ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

Automated Measurement and Signaling Systems for the Transactional Network

Mary Ann Piette
Richard Brown
Phillip Price
Janie Page
Jessica Granderson
David Riess
Stephen Czarnecki
Girish Ghatikar
Steven Lanzisera

Building Technology and Urban Systems Department
Environmental Energy Technologies Division

December 2013

Prepared for the U.S. Department of Energy
under Contract DE-AC02-05CH11231

REPORT TO DOE

Automated Measurement and Signaling Systems for the Transactional Network

Mary Ann Piette
Richard Brown
Phillip Price
Janie Page
Jessica Granderson
David Riess
Stephen Czarnecki
Girish Ghatikar
Steven Lanzisera

December 31, 2013

Prepared for

U.S. Department of Energy

under Contract DE-AC02-05CH11231

LAWRENCE BERKELEY NATIONAL LABORATORY

operated by

UNIVERSITY OF CALIFORNIA

for the

UNITED STATES DEPARTMENT OF ENERGY

under Contract DE-AC02-05CH11231

Acknowledgements

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors wish to recognize George Hernandez, Senior Advisor to the Building Technologies Office of the US Department of Energy, for his support and assistance in this work. The authors would also like to recognize the in-kind cost-share contribution from EnerNOC through their provision of OpenADR 2.0 open source code and technical assistance to integrate the OpenADR client and server with the VOLTTRON Lite™ platform.

Abstract

The Transactional Network Project is a multi-lab activity funded by the US Department of Energy's Building Technologies Office. The project team included staff from Lawrence Berkeley National Laboratory, Pacific Northwest National Laboratory and Oak Ridge National Laboratory. The team designed, prototyped and tested a transactional network (TN) platform to support energy, operational and financial transactions between any networked entities (equipment, organizations, buildings, grid, etc.). PNNL was responsible for the development of the TN platform, with agents for this platform developed by each of the three laboratories. LBNL contributed applications to measure the whole-building electric load response to various changes in building operations, particularly energy efficiency improvements and demand response events. LBNL also provided a demand response signaling agent and an agent for cost savings analysis. Both LBNL and PNNL demonstrated actual transactions between packaged rooftop units and the electric grid using the platform and selected agents. This document describes the agents and applications developed by the LBNL team, and associated tests of the applications.

Contents

Figures

Tables

Transactional Network Platform Overview

The Transactional Network (TN) project, funded by the Department of Energy's (DOE's) Building Technologies Office (BTO), is a multi-laboratory effort lead by Pacific Northwest National Laboratory (PNNL), with Lawrence Berkeley National Laboratory (LBNL) and Oak Ridge National Laboratory (ORNL) also contributing to the effort. This report provides a summary of the LBNL work to date. LBNL designed, prototyped and tested components of this platform related to measuring system response to various planned modifications to the building operations. These modifications include energy efficient control strategies and automated demand response events.

Building loads constitute a large proportion of the overall load on the electric grid, consuming about 70% of total electricity use in the United States. The TN is intended to support energy, operational and financial transactions between networked entities (equipment, organizations, buildings, grid, etc.). The underlying platform of the Transactional Network consists of the PNNL developed VOLTTRON Lite™ (VL) agent execution software and a number of agents that perform specific functions (fault detection, demand response, weather service, logging service, etc.). VL serves as a single point of contact for interfacing with devices (building equipment, power meters, etc.), external resources, and platform services such as data retrieval and archive. In the initial phase, the focus is on rooftop units (RTUs) for small commercial buildings. For more details on the platform, please refer to the PNNL report on VOLTTRON Lite™ (Haack et al. 2013).

The TN Platform is designed to facilitate "transactive energy" systems and markets. At present there are several somewhat divergent definitions of Transactive Energy. The GridWise Architecture Council defines a formal framework for Transactive Energy,that includes both economic mechanisms and control mechanisms. An alternate definition of Transactive Energy arises from TeMIX efforts, where Transactive Energy consists of frequent, small, easily-understood automated transactions between buyers and sellers. Buyers and sellers may be generators, loads, storage, or traders with no actual delivery and metering. Transactive energy as used here refers to techniques for managing the generation, consumption, or flow of electric power within an electric power system through the use of economic or market based constructs while considering grid reliability constraints. A transactional network platform supports energy, operational and financial transactions between any networked entities (equipment, organizations, buildings, grid, etc.), according to Katipamula et al., 2013.

LBNL developed new software to operate with the VL platform that demonstrated the capability of both the LBNL and PNNL transactive applications at a building at LBNL. This report begins with a summary of the agents developed as part of the project. We describe their design, input data requirements, output and function. We then present an example of the software sequence. The next section provides sample results from an implementation of the VL agents deployed at a test site at LBNL. We end the report with a discussion of the results and a summary of key findings and next steps. The appendices provide additional details on the software systems developed in this project.

Agents

Transactional Network Agent Design, Development and Testing

This report describes the VL agents developed by LBNL. These agents provide support services to augment and complement the agents from PNNL and ORNL. The PNNL agents are further described in Katipamula et al., 2013. These agents may reside directly on the VL platform, on the equipment being controlled, on a local building controller, or in the cloud, hosted by a remote internet-based server.

LBNL's agents in the TN focus on characterizing the energy savings associated with short- or long-term operational changes in a building. A demand response (DR) event would be an example of a short-term change whereas an energy efficiency (EE) measure would be a long-term change. Demand response is a change from normal patterns of electric energy consumption by end-use customers in response to changes in electricity price or incentive payments designed to induce lower electricity use when wholesale market prices are high or when the supply system reliability is jeopardized. The energy and power savings associated with these actions can be quantified and measured against the electric load that might reasonably be anticipated in the absence of those changes. These changes can be translated into economic terms based on an electricity tariff associated with a particular site. Specifically, LBNL developed applications to

Calculate a baseline electric load shape that is used to estimate the short-term peak demand reduction from DR events (kW) or long-term savings from energy efficiency measures (kWh). This baseline load shape is the basis of our measurement and verification services. This initial work is oriented toward techniques to evaluate whole building load shapes (Mathieu et al.. 2011 and Price, 2010).

Conduct measurement and verification (M&V) of energy and demand savings. Baseline loads are compared to actual metered energy use to determine the savings during DR events managed by applications such as PNNL's automated DR agent, or from energy efficiency interventions such as changes in RTU operations based on information from PNNL's fault detection agent.

Estimate the economic savings from participating in DR events or long-term savings from energy efficiency interventions based on representative electricity tariffs.

Convey demand response (DR) events using a DR event scheduler. This application provides signals that publish DR events on the VL communication bus using an open source, Open Automated Demand Response (OpenADR) client developed by an industrial partner, EnerNOC®, Inc. OpenADR is an interoperable, standards-based communications specification that provides price and grid reliability signals that allow a

building to transact changes in its electric load with utility and grid activities. This activity builds on previous work funded by the BTO described in Kiliccote et al. (2006) and the OpenADR Alliance (www.openadr.org/specification; See also Ghatikar, 2012; Holmberg, 2012).

The relationship between the baseline, M&V and economic valuation applications is shown in Figure 1.

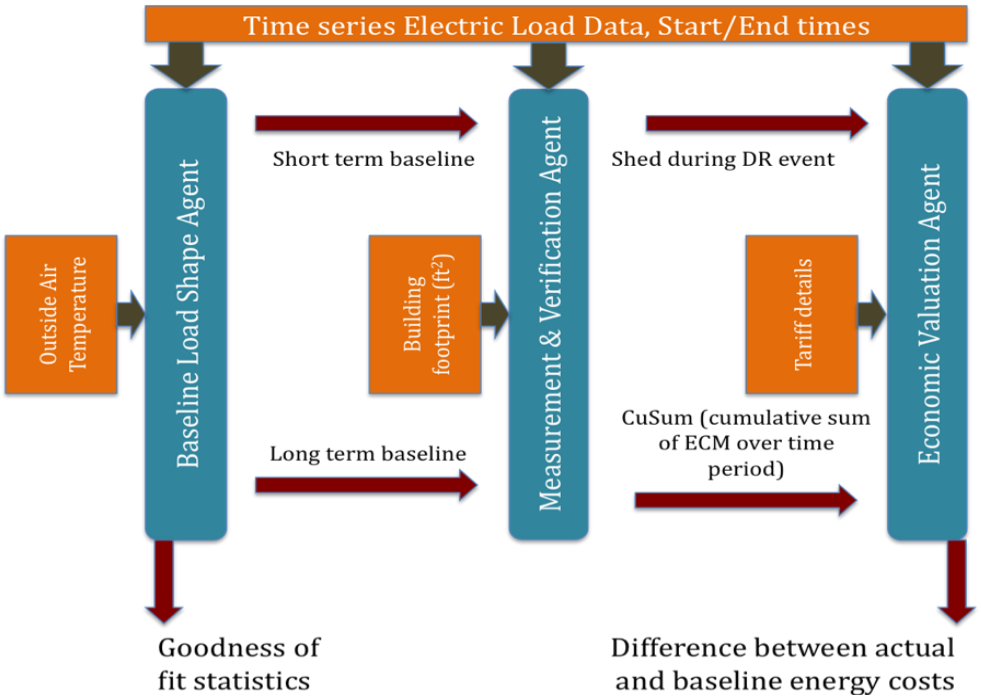

Figure 1. Relation between LBNL developed Baseline, M&V, and Economic Savings agents

In addition, LBNL provided administrative and software development support for the data historian for the VL platform. This historian archives time series data from the building control and metering systems using sMAP - the Simple Measurement and Actuation Profile (Dawson-Haggerty, 2013). The core object in sMAP is the time series, a single progression of (time+value) pairs. Each time series in sMAP can be tagged with metadata; all grouping of time series occurs using these tags.

LBNL also developed a weather data management system to automate the acquisition of reliable weather data for any location in the continental United States from a commercial weather data archive. Knowledge of outside air temperature is critical to the development of baseline load shapes, and therefore the accurate determination of energy, demand, and economic savings. This system develops a single outdoor air temperature time series by averaging data from five Weather Underground™ certified stations nearest the zip code of a given site.

Baseline Load Shape Agent

Baseline Model

In every building, the consumption of electric power (or “load”) is not constant. In most buildings the load varies with outdoor temperature due to the use of heating and cooling systems. The load also varies with time, because of (1) scheduled events such as exterior lighting being automatically turned on and off at certain times, (2) routine but not strictly scheduled events such as employees turning lights and computers on or off, and (3) non-routine variability such as the occasional use of copy machines, or people turning office equipment on or off at unusual times.

Baseline models provide a basis of comparison to determine the impact of operational changes by predicting building electric loads based on historic electric load data and explanatory variables (Addy, 2013). The model is fit to data from a “training period”, and is used to predict the load in the “prediction period.” The key output of a baseline model is the “projected baseline load”, which is a time series of the predicted energy use if the building is operated during the prediction period the same way it was operated during the training period. Ideally, a baseline prediction will account for all of the scheduled and routine uses of electricity in the building as well as electric load that varies with outdoor air temperature. The agents described here build upon baseline models that are described in detail by Mathieu et al. (2011a, 2011b), Mathieu (2012), and Price (2010).

There are two ways to use the baseline agent to predict whole-building electric load: (1) the agent can predict the load based only on previous electric load, or (2) the agent can also use outdoor air temperature data to yield an improved prediction.

Usage patterns will vary over time, as will outdoor air temperatures. The final predicted baseline load at a given time is a weighted average of several model predictions. Each prediction is the output of a model that uses a weighting scheme that is designed so that the predicted baseline load at any given time is influenced most strongly by data from close to that time, with data from the distant past (and distant future) being given less statistical weight. Details are given in the “statistical weights” section below.

Model based on load data only

When outdoor air temperature data is not used in the model, only the historic time series of electric data are used to create the baseline model. In this case the predicted load for a given time of the week will simply be the weighted average load at that time of the week. For example,

the predicted load for a particular Tuesday at 12:15 would be the weighted average load on all other Tuesdays at 12:15 in the training period.

Model based on load and outdoor air temperature

A more sophisticated statistical model is possible if outside air temperature data are provided in addition to the electric load data, The basis of the full model is an underlying linear regression model that assumes that the predicted load is the sum of a "time-of-week" effect plus a temperature effect (described in Mathieu et al., IEEE Transactions on Smart Grid, 2011). The "time-of-week" effect is implemented through the use of indicator variables for each time interval during the week: an indicator variable for 00:15 on Sunday morning, one for 00:30, and so on through the week. (Indicator variables, also called "dummy variables", are used in linear regression models to indicate whether a data point is, or is not, a member of a class; in the present case, for example, there is a variable that has a value of 1 for all data that were collected at 00:15 on a Sunday, and 0 for all other data, and so on for each time of the week).

The resulting regression coefficients account for the regular variation of load during the week that is not correlated with outdoor air temperature. The temperature-dependent part of the load, assumes a piecewise-linear relationship between temperature and load: within each of several temperature ranges, described below, the load is assumed to increase (or decrease) as a linear function of temperature, but each temperature range may have a different slope. Temperature ranges may be chosen through a statistical procedure, but in practice good results are usually attained by simply assigning bin boundaries so that they span most of the temperature range experienced by the building and include at least two bins in the range between 50F and 70F. For example, for a building the San Francisco Bay Area, bin boundaries might be chosen at 50F, 60F, 70F, and 90F. In many buildings, load will decrease with temperature in the lowest temperature range (below 50 F) since less heating would be required with warmer temperatures. At the other end of the range, load in many buildings increases with increasing temperature in the higher temperature ranges because more air conditioning would be needed with higher temperatures.

The model uses an approach described in Price et al. (2013) to separate the times of the week into two groups: a group of times in which the load depends more strongly on temperature, and one in which the load depends less strongly on temperature; a separate model is fit to data from each group. In most buildings these time groups correspond to times when the building is occupied versus unoccupied. Typically, but not always, these times correspond to similarly named HVAC modes.

Statistical weights

The underlying statistical model accounts for weekly periodicity in load, and for changes in load that are correlated with changes in outdoor air temperature. But in most buildings there sources

of load variation besides weekly periodicity and air temperature. For example, changes in nighttime lighting might lead to an increase or decrease in the load at night, so that the pattern of electricity consumption is different after the change was made than it was before. To adjust for this sort of change in behavior, in order to predict the load shape on a given day, we give more statistical weight to days that are nearby in time, whether before or after the given day. This is achieved by fitting the regression model using statistical weights that fall off as a function of time in both directions from a central day. A central time point is selected as discussed below, and the time difference between that point and every other data point is determined (in days, which may be fractional). The statistical weight, w, given to a point d days from the central time point is:

$$w(d) = \frac{D^2}{(D^2 + d^2)}$$

where D is a user-selected parameter defined by the weighting days argument. Figure 2 shows how this weighting function varies for different values of D.

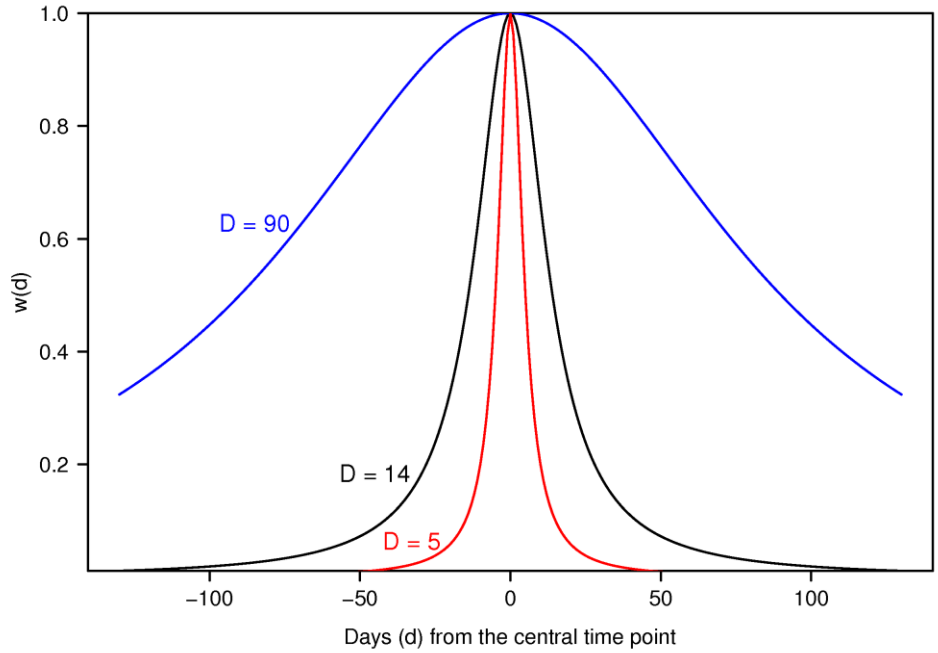

Figure 2. Weighting function for different choices of D, the metric by which "short term" is measured.

The parameter D can be thought of as a "sensitivity" parameter that determines how closely the baseline model tries to match short-term fluctuations in the load data, versus capturing long-term trends. Setting a large value for D (such as 90 days) implies that data from three months ago are almost as informative about tomorrow's energy consumption as data from one week ago; setting a small value (such as 5 days) implies that data from two or three weeks ago are almost useless in predicting tomorrow's energy consumption. Empirically D=14 days is a good choice when predicting the short term load variation for several buildings we have studied, so

we set it as the default value while allowing the user to change it if needed. Buildings that vary greatly from week to week would be better modeled with a smaller value of D, while buildings that are extremely consistent would be better modeled with a larger value of D.

To train the predictive model over a given time period, the weighted regression procedure is repeated for several different "central time points." Specifically, a set of "central time points" about D days apart is selected, spanning the time range of the data and the regression model is fit multiple times (e.g., using D=14, there would be 28 regression models generated in one complete year of training data), using each of these in turn as the "central time point." Each of these models is used to make a prediction for each of the requested output times, resulting in a set of predictions for each output time: one prediction for each regression model. For a given output time, some of these predictions are from models in which the central time point was far from the output time, and some are from when the central time point was close to the output time. The predictions are weighted, using the same w(d) function above, to give more statistical weight to the predictions from "nearby" central time points.

The process for combining the individual regression predictions to generate the final prediction is illustrated in Figure 3. The upper panel of the figure shows the final baseline prediction in blue. At any given time, the final prediction is the weighted sum of several different predictions, three of which are shown in the lower panels of the plot. For example, consider a point 11 days after the end of the training period. Since the first regression model has a central time point 0 days before the end of the training period, it is the most strongly weighted model at the point being predicted (see the red line on the second panel of the figure). The second regression model has a central time point 14 days earlier, so it has a lower weight (third panel). The third regression model has a central time point even farther in the past, and thus an even lower weight (final panel).

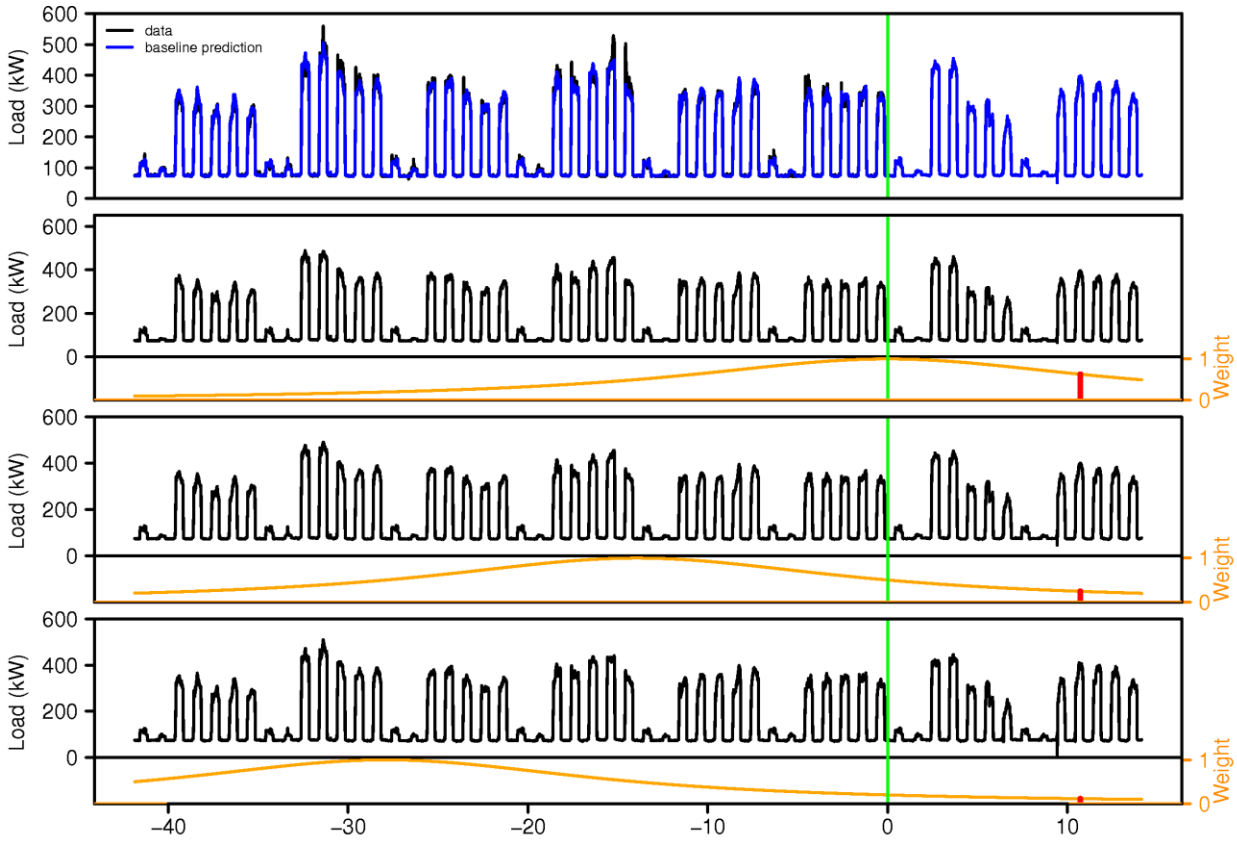

Figure 3. Illustration of different weighting functions for statistical model.

Top panel:
Data (black) go from the left side of the plot up to the green line. The baseline prediction (blue) goes all the way across the plot;. To the right of the green line the baseline prediction is a forecast, i.e. we have no data from the green line forward.

Second, third, and fourth panels (black lines) show (1) linear regression predictions with central time points 0, 14, and 28 days before the end of the data, respectively, and (2) the weight function used for each prediction.

The weighting function w(d) has the effect that a prediction for a time less than D days after the end of the training data will be based mostly on the data from near the end of the training period, but a prediction for a time more than D days after the end of the training period will be based on a more equal weighting of the training period. As an example, consider using the parameter value D = 14 days with data from all of 2013 to predict the baseline from January 1, 2014 to July 1, 2014. The prediction on January 1, 2014 is the weighted sum of regression predictions that are fit to the 2013 training data using different central time points, as previously discussed. Since one of these central time points (December 31) is just one day away from the start of the time for which a baseline will be generated (January 1), that regression has a weight of over 0.99 at the start of the baseline. A previous regression, with a central time point about 14 days earlier, has a weight under 0.5 on January 1. A regression with a central time point an

additional 14 days earlier has a weight under 0.2, and the regression centered in mid-November has a weight under 0.1, and so on back through time. In this case, training data prior to November have weights so low that they are essentially negligible. Therefore, the prediction for January 1, 2014 is based almost entirely on data from December 2013, with data from November playing a minor role and data from the rest of 2013 having a nearly negligible effect.

Now consider the prediction for June 30, 2014. This is d=180 days after the end of the training data. In making a prediction for June 30, the regression that has a central time point on December 31 is given a weight of 0.006. The regression with a central time point 14 days earlier has a weight of 0.005. The regression with a central time point another 14 days earlier has a weight of 0.004. Even the regression with a central time point a full year ago, at the end of June, 2013, has a weight of over 0.001, which is still 17% as much weight as the regression with the most recent central time point. Even the regression with the most distant time point, all the way back on January 1, 2013, is assigned 10% as much statistical weight as the regression with the most recent central time point. Thus, in contrast to the baseline prediction for early January 2014, which is based almost entirely on the previous month or two of training data, the baseline prediction for June 2014 ends up being an average of regression predictions that take into account the full year of training data, although still weighting the last half of the year more heavily than the first half.

We believe, based on limited tests of the model for several sites, that in most cases the optimal value of D will probably be of the order of 10 to 20 days both for quantifying demand response effectiveness and for making long-term predictions suitable for M&V applications (see Figure 4). Using smaller values of D cause the baseline prediction to be influenced strongly by anomalies or changes in building load shape that only last a few days or a week, whereas much larger values for D prevent the predictions from adapting to long-term changes in load patterns.

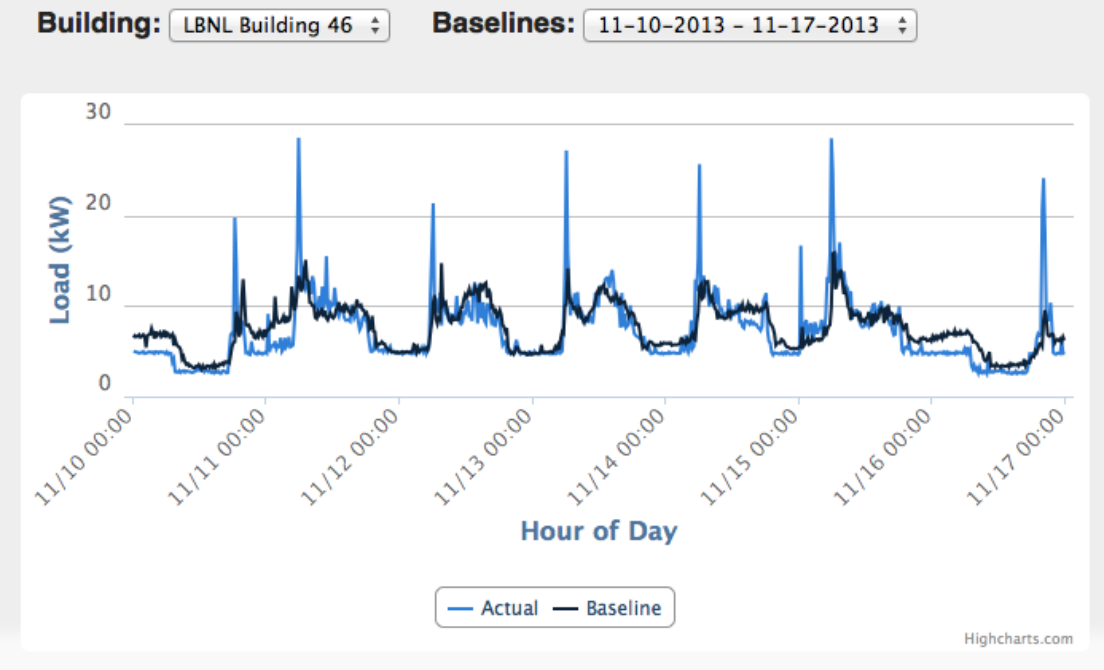

Figure 4. Example of predicted baseline load (black) and actual load (blue) for a week in November 2014 in the LBNL test building

Agent Implementation

The Baseline Load Shape agent implements the baseline model for use by other VL agents and applications. This agent's inputs include load data and timestamps for the training period; optionally, outdoor air temperature data and timestamps may be provided. The other required input is the set of timestamps that define the prediction period (i.e. start and end timestamps).

Figure 5 shows the inputs and outputs from this agent. The required input is a historical electric loads shape with associated timestamps. The time increments can be any increment, but are typically 15-minute intervals to correspond with traditional time intervals used by building electric meters.

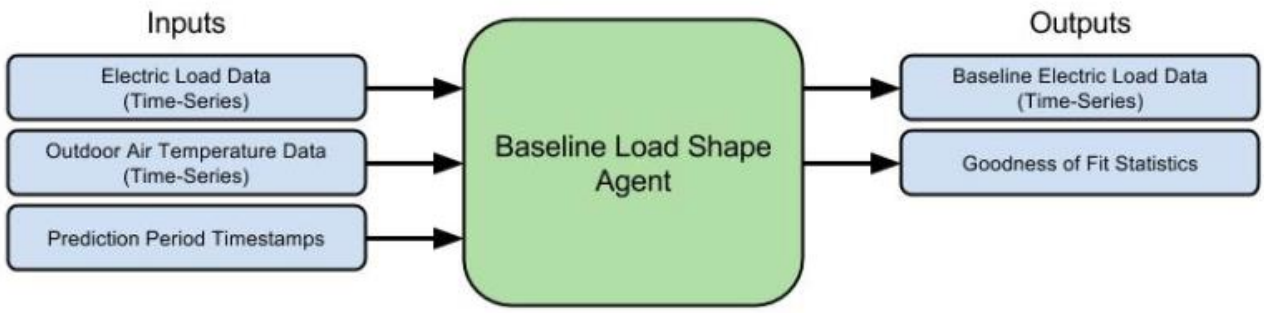

Figure 5. Inputs and outputs for the Transactional Network Baseline Load Shape Agent

Since the accuracy of baseline model predictions has important implications for the quantification of economic value from building energy transactions, a set of goodness of fit statistics is provided by the baseline agent to measure the degree to which a particular baseline model is able predict the building's load. They include (a) the standard error of the residuals during the "training" period (which is the dataset on which the model is based); and (b) a correlation coefficient that quantifies how much of the variance in load is predicted by the baseline behavior (where 1 indicates perfect fit).

To provide weather data, LBNL developed a tool (i.e., a common function used my more than one agent) to compile and aggregate weather data from Weather Underground sources, indexed by zip codes. Information is acquired from up to five weather stations in the zip code. Since these data come from sources of varying degrees of accuracy, the median temperature from the available temperatures for a given time is assigned to that time slot in the data stream.

Measurement and Verification Agent

The Measurement and Verification (M&V) agent quantifies short-term transactional load reductions, for example from use of the DR agent (Katipamula et al., 2013), as well as longer-term energy savings from efficiency measures or improved controls. The M&V agent uses the predicted baseline load provided by the Baseline Load Shape agent, combined with information about the timing and duration of transactional events or efficiency measures. Figure 6 shows the agent's inputs and outputs.

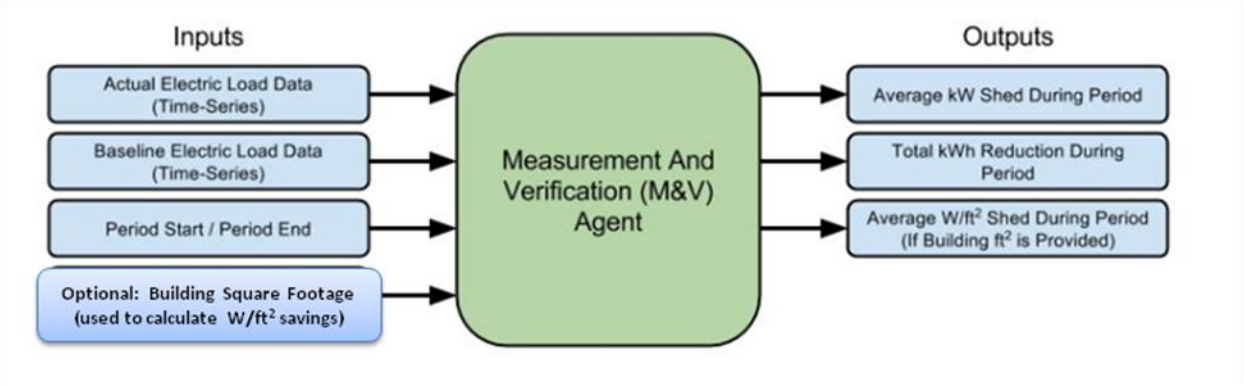

Figure 6. Inputs and outputs for the Transactional Network Measurement and Verification agent

The M&V agent is configured to quantify avoided energy use, or "energy savings" as illustrated in Figure 7. A baseline model, created by the Baseline Load Shape agent is developed to characterize the building's typical load in the absence of any transactional events or efficiency measures. Once an event or efficiency measure is implemented, the baseline model is used to project the load that would have occurred without the event or measure. The difference between the baseline-projected use and the actual metered use comprises the reported savings.

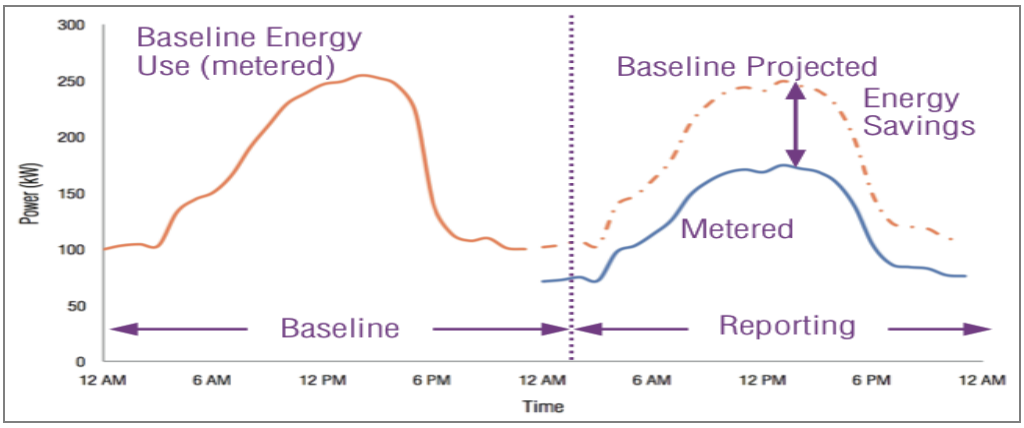

Figure 7. Method of savings quantification applied in the Transactional Network Measurement and Verification Agent

Using the M&V Agent to Quantify Demand Response Load Reductions

DR event performance is calculated using a baseline model that gives heavier weight to days immediately preceding the event than it does to days that passed months before the event. This gives more accurate predictions from the baseline model. The projected baseline load is generated for each metered time-interval in the DR event day, using weather data from the DR event day. For DR events, the M&V agent calculates the difference between the actual energy use and the predicted baseline energy use at each time interval during the DR event day, and cumulatively through the day. Dividing the average demand reduction by the building floor area yields the load reduction per square foot; dividing the cumulative energy saved by the cumulative predicted baseline energy use yields the percent reduction in load.

Using the M&V Agent to Quantify Long-Term Energy Savings

In contrast to DR, or other short-term transactional events, the impact of energy efficiency improvements can accumulate for days, months or years. As in the DR case, the projected baseline load is generated for each metered time interval following implementation of the efficiency measure, using weather data from the building location, and the projected load is compared to the actual load both cumulatively and at each measurement interval. Dividing the savings by building floor area yields savings per square foot, and dividing by predicted baseline load yields the savings as a percent of baseline load.

Cumulative sums (CuSum) represent the aggregated, or cumulative, difference between the baseline projected load and the actual metered load. Described in Granderson et al. 2011, with application examples, CuSum is useful for tracking operational persistence of savings as well as total accumulated energy savings since an improvement was made. Represented in Figure 8, the y-value of the CuSum plot shows the total accumulated energy savings. Operationally, a flat slope marks a period of no savings, or no usage in excess of baseline; a positive slope indicates a period of decrease use, or energy savings; a negative slope marks a period of usage in excess of baseline.

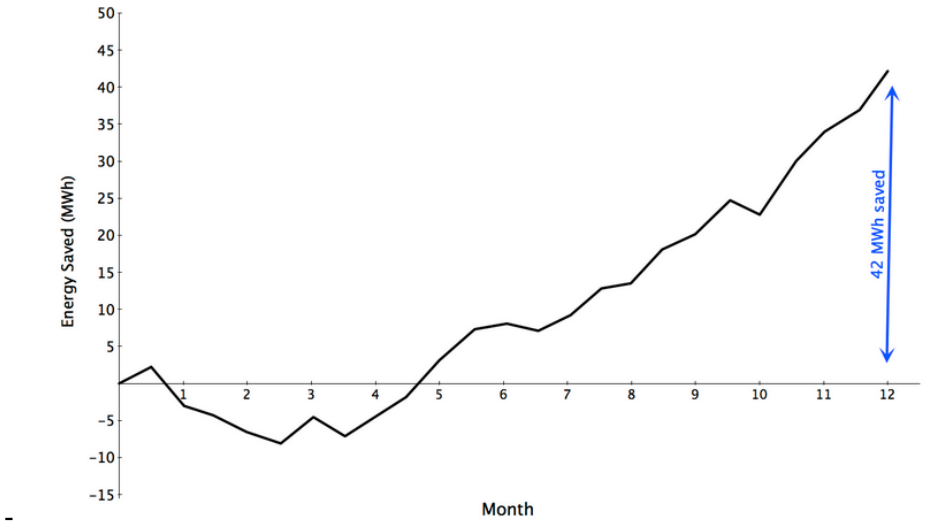

Figure 8. EE Measurement and Verification (Cumulative Summation)

Economic Valuation Agent

The cost of providing electricity varies over the course of time, with this variation represented by time-varying tariffs to which an increasing number of utility customers subscribe. Tariff information is conveyed within the TN via OpenEI format[1] that allows for time of day and time of week energy cost characterizations as well as demand charges. The transactional network optimizes building operations in the context of time-varying tariffs. To do this, the economic valuation agent converts the energy savings or load reductions described above, into financial terms. By comparing actual load to the predicted baseline load, this application quantifies the monetary impact of changes in energy use and hourly demand considering the price changes inherent in a time-of-use tariff. The agent can also handle the DR event-based changes in price associated with critical peak pricing. The result is a data stream of savings values over the course of a day (or other selected time period), typically used for DR events, or an accumulated sum over a period of time, typically used to measure the results of energy efficiency measures.

[1] http://en.openei.org/wiki/Main_Page

Price-based DR events are triggered by price signals that can indicate a series of day-ahead hourly prices or an abrupt change in the price of electricity for a given period of time. High price events provide incentives for peak power reductions to have greater economic value then medium or low price periods. To convert power savings to an economic value, a conversion using a specific tariff is required. The economic valuation agent currently supports time-of-use and common critical-peak pricing tariffs (see Figure 9).

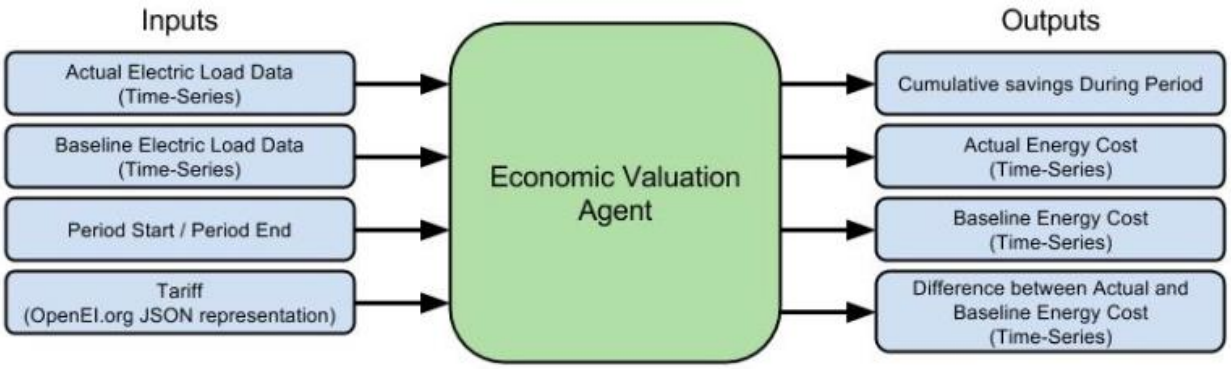

Figure 9. Economic Valuation agent

Figure 10 illustrates typical time of use and critical peak pricing tariff designs in general terms, but does not represent any utility's actual tariff. Time of use (TOU) price periods are fixed, predictable time periods for off-peak, part peak, and peak prices. Critical peak pricing (CPP) tariffs often include a dynamic price in addition to the on-peak costs. CPP price events may be associated with higher regional demand for electric energy. Some California CPP events are triggered with higher outdoor air temperatures. For example, the 10 to 15 hottest days of the summer would be designated as CPP days, when they occur, by the utility.

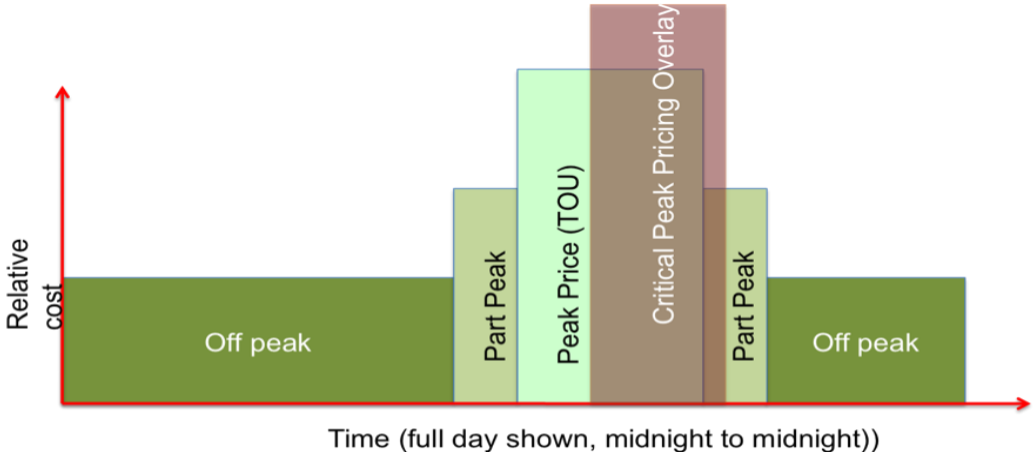

Figure 10. Illustration of TOU and CPP tariff

The economic value of the response to the DR event is measured by the difference between the anticipated cost (calculated using the baseline) and the actual cost (calculated from the actual load), using tariff price information conveyed via the DR scheduler agent.
Figure 11 illustrates how a building's energy costs might vary for each hour of the day as a result of this kind of tariff. The amount of energy use (kWh) at a given time and the TOU costs associated with that time combine to get this cost shape. The Economic Valuation Agent will host a number of other tariffs in future releases of the software.

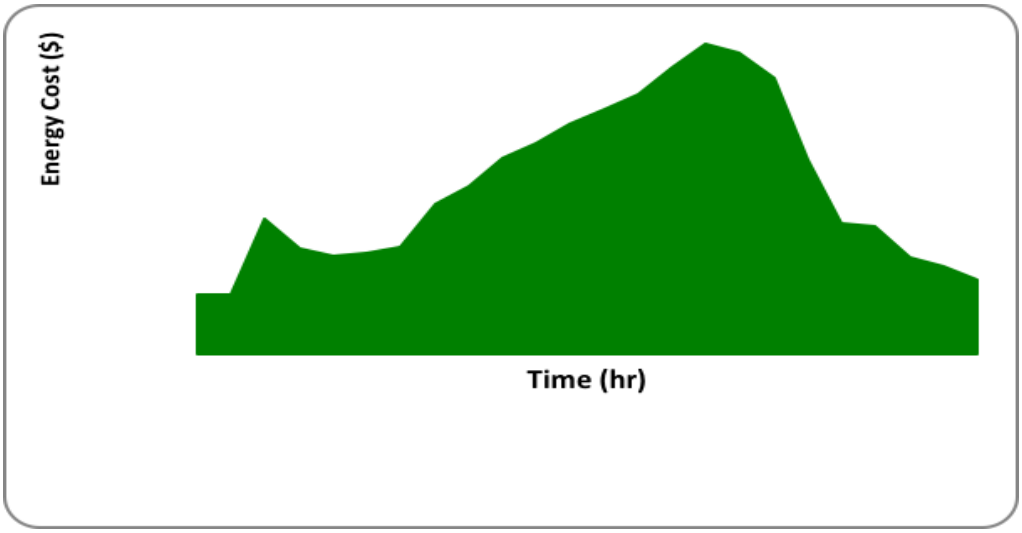

Figure 11. Resulting Economic Value of Energy Savings

Demand Response Scheduler Agent

Demand response (DR) programs, dynamic pricing, and future transactive markets provide incentives for building operators to modify their electric loads during identified times such as when there is a reduced supply of energy or high prices (Piette et al, 2012). Demand response signals typically originate at a retail electric utility or a wholesale independent system operator, indicating a need for consumers to modify their electricity usage on certain days in a given time period, or shift that usage to another time period. In this project we use OpenADR to provide pricing signals from the electric grid to the consumer. OpenADR is a client/server communication standard for conveying DR signals from the utility to a building's control system, where preprogrammed response strategies can be initiated (Ghatikar et al, 2011). Figure 12 illustrates how utilities, independent systems operators, and curtailment service providers or aggregators currently use OpenADR to convey price and grid reliability signals to end users to enable a response in a timely and standardized way.

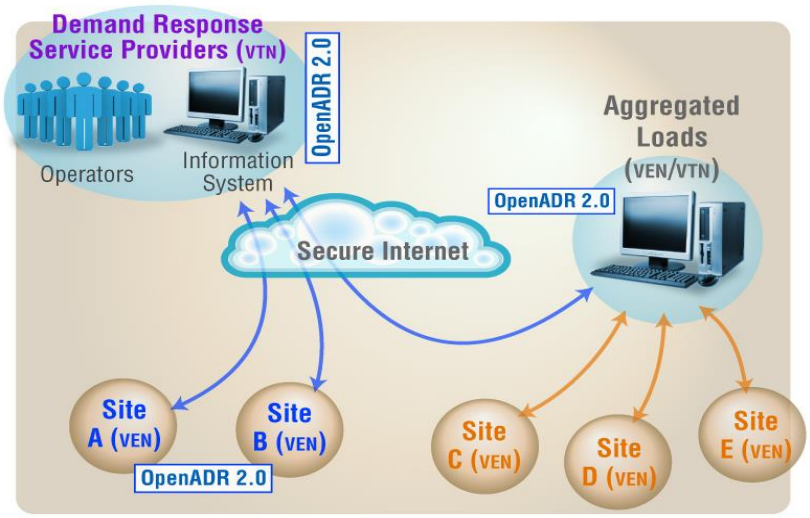

Figure 12. OpenADR 2.0 messages conveyed from server to client

The DR Event Scheduler agent receives DR event signals from an OpenADR client (also called as virtual end node or VEN). The client translates these signals into a format more easily processed by other agents in the VL platform, and then communicates these signals to the VL communication bus. The utility or grid operators use a DR server (also called as virtual top node or VTN) to generate and publish signals specific to a particular customer's participation in a DR program. These signals can support secure transactions in accordance with national smart grid standards requirements (NISTR, 2010). For this demonstration, a DR message from a server is transmitted to PNNL's DR agent, which automatically triggers preprogrammed actuation of the RTUs in response to the DR signal. Figure 13 illustrates how the DR scheduler agent works:

Figure 13. DR Scheduler Agent

Specifically, the DR Scheduler extracts the event status (none, far, near, active, completed or cancelled), event start, event end, and event ID from the signal received by the OpenADR client, and publishes this information to the VL communication bus. Figure 14, below, illustrates how these various components of DR information relate to a DR signal. Prior to the start of an event, the notification time includes both the far and near states. The entire notification time is the time

during which an event is pending. The demarcation between far and near times is the time at which the resource is expected to begin ramping to the desired change in load. The duration of the event (active state) can be subdivided, if needed, with different signals to indicate changes in pricing or severity of need (e.g. change from moderate to high as an indicator of increased need to reduce load). The completed time identifies when the DR event ends and although a recovery period is observed in some cases (e.g. HVAC load increases because of temperature rise in the building during the DR event), from a DR event perspective, that is in the completed part of the event. A typical DR event will have a signal associated with each of the identified time components. For example, typical DR signals include event status information such as far, near, active, and completed, and start and end time (or start and duration) of a DR event period. The other aspects of time components could be used by the building controls to support the key characteristics of DR program requirements (e.g. particular ramp time is required for fast DR programs).

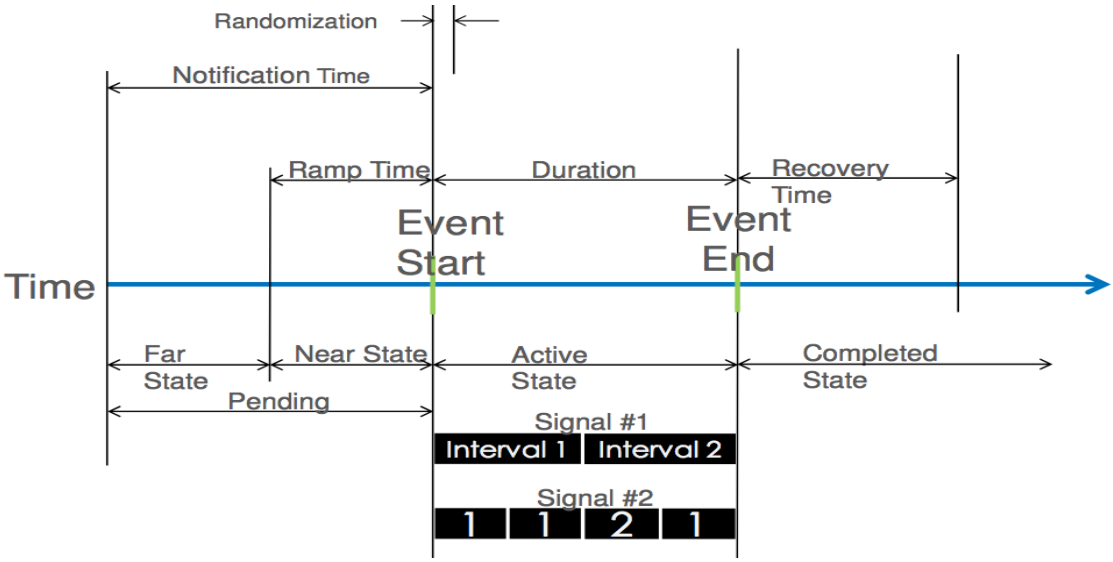

Figure 14. Key components of DR event conveyed by OpenADR

By publishing this information to the VL bus, the DR Scheduler agent enables other control agents to act on this information to modify electric loads. The first phase of development supports an automated response to critical peak prices. Figure 15, below, illustrates the flow of information in the DR signal to be used by VL to trigger the pre-programmed strategies at the individual building equipment level.

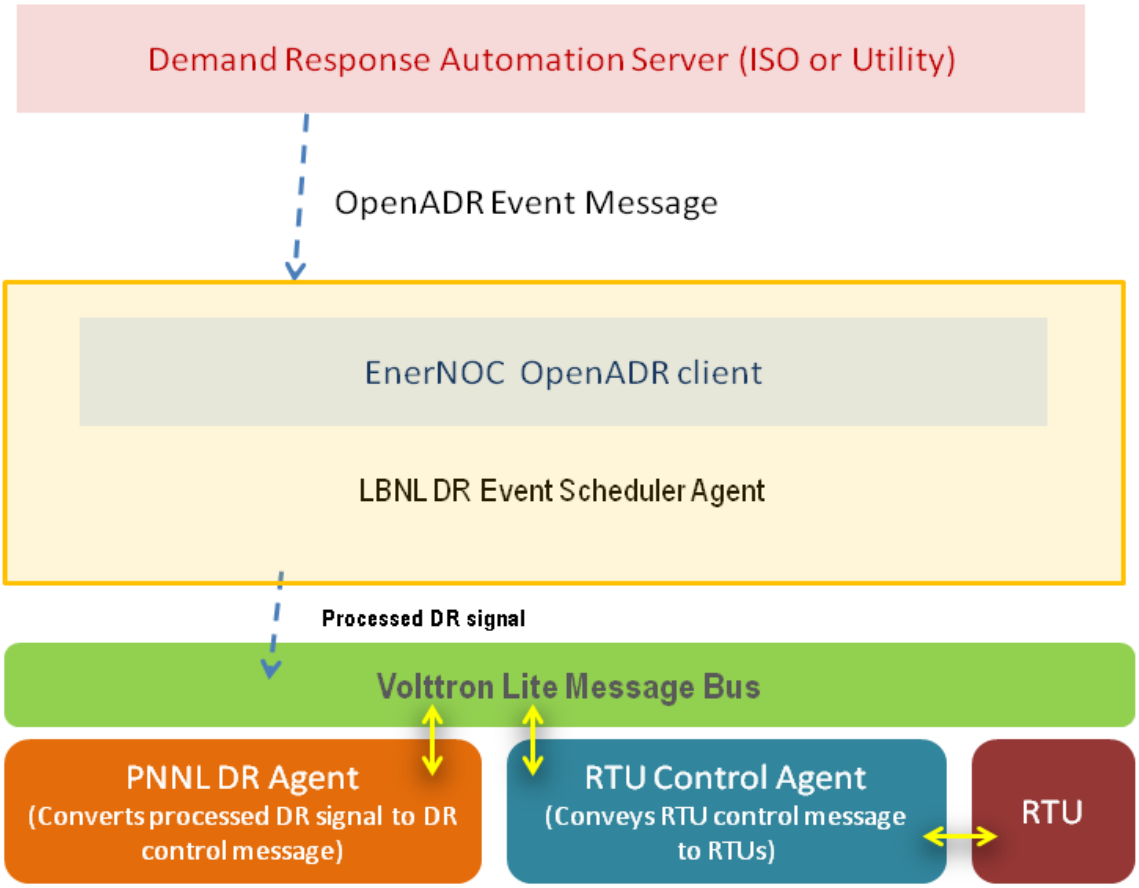

Figure 15. DR signal within the Transactional Network

Prototype Application

The agents described in the previous sections are intentionally designed to provide generalized functionality, for baselining, M&V, and other applications. To confirm the functionality of these agents, they were implemented in a prototype application at the LBNL campus. The text below describes the software and hardware aspects of that testing.

Software Implementation

During testing, building supervisory control was performed either directly via a web interface to the Catalyst controller installed on the RTUs at the LBNL campus or, in the last test, using the VL platform, LBNL's DR scheduler agent, and PNNL's DR and RTU control agents (described in Katipamula et al., 2013). Analysis of the resulting building data was performed by LBNL's baseline, M&V, and economic valuation agents using the steps listed below:

Ongoing data collection and compilation into the data historian is run daily using these steps:

1. Download the latest outdoor air temperature data for the building from a commercial source (Weather Underground™), pre-process, and upload to the sMAP database.

2. From the sMAP database, retrieve the following data up to the present time: the building's load, the outdoor air temperature, and data on which RTUs were participating in a DR event at which times.

3. From a configuration file, retrieve a list of holidays.

Steps 4-8 demonstrate the M&V application by comparing the predicted baseline load to the actual load. We have chosen November 1, 2013 as an example of a date for implementation of an energy conservation measure and we are evaluating its effectiveness. These steps are automatically performed every day.

4. Create input files for the baseline agent's training period. The training period is all days prior to November 1, 2013 that are not holidays or DR days.

5. Use the baseline agent to fit the baseline model, and generate the predicted baseline load for all days from November 1, 2013 to present.

6. Use the M&V agent to calculate the difference between the predicted baseline load and the actual load since November 1.

7. Use the economic valuation agent to calculate the electricity cost since November 1, and compare this to the cost that would have been incurred under the predicted baseline load. The economic valuation agent calculates this cost difference for each time interval, and cumulatively since November 1.

8. Store all of the results of steps 5-7 in the sMAP database, where they can be accessed by a website that can display the results.

The remaining steps estimate the effectiveness of Demand Response measures in the test building. They are carried out only if the previous day was a "DR day," where a day is defined as a "DR day" if any of the building's RTUs were manipulated for DR purposes at any point during the day.

9. Create input data files for the baseline agent's training period. The training period is all days that are not holidays or DR days,

10. Use the baseline agent to fit the baseline model, and generate baseline load predictions for all DR days.

11. Use the M&V agent to calculate the difference between the predicted baseline load and the actual load for all of the DR days. The agent calculates the difference between baseline load and actual load at each measured time interval, and also the cumulative sum of the difference throughout the day.

12. Use the economic valuation agent to calculate the electricity cost during each DR day, and compare this cost to the cost that would have been incurred for the predicted baseline load. The agent calculates this cost difference for each time interval during each DR event, as well as the cumulative sum of the difference for each event.

13. Store all of the results of steps 1010-12 in the sMAP database, where they can be accessed by a website that can display the results.

Building Demonstration

To test these software tools, LBNL installed a series of control and communications platforms similar to the configuration described in Katipamula et al (2013) at a small (5000 ft^2) office building on the LBNL campus. The building, known as 46A, is served by seven RTUs, as seen in Figure 16, each of which was equipped with a Catalyst controller. Three of the RTUs are 2-ton units, two are 3-ton units, and two are 2.5-ton units. The three-ton units were also fitted with variable speed drives for the supply air-handler fans. The 3-ton units serve the reception area and entrance to the middle portion of the building. The power demand from each unit is measured individually as well as the whole-building load. For the purposes of the testing described below, only the whole building power was used for analysis.

Figure 16. Roof of the LBNL Testbed for Transactional Network project

Testing

A series of manual and automated DR tests evaluated the complete group of electric load-shape analysis agents described in this paper. These tests included two events initiated by LBNL to reset zone temperatures and evaluate the electric load shape response, as well as one end-to-end test of PNNL's DR agent. Table 1 shows the RTU control strategies used to create a change in the electric load shape of the building.

Table 1. DR Tests at LBNL office building

Date	Time	Test Strategy
9/23/2013	2pm – 4pm	2°F increase in set point on all thermostats
9/27/2013	2 pm – 4 pm	4°F increase in set point on all thermostats
10/18/2013	1 – 2 pm: Precool 2 – 5 pm: Event 5 – 6 pm return to normal	One hour precooling, gradual increase in set point to +4°F from normal, then slow return to normal to minimize rebound effect

The first of these tests was triggered manually via a web interface to the Catalyst units controlling the thermostats in the building. It examined the interactions between components of the system and verified that the metered electric load data could be reliably obtained and used for calculations. The second test refined this process further and explored the ability of the building to respond to a more severe load reduction (modeling the response to a higher price signal indicating a greater need for load reduction). The third test verified that DR events could be triggered automatically using the OpenADR server and the DR and control agents running on the VL platform. It also tested the ability of the building RTUs to respond to more complex DR signals. To illustrate the calculations performed during testing, Figure 17 shows the results from the second of these tests.

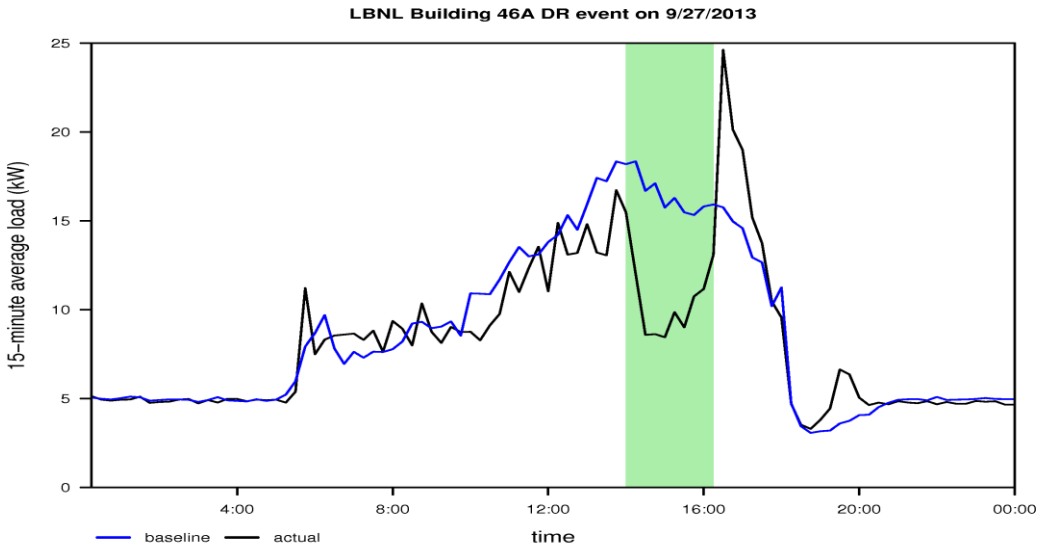

Figure 17. Data from second DR test at LBNL test bed (DR event shaded)

Note that the building uses about 5 kW during the night. Electricity use increases during the day and the peak demand shown in the baseline model reaches about 17 kW around 2 pm. The electric load was reduced to about 9 kW at the start of the event, increasing to nearly 15 kW by the end of the event. The DR strategy reduced the electric use by an average of about 6 kW for the two-hour event. This reduction is equivalent to over one-third of the electricity use and over 1 W//ft^2.

It is notable that there was a rebound to nearly 25 kW, which could have been mitigated through a variety of rebound avoidance strategies. The building could have gone into an early "unoccupied mode" to coast through the recovery event without a new peak. Or, the control could have moved the zone temperature back to normal more slowly. Characteristics of this response from the load-shape analysis agents are shown in the two tables below.

Table 2. Power and Energy changes during 9/27/2013 DR test at LBNL building

Estimated Shed (During DR Period)	Total Savings (Whole Day)
Average shed: 6.13 kW (1.17 W/ft^2)	Total energy reduction: 9.6 kWh
Average power reduction: 38%	Reduction in power consumption: 5.1%
Total energy reduction: 13.8 kWh	Total reduction in energy cost: 20.1%

It is important to understand how the zone reset strategy influences the zone temperatures. Figure 18 shows the change in zone temperatures in seven of the zones (labeled 8 – 14 to be consistent with site numbering; each zone was served by a single RTU) during the two-hour event. On average the building warms up from about 75 – 77 °F, with each zone experiencing a 2 – 3 °F increase. This is a common result, that space temperature warm up less than the reset of 4 °F. There was a greater variation among individual zones. None of the zones rose above 78 °F. The zones have a variety of external orientations, some receiving more solar gain than others.

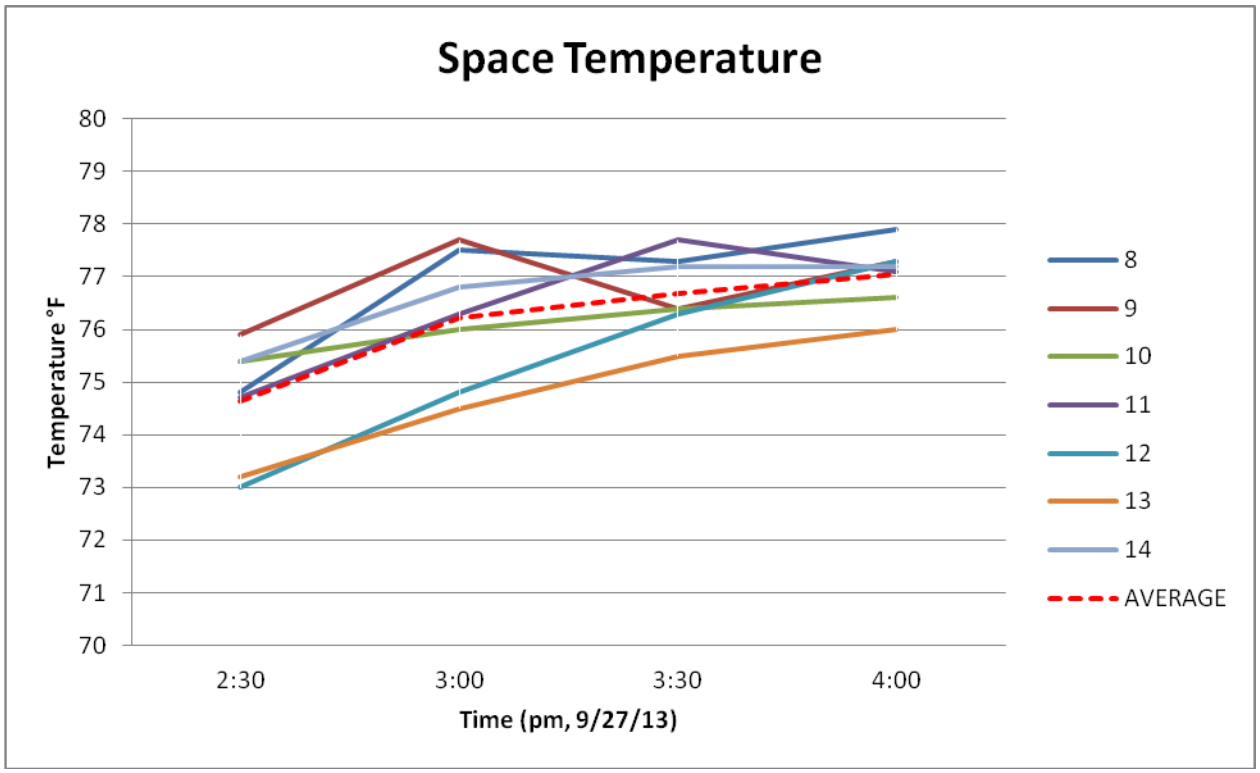

Figure 18. Change in zone space temperatures during the Friday, September 27 DR test.

The baseline agent reports several goodness-of-fit statistics that summarize how well the baseline model fits the actual load during the training period, and, by extension, provides a measure of the expected accuracy of the model predictions during the prediction period. Table 3 shows two of the goodness-of-fit statistics that report the expected error in the predicted load for 15-minute and 1-hour periods during the training period. The bottom line of the table shows that when predicting the hourly average load, the expected error is 1.07 kW in either direction (that is, either high or low), corresponding to an error of 9%.

Table 3. Goodness of fit statistics, DR test event at LBNL, 9/27/2013

Model goodness of fit	Mean absolute percent error	Root mean squared error
15 minute intervals	12.4%	1.52 kW
Hourly average	9.0%	1.07 kW

The goodness-of-fit statistics report the error both in absolute terms (kW) and relative terms (percent error). Generally, as in the example here, the longer the time period being predicted, the more accurate the prediction: in some short time periods the load will be over-predicted while in others it is under-predicted, and the accumulated errors will tend to cancel out with time.

Discussion

A team comprised of staff from three national laboratories performed development of the Transactional Network. The team has a long term goal to develop an agent based platform that can respond to signals associated with managing the consumption or flow of electric power within an electric power system through the use of economic or market based constructs while considering grid reliability constraints. The term “transactive” is used because the decisions are made based on a value. These decisions may be analogous to or literally economic transactions.

LBNL’s primary contribution to this platform was the development of M&V systems to allow automated feedback for energy efficiency and DR events. This agent platform benefits from having standard methods to measure, report, and evaluate energy use patterns. LBNL also provided an initial transactional platform where grid signals are represented as price signals conveyed using OpenADR 2.0 standard. OpenADR is U.S. smart grid interoperability standard, and also used in over eight countries with over 100 members supporting it. A final element LBNL provided uses simple electricity tariffs to translate the peak demand and energy savings data into economic, or dollar savings. While the VL platform was in development, a related set of cloud-based applications representing the VL agents were developed and tested to verify the underlying concepts associated with baseline model codification. The baseline model provides a reference against which the impact of operational changes can be measured. Conversion of these measurements to financial terms used a sample tariff based on time of use and critical peak pricing tariffs currently used by some utilities.

The initial tests of the system, using an occupied office building located at LBNL, demonstrated that the VL based network conveys signals reliably and that the resulting changes in building equipment operations can be reliably characterized in terms that provide a foundation for future transactions.

Summary and Next Steps

The work described here consisted of developing and testing a collection of M&V and automation software agents as part of an autonomous agent platform. The initial efforts tested these agents in their ability to measure and evaluate changes in whole building electric loads that resulted from changes in HVAC control strategies. The agents predicted the electric load shape for each hour of the day and used the baseline model to report the change in electric use between the baseline and the actual consumption. Each RTU received and responded to a variety of set point changes such as precooling and zone reset strategies. RTUs provide a good starting point for this platform because heating, ventilation, and air conditioning constitute a large fraction of electric demand in buildings in the US.

In collaboration with PNNL, LBNL will release software described in this effort in an open source form to allow others to build on and apply these tools. The open source licenses are intended to help spur innovation and industry adoption by fostering an open platform for third parties to collaborate with this DOE sponsored effort.

LBNL will be expanding this work to develop and test agents to control electric lighting systems. One of the emerging concepts in the transactive agent platform development is to explore how data from, and interoperable access to, end-use controllers can be leveraged to allow building energy use to be better managed. One example is to explore new ways to measure and continuously diagnose the operation of occupancy and scheduling-based controls. Two key goals are present in this concept. First, energy use can be reduced overall if the agent systems are able to evaluate and identify energy waste. Energy waste may be present if the HVAC or lighting systems are operating outside of design parameters or if the systems are running when there are no occupants present. This is a common problem in buildings. Second, by building and demonstrating control systems that are able to maintain fault-free efficient operations, and also report savings achieved over time, industry can take the needed steps to scale adoption of efficient controls.

A final goal is to understand how a building might be able to transact with a dynamic electric grid. New work is needed to understand how to represent the availability of a load to the grid. How reliable is the load reduction? How often can the reduction be called? How large of a reduction? These are questions that will be explored in future phases of the project.

References

Addy, Nathan, Johanna L. Mathieu, Sila Kiliccote, and Duncan S. Callaway. Understanding the Effect of Baseline Modeling Implementation Choices on Analysis of Demand Response Performance. IASME International Mechanical Engineering Congress & Exposition. Houston, TX, 2013.

Dawson-Haggerty, Steven. sMAP, Simple Measurement and Actuation Profile. sMAP2.0 documentation. UC Berkeley. 2013. http://www.cs.berkeley.edu/~stevedh/smap2/.

Ghatikar, Girish, and Edward Koch. Deploying Systems Interoperability and Customer Choice within Smart Grid. In Grid-Interop. Irving, TX, 2012.

Ghatikar, Girish, and Rolf Bienert. Smart Grid Standards and Systems Interoperability: A Precedent with OpenADR. In Grid-Interop. Phoenix, AZ, 2011.

Granderson, Jessica, Mary Ann Piette, and Girish Ghatikar. Building Energy Information Systems: User Case Studies. Energy Efficiency 4(1): 17-30, 2011

Granderson, Jessica, Mary Ann Piette, Girish Ghatikar, and Phillip Price. Building Energy Information Systems: State of the Technology and User Case Studies. LBNL-2899E, November 2009, http://eetd.lbl.gov/sites/all/files/LBNL-2899E.pdf

Granderson, Jessica, Mary Ann Piette, B Rosenblum, L Hu, et al. Energy information Handbook: Applications for Energy-Efficient Building Operations. CreateSpace, ISBN 1480178276 / 9781480178274; LBNL 5272E, 2013.
http://eetd.lbl.gov/sites/all/files/LBNL-5272E.pdf

Granderson, Jessica, and Phillip Price. Evaluation of the Predictive Accuracy of Five Baseline Models. LBNL-5886E, August 2012. http://eetd.lbl.gov/sites/all/files/LBNL-5886E.pdf

Haack, JN, S. Katipamula, BA Akyol, and RG Lutes, VOLTTRON Lite: Integration Platform for the Transactional Network, PNNL 22935, October 2013.
http://www.pnl.gov/main/publications/external/technical_reports/PNNL-22935.pdf

Holmberg, David G., Girish Ghatikar, Edward Koch, and Jim Boch. OpenADR Advances. ASHRAE Journal 54, no. 11, 2012.

Katipamula, S., RG Lutes, H Ngo, and RM Underhill. Transactional Network Platform: Applications, PNNL-22941, October 2013.
http://www.pnl.gov/main/publications/external/technical_reports/PNNL-22941.pdf

Kiliccote, S, M.A. Piette, and D Hansen. Advanced Controls and Communications for Demand Response and Energy Efficiency in Commercial Buildings. Proceedings of the Second Carnegie Mellon Conference in Electric Power Systems: Monitoring, Sensing,

Software and Its Valuation for the Changing Electric Power Industry, Pittsburgh PA. LBNL-5937, January 2006. http://eetd.lbl.gov/sites/all/files/LBNL-5937E.pdf

Mathieu, Johanna L., Phillip N Price, Sila Kiliccote, and Mary Ann Piette. Quantifying Changes in Building Electricity Use, with Application to Demand Response, IEEE Transactions on Smart Grid 2:: 507-518, September 2011. http://eetd.lbl.gov/sites/all/files/LBNL-4944E.pdf

Mathieu, Johanna L., Duncan S. Callaway, and Sila Kiliccote. Examining Uncertainty in Demand Response Baseline Models and Variability in Automated Response to Dynamic Pricing. In 2011 IEEE Conference on Decision and Control and European Control Conference. Orlando, FL, 2011a.

Mathieu, Johanna L., Duncan S. Callaway, and Sila Kiliccote. Variability in Automated Responses of Commercial Buildings and Industrial Facilities to Dynamic Electricity Prices. Energy and Buildings, 2011b.

Mathieu, Johanna L. Modeling, Analysis, and Control of Demand Response Resources In Engineering, Mechanical Engineering. Dissertation for Doctor of Philosophy in Mechanical Engineering. Berkeley: University of California, Berkeley, 2012.

National Institute of Standards and Technology Report (NISTR), Guidelines for Smart Grid Cyber Security, NISTR 7228, 2010. *www.nist.gov/smartgrid/upload/**nistir-7628**.*

OpenADR 2.0 standard: OASIS. 2012. Energy Interoperation Version 1.0. Committee Specification 02 [Committee Specification 01 with errata]. http://docs.oasis-open.org/energyinterop/ei/v1.0/energyinterop-v1.0.html.

OpenADR 2.0 Profile Specification: A Profile, Revision 1.0, Document Number 20110712-1, OpenADR Alliance, Morgan Hill, CA, 2011. (http://www.openadr.org/specification)

Piette, Mary Ann, Jessica Granderson, Michael Wetter, and Sila Kiliccote. Intelligent Building Energy Information and Control Systems for Low-Energy Operations and Optimal Demand Response. IEEE Design and Test of Computers 29, no. 4: 8-16, 2012.

Price, Phillip. Methods for Analyzing Electric Load Shape and its Variability. Lawrence Berkeley National Laboratory Report LBNL-3713E, May 2010. http://eetd.lbl.gov/sites/all/files/LBNL-3713E.pdf

Price, Phillip. Jessica Granderson, Michael Sohn, Nathan Addy, and David Jump. Commercial Building Energy Baseline Modeling Software: Performance Metrics and Method Testing with Open Source Models and Implications for Proprietary Software Testing. PG&E Report ET12PGE5312, September 2013.

Glossary

BLS	Baseline electric load shape
BTO	Building Technologies Office
CPP	Critical Peak Pricing
CuSum	Cumulative Sum
DOE	Department of Energy
DR	Demand response
ECM	Energy Conservation Measure
EE	Energy Efficiency
HVAC	Heating,ventilation, air conditioning
ISO	Independent System Operator
JSON	Javascript Object Notification
kW	kilowatts
kWh	kilowatt-hours
LBNL	Lawrence Berkeley National Laboratory
M&V	Measurement and Verification
OpenADR	Open Automated Demand Response
ORNL	Oak Ridge National Laboratory
PNNL	Pacific Northwest National Laboratory
RTU	Roof top unit
sMAP	Simple Measurement and Actuation Profile
TN	Transactional Network
TOU	Time-of-Use
UTC	Coordinated Universal Time
UUID	Universally unique identifier
VEN	Virtual end node in OpenADR2.0
VL	VOLTTRON Lite™
VTN	Virtual top node in OpenADR2.0

Appendices

Terminology used here:

Tool describes a common function used by more than one agent.

Agent is an application focusing on a single task that communicates via the VOLTTRON Lite™ bus

Savings indicates a decrease in energy use, relative to the projected baseline load. Negative savings represent an increase in energy use relative to the projected baseline load.

Appendix A: Load Shape Details

Introduction

The loadshape tool is included in each agent, with the wrapper of each particular agent providing the context to identify what aspects of the loadshape tool are needed. The loadshape tool generates the information needed to compare actual electric loads with predicted (baseline) electric loads. The statistical model used by this tool considers two distinct aspects of load: those that have a pattern that is consistent based on the time of week (day of the week and time of day) and those that are sensitive to outdoor air temperature. To download the loadshape library go to https://pypi.python.org/pypi/loadshape/

The Loadshape class that is provided by this tool makes it easy to manage time series electric load data, and exposes a simple interface to several underlying R functions, including the function that fits a statistical model to the input load data for the purposes of generating baselines.

Input Data

The only input data required by the loadshape tool is a set of time-series electric load data:

```
# electric load data should be provided as a List of tuples
load_data = [ ("2013-08-01 00:00:00", 5.168),
              ("2013-08-01 00:15:00", 6.235),
              ("2013-08-01 00:30:00", 5.021),
              ...,
              ("2013-09-26 23:45:00", 4.739) ]

my_loadshape = Loadshape(load_data=load_data)
```

As shown above, the load data must be provided in the form of a Python List containing Tuples with two elements each. The first element of each Tuple is a timestamp, and the second element is a value representing power (kW).

Timestamps

Timestamps may take several different forms. Any of the timestamps below are valid:

```
valid_load_data = [("2013-08-01 00:00:00", 5.168), # string: "YYYY-MM-DD HH:MM:SS"
                   (1375341300, 6.235),            # integer: seconds since Unix epoch
                   (1375342200000, 5.021),         # integer: milliseconds since Unix epoch
                   ("1375343100", 5.046),          # string: seconds since Unix epoch
                   ...,
                   ("1380264300000", 4.739) ]      # string: milliseconds since Unix epoch
```

Timezones

The timezone associated with the timestamps in the input data should be specified using the appropriate time zone name from the tz database. This time zone name should be passed as an argument to each new instance of the Loadshape class. If no timezone is specified, the module will use the timezone of the operating system.

```
my_loadshape = Loadshape(load_data, timezone="America/Los_Angeles")
```

Specifying the timezone is important to (1) maintain consistency between data from different sources (e.g. weather data and load data); (2) properly handle daylight savings time; and (3) ensure that each local day begins at midnight in the internal calculations.

Power Data

Values within the provided time-series load data are assumed to represent power (kW). This is especially important because units specified by the output of the event_performance method assumes that the power data has been provided in kW.

Outdoor Air Temperature Data

Including outdoor air temperature data in addition to electric load data will allow the loadshape module to produce much more accurate baselines. The units of the temperature data may be configured by passing a temp_units argument of either "C" or "F" to the Loadshape object's initializer.

```
# electric load data - values are expected to be power (kW)
load_data = [ ("2013-08-01 00:00:00", 5.168),
              ("2013-08-01 00:15:00", 6.235),
              ("2013-08-01 00:30:00", 5.021),
              ...,
              ("2013-09-26 23:45:00", 4.739) ]

# outdoor air temperature data
temp_data = [ ("2013-08-01 00:00:00", 54.23),
              ("2013-08-01 01:00:00", 54.60),
              ("2013-08-01 02:00:00", 54.65),
              ...,
              ("2013-09-26 23:45:00", 58.44) ]

my_loadshape = Loadshape(load_data, temp_data, temp_units="F")
```

Output Data

Instead of passing input data to the Loadshape initializer as a List of tuples, data may also be passed by referencing appropriately formatted comma separated variable (CSV) files:

```
my_loadshape = Loadshape("path/to/load_data.csv", "path/to/temperature_data.csv")
```

If this option is used, the loadshape module expects CSVs to contain two columns. As with the tuples, the first element in each column must be a valid timestamp, and the second column must be the corresponding load value.

Generating Baselines

The Loadshape object uses a baseline method that compiles the input data, passes the data to the R script that implements the baseline model, and then reads in the result. The baseline method will return an object (a Series object) containing the baseline data. The “data” method on this object is the preferred method for accessing the list of tuples containing the time series baseline data.

```
>>> my_baseline = my_loadshape.baseline()
>>> my_baseline.data()
[(1375340400, 5.1), (1375341300, 5.1), (1375342200, 5.26), ..., (1380264300, 4.9)]
```

Prediction Periods

By default, the baseline method on the Loadshape object will return a baseline for all of the input load data. To calculate the baseline for a specific period, identify that time period with additional arguments to the baseline method:

```
prediction_start = "2013-09-26 00:00:00"
prediction_end = "2013-09-26 23:45:00"

my_baseline = my_loadshape.baseline(prediction_start, prediction_end, step_size=900)
```

The step size argument above is optional, the default is 900 (seconds). Also, note that the prediction_start and prediction_end do not need to be within the date range of the input data; the module may be used to generate forecasted baselines.

Forecasting with Outdoor Air Temperature Data

To produce a temperature adjusted baseline, the module requires outdoor air temperature data that overlaps both the input load data and the prediction period.

To generate a forecasted baseline, split the temperature data into two streams: one containing historical temperatures that overlaps the historical load data, and one containing forecasted temperatures that overlaps the desired prediction period.

If no temperature data is available for the requested prediction period, then the model will not be temperature adjusted. The resulting baseline will be the same as if no temperature data had been provided.

```
# electric load data - values are expected to be power (kW)
load_data = [ ("2013-08-01 00:00:00", 5.168),
              ("2013-08-01 00:15:00", 6.235),
              ("2013-08-01 00:30:00", 5.021),
              ...,
              ("2013-09-26 23:45:00", 4.739) ]

# outdoor air temperature data
temp_data = [ ("2013-08-01 00:00:00", 54.23),
              ("2013-08-01 01:00:00", 54.60),
              ("2013-08-01 02:00:00", 54.65),
              ...,
              ("2013-09-26 23:45:00", 58.44) ]

# forecasted outdoor air temperature data
forecast_temp_data = [ ("2013-09-27 00:00:00", 52.15),
                       ("2013-09-27 01:00:00", 52.40),
                       ("2013-09-27 02:00:00", 51.85),
                       ...,
                       ("2013-09-27 23:45:00", 60.31) ]

my_loadshape = Loadshape(load_data, temp_data, forecast_temp_data)
my_loadshape.baseline("2013-09-27 00:00:00", "2013-09-27 23:45:00")
```

Exclusion Periods

If parts of the load data are anomalous, they can be omitted by registering exclusion periods from the baseline calculation. For example, if different energy management strategies have been tested and a baseline is needed to calculate energy savings from a particular strategy, then all of the periods during which other strategies were being tested should be excluded, so that only periods of normal operation are included in the baseline calculation.

```
my_loadshape.add_exclusion(first_exclusion_start, first_exclusion_end)
my_loadshape.add_exclusion(second_exclusion_start, second_exclusion_end)
```

Named Exclusion Periods

The Loadshape module also includes a mechanism for excluding periods of data that are likely to be anomalous, such as Holidays:

```
my_load_shape.add_named_exclusion("US_HOLIDAYS")
```

Note that the current implementation of named exclusions is not very sophisticated. Named exclusions currently consist of a hard-coded list of periods corresponding to holidays that are observed by Lawrence Berkeley National Laboratory.

Modeling Interval

The modeling interval determines the resolution of the model that is used to make predictions. Higher resolution models will run more slowly. By default, the modeling interval is set to 900 seconds. The argument that defines the modeling interval is passed to the baseline method as shown below.

```
my_baseline = my_loadshape.baseline(modeling_interval=300)
```

Weighting

The "weighting_days" argument allows the model to be biased toward more (or less) recent data. The default value is 14 days, meaning the most recent 14 days of training data will be weighted more heavily than data that is older than 14 days. To configure the weighting differently, pass a weighting_days argument to the baseline method.

```
my_baseline = my_loadshape.baseline(weighting_days=30)
```

Goodness of Fit Statistics

Once a baseline has been generated, some goodness of fit statistics will be available in the form of a dictionary:

```
>>>my_loadshape.baseline()
>>>my_loadshape.error_stats
{'rmse_interval': 1.723, 'corr_interval_daytime': 0.88, 'rmse_interval_daytime': 2.421,
'mape_hour': 11.343, 'mape_interval': 12.858, 'rmse_hour': 1.553,
'mape_interval_daytime': 19.576, 'corr_interval': 0.908, 'corr_hour': 0.92}
```

The goodness-of-fit statistics calculated are Root Mean Squared Error (RMSE) and Mean Absolute Percent Error (MAPE, sometimes called Mean Absolute Percentage Error).

These statistics are calculated for each time interval in baseline series (typically 15-minute intervals); for each time interval in the “daytime,” from 8am-6pm; and also for load data and baseline predictions.

Measurement and Verification

Streamlined calculation of baseline loadshapes is useful, but in most cases, baselines are being calculated for the purposes of comparing the predicted baseline to an actual load shape. The Loadshape tool provides several methods that make this comparison simple.

Difference Method

The Loadshape class includes a diff method for calculating the difference between a baseline and an actual load shape. This method passes the baseline time series and actual load time series to an R script, which interpolates the two streams and generates four streams of data:

- kW difference (difference at each interval between actual and baseline)
- cumulative kWh difference (accumulated kWh difference at each interval between actual and baseline)
- kW baseline (interpolated baseline kW at each interval)
- cumulative kWh baseline (cumulative interpolated baseline kWh at each interval)

The difference method generates these to simplify the calculation of the magnitude of the calculated differences relative to the baseline.

Cumulative Sum Method

The Loadshape class includes a cumulative_sum method for calculating the cumulative difference between a baseline and the actual load shape. The cumulative_sum method is a convenience method that simply wraps the diff method and returns only cumulative kWh difference stream. The cumulative_sum method also ensures that a baseline is available with which to compare the actual load shape data; if a baseline is not available, the method automatically generates one using the default arguments.

Economic Valuation (Event Performance Method)

The Loadshape class includes an event_performance method that is purpose built for comparing the performance of a loadshape to a baseline over a specific period of time. The period over which this comparison is calculated could be an arbitrary length of time, but in practice this method is useful for calculating load performance relative to baseline on specific days when the load may be operating in a particularly energy efficient mode, or when a new optimization is being tested. Below is an example of usage:

```
my_load_shape = Loadshape(load_data=LOAD_DATA, temp_data=TEMP_DATA,
                          timezone='America/Los_Angeles',
                          temp_units="F", sq_ft=BUILDING_SQ_FT)

# ----- build the baseline to use as a reference for performance ----- #
event_baseline = my_load_shape.baseline(weighting_days=14,
                                        modeling_interval=900,
                                        step_size=900)

# ----- calculate the performance summary for the event period ----- #
event_performance = my_load_shape.event_performance(EVENT_START,
EVENT_END)
```

The output of the event performance method will include these calculated quantities:

- average kW reduction relative to baseline
- average percent kW reduction relative to baseline
- average Watts per square foot reduction relative to baseline (if the Loadshape object was instantiated with a sq_ft argument)
- total kWh reduction relative to baseline
- percent kWh reduction relative to baseline
- total savings ($)*
- total percent savings*

*included only if the Loadshape object was instantiated with a tariff (see below)

The "dr-event-calc.py" example in the examples directory demonstrates how this event_performance method can be used to calculate load performance during a demand response event.

Tariffs

The Loadshape class includes a cost method that enables the calculation of the cost of energy for a load based on a specific tariff. In order to use this functionality, a tariff object must be passed into the Loadshape object using the set_tariff method. A Tariff object should be instantiated with a json formatted tariff file from openei.org. An example of a valid tariff file is included in examples/data/tariff.json. The below example demonstrates how a Tariff object should be initialized and passed to the Loadshape object.

```
tariff = Tariff(tariff_file='example_tariff.json', timezone='America/Los_Angeles')
tariff.add_dr_period("2013-09-23 14:00:00", "2013-09-23 16:00:00")
tariff.add_dr_period("2013-09-27 14:00:00", "2013-09-27 16:15:00")

my_load_shape.set_tariff(tariff)
```

Note that specifying DR periods, as shown above, is optional. Adding these DR periods will ensure that the DR day tariff that is specified in the tariff JSON is used during the periods specified. Also, note that if a Loadshape object has a tariff set, the event_performance method will use the cost method that is described below to calculate the financial savings during the event period.

After a tariff has been set for a loadshape object, as shown above, the cost method may be used to calculate the cost of energy and the cumulative cost of energy at each interval of the data provided to the load_data argument. If no load_data argument is provided, the input data will default to the actual load data. The example below shows how the cost data for a baseline load shape can be calculated.

```
my_load_shape.set_tariff(tariff)

#c: cost  cc: cumulative cost
c, cc = my_load_shape.cost(load_data=my_load_shape.baseline_series.data(),
                           start_at=start_at,
                           end_at=end_at)
```

Appendix B: Agent Details

Introduction

LBNL developed three agents that thinly wrap the loadshape tool to interact with VL. These agents serve as a simple interface to the loadshape tool so that agents on the VL platform can use the capabilities of the loadshape module without having to declare it as a dependency. The agent wrappers create a consistent interface for different applications of the loadshape tool, with limited additional computing overhead. Essentially the use of these agent wrappers also allows for code optimization and reuse within different environments. An example of this was described in the main report under Prototype Applications.

The three agents contained within this repository are:

- Baseline Load Shape agent
- M&V agent
- Economic Valuation agent

As the names indicate, each of these agents exposes a different piece of functionality provided by the loadshape module.

Baseline Load Shape Agent Usage

To request a baseline from the Baseline agent, a requesting agent would publish a message to the **baseline/request** topic using the publish_json method.

An example message is shown below:

```
example_message = {
  "load_data": [(1379487600, 5), (1379488500, 5), ... (1379491200, 5)],
  "temp_data": [(1379487600, 72), (1379488500, 72), ... (1379491200, 72)],
  "timezone": 'America/Los_Angeles',
  "temp_units": "F",
  "sq_ft": 5600,
  "weighting_days": 14,
  "modeling_interval": 900,
  "step_size": 900
  }
```

Except for "load_data" all keys are optional.

The contents of this message will be passed directly to the loadshape module and a baseline will be calculated using the arguments provided. Once the baseline calculation has completed, the Baseline agent will publish a message to the **baseline/responses/[requesting-AgentID]** topic. The message published to this topic will contain the requested baseline, as well as the error statistics that describe how well the baseline fits the training data.

Measurement & Verification (cumulativesum) Agent Usage

To request a cumulative sum calculation from the Cumulative Sum agent, a requesting agent would publish a message to the **cumulativesum/request** topic using the publish_json method.

An example message is shown below:

```
example_message = {
  "load_data": [(1379487600, 5), (1379488500, 5), ... (1379491200, 5)],
  "temp_data": [(1379487600, 72), (1379488500, 72), ... (1379491200, 72)],
  "timezone": 'America/Los_Angeles',
  "temp_units": "F",
  "sq_ft": 5600,
  "step_size": 900
  }
```

Except for "load_data", all keys are optional.

The contents of this message will be passed directly to the loadshape module and a cumulative sum will be calculated using the arguments provided. Once the cumulative sum calculation has completed, the Cumulative Sum agent will publish a message to the **cumulativesum/responses/[requesting-AgentID]** topic. The message published to this topic will contain a time series of kWh difference between the provided load data and the calculated baseline.

Event Valuation (eventperformance) Agent Usage

To request an event performance calculation from the Event Performance agent, a requesting agent would publish a message to the **eventperformance/request** topic using the publish_json method.

An example message is shown below:

```
example_message = {
  "load_data": [(1379487600, 5), (1379488500, 5), ... (1379491200, 5)],
  "temp_data": [(1379487600, 72), (1379488500, 72), ... (1379491200, 72)],
  "timezone": 'America/Los_Angeles',
  "temp_units": "F",
  "sq_ft": 5600,
  "start_at": "09-27-2013 00:00:00",
  "end_at": "09-28-2013 00:00:00"
  }
```

Except for "load_data" all keys are optional, but in nearly all cases "start_at" and "end_at" times should be provided.

The contents of this message will be passed directly to the loadshape module and a set of event performance statistics will be calculated using the arguments provided. Once the event statistics calculations have completed, the Event Performance agent will publish a message to the **eventperformance/responses/[requesting-AgentID]** topic. The message published to this topic will contain a set of event performance statistics that characterize the performance of the actual load relative to the calculated baseline during the time period provided.

Appendix C: Load Performance Assessment Example

This section shows how to use the Loadshape module to assess the performance of a load relative to its normal operation during a specific time period, such as a demand response event.

Specifically, this example outlines how to:

- generate a baseline for the load during a specified time period
- generate a “difference time-series”, or a time-series that is the result of subtracting the calculated baseline from the actual load data

Loadshape Module Inputs

In order to perform this calculation, four inputs will be necessary:

- **time-series load data** - This data will form the basis of the baseline prediction.
- **time-series outdoor air temperature data** - This data will enhance the baseline prediction by establishing a temperature dependence.
- **prediction time-series outdoor air temperature data** - This data will be used to make a baseline prediction using the model generated from the historical load and temperature data.
- **event start / event stop** - The will be used to select the appropriate portions of the time-series data that is passed in to the Loadshape module.

Figure 19 shows the inputs required by the Loadshape module to calculate a baseline that will be compared against the actual load data. The solid portion of each line represents data that must be provided to the Loadshape module. The dashed portion of each line represents data that is not needed by the calculation. Since the Loadshape module timeslices the input data, unnecessary data will simply be ignored.

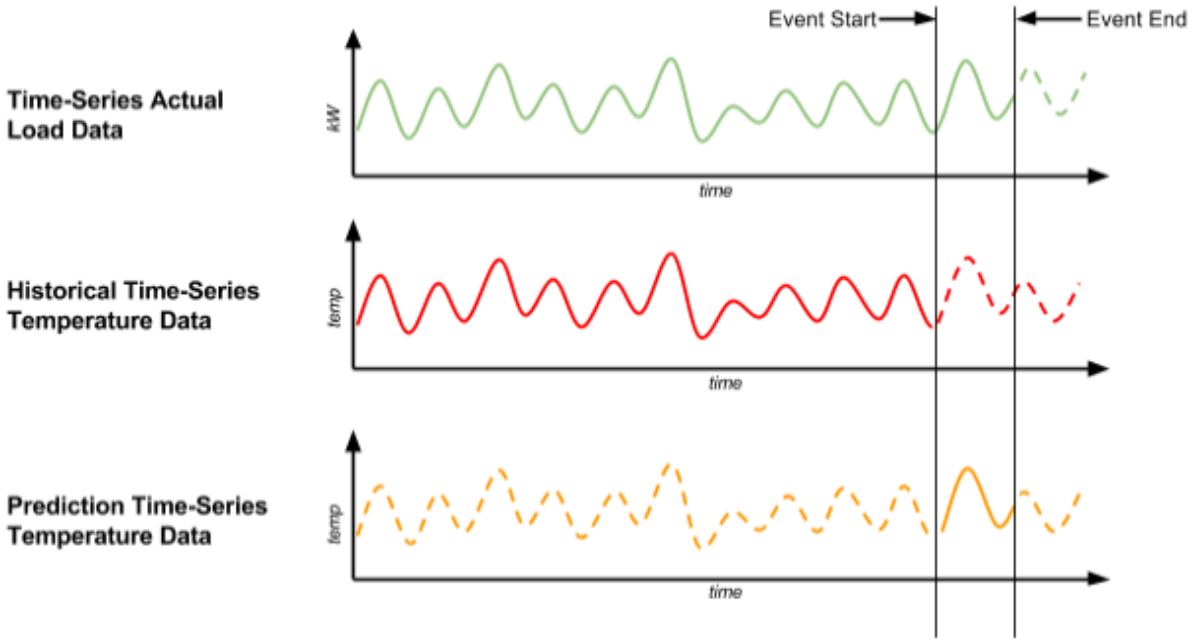

Figure 19. Diagram of inputs necessary for performing a temperature sensitive baseline prediction

In most use cases, the “historical” time-series outdoor air temperature data (in red) and the time-series outdoor air temperature data associated with the prediction period (in yellow) come from the same source. Because of this, the Loadshape module does not require that “prediction” temperature data be passed in separately from “historical” temperature data.

As noted earlier in the baseline description, “historical” temperature data is necessary to generate a temperature dependent baseline model of the load behavior. If “prediction” temperature data is provided, it will be used for predicting the baseline, however, if it is not provided, the temperature data necessary to generate the baseline prediction will be extracted from the “historical” temperature data. In other words, all temperature data may be passed in as a single “historical” stream. The rest of this example will define all temperature data in a single stream as described here.

Loadshape Module Baseline Generation

An R script contained within the Loadshape module constructs a statistical model using time-series load data and time-series outdoor air temperature data. This model is then used to make a baseline prediction for a specific time period

Typically, the purpose of generating a baseline is to characterize what the normal behavior of a load would have been during a specific prediction period during which there was some kind of load perturbation (a demand response event, for example). Data from the prediction period is excluded from the generation of the model. Data from other time periods (such as holidays) may also need to be excluded. The Loadshape module accommodates this by allowing the user to define “exclusion periods”. Data from the exclusion periods are not used when fitting the model.

Loadshape Module Conventional Baseline Generation

By convention, baselines are typically generated from data collected during the “training period” prior to the beginning of the baseline prediction period. As discussed earlier, by including time-series load and temperature data on either side of the prediction period, but not data acquired during the prediction period itself, we can obtain a more accurate baseline. The Loadshape module accommodates either of these use cases (and others too): the user provides load data and temperature data for one set of timestamps to be used for training the model, and temperature data for the timestamps for which a prediction is required, and the model will provide the predictions; there is no requirement that the prediction times must all be after the training times.

Loadshape Module: Subtracting Time-Series Data

Once the predicted baseline has been generated (using the process described above), the difference between the baseline and the actual load data may be calculated. For the purposes of this calculation, the Loadshape module does not observe the exclusion period that was defined for the purposes of generating the baseline. The difference calculation that is built into the Loadshape module will subtract the time-series load data from the calculated baseline data wherever values are present.

Figure 20 illustrates a typical use case for the Baseline module and the M&V (Difference) module. The top panel shows load as a function of time during the training period. Day 4 has been excluded because it is a holiday, when load behavior might be different than typical days. The second panel shows the outdoor air temperature as a function of time for both the training period and the subsequent prediction period. The third panel (blue) shows the predicted baseline during the prediction period, as generated by the baseline model fit to the training period. The fourth panel (black) shows the actual load during the prediction period. Finally, the fifth panel (dark green) shows the difference between the actual load and the baseline load during the prediction period, as generated by the M&V (Difference) module.

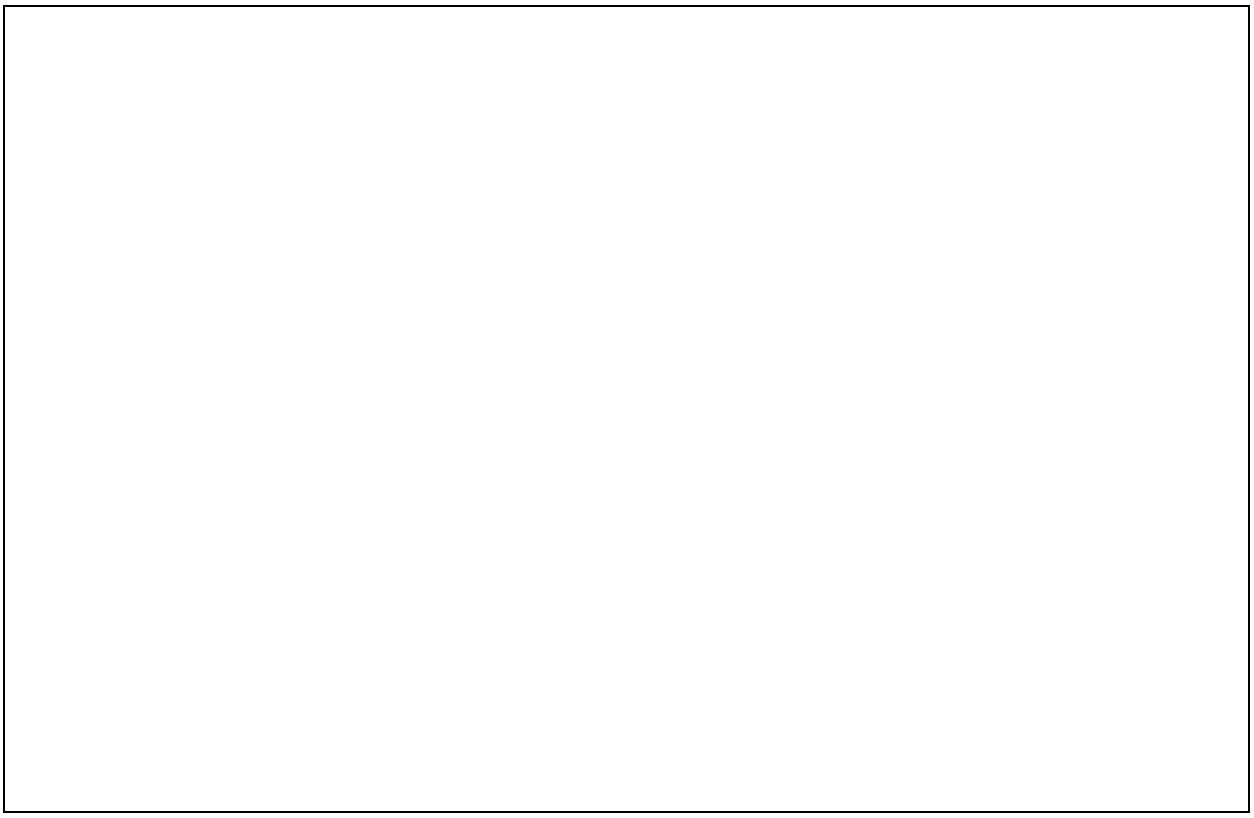

Figure 20: Time series plots that illustrate the creation of a baseline prediction, and comparison of baseline to actual load.

The Difference Calculation

The following section outlines how to use the Loadshape module to calculate a time-series containing the difference between the actual load data and a calculated baseline during a specified time period. The description below is simplified: some of the steps described in the conceptual overview are assumed to be completed prior to taking the difference, so they are not shown here.

In this example we will be calculating the difference between the actual load profile and a calculated baseline for the following event period that spans one day:

- event period start: "2013-09-27 00:00:00"
- event period end: "2013-09-28 00:00:00"

Step 1: Instantiate a new Loadshape module

- pass in time-series load data
- pass in time-series outdoor air temperature data
- set timezone of the time-series timestamps
- set the units of the temperature data

```
my_load_shape = Loadshape(load_data='load.csv',
                          temp_data='temp.csv',
                          timezone='America/Los_Angeles',
                          temp_units='F')
```

Step 2: Add an exclusion period

add an exclusion period to the Loadshape object - in this example a conventional baseline is used, therefore all data after the beginning of the event period will be excluded. This type of exclusion can be achieved by setting an exclusion end date that is later than the end of the input data, or to be safe, one that is well in the future.

```
my_load_shape.add_exclusion("2013-09-27 00:00:00", "2020-01-01 00:00:00")
```

Step 3: Calculate the difference

Now that everything is set up, calculating the difference is simply a matter of calling the “diff” method.

```
diff = my_load_shape.diff("2013-09-27 00:00:00", "2013-09-28 00:00:00")
```

Note that the Loadshape diff method will return a python List containing four Series objects. The first element in this array will be the Series that contains the difference between the actual load data and the baseline.