0834243, and CNS-0845349.

BuildSys'11, November 1, 2011, Seattle, WA, USA.

is widely used in smart homes that require automation or remote load actuation. Unfortunately, these protocols are not capable of tracking fine-grain power usage, since they only expose coarse-grain usage events, e.g., on-off-dim switch transitions or periodic outlet power consumption.

Thus, while monitoring and control systems are important for demand-side energy management, they have evolved independently using disjoint protocols and standards. In this paper, we examine how to unify a smart home's monitoring and control substrates into a single infrastructure. Rather than designing a new substrate from scratch or embedding control features into today's dedicated monitoring systems, we instead investigate how to exploit existing HA protocols and products to monitor power for individual loads. We adopt this approach for several reasons.

Commercial Availability and Cost. A variety of HA products are commercially available, including load control switches for appliances, lamps, wall switches, and outlets. Cost is an important consideration when deciding how to provide remote control and monitoring for a large set of electrical loads. As noted above, these products are inexpensive and have standard form-factors that fit today's buildings.

Backwards Compatible. Since many smart homes already use HA protocols, our approach will augment existing deployments with monitoring functions. HA protocols are already being widely adopted in early smart grid field trials [3].

Open Standards. Finally, both Insteon and X10 are open and well-tested standards that vendors are able to integrate directly into appliances that require more than simple load control switches, i.e., those with multiple power states and sophisticated setpoint control algorithms [14].

Despite the benefits, using HA-based protocols for monitoring poses significant challenges.

Scalability. Since HA protocols are intended for sending short control commands, such as on, off, and status, they were not designed for high bandwidth communication. In practice, data rates are often less than 180bps [6]. Further, the MAC layers for HA protocols do not employ "standard" features, such as collision avoidance. Thus, monitoring tens of electrical loads at high resolution is challenging. A key research challenge is *scaling* HA protocols to monitor large numbers of loads despite the protocol limitations.

Accuracy. HA protocols are only capable of monitoring on-off-dim state changes for switches or coarse-grain power usage for plug outlets. Thus, translating switch state change events and coarse plug power data into fine-grain power usage measurements represents another research challenge.

In this paper, we present AutoMeter, which addresses the scalability and accuracy challenges of using HA protocols for load monitoring. For scalability, we take empirical measurements from an actual home deployment that quantify the limits of HA protocols when monitoring and controlling large numbers of loads. We show that Insteon's protocol limits device status queries to once every 10 seconds to achieve loss rates less than 5%. To enhance scalability, we present the early design of a smart polling technique that adaptively determines the frequency to poll loads, based on their past usage patterns, without sacrificing fidelity. For accuracy, we present techniques that couple a building-wide

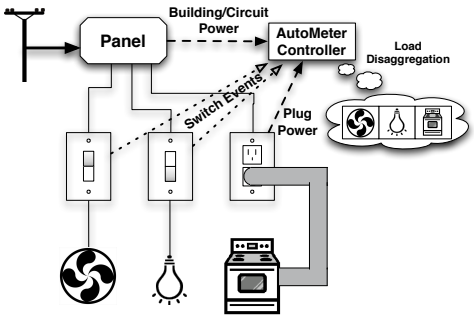

Figure 1. An overview of AutoMeter's architecture.

(or smart) meter with HA events to disambiguate energy usage of individual loads from aggregate usage measurements. For instance, we show that a proactive technique for learning switched load power over 93% accurate, while a reactive technique is accurate for loads with many events.

2 System Overview

The primary drawback in using HA protocols for load monitoring at scale is their extreme bandwidth limitations. While more recent powerline-based protocols, such as HomePlug [2], substantially increase bandwidth, they have not yet been applied to HA. Instead, these new protocols target high-bandwidth data from general Internet traffic and multimedia devices, e.g., HDTVs. We focus on Insteon in our home deployment, since it is generally more reliable and provides more bandwidth than X10.

Figure 1 depicts AutoMeter and its two types of Insteon-enabled sensors: *wall switches* and *plug power meters*. An Insteon-enabled wall switch sends a notification, via a PowerLine Modem (PLM), to a controller running on a server whenever someone changes the on-off-dim state at the wall. In contrast, the controller must explicitly query each plug power meter to determine its power consumption. We discuss details of our home deployment further in the next section. Below, we briefly summarize the Insteon protocol to highlight its limitations for scalable load monitoring.

2.1 Insteon Protocol

Insteon senders broadcast messages over the powerline, while receivers listen for messages and send acknowledgements upon receipt. The protocol limits transmissions to short intervals near where AC current crosses zero: electrical noise impairs communication and is at minimum during the zero crossing [6]. However, due to harmonic noise from power supplies or signal attenuation over long distances, it is still possible that a device may not receive a message.

To increase reliability and range, all Insteon devices also act as repeaters that automatically repeat messages they hear a fixed number of times, based on the *hops* field, as depicted in Figure 2. Additional hops effectively increase each message's length by a factor of (1+*hops*). The simple broadcasts and hops alleviate the need for complex routing protocols to transfer messages. The protocol avoids flooding and collisions when repeating, since all devices synchronize retransmissions using the 60Hz AC powerline frequency—each transmission begins exactly 800 microseconds before the zero crossing and ends exactly 1023 microseconds after the zero crossing. Thus, when repeating, all devices transmit

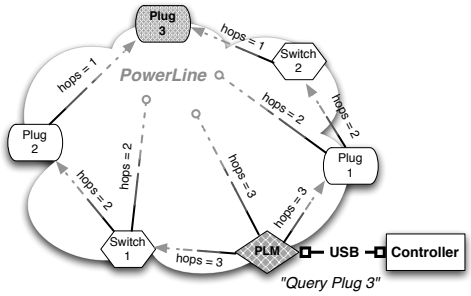

Figure 2. Depiction of the Insteon powerline protocol.

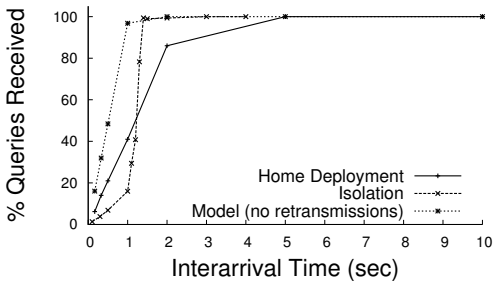

Figure 3. Insteon does not accommodate fast query rates for plug power meters.

the same data at exactly the same time for the same number of times. Note that Insteon *does not* prevent two devices from sending different messages at the same time, although, since all devices are repeaters, they inherently wait for in-transit messages to end before starting a new transmission.

The protocol supports two types of messages: 10 byte standard messages and 24 byte extended messages, which require 6 and 13 zero crossings to transmit, respectively. Since there are 120 zero crossings per second with 60Hz AC power, a standard message takes 50ms to transmit and an extended message takes 108.33ms, with no additional hops. While Insteon's maximum theoretical bandwidth is 2880bps, in practice, devices typically use three hops and acknowledgements, which reduces the maximum bandwidth by 16X to 180bps. In addition to repeated messages, a sender that does not receive an acknowledgement within a timeout will retransmit a message up to five times. For noisy lines that require retransmissions, actual bandwidth may be less than 180bps with three hops. Finally, Insteon uses RF communication (900MHz) to supplement the powerline, either to cross phases in multi-phase systems or to extend its range.

2.2 Protocol Limitations

We experiment with Insteon in our home deployment and in isolation to evaluate its performance, and determine an appropriate query rate for plug power meters. Setting the query rate presents a tradeoff: a rate too high saturates available bandwidth and results in the loss of either asynchronous switch notifications or load control commands; a rate too low results in less accurate plug power data. In our experiment, we vary the time between issuing plug meter queries, and determine both the percentage of queries lost (Figure 3) and the percentage of wall switch events lost (Figure 4). We ran each experiment for 10 minutes and turned switches on and off 50 times; the time between switch events was uniformly random between 0 and 20 seconds. We performed a similar experiment in isolation in an office building using adjacent outlets and replacing the wall switch with a PLM.

Each meter query includes three standard messages and one extended message (a standard query message from PLM $\rightarrow$ meter, an extended response message from meter $\rightarrow$ PLM with the power data, and a standard acknowledgement for each message). We use 3 hops for all experiments. We can only control the hops for the initial message sent from the PLM; the other messages always use 3 hops and originate from device firmware that we cannot change. As a result, altering the hops on the initial message does not significantly alter the results. Based on the protocol specification, each meter query takes 4*(0.05+0.05+0.1083+0.05) = 1.0333 seconds. We use the specification to model both the percentage of queries we expect to receive and the percentage of switch events we expect to lose for different query rates.

We model below the probability of losing a switch event (S_{lose}) as a function of the interarrival time of plug meter queries (T_i) and the length of an individual meter query ($T_q = 1.0333$), assuming no retransmissions from losses. Our model also assumes that when two transmissions collide, e.g., a switch event and a meter query, the device physically closer to the PLM is successful in transmitting.

$$S_{lose} = \frac{T_q}{2 * T_i} = \frac{0.5166}{T_i} \quad (1)$$

We also model the probability of receiving a query $Q_{receive}$ as the function below. If we issue queries at an interval greater than the query length, then we expect to receive every query. For intervals less than the query length, we expect queries to increasingly collide with each other.

$$Q_{receive} = \begin{cases} 1 & : T_i > 1.0333 \\ \frac{T_i}{T_q} = \frac{T_i}{1.0333} & : T_i < 1.0333 \end{cases}$$

Figure 3 demonstrates that performing our experiment in isolation results in a massive drop in the percentage of meter queries received once the query interarrival time hits the protocol's saturation point at 1.0333 seconds. The drop is more sudden than the model, since the model does not account for retransmission of lost messages, which immediately collapses the channel. Our home experiment shows query losses slightly before the saturation point, likely due to losses from powerline noise, collisions with switch events, and the resultant retransmissions. Figure 4 shows the percentage of wall switch events lost during the experiment. We lose less switch events after the saturation point at 1.0333 seconds in our home deployment than the model indicates. For the infrequent switch events, retransmissions of lost messages increases the percentage the controller receives.

2.3 Smart Polling

Our results motivate a more efficient scheduling approach for querying plug meters. We are currently investigating a smart polling technique that schedules plug meter queries to bound the amount of energy not detected by AutoMeter. Currently, in our home deployment, we issue queries in round-robin fashion every 10 seconds, which reduces the probability of losing a wall switch event to near 5%. Since our deployment currently has 30 plug meters,

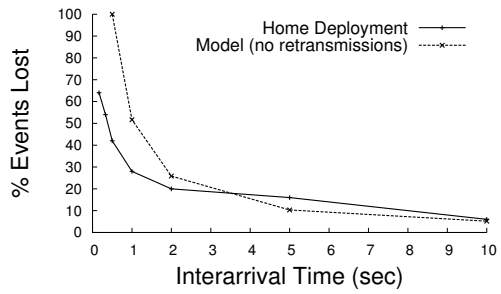

Figure 4. Asynchronous wall switch events may collide with plug meter queries, causing losses.

we query each plug meter once every 5 minutes. The insight behind smart polling is that continually querying loads that remain in a single power state, e.g., off or on, for long periods of time wastes precious bandwidth. Instead, smart polling tracks the maximum power consumption P_{max} for each load over time, and determines per-load query rates based on an energy bound E_{miss} that limits the amount of energy we may miss by not querying each load over a specific interval. Thus, smart polling queries larger loads, e.g., microwaves, at higher rates than smaller loads, since they contribute more to overall energy consumption.

As a simple example, if we compute the time between queries for each device as E_{miss}/P_{max}, then with $E_{miss} =$ 0.05kWh (about 0.10% of our home's average daily energy consumption) a 3kW microwave requires a monitoring interval of 1 minute to ensure that we detect any power state change that contributes at least 0.05kWh. In contrast, a LCD television that increases consumption by 200W when turned on requires a monitoring interval of 15 minutes. Since only a small number of plug loads in our deployment draw more than 1kW and most draw under 400W, monitoring intervals on the order of minutes should be sufficient in most cases to ensure a small bounded energy error. We are also exploring optimizations to this basic approach that modify each monitoring interval based on the duration and period of a device's power state changes, i.e., to query a periodic load only once per period. We believe smart polling will increase both the accuracy of coarse plug meter data, as well as the number of devices we are able to effectively monitor.

3 AutoMeter Deployment

We deployed our AutoMeter prototype in an average 3-bedroom, 2-bath house. The house has 34 wall switches, which control lights and exhaust fans. We replaced 30 of these mechanical wall switches with 20 Insteon SwitchLinc Relays and 10 Insteon SwitchLinc Dimmers. The 30 switches control 24 loads, since the house has two 4-way switches and two 3-way switches. As discussed below, we use a TED meter to monitor other switches. We use 30 Insteon iMeters to monitor plug loads in the home. The iMeters monitor all but 12 of the home's permanent plug loads. The unmonitored loads, e.g., night lights, electric toothbrushes, etc. consume little power in aggregate. We use a TED 5000 in the home's electrical panel to measure building-wide power consumption each second. The TED transmits data over the powerline to a gateway server. Since TED is also able to monitor power for 6 circuits using additional CTs, we use it to monitor loads not connected to SwitchLincs or iMeters, including a clothes dryer, garbage disposal, dishwasher, basement lights, HRV duct heater, and the electrical components of the gas furnace, including an exhaust fan. In total, the sum of the energy use from the plugs and switches is 96.7% of the energy use of the TED over the last two weeks.

Name	*Proactive*	*Reactive (#)*	*Actual*
kitchen:lights:dim	257W	290W (62)	260W
kitchen:sink	67W	69W (15)	65W
kitchen:lights1:dim	190W	192W (13)	195W
hall:lights1:dim	193W	39W (5)	195W
guest:lights:dim	255W	279W (10)	260W
guestbath:fan	51W	147W (36)	50W
guestbath:overheadlight	101W	100W (107)	100W
guestbath:sinklight	57W	60W (55)	60W
livingroom:dininglights:dim	128W	38W (25)	130W
livingroom:firelights:dim	148W	925W (8)	130W
livingroom:sideporch:dim	121W	850W (7)	130W
livingroom:lights1:dim	255W	361W (9)	260W
livingroom:lamp	17W	18W (61)	18W
frontporch:light	12W	185W (16)	20W
stairs:light1	72W	70W (20)	65W
masterbath:overheadlight	102W	100W (147)	100W
masterbath:fan	54W	110W (114)	50W
masterbath:sinklight	58W	59W (313)	60W
master:lights:dim	256W	26W (19)	260W
master:closet:a	12W	20W (57)	20W
master:closet:b	12W	9W (15)	20W
bedroom:lights:dim	254W	258W (21)	260W
bedroom:maincloset	18W	19W (14)	20W
bedroom:linencloset	22W	20W (38)	20W
bedroom:closet	22W	22W (35)	20W

Table 1. Table of switch power using proactive and reactive learning versus actual power consumption.

We implement AutoMeter's controller on a low-power and compact GuruPlug server. The server attaches to an Insteon PLM, which plugs into a standard outlet, via USB. Our software leverages the open-source plmtools package, which includes the `plmsend` and `plmcat` programs to send and receive raw Insteon messages using the PLM. We wrote an Insteon monitoring daemon using these programs to 1) detect asychronous notifications broadcast on the powerline whenever a switch turns on, off, or dims, and 2) continuously query the iMeters every 10 seconds. Since the switch notifications do not encode the dim percentage, our daemon issues a status query to determine it whenever the dim level changes. Unfortunately, the commercial HouseLinc software that supports the iMeter is not designed for constant monitoring, since it forces users to manually enter daily events that specify iMeter query times, which must be at least one minute apart. Thus, we reverse-engineered the iMeter protocol and modified the `insteon` command-line program in the plmtools package to support querying iMeter power, as well as sending asynchronous messages, i.e., not waiting for a reply from plmcat. The modifications allow us to issue iMeter queries at arbitrarily fast rates; we use this functionality for the experiments in the previous section.

The controller stores a timestamped record of each switch event and plug meter power usage in a SQLite database. We store the devices's name and the event timestamp, in addition to either the on-off-dim state between 0 and 100 or the plug power consumption. The controller also fetches the second-level TED data from TED's webserver, and stores it in the database. The controller uploads its

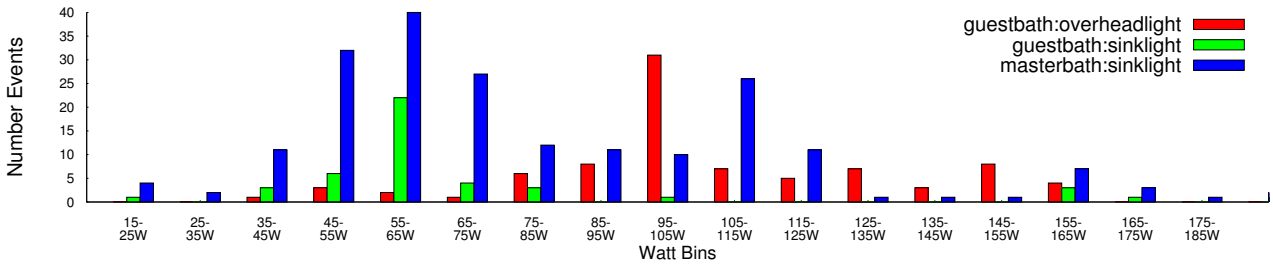

Figure 5. Histogram of events for each 10W power bin for selected switches using the reactive learning technique.

SQLite database to a centralized off-site MySQL database each morning for long-term storage. We plan to release AutoMeter's software as open-source for others to use, including the updated plmtools package.

4 Load Disaggregation

We explore disaggregation techniques for Insteon wall switches, which do not report power consumption, and plug meters, which have coarse query rates. For disaggregation, we add a device column to our database of building-wide power consumption data: the column specifies the name of the switch, plug meter, or circuit meter causing the subsequent power variation. Our goal is to tag the building-wide power data with device names whenever it changes. Since switch data does not specify the load's power consumption, we first develop techniques to learn switch power. We then discuss tagging building-wide power data with specific switch, circuit, and meter names.

4.1 Learning Switch Power

While individual switched loads (primarily lights) do not individually consume much power, they are the 12th largest load in our home in aggregate. If we remove summer-only loads, e.g., A/Cs, fans, then the switched loads are the 5th largest load. Recent estimates attribute 5-10% of home energy and 20-50% of building use to lighting [1].

Since switches send notifications every time they turn on, off, or dim, learning a switched load's power consumption should be straightforward: simply record the change in the building-wide power data whenever a switch changes state. Learning switch power is more complicated for two reasons: 1) multiple power events may occur within the building meter's monitoring granularity and 2) power and timing errors may occur in the building meter. In particular, we observe frequent timing errors that delay new power readings due to communication problems, since TED timestamps readings only after they have been successfully transmitted to the gateway over the powerline. While we observe simultaneous events, monitoring building-wide power every second mitigates their impact. Simultaneous events are a more significant issue for coarser monitoring intervals. We discuss both *proactive* and *reactive* techniques to learn switch power consumption that is robust to both coarse data and data errors.

In our proactive approach, we write a simple program to remotely toggle each switch one by one from the Insteon PLM, and observe the change in building power 2 seconds before and 10 seconds after toggling. For the experiment, we turn off most loads in the home to decrease simultaneous power events from other devices and data errors, which are proportional to the home's total load. Table 1 reports the power for each switch, as well as the switched load's rated power, and shows the approach is over 93% accurate on average across all loads. While the proactive approach is accurate, not all buildings will be able to shutdown most loads to reduce errors and cycle through every load in order to determine power usage. Thus, we also explore a reactive approach that learns power usage over time based on collected data.

The approach also computes the change in building-wide power whenever a switch changes state; again, we use 2 seconds before and 10 seconds after the change. We normalize the power step by a switch's dim level: if the dim level is 50% then we multiply the power step by two. We confirmed the linear relationship between power and dim level by recording the change in building-wide power as we vary the dim level from 1 to 100. Due to power and timing errors or other loads changing their power consumption, the power step over the interval will not always correspond to a switch's power consumption. However, our premise is that over long periods with many events, the plurality of power steps will be near the actual power consumption of the switch, since building loads alter their power states at human time-scales, e.g., minutes to hours. Thus, our reactive approach groups every observed power step for each state change for a given switch into bins, e.g., 5-15W, 15-25W, 25W-35W, etc. We then select the bin with the most events, average its values, and record that value as the switch's power consumption.

Table 1 shows the results of the reactive approach for 2 weeks of data, as well as the total number of switch events in parenthesis.[2] We find that the reactive approach is accurate for switches with many events over the 2 week period, and less accurate for rarely-used switches. Interestingly, the approach is not accurate for the exhaust fan in each bathroom, since they nearly always change state at the same time as a 100W overhead light or a 60W sinklight. We are currently augmenting the technique to identify these correlated switches. Figure 5 shows a histogram of the number of events in each bin for three of the thirty switches. The data demonstrates how power and timing errors in the building data, as well as simultaneous power events, cause a wide range of power values for each switch event, which complicates learning switch power.

4.2 Tagging Power Variations

Tagging power variations in the building-wide data should also be straightforward: simply observe the interval

[2]The table has less than 30 switches, since we only learn a single value for each set of multi-way switches.

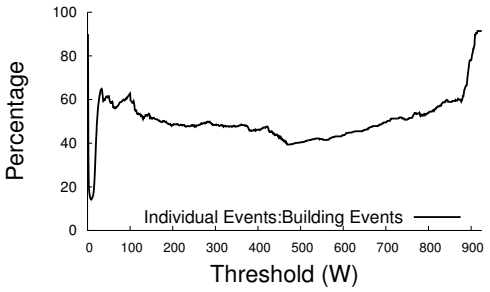

Figure 6. We use AutoMeter's switch, plug, and circuit meters to tag power variations in building-wide data.

over which we detect a state change and tag the point in the building-wide data that matches the state change. However, coarse plug data, in addition to power and timing errors, also complicates tagging events. To quantify how well we are able to tag data using the straightforward approach, Figure 6 shows the number of power events from switch, plug, and circuit meters we are able to tag as a percentage of the total number of events in the building trace, for different power thresholds. For each threshold value on the x-axis, the y-axis shows the percentage of tagged power events $\geq x$ in the individual switch, plug, and circuit meters, as compared to the building-wide meter. The figure demonstrates that our plug, switch, and circuit meters have nearly the same events as the building-wide meter as the threshold approaches 900W. For 30W-900W thresholds, the number of events in the individual meters is 40-60% of the events in the building-wide data.

Some of the missing events are due to power and timing errors in the building trace, while others are due to the 5-minute granularity of our plug meter data. For instance, TED's stated error is 2% of the total load; if the load is 3kW or greater, power variations of 60W or greater may be the result of meter inaccuracy. To mitigate errors, we are upgrading to a building-wide meter that timestamps readings at the point of measurement, uses a reliable transport protocol to send them to our server, and supports additional, and more appropriately-sized/accurate, circuit CTs to aid in disaggregation. While smart polling should improve the granularity of plug meter data, we are also investigating NILM techniques using individual circuits in conjunction with our coarse plug meter data, similar to those in [15].

5 Related Work

There exist a range of systems for monitoring the power consumption of electrical loads. These systems present various tradeoffs in accuracy, monetary cost, installation time, and calibration overhead. Early work on NILM recognized the difficulty and cost of instrumenting every individual building load [9]. Thus, NILM focuses on algorithms for disaggregating building-wide power to extract individual loads. While useful, NILM also presents challenges, including collecting accurate load power signatures and distinguishing loads with similar signatures.

To address NILM's challenges, a recent approach augments building power meters with heterogeneous sensors, as well as strategically-placed circuit and plug meters [11, 12, 15]. The additional data aids in distinguishing events in building-wide power data. Another approach is single-point sensing [8], which monitors AC power to detect precise power signatures at high frequencies, and associate them with events from specific loads. Our work shows that associating these events with power data from a building-wide meter also presents challenges. While existing approaches aid in disaggregation, they do not address load control. Unlike load monitoring, control requires integrating additional communication and switch hardware with devices. Since we target AutoMeter for smart buildings with HA-driven load control, it complements dedicated monitoring systems.

An advantage of AutoMeter for researchers is its use of widely available commercial out-of-the-box hardware and open-source software, rather than custom-built research prototypes. One goal of AutoMeter is to support higher-level smart grid research, e.g., developing load scheduling algorithms, improving NILM via machine learning, strengthening smart meter privacy, etc. At $40 each, purchasing 100s of Insteon devices is within the bounds of a modest research budget—our deployment, including 30 switches, 30 plug meters, a GuruPlug, and a TED meter, cost $3025.

6 Conclusion and Future Work

In this paper, we demonstrate Insteon's limitations for load monitoring, and evaluate straightforward load disaggregation techniques using data from an operational deployment. To further increase AutoMeter's scalability and accuracy, as part of future work, we are experimenting with smart polling to collect more accurate plug meter data, as well as improved disaggregation techniques. We are also extending our approach to buildings larger than single-family homes. Large buildings are more challenging, since they have many more loads (resulting in lower query rates) and longer powerlines (resulting in higher loss rates). While originally targeted for residential homes, recent work suggests that powerline communication is applicable to larger buildings [16].

7 References

[1] Energy Use for Lighting. http://www.dmme.virginia.gov/DE/ConsumerInfo/HandbookLighting.pdf.
[2] HomePlug Powerline Alliance. http://www.homeplug.org/home/.
[3] Insteon for the Smart Grid. http://www.insteonsmartgrid.com.
[4] Plogg Wireless Energy Management. http://www.plogginternational.com.
[5] Tweet-a-Watt. http://www.ladyada.net/make/tweetawatt/.
[6] Insteon: The Details. www.insteon.net/pdf/insteondetails.pdf, 2005.
[7] U.S. Department of Energy. Building Energy Data Book. http://buildingsdatabook.eere.energy.gov/, 2010.
[8] S. Gupta, M. Reynolds, and S. Patel. Electrisense: Single-Point Sensing Using EMI for Electrical Event Detection and Classification in the Home. In *UbiComp*, 2010.
[9] G. Hart. Nonintrusive appliance load monitoring. *IEEE*, 80(12), December 1992.
[10] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and Implementation of a High-Fidelity AC Metering Network. In *IPSN*, 2009.
[11] X. Jiang, M. V. Ly, J. Taneja, P. Dutta, and D. Culler. Experiences with a High-Fidelity Wireless Building Energy Auditing Network. In *SenSys*, 2009.
[12] Y. Kim, T. Schmid, Z. Charbiwala, and M. Srivastava. Viridiscope: Design and Implementation of a Fine Grained Power Monitoring System for Homes. In *UbiComp*, 2009.
[13] S. Lanzisera. The "Other" Energy in Buildings: Wireless Power Metering of Plug-in Devices. Environment Energy Technologies Division Seminar, Lawrence Berkeley National Labs, June 17 2011.
[14] J. Lu, T. Sookoor, V. Srinivasan, G. Ge, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes. In *SenSys*, 2010.
[15] A. Marchiori and Q. Han. Using Circuit-Level Power Measurements in Household Energy Management Systems. In *BuildSys*, 2009.
[16] P. Pannuto and P. Dutta. Exploring Powerline Networking for the Smart Building. In *IP+SN*, 2011.
[17] J. Taneja, D. Culler, and P. Dutta. Towards Cooperative Grids: Sensor/Actuator Networks for Renewables Integration. In *SmartGridComm*, 2010.