ScorePlus: An Integrated Scalable Cyber-Physical Experiment Environment for Smart Grid

Song Tan*, Wen-Zhan Song†‡, Steve Yothment†, Junjie Yang‡ and Lang Tong§

*Department of Computer Science, Georgia State University, GA, USA
†Senior Electrical Engineer, Panasonic Automotive Systems of America, USA
‡School of Electric and Information Engineering, Shanghai University of Electric Power, China
§School of Electrical and Computer Engineering, Cornell University, NY, USA

stan.songwz@cs.gsu.edu, yothments@us.panasonic.com, yangjj@shiep.edu.cn, ltong@ece.cornell.edu

Abstract—Smart Grid is a complex cyber-physical system that modernizes the traditional electric power infrastructure by sensing, control, computation and communication. Validating the functionality, security and reliability of Smart Grid applications within such a system requires the modeling and emulation of both power networks and communication networks, as well as the interactions between them. In this paper, we present the design, implementation and evaluation of an integrated scalable cyber-physical experiment environment for Smart Grid, called ScorePlus. Compared with previous related works, ScorePlus fills the gap by: 1) Creating and integrating both software emulator and hardware testbed, such that they all follow the same architecture and interface, and the same Smart Grid application program can be tested on either of them without any modification; 2) Providing remote access to the hardware testbed such that users can configure physical devices of the hardware testbed through Internet; 3) Supporting scalable distributed experiments such that multiple software emulators and hardware testbeds running at different locations are able to connect and form a larger Smart Grid system.

Index Terms—Cyber-Physical System, Smart Grid, Testbed.

I. INTRODUCTION

Smart Grid is a complex cyber-physical system that modernizes the traditional electric power infrastructure by sensing, control, computation and communication. The Smart Grid innovation brings a lot of new research challenges to the agenda, such as secure communication protocol, anomaly detection and identification, real-time pricing scheme, and renewable resource integration, etc. Due to the system complexity, typically the high cohesion of communication and power network, validating and demonstrating these new ideas are not easy jobs, especially in a lab environment [1].

The existing works to solve the problem generally fall into two categories: real hardware testbeds or software simulators. The real hardware testbeds achieve high fidelity by employing dedicated devices as integral parts of the testbeds. In [2], the Idaho National Lab incorporates the actual Smart Grid components including power generators, storage batteries, and substations to facilitate the cyber security research of power transmission in Smart Grid. In [3], Renewable Energy Laboratory in Greece set up a central-controlled microgrid testbed consisting of PV-panels, battery banks and inverters to investigate the renewable integration issues. Stanovich et al. in [4] and Hahn et al. in [5] employ field devices like Intelligent Electronic Devices (IEDs) and Programmable Logic Controllers (PLCs) within the testbeds, to achieve the hardware-in-the-loop testing. However, the substantial cost and resource needed to deploy these testbeds limits the repeatability of these efforts in a lab environment. Moreover, these testbeds cannot be accessed and shared remotely by the public research community and are difficult to scale when the test case becomes quite large. The software simulations, on the other hand, achieve much better availability, usability and scalability. It usually combines multiple software simulation tools, typically a network simulator and an electric power grid simulator, and a middleware is used to exchange messages periodically and synchronize all the simulators. In [6], Godfrey et al. simulate the Smart Grid system using a network simulator NS2 and a power network simulator OpenDSS. In [7], the PowerWorld simulator and OPNET network simulator are employed together to evaluate the SCADA system security. In [8], NS2 and TH-STBLT power simulator are employed to study the data compensations in Smart Grid. Within software simulation platforms, the models of various Smart Grid objects can be easily scaled and statistically analyzed. However, since software simulation typically abstracts the operating system, communication protocols and power dynamics into various mathematical simulation models, it can only duplicate the behavior and structure of the system, but not the execution environment of Smart Grid applications. Therefore, a much ideal approach is to combine the merits from both hardware and software platforms such that the users may examine the performance of Smart Grid applications under realistic communication and computation constraints in hardware platform, while evaluating the corresponding scalability in software platform at the same time.

In light of the above issues and our previous works in [9], through this paper, we present ScorePlus, an integrated and scalable experimental environment whose design framework and major components can be extended to build other cyber-physical testing systems in general. ScorePlus fills the gap by following the key features:

- **Integrated experiment environment:** ScorePlus includes both software emulator and hardware testbed. The software emulator emulates the communication and energy behavior of Smart Grid entities. The hardware
testbed hosts physical devices, such as micro renewable energy generators (solar panel, wind turbine), energy demanders, storages, and topology switches, which are all connected by power network and communication network in a physical environment. Both the software emulator and hardware testbed follow the same architecture and interface, and the same Smart Grid application program can be tested on either of them without any modification. With this integration, researchers may examine the performance of Smart Grid applications under realistic communication and computation constraints in hardware testbed, while evaluating the corresponding scalability in software emulator at the same time.

- **Remote access and configuration**: ScorePlus provides remote access to the hardware testbed such that users can configure physical devices of the hardware testbed through Internet. To the best of our knowledge, our work is the first Smart Grid hardware testbed that supports remote access and configuration. This feature will greatly increase the accessibility of the experiment environment, and benefit researchers from a wide range of institutions constrained by the unavailability of the hardware testbed.

- **Scalable**: ScorePlus supports scalable distributed experiments such that multiple software emulators and hardware testbeds running at different (local or remote) locations are able to connect and form a larger Smart Grid system. This makes it possible to scale up experiments without being constrained by the computation resources of a single software emulator server or the number of physical devices of a single hardware testbed.

**II. OVERALL SYSTEM DESIGN**

Figure 1 demonstrates the overall architecture of our platform, ScorePlus consists of Graphical User Interface (GUI), software emulator, and hardware testbed. The GUI connects with software emulator and hardware testbed remotely through Internet. Since the software emulator and the hardware testbed expose the same architecture and interfaces to the GUI, the user can run the same Smart Grid application test case on either of them without any migration issue. Meanwhile, by integrating the communication network and power network from both software emulator and hardware testbed, multiple software emulators and hardware testbeds are able to connect and interact through Internet. Upon specifying the connection interfaces between each other, communication and power flow could be established between each individual node, such as a virtual demander in software emulator and a real supplier in hardware testbed.

**III. SOFTWARE EMULATOR**

As shown, the software emulator consists of Service Layer, Virtual nodes, Linux Ethernet Bridging, Communication Module, and Power Module. The Service Layer is essentially a socket server that provides various event handlers to the formatted messages from external interacting system. It is responsible for initializing the emulation case, collecting and forwarding the system request, and managing multiple emulation sessions, etc.

![Fig. 2. Design of Software Emulator](image)

**A. Virtual Nodes: Light Weighted Virtualization**

The emulation feature of software emulator is implemented using Linux network namespace techniques, which is a recent light weighted paravirtualization technique supported by mainstream Linux kernel. By calling the clone() system call, each created virtual node can have its own instance of Linux OS network stack and process space while sharing the same local file systems and hardware with other virtual nodes. From the perspective of codes running inside, each virtual node is just another piece of hardware platform controlled by Linux OS. Therefore, the virtual nodes can directly execute unmodified Smart Grid application codes from physical Linux-embedded devices, and vice versa. Figure 3 illustrates the software emulator scalability that about 180 virtual nodes can be created on a 64 bits HP desktop with Pentium(R) Dual-Core CPU E5700 @ 3.00GHz and 4GiB memory.
B. Linux Ethernet Bridging and Communication Module

All the virtual nodes in the emulator are equipped with virtual Ethernet interfaces and they are linked by Linux Ethernet Bridging. This approach provides underlying communication capabilities between each virtual node. Based on Linux Ethernet Bridging, the communication module in software emulator provides comprehensive support of various wired and wireless communication network models. All the models are simply the manipulations of the underlying communication infrastructure. More importantly, virtual Ethernet interfaces can be directly mapped to a physical Ethernet interface on the emulation host, such that all the traffic passing through that physical port can be transferred to the emulation environment. This enables the interactions between the software emulation environment with outside physical networks.

We employ the above key feature to achieve the integration of communication networks between the software emulator and hardware testbed, which will be presented in Section V.

C. Power Module

The power module in software emulator emulates the power flows analysis within Smart Grid and also provides implementations of pre-defined energy models. The power module receives initial power network topology, energy model configuration information and the connection interface information from service layer to formulate the power network model.

1) Power Network Model: General Description

The power network model is a DC current model to emulate DC power flow analysis. Assume a power grid is composed of \( n \) nodes and \( b \) branches. Since the power network dynamics is subject to Kirchhoff’s current and voltage laws (KCL and KVL), in order to calculate the voltages of all nodes, we apply nodal analysis to the grid and get the linear equations for the whole system:

\[
AV = I
\]  

(1)

where coefficient matrix \( A \) is the \((n - 1) \times (n - 1)\) reduced nodal admittance matrix since we have chosen a reference node. Let \( Nbr(i) \) represents the neighbour set of node \( i \) in the power network, we get:

\[
a_{ij} = \begin{cases} 
    \sum_{s \in Nbr(i)} g_{is} & i = j, \\
    -g_{ij} & j \in Nbr(i) \\
    0 & \text{otherwise}
\end{cases}
\]  

(2)

\( g_{ij} \) is the admittance between node \( i \) and node \( j \), \( V \) and \( I \) are the unknown node voltage vector and the known nodal current injection vector, respectively.

2) Power Network Model: Domain Decomposition

Power network is generally a network of loosely coupled sub power networks. Each sub network is a relatively independent partition of the whole energy system and only few in-between connection lines join them together. Inside each sub network, we divide the nodes into two sets:

- Internal nodes: nodes that only have connections with the nodes inside the same sub network.
- Boundary nodes: nodes that have connections with the nodes in other sub networks.

The architecture of the power network is illustrated in Figure 4. Based on the previous analysis, we apply the Schur complement domain decomposition method [10] to our power network model. Specifically, suppose there are \( k \) sub networks, by grouping the internal nodes of each sub network and putting all the boundary nodes of the network in the back, we formulate the nodal analysis model for the whole power network as the following:

\[
\begin{bmatrix}
Y_{A_1 A_1} & 0 & \cdots & 0 & Y_{A_1 B} \\
0 & Y_{A_2 A_2} & \cdots & 0 & Y_{A_2 B} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & Y_{A_k A_k} & Y_{A_k B} \\
Y_{B_1 A_1} & Y_{B_1 A_2} & \cdots & Y_{B_1 A_k} & Y_{B_1 B} \\
Y_{B_2 A_1} & Y_{B_2 A_2} & \cdots & Y_{B_2 A_k} & Y_{B_2 B} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
Y_{B_k A_1} & Y_{B_k A_2} & \cdots & Y_{B_k A_k} & Y_{B_k B} \\
\end{bmatrix}
\begin{bmatrix}
V_{A_1} \\
V_{A_2} \\
\vdots \\
V_{A_k} \\
V_{B_1} \\
V_{B_2} \\
\vdots \\
V_{B_k} \\
\end{bmatrix}
= 
\begin{bmatrix}
I_{A_1} \\
I_{A_2} \\
\vdots \\
I_{A_k} \\
I_{B_1} \\
I_{B_2} \\
\vdots \\
I_{B_k} \\
\end{bmatrix}
\]  

(3)

Notice that \( B \) is the set of all boundary nodes in the whole network, consisting of \( B_1, B_2, ..., B_k \). Therefore, \( Y_{A_i B} \) only has non zero entries in its submatrix \( Y_{A_i B_i} \), for all \( i = 1, 2, ..., k \).

From (3), if the voltages for boundary nodes set \( V_B \) is known, then the voltages for the nodes in each sub network can be calculated as the following:

\[
Y_{A_i A_i} V_{A_i} = I_{A_i} - Y_{A_i B} V_B, \forall i \in \{1, 2, ..., k\}.
\]  

(4)
Meanwhile, if we keep the corresponding part for the boundary node set $B$ in equation (3), we can get:

$$
\tilde{Y}_{BB}V_B = \tilde{I}_B
$$

(5)

where

$$
\tilde{Y}_{BB} = Y_{BB} - \sum_{i=1}^{k} Y_{BA}Y_{A,A_i}^{-1}Y_{A,B}
$$

(6)

$$
\tilde{I}_B = I_B - \sum_{i=1}^{k} Y_{BA}Y_{A,A_i}^{-1}I_{A_i}
$$

(7)

Define

$$
x_i = Y_{BA}Y_{A,A_i}^{-1}Y_{A,B}
$$

(8)

$$
y_i = Y_{BA}Y_{A,A_i}^{-1}I_{A_i}
$$

(9)

for all $i = 1, 2, ...k$. Notice that $x_i$ and $y_i$ only requires local information for sub system $i$.

We employ the above key feature to achieve the integration of power networks between the software emulator and the hardware testbed, which treats each of them as a sub network of the whole power network. The details for that would be given in section V.

IV. HARDWARE TESTBED

Figure 5 shows the design of our hardware testbed, which follows the same architecture as the software emulator. Each node in the hardware testbed is a physical energy devices emulating an energy entity in Smart Grid system. Through communication interface and power interface, all the energy devices are connected by communication network and power network.

![Design of Hardware Testbed](image)

Fig. 5. Design of Hardware Testbed

A. Overview of Energy Devices

The hardware testbed is composed of the following energy devices to emulate the energy entities in a Smart Grid system:

- 1 Supplier (This device emulates a general power generation. Output to the Smart Grid is up to 200mA. The Smart Grid voltage is typically 2.5V, but it may be higher or lower when source and load are unbalanced.)
- 5 Solar Panel Controller (Output is up to 30mA each, depending on light intensity.)
- 5 Wind Turbine Controller (Output is up to 30mA each, depending on wind intensity.)
- 5 Storage (Capacitor bank with electronics. It can source or sink up to 50mA depending on the demands of the Smart Grid system.)
- 15 Demander (A load only, drawing up to 20mA each from the Smart Grid.)
- 5 Topology Switch (Has 6 ports and can switch current flow in multiple ways. Must be able to handle up to 300mA on all ports.)
- 1 Interface device, which serves as the energy tunnel when the hardware testbed is connected with the software emulator.

Figure 6 shows various energy devices, solar panel and wind turbine in use. The LCD display shows the current drawn and sourced by each energy device except the one in Topology Switch Device, which displays the connection status between all the 6 ports.

B. Energy Device Design Details

Each of the above energy device includes three boards: a Beagleboard [11], a Telosw board [12] and an Energy board. Figure 7 shows the process of remote access and configuration of these energy devices from GUI by the user. The user specify the hardware configurations from the GUI and send requests to the Service Layer as formatted messages. Upon receiving and mapping these messages, the Service Layer forwards the corresponding request to the Beagleboard of the designated energy device. Then the control program in that Beagleboard will communicate with its Telosw board, which ultimately interacts with the Energy board and put those configurations into effect, such as load of the demander, output of the renewable controller and connection status of the topology switch, etc. Likewise, the status of the energy devices are periodically queried and reported to GUI backward. The roles of the above three boards are summarized as the following:

1) Beagleboard

- Interact with the Service Layer and control the behavior of each energy device correspondingly.
- Provide Linux-based environment in each energy device to test the Smart Grid applications.
- Enable the communications between each energy device, through both wired and wireless networks.

![Remote access and configuration of energy devices](image)

Fig. 7. Remote access and configuration of energy devices
2) *Telosw board*

- Adjust of smart resistors to emulate power profiles.
- Monitor/measure power generation and consumption rate of each energy device.
- Provide I/O to mating Energy board for LED and display.

3) *Energy board*

- Provide different DC current circuit to fulfill the corresponding power requirements of different devices.

Figure 8 shows the architecture of a Solar Panel Controller. The Beagleboard provides physical communication interfaces, such as the Ethernet port and mini USB to connect the wireless radio. It is connected with TelosW board through USB cable such that the programs running in the Beagleboard can directly access and control the energy profile through the TelosW. The TelosW is mated to the Energy board, which are all enclosed in a black plastic container. The port at the top of the container is connected with a physical solar panel. There are three ports at the bottom of the container. The left one serves as the power source of the device and the right one is the actually power interface for Smart Grid system. The middle port is connected to the Ground.

Figure 9 shows the schematics for the Energy board on Solar Panel Controller. An analog voltage from the TelosW board indicates the amount of current output to the Smart Grid that is desired (0 to 2.0V corresponds to 0 to 30mA of desired output). An analog voltage to the TelosW board indicates the measured current output (0 to 2.0V corresponds to 0 to 30 mA of actual output current). The Solar Cell input to the Energy Control board will have a capacitor of 10,000uF (C6) for the purpose of temporarily storing energy from the generating source. 0.4 times the voltage on this capacitor will be supplied to the TelosW board for the purpose of indicating energy capability of the Solar Cell. The same device can be used for wind turbine control. Figure 10 shows a sample current output of a Solar Panel Controller device when the solar panel is exposed to a 50W 3-level light intensity lamp over time.

C. *Power Network in Hardware Testbed: Dynamic Topology Configuration*

The power network in hardware testbed connects all the energy devices through power interfaces through power cables. To facilitate the dynamic configurations of various power network topologies, we design the Topology Switch device as Figure (6d), which serves as the current hub for all the other energy devices. The Topology Switch can be accessed
and configured dynamically through TelosW by the programs running inside the mated Beagleboard, such that the connection status between the 6 ports can be changed accordingly. Different power network topologies can be set up based on the users’ requirements. The connection status is also visualized through the LCD. The Energy board of Topology Switch is shown in Figure 11.

**D. Communication Network in Hardware Testbed: Wire and Wireless Network**

The communication network in hardware testbed are set up to emulate the wire and wireless communication in Smart Grid. The Generic Routing Encapsulation (GRE) tunnels are created between each energy device above the underlying communication network, such that the physical communication network can be unified with the ones in software emulator. For wired network, we use Ethernet to connect each energy device.

For wireless network, we are employing the WISP-2 outdoor antenna from ALFA Network, Inc, which can be directly connected with Beagleboard through USB cable. WISP-2 is a Low-Cost IEEE 802.11n outdoor AP/CPE operating in 2.4GHz band that is also compliant with the standard IEEE 802.11b/g. In our indoor environment, the radio range of each node covers all the other nodes, so physically, all nodes can communicate with each other. To formulate a wireless mesh network with a specific topology, the iptables tool chain is employed within each energy device. Through iptables, we can filter the sent and received packets for each node, such that they can only communicate with the nodes as specified. Then the corresponding wireless network topology is formulated. The wireless network in our testbed is still under development and the preliminary performance testing result from iperf tool are listed in Table I.

![Fig. 11. Design of Energy board for Topology Switch](image)

<table>
<thead>
<tr>
<th>Table I</th>
<th>Performance of Wireless Network in Hardware Testbed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>hops</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>bandwidth (kbps)</strong></td>
<td>1.05</td>
</tr>
<tr>
<td><strong>jitter (ms)</strong></td>
<td>1.83</td>
</tr>
</tbody>
</table>

**V. Integrating Software Emulators and Hardware Testbeds**

ScorePlus supports scalable distributed experiments such that multiple software emulators and hardware testbeds running at different (local or remote) locations are able to connect and form a larger Smart Grid system. Here we only present the integration between software emulator and hardware testbed. Note integrations only among software emulators or only among hardware testbeds follow the same mechanism. In order to enable the interactions between the software emulator and the hardware testbed, both the communication network and power network within them should be integrated.

**A. Integrating the Communication Network**

The communication network emulated in software emulator runs in real time, so they can be connected to live physical networks. We build GRE tunnels between/among software emulation servers and the energy devices in hardware testbed. Figure 12 illustrates the details of how a virtual node in software emulator sends a packet to a physical node in hardware testbed. GRE tunneling is built between the software emulation server and the gateway machine of the hardware testbed. The actual IP address of the two are in domain 131.96.x.0/24. When the packet reaches the edge of the network in software emulator, which is the physical network interface of emulation server, the tunnel entry in routing table would enable the encapsulation of the packet with GRE header and the tunnel destination, such that they can reach the hardware testbed environment. Also, by using this approach, all the ip addresses in the experiment environment are in 10.0.x.x/16 domain, and the physical networks can be abstracted as needed for the experiments.

![Fig. 12. Communication between virtual node and real node through GRE tunneling](image)

Figure 13 presents the performance of integrated communication network by showing the time elapsed for Open Shortest Path First (OSPF) routing protocol to converge. We employ different numbers of hardware testbed nodes and record time under different network delays set up in software emulator. As shown in Figure 13, when hardware testbed node is integrated within the communication network of the software emulator (the number of real nodes from 0 to 2), there is a dramatic increase in elapsed time. This indicates the GRE tunneling does impose a significant overhead. When the network delay
B. Integrating the Power Network

The power network integration employs the domain decomposition model in Figure 4. Instances from either hardware testbeds or software emulators are treated as sub networks of whole power network.

Specifically, suppose there are already \( k \) connected instances of software emulators and hardware testbeds, and another instance \( k + 1 \) joins in run time. Also assume that the \( k + 1 \) instance connects with the instance set \( E \) directly, \( E \subseteq \{1, 2, ..., k\} \). Then the composition process for each computation host \( i \), \( i = 1, 2, ..., k, k + 1 \), is executed as the following:

- If \( i \in E \), then adjust the boundary node set \( B_i \) by adding the new boundary nodes connected with instance \( k + 1 \) and also adjust the internal node set \( A_i \) by removing the corresponding boundary nodes connected with instance \( k + 1 \). Compute \( x_i \) and \( y_i \) based on equations (8) and (9) respectively. Send the results to the coordinator host.
- The coordinator first reforms the boundary node set \( B \) by adding the new boundary nodes in \( B_i, i \in E \) and \( B_{k+1} \), then rebuilt \( Y_{BB} \) and \( I_B \) for the whole system. Secondly, it collects \( x_i \) and \( y_i \) from each host, and calculate \( V_B \) based on equations (5) (6) (7). Finally, it sends \( Y_{BB}V_B \) back to each host.
- Each host \( i \) receives \( Y_{BB}V_B \) from the coordinator and calculate \( V_{A_i} \) based on equation (4).

After the above process, each sub power network can set its updated status based on the calculated results. If the sub network is a software emulator, we can update the status in our programs directly to achieve the resulting effects. However, if the sub network is a hardware testbed, physical changes must be made. To this end, we employ our specifically designed Interface device, which is essentially a combined supplier and demander with large capacity. When the calculated result indicates the hardware testbed sub network is requesting power from outside domains, the Interface device is set as a power supplier to provide corresponding power to the testbed. When the calculated result indicates the hardware testbed sub network is providing power to outside domains, the Interface device is set as a power demander to absorb power from the testbed correspondingly.

When one sub network is disconnected with the rest of the system, the steps are similar with the above except that instead of adding boundary nodes to the boundary set, the coordinator will remove the boundary nodes related with the exiting instances.

VI. EVALUATION AND EXPERIMENTATION

In this section, we demonstrate and evaluate the capabilities of ScorePlus to support cyber-physical analysis in Smart Grid, particularly in Microgrid [13]. The future Smart Grid is expected to be an integration of Microgrids featured by localized power generation, storage and consumption [14]. Microgrid works as an independent local power system that has the flexibility to connect (connecting mode) and disconnect (islanding mode) from the main grid as needed in order to minimize the energy cost and maximize the grid stability [15]. Figure 14 illustrates a typical Microgrid structure. Note that in ScorePlus we use Topology Switch to serve as circuit breakers.

A. Experiment setup

The test case is created upon the AMI network test case in [16] and demand response model in [17]. We investigate the Microgrid demand response behavior in one virtual day. For each virtual time period \( h \) in the virtual day (2 real time seconds), \( h = 0, 1, 2, 3, ..., 23 \), each kind of nodes behaves as the following:

- **Supplier:** Providing power with maximum 2kw to the whole grid and broadcasting its hourly price [17] to all the topology switches. Calculating the bills for each demander based on the real time price and the collected energy consumption from Demanders.
- **Topology Switch:** Receiving the energy price information from the Supplier and then relay the messages to the demanders immediately.
- **Solar Panel:** When \( h \in [0, 5] \cup [19, 23] \), setting its maximum output to 0. When \( h \in [6 : 18] \), randomly setting its maximum output between 200w and 300w.
- **Wind Turbine**: Randomly setting its maximum output between 100w and 200w for each hour.
- **Storage**: Its capacity is set to 1kwh with initial energy 0.1kwh; sending its current energy residual to demanders and charges/discharges with maximum 100w as requested from Demanders.
- **Demander**: Either critical Demander or adjustable Demander. Critical Demander must work at 200w all the time. Adjustable Demander receives price from Topology switch and energy residual from Storage and adjust its setpoint based on the demand response model in [17]; sending its energy consumption to Supplier for billing.

An AMI communication network is set up between all the nodes, which consists of both Ethernet and IEEE 802.11 RF Mesh network with 54Mbps bandwidth, 0.1% packet loss rate. All the above node behaviors are implemented by the control programs running inside each node. Figure 15 shows Xterminal for a Demander node. We use command `ps aux` to check programs running inside. They include the OSPF routing process and the demander energy daemon process.

![Fig. 15. Smart Grid applications running in a Demander node](image)

Five Microgrids are created as in Table II, running on four software emulation servers and one hardware testbed respectively.

![TABLE II
NUMBER OF NODES IN EACH MICROGRID](image)

<table>
<thead>
<tr>
<th>Microgrid ID</th>
<th>Type</th>
<th>Demander</th>
<th>Supplier</th>
<th>Switch</th>
<th>Topology</th>
<th>Storage</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software Emulator</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>Hardware testbed</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>3</td>
<td>Software Emulator</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>4</td>
<td>Software Emulator</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>5</td>
<td>Software Emulator</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>150</td>
</tr>
</tbody>
</table>

### B. Islanding Mode

In islanding mode, each Microgrid is working independently and self-sustained. We first examine Microgrid 1 and 2 for comparison. Figure 16 shows the real-time energy price and the adjustable Demander energy consumption over time, one in software emulator and one in hardware testbed. We can see that the control program for the price responsive model in [17] tends to shift the energy consumption to the lower price period of the day. In addition, we can see the difference of the power profiles between the two adjustable Demanders. Theoretically, the two curves should be exactly the same since they are controlled by the same program as previously specified. However, since the software emulator is event-based processing, it can only update with a large discrete step size. As a comparison, due to the high ADC sampling rate of TelosW, the one in hardware testbed has the advantage of capturing the transient dynamics in a finer grain.

![Fig. 16. Real-Time Energy Price and Demander Energy Consumption](image)

Figure 17 shows the renewable share of total energy consumption over time in all five Microgrids. We can see that generally, when the real time energy price increases, the renewable share increases correspondingly. In addition, as the size of the Microgrid increases, the renewable share become less sensitive to the price fluctuations since more renewable energy surplus can be distributed and stored by the demand response process.

![Fig. 17. Renewable Share of Total Energy Consumption](image)

### C. Connecting Mode

In connecting mode, Microgrid is connected with each other. We first examine our implementation for the integrations between different Microgrids. The software emulation servers and hardware testbed are connected with each other through Internet. Note that we specifically design the five Microgrids in Table II, such that Microgrid 1 connected with 2 is the same as Microgrid 3, and Microgrid 3 with 4 is the same as Microgrid 5. The study case No. and the Microgrids connected are listed in Table III.

![TABLE III
CASES IN CONNECTING MODE](image)

<table>
<thead>
<tr>
<th>Case No.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microgrid connected</td>
<td>1,2</td>
<td>3</td>
<td>2,3</td>
<td>4</td>
<td>3,4</td>
<td>5</td>
<td>1,2,4</td>
</tr>
</tbody>
</table>
Figure 18(a) shows the wind turbine output in case 1 and 2. The two curves conform with each other and the integration between software emulator with hardware testbed preserves the power profiles accurately. Figure 18(b) shows the ping time between two farthest nodes and the time needed for OSPF routing to converge in case 3, 4, 5, 6, 7. It indicates that for communication network, the integrations with software emulators introduce minor delays but major delays with hardware testbed, which is about 0.57s difference in ping time.

In connecting mode, the influence between different Microgrids becomes extremely important since the fault or failure in one Microgrid could impact the stability of other Microgrids. To this end, we conduct contingency analysis for case 1 under topology attacks, in which an attacker intends to disrupt the normal physical conditions of power system through malicious control of Topology Switch. As illustrated by GUI in Figure 19, suppose an attacker injects a malicious control program to the Topology Switch n4 in house 3 and runs it at time h=7, such that it disconnects house 4 with house 5 as well as the hardware testbed. Only port 2, 3 and 4 are connected. In normal condition, the Topology Switch in house 3 is set such that port 1, 2, 3, 4, 5, 6 are all connected. So the power needed in house 3 can be supplied from house 4, house 5 and the power network hardware testbed. After manipulating the connection status in the Topology Wwitch, the previous balanced power flow in the network is changed to unbalanced one, leading to a dramatic increase in the power flow between house 1 and house 3 and 4 (indicated by the red bold line in GUI). Meanwhile, since the hardware testbed is not providing power to the software emulator any more but the power generation is not aware of the attack at the moment, the extra power flow would be inevitably forwarded to the Storage devices in hardware testbed, which results in a sudden peak charging rate at the storage device, as marked by the red circle. The charging rate will reach a sudden peak around h=7 but return a higher but relative stable level afterwards since more energy surplus is being forwarded to Storage device.

VII. CONCLUSION

In this paper, we present the design, implementation and evaluation of ScorePlus, an integrated scalable cyber-physical experiment environment for Smart Grid. Our platform provides an extendable design paradigm for the creation of general cyber-physical testbeds. Testing cases such as distributed control algorithms, demand responses, and cyber-physical security issues, can all be evaluated in our platform. Future work would investigate the design of more hardware components like Phaser Measurement Unit and the decrease in system integration overhead.

REFERENCES