Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

DESIGN OF A HIGH-FIDELITY TESTING FRAMEWORK FOR SECURE ELECTRIC GRID CONTROL

Srikanth B. Yoginath
Kalyan S. Perumalla

Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6085, USA

ABSTRACT

A solution methodology and implementation components are presented that can uncover unwanted, unintentional or unanticipated effects on electric grids from changes to actual electric grid control software. A new design is presented to leapfrog over the limitations of current modeling and testing techniques for cyber technologies in electric grids. We design a fully virtualized approach in which actual, unmodified operational software under test is enabled to interact with simulated surrogates of electric grids. It enables the software to influence the (simulated) grid operation and vice versa in a controlled, high fidelity environment. Challenges in achieving such capability include achieving low-overhead time control mechanisms in hypervisor schedulers, network capture and time-stamping, translation of network packets emanating from grid software into discrete events of virtual grid models, translation back from virtual sensors/actuators into data packets to control software, and transplanting the entire system onto an accurately and efficiently maintained virtual-time plane.

1 INTRODUCTION

Energy delivery control systems are predominantly software-driven. Much of the software is supplied by third party vendors, including hardware providers, application vendors, tool vendors, or operating system companies. The reality is that software keeps rapidly evolving. Software updates are very common; update frequencies vary wildly, some daily such as patches to popular Linux distributions, to weekly such as to popular open software applications, to monthly such as to the Microsoft Windows operating systems, to yearly such as to software maintained under specific contracts.

Due to the complex and interlinked nature of software components, any update to any portion of the software system has the potential to alter the operational characteristics of the system as a whole. Such fragility is unfortunate, but reflects the reality of today. Naturally, as a result, utility operators can be expected to be reticent to making any changes to existing, operational software systems lest they introduce unknown, unanticipated or undesirable instabilities. On the other hand, to ensure continued stable operations and to improve the security robustness of the overall system, avoidance of the software updates may in fact be precisely the opposite to system stability and security of the system.

The crux of the problem is that there is currently limited methodology and apparatus to uncover or predict the effects of software updates on the system operation *before the updates are incorporated*. Also, the problem is fundamental in nature (i.e., it will not go away by itself if ignored or tolerated) because software updates cannot be adequately modeled or abstracted to analyze off-line. Software applications and their updates are self-models. They are often in binary (executable) form and often closed-source. Installations and inter-module interactions are often too complex to be abstracted to predict the net outcome of their interactions. Also, cyber technologies (protocols, tools, applications, malware) evolve so

quickly that they are fast outpacing our ability to model and analyze them with traditional techniques. It has now come to the point where complex, distributed software systems are often the best and only feasible *models of themselves*. Specific TCP/IP stack implementations or complex peer-to-peer protocols, for example, are extremely difficult to model as abstractions without losing the effects of their critical, often poorly understood, bugs, features, and performance dynamics. Due to such intractable nature of software effect prediction, the only way to be assured of acceptability of updates is to somehow incorporate the updates on the side, exercise the software in full, gory detail as is, and study in a controlled setting its effects on the grid operation. An effective, high-fidelity experimentation platform is needed to increase the speed and fidelity with which cyber technologies are designed and tested in conjunction with the operation of electric grids.

1.1 Technical Challenges

Among the primary challenges is the problem of hosting actual control software systems in full-blown, gory detail to be tested as is. This requires that the simulated software interact with a controllable surrogate of grid hardware. Traditional emulation techniques are inadequate because of real-time execution challenges – the emulated hardware must guarantee real-time emulation to keep in synchrony with the software which is executing based on a real-time clock. Sophisticated and customized emulation systems exist for use as electric grid surrogates (US DOE 2009; Nicol, Davis, and Overbye 2009; Bergman et al. 2009); however, they are prohibitively with increasing scale of the grid simulated expensive (some costing to the tune of a million dollars per site installation) and untenable in the long run when the level of detail in the surrogate grid needs to be increased (e.g., more complex sensors and actuators, or larger grid network sizes). Thus, the problem of real-time coupling must first be resolved (ideally, eliminated) if the actual software must interact directly with virtual hardware.

The solution we adopt is based on virtual machine technologies that have lately gathered significant research and development. Note, however, that traditional VM technologies as is cannot be used in grid software testing because traditional VMs also execute over real-time clocks, albeit by time-sharing the computer hardware among multiple VMs. The VM approach, however, brings with it the ability to control the time advances, which we exploit for our present purposes by developing a new, customized scheduler that is suited for interfacing with a virtual grid. This new scheduler effectively lifts the software away from real-time clock to a virtual time clock, which we fully control and synchronize with the virtual time clock of the virtual grid.

The reason we employ VMs in this proposed solution methodology is that VMs are the best way to sustain the immense level of fidelity and ease of experimentation needed to properly account for the fundamental, sensitive effects of software patches. Any other method would necessarily have to resort to degenerated abstractions, which are unknown to begin with, because of the ill-characterized nature of rapid software updates themselves.

Also, our virtual testing system approach opens the possibility to introduce a virtual network between the software systems and the sensors/actuators that control the grid. This virtual network, wired or wireless in nature, also becomes important to capture in sufficiently high level of detail in order to uncover important unforeseen effects such as congestion from simultaneous software updates/patches over relatively low bandwidth or shared links (e.g., as is the case with smart grid devices).

1.2 Time Control and Synchronization

The heart of the hypervisor is its scheduler, which determines which VM gets to execute next at any given moment on the host processor on which the grid software is being virtually tested. The custom scheduler is built from the ground up to suit the purposes of time control and synchronization of the software VMs with the virtual grid. In our project, we will use a virtual time control scheduler that introduces the relaxation necessary in the relation between real time and virtual simulation time.

The virtual machines capture all the detail needed, including the actual versions of the operating systems and software tools executing on those computers. While VM technology is in use for common applications, the type of VM technology needed to enable our solution approach is unique and requires further research and development before VMs can in fact be interface correctly and efficiently with virtual electric grid sensors, actuators, communication network and electric network (transmission or distribution). Conventional VM technologies are concerned with utilization of the computer, whereas testing software accurately interfaced with the complex timing and functionality of virtual electric grid requires a different type of VM execution altogether.

1.3 Time Controlled Communication

Another important technical challenge to realize virtual software testing with virtual grid lies in achieving time-synchronized communication between the software and the grid interface elements. The control mechanism of the software ultimately manifests itself in the form of network packets emerging out of network devices of the software host platform. These packets, routed over the network, reach the various sensors and actuators which are equipped to translate the command data from network data language to control signals on the grid side. Similarly, this sequence operates in reverse, with the sensors translating the sensed information back to packets and conveying them over the network to the software. The challenge in "in vitro" testing is to correctly virtualize this entire control sequence as well, with correct timing, pacing and ordering of the software such that the network packets arrive and depart in the correct sequence with respect to simulated hardware. None of the existing testing methods or systems currently provides the technology to enable such controlled communications.

We resolve this by developing a virtual time-controlled communications module in the hypervisor, trap all incoming and outgoing network activity of the VMs in which the software is executing, and wrap the packets inside time-stamped events. These time-stamped events are then synchronized with virtual time progress across the entire virtual system, ensuring correctness of time advancements and relative ordering of events between the software and virtual grid. To develop and implement this scheme, we will build over recent success with a similar approach in the area of VM-based simulations of battlefield communications.

2 ARCHITECTURE

A core requirement in developing a testing framework for electric grid control is to interface the electric grid simulator with supervisory control and data acquisition (SCADA) and telecommunication network simulators/emulators (Lemay, 2013). In the rest of the paper, the terms telecommunication network simulator and network simulator are synonymously used. High-fidelity time-accurate network simulation systems such as NetWarp (Yoginath and Perumalla, 2011; Yoginath, Perumalla, and Henz 2012), based on VMs are extremely attractive for the task of including networked VMs to model grid control networks. However, such network simulation test-beds for distributed protocols and applications are not directly amenable to work in tandem with a electric grid simulator. Also, executing parallel simulations in a VM environment suffers a performance degradation, unless the scheduling of the Virtual Central Processing Units (VCPUs) of VMs hosting the parallel simulator is handled carefully. To extract the best runtime from the electric grid simulator and to utilize the high-fidelity VM based network simulations, such as the NetWarp simulator, to work with the electric grid simulator, we introduce a framework. This framework, in principle, can interface any application-specific simulators with VM-based network simulator. In particular, the design presented in this paper focuses on electric-grid simulations.

2.1 Electric Grid Modeling and Simulation Framework

Figure **1**, shows different layers of mapping of (a) simulation model to a logical parallel executive framework (b) logical executive framework to VM-based execution framework (c) VM based execution

framework on to the physical hardware. Layer (a) is well understood and often uses a spatial decomposition approach, (b) and (c) mapping levels are customized to ensure correctness of simulations and to yield good performance.

The entire framework is developed over a hypervisor customized for virtual time-ordered execution of simulations. This customized hypervisor is capable of scheduling the VCPUs of the virtual machines in virtual-time-ordered manner for ensuring correctness in simulations involving VMs (network simulations) and enhancing the performance of parallel simulations executing on this customized hypervisor.

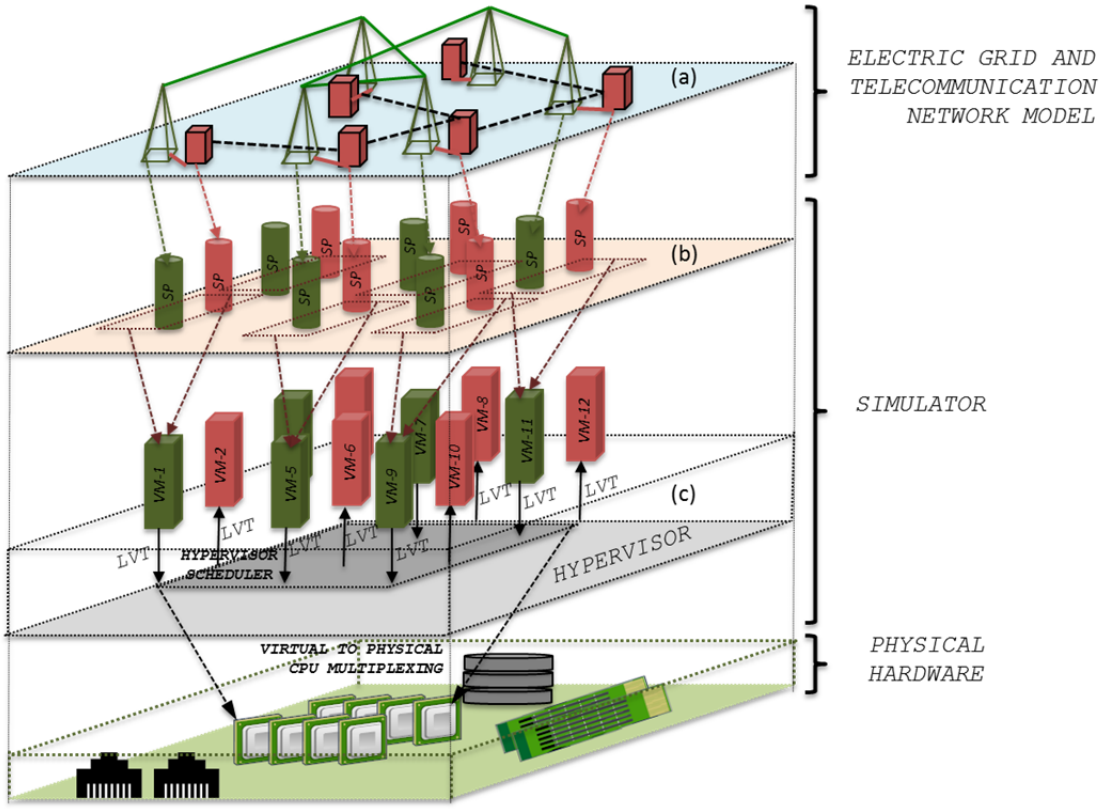

Figure 1: Functional diagram of the test framework for secure electric grid.

Figure **1** identifies two distinct networks in the simulation model namely, the electric grid and the networked SCADA units or telecommunication network. The dynamics of the electric grid is different from the telecommunication network dynamics and so does their simulation models. The logical mapping of network simulation to NetWarp like simulators using the VMs as the end-hosts and, the mapping of electric grid simulations to the generic parallel discrete event simulation (PDES) processes (Fujimoto 2000) using the VMs as execution environment, provides us with a powerful and high-fidelity simulation framework for secure-electric grid.

To achieve this all the simulation processes corresponding to both electric-grid and network simulators are instantiated. Note that all these simulation processes are hosted on VMs as shown in the mapping of PDES onto hypervisor in Figure **1**. In addition, a selected set of simulation processes corresponding to the network simulator maintain additional VM surrogate. The events from the electric-grid simulation processes to the network simulation processes are delegated to their corresponding VMs by these network simulation processes. A VM instantiated for a network simulation process is referred as VMSP in the rest of the paper and every VMSP hosts the actual SCADA application that services the events from electric grid. Hence, the high-fidelity simulation characteristic can be expected from the SCADA unit components of the electric-grid. Given this setup, the synchronization between the electric-grid simulation pro-

cesses, network simulation processes and the VMSPs become an issue. To address the synchronization issue the local virtual time (LVT) from the simulation processes and the VMSPs are communicated through the hypervisor.

The hypervisor layer in the Figure **1** shows LVT values passing from the VMs hosting simulation processes to the hypervisor and also shows the VMSPs reading this LVT values, this communication of LVTs ensures synchronized execution of the secure-electric grid framework. Hence, a few VMs hosted by the hypervisor will be executing the simulation, while the other VMs will actually be working as a component of the same simulation. Hence, VMs hosting simulations as well as VMs serving as components of the simulation will be executing on the same physical machine sharing the compute resources.

The main challenge in such an assembly of simulators that are functionally and operationally different is the handling of the multiple simulation-timelines and being able to consolidate the simulation-time mismatch between the simulators. Unlike VM-based network emulators (Liu 2008; Apostolopoulos and Hasapis 2006; Bergstrom, Varadarajan, and Back 2006) that rely on wall-clock time alone, NetWarp maintains its own simulation timeline, keeping track of the number of physical CPU cycles utilized by the VM. Hence, the integrated framework requires the simulation time from the electric-grid simulator and network simulator surrogates be passed to the hypervisor. At this point, the VMSPs perform necessary functionality, while communicating the elapsed simulation time along with the relevant subsequent events for the electric simulator.

3 DESIGN ASPECTS

3.1 Simulation Events

With multiple simulation modules in the framework, the types of events within the framework can be broadly classified into two, namely (a) module specific events and (b) inter-module events.

3.1.1 Module-Specific Events

Module-specific events are those events which do not affect the behavior of the other simulator modules in the framework. For example, the network simulator module-specific events does not affect the behavior of the electric-grid simulator module and vice-versa. However, their precision in simulation time affects the fidelity and state at any point in the simulation and hence, are important. This is because when inter-module interactions occur, these independent individual modules together determine the state from which the whole simulation state evolves. Hence, these events are significant in cyber security application (worms or hot-patches) effects are to be tested. Thus, synchronization among the electric-grid SPs and the network simulation VMSPs is necessary even during handling module-specific events. This enables maintaining a single virtual time line in which both these simulation modules evolve.

3.1.2 Inter-Module Events

Inter-module events are those events, which affect the behavior of the peer simulator module and in turn might expect reactive set of events from the affected peer. In the context of secure electric grid simulator framework these events can be a) network simulation process events (from VMSP) affecting electric-grid simulation process; and b) electric-grid simulation process events affecting the network simulation processes and hence VMSP. Note that the each of the above mentioned inter-module events may or may not trigger state transformation in the simulation process, where it scheduled an event. Hence, based on the above mentioned combinations four types of inter-module events is expected in this simulation framework.

3.2 Synchronization in VM Platforms

To provide the insight on synchronization scheme in this framework, where VMs are assumed to be used both as execution platform as well as simulation processes of simulation (VMSP). We discuss each simulation module in this framework separately.

3.2.1 Electric-Grid and Telecommunication Network Simulation Modules

Within the electric-grid simulation module various grid nodes including the SCADA unit objects that delegate the event to their VM counter-parts are realized as DES based simulation processes (SP). The synchronization scheme for this module is either traditional conservative or optimistic synchronization schemes. However, as the secure electric grid framework is realized over a VM environment, for the performance reasons the hypervisor scheduler needs to be modified as has been demonstrated in Yoginath and Perumalla (2013). Similar arguments hold true for telecommunication network simulation modules.

This modified hypervisor scheduler changes the strategy of scheduling virtual CPUs on to physical CPUs, it uses least-LVT first principle to schedule the VCPUs. An LVT value is held by each VCPU and these LVT values are delegated to the VCPU objects in hypervisor by the SPs. However, this scheme is prone to deadlock and livelocks (Yoginath and Perumalla 2013) due to the need of the global-virtual-time computations in PDES. While this limitation can be overcome with algorithmic modifications, these modifications make the hypervisor scheduler very specific for the PDES applications. This essential specificity compromises the generality of least-LVT-first principle of hypervisor scheduling.

3.2.2 VM-Based Network Simulation Module

The VM based network simulation module in consideration within the secure electric-grid framework utilizes VMs as the end-hosts. In order to ensure correctness of the simulation time-ordered scheduling of VMs becomes necessary (Yoginath and Perumalla 2011; Yoginath, Perumalla, and Henz 2012). To overcome this, the virtual time for each VM based on the number of CPU cycles utilized is recorded by each VCPU of the VM and this value is the LVT in the network simulator context. Again, with least-LVT-first principle the hypervisor scheduler overcomes the correctness issue. However, as the network simulation advances the staggering of LVTs from different VCPUs spanning across different VMs become pronounce. In order to maintain a single simulation time-line across all the VMs involved in the network simulation, resetting the LVTs of all VCPUs to an appropriate approximation periodically overcomes this LVT staggering issue. However, this essential synchronization specificity compromises the generality of least-LVT principle of hypervisor scheduler.

3.2.3 Combined Execution of VM-Based Electric-Grid and Network Module

The secure electric grid framework requires the electric-grid and network simulation processes and VM based network simulation process instances to work in tandem. The distinction between the two hypervisor schedulers used in either of these applications need to be clear for realizing a combined framework. The electric grid and network SPs pass the LVT value to the hypervisor and VMSPs corresponding to the network SPs read their respective LVTs from the hypervisor.

The LVT values must be passed to the hypervisor, without which the VMSPs would by no means know the current simulation time of the electric-grid simulator. Also, the electric-grid simulator would suffer from deadlock and/or livelocks in the absence of the passage of LVT value from SPs to hypervisor. Hence, the onus to ensure working of the electric-grid simulation and the network simulation in tandem lies in intelligently adapting the VM based network simulation functionality to the specific needs of the framework.

The simulation time advancement in all SPs and VMSPs should be advanced synchronously requiring regular consolidation of the timelines of among the VM hosting SPs, and VMSPs. In addition to the syn-

chronized advancements, the simulator modules should have the capability to schedule an event to its peer and such events should be evaluated in simulation time order.

3.3 Data Interoperability

We know that the electric grid simulation framework comprises two different simulation models namely, electric-grid and network simulation models. However, the implementation of these models can widely differ in terms of their data-models, for example: the representation of event in the electric-grid can be very different from the event of the SCADA network simulator. It should also be noted that two different implementations of same simulations can highly differ.

Based on such implementation differences, data-interoperability between the events from one simulator to other becomes necessary. The events from one simulator with certain implementation specifics should be interpreted by an intermediate entity before scheduling it to another simulator such that the meaning of such an event is rightly conveyed to the simulator on which the event is scheduled. With a basic minimal support from the simulators like, APIs to read simulation time at any point in simulation, APIs to schedule external discrete events at a desired instance of simulation time and by being aware of data-model of the simulators, one can realize a data interoperability entities.

In the secure electric grid framework we envision using third party simulators for electric grid and network simulator, to work with high-fidelity VM based network simulator. The idea is while most of the electric-grid and SCADA network simulations are executed using low fidelity simulation processes, only few selected SCADA simulation processes will utilize VMSPs. Such a requirement necessitates data interoperability between electric-grid simulator and SCADA network simulators, SCADA network simulator and VM based network simulator. Further, the interoperability between two simulators has to be both ways, i.e. data or events from the electric-grid simulator to SCADA network simulator must be converted appropriately to SCADA network simulator understandable data-model and vice-versa. Note that with the knowledge of the data-model of each of the simulators along with supporting APIs for necessary simulation functionalities the realization of data-interoperability is merely a laborious implementation exercise rather than a technical hindrance.

4 IMPLEMENTATION ISSUES

4.1 Realizing Secure Electric Grid Framework

Three important aspects arise in realizing a secure electric grid framework (a) To be able to schedule inter-module events (b) To evaluate the inter-module events in simulation time-order (c) To be able to maintain a synchronized simulation timeline between the modules through out the simulation.

The network simulation process SP corresponding to the SCADA unit delegates event information to the VMSP. Similarly, the VMSP event information to the network simulation SP, which schedules an event in the electric grid simulation process based on the input from VMSP. For this to happen, we need a database (DB) that is accessible by all the VMs and can store event information. In Xen (D. Chisnall, 2008), the Xenstore serves this purpose and is ideal for small information exchange among VMs.

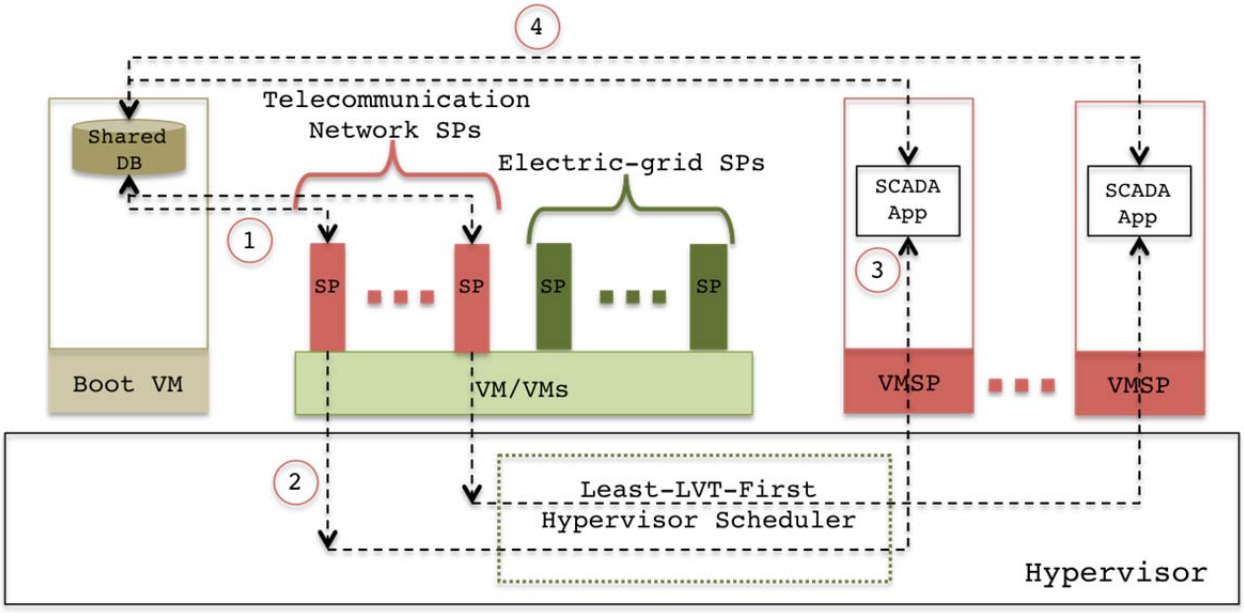

Figure 2: Inter-module event handling.

Consider, the electric simulator advances in periodic steps of certain granularity. At each step the network simulation process SPs write to the shared database if it needs to schedule an event in the VMSP and reads for any events to be processed at that simulation instance. The network SP then passes the simulation time to the hypervisor. The hypervisor scheduler updates the VCPU-LVT value of the corresponding VMSP. When the VMSP gets scheduled for execution, it reads its VCPU LVT value and checks for any event request in the shared DB. If there is a request it will service the request, stores the execution time in shared DB and it also schedules any event for the electric grid by providing the simulation time of future event and type of event information. These steps are numerically marked in the sequence of their functioning, as shown in Figure **2**. Note that the VMSP is scheduled only after the LVT value is sent and it's different from the currently read LVT value of its VCPU. After scheduling events for the electric simulator and if it does not have to service any module-specific events then the VMSP assigns its VCPU a very large LVT value, thus not allowing its VCPU to be scheduled for execution.

In the presence of module-specific events the CPU-cycles equivalent to the single step-size of the electric grid simulation are utilized by the VMSP in servicing those events, after which it assigns a high LVT value to its VCPU. Hence, this scheme of processing events helps addressing both module-specific and inter-module events.

4.2 Event Processing Algorithms

Apart from the module-specific events there could be four types of inter-module events as discussed previously. Based on these types of events the flow chart for network SP and its corresponding VMSP functioning is illustrated in Figure **3** and Figure **4**, respectively. These event processing algorithms are used to ensure time-ordered execution of electric-grid SPs, network SPs and VMSPs in synchronization with each other. Note here we have not discussed the synchronization between electric-grid SPs and network SPs because it is ensured by the PDES synchronization algorithms.

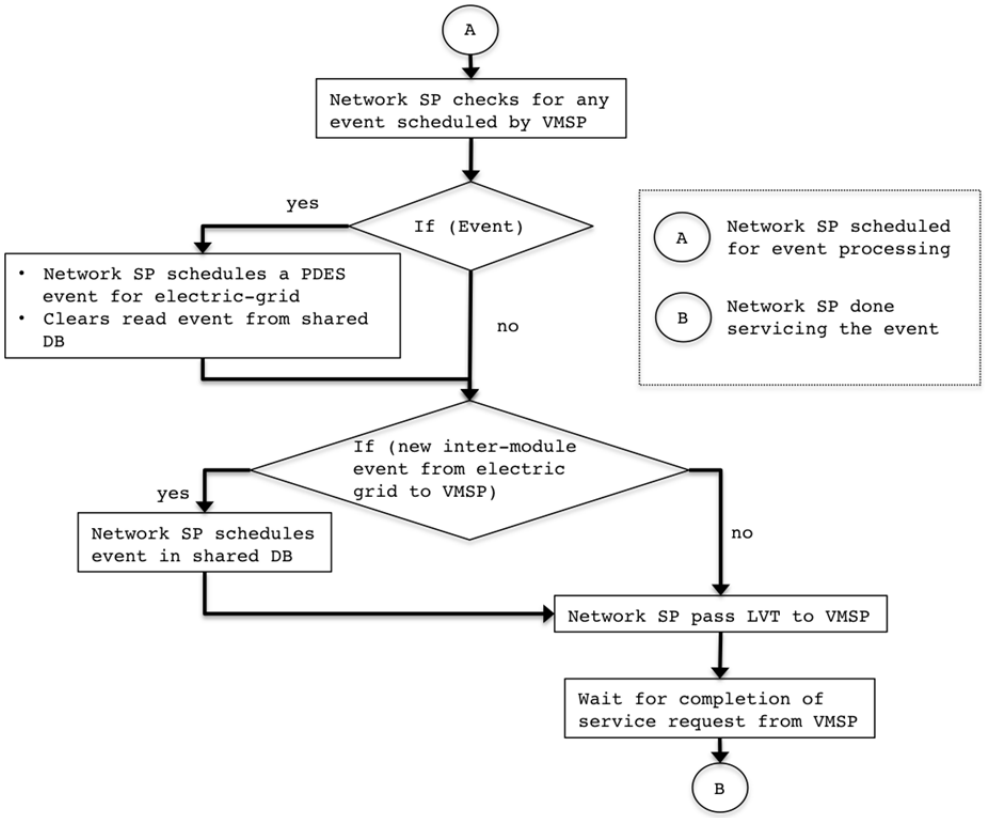

Figure 3: Telecommunication network SP functioning when scheduled to process an event.

In Figure **3** the algorithm on the functioning of a network SP with its corresponding VMSP is shown. When the VM hosting network SP obtains time-slice for execution, it first checks for any inter-module event from the VMSP to the electric grid in the shared DB. In the presence of such an event the network SP using its corresponding data-interoperability entity schedules corresponding event in the corresponding electric grid SP and clears the event from the shared DB. Then the network SP checks its event list for the presence of any inter module event scheduled by the electric-grid SP and schedules an event for VMSP in the shared DB in presence of such an event. Then the network SP passes its LVT or current simulation time value to the VCPU of its corresponding VMSP and waits for the response from its VMSP. This algorithm is iteratively executed by the network SP and hence delegates the inter module events from electric-grid SP to its corresponding VMSP.

Figure **4** provides the algorithm of VMSP functioning when it is scheduled for execution by the hypervisor scheduler. By default, the VCPU of any non-executing VMSP is held at a large value (MAX_LVT) and the least-LVT first based scheduler ensures that the VCPU is not given any CPU cycles at this stage. The LVT provided by the SP to the VCPU of its corresponding VMSP ensures that VMSP is given CPU cycles for execution. At this stage the VMSP reads the shared DB for any events for that LVT instance from the electric grid simulator. In presence of such an event the SCADA application hosted by the VMSP is invoked to service the event, while keeping track of the exhausted cycles in the process. If the SCADA application in response to the electric-grid event serviced creates further events to the electric-grid, then to convey these events to the electric-grid SPs appropriate events with simulation time (LVT + *exhausted_cpu_cycles*) is created in shared DB. Then VMSP checks for any module-specific event and if present, exhausts (LVT + *step_size*) worth of CPU cycles to process module specific events. Here, the step_size corresponds to the simulation time step size of the electric grid simulator. Af-

ter this the VMSP lets its corresponding SP (waiting) know about its completion and overwrites its VCPU LVT to the default MAX_LVT.

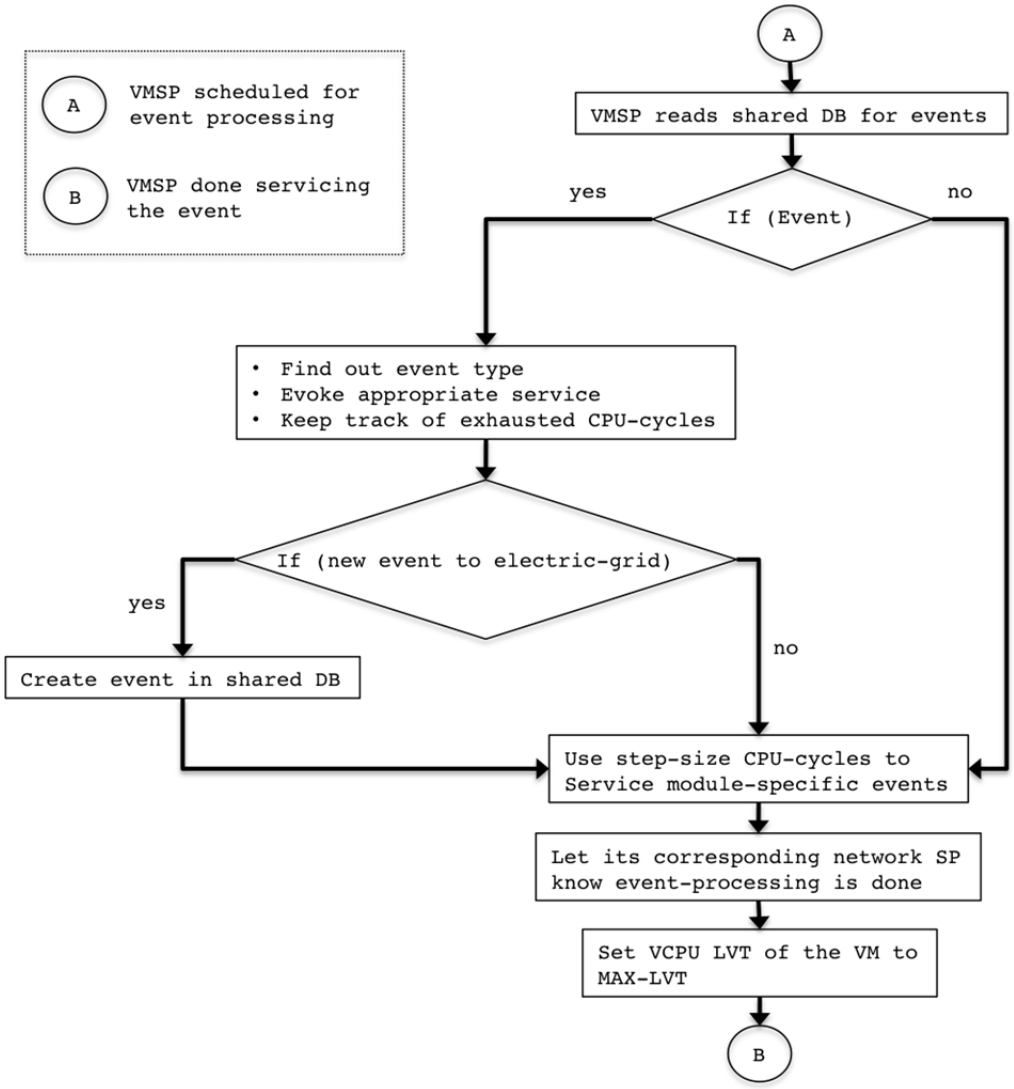

Figure 4: VMSP functioning.

5 SUMMARY

The design of an integrated simulation framework has been presented for high-fidelity testing of electric grid control software. The design using the new simulation algorithms and supporting modeling tools makes it possible to virtualize the four major components of advanced energy system: (1) application software, (2) operating system, (3) digital networks linking these, and (4) the physical machinery that the applications monitor and control. Our design provides the ability to probe the (simulated) grid to a high degree of detail to analyze the details of an undesirable operation uncovered from an applied software update, with minimal perturbation from probing or visibility inside the system. The design supports the following:

- **Nearly Universal Visibility**: The Direct Execution methods provide the ability for our system to potentially insert sensor and probing functionality at will at various points of execution. Visibility is not restricted to network activity or actual hardware sensors alone.

- **Defeating the "Uncertainty Principle"**: The design allows the insertion of arbitrary types and amounts of instrumentation at various execution points without affecting the original behavior of the system under test. Since our hypervisor freezes time advance during query or probing of the system state, the system behavior is unaffected by the amount of real time taken by logging and archival activity imposed on the system for observation.
- **Repeatability and Determinism**: The behavior of the system under test is unaffected with visibility; hence the execution remains deterministic and repeatable despite variation in the levels of probing. Time advances and temporal buffering inside the runtime engine ensure preservation of correct mutual orderings despite real time variation during execution of the experiments. In other words, our approach provides real time resilience to the system under test.

The integrated execution is realized via virtual time-ordered scheduling of virtual machines hosting the distributed grid control software, electric grid simulator, and network simulator. The virtual machines are divided into two types: one "time-regulating" class to host discrete event simulators that push their local virtual time (next earliest event time) to the hypervisor, and the other "time-constrained" class to host unmodified grid software whose virtual time is charged dynamically by the hypervisor. Inter-class and intra-class interactions are also time-synchronized as events delivered in global virtual time order. The combined operation enables the use of many virtual machines on a single multi-core platform integrated with an electric grid simulator and a network simulator also running on the same platform. Overall, the combined operation enables the possibility of a unique combination of scale and fidelity for simulating the effects of actual grid cyber infrastructure on grid operation.

ACKNOWLEDGMENTS

This paper has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, 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 authors thank James Nutaro for useful discussions on the internal organization of electric grid surrogates.

REFERENCES

Apostolopoulos, G. and C. Hasapis. 2006. "V-eM: A Cluster of Virtual Machines for Robust, Detailed and High-performance Network Emulation." In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis and Simulation of Computing and Telecommunication Systems*, 117-126.

Bergstrom, C., S. Varadarajan, and G. Back. 2006. "The Distributed Open Network Emulator: Using Relativistic Time for Distributed Scalable Simulation." In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation*, 19-28.

Bergman, D. C., D. Jin, D. M. Nicol., and T. Yardley. 2009. "The Virtual Power System Testbed and Inter-Testbed Integration." In *Proceedings of the 2nd Workshop on Cyber Security Experimentation and Test*.

Chisnall, D. 2008. *The Definitive Guide to the Xen Hypervisor*. Prentice Hall.

Fujimoto, R. M.. 2000. *Parallel and Distributed Simulation Systems*. Wiley Interscience.

Liu, J., 2008. "A primer for real-time simulation of large-scale networks." *In Proceedings of the 41st Annual Simulation Symposium*, 85-94.

Lemay, A., J. Fernandez, and S. Knight., 2013. "An isolated virtual cluster for SCADA network security research." In *Proceedings of the 1st International Symposium for ICS & SCADA Cyber Security Research*, 88-96.

Sztipanovits, J., G. Hemingway, A. Bose, and A. Srivastava. 2011. "Model-based Integration Technology for Next-Generation Electric Grid Simulations." In *Computational Needs for Next Generation Electric Grid*, edited by J. H. Eto and R. J. Thomas, 4-1—4-44.

Maier, S., A. Grau, H. Weinschrott, and K. Rothermel. 2007. "Scalable Network Emulation: A Comparison of Virtual Routing and Virtual Machines." In *Proceedings of the 12th IEEE Symposium on Computers and Communications*, 395-402.

Nicol, D. M., C. M. Davis, and T. Overbye. 2009. "A testbed for power system security evaluation." *International Journal of Information and Computer Security* 3(2): 114-131.

Yoginath, S. B., and K. S. Perumalla. 2011. "Efficiently Scheduling Multi-core Guest Virtual Machines on Multi-core Hosts in Network Simulation." In *Proceedings of the 25th Workshop on Principles of Advanced and Distributed Simulation*, Nice, France.

Yoginath, S. B., and K. S. Perumalla. 2013. "Optimized hypervisor scheduler for parallel discrete event simulations on virtual machine platforms." In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, 1-9. Brussels, Belgium: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.

Yoginath, S. B., and K. S. Perumalla. 2013. "Empirical evaluation of conservative and optimistic discrete event execution on cloud and VM platforms." In *Proceedings of the 2013 ACM SIGSIM conference on Principles of Advanced Discrete Simulation*, 201-210. New York, NY: ACM.

Yoginath, S. B., K. S. Perumalla, and B. J. Henz. 2012. "Taming Wild Horses: The Need for Virtual Time-based Scheduling of VMs in Network Simulations." In *Proceedings of the 20th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 68-77.

US DOE. 2009. "National SCADA Test Bed (NSTB) Fact Sheet." U.S. Department of Energy. www.energy.gov/sites/prod/files/oeprod/DocumentsandMedia/NSTB_Fact_Sheet_FINAL_09-16-09.pdf

AUTHOR BIOGRAPHIES

SRIKANTH B. YOGINATH is a Research Staff Member in the Discrete Computing Systems Group of the Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. He is also a PhD candidate at the School of Computaional Sciences and Engineering, Georgia Institute of Technology, Atlanta, GA. His email address is yoginathsb@ornl.gov.

KALYAN S. PERUMALLA founded and leads the Discrete Computing Systems Group of the Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA. He is a Senior R&D Staff Member and Manager at the Oak Ridge National Laboratory, and an Adjunct Professor at the Georgia Institute of Technology. He holds a PhD in Computer Science (1999, Georgia Institute of Technology). His email address is perumallaks@ornl.gov.