sustainability

MDPI

Article

Power Grid Simulation Testbed for Transactive Energy Management Systems

Ozgur Ozmen [1,*], James Nutaro [1], Michael Starke [2], Jeffrey Munk [3], Larry Roberts [1], Xiao Kou [4], Piljae Im [3], Jin Dong [3], Fangxing Li [4], Teja Kuruganti [1] and Helia Zandi [1]

[1] Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA; nutarojj@ornl.gov (J.N.); lar@cynandlar.com (L.R.); kurugantipv@ornl.gov (T.K.); zandih@ornl.gov (H.Z.)

[2] Electrical and Electronics Systems Research Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA; starkemr@ornl.gov

[3] Energy and Transportation Science Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA; munkjd@ornl.gov (J.M.); imp1@ornl.gov (P.I.); dongj@ornl.gov (J.D.)

[4] Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, USA; xkou1@vols.utk.edu (X.K.); fli6@utk.edu (F.L.)

* Correspondence: ozmeno@ornl.gov

Received: 10 April 2020; Accepted: 21 May 2020; Published: 28 May 2020

Abstract: To effectively engage demand-side and distributed energy resources (DERs) for dynamically maintaining the electric power balance, the challenges of controlling and coordinating building equipment and DERs on a large scale must be overcome. Although several control techniques have been proposed in the literature, a significant obstacle to applying these techniques in practice is having access to an effective testing platform. Performing tests at scale using real equipment is impractical, so simulation offers the only viable route to developmental testing at scales of practical interest. Existing power-grid testbeds are unable to model individual residential end-use devices for developing detailed control formulations for responsive loads and DERs. Furthermore, they cannot simulate the control and communications at subminute timescales. To address these issues, this paper presents a novel power-grid simulation testbed for transactive energy management systems. Detailed models of primary home appliances (e.g., heating and cooling systems, water heaters, photovoltaic panels, energy storage systems) are provided to simulate realistic load behaviors in response to environmental parameters and control commands. The proposed testbed incorporates software as it will be deployed, and enables deployable software to interact with various building equipment models for end-to-end performance evaluation at scale.

Keywords: demand response; distributed energy resources; simulation-based software testbed; transactive control; transactive energy

1. Introduction

The development of embedded software has long benefitted from extensive support by simulation. The central feature of simulation technology in that domain is the ability to simulate the computer hardware on which the software runs, and to position the computer hardware model within a simulated world; see, e.g., [1–4] and the industry oriented overview by Zeeshan Anwar [5]. However, the commercial offerings and literature are extensive. Indeed, crucial stages in the development and testing of a software system may be completed before hardware is available. Testing is greatly facilitated by being able to execute the operational software stack—including device drivers, operating system, and application—on computer hardware models that run faster than real-time. In other environments, where detailed physical processes must be simulated to stimulate the software system, the ability to run fielded software stacks in a slower than real-time test is

indispensable for ensuring high quality software. More important still, simulated hardware can be made available to every software developer that needs it; hardware in the loop rigs and detailed, expensive real-time simulators—though still essential for final checkout—cannot be economically supplied to every software developer for day to day needs.

A distinguishing feature of most embedded computer systems is that they involve computing hardware which is much less powerful than the engineering workstations on which simulations are run. Consequently, detailed models of the target hardware in the target environment can be executed on a real-world timescale that is commensurate with development and testing. This is often not true of modern, transactive energy management systems. These complex software systems can involve modern workstations interacting with a variety of web-based services executing on modern hardware.

Nonetheless, we contend that the cost-effective creation of very large scale transactive energy management systems will involve the development of large, complex software systems that must be subjected to testing at scale. This can be done cost-effectively if simulation tools, like those available to embedded systems developers, are made available for the creators of transactive energy systems. Here, we demonstrate this possibility by modifying the well-known QEMU virtualization tool so that its simulated computing hardware is governed by a simulation clock rather than real-time. Our new simulation capability is integrated with a large-scale model of the energy infrastructure to be controlled, and this simulation is used to exercise the software under test as if it had been fielded.

In the past decade, significant research effort has been invested into demand response (DR) and distributed energy resource DER projects and deployments. Early on, a substantial focus was placed on the quantity and capability of DR and DER assets needed to provide ancillary services [6]. In 2014, an extensive study was completed that evaluated the potential quantity and quality of DR and DERs, their potential market value, and barriers to technology maturation [7–9]. Although the study noted that the residential sector appeared to have significant resource potential, these resources were often deemed cost-prohibitive to implement compared with larger providers, such as commercial buildings and industrial loads. Specifically, the available equipment and communication infrastructure was not sufficient to realize the DR potential in a cost-effective way. Hence, an approach for integrating residential energy systems into broader system services at a sufficiently low cost is crucial to wide scale implementation (Abrishambaf et al. [10] provides a detailed review of transactive energy (TE) system architectures).

The changing landscape of computer technology and the development of the internet have led to the interconnection of devices, typically called through the Internet of Things (IoT). Today, many household devices such as heating, ventilation, and air conditioning (HVAC) [11–14] and electric water heaters can interconnect to cloud-based services through residential Wi-Fi [15,16]. This has provided a wealth of opportunities to observe, control, and optimize energy use in ways that were previously impossible.

For example, IoT and new smart grid technologies now offer the ability to incorporate residential loads into real-time energy markets operated via transactive energy (TE). As defined by the GridWise Architecture Council, TE is "a system of economic and control mechanisms that allows the dynamic balance of supply and demand across the entire electrical infrastructure using value as a key operational parameter" [17]. Hence, a main component for performing TE is ensuring that a value term is shared between parties for operation.

Liu et al. [18] discusses the challenges and opportunities of transitioning from wholesale electricity markets to a TE system framework and explores the new responsibilities of various parties within the electricity market. Hammerstrom et al. [19] presents state-of-art TE techniques and several pilot projects with transactive control, including Pacific Northwest Gridwise Testbed [20], AEP Ohio gridSmart project [21], and Pacific Northwest Smart Grid Demonstration [22]. In Hu et al. [23], the effects of advanced home energy management systems (HEMS) on distribution networks are studied. In Stecklein et al. [24], the architecture of a smart HEMS is presented with case studies that show the benefits of HEMS. In Gkatzikis et al. [25], a system-level hierarchical structure for wide-area energy

management systems is proposed, and the benefits for introducing load aggregators to the electricity market are discussed.

This previous research has led to implementations of TE management systems. However, distribution systems and residential buildings vary by climate region and utility. Barring a full-scale implementation of a proposed system, it is impossible to fully understand and anticipate the impact of communication, control, and optimization schemes operating at a large scale in real-time. This makes simulation an indispensable tool for creating practical TE systems.

The challenges for creating a scalable simulation testbed for TE systems include: (1) allowing users to flexibly customize the system scale, architecture, and control mechanism for various distribution networks; (2) establishing the detailed and realistic home appliance and electric devices models to accurately simulate the behaviors of residential loads towards different scenarios and controls; (3) enabling two-way communication among different agents to study multi-agent interactions; and (4) hosting operational software in a non-real-time simulation environment. To address these issues, a simulation platform was developed and is presented in this paper to compare the performances of various designs and control schemes for TE systems. The main contribution of this paper is a testbed that:

(1) supports the modeling of multiple houses by interacting with their individual HVAC, electric water heater, energy storage, and photovoltaic (PV) systems and simulating various control mechanisms for residential energy management systems;
(2) supports detailed models of responsive home appliances and DERs, such as HVAC, electric water heaters, energy storage, PV, and electric vehicle (EV) stations, which provide increased fidelity for comparing different environmental parameter and control command impacts;
(3) contains the complete, deployable software agents that constitute the TE system;
(4) evaluates the communication among interacting agents at subsecond timescales, thereby allowing detailed assessments of the communication infrastructure that will be needed in the actual deployment; and
(5) hosts operational software in a simulated computer system whose progress through time is not linked to the progress of the wall clock allowing faster than real-time testing.

The latter testbed feature ensures that the computational complexity of models of buildings, building equipment, communication systems, and other environmental aspects are not restricted by a need to execute in real-time. A testbed without a real-time requirement also permits the use of computational resources, particularly very large high-performance computing (HPC) systems, using task schedulers that might be incompatible with the execution of a real-time simulation. This capability for fielding the non-real-time execution of software distinguishes the proposed testbed from related work, such as the Building Controls Virtual Testbed [26], the agent-based control in the loop simulator described by Huang et al. [27], and similar systems to integrate building control software into a virtual building (as seen in Huang et al. [27]). The proposed testbed can be conceived of as a controller in the loop technology, similar in most respects to those cited but without the associated requirement for the real-time execution of the simulation models.

The rest of this paper is organized as follows. Section 2 introduces existing work in platforms of this nature. Section 3 discusses the methods and materials of the platform, including the quick emulator (QEMU) computer system emulator, agent-based architecture for the testbed, and mathematical models for responsive devices in each house. Section 4 presents the results on how this infrastructure has been tested, and Section 5 concludes the paper with discussions.

2. Background

A primary goal of the proposed testbed is to allow software as it will be deployed to interact in simulated time with simulated equipment. This capability is essential for assessing the performance, robustness, and reliability of the actual software system when it is put into the field. Over the past decade, considerable progress has been made toward simulating computer systems with sufficient

speed and fidelity to execute complete software stacks—comprising an operating system, device drivers, and applications—within a larger, simulated environment.

Recent approaches to building these types of simulations have taken two approaches. One is to modify a hypervisor, turning it into a simulation engine [1,28,29]. The other is to use a computer system emulator with its simulated hardware advancing in step together with an overarching simulation clock [2,3,30–32], which needs QEMU modification to achieve that [33]. The chief goal of these prior efforts has been to avoid modeling the elements of a software system. For instance, the ability to use real software in a simulation is essential for simulating attacks on computer systems or for building software that will use hardware that is not yet available.

Prior work was followed using QEMU as the computer system emulator, extending it to interact with time-managed simulations. The QEMU computer system simulator modifications can be used as a component in a large, discrete event simulation. Within the overarching simulation, communication networks can be simulated at various levels of fidelity and physics-based models of electrical networks, building equipment, energy storage devices, and other relevant subsystems [34–37]. A central feature of the proposed simulation is its use of the split-system method to combine models of discrete events and physical processes. This technique preserves the desirable properties of numerical integrators used to solve differential equations that describe physical equipment while retaining the temporal precision and speed of the event-scheduling simulation [38]. The numerical technique employed in this simulation tool has been discussed extensively in the aforementioned references. Hence, this work focused on an approach for integrating the computer system into the discrete event simulation. This is followed by a case study that demonstrates its application to testing, and an analysis of deployable, large-scale, TE systems.

3. Materials and Methods

3.1. The QEMU Computer System Emulator

QEMU is a computer system emulator for executing real software on simulated hardware. The simulated hardware includes memory, system buses, timers and clocks, and peripheral equipment [39,40], such as disks, network cards, video cards, mice, keyboards, and serial ports. The simulation time for these subsystems is managed with an event list called the *virtual timer*. This timer allows events to be scheduled that will occur at some future value of the simulation clock. When the event time is reached, QEMU stops executing the emulated computer and invokes an event handler. The execution of the emulated computer resumes when the event handler returns. For this work, it is crucial that: (1) event handlers manage the states of simulated timing hardware and (2) the emulated microprocessor can be halted while an event handler is running. The virtual timer makes it possible to use QEMU as a component in an overarching, constructive, discrete event simulation by leveraging a discrete event simulation (ADEVS) [39,40].

The emulated microprocessor can execute instructions in one of two ways. In the first, sequences of software instructions are broken into chunks called *translation code blocks*. These are passed to a just-in-time compiler that translates the code blocks into instruction sequences for the computer on which QEMU is executing. In the second, QEMU passes instructions to the Linux kernel virtual machine (KVM) to directly execute on the available hardware.

The just-in-time compiler is enabled when QEMU operates in its *icount* mode. In this mode, time within the emulator is advanced by a fixed (usually single) number of nanoseconds for every emulated instruction that is executed. When an event handler is invoked, the emulated microprocessor stops immediately, and because time is advanced by the execution of instructions, the actual instruction execute rate does not deviate from the desired instruction execution rate. The *icount* mode can be configured so that the simulation proceeds forward in time to the next event in the virtual timer's event list when the emulated computer has no work to do. Hence, when the emulated hardware is idle, QEMU proceeds as ADEVS by making instantaneous jumps through virtual time (applications can be found in Nutaro and Ozmen [39,40]). On the other hand, executing each instruction with the emulated microprocessor requires numerous instructions on the physical

processor on which the just-in-time compiler executes. Hence, computationally active instruction sequences will tend to force any simulation to run much slower than real-time.

KVM overcomes this problem using the physical microprocessor as the emulated microprocessor, thereby skipping the translation step and enabling near real-time execution speeds. In this mode, time advances at the wall-clock rate while the emulated microprocessor is processing instructions. To synchronize time with an external model, the emulated microprocessor is halted while the synchronization protocol is executing. A request to suspend the threads executing the guest instructions is issued to the operating system. The delay between issuing this request and the operating system acting upon it can cause the emulated microprocessor to overrun the intended synchronization time. Thus, although this execution mode offers near real-time performance, it gives rise to the possibility of intermittent synchronization errors.

The coordinated advancement of time within QEMU and the overarching simulator are illustrated in the sequence diagram of Figure 1. Upon startup, the modified QEMU software attaches to a UNIX domain socket, called the synchronization socket, that is created by the overarching simulator. QEMU waits for the external model to write a time advance value to this socket. Upon receiving this time advance, QEMU schedules a virtual timer event for that interval into the future. The emulator executes instructions until that virtual timer event occurs. When the event occurs, QEMU halts the emulated microprocessor, records the simulation time that has actually elapsed when execution halts, and writes that time to the socket. In the *icount* mode, this time will match the scheduled event time; with KVM, the elapsed time could exceed the scheduled event time. If this occurs, then the error is corrected by scheduling a catch-up event in the overarching simulator for the reported overrun, and then allowing QEMU to execute again after the catch-up event has occurred. This process repeats until the simulation is terminated.

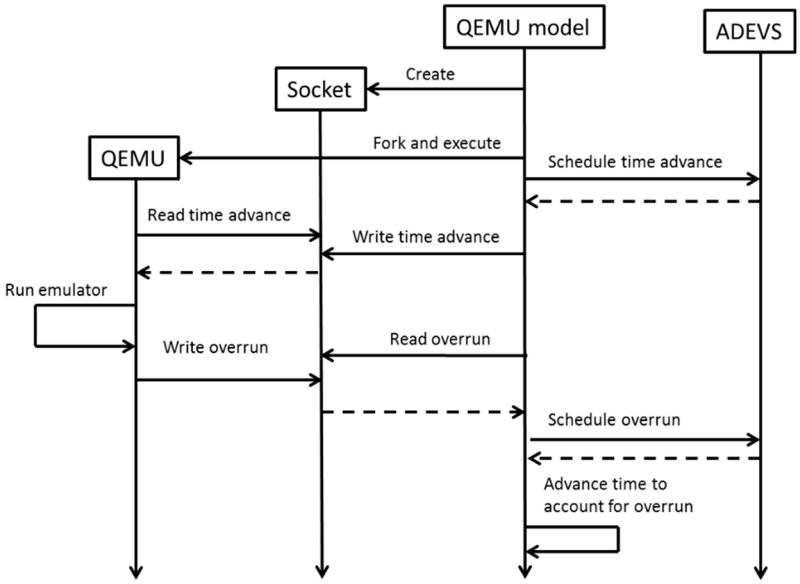


Figure 1. Time Synchronization between QEMU and the overarching simulator.

The software executing on the emulated computer can interact with other models via an emulated serial port and network interface card. Communication between these QEMU device models and the discrete event simulator also occurs through a UNIX domain socket. During the execution of a time advance by QEMU, the simulated devices may write data to this socket. For network cards, these data are ethernet frames sent by the software executing on the emulated computer. For serial ports, these data are sequences of characters written to the serial device by the software running on the emulated computer. When QEMU completes its time advance, the overarching simulator reads these data from the appropriate UNIX domain sockets, and it writes to these sockets any frames or characters that have been transmitted to the emulated computer.

If large amounts of data are to be exchanged between QEMU and the overarching simulator, then the buffers backing the UNIX domain sockets could become full and cause the sender to block data. This will cause the simulation time to stop advancing if the intended recipient is waiting for a time advance to complete before reading from the socket. This problem is eliminated by having the overarching simulator dedicate a thread for reading and writing to each socket. The reading thread loops on a blocking read of the socket. As messages become available, they are placed into a queue from which the overarching simulator will extract them at the next synchronization point. The writing thread accepts data to be sent to the QEMU device and queues this for transmission. It writes queued messages to the socket using a blocking write. Thus, a thread is always available to extract data from the buffer and to write data to the buffer while the simulator advances.

Timing errors in this synchronization protocol can result from two sources. One source is the timer overrun that can occur when using KVM. The other source is the artificial delay of messages exchanged with the emulated computer. With the proposed scheme, data are exchanged at the synchronization points between QEMU and the overarching simulation. A necessary consequence of this approach is that messages will experience a delay between zero- and single-time advance plus the overrun. This error could be further exacerbated by real delays imposed by the UNIX domain sockets and the scheduling of threads that monitor those sockets. If these real delays are longer than the real-time required to execute a time advance, then the difference will cause an undesired delay in the simulation equal to the number of time advances that elapse while I/O operations are completed. Practical experience with the simulation tool suggests that these errors are negligible in the context of the control applications of interest. This is partially because the simulations will run slower than real-time, which causes the simulation models to perceive real-time activity, such as activity within the host operating system kernel, as occurring at a highly accelerated rate. A detailed analysis of these errors is deferred to future work.

3.2. Agent-Based Architecture

The proposed testbed was used to evaluate the robustness of a hierarchical, distributed, agent-based control system that coordinates the interactions between appliances in several houses to create a desired load shape. The proposed scalable TE simulation testbed is based on an agent-based execution in an IoT architecture. The agent-based approach has been widely applied in applications that require interactions and coordination among individuals within a community. The biggest advantage of an agent-based control is that relatively simple behavioral rules for each agent can emerge from complex system-level behaviors.

In the proposed architecture, the concept is to incorporate multiple decision-making entities into a common framework. These entities include the residential building, aggregators, distribution utility, and independent system operator (ISO), as shown in Figure 2. Today, in the existing utility framework, the data pass through many layers of prediction and arbitration, with the physical flow of information being bottom-up or top-down. The agent-based control follows this model while allowing decision-making and evaluation to occur at each level. Figure 3 gives a high-level overview of how the agents are organized in a hierarchical fashion.

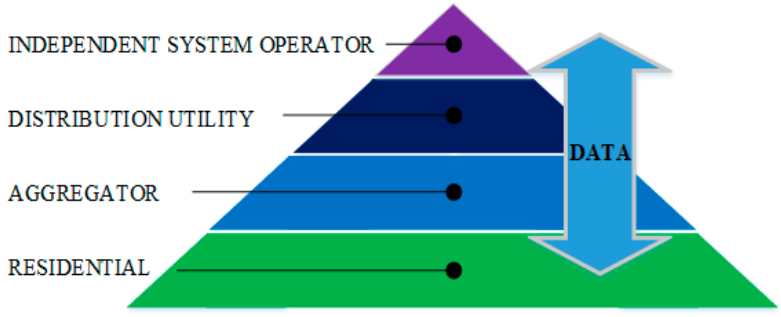


Figure 2. System hierarchy.

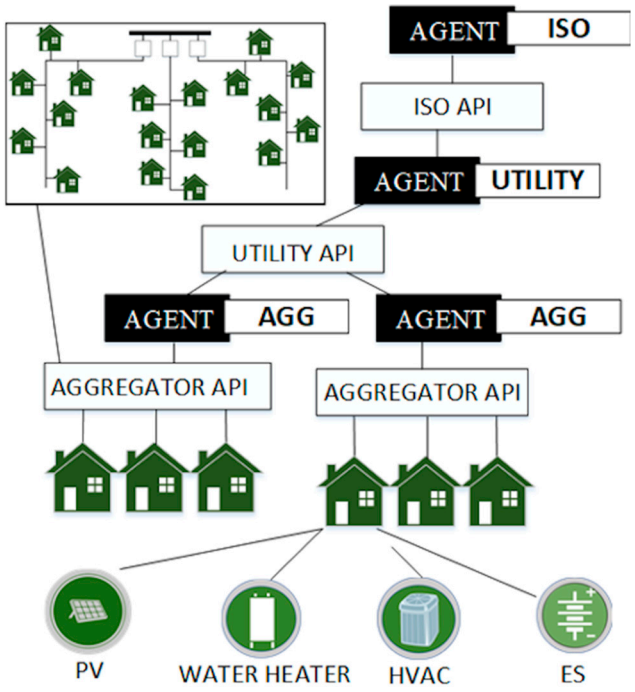


Figure 3. Implementation of agents and RESTful API.

The software that implements individual agents executes on top of the Ubuntu Linux operating system. The agent software and its operating system are installed in the modified QEMU, which allows the agents to interact in a simulated operational environment. A representational state transfer (RESTful) application program interface (API) is used to exchange data between components. The API is written in Python and uses the Flask web application framework. HTTP requests GET and POST are used to transfer data in a JavaScript Object Notation format.

Each API has a historian provided by the PostgreSQL object-relational database system using the standard querying language (SQL). This provides a record of web-service activity that hosts API, and the logs are used for posttest review and analysis. Agents are also coded in Python, and they exchange information indirectly by writing to and reading from the APIs hosted by the various web services.

The simulated environment assigns a unique IP addresses to each host. A simple model of a communications network is used to exchange ethernet frames generated by the simulated network cards using transmission control protocol/internet protocol within the QEMU computer system models. This arrangement allows the agents being tested to interact in simulated time with network services that are identical to those appearing in the anticipated operational environment. Crucially, this allows developers to evaluate the agent software via simulation in the exact form in which it will be deployed for operation.

Table 1 lists the agents and their basic functionality. At each level, separate electrical models can be incorporated to link the electrical consumption of the residential buildings to the grid consumption (modeled as constant real and reactive power loads). Any number of utility and aggregator agents can exist to distribute building management as their numbers grow.

Table 1. Stakeholder agents.

Agent	Purpose
ISO	GETs data from ISO API, evaluates and POSTs a corresponding value signal to the ISO API for the utility.

Utility	GETs data from utility API evaluates and POSTs data to ISO API. Also, GETs data from the ISO API, evaluates and POSTS corresponding value signal to utility API for the aggregator.
Aggregator	GETs data from aggregator API evaluates and POSTs data to utility API. Also, GETs data from the utility API, evaluates and POSTs corresponding value signal to aggregator API for the residential building.

At the residential level (or at each home), an additional framework of agents and APIs was created to establish flexibility and fidelity of the residential models and controls, as presented in Figure 4. At the building level, agents are present that extract data (value signal) from the aggregator, perform optimization and decision-making, and distribute the resulting control commands to other agents who interface with a set of APIs that control building equipment. Details of the agents are provided in Table 2.

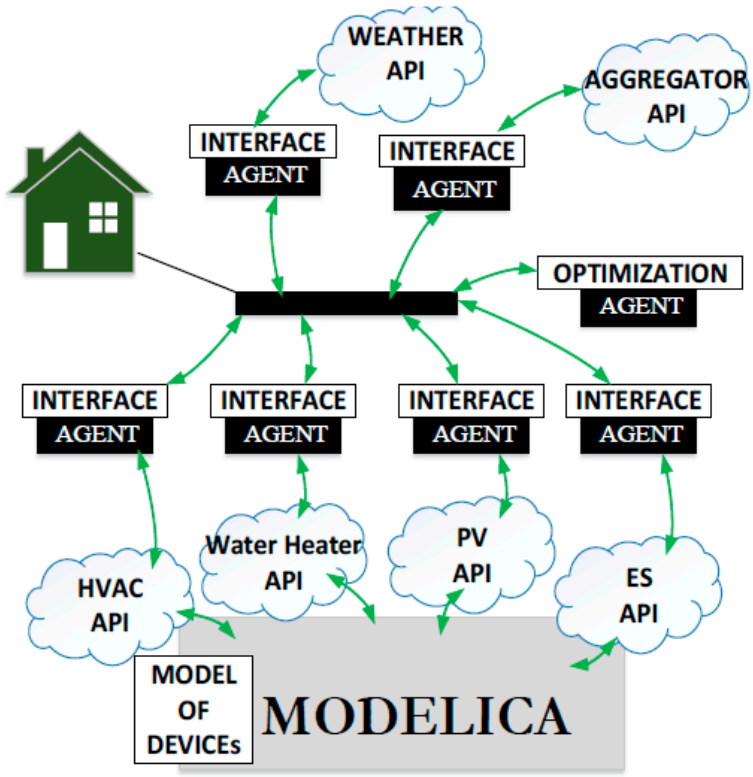


Figure 4. Residential building level of APIs and AGENTS and links to models.

Table 2. Residential building agents.

Agent	Purpose
Interface	*Aggregator*—GET value signal information from the aggregator (AGG) API and POSTs to the message bus. Extracts data from the message bus and POSTs to the AGG. *HVAC, water heater, PV, energy storage (ES)*—Extract control commands from the message bus and POST these to the load API and GET measurement and performance data from the APIs and send these to the message bus.

	Weather—GET forecasted weather data (e.g., solar irradiance and temperature) and send this information to the message bus.
Optimization	Extract data from a local message bus regarding the current states of the HVAC, electric water heater, PV, and energy storage and weather forecast data; perform optimization based on value signal from aggregator; and POST control decisions.

3.3. Models Utilized for Each Residential Home

Resources such as energy storage, PV, HVAC, and electric water heaters are modelled using the MODELICA language and are incorporated into the discrete event simulation using the functional mockup interface (FMI) standard [41]. The web servers that manage these modeled resources communicate with them via a serial protocol. Data are exchanged between these models and the web server software via a simulated serial port that is part of the modified QEMU.

3.4. Building Model

A simplified building model and HVAC performance curves are used to simulate the thermal response of the building. The approach used to develop the simplified model is to (1) create a representative EnergyPlus single house building model; (2) simulate the building thermal response under constant and varying heating and cooling setpoint conditions; (3) use a portion of the EnergyPlus simulation results to train the parameters of the simplified model; and (4) use the remaining simulation results to validate the trained parameters. The original EnergyPlus model was developed to represent a prototypical code-compliant house in Nashville, Tennessee [42]. The house is a two-story, 229 m^2 home with three bedrooms and two bathrooms based on data collected by the US Census Bureau [43]. The model determined the average air temperature inside the home based on the resistor capacitor (RC) model used by Cui et al. [44]. The simplified building model consists of a set of four ordinary differential equations used to represent the average temperatures of the indoor air, internal mass (e.g., furniture and internal walls), exterior walls, and attic temperatures (Equations (1)–(4)). The heat transfer through the exterior walls and roof is driven by the sol-air temperature, which is a representative temperature used in simple models to account for convective and radiative heat exchange in a single term [45]. To capture the effect of varying levels of solar radiation throughout the day, the incident solar radiation is calculated for each face of the building, and an area-weighted average is used to calculate a single sol-air temperature for a group of surfaces. Similarly, the area-weighted average of the solar radiation on window surfaces is calculated as Q_{solar} and represents the solar gains through windows. Infiltration into the homes is simulated using a linear model based on the outdoor wind speed [46].

$$C_{in}\frac{dT_{in}}{dt} = \frac{T_{wall}-T_{in}}{R_{wall}/2} + \frac{T_{mass}-T_{in}}{R_{mass}} + \frac{T_{attic}-T_{in}}{R_{attic}} + Q_{IHL} + A_{HVAC}Q_{HVAC} + A_{infil}v_{wind}\,(T_{out} - T_{in}), \quad (1)$$

$$C_{wall}\frac{dT_{wall}}{dt} = \frac{T_{in}-T_{wall}}{R_{wall}/2} + \frac{T_{sol,wall}-T_{wall}}{R_{wall}/2}, \quad (2)$$

$$C_{wall}\frac{dT_{attic}}{dt} = \frac{T_{in}-T_{attic}}{R_{attic}} + \frac{T_{sol,roof}-T_{attic}}{R_{roof}}, \quad (3)$$

$$C_{mass}\frac{dT_{mass}}{dt} = \frac{T_{in}-T_{mass}}{R_{mass}} + Q_{solar}, \quad (4)$$

3.4.1. HVAC Model

The HVAC capacity and power consumption are modeled using bi-quadratic performance curves that are a function of outdoor air temperature and indoor dry bulb temperature for heating

and indoor wet bulb temperature for cooling (Equations (5) and (6) with subscript x indicating dry bulb or wet bulb temperature depending on the mode). These curves are the same format as those used in EnergyPlus [47], and the coefficients used are from BEopt [48] for a single-stage, 14 SEER (seasonal energy efficiency ratio), and 8.2 HSPF (heating seasonal performance factor) heat pump. The curves return a capacity or energy input ratio (EIR) modifier that is multiplied with the rated capacity or EIR to determine the capacity or EIR at the specified conditions. The rated values are based on the Air-Conditioning, Heating, and Refrigeration Institute 210/240 test conditions for cooling [49]. The EIR is defined as the ratio of energy consumed to the heating or cooling energy delivered, and is the inverse of the coefficient of performance.

$$QMod = A_1 + A_2(T_{out}) + A_3(T_{out})^2 + A_4(T_{in,x}) + A_5(T_{in,x})^2 + A_6(T_{out})(T_{in,x}), \quad (5)$$

$$EIRMod = A_7 + A_8(T_{out}) + A_9(T_{out})^2 + A_{10}(T_{in,x}) + A_{11}(T_{in,x})^2 + A_{12}(T_{out})(T_{in,x}), \quad (6)$$

To create an arbitrary number of distinct building models, the first step taken was to create a simplified model of a house calibrated to the results of EnergyPlus simulations. The model was run for the first week in January and the first week in July with varied heating and cooling setpoints to generate data suitable for training the unknown variables in the simplified building model. Particle swarm optimization was used to minimize the sum of the root mean squared error of the indoor temperature, the error fraction of the total heating/cooling, and the error fraction of the number of heating/cooling cycles. The calibrated model parameters are revised randomly in Table 3 to generate the desired number of realistic houses with different configurations.

Table 3. Range of values for generating 100 building models.

Parameter	Lower Range	Upper Range
Building *R* values	−15%	+15%
Building *C* values	−15%	+15%
Qrated	−10%	+40%
EIR	−30%	+10%

Hourly internal heat loads (IHLs) were generated with the generation of an indoor heat and moisture tool [50], which provides stochastic loads based on underlying statistical distributions of occupant behavior.

3.4.2. Water Heater Model and Hot Water Use

The water heater is modeled as a standard storage tank model with an electric resistance heating element using a simple single-node model (Equation (7)). Stratification within the tank is not modeled. The water heater is modeled with a single heating element of 4500 W. The *R* and *C* values for the tank can be calculated based on the volume of the tank and measured insulation thickness. As with the building models, the water heater models are varied to create new instances for each home. Since there is not much variation in residential water heaters, the *R* and *C* values are calculated for the most common sizes and efficiencies of water heaters, and these values are selected randomly for each home (see supplementary material TableS1 for the overview of notations).

$$C_{tank}\frac{dT_{tank}}{dt} = \frac{(T_{amb} - T_{tank})}{R_{tank}} + Q_{WH} + (\rho\dot{V}c_p)_{water}(T_{cold} - T_{tank}), \quad (7)$$

Hot water consumption varies significantly between homes. To provide varied hot water draws for the multiple homes, the Building America Domestic Hot Water Event Schedule Generator [51] was used to generate randomized hot water use profiles based on statistical data.

3.4.3. Energy Storage Model

In developing the ES model, the primary concern is the energy entering and leaving the energy storage system during the time interval. This is captured in the element state of charge (SOC), which represents the percentage of energy within the energy storage system in relation to the capacity of the energy storage medium. The battery model includes losses for charging and discharging and steady-state losses for powering the required electronics when the battery storage controls are on. The storage model also limits the maximum charging/discharging rate of the battery to the specified limit, and maintains the SOC within prescribed limits to ensure battery functionality and reliability. The charge and discharge effects on the energy storage system are given by Liu et al. [52]. Figure 5 shows the ES model diagram.

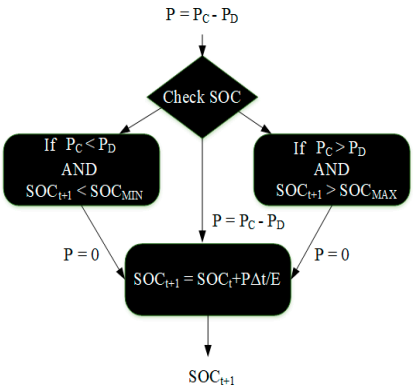


Figure 5. Energy storage system diagram.

3.4.4. Photovoltaic Generation Model

The produced PV generation for the model considers a simplified model that is directly correlated with the solar irradiance. In this case, the output is assumed to be driven by a maximum power point-tracking inverter [53]. The received solar irradiance is a function of the PV array angle and is reflected directly in the calculation. Figure 6 illustrates the PV model.

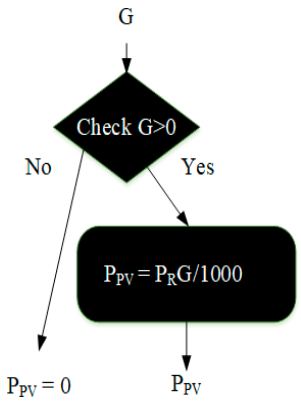


Figure 6. PV model diagram.

3.4.5. Coupled Building Model

A building model represents the residential home dynamics that receive IHLs (Q_{IHL} in Equation (1)) and the weather file (weaDat) as inputs. The HVAC model also receives weather data as an input. The PV model is incorporated in the storage model representing the energy storage system. Its inputs are solar calculations and energy demands from the other devices. The solar model is implemented separately as a direct solar irradiation based on the tilted surface model (as seen in the directed tilted surface model in Wetter et al. [54]). The home model uses solar irradiance in solar heat gain through the windows and the average sol-air temperatures of the roof and wall surfaces. The solar-air

temperature is the effective outdoor temperature that, in the absence of radiation heat transfer, provides the same heat transfer through a surface as the outdoor temperature alone and all radiation heat transfer sources. The sol-air temperature allows radiation exchange to be omitted from the building model equations since the effects are captured in the solar-air temperature. The solar-air temperature and solar irradiance are calculated for each surface, and an area-weighted average is used to calculate the average solar-temperature and solar irradiance on a group of surfaces (i.e., roof and wall surfaces). The water heater model represents the standard storage tank model with an on/off controller and inputs hot water consumption (HotWater). The HVAC schedules and home features (Table 3) are also inputted to the coupled model at initialization. Figure 7 represents the coupled model of all device and physical submodels in the Dymola [55] User Interface. This model is exported as a functional mockup unit and coupled with computer system emulators that run the whole TE management system software using the overarching simulator.

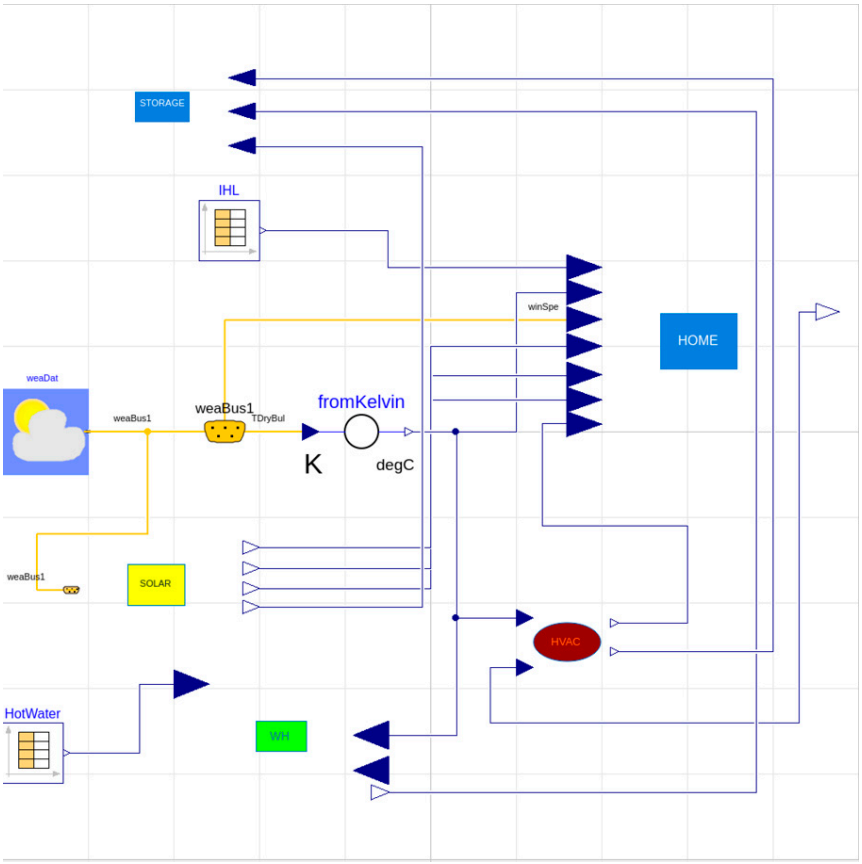


Figure 7. Simulation models of building dynamics and the devices on Dymola User Interface.

3.5. Distributed Architecture

The configuration of an example distributed control topology used on the testbed is illustrated in Figure 8. In decentralized structure, all the subsystems conduct local optimizations based on the information (e.g., price single or some value signal) published by the control center, regardless of the actions of other subsystems. Unlike the centralized control system, consumers in the decentralized control system do not have to release their cost function/constraint, which considerably protects consumer privacy and reduces the communication burden. Another advantage of the distributed structure is that the computational burden of the control center is significantly relieved.

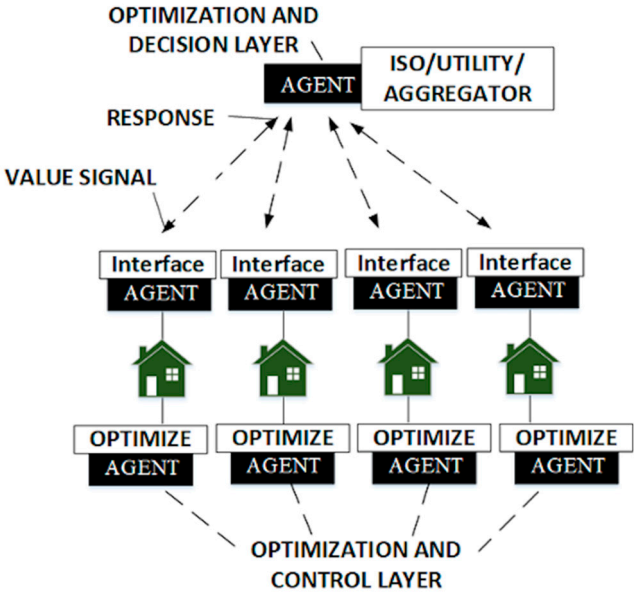


Figure 8. Configuration of distributed control topology.

4. Results

The testbed was deployed on a server with 128 physical cores and 256 GB of memory. The initial test setup included 100 homes that were orchestrated by four aggregators; the homes were equally distributed to aggregators. Aggregators were associated with two utilities, and there was one ISO. A virtual machine (VM) was associated with each component running its corresponding control software agents. Also, four VMs were associated with load, aggregator, utility, and ISO APIs. All the communication channels were replicated within the testbed based on Figures 3 and 4 using simulated network cards. During the test, a price signal and desired load shape were passed down from ISO to utility, then to aggregator, and then to homes. After the homes received the signal, each house optimized its resources and sent a load forecast for the next 24 h back to the aggregator and all the way up to the ISO.

Verification and validation runs were conducted to assert that all components and APIs were communicating necessary information between the components. The verification and validation experiments also served as a testing practice that cleaned bugs from the code. Another benefit of the testbed was that it also served as a developer environment in which developers continued to revise the software agents for which the immediate outcomes could be observed and tested on all components with full functionality. This is important because, as mentioned, testing the whole software stack and their inter-functionality together requires physical computers with network connections. Even then, the scalable software performance test was impossible, since the software had to interact with real devices and the environmental conditions. The software system was seamlessly deployed, and the testbed could scale the number of buildings up at will, being only limited by the computational resources.

The testbed was run for 24 h starting at midnight on 15 July 2013, inputting the real weather conditions from the Nashville, Tennessee area. A 24-h testing time was accomplished in about ~25 h of real-time. The testbed results were presented after a 1000-s warm-up period. The warm-up period was conducted to ensure that all initial authentication and network handshakes of the software agents and web APIs, as well as the simulation models, had been accomplished. These handshakes and the exchange of authentication steps were implemented to address the privacy and security of the TE system under test (see Zia et al. [56] and Hassan et al. [57] for more sophisticated security and privacy of TE systems). Figure 9a presents the setpoints decided by the optimizer and the evolution of room temperature over time for a single home. *Tmin* and *Tmax* values were acquired from the

HVAC schedule that would be entered by users in real equipment, but were inputted to the simulation in this case. They serve as the upper and lower bounds of the comfort settings for cooling. Figure 9b also shows the room temperatures for all homes representing various behavioral patterns.

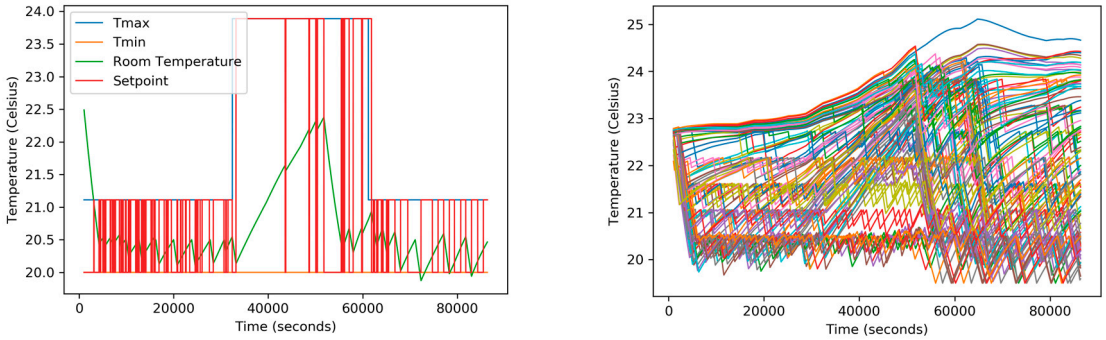


Figure 9. (**a**) Room temperature, setpoint, Tmin and variable Tmax for a single home over time; (**b**) Room Temperature over time for all 100 homes.

As observed in Figure 9, the setpoint is orchestrated based on the HVAC schedules and the current room temperature over time. Similarly, Figure 10a,b present the hot water temperature in the water heater tank for a single home and the water temperatures in all homes, respectively. The hot water temperatures varied between 48 °C and 60 °C degrees, as decided by the water heater agent and the optimizer.

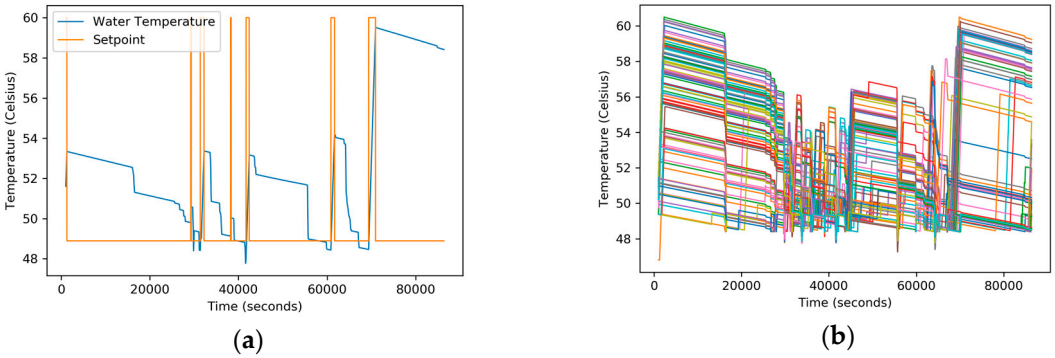
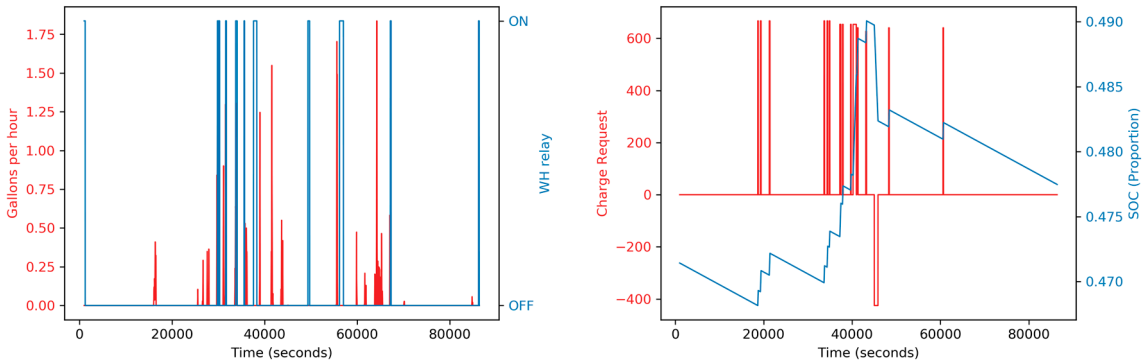


Figure 10. (**a**) Hot water temperature and water heater setpoint over time for a single home; (**b**) Hot water temperatures over time for all 100 homes.

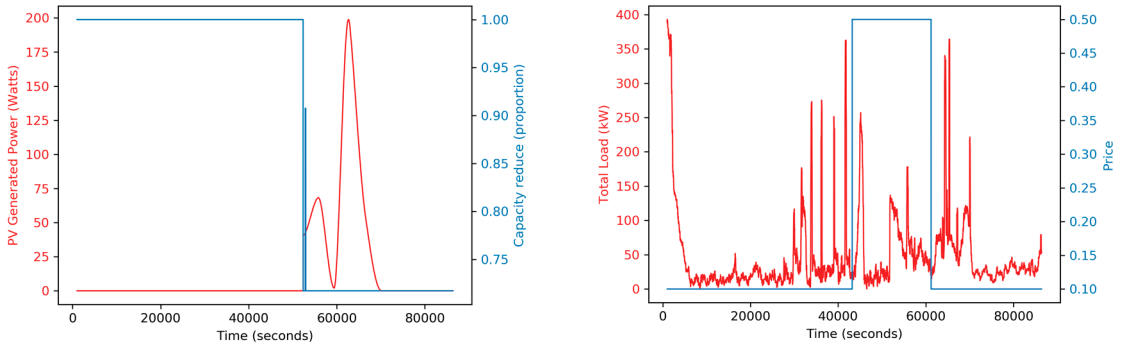
Figure 11a presents the on/off decisions for the water heater relative to the hot water consumption for a single home. Regarding the energy storage, Figure 11b shows how the charge requests received from the optimizer interact with the SOC of the energy storage system for a home with positive charge and negative charge requests.



(**a**) (**b**)

Figure 11. (**a**) Hot water consumption over time in red (left *y*-axis) and water heater relay over time in blue (right *y*-axis); (**b**) Charge request over time in red (left *y*-axis) and SOC for the energy storage system in blue (right *y*-axis).

PV models receive a capacity-reduce decision from the optimizer that coordinates the percentage of the potential solar power that should be generated and used/sent to the grid. Figure 12a shows the generated PV power and the capacity-reduce decisions for a single home.



(**a**) (**b**)

Figure 12. (**a**) Solar power generated by the PV in red (left *y*-axis) and capacity reduce decision that is received from the optimizer in blue (right *y*-axis) for a single home; (**b**) Total load over time in red (left *y*-axis) and the price signal over time in blue (right *y*-axis).

Optimizers impose a predicted load shape to the homes based on the total demand, comfort settings of the HVACs, and the price signal. Figure 12b shows the total load of 100 homes over time against the price signal that was realized. The spikes in the load shape were caused by the hot water consumption patterns that occurred during roughly the same timeframe for each home. When the price signal was higher, the PV generation was prioritized for some homes and the total load was decreased. Figure 10b also shows that hot water was preheated before the price signal went up, and during midday, when the prices were up, so it was unlikely to turn the water heaters on.

5. Discussion

This paper presents the implementation and evaluation of a simulation testbed for supporting the design and development of TE management systems. The proposed testbed allows users to flexibly reconfigure the system network and model the behaviors of houses and appliances. Moreover, the proposed testbed improves previous studies by using the detailed models of responsive domestic appliances and DERs, which yields a more accurate response toward different environmental parameters and control commands. Furthermore, bidirectional communication

among agents are enabled so that the information exchange behaviors and delays in TE management systems are also tested.

The demonstration and results presented herein show how the transactive software system as it will be deployed interacts with coupled mathematical models of the physical environment and the devices that are solved dynamically. This capability allows for the evaluation of communication among interacting agents at subminute (and even subsecond) timescales using simulated network cards and operating system network protocols, thereby allowing detailed assessments of the communication infrastructure to be made that will be necessary for real-world deployment. Aside from communication and control mechanism testing, the testbed also served as a verification tool that helped developers remove bugs and perform unit tests on the functionality of the overall TE management system after the individual software agents were revised. Considering the number of components in the TE management system software developed in this work, this capability decreased the software development cost significantly by providing a testbed that also serves as a development environment. The proposed technology allowed high numbers of testing hours to be accumulated, which had an additional positive effect on reliability (Nutaro and Ozmen [39] provides further discussions).

In conclusion, with the support of the proposed simulation testbed, researchers can not only test the performance of various existing control mechanisms, but can also validate the assumptions for future TE management system designs for scales of large neighborhoods and cities. The proposed approach is being parallelized using HPC platforms. When the scalability of the testbed is achieved on modern HPC systems, the number of compute cores available for the testbed will be on the order of 10^4. This will allow more sophisticated control mechanisms and optimization algorithms to be tested at a larger scale.

Supplementary Materials: The following are available online at www.mdpi.com/2071-1050/12/11/4402/s1.

- The nomenclature of the equations.
- The information to gain access to the code repositories.

Author Contributions: O.O. and J.N. developed the testbed and conducted the analysis. J.M. and O.O. developed the device and building models. M.S., H.Z., and L.R. developed the software agents and the cloud APIs. X.K. and M.S. developed the optimization algorithms. P.I., J.D., F.L, and T.K contributed to the model development. All authors contributed significantly to the manuscript and the technical merits of this work. All authors have read and agree to the published version of the manuscript.

Funding: Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy.

Acknowledgments: This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the US Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chiang, M.-C.; Yeh, T.-C.; Tseng, G.-F., A QEMU and SystemC-based cycle-accurate ISS for performance estimation on SoC development. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2011,** 30, (4), 593-606.
2. Monton, M.; Portero, A.; Moreno, M.; Martinez, B.; Carrabina, J., Mixed SW/SystemC SoC Emulation Framework. In *2007 IEEE International Symposium on Industrial Electronics*, IEEE: Vigo, Spain, 2007.
3. Kurimoto, Y.; Fukutsuka, Y.; Taniguchi, I.; Tomiyama, H., A hardware/software cosimulator for Network-on-Chip. In *2013 International SoC Design Conference (ISOCC)*, IEEE: Busan, South Korea, 2013.

4. Shankar, S. S.; Desai, K.; Dutta, S.; Chetwani, R. R.; Ravindra, M.; Bharadwaj, K. M., Mission critical software test philosophy a SILS based approach in Indian Mars Orbiter Mission. In *International Conference on Contemporary Computing and Informatics*, IEEE: Mysore, India, pp 414-419.
5. Anwar, Z., Reusable Device Simulation Models for Embedded System Virtual Platforms URL: https://www.design-reuse.com/articles/24201/embedded-system-virtual-platforms.html. *Accessed: 25 May 2020*.
6. Ma, O.; Alkadi, N.; Cappers, P.; Denholm, P.; Dudley, J.; Goli, S.; Hummon, M.; Kiliccote, S.; MacDonald, J.; Matson, N.; Olsen, D.; Rose, C.; Sohn, M. D.; Starke, M.; Kirby, B.; O'Malley, M., Demand Response for Ancillary Services. *IEEE Transactions on Smart Grid* **2013,** 4, (4), 1988-1995.
7. Olsen, D. J.; Matson, N.; Sohn, M. D.; Rose, C.; Dudley, J.; Goli, S.; Kiliccote, S.; Hummon, M.; Palchak, D.; Denholm, P.; Jorgenson, J. *Grid Integration of Aggregated Demand Response, Part 1: Load Availability Profiles and Constraints for the Western Interconnection*; Office of Scientific and Technical Information (OSTI): Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States); National Renewable Energy Lab.(NREL), Golden, CO (United States), 2013.
8. Ma, O.; Cheung, K. *Demand Response and Energy Storage Integration Study*; Office of Scientific and Technical Information (OSTI): National Renewable Energy Lab.(NREL), Golden, CO (United States), 2016.
9. Alkadi, N. E.; Starke, M. R.; Ma, O. *Assessment of Industrial Load for Demand Response across Western Interconnect*; Office of Scientific and Technical Information (OSTI): Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2013.
10. Abrishambaf, O.; Lezama, F.; Faria, P.; Vale, Z., Towards transactive energy systems: An analysis on current trends. *Energy Strategy Reviews* **2019,** 26, 100418.
11. Nizami, M. S. H.; Haque, A.; Nguyen, P. H.; Hossain, M. J., On the application of Home Energy Management Systems for power grid support. *Energy* **2019,** 188, 116104.
12. Kassai, M., Prediction of the HVAC energy demand and consumption of a single family house with different calculation methods. *Energy Procedia* **2017,** 112, 585-594.
13. Silva, M.; Morais, H.; Vale, Z., An integrated approach for distributed energy resource short-term scheduling in smart grids considering realistic power system simulation. *Energy Conversion and Management* **2012,** 64, 273-288.
14. Kassai, M.; Poleczky, L.; Al-Hyari, L.; Kajtar, L.; Nyers, J., Investigation of the energy recovery potentials in ventilation systems in different climates. *Facta Universitatis, Series: Mechanical Engineering* **2018,** 16, (2), 203-217.
15. Javed, A.; Larijani, H.; Ahmadinia, A.; Emmanuel, R.; Mannion, M.; Gibson, D., Design and Implementation of a Cloud Enabled Random Neural Network-Based Decentralized Smart Controller With Intelligent Sensor Nodes for HVAC. *IEEE Internet of Things Journal* **2017,** 4, (2), 393-403.
16. Namdeo, D. S.; Pawar, V. R., IoT based smart home for power & security management. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE: Madurai, India, 2017.
17. Melton, R. B. *Gridwise transactive energy framework*; Office of Scientific and Technical Information (OSTI): Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2013.
18. Liu, Z.; Wu, Q.; Huang, S.; Zhao, H., Transactive energy: A review of state of the art and implementation. In *2017 IEEE Manchester PowerTech*, IEEE: Madurai, India, 2017.
19. Hammerstrom, D. J.; Ambrosio, R.; Carlon, T. A.; DeSteese, J. G.; Horst, G. R.; Kajfasz, R.; Kiesling, L. L.; Michie, P.; Pratt, R. G.; Yao, M.; Brous, J.; Chassin, D. P.; Guttromson, R. T.; Katipamula, S.; Le, N. T.; Oliver, T. V.; Thompson, S. E. *Pacific Northwest GridWise? Testbed Demonstration Projects; Part I. Olympic Peninsula Project*; Office of Scientific and Technical Information (OSTI): Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2008.
20. Thomas, P. R.; Walker, T. J., Demonstration of Community Energy Storage fleet for load leveling, reactive power compensation, and reliability improvement. In *2012 IEEE Power and Energy Society General Meeting*, IEEE: San Diego, CA, USA, 2012.
21. Agalgaonkar, Y. P.; Hammerstrom, D. J., Evaluation of Smart Grid Technologies Employed for System Reliability Improvement: Pacific Northwest Smart Grid Demonstration Experience. *IEEE Power and Energy Technology Systems Journal* **2017,** 4, (2), 24-31.
22. Pratt, A.; Krishnamurthy, D.; Ruth, M.; Wu, H.; Lunacek, M.; Vaynshenk, P., Transactive Home Energy Management Systems: The Impact of Their Proliferation on the Electric Grid. *IEEE Electrification Magazine* **2016,** 4, (4), 8-14.

23. Hu, Q.; Li, F., Hardware Design of Smart Home Energy Management System With Dynamic Price Response. *IEEE Transactions on Smart Grid* **2013,** 4, (4), 1878-1887.
24. Stecklein, J. M.; Dabney, J.; Dick, B.; Haskins, B.; Lovell, R.; Moroney, G. In *Error cost escalation through the project life cycle*, 14th Annual International Symposium, Toulouse, France, 2004; Toulouse, France, 2004.
25. Gkatzikis, L.; Koutsopoulos, I.; Salonidis, T., The role of aggregators in smart grid demand response markets. *IEEE Journal on selected areas in communications* **2013,** 31, (7), 1247-1257.
26. Wen, Y.-J. *Rapid-prototyping control implementation using the building controls virtual test bed*; Philips Res. North Amer.: Briarcliff Manor, NY, USA, 2011.
27. Huang, S.; Wang, W.; Brambley, M. R.; Goyal, S.; Zuo, W., An agent-based hardware-in-the-loop simulation framework for building controls. *Energy and Buildings* **2018,** 181, 26-37.
28. Gupta, D.; Vishwanath, K. V.; McNett, M.; Vahdat, A.; Yocum, K.; Snoeren, A.; Voelker, G. M., DieCast. *ACM Transactions on Computer Systems* **2011,** 29, (2), 1-48.
29. Yoginath, S. B.; Perumalla, K. S.; Henz, B. J., Virtual machine-based simulation platform for mobile ad-hoc network-based cyber infrastructure. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* **2015,** 12, (4), 439-456.
30. QEMU, the fast processor emulator. URL: http://www.qemu.org. *Accessed: 25 May 2020*.
31. Mueller, W., 1st International QEMU Users' Forum. In Citeseer: Grenoble, France, 2011.
32. Weingärtner, E.; Schmidt, F.; Heer, T.; Wehrle, K., Synchronized network emulation. *ACM SIGMETRICS Performance Evaluation Review* **2008,** 36, (2), 58.
33. Nutaro, J. J., Building Software for Simulation. In John Wiley & Sons, Inc.: Hoboken, NJ, 2010.
34. Nutaro, J., An extension of the OpenModelica compiler for using Modelica models in a discrete event simulation. *SIMULATION* **2014,** 90, (12), 1328-1345.
35. Nutaro, J.; Kuruganti, P. T.; Miller, L.; Mullen, S.; Shankar, M., Integrated Hybrid-Simulation of Electric Power and Communications Systems. In *2007 IEEE Power Engineering Society General Meeting*, IEEE: Tampa, FL, USA, 2007.
36. Nutaro, J.; Kuruganti, P. T.; Protopopescu, V.; Shankar, M., The split system approach to managing time in simulations of hybrid systems having continuous and discrete event components. *SIMULATION* **2011,** 88, (3), 281-298.
37. Sydney, A.; Nutaro, J.; Scoglio, C.; Gruenbacher, D.; Schulz, N., Simulative Comparison of Multiprotocol Label Switching and OpenFlow Network Technologies for Transmission Operations. *IEEE Transactions on Smart Grid* **2013,** 4, (2), 763-770.
38. Nutaro, J. A., URL: https://web.ornl.gov/~nutarojj/adevs/. *Accessed: 25 May 2020*.
39. Nutaro, J.; Ozmen, O. In *Using Simulation to Quantify the Reliability of Control Software*, Winter Simulation 2019, National Harbor, MD, 2019, 2019; IEEE: National Harbor, MD, 2019; pp 3267-3276.
40. Ozmen, O.; Nutaro, J. J.; Sanyal, J.; Olama, M. M. *Simulation-based Testing of Control Software*; Oak Ridge National Laboratory: Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), 2017.
41. FMI, functional mock-up interface. URL: https://www.fmi-standard.org. *Accessed: 25 May 2020*.
42. Winkler, J.; Munk, J.; Woods, J., Effect of occupant behavior and air-conditioner controls on humidity in typical and high-efficiency homes. *Energy and Buildings* **2018,** 165, 364-378.
43. New Census Bureau Data Highlight Changes in Housing Values Through 2005. In *PsycEXTRA Dataset*, American Psychological Association (APA): 2006.
44. Cui, B.; Dong, J.; Munk, J. D.; Mao, N.; Kuruganti, T. *A simplified regression building thermal modelling method for detached two-floor house in US*; Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States): 2018.
45. ASHRAE Handbook—1981 Fundamentals. In *Building Services Engineering Research and Technology*, SAGE Publications: Atlanta, GA, 1981; pp 193-193.
46. EnergyPlus Engineering Reference: the reference to EnergyPlus calculations. In US Department of Energy: 2010.
47. EnergyPlus+ version 8.7.0. URL: https://energyplus.net. *Accessed: 25 May 2020*.
48. BeOpt, version 2.7. URL: https://beopt.nrel.gov/home. *Accessed: 25 May 2020*.
49. Standard, A., Performance rating of unitary air-conditioning & air-source heat pump equipment. *AHRI Standard* **2008,** 210, 240.
50. Pallin, S.; Boudreaux, P.; Jo, S. J.; Perez, M.; Albaugh, A., Simulations of Indoor Moisture Generation in U.S. Homes. In *Advances in Hygrothermal Performance of Building Envelopes: Materials, Systems and Simulations*, ASTM International: 2017; pp 261-290.

51. Hendron, R.; Burch, J., Development of Standardized Domestic Hot Water Event Schedules for Residential Buildings. In *ASME 2007 Energy Sustainability Conference*, ASMEDC: Long Beach, California, USA, 2007.
52. Liu, G.; Starke, M.; Xiaohu, Z.; Tomsovic, K., A MILP-based distribution optimal power flow model for microgrid operation. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, IEEE: Boston, MA, USA, 2016.
53. Huld, T.; Müller, R.; Gambardella, A., A new solar radiation database for estimating PV performance in Europe and Africa. *Solar Energy* **2012,** 86, (6), 1803-1815.
54. Wetter, M.; Zuo, W.; Nouidui, T. S.; Pang, X., Modelica buildings library. *Journal of Building Performance Simulation* **2014,** *7*, (4), 253-270.
55. Brück, D.; Elmqvist, H.; Mattsson, S. E.; Olsson, H. In *Dymola for multi-engineering modeling and simulation*, 2nd International Modelica Conference, Oberpfaffenhofen, Germany, 2002, 2002; Oberpfaffenhofen, Germany, 2002.
56. Zia, M. F.; Benbouzid, M.; Elbouchikhi, E.; Muyeen, S. M.; Techato, K.; Guerrero, J. M., Microgrid Transactive Energy: Review, Architectures, Distributed Ledger Technologies, and Market Analysis. *Ieee Access* **2020,** 8, 19410-19432.
57. Hassan, N. U.; Yuen, C.; Niyato, D., Blockchain Technologies for Smart Energy Systems: Fundamentals, Challenges, and Solutions. *IEEE Industrial Electronics Magazine* **2019,** 13, (4), 106-118.