SCORE: Smart-Grid Common Open Research Emulator

Song Tan, Wen-Zhan Song
Department of Computer Science
Georgia State University
Atlanta, GA, USA
stan6,wsong@cs.gsu.edu

Qifen Dong
College of Information Engineering
Zhejiang University of Technology
Hangzhou, China
qdong@cs.gsu.edu

Lang Tong
School of Electrical and Computer Engineering
Cornell University
Ithaca, NY, USA
ltong@ece.cornell.edu

***Abstract*—A Smart Grid is a digitally enabled electrical grid that equips with various embedded devices that can sense, communicate, compute and control. Validating, analyzing and evaluating new ideas and technologies in Smart Grid require the modeling and emulating of both communication network and power network, as well as the interactions between them. This paper presents the design, implementation and evaluation of Smart-Grid Common Open Research Emulator (SCORE), the *first* integrated Smart Grid *emulator* of both power and communication network. Comparing to the existing works of co-simulation of Smart Grid, SCORE (as an *emulator*) will significantly reduce the development and test time of new ideas, since the same application program running in SCORE can be easily ported to embedded devices with little or no modification. SCORE supports large-scale emulations since it can be run across multiple network connected computers.**

I. Introduction

A smart grid is an electrical grid that uses information and communications technology to gather and act on information, such as information about the behaviors of suppliers and consumers, in an automated fashion to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity. The implementation of smart grid requires solving a lot of challenging research issues, such as demand response, dynamic pricing, renewable resource integration, security control, sensing and automation. On the other hand, validating and evaluating those new ideas in a lab environment needs a good experimental platform, either a software simulator and a real testbed [1]. In the literature, some existing software simulation tools for smart grid adopt co-simulation strategy, which combines multiple simulation tools, typically a network simulator and a power grid simulator. A middleware needs to be used to exchange messages periodically and synchronize the two simulators. Godfrey et al. [2] simulate the Smart Grid using NS2 and OpenDSS and Lin et al. [3] employ NS2 and PSLF. These software simulation tools typically run on a single PC and abstract the operating system, communication protocols and power dynamics into simulation models for producing overall statistical analysis. Although they enable large scale evaluation of Smart Grid applications, the codes' migration of Smart Grid application from simulation platform to real embedded systems is a headache in software engineering practice. On the other hand, the real testbed approach can eliminate the code migration problems but often involves dedicated and specialized hardware devices and resources, which are not readily available in large scale in an academic lab environment. Stimoniaris et al. [4] set up a central-controlled microgrid testbed consisting of PV-panels, battery banks and inverters to investigate the proposed Smart Grid topologies. Song et. al. [1] designed SmartGridLab testbed that consists of intelligent power switch, power generator, renewable energy sources, smart appliances, and power meter. The intelligent power switch can dynamically configure power grid topology.

In this paper, we present a Smart-grid Common Open Research Emulator (SCORE) to bridge the gap between simulator and testbed. It is built upon CORE [5], an open source communication network emulator from Naval Research Laboratory. SCORE emulates both information networks and power grid and their integration. The same application program running in SCORE can be easily ported to embedded devices in a real smart grid with little or no modification. In the design of SCORE, incremental updating algorithms have been employed to support highly efficient and scalable power grid emulation. SCORE supports large-scale emulations and can be run across multiple computers in the Internet. To our best knowledge, we believe that SCORE is the first emulation platform for Smart Grid. SCORE has the following key features:

1) **Fidelity:** SCORE provides an integrated power and communication network emulation environment to capture the dc power flow analysis in real time for Smart Grid applications.
2) **Portability:** SCORE is a Linux emulation platform that the codes of Smart Grid applications executing inside the emulated nodes can be easily ported to real embedded system devices and testbeds controlled by Linux.
3) **Scalability:** SCORE allows distributed emulation using general PCs for power and communication networks, providing the capability to deal with large-scale Smart Grid applications.

This research is supported by NSF grant (NSF-CPS-1135814) to Sensorweb Research Laboratory at Georgia State University.

II. System design and Implementation

SCORE captures the behavior and interactions of power networks and communication networks and emulates Smart Grid applications in an integrated and scalable environment. The researchers can verify the correctness of their new Smart Grid applications on a few devices and then use SCORE for large scale evaluation and performance prediction studies. Also, they can use SCORE to evaluate and validate the design and performance of their Smart Grid applications before large-scale deployment.

A. Overview

Our design of SCORE takes advantage of CORE's structure. Figure 1 provides an abstract overview of SCORE's architecture and our integration approach. As shown, SCORE consists of GUI, Service Layer, Communication Module and Power Module. The GUI provides an easily drag-and-draw environment for the users to set up the emulations. Configuration parameters for the emulated nodes, power lines, communication links and distributed emulation servers can all be specified. Interacting with GUI through Start-UP daemon, the service layer are python frameworks responsible for creating emulation sessions, instantiating the virtual nodes, communication links and power lines, etc. The service layer also wraps up the low level implementations from communication module and power module as various Smart Grid services. Those services can all be employed or customized by the users to build up their own Smart Grid applications. More importantly, Linux network namespaces is employed for Linux containers support so that each emulated node can have its own instance of virtual or real network devices, network protocol stack and process space while sharing the file system of emulation server with other nodes. This light-weighted virtualization approach enables the scalability of SCORE. Moreover, from the perspective of Smart Grid applications running inside the virtual node, each emulated node is just another piece of hardware platform controlled by Linux OS, which equips SCORE with the property of portability that the emulated node can execute unmodified Smart Grid application codes, which may be originally running inside real Linux-controlled hardware devices, and vice versa.

B. Communication Module

The communication module in SCORE leverages the comprehensive support of various wired and wireless communication network models and protocols from CORE. Each emulated node has its own instance of OS implemented TCP/IP stack supported by Linux namespaces, from the perspective of Open Systems Interconnection (OSI) model, thus SCORE allows high fidelity emulation of network layer and above. By default, a simplified simulation of link and physical layers is enabled, which is implemented using netem with Ethernet bridging in Linux. Statistical network effects such as bandwidth, loss rate, bit error rate and noise level can be configured and applied. For higher-fidelity link and physical layer emulation, other network simulation tools, such as EMANE [6], can be easily integrated. In addition, the virtualized Ethernet interface 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, allowing real time communication between the virtual nodes inside a running emulation and external physical networks. We employ this feature to enable the interactions between the communication module and the power module discussed in the next section. The idea is that the power module is running on a host physically in the same network with the communication emulation host so that the power module can receive and react to the queued-up messages sent by all the emulated virtual node in real time.

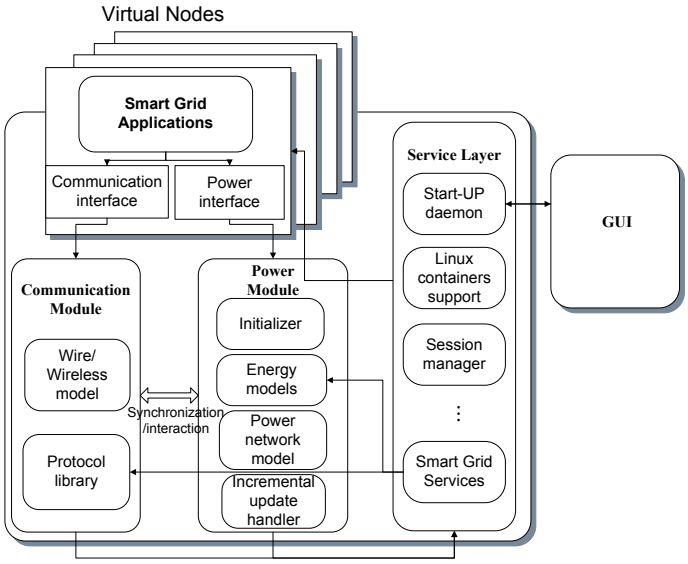

Fig. 1. SCORE Architecture

C. Power Module

The power module in SCORE emulates the dc power flows within Smart Grid and also provides implementations of predefined energy models. Figure 2 shows the data flow diagram of power module. The power module receives initial power network topology and energy model configuration information from service layer to formulate the initial power network model, which will be introduced later. The model is updated when the corresponding model updates are received from the interactions with communication module. SCORE is intended to emulate large scale power network in real time. Due to this requirement, we adopt incremental model updating in our computation to react to the system status changes. More importantly, as the size and order of the power network increase, parallel computation for power network becomes a necessity for efficient Smart Grid emulation. SCORE highlights itself in scalability by enabling the user to conduct the emulation in a distributed way when a single PC cannot provide enough computation capabilities. We choose to split the power network model into several partitions and let each

partition be computed and updated separately in parallel. With appropriate coordinating among those separate computing and updating processes, the merged result of power flows in Smart Grid is solid without any loss of precision compared with centralized computation.

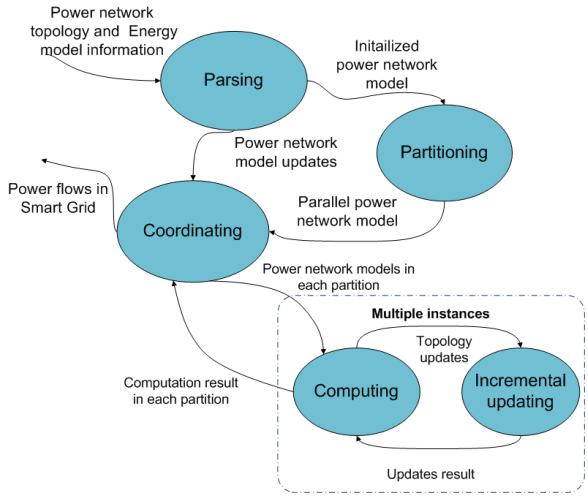

Fig. 2. Data flow diagram of Power Module

1) Power network model

Now let us introduce the power network model. 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:

$$AV = I \tag{1}$$

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 neighbor set of node i in the power network, we get:

$$a_{ij} = \begin{cases} \sum\limits_{s \in Nbr(i)} g_{is} & i = j. \\ -g_{ij} & j \in Nbr(i) \\ 0 & otherwise \end{cases} \tag{2}$$

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) Incremental updating

Based on previous model, let's consider the situation when the power network topology changes. Suppose the power grid status changes, such as the admittance between node i and node j is changed by Δg_{ij}. So the new coefficient matrix $\widetilde{A}$ can be written as:

$$\widetilde{A} = A + U \Delta g_{ij} U^T \tag{3}$$

where

$$U = \begin{bmatrix} 0 & \cdots & \underset{i}{1} & \cdots & \underset{j}{-1} & \cdots & 0 \end{bmatrix}^T$$

Particularly, the changed admittance Δg_{ij} equals to $-g_{ij}$ when the branch is removed and $\Delta g_{ij} = g_{ij}$ when a new branch is added. Notice that [7]

$$\widetilde{A}^{-1} = A^{-1} - A^{-1} U \left({\Delta g_{ij}}^{-1} + U^T A^{-1} U \right)^{-1} U^T A^{-1} \tag{4}$$

then we can get the $\widetilde{A}^{-1}$ with a much lower computation cost when we store previously computed A^{-1}.

3) Parallel computation

For parallel computation, the whole power network model are divided into several sub-grids [7]. The branches that connecting two sub-grids are called connection branches. Suppose the whole grid is divided into K sub-grids, and there are c connection branches between them. Let A_{kk} denote the nodal admittance matrix of sub-grid k, $k = 1, \cdots, K$. Then we can use block diagnal matrix A^- to represent the grid system without the c connection branches:

$$A^- = \begin{bmatrix} A_{11} & 0 & \cdots & 0 \\ 0 & A_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & A_{KK} \end{bmatrix} \tag{5}$$

Define $c \times c$ matrix C as the branch admittance matrix for all the c connection branches:

$$c_{ij} = \begin{cases} y_t & i = j. \\ 0 & otherwise \end{cases} \tag{6}$$

where y_t is the branch admittance for branch t, $t = 0, 1, ...c-1$.

Define $(n-1) \times c$ node-to-branch incidence matrix P for the c connection branches:

$$p_{ij} = \begin{cases} 1 & power\ flow\ j\ leaves\ node\ i \\ -1 & power\ flow\ j\ enters\ node\ i \\ 0 & power\ flow\ j\ is\ not\ incident\ with\ node\ i \end{cases} \tag{7}$$

where $i = 0, 1, ...n-2$, $j = 0, 1, ...c-1$.

So from (5) (6) (7), we have:

$$A = A^- + PCP^T \tag{8}$$

We define matrix Q by simply replacing all the -1 elements in matrix P with 1. As a result, we have:

$$A = A^+ - QCQ^T \tag{9}$$

where

$$A^+ = A^- + PCP^T + QCQ^T \tag{10}$$

A^+ consists of K blocks, A^+_{ii}, $i = 1, 2, ...K$ on its main diagonal and each sub-linear system formed by those blocks is guaranteed to have a unique solution [7]. Introduce

$$\tau = -CQ^T V \tag{11}$$

Use (9) and (11), (1) is re-written as:

$$\begin{bmatrix} A^+ & Q \\ Q^T & C^{-1} \end{bmatrix} \begin{bmatrix} V \\ \tau \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{12}$$

So we have:

$$\tau = (C^{-1} - \sum_{i=1}^{K} Q_i^T (A^+_{ii})^{-1} Q_i)^{-1} (- \sum_{i=1}^{K} Q_i^T (A^+_{ii})^{-1} I_i) \tag{13}$$

and

$$V_i = (A^+_{ii})^{-1}(I_i - Q_i \tau) \tag{14}$$

where $i = 1...K$.

According to (13) and (14), the parallel computation is executed as the following three steps:

- **Step 1**: Each computing process k ($k = 1, 2, ...K$) calculates

  $$x_k = Q_k^T (A^+_{kk})^{-1} Q_k \tag{15}$$

  and

  $$y_k = Q_k^T (A^+_{kk})^{-1} I_k \tag{16}$$

  then x_k and y_k is delivered to the coordinating process.
- **Step 2**: Use the results x_k ,y_k ($k = 1, 2, ...K$) in step 1 from all the computing process, the coordinator calculate τ in (13). Then it broadcasts τ to all the computing processes.
- **Step 3**: Upon receiving τ from the coordinating process, the computing process k calculates the voltages of all the nodes V_k in subgrid k by (14).

D. Energy model

Energy models are implemented and wrapped as an independent component in power module to provide implementations to service layer. We provide energy model programming interfaces for shiftable and non-shiftable loads, renewable energy supplies (including solar panel and wind turbine), etc. Users can implement and integrate their own customized implementations in SCORE to support more advanced energy models.

III. Evaluation

We evaluate SCORE's key features for Smart Grid emulation by a demand response walk-through case in this section. Note that our contribution here is not the demand response algorithm or power allocation strategy itself. We just intend to present that SCORE has sufficient capabilities to emulate Smart Grid applications.

The test case is created as a Smart Grid power distribution network with one power plant and five houses. Each house is connected with the power network through an intelligent power switch, which serves as the energy control center for the house. Within each house, there are five kinds of nodes: shiftable loads (represented by Plug-in Hybrid Electric Car), non-shiftable loads (represented by Bulb), renewable resources (represented by solar panel and wind turbine), and power storages (represented by battery). Two wireless networks (the cloud icon) wlan33 and wlan34 equip each node with communication capabilities. We configure the wireless channel to be unlimited bandwidth, no delay or interference and the resistances of all the power lines to be the same constant value. Figure 3 shows the setting up of the test case. Two emulation servers, SCORE SERVER ONE and SCORE SERVER TWO, are used to conduct distributed emulation. Each is assigned part of the nodes in the Smart Grid. For each virtual time

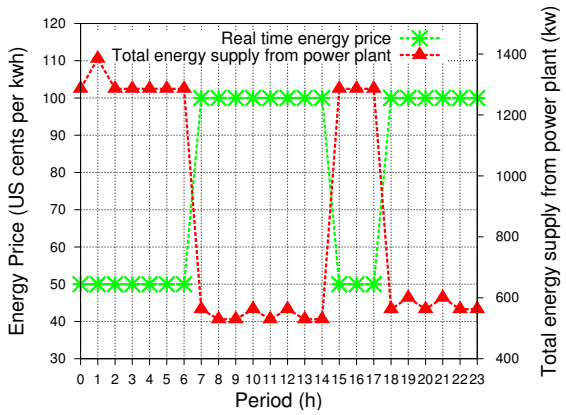

Fig. 4. Real time price and Total energy supply from power plant

period h in a virtual day, $h = 0, 1, 2, 3, ...23$, each kind of nodes in the study case behaves as the following:

- **Power Plant:** Broadcasting its real time energy prices to all the intelligent power switches. The energy price has only two values: HIGH (100 cents per kwh) and LOW (50 cents per kwh).
- **Intelligent Power Switch:** Receiving the energy price information from the power plant and then relay the messages to the other nodes within the same house immediately.
- **Bulb:** Receiving energy price information from intelligent power switch. When the price is HIGH, it lowers its load from 200w to 100w. When the price is LOW, it increases its load from 100w to 200w.
- **Plug-in Hybrid Electric Car:** Receiving energy price information from intelligent power switch. When the price is HIGH, it stops charging. When the price is low, it resumes charging to 100w.
- **Solar Panel:** When $h \in H1 = \{h | h = 0, 1, 2, 3, 4, 5, 19, 20, 21, 22, 23\}$, setting its maximum output to 0. When $h \in H2 = \{h | h = 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}$, setting its maximum output to 30w.
- **Wind Turbine:** Randomly setting its maximum output to 30w or 60w.
- **Power Storage:** Receiving energy price information from intelligent power switch. When the price is HIGH, it

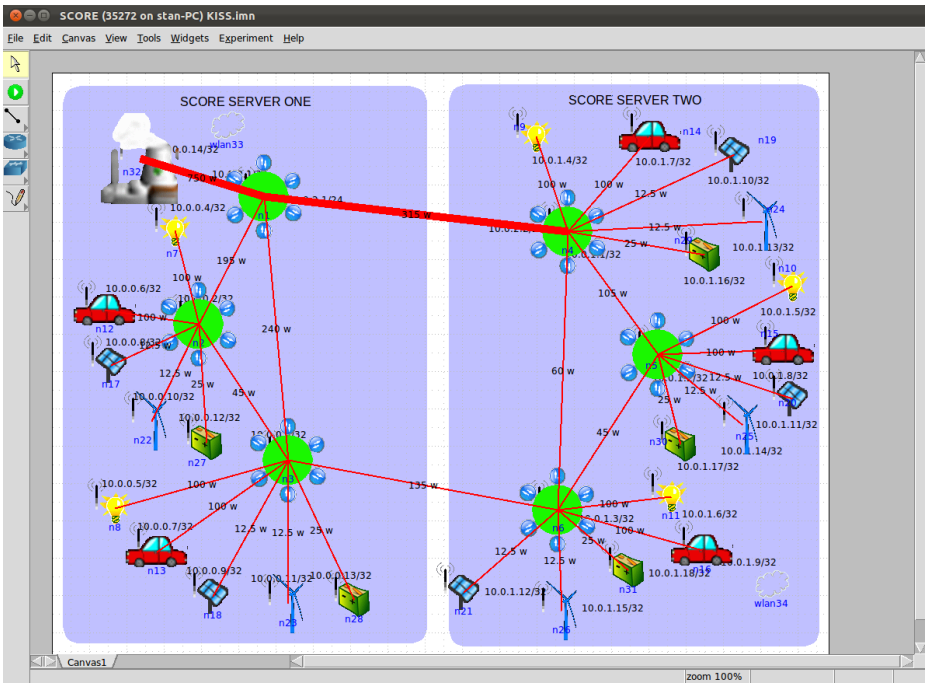

Fig. 3. The Study Case

discharges at maximum of 60w. When the price is LOW, it stops discharging.

A. Fidelity

SCORE captures the dc power flow analysis for Smart Grid applications in real time. Figure 4 shows the real time energy price and the total energy supply from power plant in a 24 hours virtual day. The impact of the demand response strategy on total energy supply is clearly illustrated by the two trend curves in the test case: when the price goes up, the total energy supply goes down. And when the price goes down, the total energy supply goes up correspondingly. Renewable resources result in the fluctuations of total energy supply curve. More specifically, table I presents the power values passing through the nodes and the power lines between intelligent power switches during execution periods. For example, when the virtual clock $h = 6$, the energy price is low, so that the bulb is in 200w mode, the PV-car is in 100w mode, the storage stops charging. The power flows of all power lines conform to Kirchhoff's circuit and voltage laws in power network nodal analysis.

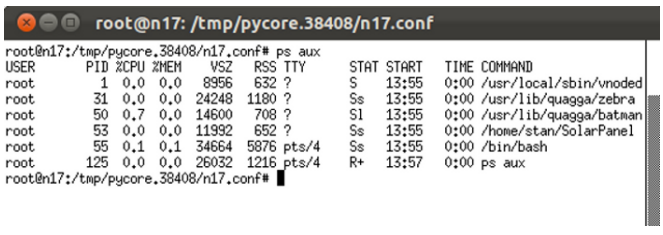

Fig. 5. Processes running inside the emulated node in SCORE

B. Portability

SCORE highlights itself in portability that Smart Grid applications running inside each emulated node is actually running in a real Linux environment. Therefore, all the emulated applications can be directly ported to Linux-enabled devices, such as Smart Grid Infrastructure Evaluation Board in TI [8]. Figure 5 shows the processes running inside node n17 during the case execution, which are listed using $ps\ aux$ command in SCORE's run-time shell. The Smart Grid application running in it at that moment consists of a energy model process $Solar\ Panel$, and a routing protocol process $batman$, which are all installed in our Ubuntu 11.10 machine.

C. Scalability

SCORE extends its scalability by distributed emulation. We evaluate SCORE's scalability using the previously introduced test case and extend it to a much larger scale. Our testing machines are 64 bits HP destop with Pentium(R) Dual-Core CPU E5700 @ 3.00GHz and 4GiB memory. Figure 6 and Figure 7 shows the peak CPU usage and memory usage of SCORE running on one, two and four machines when the scale of the Smart Grid increases. We can see from the figures that as the number of emulation servers increases, the CPU and memory usage of each machine is decreased since each of them can take care of the instantiation, computation and communication in parts of the Smart Grid. SCORE's distributed emulation capability greatly release the burden of each emulation server and enable large scale emulation.

Figure 8 shows the execution time of the study case for 24 clocks. Note that when the scale of power grid is small, which means the computation cost is relatively low, less ma-

TABLE I
TEST CASE RESULT

Period	h=0	h=2	h=6	h=8	h=12	h=14	h=16	h=18	h=20	h=22
Bulb(n7)	200w	200w	200w	150w	150w	150w	200w	150w	150w	150w
Plug-in-electric Car (n12)	100w	100w	100w	0w	0w	0w	100w	0w	0w	0w
Solar Panel(n17)	0w	0w	21.4286w	8.82353w	9.375w	8.82353w	21.4286w	9.375w	0w	0w
Wind Turbine(n22)	42.8571w	42.8571w	21.4286w	17.6471w	9.375w	17.6471w	21.4286w	9.375w	18.75w	18.75w
Storage(n27)	0w	0w	0w	17.6471w	18.75w	17.6471w	0w	18.75w	18.75w	18.75w
Power line (n1,n2)	334.286w	334.286w	334.286w	137.647w	146.25w	137.647w	334.286w	146.25w	146.25w	146.25w
Power line (n1,n3)	411.429w	411.429w	411.429w	169.412w	180w	169.412w	411.429w	180w	180w	180w
Power line (n1,n4)	540w	540w	540w	222.353w	236.25w	222.353w	540w	236.25w	236.25w	236.25w
Power line (n2,n3)	77.1429w	77.1429w	77.1429w	31.7647w	33.75w	31.7647w	77.1429w	33.75w	33.75w	33.75w
Power line (n4,n5)	180w	180w	180w	74.1176w	78.75w	74.1176w	180w	78.75w	78.75w	78.75w
Power line (n4,n6)	102.857w	102.857w	102.857w	42.3529w	45w	42.3529w	102.857w	45w	45w	45w
Power line (n5,n6)	77.1429w	77.1429w	77.1429w	31.7647w	33.75w	31.7647w	77.1429w	33.75w	33.75	33.75w
Power line (n3,n6)	231.429w	231.429w	231.429w	95.2941w	101.25w	95.2941w	231.429w	101.25w	101.25w	101.25w

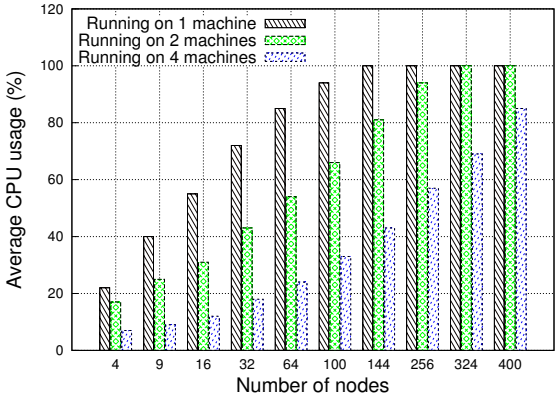

Fig. 6. Peak CPU usage running the study case in SCORE

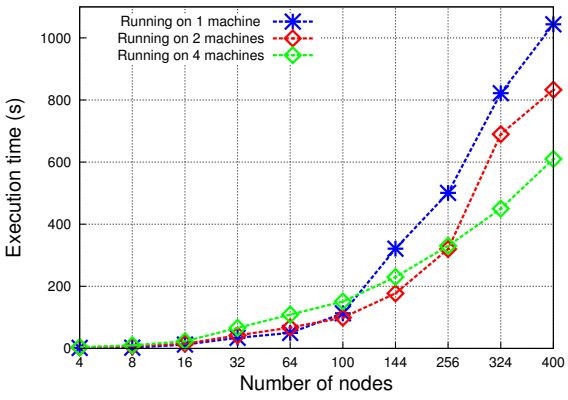

Fig. 8. Execution time of the study case for 24 clocks in SCORE

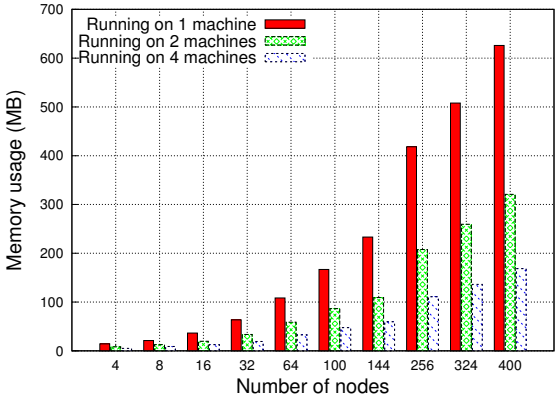

Fig. 7. Memory usage running the study case in SCORE

chines can finish the emulation relatively faster because there is less communication overhead between emulation servers. However, when computation time dominates the execution time, the advantages of parallel computation begin to emerge.

IV. CONCLUSION

In this paper, we present the design, implementation and operation of SCORE for Smart Grid emulation. From the evaluation, we demonstrate the fidelity, portability and scalability of SCORE when dealing with Smart Grid emulations.We have successfully released our first version of SCORE software and manual through http://sensorweb.cs.gsu.edu/?q=score.

REFERENCES

[1] W.-Z. Song, D. De, S. Tan, S. Das, and L. Tong, "A wireless smart grid testbed in lab," *Special Issue on Recent Advances in Wireless Technologies for Smart Grid, IEEE Wireless Communications Magazine*, 2012.

[2] T. Godfrey, S. Mullen, R. Dugan, C. Rodine, D. Griffith, and N. Golmie, "Modeling smart grid applications with co-simulation," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, oct. 2010, pp. 291 –296.

[3] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili, "Power system and communication network co-simulation for smart grid applications," in *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, jan. 2011, pp. 1 –6.

[4] D. Stimoniaris, D. Tsiamitros, T. Kottas, N. Asimopoulos, and E. Dialynas, "Smart grid simulation using small-scale pilot installations. - experimental investigation of a centrally-controlled microgrid," in *PowerTech, 2011 IEEE Trondheim*, june 2011, pp. 1 –6.

[5] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim, "Core: A real-time network emulator," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, nov. 2008, pp. 1 –7.

[6] J. Ahrenholz, T. Goff, and B. Adamson, "Integration of the core and emane network emulators," in *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, nov. 2011, pp. 1870 –1875.

[7] B. Zhang, S. Sun, and Z. Yan, *Advance Power analysis*, T. U. Press, Ed., 2007.

[8] TI. [Online]. Available: http://www.ti.com/tool/tmdssgi-evml138