A DISTRIBUTED INTELLIGENT AUTOMATED DEMAND RESPONSE BUILDING MANAGEMENT SYSTEM

1 DOE Award Number & Name of Recipient

1.1 DOE Award Number:

DE-EE0003847

1.2 Recipient:

Regents of the University of California, (UC Berkeley campus), with Lawrence Berkeley National Lab and Siemens Corp as sub-awardees.

2 Project Title and Principal Investigators

2.1 Project Title:

A Distributed Intelligent Automated Demand Response Building Management System

2.2 Task Title:

Task 6: DIADR local control testing in a laboratory environment

2.3 Date of Report:

7 September 2011

2.4 Principal Investigators:

Professor David Auslander, Department of Mechanical Engineering, U.C. Berkeley (UCB)
Professor David Culler, Department of Electrical Engineering, U.C. Berkeley (UCB)
Professor Paul K. Wright, Director, Center for Information Technology Research in the Interest of Society (CITRIS) and the Department of Mechanical Engineering, U.C. Berkeley (UCB)
Dr. Yan Lu, Siemens Corporate Research Inc (SCR)
Mary Ann Piette, Lawrence Berkeley National Laboratory (LBNL)

2.5 Key personnel:

Thomas Gruenewald, Siemens Corporate Research Inc (SCR)
Prasad Mukka, Siemens Corporate Research Inc (SCR)
Sila Kiliccote, Lawrence Berkeley National Laboratory (LBNL)
Dr. Therese Peffer, Project Manager, U.C. Berkeley and CIEE (UCB)

2.6 Authors:

Daniel Arnold, Graduate Student Researcher, UC Berkeley, Mechanical Engineering
Nathan Murthy, Student Researcher, UC Berkeley, Applied Mathematics
Michael Sankur, Graduate Student Researcher, UC Berkeley, Mechanical Engineering
Jason Trager, Graduate Student Researcher, UC Berkeley, Mechanical Engineering
Prasad Mukka, Siyuan Zhou, Siemens

Table of Contents

3 Executive Summary

The Distributed Intelligent Automated Demand Response (DIADR) project endeavors to develop an energy management system with intelligent optimization and control algorithms to achieve 30% peak load reduction while still maintaining the building as a healthy, productive, and comfortable environment for the building occupants. The selected building site is Sutardja Dai Hall at UC Berkeley. Four earlier reports outlined the System Architecture, the Building Management System integration with the OpenADR protocol that administers the demand response signals, the Service Oriented Gateway, and the development of central and distributed load control algorithms. This report describes the tests of the local control in the test office environment, Room 464 of Sutardja Dai Hall.

This report describes the distributed load control architecture and the final implementation of the Energy Information Gateway, and testing. Both teams from UC Berkeley and Siemens Corporate Research developed testbeds, simulation tools, demand response algorithms, and both demonstrated the capability of the gateway as a local controller. UC Berkeley demonstrated two Gateways running on computers in the same room that controlled typical office appliances with wired and wireless connections as well as simulated appliances. The DR strategy employed the reduction of loads given a priority assigned to each appliance. SCR showed a simulation of the Smart Energy Box, two Gateways, and simulated appliances using a multi-agent approach with a cost function to reduce energy consumption.

This research will add to the body of demand response research by pushing the boundaries of typical DR goals. While a typical DR goal for commercial buildings is on the order of 10%, this project plans to triple that energy reduction by expanding the role of control to distributed nodes. Reliance upon centralized pre-programmed controllers to take action based on a demand response signal can result in a loss in the system responsiveness to the dynamic changes of energy price, occupancy patterns, load requirements as well as weather conditions. The distributed approach embeds as much autonomy as possible in local sections of the network to enable distributed optimization and control functions.

The techniques involved in this work mostly rely on software; this implies that applying this process to other buildings is relatively inexpensive. Intelligent algorithms rely on distributed sensors (hardware) to provide information. However, a goal of this project is to optimize economic feasibility, by balancing information points with cost to provide the best control strategies with the fewest sensing points.

Demand Response in general aims to reduce peak electricity consumption, which can benefit the public by slowing the pace of creating new power plants and reducing air pollution created by older peaking power plants. This project not only deepens a building's demand response, but also actively engages occupants in order to maintain a productive work environment.

4 Comparison of Project Accomplishments and Project Goals

The DIADR local control (Tier 3) testing in a laboratory environment (Task 6) includes six subtasks. The first subtask was local control DIADR integration. The second subtask entailed the testbed architecture design, followed by the Service Oriented Architecture integration into the room test bed, and the DR algorithm integration into the room test bed. The fifth subtask was the test and optimization. Finally, the six task involved room demonstration.

The UC Berkeley team set up room 464 in Sutardja Dai Hall as a smart office laboratory for local control testing. The SCR team also developed a similar space locally as a testbed. Michael Sankur and other students from UC Berkeley integrated the agent-based distributed load management system (Energy Information Gateway with which plug loads in the building will be controlled directly), into this test bed; the Gateway development is discussed in the Task 3 report available at http://i4energy.org/diadr-project-sutardja-dai-hall-0. The DR algorithms utilize a prioritization of appliances (such as a lamp and fan) to be managed during a DR event by the Gateway. Nathan Murthy, Kevin Ding, and Dan Arnold developed a web-based user interface for the Gateway and integrated it into the test bed. The user interface allows office workers to enter personal appliances and set priorities; it can also collect local parameters for the HVAC system and lighting control and submits them to the central control system for further strategy implementation. Siemens introduced the UC Berkeley team to JADE as the control architecture for the interaction between the central load management system and distributed load management system, setting up the protocol for communication between the two acting agents. Both teams created simulations for testing. Siemens developed several optimization scenarios, and developed a few for testing. UC Berkeley showed a demonstration of two gateways in a single room controlling various real and simulated smart appliances to reduce peak load consumption. Siemens Corporate Research demonstrated the multi-agent demand response negotiation.

5 Project Summary

5.1 Introduction

Sutardja Dai Hall is a 141,000 square foot building on the UC Berkeley campus that houses the Center for Information Technology Research in the Interest of Society (CITRIS). Much of the building occupancy is dedicated to offices, with a few classrooms, an auditorium, and café; the nanofabrication lab is not included in the project's energy reduction goals. The whole building demand load is approximately 940kW.

The diagram below describes the basic system architecture: a demand response signal is sent by the Demand Response Automated Service (DRAS) and is received by the Siemens Smart Energy Box (SEB) installed in Sutardja Dai Hall at UC Berkeley. The SEB then generates a load shedding goal sent to the Building Automation System for the HVAC system, the WattStopper lighting system, and to the distributed load gateways that control appliances.

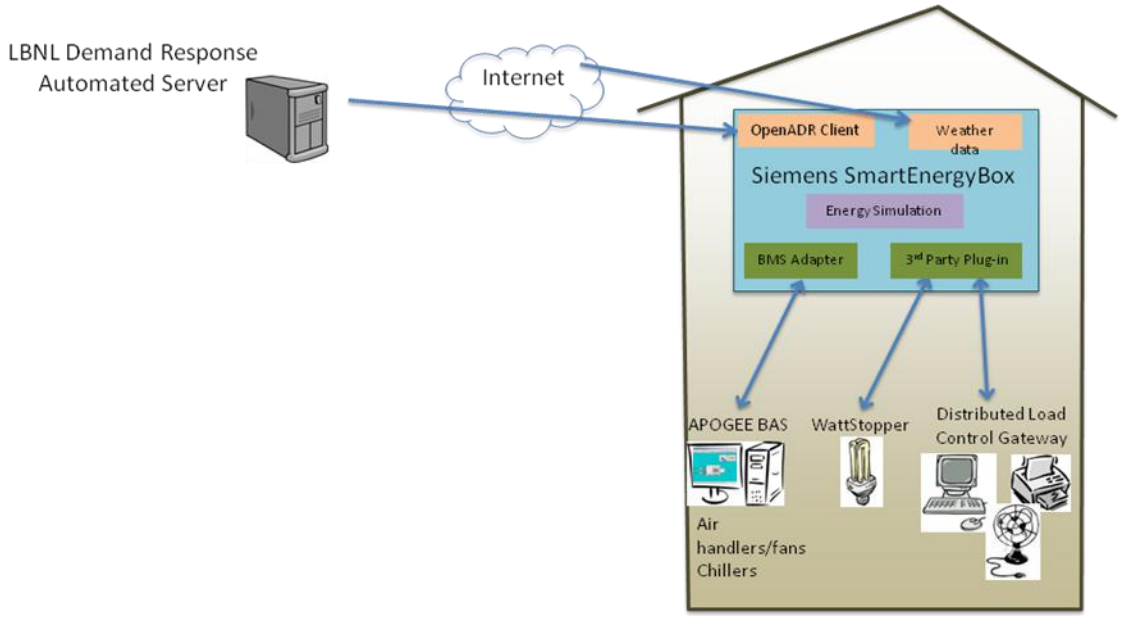


Figure 1: Overall DIADR control management system.

Siemens met with the facilities manager to determine which elements of the building could be subject to demand response control and which were not allowed. (The building houses an energy-intensive nanofabrication laboratory that is not included in the energy reduction goals of the project, but shares chilled water with the whole building). Initial control strategies include: expand the allowable building zone setpoints (minimum to 69F and maximum to 78F), precooling to 69F before occupied, change duct static pressure and supply air speed, and turn off some air handling units, supply fans, and exhaust fans. For lighting, some overhead lighting can be dimmed (stepped dimming). UC Berkeley conducted a plugload audit of the building to determine loads that could be controlled locally by a Commercial Energy Gateway, such as computers, monitors, printers, coffee/tea pots, task lamps, and portable heaters and fans.

5.2 Distributed DR load control architecture

The architecture of the distributed DR load control for the DIADR project is the Smart Energy Box communicating with multiple gateways each controlling multiple typical office appliances. See schematic in Figure 2 below.

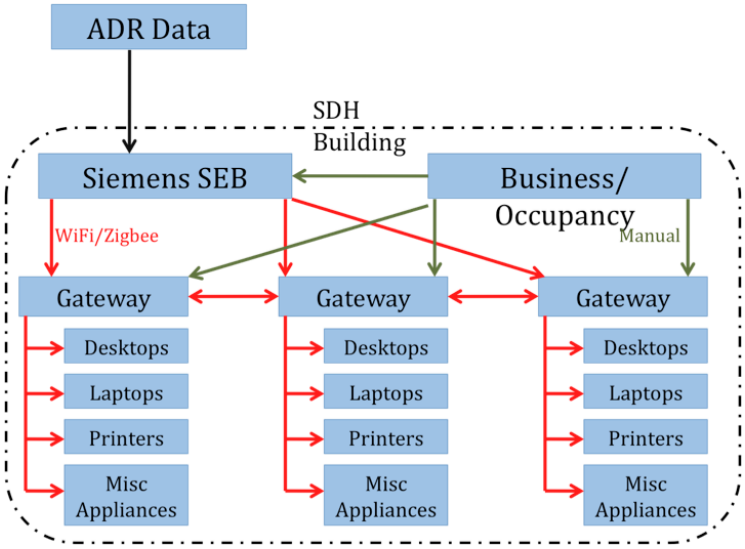


Figure 2: Gateways as distributed load controllers.

For the purposes of this project, the final demonstration is expected to implement just a few gateways, covering one thermal zone of the building to demonstrate the proof of concept. However, we will demonstrate the full capability of implementing gateways across the entire building using a simulation tool. The architecture of the Energy Information Gateway is shown below.

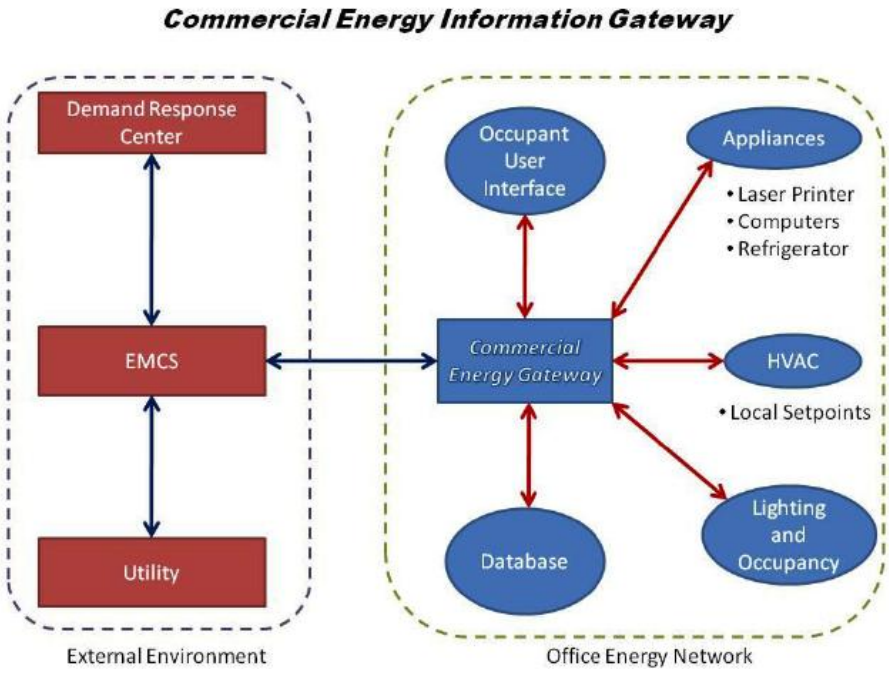


Figure 3: architecture of the Gateway.

Finally, the gateway interacts with the appliances via several communication methods. The figure below shows the relationship between the Gateway and underlying appliances in its control. Not shown is the communication with the ACme plugload monitors.

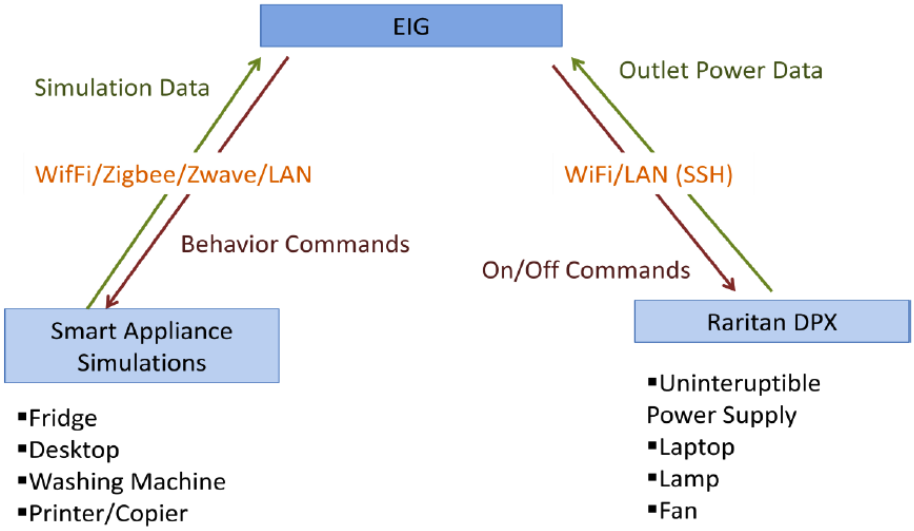


Figure 4: Communication between a Gateway and both real and simulated appliances.

5.3 Final Gateway implementation

The Energy Information Gateway was developed in Java, using the OSGi framework. The Gateway as used in the final demonstration has several bundles: the Raritan Driver from the UCB team, laptop power control, printer driver, the ApplicationControl bundle which has the new control logic with embedded optimization algorithm, and the DataAccessLayer bundle which controls the configuration file. The figure below shows the software architecture of the Gateway.

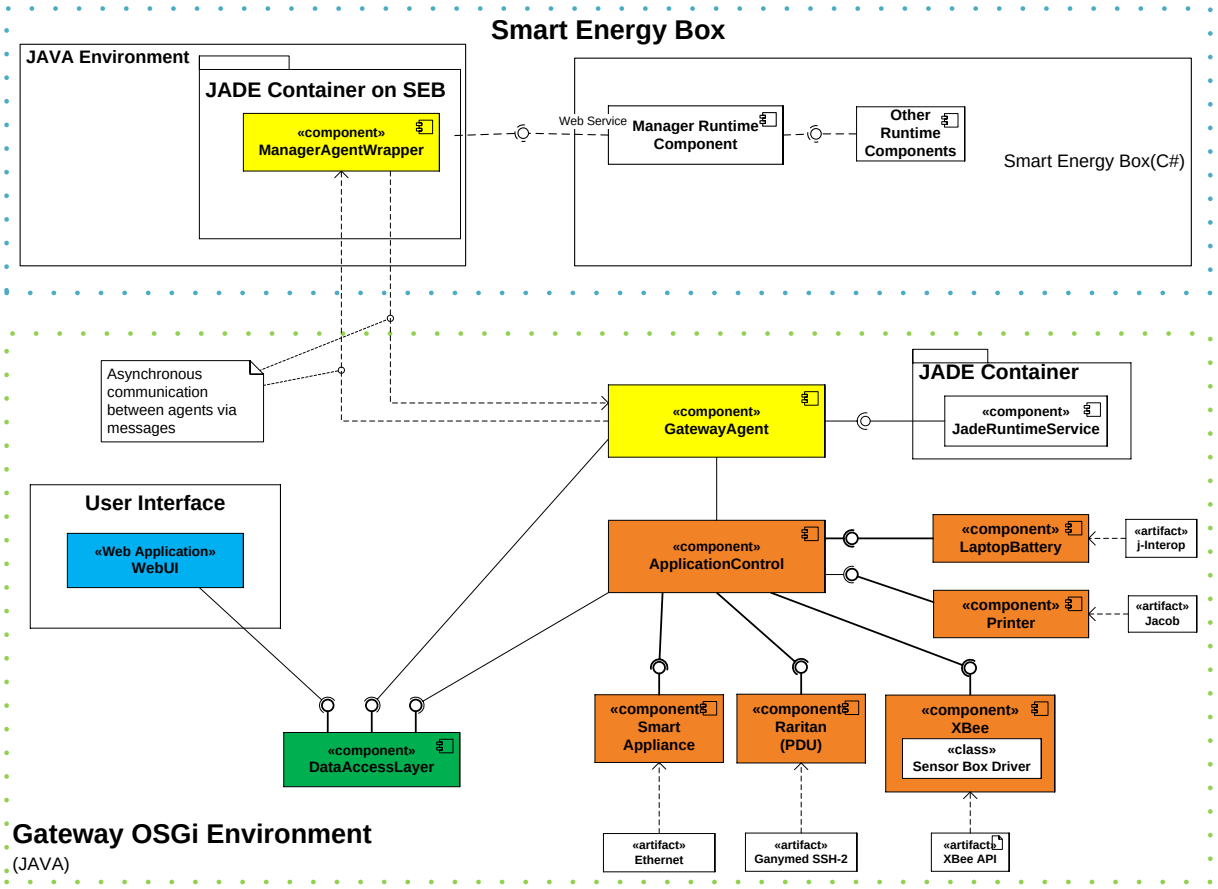


Figure 5: Gateway Software Architecture.

5.3.1.1 Smart Energy Box

The Smart Energy Box (SEB) is a device for commercial building automation dealing with advanced energy elements. The most important elements are a demand response gateway, energy simulation, and adopting energy efficient control strategies (both built-in and third party control integration) for commercial buildings. The system architecture supports the implementation of distributed control architecture where some parts of the building energy will be optimized by distributed systems while others are controlled and optimized centrally in the box.

The central load control algorithms use two different modes: Instant Mode and PeakDayPricing Mode (PDP).

The distributed load-control scenario establishes communication between Smart Energy Box and Gateway using JADE mechanism, and applies control on plug loads in response to DR events.

5.4 Physical test bed

5.4.1 UC Berkeley test bed—room 464 Sutardja Dai Hall

The test lab, room 464 in Sutardja Dai Hall, was developed as a smart office laboratory. The equipment selected to represent typical office equipment includes desktop and laptop computers, a small

refrigerator, a laser printer, small fan, and small heater. One Demand Response strategy is to supply the computers with power during a demand response event with UPS (Uninterruptible Power Supply) devices capable of supplying power during an entire demand response event. In addition, the lab has a combination of ACme plug-load electricity meters and Raritan Dominion PX8 metered and switched power strips for monitoring the power consumption of appliances. This instrumentation reports its data to sMAP, a physical data store. Currently, computers are acting as gateways. In addition to the actual appliances, several smart appliances are simulated.

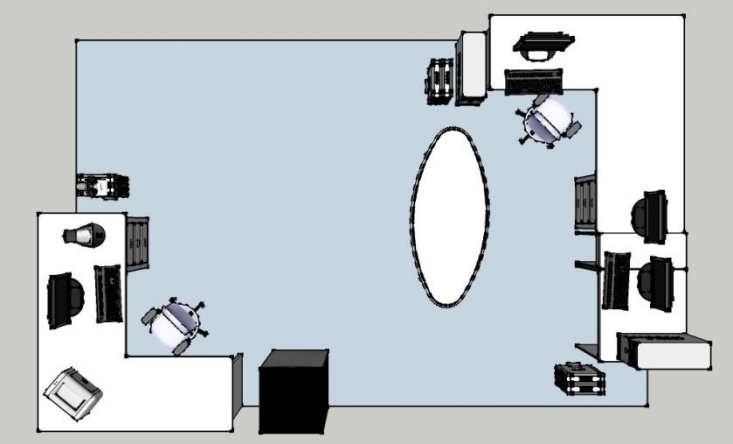


Figure 6: Floor plan of room 464 Sutardja Dai Hall.

5.4.2 Siemens' test bed

The physical test bed at SCR has a laptop, printer, task lamp, sensor, and smart appliance simulation. The AC power supply of task lights and laptop will be controlled by the Raritan PDU; the Smart Appliance Simulation is running on the laptop and will be controlled through an Ethernet connection, and the printer will also be controlled through an Ethernet connection. The figure below shows the layout of the physical test bed.

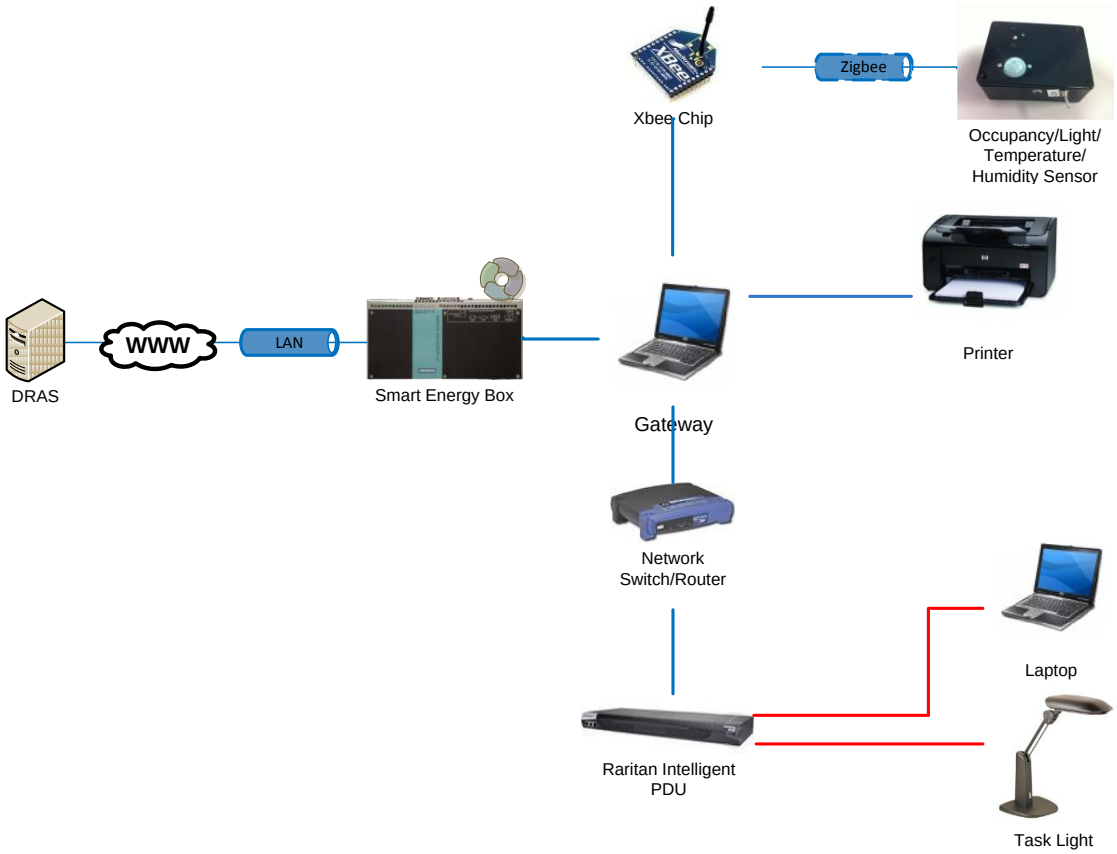


Figure 7: Physical Test Bed Layout.

5.4.2.1 Hardware and Software

The following configuration is required/used for the test and demonstration described:

- SIMATIC Microbox computer with Windows XP operating system (Smart Energy Box)
- Smart Energy Box software
- .Net Framework 3.5 SP1
- Demand Response Automation Server
- Internet accessibility for Smart Energy Box
- A desktop or laptop computer with two network card, running Windows XP operating system (Gateway)
- Internet accessibility for Gateway
- Gateway software
- Eclipse 3.0 or up with OSGi plug-in
- JADE
- Java
- Physical Appliances
    - XBee Pro Chip with COM-USB cable
    - Raritan Intelligent Power Distribution Unit
    - Ethernet switch
    - Sensor box built by Siemens Corporate Research
    - Two task lamps with bulbs

- Smart appliance Simulation (demo used smart appliances)
    - 4 Smart Appliance Simulations with maximum power consumption of 500W

5.5 Simulation

5.5.1 UC Berkeley building simulation—multiple gateways, people, appliances

The purpose of creating a building simulation is to evaluate the efficacy of the Gateway and DR algorithms. This is done by simulating a building, its inhabitants and appliances as well as a DR event for both a single Gateway zone and multiple Gateway zones.

While the Gateway can be tested in a real laboratory setting with actual appliances, implementing multiple Gateways in the building would be overly burdensome and expensive. Furthermore the use of accelerated time would not be an option meaning Gateway and algorithm testing would need to take place at an unacceptably slow pace.

5.5.1.1 Gateway Simulation Structure

As the Gateway is a local controller and will only cover a portion of a building, the simulation is broken up into zones. Each zone has one Gateway and represents a sphere of influence a real Gateway would have in a real building. To avoid complications with differentiating between zones in the simulation, each zone is given its own simulation instance. For example if three zones are to be tested, the simulation program will be started three separate times concurrently, each with different parameters corresponding to the zone it represents.

The simulation utilizes the OSGi framework for its structure. The actual Gateway also utilizes this framework; see the Task 3 report for more information. The simulation was programmed to take advantage of OSGi's features such as being able to turn on and off discrete modules of code known as bundles.

5.5.1.1.1 Bundles

The simulation consists of nine bundles that are started in a specific order. Each bundle has a specific function and component of the simulation. There are dependencies between certain bundles that lead to the start order.

The order in which the bundles are started is:

1.	Startup Bundle
2.	Time Service Bundle
3.	Appliance Bundle
4.	Control Bundle
5.	People Simulation Bundle
6.	Data Bundle
7.	JADE Bundle
8.	Gateway Agent Bundle
9.	Admin Bundle

5.5.1.1.2 Utility Bundle

The Utility Bundle contains the interfaces through which other bundles export their services. These interfaces are listed below:

Package	File	Description
.IAdminService	IAdminService	Interface for Admin Service
.IAppliance	IAppliance	Interface for Appliance portion of simulation
.IControl	IControl	Interface for simulated Gateway (controller)
.IData	IData	Interface for simulation data collection service
.IGatewayAgent	IGatewayAgent	Interface for JADE Gateway Agent
.IPeopleSimulation	IPeopleSimulationService	Interface for People portion of simulation
.IStartupService	IStartupService	Interface for Startup service
.ITimeService	ITimeService	Interface for Time Service

The Utility Bundle is not started as it can remain passive.

5.5.1.1.3 Startup Bundle

The Startup Bundle is the first bundle to be started by the OSGi framework. This bundle provides a variety of startup and generalization services for other bundles. The Startup Bundle takes in arguments from the command line or from a function call that starts the individual simulation runtime. These arguments describe the zone and file where the zone parameters can be found. The bundle parses the information and provides it to other bundles.

Future work for this bundle will be to read from Stream FS or a file to obtain which zone the simulation is representing and the parameters of the specific zone.

Service consumed from	Details
N/A	N/A
Service consumed by	Details
Time Service	Obtains simulation start and stop times and time step size for clock function
Appliance	Obtains types of appliances and appliance parameters
Control	Obtains control algorithm type, DR event type and time
People Simulation	Obtains the simulation zone and number of people to populate zone with
Data	Obtains pertinent data to collect and where to output data
JADE	Obtains zone number and JADE container properties
Gateway Agent Bundle	Obtains zone number and Gateway Agent properties
Admin	Obtains zone number

5.5.1.1.4 Time Service Bundle

The Time Service Bundle is the second bundle to be started by the OSGi framework. This bundle's main function is to serve as a central clock for each instance (zone) of the overall simulation. The bundle also provides a service to utilize the system clock so that all zone clocks are synchronized.

Service consumed from	Details
Startup	Obtains simulation start and stop times and time step size
Service consumed by	Details
Appliance	Obtains current time for appliance function and control
Control	Obtains current time for control and DR event action and monitoring
People Simulation	Obtains current time for time based person modeling
Data	Obtains current time timestamp collected data
Gateway Agent	Obtains current time for bargaining and negotiation with other agents

5.5.1.1.5 Appliance Bundle

The Appliance Bundle contains classes for a variety of appliances and a class to create and track multiple appliances.

Each type of appliance that is used in the simulation is described by its own class. Each appliance class has specific behaviors and methods unique to its real counterpart. Appliances are kept track of in array lists containing a multitude of each specific appliance object. For example there is an array list of desktop computer objects, an array list of desk lamp objects and one for fans.

Service consumed from	Details
Startup	Obtains number and type of appliances as well as appliance details
Time Service	Obtains current simulation time for simulated appliance operation
Service consumed by	Details
Control	Utilizes IAppliance interface to control appliances during DR events
People Simulation	Utilizes IAppliance interface to control appliances during simulation
Data	Obtains pertinent appliance data
Admin	Monitors appliance simulation

5.5.1.1.6 Control Bundle

The Control Bundle simulates the function of a local Gateway. This entails gathering data from appliances and enacting DR response algorithms during a DR event. For more details about Gateway operation, please see the Task 3 report.

The Control Bundle creates DR events using Java's Date class and monitors the timing of events. The control bundle has four stages of DR events: not in event, transitioning into event, in event, and transitioning out of event.

When in an event, the Control Bundle uses the IAppliance interface to gather total power usage information and controls appliances according to its DR algorithm.

Service consumed from	Details
Startup	Obtains Gateway control algorithm to use
Time Service	Obtains current simulation time for simulated appliance operation
Appliance	Utilizes IAppliance to obtain appliance data and control appliances
Service consumed by	Details
People Simulation	Utilizes IControl to change appliance preference settings and override Gateway control
Data	Obtains pertinent control and Gateway data
Gateway Agent	Utilizes IControl to obtain information necessary for bidding process with other agents
Admin	Monitors functionality of Gateway functions

5.5.1.1.7 People Simulation Bundle

The People Simulation Bundle contains the class Person which is used to create People objects. These are kept in an array list. When initialized each Person object registers a host of appliances that they "own". These appliances are registered in the Appliance Bundle and have the unique person ID as a field. Each Person has a unique ID tag that is contains the zone number and the person number. For example Z5P33 is the 33rd person in the fifth zone.

The behaviors coded into the Person class are time based.

Service consumed from	Details
Startup	Obtains zone and number of people as well as specific people behaviors
Time Service	Obtains current simulation time for simulated person behaviors
Appliance	Utilizes IAppliance to obtain appliance data and control appliances
Control	Utilizes IControl to change appliance preferences and manually override control
Service consumed by	Details
Data	Obtains pertinent person activity data such as usage of appliance and time in office
Admin	Monitors functionality of Person Simulation

5.5.1.1.8 Data Bundle

The Data Bundle collects time series data from other bundles. This data includes appliance power settings and usage, Gateway control data and simulated person time series data. This bundle includes functionality that the Gateway would normally provide but is separate from the Control Bundle for simplicity and to be able to obtain data from the people simulation.

Service consumed from	Details
Startup	Obtains zone number and data file output folder
Time Service	Obtains current simulation time for simulated appliance operation
Appliance	Utilizes IAppliance to obtain appliance data and control appliances
Control	

People Simulation	
Service consumed by	Details
Gateway Agent	Utilizes IData to obtain pertinent data for bidding
Admin	Monitors functionality of Gateway

5.5.1.1.9 JADE Bundle

The JADE Bundle enables the JADE framework to run be started by the OSGi framework. This bundle takes in parameters from the Startup bundle that specify the JADE connection properties.

Service consumed from	Details
Startup	Obtains JADE container properties
Service consumed by	Details
Gateway Agent	Utilizes JADE runtime to create agent

5.5.1.1.10 Gateway Agent Bundle

The Gateway Agent Bundle utilizes the JADE runtime to create an Agent with coded behaviors. This agent utilizes data collected from the simulation for bidding purposes to explore multi-agent systems. Currently each agent searches for an Agent representing the Siemens Smart Energy Box (SEB) and for agents representing other Gateways (other simulation zones). Each agent then requests DR event data from the SEB, and Gateway information from each Gateway Agent.

Service consumed from	Details
Startup	Obtains zone number and agent name
Time Service	Obtains current simulation time
Appliance	Utilizes IAppliance to obtain appliance data
Control	Utilizes IControl to obtain internal Gateway information
People Simulation	Utilizes IPeopleSim to obtain person simulation
Service consumed by	Details
Admin	Monitors functionality of Gateway Agent

5.5.1.1.11 Admin Bundle

The Admin Bundle monitors the actions and events of the other bundles. An important function is to monitor the successful initiation of all other bundles and to signal the Time Service bundle that to start its time stepping.

Service consumed from	Details
Startup	Obtains zone number and agent name
Time Service	Obtains current simulation time
Appliance	Utilizes IAppliance to obtain appliance data
Control	Utilizes IControl to obtain internal Gateway information
People Simulation	Utilizes IPeopleSim to obtain person simulation information

JADE	
Gateway Agent	Monitors bidding process

All bundles and their functions described in this report have been coded, implemented and tested; the demonstration of two gateways is described in section 5.8. However, the simulation is not currently complex enough to model an entire zone or building as realistically as desired. For example the People Simulation has a very simple behavioral model. The simulated people also do not yet utilize the Control features (setting their preferences).

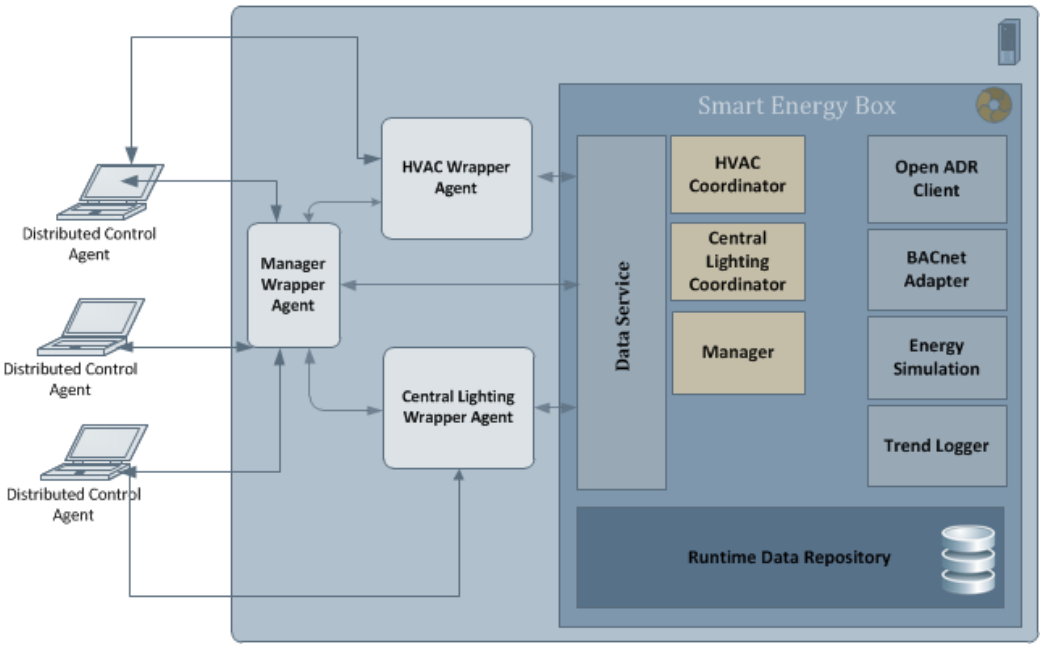
Future efforts include:

- Expanding the person model to include behaviors such as leaving an office temporarily, and printing documents
- Expanding the simulated appliances to include the option of having a backup power supply
- Adding more appliances to the simulation
- Reading from Stream FS to gather realistic zone settings

5.5.2 Siemens' emulator

5.5.2.1 Load Allocation Control

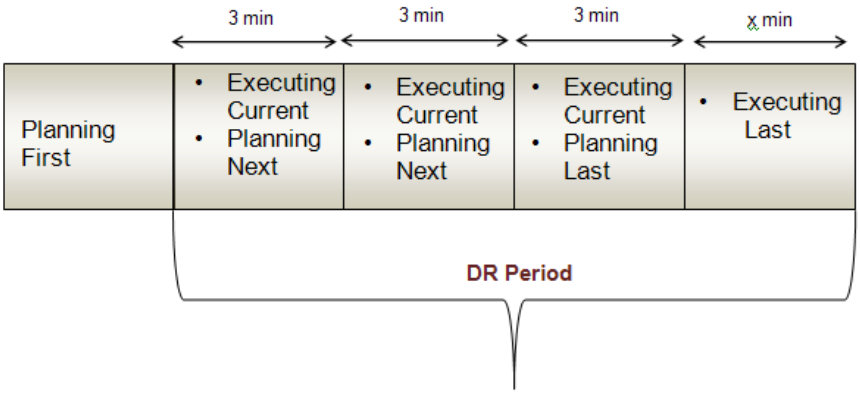
In order to extend SEB capabilities to all office equipment (plug load) in the building, an Agent based mechanism (FIPA compliant) was implemented: JADE (Java Agent Development Environment). This is a FIPA compliant open source agent based platform, which supports a variety of communication protocols, standards to implement effective market based solutions among multiple parties in the building such as HVAC, Lighting, Plug Loads, Occupancy etc. using collective intelligence. A crucial component in this mechanism is the Manager, which uses the collective information from all parties and decides on an optimal energy saving strategy.



Adaptive partial centralized mechanism have been selected to implement the market based solution for local control in response to a demand response event

5.5.2.2 Negotiation cycle

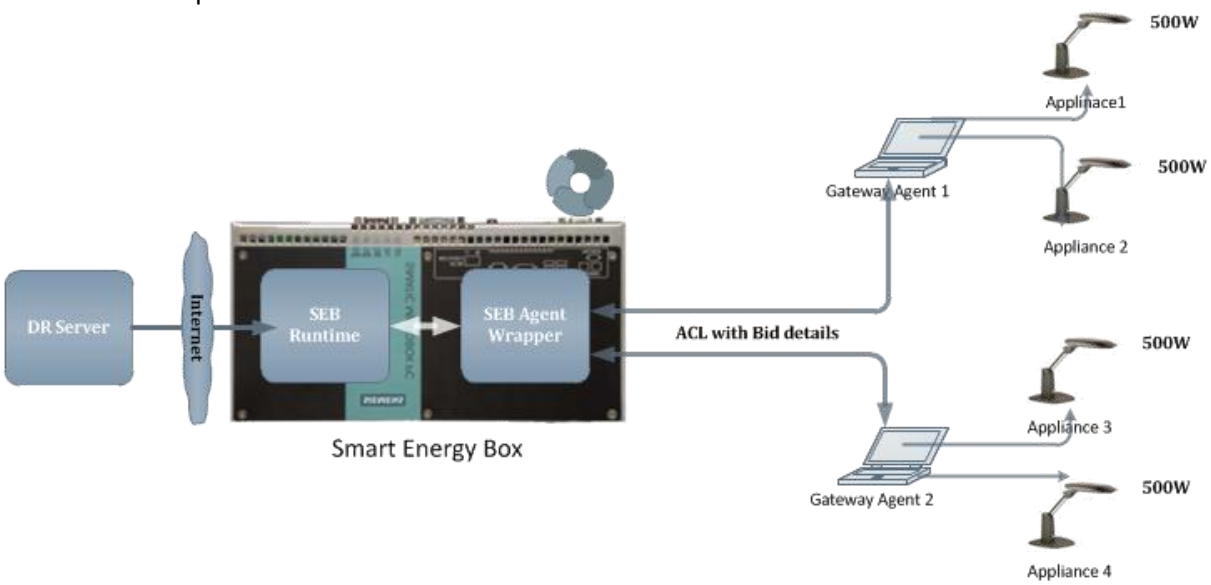
The Manager negotiates with all participants in regular intervals during the DR period based on the load allocation mechanism. While executing the current cycle, the Manager will plan for load shedding in the next cycle. This is a continuous process until the DR period ends. The following figure shows an example of negotiation cycle in 3 min. In a real building it would be 15 min.



5.5.2.3 Demonstration description

The following diagram shows the test setup which has the following major components

- Smart Energy Box runtime system
- Manager Wrapper Agent, a communication agent between JADE based distributed control agent and SEB
- Two Distributed control Gateway Agents each with two Smart Appliance Simulation of 500W each
- Demand response server



5.6 DR algorithm integration

5.6.1 UCB's algorithm

The current control algorithm is a user defined priority scheme for curtailment during demand response events. Appliances connected to a gateway are assigned a numerical priority by the user, between zero and ten. Currently a gateway controls (curtails or turns down or off) appliances with a low assigned priority first and incrementally controls appliances with higher priorities if needed. For example, appliances with priority 0 will be controlled or turned off first, whereupon the gateway will do the same to priority 1 appliances if needed to reach the load reduction goal. This can extend all the way to 10, or a maximum priority limit can be set. Generally lower priority appliances are less important to the user (such as a heater) or have some sort of backup power supply, such as a laptop with its battery. Thus, a task lamp or fan might be turned off first during a demand response event to reduce peak demand, but a computer might have higher priority or be considered non-curtailable. The following diagram outlines the control sequence during a demand response event. At each iteration, the current load is calculated and the decision made to further curtailment or stop.

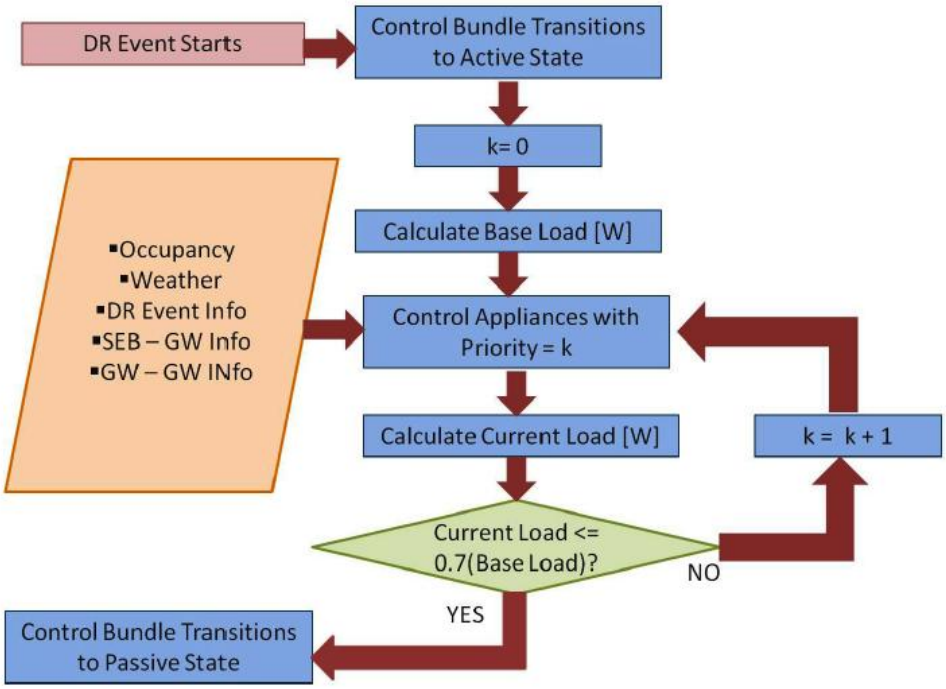


Figure 8: Logic diagram of the control bundle during a demand response event.

5.6.2 SCR's Demand Response algorithm

Similar to UC Berkeley, Siemens Corporate Research considered several plug-in loads: one Smart Appliance Simulation, one task light, one printer and one laptop. Each of the appliances has a priority, and is adjustable during runtime. The default priorities are in Table 1 below:

Table 1: Default priorities.

Appliance	Default Priority

Smart Appliance Simulation	1
Task Light (Normal Bulb)	2
Printer	3
Laptop	4

The smaller the number, the lower the priority is. An appliance could also be set to Critical, in which case it cannot be controlled during a DR Event, and the priority is set to 2. The conditions for the appliances to be Critical are in Table 2. Whenever the critical condition is satisfied for the appliance, the priority of corresponding appliance is set to 2, and its status is not changed during the DR event. If the critical condition is not satisfied, the priority of the appliance returns to the default value.

Table 2: Critical Conditions.

Appliance	*Critical Condition*
Smart Appliance Simulation	Opt out from DR Event
Task Light (Normal Bulb)	Occupancy == True, Light < Strong
Printer	User defined
Laptop	Battery < 40%

The cost function of an appliance reflects how the power reduction of this appliance affects human comfort. We are assigning a cost corresponding to each power consumption level of the appliance to represent the estimated comfort loss when the appliance is operating with that level of power consumption. The higher the cost is, the less comfortable a user feels. We also take the priority and critical conditions into account. Therefore, the cost is assigned as follows.

a) If the appliance is in Critical condition, the cost is set to 2.
b) If the appliance is not in critical condition, cost = $\frac{Priority}{GWMaxPriority} \times \frac{Power}{MaxPower}$, where Priority is the default priority for the appliance, the GWMaxPriority is the maximum priority of all appliances connected to the Gateway, and Power and MaxPower are the current and maximum power consumption of the appliance respectively.

The Gateway will compose one cost function for each of the appliances, and the cost functions will be used in the optimization algorithm to calculate the best load allocation for each appliance. The details of the optimization algorithm will be described in Section 5.7.

5.6.3 Communication Scenarios with SEB

There are several methods identified to implement collective intelligence, which requires different information exchanged between the Gateway Agent and the Smart Energy Box:

- Adaptive Centralized Load Allocation

- Adaptive Partial-Centralized Load Allocation
- Non-Cooperative Market-based Load Allocation
- Non-Cooperative with Group Interruption Cooperation Market-based Load Allocation
- Partial-Cooperative Market-based Load Allocation
- Peer-to-Peer Market-based Load Allocation

For this report, we discuss three communication scenarios: Adaptive Centralized Load Allocation, Adaptive Partial Centralized Load Allocation and one interruption scenario.

5.6.3.1 Adaptive Centralized Load Allocation

In this scenario, each Gateway Agent submits a portfolio (Priority, Power, Current Status) and Cost Function of each Appliance to the SEB Agent; the SEB Agent make a decision for each appliance (it calculates load allocation and control command of each Appliance), and sent the decision back to every Agent. The Gateway Agent then applies the control command to the Appliances, and sends a new portfolio of each Appliance to the SEB Agent.

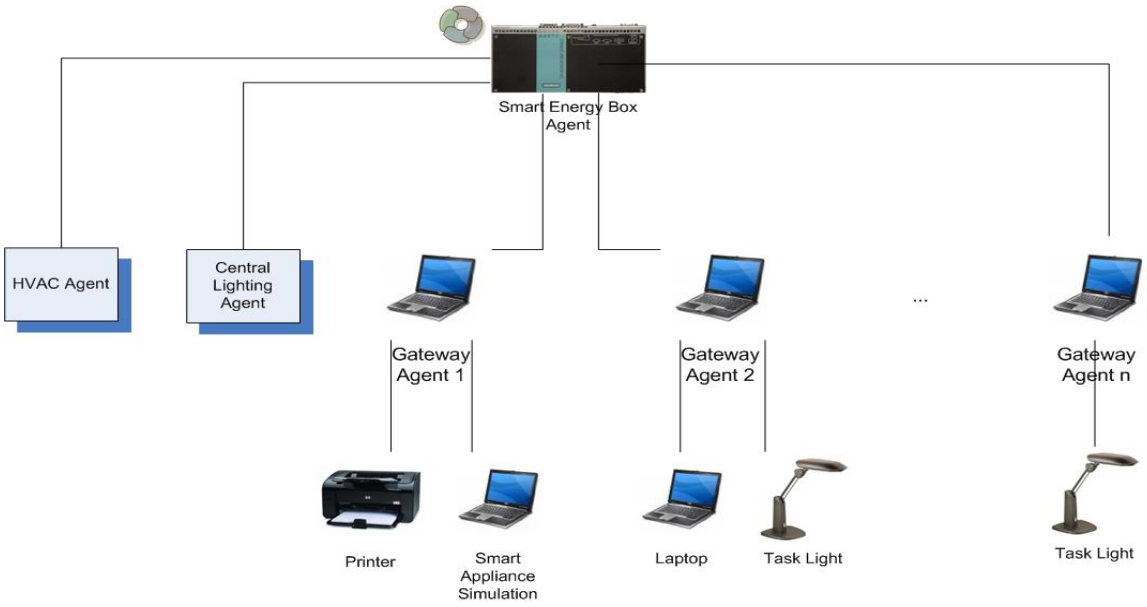


Figure 9: Physical Connection Plan for Adaptive Centralized Load Allocation.

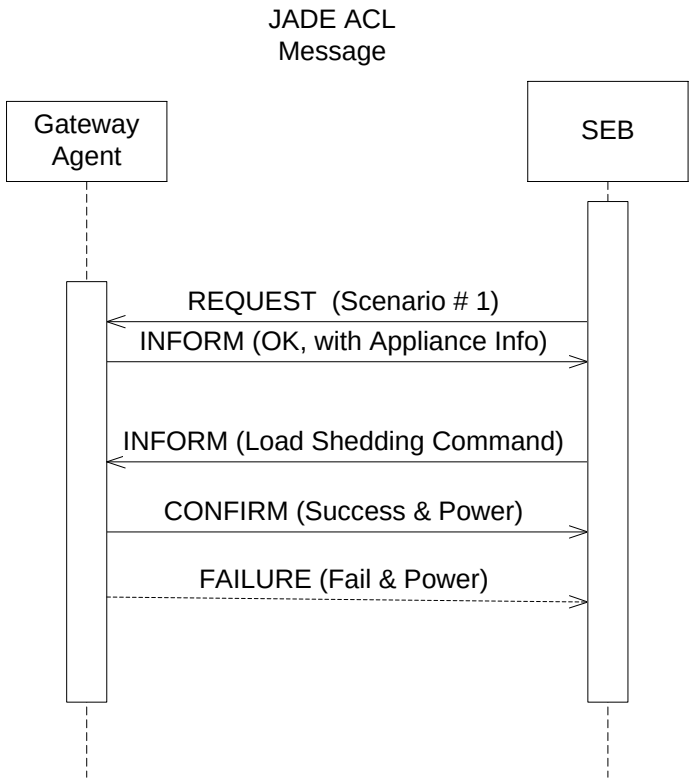


Figure 10: Sequence Diagram for Adaptive Centralized Load Allocation.

5.6.3.2 Adaptive Partial Centralized Load Allocation

In this scenario, each Gateway Agent calculates and sends to SEB Agent the total Cost Function according to the portfolio of each Appliance. SEB Agent calculates and sends the load allocation to each Gateway Agent; the agent calculates the load allocation to each appliance and applies the corresponding control command to the Appliances, and reports total power consumption to the SEB Agent. In this case, the SEB Agent will not make decisions directly for the appliances, but only assign the total load reduction to a Gateway agent. The Gateway agent has the ability to further decide how to allocate loads between appliances.

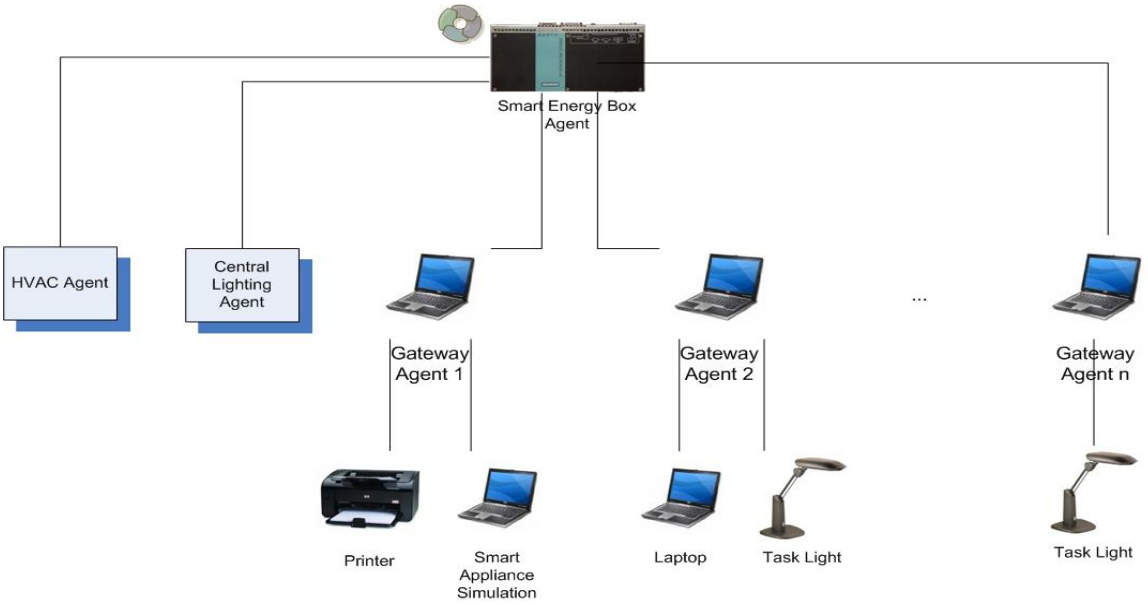


Figure 11: Physical Connection Plan for Adaptive Partial Centralized Load Allocation.

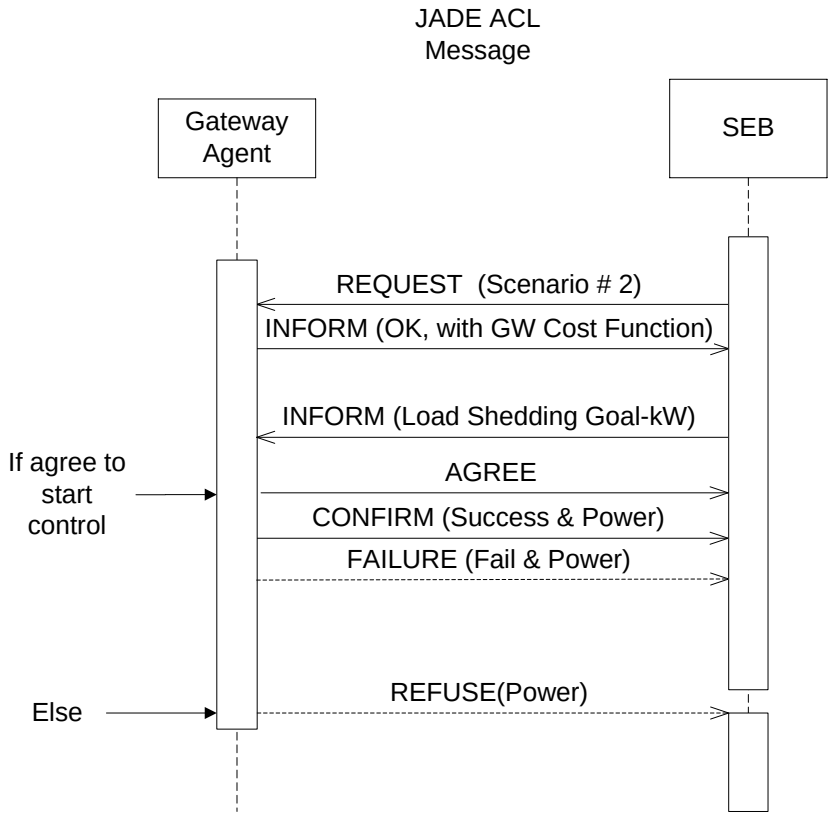


Figure 12: Physical Connection Plan for Adaptive Centralized Load Allocation.

5.6.3.3 Interruption scenario

In this scenario, after the Gateway applies a control command, it can be interrupted by Human control or other factors, and a Gateway can ask for help from another Gateway within a pre-defined group before reporting the interruption to the SEB.

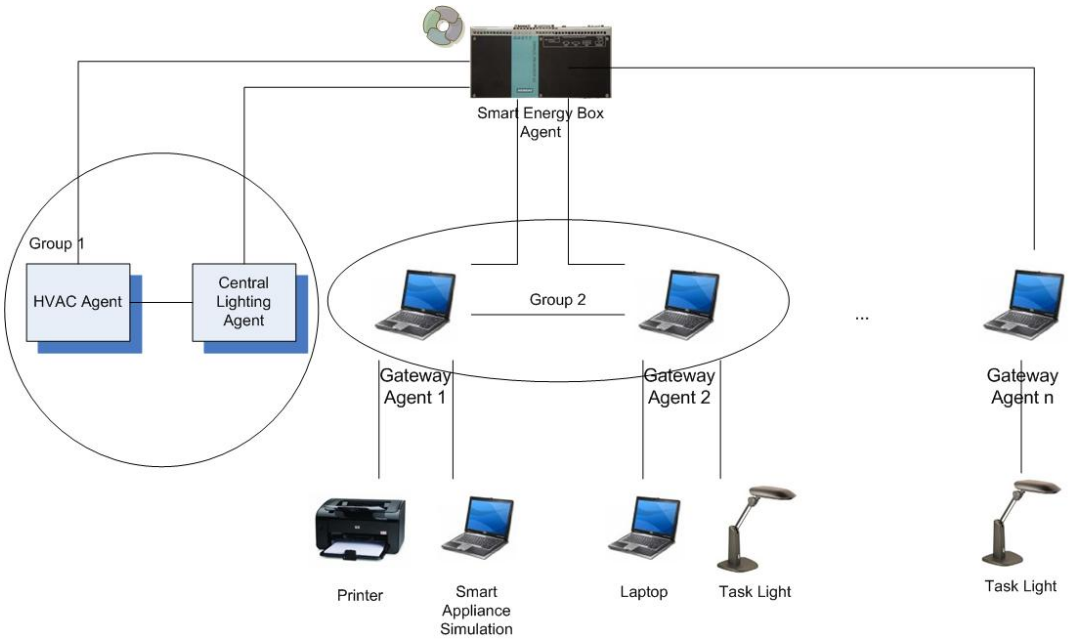
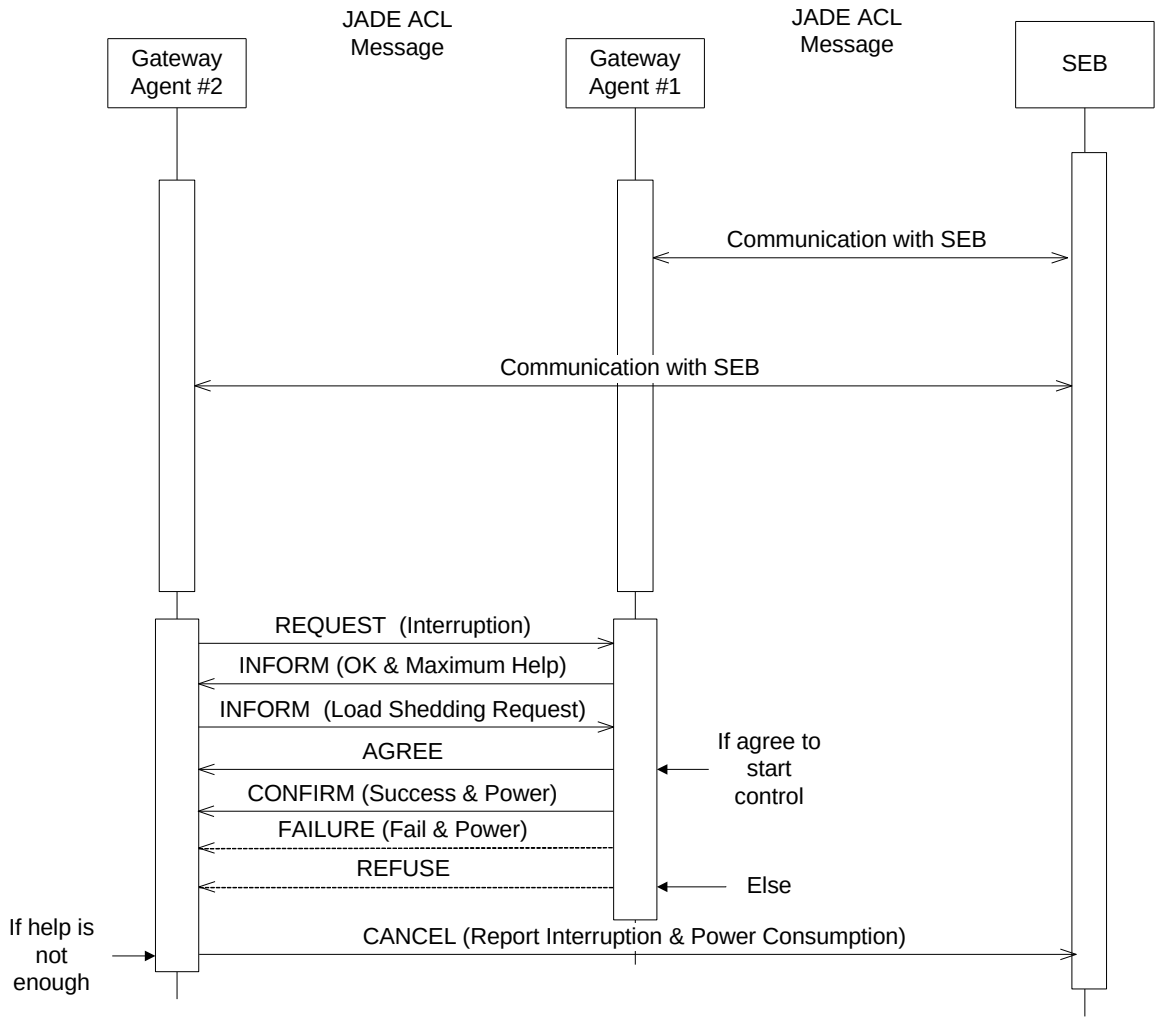


Figure 13: Physical Connection Plan for Interruption Scenario.

Figure 14: Sequence Diagram for Interruption Scenario.



5.6.4 DR algorithm optimization

We formulated the problem into an optimization problem. We focused on Adapting Partially Centralized market approach to negotiate and adapt the DR strategy in response to DR event. Finally we implemented the optimization algorithm into the test bed.

5.6.4.1 Problem formulation

The control layers of the DIADR system is shown in Figure 15.

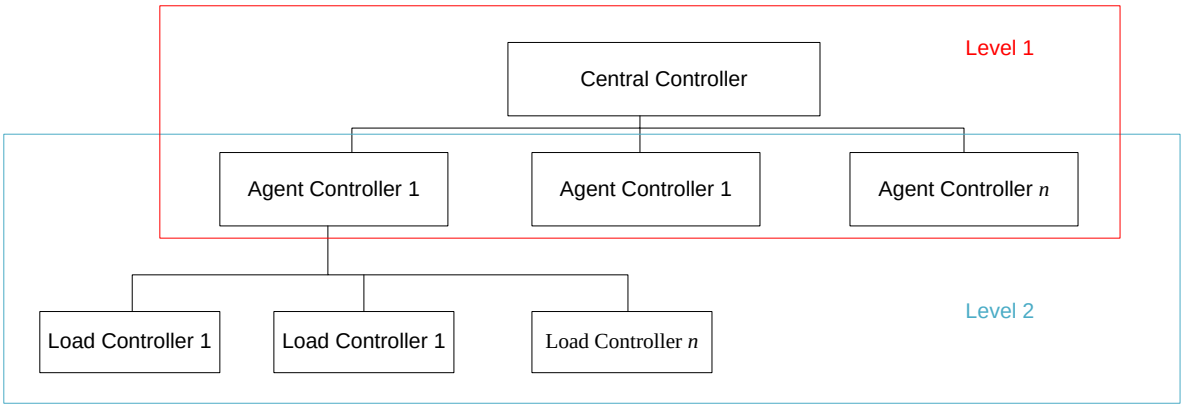


Figure 15: Control layers of DIADR system

The central controller is the central DR manager like the SEB, and the agent controller is the distributed load manager like the Gateway.

All the load controllers can control the loads according to a predefined way according to load types; some can be tuned continuously, while others can only be tuned discretely. The loads can be single load, and aggregation of certain amounts of loads which can include lights, computers, ice machines, water heaters, HVAC systems etc. The objective of DR is to satisfy the load reduction requirement while minimizing the cost of the loads. Actually, this is a problem of mathematic optimization; in order to achieve the goal, developing certain models to signify the cost function of the load is required, and optimality algorithm based upon the cost modeling is also needed.

We consider two approaches for this optimization problem. The first is called the Centralized method, corresponding to the Adaptive Centralized Load Allocation scenario in the Gateway implementation, which is to consider the Agent controller as only communication tool between the central controller and the appliances. The central controller will gather all information of the appliances, do centralized optimization and make decisions for the appliances. The other one is called Two-level optimization, corresponding to the Adaptive Partial Centralized Load Allocation scenario in the Gateway implementation, which is to consider the Agent controller having enough intelligence to perform some optimization as well. Therefore, each Central controller will give optimized decisions to all Agent controllers, and the Agent controllers will decide to optimally dispatch loads across the appliances within its range.

5.6.4.2 Cost function development for the load

In [1] and [2], the cost functions (also as disutility function) are assumed to be continuous, differentiable, and convex. In [1] the authors assumed that each customer has a cost of quadratic function as $u = aq + bq^2$, where u denotes disutility of the customer, and q denotes the shed load. In [2], the authors used exponential function $u = c|T - T_{des}|^{\rho}$ to denote the disutility of energy reduction for temperature control, where T_{des} and T the temperature of the house and the baseline of temperature, and they are all proportional to energy.

The continuous models are suitable for the scenarios where loads can be controlled continuously. And if lots of loads are aggregated together by aggregators, the continuous models can also be utilized as lots of discrete controlled load can be approximated to be continuous. However, if we only consider one or few appliances, the discrete models are preferred.

In this report, we developed three scenarios for testing which contains four types corresponding to four different appliances.

1) Scenario 1
   - Smart Appliance

   Smart appliance has three levels, OFF, Medium and ON, and the cost function constants are as follows:
   (0, 0), (0.3, 0.075) and (0.25, 1)
   - Task Light

   The light is not dimmable light, can only be turned ON or OFF, so it has only two levels and its cost function constants are as follows:
   (0, 0) and (1, 0.5) when daylight is sufficient, and (0, 0), (2, 1) while daylight level is low.
   - Laptop

   Laptop is also modeled as an ON and OFF type load. The cost function constants are as follows:
   (0, 0) and (0.75, 1) when battery charge is higher than twenty percentage of the total capacity of the battery and (0, 0), (2, 1) when the battery charge is lower than twenty percentage of the total capacity of the battery.
   - Printer

   Printer is modeled as an ON and OFF load, and cost function constants are as follows:
   (0, 0), (1, 1)

2) Scenario 2
   - Smart Appliance

   Smart appliance has three levels, OFF, Medium and ON, and the cost function parameters are as follows:
   (0, 0), (0.075, 0.3) and (0.25, 1)
   - Task Light

   The light is not a dimmable light, and in this scenario, we have two different lights with load ratio of 13:60. The cost function sequences are as follows:
   The light is not a dimmable light, can only be turn ON or OFF, so it has only two levels and its cost function parameters are as follows:
   (0, 0) and (0.17, 0.18), (0.17, 0.82) and (0.5, 1) when daylight is sufficient, and (0, 0), (0.17, 0.18), (0.17, 0.82) and (2, 1) while daylight level is low.
   - Laptop

   Laptop is also modeled as an ON and OFF type load. The cost function parameters are as follows:

(0, 0) and (0.75, 1) when the battery charge is higher than twenty percentage of the total capacity of the battery and (0, 0), (2, 1) when the battery charge is lower than twenty percentage of the total capacity of the battery.

- Printer

Printer is modeled as an ON and OFF load, and cost function parameters are as follows:
(0, 0), (1, 1)

3) Scenario 3
   - Smart Appliance

   Smart appliance has continuous cost function which is linear between (0, 0) and (0.25, 1).
   - Task Light

   The light is not a dimmable light, and in this scenario, we have two different lights with load ratio of 13:60. The cost function parameters are as follows:
   The light is not a dimmable light, can only be turned ON or OFF, so it has only two levels and its cost function parameters are as follows:
   (0, 0) and (0.17, 0.18), (0.17, 0.82) and (0.5, 1) when daylight is enough, and (0, 0), (0.17, 0.18), (0.17, 0.82) and (2, 1) while daylight level is low.
   - Laptop

   Laptop is also modeled as an ON and OFF type load. The cost function parameters are as follows:
   (0, 0) and (0.75, 1) when the battery charge is higher than twenty percentage of the total capacity of the battery and (0, 0), (2, 1) when the battery charge is lower than twenty percentage of the total capacity of the battery.
   - Printer

   Printer is modeled as an ON and OFF load, and cost function parameters are as follows:
   (0, 0), (1, 1)

These three scenarios will be tested using the proposed algorithm in the test result section.

5.7 Testing and optimization

5.7.1 Optimization algorithm

There are two methods to implement the DR optimization. One way is the centralized method: each load controller sends its cost function to the central controller, and the controller utilizes the optimization algorithm to obtain the optimal allocation for each load, and in this the agent controller is used as a communication junction only, say a gateway. The other way is a distributed method: the central controller assigns the load reduction task to each agent controller using the optimization method, then the agent controller obtains the optimal load reduction for each using the optimization algorithm.

First, we will discuss the centralized optimization, which will help derive the method used for distributed control algorithm.

5.7.1.1 Centralized method with continuous cost function

Consider a demand response control problem with participants of *n*. The original load is R_0, the objective of load shedding is R_p, and, the reduction load is represented by *R*, which satisfies following equation.

$$R=R_0-R_p \tag{1}$$

Use r_i (i=1,2⋯*n*) to denote the allocation of participant *i*, and $f_i(r_i)$ to represent the reduction cost function, so the problem can be described as a task allocation problem, which can be expressed as follows [3]:

$$\begin{cases} \min\sum_{i=1}^{n} f_i(r_i) \\ st.\ \sum_{i=1}^{n} r_i = R \quad r_i^L \le r_i \le r_i^u \end{cases} \tag{2}$$

Suppose that the cost function $f_i(r_i)$ are convex and differentiable and is continuous in a surrounding of allocation, r_i, so, Lagrangian function can be written as follows:

$$\text{Min}\ \ \sum_{i=1}^{n} f_i(r_i) - \lambda\left(\sum_{i=1}^{n} r_i - R\right) \tag{3}$$

The optimal allocation satisfies that

$$\begin{cases} \frac{\partial f_i(r_i)}{\partial r_i} - \lambda = 0 \\ \sum_{i=1}^{n} r_i - R = 0 \qquad r_i^L \le r_i \le r_i^u \end{cases} \tag{4}$$

1. Unconstrained optimization

According to (4), if there exists a r_i in range [r_i^L r_i^u] which satisfies the first equation of Eq. (4) for each $f_i(r_i)$, then we define this kind of allocation as a unconstrained allocation. We can solve the equation (4) without considering the inequality constraints ($r_i^L \le r_i \le r_i^L$). Obviously, if an optimal allocation is achieved, the derivatives of $f_i(r_i)$ for all the participants equal to λ, as $f_i(r_i)$ is defined as cost function, thus, we define the parameter λ as the clearing marginal cost. The discovery process of clearing margin cost can be defined as follows:

1) Each load controller sends its cost function to the central controller.
2) The central controller initiates a λ_0.
3) The central controller calculates the load reduction r_i of each load according to the first equation of Eq. (4).

Usually, Eq. (4) is a nonlinear equation, thus we can use Newton-Raphson method to solve it.

The formulas are shown in Eq. (5)

$$\begin{cases} \Delta r_i^k = -\frac{\partial f_i(r_i)}{\partial r_i} \Big/ \frac{\partial^2 f_i(r_i)}{\partial^2 (r_i)} \\ r_i^{k+1} = \Delta r_i^k + r_i^k \end{cases} \tag{5}$$

4) The central controller calculates the sum of load reduction of all participants and compares it to R.

If $|\sum_{i=1}^{n} r_i - R| \leq \varepsilon$, If the $|\sum_{i=1}^{n} r_i - R| < \varepsilon$, the central controller marks this clearing cost λ as the final clearing margin cost, and records the load reduction of all the loads, then go to step 5.

Else, if $\sum_{i=1}^{n} r_i - R > \varepsilon$, **decreases** the clearing cost λ, whereas if $\sum_{i=1}^{n} r_i - R < -\varepsilon$ **increase** it, where ε is the arbitrary constant as for error, then go to step 3.

The clearing cost λ is updated according to the following expression:

$$\lambda^{k+1} = \lambda^k + r^k(R - \sum_{i=1}^{n} r_i) \tag{6}$$

Where $r^k > 0$ is the iteration step-size at step *k*.

5) The central controller sends the optimal load reduction to all the loads, once the load controller receives the command, it sheds load.

6) Constrained optimization

If there does not exist such a r_i which **satisfies** the first equation of Eq. (4) for one of the $f_i(r_i)$ or some of the $f_i(r_i)$, then we define this kind of optimization as a constrained optimization.

The process of strained optimization is much the same as unstrained optimization process, however, the step (3) is replaced by following step.

$$\frac{\partial f_i(r_i)}{\partial r_i} = \begin{cases} \leq \lambda_i & r_i = r_i^u \\ = \lambda_i, & r_i^l \leq r_i \leq r_i^u \\ \geq \lambda_i & r_i = r_i^l \end{cases}$$

Actually, the unconstrained optimization is the special case of constrained optimization.

The reduction cost of each bidder is determined by disutility function. The higher the load reduction, the higher value the disutility is. For many applications, it is very normal to have a concave function **[4, 5]** to denote disutility or utility. In this paper, we use the polynomial expression to evaluate cost function, as shown in Eq. (7) as used in [1].

$$u(r) = a_0 + a_1 r + a_2 r^2 \tag{7}$$

Where, *r* is the reduction of load, and the coefficients a_0, a_1, a_2 vary according to different occupancies. For each load, *r* has a lower limit and upper limit.

In the test section, an example (test 1) is given to illustrate the algorithm presented.

5.7.1.2 Centralized method with discrete cost function

The method described above is deterministic and it requires continuous and differentiable cost function for each participant, however, it is not practical to obtain such kind of function for every participant. This section we will discuss another optimization method for handling this issue.

For the cost function, $f_i(r_i)$, we assume that it is discrete and follows a step change, as showing in Fig.10.

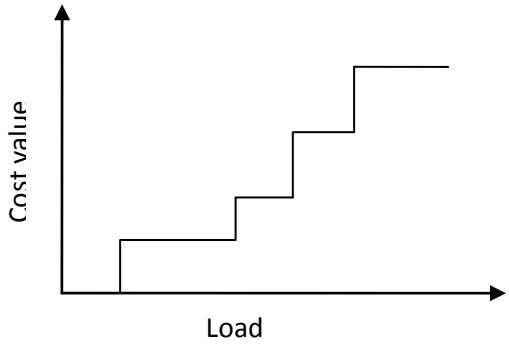


Figure 16 Discrete cost function of the load.

We use the particle swarm optimization method to solve the load shedding problem described in Exp. (4) of the previous section. The constraints include both equalities and inequalities. Use $x = [r_1, r_2, \cdots r_n]$ to represent the allocated load of the participants.

The inequalities constraints can be written into following expression, where *J* is the number of inequalities.

$$g_i(x) \leq 0 \quad i = 1,2,3, \ldots . J \tag{8}$$

Since an inequality constraint of the form, $g_i(x) \geq 0$, can also be represented as $-g_i(x) \leq 0$, so the Exp. (8) denotes general form of inequalities constraints.

Rewrite the equalities constrains as Eq. (9), where *K* is the number of equalities constraints.

$$h_j(x) = 0 \quad j = 1,2,3 \ldots . . K \tag{9}$$

In order to implement PSO, Eq. (9) is transfered into the following inequality expression, where ε is a infinitesimal arbitrary constant.

$$|h_j(x)| \leq \varepsilon \tag{10}$$

Furthermore, it can be represented by the inequalities (11) and (12).

$$-h_j(x) - \varepsilon \leq 0 \tag{11}$$

$$h_j(x) - \varepsilon \leq 0 \tag{12}$$

Thus, we convert the equalities constraints into inequalities constraints, which follow the formation of Exp. (8). So the constraints, in general can written as Exp. (13.)

$$g_i(x) \leq 0 \quad i = 1,2,3, \ldots . J + K \tag{13}$$

The penalty function is introduced to handing the constraints, and which is defined as follows:

$$P(x) = y(k)H(x) \tag{14}$$

k is the algorithm's current iteration number and $y(k)$ is dynamically modified penalty value, usually, following formulas are used to determine this value.

$$y(k) = \sqrt{k} \text{ or } y(k) = k\sqrt{k} \tag{15}$$

$H(x)$ is the penalty factor, which is determined by Eq. (13)

$$H(x) = \sum_i^{J+K} \theta\big(q_i(x)\big)\, q_i(x)^{\gamma(q_i(x))} \tag{16}$$

Where, $q_i(x) = \max\{0, g_i(x)\}, i = 1,2, \cdots J + K$ is a relative violated function of constraints, $\theta\big(q_i(x)\big)$ is a multi-stage assignment function **[6]**, $\gamma(q_i(x))$ is the power of the penalty function, $g_i(x)$ is described in Exp. (10).

In this report, $\theta\big(q_i(x)\big)$ is defined as follows:

$$\theta\big(q_i(x)\big) = \begin{cases} 1 & q_i(x) < 0.1 \\ 5 & q_i(x) < 1 \\ 10 & q_i(x) < 10 \\ 100 & q_i(x) < 20 \\ 200 & otherwise \end{cases} \tag{17}$$

$r(q_i(x))$ is defined as Exp. (18).

$$\gamma\big(q_i(x)\big) = \begin{cases} 1 & q_i(x) < 1 \\ 2 & otherwise \end{cases} \tag{18}$$

The objective of PSO is defined according to (19), where $f(x) = \sum_{i=1}^{n} f_i\,(r_i)$, and the value of which can be obtain using the cost function depicted in Fig. 1.

$$F(x) = f(x) + P(x) \tag{19}$$

Assume that we use NP particles to implement the optimization, and for the *n* participants load shedding, the search space for each particle is *n*-dimensional, which can be represented by the vector $x_i = (r_{i1}, r_{i2} .., r_{in})$.The best particle of the swarm is denoted by index $p_g = (p_{g1}, p_{g2} \cdots p_{gn})$, and the best previous position of the *i*-th particle represented as $p_i = (p_{i1}, p_{i2} \cdots p_{in})$ and the position change (velocity) of the *i*-th particle is $v_i = (v_{i1}, v_{i2}, \ldots v_{in})$. The process of iteration of PSO follows equations **[7]**:

$$v_i^{k+1} = \lambda\left(w v_i^k + c_1 r_{i1}^k \big(p_i^k - x_i^k\big)\right) + c_2 r_{i2}^k \big(p_g^k - x_i^k\big) \tag{20}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{21}$$

Where, $i = 1,2 \cdots NP$, is the size of the population of particles, λ is a constriction factor which is used to control velocities, *w* is the inertia weight, c_1 and c_2 are two positive constants, namely the cognitive and social parameter respectively, r_{i1} and r_{i2} are random numbers uniformly distributed within the arrange

[0 1]. The inertia weight *w* is employed to control the impact of the previous history of velocities on the current velocity, in this report we change the value of *w* dynamically according to (22).

$$w^k = w_{max} - \frac{w_{max} - w_{min}}{N_{max}} \times k \tag{22}$$

Where, w_{max} and w_{min} are the maximum weight and the minimum weight respectively, in this paper we use w_{max}=0.9 and $w_{max} = 0.2$, N_{max} is the maximum iteration number, *k* is the current iteration number.

In the test section, an example (test 2) is presented to illustrate the PSO optimization for optimal load reduction.

5.7.1.3 Two-level optimization with continuous cost function

In this section, we discuss the two-level optimization with continuous cost function based upon the same optimization used in the centralized optimization.

Assume that we have Na agents, and we use $\mathrm{r_i}$ to denote the load of agent *i* (*i*=1, 2, ⋯ Na), and L_{ij} to represent load of agent *i*, where *j*=1, 2, ⋯N_{ai}, N_{ai} is the number of load of agent *i.* r_i represents the load reduction of agent *i*, and it can be expressed as Eq. (23)

$$r_i = \sum_{j=1}^{N_{ai}} L_{ij} \tag{23}$$

The polynomial function is used to evaluate the cost function of each load, as shown in Eq. (24)

$$u(L_{ij}) = a_0^{ij} + a_1^{ij} L_{ij} + a_2^{ij} L_{ij}^2 \tag{24}$$

Two approaches to obtain the allocation of each load are feasible. The first one is using the cost function of each directly, and each load sends its cost function to control center, then the control center uses the optimization methods discussed previous to determine the optimal allocation. The second one is the control center first to determine the optimal allocation for each agent, and then the agents determine the optimal allocation for each load. In this scenario, the loads do not need to send their cost functions to the control center, and they send them to the corresponding agents instead. Each agent is supposed to send its cost function to control center on the behalf of obtaining optimal allocation.

As for the first approach, we can use the optimization method discussed in previous section, which can assure the optimal solution. The second approach is actually the two-layer optimization problem, the first layer is between the control center and the agents, the second layer is between the load sets and agents corresponded. If the $\mathrm{r_i}$ is allocated to the agent *i* through level 1, for the level 2, the optimal load reduction can be obtained by utilizing the cost functions of load sets using the optimization method.

So the key issue is to determine r_i through the level 1 to minimize the total cost while the cost function of each agent $f_i(r_i)$ is unknown. As discussed above, for the specified allocation of an agent, the $f_i(r_i)$ is the optimization result of the load sets, so as the quantity of allocation vary, the cost value of which

$f_i(r_i)$ should change tracing along the minimum cost value trajectory. So, we can obtain the $f_i(r_i)$ through a numerical way:

1. Initialize r_i with the minimum load reduction r_i^0 which can be obtained using Eq. (23) while each load L_{ij} set to be the minimum value, and set the iteration number *k*=0.
2. Solve the optimization problem described in Exp. (25), we gain the one optimal cost value $f_i(r_i^k)$ with the load reduction of r_i^k.

$$\begin{cases} \text{Obj.} \quad \min\left(\sum_{j=1}^{N_{ai}} u(L_{ij})\right) \\ Cons. \quad r_i^k = \sum_{j=1}^{N_{ai}} L_{ij} \end{cases} \tag{25}$$

3. Increase r_i utilizing the Eq. (26) until it reaches the maximum value which can be also calculated using Eq. (23) where each load are set to be the maximum value.

$$r_i^k = r_i^k + \Delta r_i \tag{26}$$

4. Set *k*=*k*+1, then continue to Step 2 to calculate $f_i(r_i^k)$, until the maximum r_i^k is achieved.

Through the abovementioned process, we actually obtain serials of values of $f_i(r_i)$ corresponding to specified load reduction allocation of r_i. As the cost function of each load is polynomial function, the total cost function of an agent is also considered to be polynomial function. We use the polynomial fitting method to approximate the analytical expression of $f_i(r_i)$, this method can be found in **[8]**, thus we can write the $f_i(r_i)$ into Eq. (27)

$$f_i(r_i) = a_0^{r_i} + a_1^{r_i} r_i + a_2^{r_i} r_i^2 \tag{27}$$

Since we obtain the cost function expressions of all the agents, we can use the optimization method to obtain the optimal allocation for the level 1.

In practice, the progress is described as follows:

1) All the load sets send their cost functions to the corresponding agents.
2) The Agents calculate their cost functions by solving problem described in Exp.(4) and then obtain the expressions like Eq. (27)
3) The agents send their cost functions to the control center
4) The control center again determines the load reduction allocation by solving the problem in Exp. (4) and sends the optimal results to each agent.
5) Since each agent has received the allocation, it determines the optimal load reduction of the load sets and sends the results to them.
6) After the each load receives the load reduction command, it will do the load shedding.

In the test section, a test (test 3) is presented to show the performance of this method.

5.7.1.4 Two-level optimization with discrete cost function

When the cost function of the load is discrete, it cannot be expressed as the Eq. **(25)**, instead, the cost function is represented by sequences as

$$(L_{ij}^1, U_{ij}^1), (L_{ij}^2, U_{ij}^2)\ldots(L_{ij}^k, U_{ij}^k) \quad (28)$$

Where *k* is the number of the points the discrete function holds. And the load reduction of the agent also satisfies the equation (23).

As the cost functions of the agents are discrete, as a result, the cost function we obtained cannot be continuous, thus, we also use a serials of points to denote that. First we should decide how many points we need. If agent *i* have N_{ai} loads, and each load holds a discrete cost function of p_{ij}, the possible maximum combination of the loads is $\prod_{j=1}^{N_{ai}} p_{ij}$,however, we may not obtain this maximum values, as some combinations my result the some load reduction. Another issue posed is that we may not get exactly sequences of the agent can obtain, unless we enumerate all the possible combinations. So, in this report, we predefine the number of points for the agent, say *m*, then, simulate the sequences using Eq. (26), where Δr_i is calculated using Eq. (29).

$$\Delta r_i = (r_i^u - r_i^L)/m \quad (29)$$

As a result, the progress of generating the cost function of the agent is described as follows:

1) All the loads sets send their discrete cost functions (points) to the corresponding agents.
2) Initialize r_i with the minimum load reduction r_i^0 which can be obtained using Eq. (23) while each load L_{ij} set to be the minimum value, and set the iteration number *k*=0.
3) Solve the optimization problem described in Exp. (25), where the cost function $u(L_{ij})$ is denoted by the sequences of Exp. (28). We gain the one optimal cost value $f_i(r_i^k)$ with the load reduction of r_i^k by utilizing the PSO optimization described in previous section which results a cost function point of the agent (r_i^k, $f_i(r_i^k)$)
4) Increase r_i by utilizing the Eq. (26) until it reaches the maximum value (*k* equals to *m*) where Δr_i is defined in Eq. (29).

Through above procedure, we gain the discrete cost function sequences of the agent as Exp. (30), which can be utilized for the level 1 optimization.

5) $(r_i^1, f_i(r_i^1)), (r_i^2, f_i(r_i^2))\cdots(r_i^m, f_i(r_i^m))$ (30)

The progress to implement this method is the same as the progress described in previous section.

In the test section, a test (test 4) is given to show the performance of this optimization policy.

5.7.2 Discrete cost function models test results

In this section, we present the test results using the discrete cost function models which are proposed in cost function development section. We tested scenario 1 and scenario 3, and in order to define the importance of different appliances, we assign the smart appliance with priority of 1, task light of 2, laptop of 3 and printer of 4. We only presented the results with high light and high battery and results are shown in Table 3 and Table 4.

Because PSO is a randomized algorithm, the result can be different when it runs several times. In the test, the same optimization algorithm ran 5 times for each of the Scenario, and the best was chosen (grey line) as the best result.

Table 3. Two level optimization results

		Agent 1				Agent 2				Agent 3				Total	Cost
Scenario1		Load Shed=900W, Range[900 1000]													
		500	0	0	0	500	0	0	0	0	0	0	400	1000	0.500
		0	0	0	0	500	0	0	400	0	0	0	0	900	4.250
		0	0	0	0	500	0	0	400	0	0	0	0	900	4.250
	H	0	0	0	0	500	0	0	0	500	0	0	0	1000	0.500
	L	500	0	0	400	0	0	0	0	0	0	0	0	900	4.250
	H	Load Shed=800W, Range[800 900]													
	B	500	0	0	0	500	0	0	0	0	0	0	0	1000	0.500
		500	0	0	0	0	0	0	400	0	0	0	0	900	4.250
		150	0	0	0	150	0	0	0	150	0	0	400	850	4.225
		500	0	0	0	0	0	0	0	500	0	0	0	1000	0.500
		150	0	0	0	500	0	0	0	0	0	0	400	1050	4.325
Scenario3		Load Shed=900W, Range[900 1000]												Total	Cost
		450	0	0	0	470	0	0	0	0	0	0	0	920	0.460
		450	0	0	0	0	0	0	0	410	60	0	0	920	0.763
		490	0	0	0	0	0	0	0	430	0	0	0	920	0.460
	H	0	0	0	0	410	60	0	0	450	0	0	0	920	0.763
	L	490	0	0	0	430	0	0	0	0	0	0	0	920	0.460
	H	Load Shed=800W, Range[800 900]												Total	Cost
	B	390	0	0	0	370	60	0	0	0	0	0	0	820	0.713
		430	0	0	0	0	0	0	0	390	0	0	0	820	0.410
		0	0	0	0	430	0	0	0	390	0	0	0	820	0.410
		410	60	0	0	0	0	0	0	290	60	0	0	820	1.017
		0	0	0	0	430	0	0	0	330	60	0	0	820	0.731

In Table 3, each column under each Agent is corresponding to one appliance. The number displaying is the power to shed for this appliance. The “Total” column gives the overall load shedding for all agents and the “Cost” is the overall cost of all agents. The “Load Shed” means the total load shedding goal for all agents, and if the exact load shedding cannot be reached because of the discrete nature of the appliance, the next level within the “Range” will be selected.

When we generate the cost function of the agents for the level 1 optimization, we assume that the load reduction of the whole agent can be changed in a small step size, then we use these cost function curves to obtain the load reduction of each agents. Actually, most the load value of cost function of agent cannot exist when the cost function curves of load are discrete. In the scenario 1 when load shed=800 W, we get the load reduction of 121W for agent 1, 486 W for agent 2 and 202 W for agent 3, in the level 2 optimization, the loads try to find the optimal combination which close to the value, one of the results can be 150W, 500W and 400W, then the total load will be 1050W, if we consider a window of 100W ([800, 900]), this is not a reasonable solution. This is caused by the discretion of load cost function curves, if the curves are continuous or less discrete, the results can be improved, for example, the scenario 3 can obtain better results.

Table 4. Centralized optimization results

		Agent 1				Agent 2				Agent 3				Total	Cost
Scenario1		Load Shed=900W, Range[900 1000]													
		0	0	0	0	500	0	0	0	500	0	0	0	1000	0.500
		500	0	0	0	500	0	0	0	0	0	0	0	1000	0.500
		500	0	0	0	500	0	0	0	0	0	0	0	1000	0.500
	H	0	0	0	0	400	73	0	0	500	0	0	0	900	4.250
	L	500	0	0	0	0	0	0	0	500	0	0	0	1000	0.500
	H	Load Shed=800W, Range[800 900]													
	B	500	0	0	0	150	0	0	0	150	0	0	0	800	0.400
		500	0	0	0	150	0	0	0	150	0	0	0	800	0.400
		150	0	0	0	500	0	0	0	150	0	0	0	800	0.400
		500	0	0	0	150	0	0	0	150	0	0	0	800	0.400
		500	0	0	0	150	0	0	0	150	0	0	0	800	0.400
														Total	Cost
Scenario3	H	Load Shed=900W, Range[900 1000]													
	L	500	60	0	0	250	0	0	0	40	60	0	0	910	1.062
	H	290	60	0	0	500	60	0	0	0	0	0	0	910	1.062
	B	390	60	0	0	200	0	0	0	200	60	0	0	910	1.062
		450	0	0	0	250	60	0	0	90	60	0	0	910	1.062
		350	0	0	0	220	0	0	0	280	60	0	0	910	0.758
		Load Shed=800W, Range[800 900]													
		390	0	0	0	110	0	0	0	250	60	0	0	810	0.708
		390	0	0	0	150	0	0	0	210	60	0	0	810	0.708
		220	60	0	0	230	0	0	0	240	60	0	0	810	1.012
		500	0	0	0	190	0	0	0	110	0	0	0	800	0.400
		390	0	0	0	110	0	0	0	250	60	0	0	810	0.708

Comparing Table 3 and Table 4, we can see that the centralized algorithm is usually better than the two-level optimization in terms of getting a total load shedding closer to the Load Shedding goal, or while the total load shedding is the same, the cost is smaller.

5.8 Room Demonstration

5.8.1 UC Berkeley demonstration

This section covers the implementation and testing of multiple Gateways and a Smart Energy Box (SEB) proxy in the UC Berkeley test lab of 464 Sutardja Dai Hall. The Gateway(s) used in for this test remain largely unchanged from those used in the April 27 demonstration and also described in the Task 3 report. The most discernible change is the inclusion of OSGi bundles for the JADE runtime and Gateway Agent.

The test laboratory of 464 SDH contains three desktops, named Lemon, Lime and Grapefruit. For testing purposes the simulated SEB was run on Lime. Lemon and Grapefruit each had a Gateway running on each representing a Gateway zone. Additionally, a simulated smart appliance, in this case a laptop, was run on both Lemon and Grapefruit and communicated with its respective Gateway. These smart appliances utilize TCP/IP sockets to communicate with the Gateway and utilize an XML schema to transmit data and commands. The laptop simulations simulate a laptop drawing power to charge its battery and function, including battery dynamics. This simulation is also described in detail in the Task 3 report.

Both Lemon and Grapefruit have a Raritan DPX-8 adjacent to them that served as the method of control for legacy appliances for each Gateway. The appliances connected to each DPX-8 are listed below:

Outlet Number	Grapefruit DPX-8	Lemon DPX-8
1	Fan	Uninterruptible Power Supply
2	Desk lamp	Desk lamp
3	Empty	Desk lamp
4	Tablet PC charger	Laser Printer
5	Laptop charger	Empty
6	Empty	Empty
7	Empty	Empty
8	Uninterruptible Power Supply	Empty

The desktop named Lime was where the simulated SEB was run. For this test, the primary function of the SEB was to disseminate DR event information to Gateways it is in communication with. The SEB also serves to assign a maximum control priority to the Gateways when prompted. The simulated SEB was contained in a JADE agent, called the SEB Agent. This allows the SEB Agent (therefore the SEB) to communicate with other JADE agents, such as those of the Gateways.

When a SEB Agent is initiated three JADE behaviors are also started. The first is a timed behavior that is called *GatewayAgentRequest*, which searches the JADE runtime for Gateway agents, and when found gathers pertinent information such as the address of the Agent. The "timed" refers to this behavior being executed on a clock basis with a set and fixed delay between each execution. The second behavior initiated is called *ReturnDREventData*. The function of this is to receive requests from Gateway Agents on DR event information and return the information. This is executed on a cyclic behavior meaning that it passively listens for a request and only acts when the request is received then returns to the passive state. The third behavior is a cyclic one called *RetrunControlPriority*. This behavior's purpose is to passively listen for a request from a Gateway Agent for the control priority. When the request is received, the SEB calculates the priority and returns it to the Gateway Agent.

The Gateway Agents are also JADE agents that are coded to perform behaviors before and after a DR event has started. When initiated, the Gateway Agent starts two timed behaviors, called *SmartEnergyBoxAgentRequest* and *GatewayAgentRequest*, which search the JADE runtime for Smart Energy Box Agents and other Gateway Agents, respectively. Once a SEB Agent is found, a third behavior called *DREventDataRequest* is initiated. This is a timed behavior that contacts the SEB Agent for DR Event information. Once received, the Gateway Agent creates an event in the Gateway with the times, type and goal received. The Gateway Agent suspends this request once event information is received.

When a DR event starts and the Gateway enters the active state, the Control Bundle (see Task 3 Report) obtains control of the Gateway Agent. As described in the Task 3 Report, the control bundle calculates the total load of all appliances connected to the Gateway. It then uses the Gateway Agent behavior

InformSEB to obtain a control priority. This behavior sends the total starting load of the Gateway to the SEB Agent, and listens for a response. When the response is received the Gateway's Control Bundle starts to enact its control algorithm.

For this test a priority based control algorithm was used. This is described in detail in the Task 3 Report. In short, appliances are assigned a priority from zero to ten. Lower number appliances are controlled or turned off first, and higher number ones are controlled if necessary. In this test case, the SEB would return a maximum priority. This means that the Gateways would control appliances up to and including this priority, but not any further regardless of whether the load reduction goal was met.

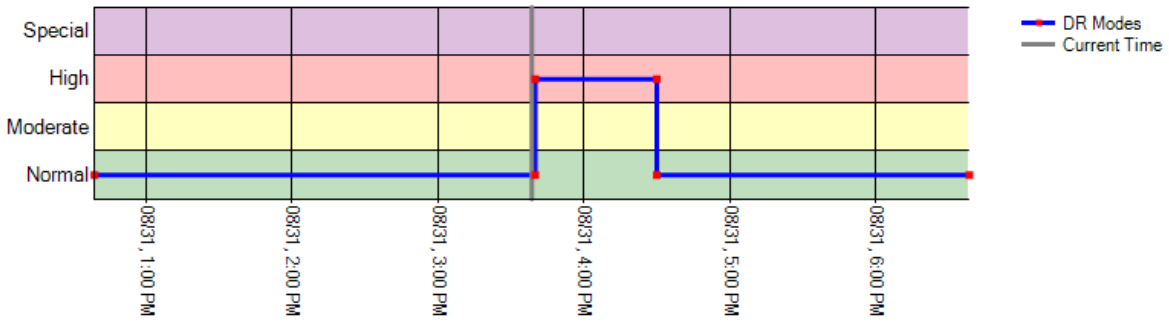
Testing took place on September 6, 2011 in 464 SDH. A few sample events were set up and system functionality was observed. This will describe the course of events of a sample test. We note that both Gateways do not act in perfect synchronization due to initialization time differences as well as computer time differences. This describes the general course of events.

- SEB Agent is initialized on Lime along with JADE runtime
- Gateway running on Grapefruit is initialized along with laptop simulation
- Grapefruit Gateway Agent is started by Grapefruit Gateway
- Gateway running on Lemon is initialized along with its laptop simulation
- Lemon Gateway Agent is started by Lemon Gateway
- Both gateway agents search for other gateway agents and the SEB Agent
- Once the SEB found by the gateway agents, both gateway agents request DR Event information
- Upon reception of DR event information, both gateway agents create an event in their respective gateways.
- Both gateways remain in the passive state until the event starts
- Upon event start, both gateways go through a 15 second delay (this is to account for gateway clocks not being synchronized)
- Grapefruit Gateway calculates its total starting load, which happens to be 250 W
- Grapefruit Gateway requests the control priority from the SEB, and is given 4
- Lemon Gateway calculates its total starting load, which is 150 W
- Lemon Gateway requests the control priority from the SEB, and is given 2
- Grapefruit Gateway turns off all appliances connected to it up to priority 3, whereupon its load reduction goal is met. It then transitions to a passive state and waits until the DR event ends
- Lemon Gateway turns off appliances up to priority 1, whereupon its load reduction goal is met. It then transitions to a passive state.
- When the event ends, both gateways release control of the appliances.

5.8.2 Siemens' demonstration

5.8.2.1 DR Events

Notification Time	DR Event Start Time	DR Event End Time	Event Category
08/31/11 15:35	08/31/11 15:40	08/31/11 16:30	High



5.8.2.2 Load Shedding details

Parameter	Value
TotalPeakLoad	2000 Watts
Negotiation Cycle	3 min
Load Shedding goal	30%
Number of Agents	2

5.8.2.3 Test Cases

5.8.2.3.1 Adapting partial centralized load allocation method

With this approach Participant is mainly responsible to manage appliances, consolidate cost function and meet load setting goal. Here is the detailed sequence of operations with this approach

- SEB manager requests proposal for a particular period of time called negotiation cycle
- Each Participant (Agent) calculates and proposes the total Cost Function according to the portfolio of each Appliance to SEB Manager
- SEB Manager calculates and sends load shed target to each Participant (Agent)
- Participant applies the corresponding load shed command to the Appliances, and report total power consumption to SEB Manager
- This continues until DR period ends

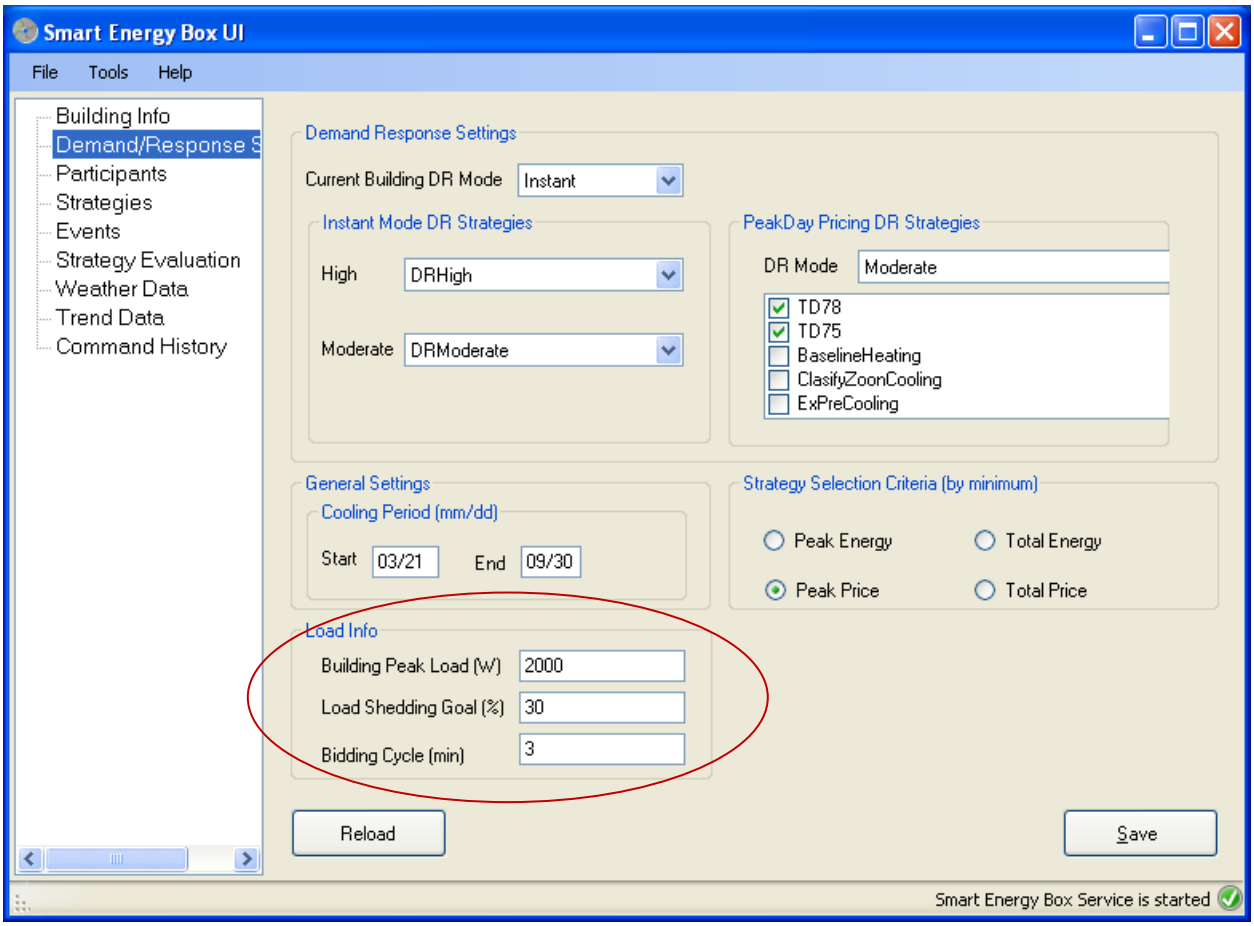
5.8.2.3.2 Precondition

- Establish connection between DRAS and Smart Energy Box
- Establish connection between Smart Energy Box and Smart Energy Box Agent Wrapper (SEBAW)
- Establish connection between Gateway1, Gateway 2 with SEBAW
- Establish connection between Gateway and its smart appliances

5.8.2.3.3 Results

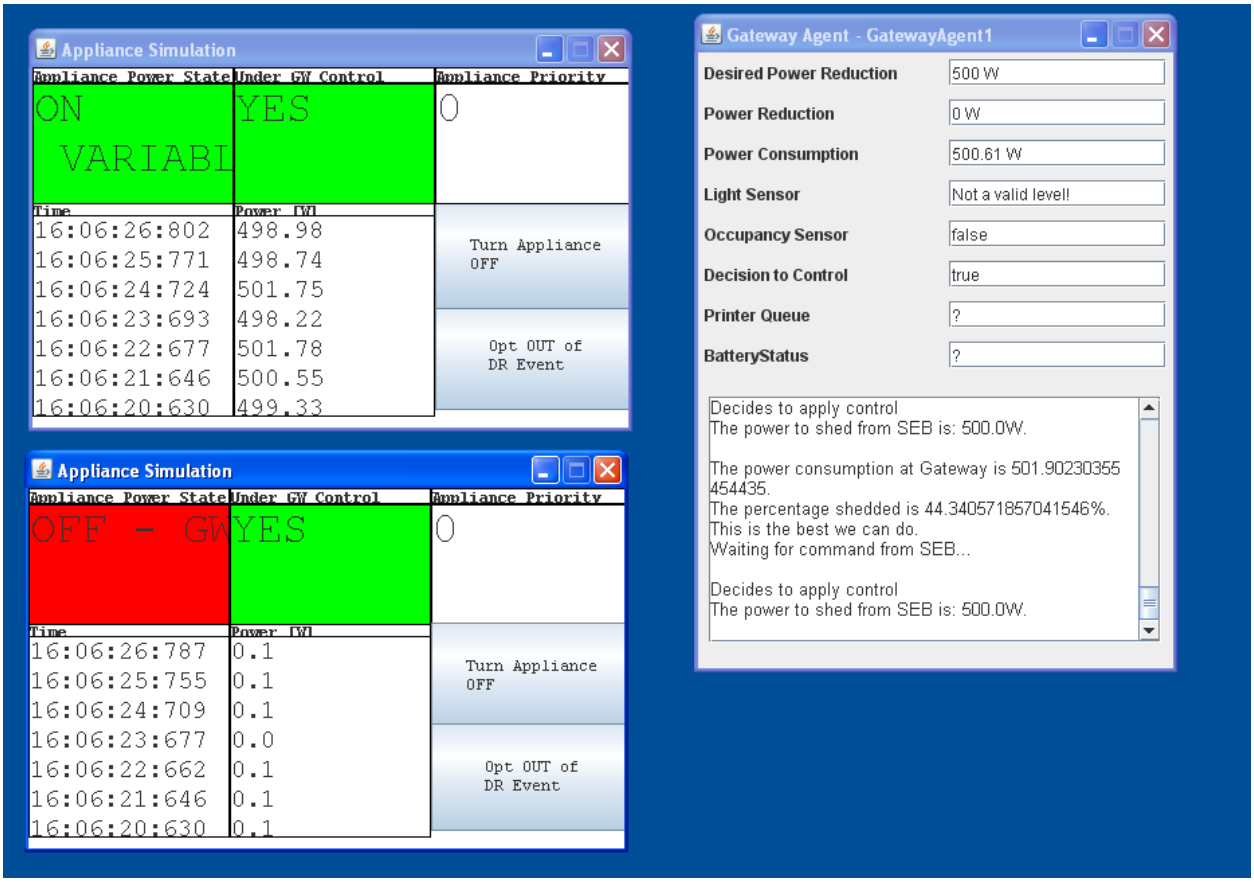
Since total peak load of this setup is 2000W, and load shedding goal is 30%, the Manager tries to shed around 600W of power

5.8.2.3.4 Load shedding settings



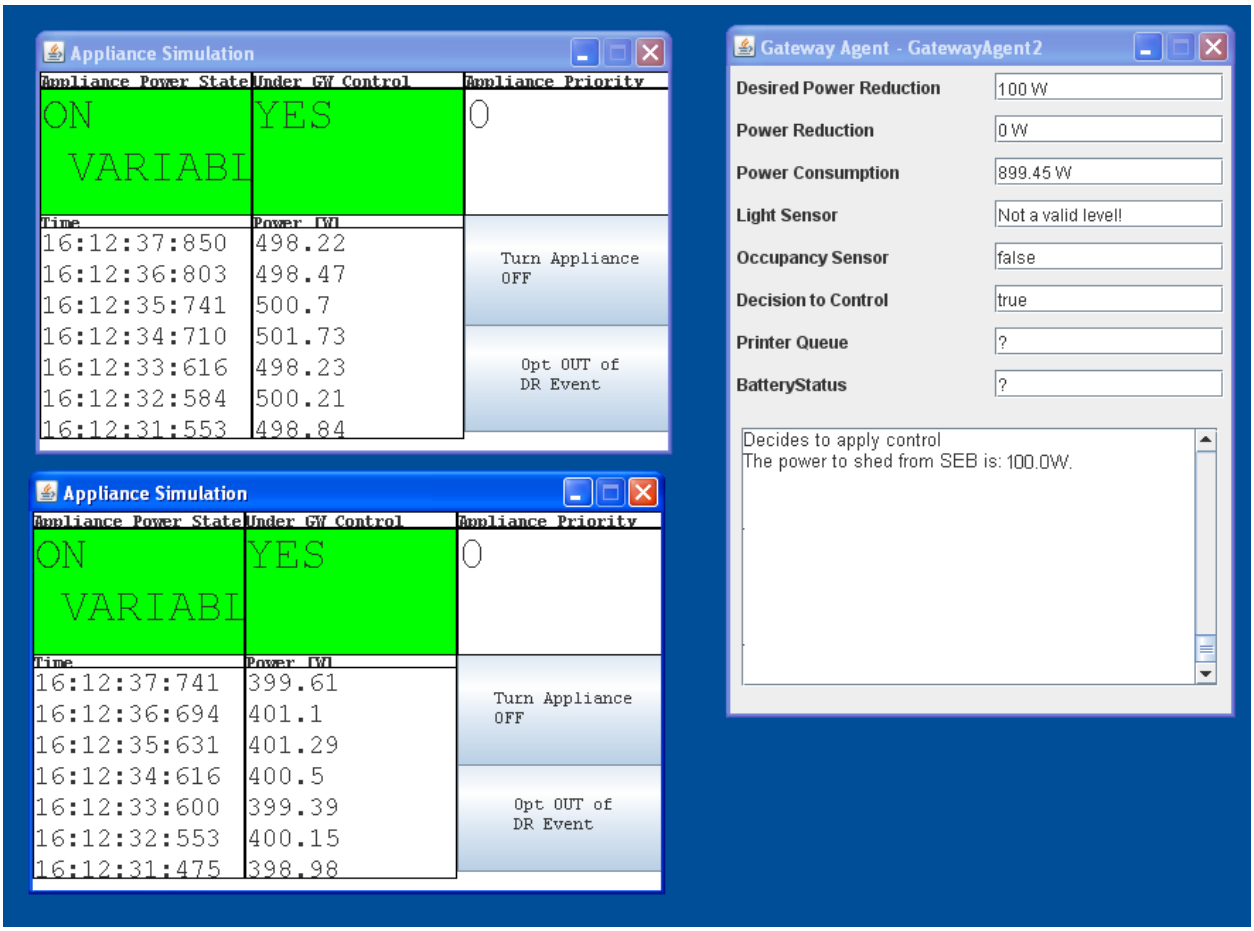
5.8.2.3.5 Gateway Agent 1 and its Appliances

Gateway Agent 1 receives 500W of load shedding goal out of 600W overall goal. The following figure show one of the appliances having capacity of 500W is completely turned down to achieve this goal



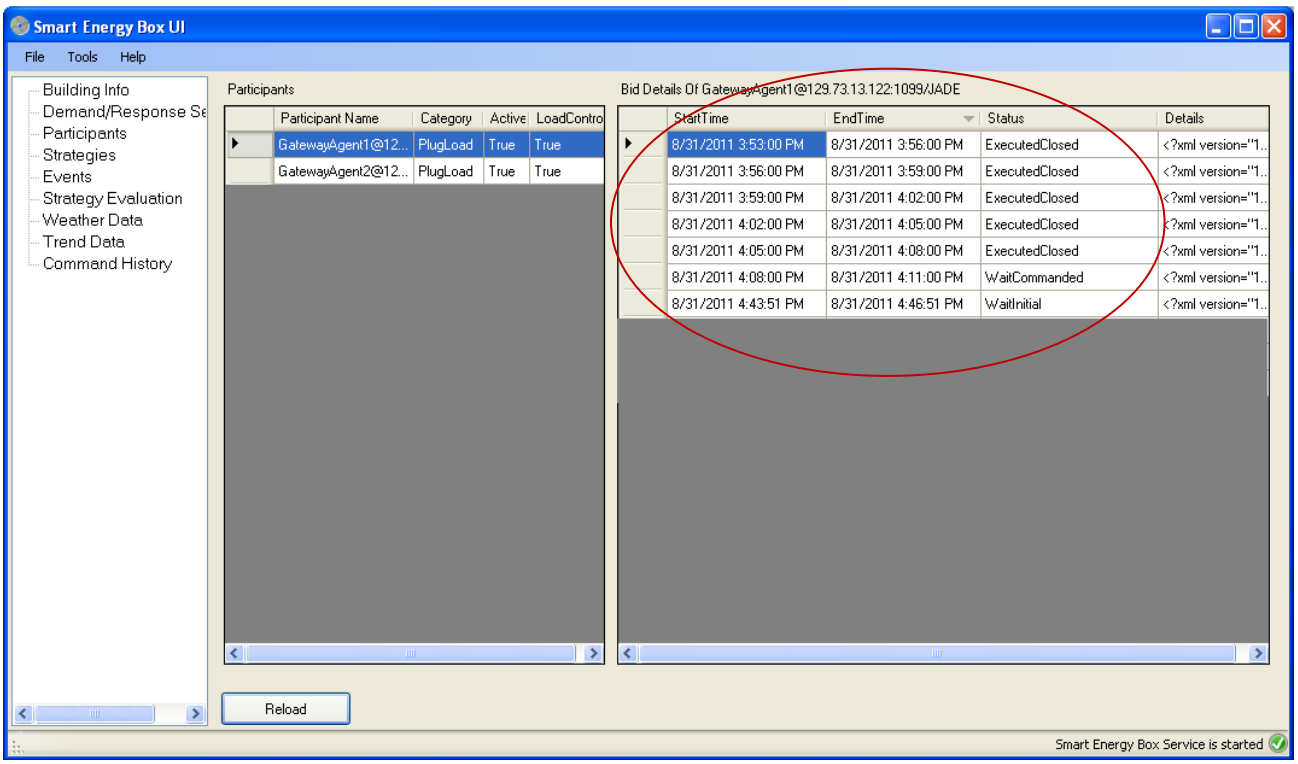
5.8.2.3.6 Gateway Agent 2 and its appliances

Gateway Agent 2 receives 100W of load shedding goal out of 600W overall goal. The following figure show one of the appliances sheds 100W to achieve this goal



5.8.2.3.7 Bid details

The Manager communicates with the Agents to make this negotiation, the following shows bids for every negotiation cycle, i.e. 3 minutes for each agent participates in the system.



5.8.2.3.8 Bid Status

The following table shows status description of Bid

Status Code	Description
Initial	SEB created bid
WaitInitial	Proposal is being prepared by Agent
Proposed	Agent proposes with Cost Function
Command	SEB Commanded with Goal
WaitCommand	Agent processing the command
Execute	Agent Executed the command
ExecuteClosed	Agent successfully Executed, SEB acknowledged the execution
Reject	Agent Rejects the bid
RejectClosed	SEB closes Rejected bid

6 Products of the Project

6.1 Website or other Internet sites with results of this project.

A website has been created and provides the deliverables and the quarterly reports thus far in the project: http://i4energy.org/diadr-project-sutardja-dai-hall-0

The user interface for the commercial gateway has been developed as a result of this project.

6.1 Networks or collaborations fostered.

This project is a joint effort between UC Berkeley, LBNL and Siemens. The project was facilitated by CITRIS. Other collaborations include: Dhaani Systems who has developed a tool to manage computer power and Raritan who has a switchable power strip that also measures energy consumption. In addition, the project has developed collaborations across campus, notably working with Dan Arnold and Kevin Ding who developed the Residential Energy Gateway and user interface, and Andrew Krioukov who developed lighting controls and data visualization tools.

6.2 Technologies/Techniques.

The Gateway (Task 3 report available at http://i4energy.org/diadr-project-sutardja-dai-hall-0)

6.3 Databases

7 References

[1] L. Chen; N. Li; S. H. Low and J. C. Doyle, Two Market Models for Demand Response in Power Networks, 2010 First IEEE International Conference on Smart Grid Communications, pp.397-402, Oct. 2010

[2] A. Papavasiliou, H. Hindi and D. Greene, Market-based control mechanisms for electric power demand response, Decision and Control (CDC), 2010 49th IEEE Conference on , pp.1891-1898, Dec. 2010.

[3] F. Ygge, Market-Oriented Programming and its Application to Power Load Management, Doctoral dissertation of Lund University, 1998.

[4] J.F.Kurose and R. Simha, A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems, IEEE Transactions on Computers, Vol. 38, No. 5, May 1989, pp.705-717.

[5] K. Steiglitz, M. L. Honig and L.M. Cohen, A Computational Market Model Based on Individual Action, in Market-Based Control: A Paradigm for Distributed Resource Allocation, Scott Clearwater (ed.), World Scientific, Hong Kong, 1995, pp. 2-17.

[6] A. Homaifar, C. X. Qi, S. H. Lai, Constrained Optimization via Genetic Algorithms, Simulation, Vol. 2(4) 1994, pp. 242-254.

[7] K. Parsopoulos and M. Vrahatis, Particle Swarm Optimization Method for Constrained Optimization Problems. Intelligent Technologies, Theory and Applications: New Trends in Intelligent Technologies, IOS Press, 2002.

[8] J. H. Mathews and K. K. Fink, Numerical Methods Using Matlab, 4th Edition, 2004.