# Cyber-Physical Co-Simulation of Smart Grid Applications using ns-3


Muhammad Umer Tariq
School of Electrical and Computer Engineering
Georgia Institute of Technology
m.umer.tariq@gatech.edu

Brian Paul Swenson
School of Electrical and Computer Engineering
Georgia Institute of Technology
bpswenson@gatech.edu

Arun Padmanabhan Narasimhan
College of Computing
Georgia Institute of Technology
arunpn@gatech.edu

Santiago Grijalva
School of Electrical and Computer Engineering
Georgia Institute of Technology
sgrijalva@ece.gatech.edu

George F. Riley
School of Electrical and Computer Engineering
Georgia Institute of Technology
riley@ece.gatech.edu

Marilyn Wolf
School of Electrical and Computer Engineering
Georgia Institute of Technology
marilyn.wolf@ece.gatech.edu



## ABSTRACT

In this paper, we present a simulation tool that supports co-simulation of the cyber and physical aspects of an electric smart grid. This cyber-physical co-simulator combines a state-of-the-art network simulator, ns-3, and a state-of-the-art power system simulator, PowerWorld. This combination has been achieved by extending ns-3 with a power system module, a physical system interface module, and a cyber-system module. We present the details of these new modules as well as a case study of the application of the co-simulation tool using a demand response scenario.



## Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: General; J.2 [**Computer Applications in Physical Sciences and Engineering**]:

## General Terms

Simulation, Algorithm, Design, Measurement

## Keywords

ns-3, Smart Grid, Network Simulation, PowerWorld, Co-Simulation


## 1. INTRODUCTION

Increasing concerns about global warming and environmental sustainability have resulted in numerous worldwide initiatives towards renewable energy resources such as solar and wind power. Unlike the traditional electricity generation technologies, solar and wind power are inherently intermittent in nature. The intermittent nature of new generation sources is not consistent with the design of the traditional electric grid that was designed to incorporate non-intermittent generation sources such as coal-based power plants, gas turbines, and nuclear power plants. In the traditional power grid, there is no electric energy storage. Therefore generation must always match to the load. This is done by controlling the power output of certain generation sources. This power grid design can not accommodate a high penetration level of new renewable energy resources. The mismatch has led to a vision for a smart grid. Smart grid initiatives envision overlaying the existing electric grid with a more extensive communication, computation, and sensing infrastructure that will enable the grid to handle a higher penetration of renewable energy resources without compromising reliability of service.

Implementation of the proposed vision for a smart grid will result in a complex system with a lot of complicated interactions between traditional power infrastructure and the new communication infrastructure. Analytical solutions are not sufficient for helping us understand the behavior of this new infrastructure. As a result, simulation has to play a major role in enhancing our understanding of smart grid. Moreover, appropriate simulation tools for smart grid must support co-simulation of power and communication systems. Some promising early attempts have been made at co-simulation environments for smart grid. These attempts combine a network simulator (such as ns-2) with a specific power system simulator. Because traditional network simulators generally do not provide an easily extensible model of application layer of a computing node, these smart grid co-simulation environments are not capable of simulating complex smart grid applications. On the other hand, ns-3, by design, provides a more faithful representation of various layers of a real computing node. In particular, ns-3 supports an easily extensible model for the application layer. Therefore, an ns-3-based cyber-physical co-simulation tool for smart grid can do a better job simulating complex smart grid applications.



*WNS3,* May 07 2014, Atlanta, GA, USA.



http://dx.doi.org/10.1145/2630777.2630785.

In this paper, we present a cyber-physical co-simulation environment for smart grid simulation that has been developed by combining ns-3 with a state-of-the-art power system simulator, PowerWorld. This combination of ns-3 and PowerWorld has been achieved by making various extensions to ns-3. In particular, we have added three new modules to ns-3: a power system module, a physical system interface module, and a cyber system module. We also illustrate the utility of this cyber-physical co-simulator by simulating a demand response scenario. The remainder of the paper is organized as follows. In section 2, we outline some related work. In section 3, we provide background material on the components of our co-simulation tool: ns-3 and PowerWorld. Section 4 describes our co-simulator implementation and the changes made in ns-3 to incorporate runtime interaction with the PowerWorld simulator. In section 5, we describe our experiments and provide results for a demand response scenario. Section 6 gives the conclusion and future work.

## 2. RELATED WORK

Similar work tying communication and power simulation was performed by Pruckner et al[10]. They used the simulation software AnyLogic. The goal of their work was to examine how packet loss and delay affected the suppliers ability to match ever-changing demand. To model demand, the authors used real load profiles over a three year period. For generation, the authors considered standard power plant types, including coal, gas, and nuclear, as well as solar and wind. The computing and communication models the authors used were self-admittedly fairly simplistic. Communication delay was modeled using a exponential distribution. The authors also considered packet loss using a uniform distribution. Our work differentiates itself from this work by using dedicated network and power simulators which provide much more detailed and accurate models.

Ngyuen et al[9] examined the effect of communication latency on smart grid performance. They assumed that the communication network bandwidth within the network was great enough to handle the throughput of the system, i.e. no packet drops due to overflowing queues. Also the authors did not attempt to model the individual network devices in any detail. For modeling communication latency the authors used a simple discrete probability density function. By using ns-3, our simulation will allow us to model the communication infrastructure in much more detail. It will allow us to model specific applications that the nodes will be running and model network communication down to the level of individual packets.

Mets et al used OMNeT++ to simulate a combined communication and power network[8]. They used the existing INET network libraries available in OMNeT++ and created new power system models. Specifically they examined a scenario made up of residential houses. A percentage of the homes were given batteries and a percentage of the homes were given photovoltaic devices. Each house was also given an energy consumption value. During the execution of the simulation, various parameters including house consumption and solar irradiance were varied. The simulation produced a variety of data however they presented data showing how voltage at a particular house varied over a six hour window. Our work differs from this work in that we are using a dedicated, industry proven, power solver, which will allow us to simulate much larger and more complex power networks. As mentioned previously, PowerWorld is capable of efficiently solving systems of up to 100,000 buses.

Godfrey et al[5] also used a co-simulation approach to simulate smart grid performance. They used OpenDSS, an open source electrical system simulation tool, and ns-2, the precursor of ns-3, to create a smart grid simulator. The overall approach the authors used is very similar to ours. In their work the authors examine a situation where the output of photovoltaic devices is reduced, due to clouds for example, and messages are dispatched to remote energy storage devices to release electricity to compensate for the loss. The authors use ns-2 to simulate the transfer of wireless messages between the solar devices and the storage devices. When the messages arrive at the storage devices, they generate OpenDDS scripts to update the power simulator and perform a power flow analysis.

## 3. COMPONENT SIMULATORS

In this section, we discuss the two simulation platforms that we have incorporated into the co-simulator.

### 3.1 PowerWorld

Power world is a power system simulator for simulating high voltage power systems[2]. The simulator is capable of efficiently solving systems up to 100,000 buses. The power flow analysis tool, provided by PowerWorld, uses a full Newton-Raphson algorithm, with non-divergence control. The software also contains modules for contingency analysis, optimal power flow analysis, and transient stability. PowerWorld is only available on Windows Operating Systems.

A PowerWorld add-on, called SimAuto, allows interaction with PowerWorld from external software through its Simulator Automation Server, which functions as a Component Object Model (COM) server in the Windows Operating System environment. COM is a standard which allows a way of implementing objects that can be used in environments different from the one in which those objects were created. In particular, a COM server exposes its services by implementing COM interfaces, and any COM client can access these service by calling these interface methods through a pointer to the COM server object.

### 3.2 ns-3 for Windows

To model the communication network in our simulations, we used the ns-3 network simulator[1]. ns-3 is a discrete-event simulator and comes with numerous models for all levels of the networking protocol stack. Since Power World is a Windows only product, we decided to use the Visual Studio 2012 version of ns-3, which is based on ns-3 version 3.18[3]. Unlike the previous Visual Studio 2010 release, the changes made for this version are intended to become merged into the main development tree. The code has been tested and it passes all unit and regression tests included in ns-3 development repository. The release is fully functional with the exception of certain third party and Linux-centric libraries.

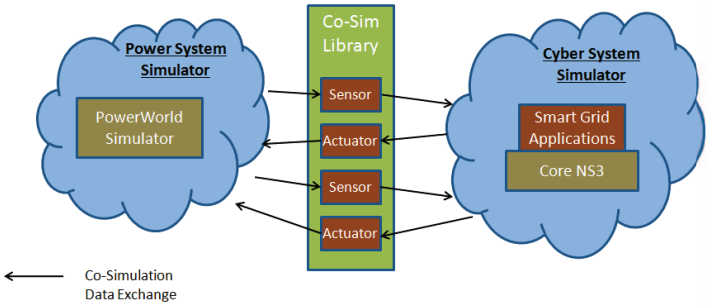


**Figure 1: High level overview of the cyber-physical smart grid co-simulator**

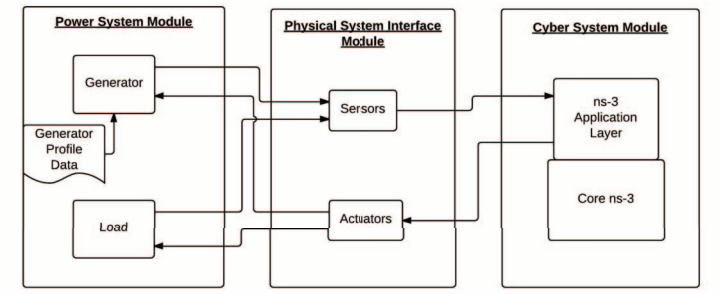


**Figure 2: Data flow within the modules of the co-simulator**

## 4. DESIGN AND IMPLEMENTATION OF CYBER-PHYSICAL CO-SIMULATOR FOR SMART GRID

In this section, we describe the design and implementation of our smart grid co-simulator. As shown in Figure 1, the design of the co-simulator is based on the integration of a network simulator, ns-3, and a power system simulator, PowerWorld. This integration has been achieved by adding three new modules to the ns-3 code base: a power system module, a cyber system module, and a physical system interface module. Figure 2 shows these three modules and the dataflow between the major entities involved in these three modules. In figure 3, we show a UML class diagram depicting the major classes involved in the design of the cyber-physical co-simulator.

The power system module provides two major classes: *PowerSystem* and *PowerWorldWrapper*. The *PowerSystem* class contains lists of power system entities such as bus, branch, generator, and load. The *PowerWorldWrapper* class serves as a wrapper around the PowerWorld simulator. As explained in the last section, PowerWorld allows us to programmatically access its functionality through a COM server, called the Simulator Automation Server. The *PowerWorldWrapper* class is essentially a wrapper over the COM client that talks to the Simulator Automation Server. The main purpose of this wrapper class is to hide the underlying COM functionality. This wrapper class exposes PowerWorld-specific entities like loads and generators as C++ objects and PowerWorld-specific operations like power flow analysis as C++ functions. These objects and functions can then be used by the rest

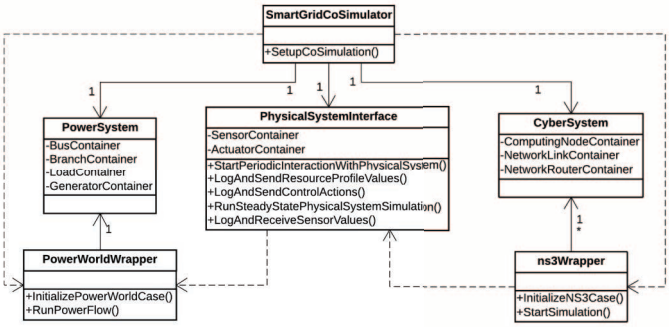


**Figure 3: UML Class Diagram for Co-Simulator**

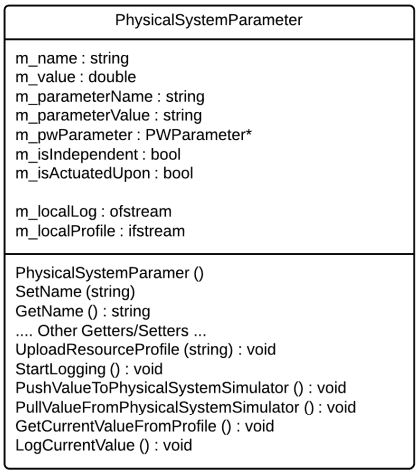


**Figure 4: Physical System Parameter Class Diagram**

of the co-simulator code. The *PowerWorldWrapper* class allows us to create and modify these power system objects. Moreover, it enables us to request PowerWorld to perform various services such as Newton-Raphson power flow analysis on a certain power system scenario.

The cyber system module provides two major classes: *CyberSystem* and *NS3Wrapper*. The *CyberSystem* class contains a list of cyber system entities such as computing nodes, network links, and network routers. The *NS3Wrapper* class serves as a wrapper around the standard ns-3 code. The *NS3Wrapper* class sets up the ns-3 simulation scenario based on the information that it receives through a *CyberSystem* object.

The physical system interface module provides four major classes: *Sensor*, *Actuator*, *PhysicalSystemParameter*, and *PhysicalSystemInterface*. In our co-simulation infrastructure, computation tasks running at the ns-3 application layer use these *Sensor* and *Actuator* objects to interact with the physical system model. *Sensor* and *Actuator* classes hold an instance of the *PhysicalSystemParameter* class, which represents the physical system entity sensed by a sensor or

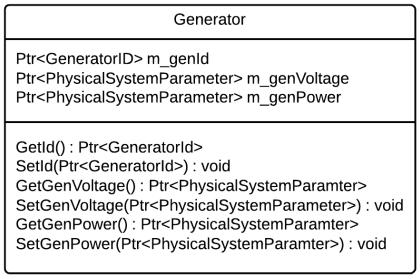


**Figure 5: Generator Class Diagram**

actuated upon by the actuator. In our co-simulation infrastructure, an attribute can only be transferred between the two component simulators (ns-3 and PowerWorld) if it is modeled as an instance of *PhysicalSystemParameter* class. The *PhysicalSystemParameter* class, see figure 4, contains the value for the attribute itself as well as other information regarding how the parameter is used in the simulation. The `m_pwParameter` member contains the needed information to find the corresponding value in the physical system simulator through the PowerWorld COM interface. Moreover, *PhysicalSystemParameters* can also be configured to output log files containing the time-stamped values of the parameter for a simulation run. Figure 5 shows the UML diagram of the *Generator* class as an example of how *PhysicalSystemParameters* are used within the power system module.

The *PhysicalSystemParameter* object can be configured to be independent or dependent. Independent parameters are values that are either set by an ns-3 application using an *Actuator* object or are updated by an external profile, such as a historic wind profile. Dependent parameters are values that are read from the physical system simulator using a *Sensor* object. An example of a dependent parameter would be the line flow on a transmission line after PowerWorld has completed power flow analysis. An example of an actuated independent parameter would be the desired MW value of a load. This value could be increased or decreased by an *Actuator* within ns-3 in response to a change in available power.

In the proposed co-simulator, the interaction between the cyber and physical system simulators is done on a periodic basis through the *PhysicalSystemInterface* class. Listing 1 shows the periodic interaction procedure. In the first step, **LogAndSendResourceProfileValues**, all physical system parameters that are configured as independent and not actuated upon have their current value uploaded to the physical simulator. These values are typically values from historical time-stamped data, such as wind data for wind farms or solar data for photovoltaic generators. In the next step, **LogAndSendControlActions**, all variables that are modifiable by ns-3 computation tasks are uploaded to the physical simulator. After all of the input values have been updated to the physical system, the **RunSteadyStatePhysicalSystemSimulation** step tells PowerWorld to run the power flow analysis. Next, the **LogAndReceiveSensorValues** step pulls all of the updated values for the dependent physical system parameters from the physical simulator. At this point computation tasks in ns-3 are now able to read the updated physical state through their *Sensor* object. The last step in the procedure is to schedule the next cyber-physical interaction. Throughout this update process, all communication with PowerWorld is made through its COM interface.

**Listing 1: Co-Simulation Communication**

```
void PeriodicInteractionWithPhysicalSystem()
{
  //Load external state values such as
  //wind generator data
  LogAndSendResourceProfileValues();

  //Update actuator values
  LogAndSendControlActions();

  //Run the power flow
  RunSteadyStatePhysicalSystemSimulation();

  //Update sensor values
  LogAndReceiveSensorValues();

  //Schedule next update
  Schedule(&PeriodicInteractionWithPhysicalSystem)
}
```

# 5. CASE STUDY: A DEMAND RESPONSE SCENARIO

A difficulty with managing the electric grid is that the supply of electricity must always match the ever changing demand. This difficulty is compounded with the integration of renewable resources into the power grid. Due to inadequate storage technologies, these renewable generators can only be utilized when the wind is blowing or the sun is shining. Therefore, in order to ensure that there is enough supply to meet demand peeks, electricity producers must either build enough of the traditional non-intermittent power stations, such as coal or gas, or find a way to shift elastic loads from one time to another. In order to achieve this control on the energy consumption, demand response applications have been developed. These applications either directly control the consumer-side devices or provide price signals to the electricity consumer. The goal of demand response is to achieve a load-shaping capability where demand-side resources are managed to meet the system's current generation capacity [6]. Numerous approaches for demand response have already been proposed including [4] and [7].

Demand response applications heavily rely on communication between the energy supplier and energy consumer. In a typical situation, consumers need to be notified quickly enough so they are able to respond to the price signal. In an emergency situation, load must be shed as quickly as possible to avoid a disastrous system response, such as a voltage drop. Therefore in order to run effectively, a demand response system needs a robust communication system and can be heavily impacted by communication delays and failures.

## 5.1 Demand Response Simulation

In this section, we illustrate the utility of the proposed smart grid co-simulation tool by simulating a demand response application scenario. Figure 6 shows the power schematic for this scenario. The topology consists of three buses, two generators and a load. The generator at bus two is a wind generator that produces an ever-changing power

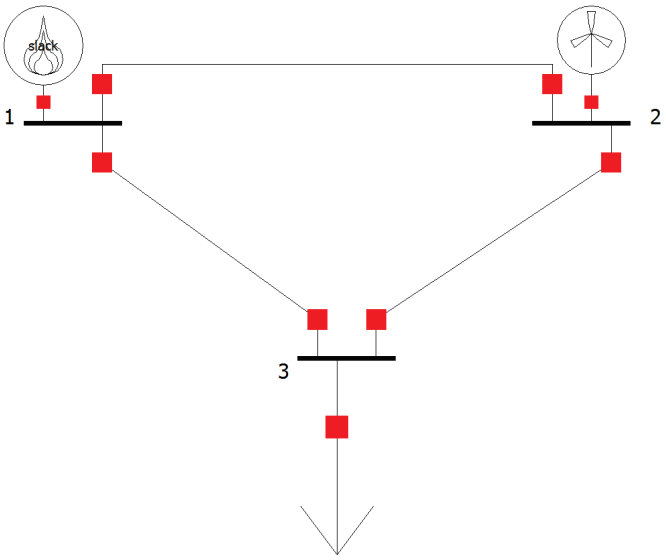


**Figure 6: Power System Topology for Demand Response Scenario**

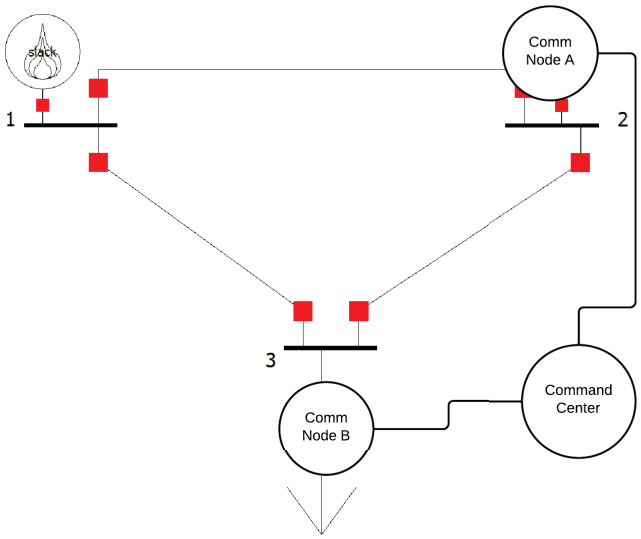


**Figure 7: Communication System Topology for Demand Response Scenario**

output based on the wind behavior. Using demand response, the load at bus three attempts to follow the power output at bus two by changing its demand to meet the available supply. However, due to communication delays, the load at bus three cannot perfectly follow the generator's output. Since the power generation must be equal to the sum of power consumption and power losses, the difference is picked up by the gas powered slack bus at bus one.

Figure 7 shows the communication topology for this scenario. We have added communication nodes at bus two and bus three. There is also a command center through which the two nodes communicate. This type of communication topology is consistent with current SCADA network typologies commonly seen in power systems. For modeling our communication system, we used ns-3's *PointToPointLinks* and corresponding net devices. Communication between the nodes is done using UDP sockets. UDP is the best choice here because the nodes want the latest information and therefore packet re-transmissions of lost previous data values is unnecessary.

During the simulation, Comm Node A reads the wind generator's output using its sensor and sends the value to Comm Node B. The rate at which Comm Node A sends power values to Comm Node B is configurable so we can study the effect of different rates. Whenever Comm Node B receives a new value, it uses its load actuator to set the desired power. Currently it is assumed that the load is able to respond without delay. In future simulations, we will adjust this assumption depending on the scenario.

We have used the co-simulation tool to investigate the effect of link delays, packet drops, and application update periods on the performance of demand response in our scenario. Figure 8 shows the power output of wind generator at bus two and power consumption at bus three for various link delays. As expected, the load is slower to respond to the changes in wind generator power output as the link delay increases. In figure 9, we show the behavior of the slack gas generator at bus one for various link delay values.

Figure 10 shows the power generation profile of wind and power consumption profile of load for various packet drop rates. However, it can be seen that performance of the demand response deteriorates significantly only for a very high packet drop rate. This occurs because the demand response algorithm is inherently more sensitive to the application update period than the packet drop rate. In Figure 11 and Figure 12, we explore the effect of various application update periods on the performance of demand response algorithms. Figure 11 shows the power output of wind generator at bus two and power consumption at bus three for various application update perods, while Figure 12 shows the voltage profile at load bus (bus3) for various application update periods.

## 6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a cyber-physical co-simulation tool for smart grid scenarios. The tool has been developed by combining a state-of-the-art network simulator, ns-3, with a state-of-the-art power system simulator, PowerWorld. We have used the proposed co-simulation tool for investigating the effect of link delays, packet drop rates, and application update period on the performance of a demand response scenarios. Although results have been presented for a simple smart grid application (demand response), the proposed co-simulation infrastructure is uniquely qualified to simulate complex smart grid applications, because of the easily extensible application layer model provided by ns-3. In the future, we plan to develop an XML-based front-end for this simulator that will help to easily set up the co-simulation tool for various large-scale smart grid scenarios.

## 7. ACKNOWLEDGMENT

This research was supported by the United States Department of Energy under grant DE-AR0000225.

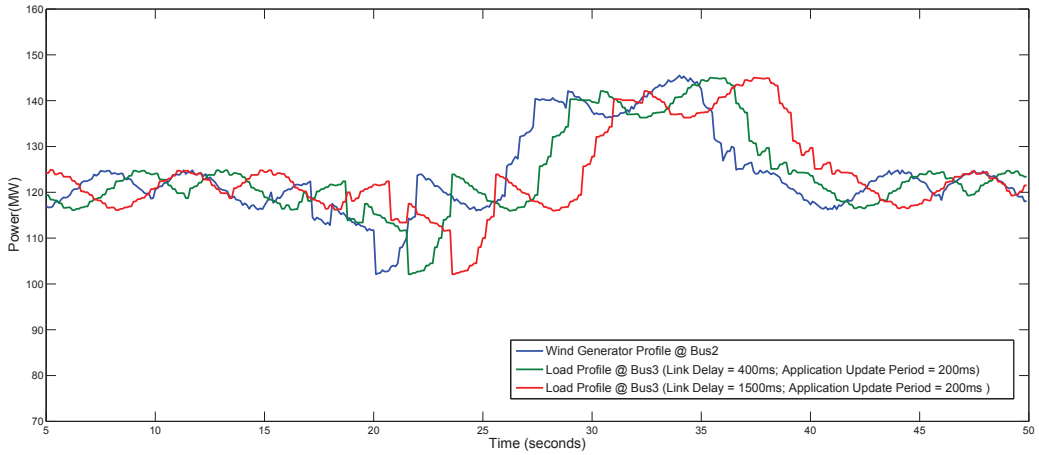


**Figure 8: Power Generation and Consumption Profiles for different Communcation Delays**

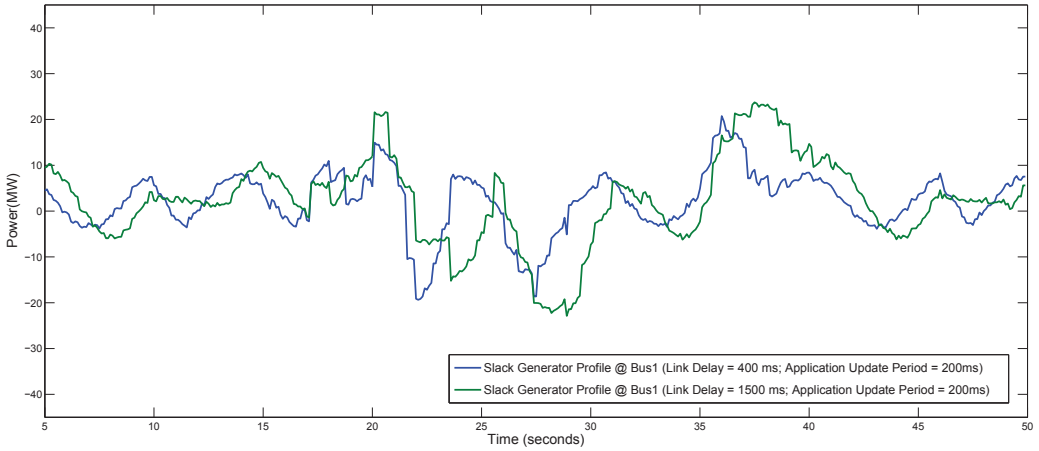


**Figure 9: Slack Generator Profile for different Communication Delays**

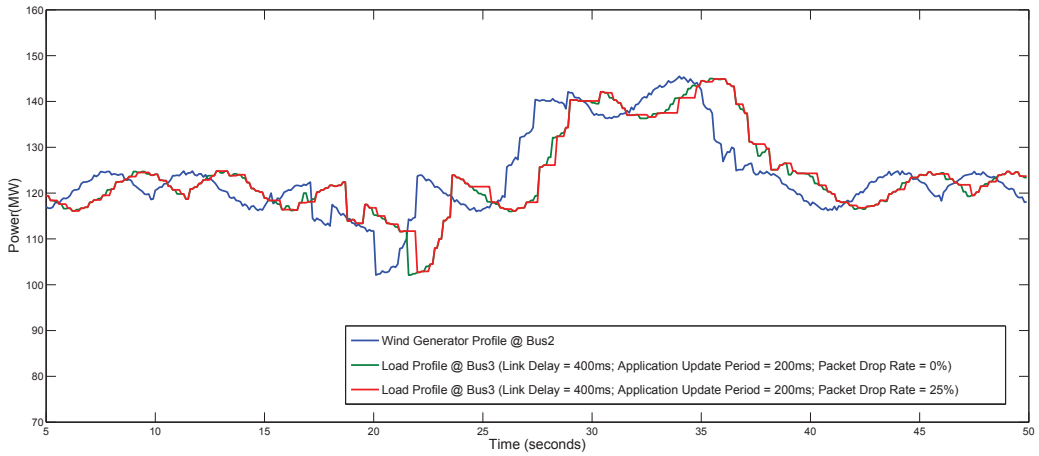


**Figure 10: Power Generation and Consumption Profiles for different Packet Drop Rates**

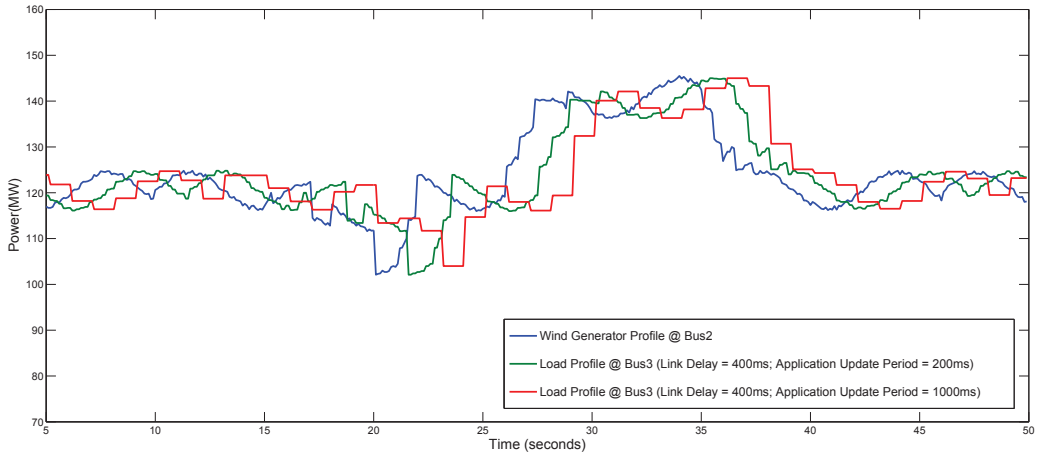


**Figure 11: Power Generation and Consumption Profiles for different Application Update Periods**

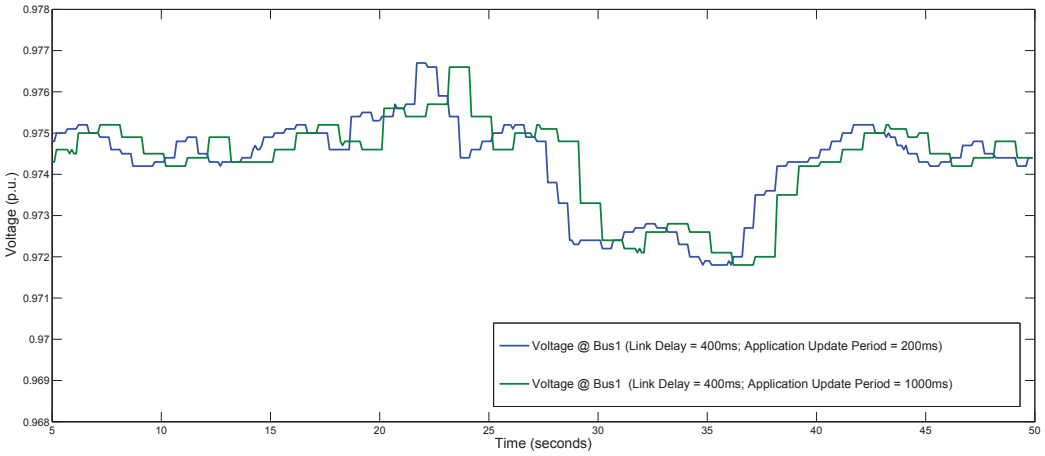


**Figure 12: Voltage at Bus 1 for different Application Update Periods**

## 8. REFERENCES


[1] ns-3 manual (release 15). http://www.nsnam.org/docs/release/3.15/doxygen/index.html [Accessed 06/19/2014].

[2] Power world simulator. http://www.powerworld.com/products/simulator/overview [Accessed 06/19/2014].

[3] J. Abraham and B. Swenson. ns-3 on visual studio 2012. http://www.nsnam.org/wiki/Ns-3_on_Visual_Studio_2012, 2013.

[4] A. Conejo, J. Morales, and L. Baringo. Real-time demand response model. *IEEE Transactions on Smart Grid*, 1(3):236–242, 2010.

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

[6] A. Ipakchi and F. Albuyeh. Grid of the future. *IEEE Power and Energy Magazine*, 7(2):52–62, 2009.

[7] J. Medina, N. Muller, and I. Roytelman. Demand response and distribution grid operations: Opportunities and challenges. *IEEE Transactions on Smart Grid*, 1(2):193–198, 2010.

[8] K. Mets, T. Verschueren, C. Develder, T. Vandoorn, and L. Vandevelde. Integrated simulation of power and communication networks for smart grid applications. In *16th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 61–65, 2011.

[9] C. Nguyen and A. Flueck. Modeling of communication latency in smart grid. In *IEEE Power and Energy Society General Meeting*, pages 1–7, 2011.

[10] M. Pruckner, A. Awad, and R. German. A study on the impact of packet loss and latency on real-time demand response in smart grid. In *GLOBECOM Workshops*, pages 1486–1490, 2012.