S^{2}Sim: Smart Grid Swarm Simulator

Alper Sinan Akyurek*, Baris Aksanli†, Tajana Simunic Rosing‡
University of California, San Diego
*aakyurek@ucsd.edu, †baksanli@ucsd.edu, ‡tajana@ucsd.edu

***Abstract*—The Smart Grid is drawing attention from various research areas. Distributed control algorithms at different scales within the grid are being developed and deployed; yet their effects on each other and the grid's health and stability has not been sufficiently studied due to the lack of a capable simulator. Simulators in the literature can solve the power flow by modeling the physical system, but fail to address the cyber physical aspect of the smart grid with multiple agents. To answer these questions, we have developed S^{2}Sim: Smart Grid Swarm Simulator. S^{2}Sim allows any object within the grid to have its own independent control, transforming physical elements into cyber-physical representations. Objects can have any size ranging from a light bulb to a whole microgrid and their representative data can be supplied from a real device, simulation, distributed control algorithm or a database. S^{2}Sim shields the complexity of the power flow solution from the control algorithms and directly supplies information on system stability. This information can be used to give feedback signals like price or regulation incentives by virtual coordinators to form closed-loop control. Using three case studies, we illustrate how different distributed control algorithms can have varying effects on system stability, which would go undetected in the absence of our simulator. Furthermore, the case studies show that a control algorithm cannot be justified without being tested within the grid picture.**

I. Introduction

With the growth in information technology and increasing demand for power, interest in the smart power grid has risen rapidly. As smarter loads, devices, appliances, storage elements and generators, or, in general, a swarm of *objects* with sensing and/or actuation capabilities connect to the grid, the need for scalable, stable and distributed control algorithms rises rapidly. There is large body of research on the control of both the client side and the utility-provider side of the smart grid separately. This multi agent system is shifting the physical electrical grid into a Cyber Physical System (CPS). One of the most important aspects of the smart grid is the electrical stability of the system. The classical power grid model has more concrete separation of the demand and generation sides. The smart grid, in contrast, with its CPS side of distributed control, distributed generation and energy storage devices [7] is forced to be smarter to address the dangers of instability that can cause major problems as blackouts and equipment damage.

The elements in the smart grid are moving towards a more autonomous and distributed structure, with more diverse control algorithms. Home automation [20], office buildings with HVAC controllers, microgrids, datacenters using Photovoltaic (PV) [9] or energy storage devices at substation levels [10] are examples of increasing autonomous decision making within the grid. But, majority of the control algorithms are designed from a local perspective, modeling the grid as an uninterruptible power supply. This approach has two major flaws: 1) The cross-effect of multiple controllers on each other is not studied, 2) The cumulative-effect of the control decision on the grid itself is left out. It is crucial to test and evaluate any control solution not only from its own point of view in an isolated environment, but also with respect to the big picture of the constituent smart grid in order to get a more realistic success metric. Recently, a white paper from a multi-institutional collaboration mentions a need for a smart grid simulator that can connect loads from different physical locations, including real hardware to bring the physical aspect into the loop [14]. In order to achieve these goals, there is a need for a smart grid simulation tool, which can handle the swarm of objects with distributed, diverse (possibly heterogeneous) control algorithms in a dynamic fashion, without introducing any constraints on the objects.

In order to address these needs, we designed and implemented S^{2}Sim , Smart Grid Swarm Simulator. S^{2}Sim allows real-time co-simulation of distributed control algorithms within the smart grid and studying the grid's behavior and health under various desired conditions. To the best of our knowledge, existing simulators in the literature either don't support dynamic, real-time object behavior [3] or constrain the object control strategies to predefined libraries with predefined behavior [1][2][4]. Section II has a detailed analysis of existing tools and their limitations. The main contributions of this paper are as follows:

- A smart grid simulator, capable of evaluating independent distributed control algorithms to analyze stability and control issues in the smart grid with heterogeneous objects connected to it. The simulator shields the complexity of the non-linear power flow equations from the control algorithms.
- A multitude of objects within the grid can be represented as an external (possibly real-time) data stream, a real hardware, simulation code or control algorithm over a reliable TCP/IP connection. These objects can represent any type of grid element, ranging from loads, generators, micrograds to energy storage elements at any scale, such as a single light bulb or a whole microgrid. In contrast to classical simulators, the objects enable the simulation of the CPS aspect of smart grid.
- Multiple coordinators can connect and access system-wide information to emulate coordination logics such as the microgrid or a home control hub. These coordinators can provide feedback signals such as real-time pricing or

stability related information to the objects.

- S^{2}Sim handles time synchronization among objects despite their different time constants such as an air conditioner and a photo-voltaic (PV) cell simultaneously.
- Our simulator provides an application layer communication protocol for remote access over any network interface. This enables objects that are physically distant from each other to form a virtual grid, enabling parallel and remote computing capabilities. We used this property to perform a US-wide case study.

II. Related Work

There are various smart grid power flow simulators in the literature: open source simulators OpenDSS [3] and GridLab-D [1] or commercial products as RTDS [4] and Paladin Live [2]. The objects in these simulators are static objects with fixed behavior, predefined with a time series throughout the simulation. This static behavior prevents any reaction from either the objects or the utility, making it impossible to co-simulate distributed control algorithms. The only way to overcome this is to set the simulation time to a single step and readjust the scenario for the next time step. One common point of the mentioned simulators is that they can all solve the complex non-linear power flow equations efficiently.

OpenDSS and GridLab-D represent the grid by impedances and lines connecting them. There are two ways to control object behavior. The first pre-loads the object behavior as time series before the simulation. The second uses a Dynamic Link Library (DLL) that represents the object behavior during the simulation. The main disadvantage of the first method is the static simulation, where the objects cannot react to anything due to preset object behavior. The second method adds dynamism to the object behavior, but is constrained by the implementation guide of the DLL.

None of the simulators have an interface for a coordinator that can give feedback signals like price or regulation incentives back to the objects. These simulators are thus limited to an open-loop control in nature. RTDS is very powerful in terms of connecting actual devices to the simulation environment. But it is again limited to the libraries provided by the simulator and thus constrains the control application scenarios. Paladin Live allows real-time system monitoring and provides tools to analyze the system health. However, its simulation mode is for general power system design and is not able to do distributed control simulations.

Other studies on specific load models and their real time simulations also exist in the literature [12], but they fail to consider general and heterogeneous control cases, but rather concentrate on specific scenarios. In [11], the authors introduce a real-time combined power flow simulator and electromagnetic simulator, but the scenarios and the system are all static, i.e. flow of simulation is preset before run-time. In [15], Real Time Digital Simulator (RTDS) [4] has been used to simulate a fuel cell vehicle, where the operation is limited to the specific load scenario of a fuel cell vehicle.

As a summary, existing simulators have very powerful non-linear power flow solvers that can calculate the voltage drops efficiently in the physical system. Yet, they lack the ability and the interface to connect and test dynamic online scenarios, distributed control algorithms, reactive control algorithms and feedback based (closed-loop) control algorithms, representing the emerging cyber physical aspect of the smart grid. Furthermore, classical simulators fail to address time synchronization since the scenario is a static simulation. To answer all these missing points and still maintain the powerful aspects, we have developed S^{2}Sim.

III. S^{2}Sim Architecture

The classical power grid is a network of many different grid elements connected to each other over the electrical lines. This graph is mostly represented by an impedance matrix. We use this *physical* circuit as the basis of our architecture. But, with the emergence of Smart Grid, we need to add additional concepts on top of the physical electrical circuit in order to represent the resulting *CPS*.

The first concept we introduce with S^{2}Sim in the *object*. An object is the cyber/virtual representation of a physical circuit element. It controls the behavior and the loadshape of the physical element it is representing. It is of crucial importance to represent these elements correctly for power system simulation [6]. The second concept is the optional component of *coordinator*. The coordinator is a completely virtual entity, which implements the feedback logic that will be present in the CPS. Figure 1 shows an example scenario for the overall architecture of S^{2}Sim.

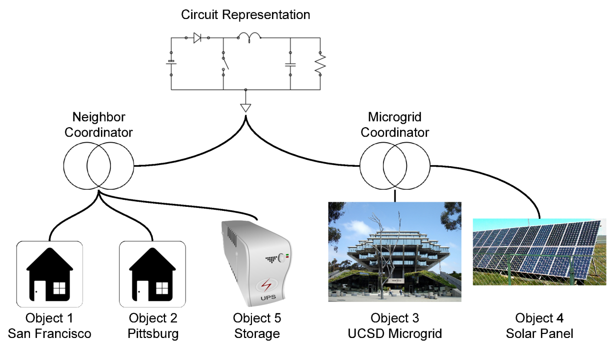

Fig. 1. Example Architecture of 3 main elements

A. Object

An object is the cyber representation of any physical entity defined on the physical circuit. It can represent any type, such as loads, generators, energy storage devices or a combined system as a single entity. Objects can be of any size in the grid, ranging from a toast machine to a whole microgrid. Objects can be self-aware and implement distributed control algorithms to adjust their behavior, such as real-time consumption of a building, output of a solar panel, charging characteristic of a battery or the output of any simulation. Object behavior is controlled over a TCP/IP communication interface, which allows it to be virtually anywhere. S^{2}Sim does not implement the behavior of the objects, but provides the communication

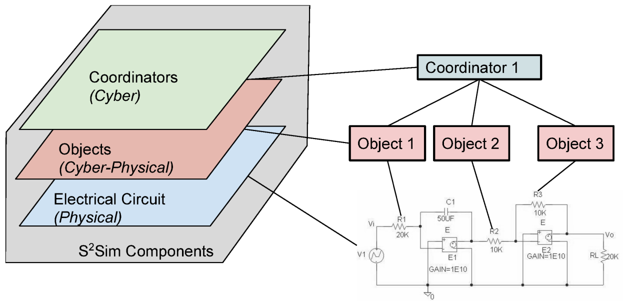

Fig. 2. CPS Layers of S^{2}Sim.

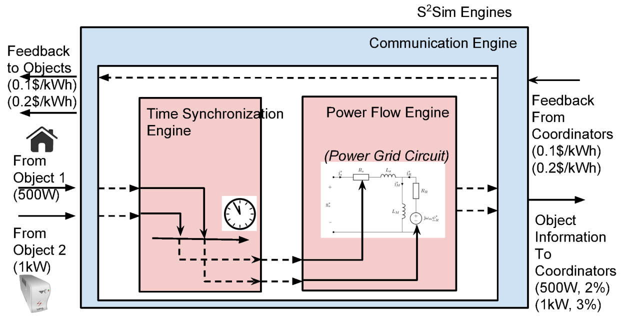
S²Sim Engines
Communication Engine
Feedback to Objects (0.1$/kWh) (0.2$/kWh)
Time Synchronization Engine
Power Flow Engine
(Power Grid Circuit)
From Object 1 (500W)
From Object 2 (1kW)
Feedback From Coordinators (0.1$/kWh) (0.2$/kWh)
Object Information To Coordinators (500W, 2%) (1kW, 3%)

Fig. 3. Functionalities of S^{2}Sim divided among its internal components.

framework the objects have to use in order to co-simulate their outputs. The minimum requirement for an object implementation is for the object to output consumption values, which will be used to adjust the physical representation of it. There is no limitation on the frequency of the output as all time synchronization is done by S^{2}Sim.

B. Coordinator

A coordinator is a special virtual element that can oversee and get information regarding the whole or a part of the grid at any time. It is an optional component that provides feedback information to the objects, such as dynamic pricing or stability related sensor information. As an example, it can represent the grid perspective, a home control hub or a microgrid coordinator and serves as a feedback provider to its intended operation region. The coordinator constructs the missing link in a closed-loop control scheme, providing various feedback signals required for normal daily operation on different scales. The simplest of these is price. Each coordinator has a different strategy for different types of consumers. Another common signal is regulation incentive, which guides the consumption of participating customers by giving incentives. The specific implementation of a coordinator is external to S^{2}Sim , but requires a specific communication framework to connect to the simulation. The complex solution of the power flow equations is completely shielded from the coordinators and is handled by S^{2}Sim.

C. Electrical Circuit

The only physical and static part of the simulator is the electrical circuit itself. The circuit represents the networked connection of objects and is determined statically before the simulation starts. This component represents the classical power flow simulators. Any electrical grid item can be defined ranging from loads, generator, energy storage devices to transformers and circuit breakers. The electrical circuit is implemented within the simulator and is one of the essential components. The three layers of components of the composite simulated CPS is shown in Figure 2.

D. Internal Architecture

For modular structure and flexibility, the internal structure of S^{2}Sim has three major engines, corresponding to three major tasks: Communication, Time Synchronization and Power Flow Engines. Figure 3 shows how these engines interact with each other as well as the external components, namely objects and coordinators. At any time, the simulation's information flow starts from the objects into the simulator. The incoming information is parsed and processed through the Communication Engine and is supplied into the Time Synchronization Engine. The data is then time filtered and time synchronized, and passed to the Power Flow Engine in order to obtain the power flow solution for the current time interval. The Electrical Circuit component is modeled within the Power Flow Engine. All obtained information is then forwarded to the respective Coordinators for feedback calculation. Finally, any feedback is sent back to the Objects, closing the information flow loop. During the process, the information flow from/to the external components can happen asynchronously, as this is handled by the Time Synchronization Engine.

1) Communication Engine: Communication engine maintains end-to-end communication using TCP/IP between the objects and S^{2}Sim. Since the exchanged data can be sensitive such as the consumption information of a residential building, we provide an optional security layer with end-to-end encryption. The encryption is performed using Secure Sockets Layer (SSL).

We create a common message structure as an application layer messaging protocol to regulate the communication between the simulator and the objects. The protocol is flexible, extensible, lightweight, low overhead and has minimal dependency on the underlying infrastructure. This protocol establishes the minimum framework required by the implementation of every object. Every communication command is represented by a separate packet. Examples to these commands are registration message, consumption reporting message or price notification message. Although each command has a unique internal structure, all messages have a common header and ending for message identification. The common structure is shown in Figure 4.

Start and End of Message: A communication protocol must be independent of the communication layers underneath it. These two fields mark the start and end of a single message. Since a periodic field may endanger the keys of the encryption, this field must be transmitted unencrypted.

Sender & Receiver IDs: Each object is assigned a unique identification number when it registers to the simulator. This

0	Start of Message	Sender ID	Receiver ID	8
8	Sequence Number	Message Type	Message ID	16
16	Data Size (N)	Unique Structure	End of Msg.	N+24

Fig. 4. Common structure of S^{2}Sim messages.

ID will be used for every communication for end-to-end identification.

Message Type & Message ID: The unique structure of a message is decoded through a two-level hierarchy. Message Type defines the higher level (e.g. System Messages), whereas the Message ID value determines the lower level (e.g. Registration Message). The specific values are defined in the Interoperability Document of the simulator in a separate document, obtainable from the author.

Any control algorithm that wishes to be represented in the system, needs only to implement this communication protocol framework.

2) Time Synchronization Engine: Multiple Objects with various behaviors imply a distributed sense of timing. They may have different time resolutions and time constants. Consider 2 objects: A phasor measurement unit (PMU) connected to a PV and a simulated office building with heating ventilating and air conditioning (HVAC). PMU provides high resolution, on the order of seconds, near real-time data and has a small time constant due to rapid solar variations. In contrast, HVAC simulation has low time resolution, on the order of hours, may provide simulated information for the future and has a large time constant due to the slow adapting nature of thermodynamics. Time Synchronization Engine enables both objects to connect in real-time and be represented in the same simulation environment.

Time Synchronization Engine filters out past incoming data, stores future data and provides the current information. It integrates different resolutions by linear interpolation and time averaging for low and high resolutions respectively. It uses any future information as a prediction and provides it to the coordinator as an input. The prediction can be updated if the actual information changes as time advances. For the previous example, the PMU's measurements are processed in real-time, whereas the low-resolution information from the HVAC is interpolated to obtain the missing points compared to the high resolution PMU. To represent a broad scale of objects, S^{2}Sim defines 2 types of object connections:

Active Connection: The object is time synchronized to the simulator and provides real-time information or future prediction. In return, the object receives feedback information sent by the local coordinator. If the object fails to communicate within a time interval, previously sent information or prediction is used automatically.

Passive Connection: The object connects to the system, uploads bulk consumption data and disconnects. The bulk data is filtered and processed. But the coordinator does not provide feedback, as the object is disconnected and is assumed to be irresponsive to any feedback. This type of connection enables the connection of consumption databases or data sources

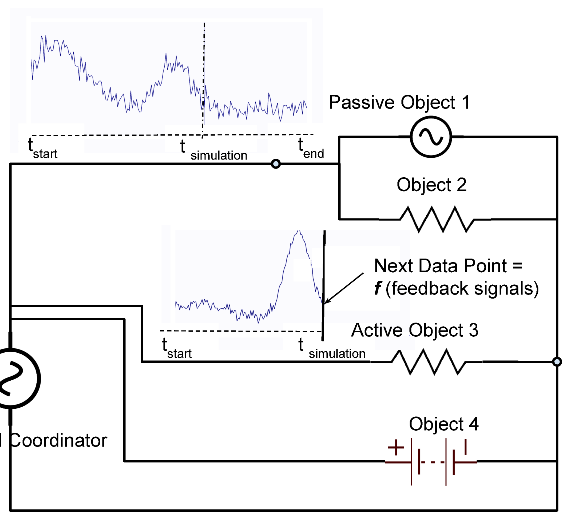

Fig. 5. Example circuit with passive and active connections. Passive connection sends bulk consumption data, whereas the active connection determines the next interval using feedback signals.

requiring no feedback and is an easy way to represent an object without any control or automation.

Figure 5 explains the timing with an example. The passive object provides bulk consumption data for the whole simulation, whereas the active object is time synchronized and determines its behavior based on the feedback signals it receives at every time step.

3) Power Flow Engine: Since one of the purposes of S^{2}Sim is to provide abstraction of the power flow problem to the coordinators, the coordinator may require an additional "sandbox" or "playground" environment. As an example, the coordinator knows the current voltage deviation of an object's terminal and the object's consumption. But it does not know what the deviation of the same terminal or the cross-effect on any other terminal will be if the consumption is changed as the result of a control decision. Simply put, the coordinator cannot calculate possible states of the power-flow under different conditions that the current ones. The playground environment is provided for this purpose. The Power Flow Engine uses the power flow solution interface of OpenDSS [3] over a DLL and constantly maintains two parallel instances of it. The real circuit handled on the first instance is modified only to reflect the actual behavior of objects and any modification represents actual snapshots of the physical circuit. The second instance has the exact same circuit at the beginning of each time interval as the real circuit, but is used as a sandbox to be modified and reset multiple times to get answers to different "What if" scenarios that the coordinator might be interested in.

E. Graphical User Interface

We implemented a web-based graphical user interface (GUI) that visualizes the outputs of our simulation environment. The website can be accessed to get real-time runtime information at http://seelabc.ucsd.edu:26995. The GUI gives information on the system health and all individual objects, in both real-time and historical modes. Sample snapshots of the GUI is showin Figure 6.

Fig. 6. 6 Snapshots from the GUI.

IV. Distributed Control Simulation

In this section, we demonstrate multiple case-studies how our simulator, S^{2}Sim, can be used to show how heterogeneous distributed control algorithms can affect each other and the grid. We first simulate an average sized U.S. town to show that a complete greedy distributed control of loads may lead to unstable conditions given static time of use (ToU) pricing. In response, we show that introducing adaptive pricing heuristic on the coordination side to guide the grid to stable operating regions can avert this situation. In the second case, we use the test bed of a joint project between 6 universities [8] to test a distributed heterogeneous control scenario. Each university from different regions of the United States deploys its own control algorithm. S^{2}Sim combines and synchronizes all objects and provides a smart pricing heuristic from the coordinator to guide the grid to stable operating regions. In the third case, we use HomeSim [20], a residential energy simulator to simulate multiple houses in a neighborhood to test various control strategies.

A. Validation and Performance Overhead

The simulator has been validated against University of California, San Diego campus Microgrid measurements, by comparing measured and simulated voltage deviation information at building terminals.

To give an estimate for the communication overhead, we look at a sample problem size of 100.000 simultaneous objects. At each simulation time step, the default communication overhead is the consumption message from every object to the simulator and, a price and a regulation message from the simulator to every object. The messages are only 28 bytes in total. This results in $56N$ bytes of overhead for N objects in every time step. The default setting runs one time step per second, so for a circuit with 100000 simultaneous objects, this results in $5.6MB/s$ of communication overhead, easily maintainable with an everyday home network.

The processing overhead of the 3 main engines are as follows: Communication Engine has $O(N)$ message processing complexity for parsing and distributing messages. Time Synchronization Engine has $O(N)$ complexity for filtering and interpolation. Power Flow Engine has at least $O(N^3)$ due to the power flow solution. Extra overhead caused by S^{2}Sim besides the power flow solution is only $O(N)$.

B. Time of Use vs. Adaptive Pricing

We use a university campus distribution circuit with both residential and office buildings as the loads. The average total grid consumption is $10MW$, about the size of an average U.S. town with 81 buildings represented as individual objects. Each object runs a distributed control algorithm, unaware of its surroundings or the grid and only uses the price signal provided by the utility to adjust its consumption. The distributed control algorithm of the objects is a greedy heuristic, which adjusts the consumption in proportion with the ratio of the average price to the current price. The remaining consumption is adjusted to fix the total energy consumption, in order to give a fair comparison among different pricing strategies. The algorithm at i^{th} step is given below:

$$\text{AdjustedPower}_i = \text{Power}_i \frac{\text{Avg(Price)}}{\text{Price}_i} \tag{1}$$

$$\text{Power}_j = \text{Power}_j + \frac{\text{AdjustedPower}_i - \text{Power}_i}{N-j+1}, \forall j \in (i, N) \tag{2}$$

This scheme is a simple heuristic assuming an energy storage device connected to the load, capable of reacting to price changes. We consider two pricing strategies: 1) Completely static pricing, open loop without feedback and distributed control case; 2) adaptive consumption, dynamic price guided, distributed closed control loop case.

Static pricing uses a ToU pricing scheme with 3 price regions dividing the day into 4 intervals representing peak, off-peak and super off-peak hours [5]. The price is static as it doesn't react to the state of the grid and is the same for every object. Adaptive pricing computes a dynamic price for each individual object. The heuristic uses the information of object's terminal voltage deviation as a stability metric, then multiplies it with the object's current consumption and maps the value to a price range. The heuristic not only penalizes high consumption, but also takes into account the voltage deviation, which is affected by every object in the grid. High deviation caused by any object thus has a higher price effect on all objects, yet the object that has caused the condition will have the highest penalty. To avoid rapid variations in pricing, we pass the immediate price values through an exponentially weighted moving average filter to smooth out the price decisions. We take the maximum voltage deviation within the grid as our stability metric and mark the widely accepted 10% value as the limit of danger and start of instability.

Figure 7 shows that the result of combined greedy behavior under ToU pricing in a completely distributed scenario leads to unstable system behavior, pushing the voltage deviation beyond its safe limits. The initial spike is largely due to the fact that the controllers are unaware of each other and react to the low price in a greedy manner.

Figure 8 shows the results for the adaptive pricing scheme. As with previous results, there is a spike in consumption due to the greedy distributed control in the low price region. However, the price adapts to consumption and stability values and, guides the system to be within stable boundaries to avoid instability. Although both consumption and pricing control algorithms are simple heuristics, we show that good performance for a control algorithm under isolated conditions is misleading. S^{2}Sim enables each algorithm to be designed and simulated within the greater picture of the grid, exploring cross-correlated effects in more depth.

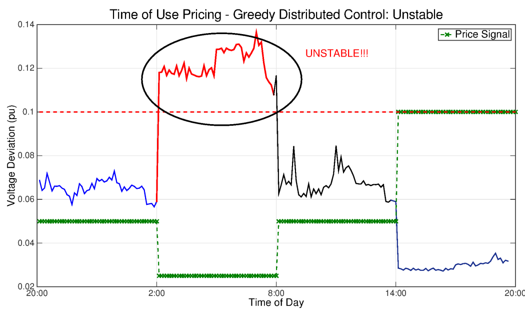

Fig. 7. Time of Use Pricing resulting in unstable system behavior.

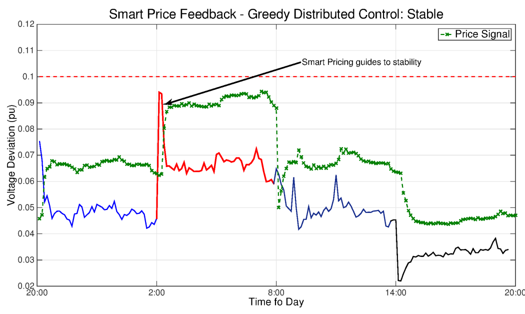

Fig. 8. Adaptive Pricing resulting in stable system behavior.

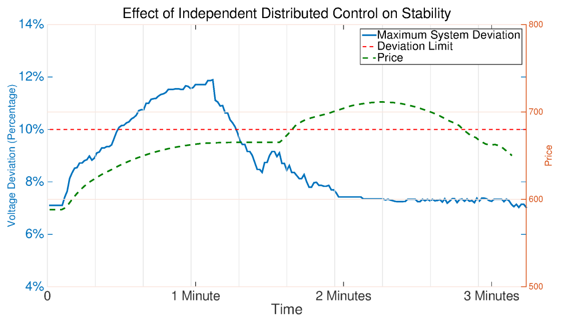

Fig. 10. Effect of Distributed Heterogeneous Control on system stability.

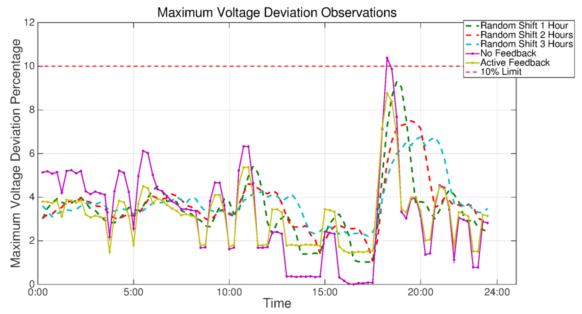

Fig. 11. Effect of Distributed Heterogeneous Control on system stability.

C. Distributed Heterogeneous Control

We use a university Microgrid circuit with 12 major buildings represented by a combination of real and simulated objects from 6 different universities. Their physical locations are in California, Michigan and Pennsylvania, all connected remotely over TCP/IP to S^{2}Sim. We use home automation controller simulation [20], actual battery bank controller [16], real-time consumption of an actual building with actuation [19] and 3 different HVAC control simulations with different strategies [13][17][18], summarized in Figure 9. We use the same heuristic pricing as in the previous section.

Figure 10 shows that the independent distributed controllers increase their consumption leading in an increasing voltage deviation (solid line) within the system, endangering the system health by coming close and exceeding the 10% limit (horizontal line) for a short time. The adaptive price (dashed line) increases to guide the system back into the stable region and later achieves it. S^{2}Sim in this study shows that, well performing algorithms in isolated situations, may lead to an unstable system, when working together.

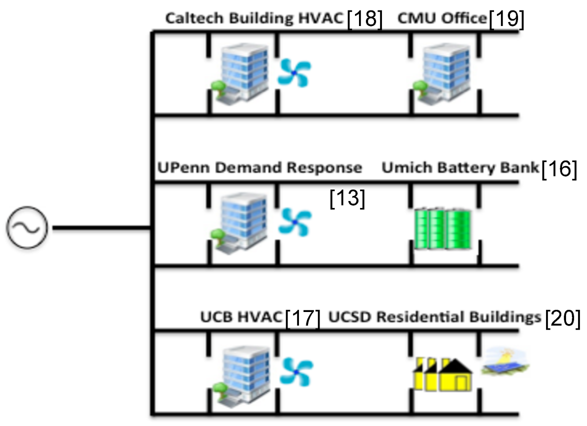

Fig. 9. Joint collaborative simulation of heterogeneous control algorithms.

D. Neighborhood Simulation

To demonstrate the abilities of our simulator even further, we extend the first case study for time of use pricing by eliminating the coordinator entirely and randomizing the consumption intervals of each object in order to distribute the total consumption amount over time. We consider a residential neighborhood with 160 buildings. The consumption values are obtained from a residential simulator called HomeSim [20]. To decrease the probability of a high consumption correlation, each building selects a random shifting amount without any further knowledge and shifts its consumption value by the selected value in time. The random value is a uniformly distributed value drawn from three different intervals for the three cases considered: 1) $[0, 1]$ Hour, 2) $[0, 2]$ Hours, 3) $[0, 3]$ Hours. Furthermore, we use the two algorithms used in the first case, where the buildings implement greedy distributed control and the coordinator is providing static and dynamic pricing feedback. 50 iterations have been averaged to get stable results.

Figure 11 shows the results for all 5 control algorithms considered. The maximum observed deviation values are shown in Figure 12. The only algorithm that fails the voltage deviation limit is the greedy control case with static pricing as in the first case. Furthermore, the greedy control with active feedback

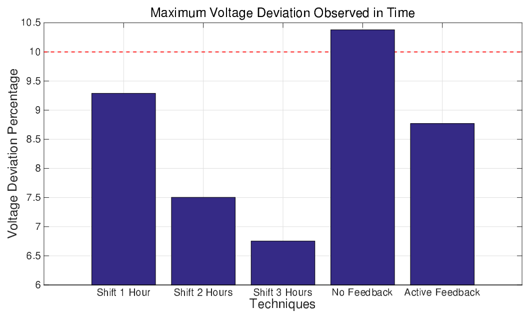

Fig. 12. Effect of Distributed Heterogeneous Control on system stability.

again manages to keep the stability within the limit as in the first case. The additionally tested randomization algorithms without feedback manage to decrease the deviation and the peak is decreased by increased randomization. This is an expected result, as increased randomness results in a more uniform consumption, decreasing the peak consumption in the grid. Although wider randomization intervals decrease the deviation, it is also harder to implement them, creating a tradeoff.

V. Conclusion

The classical physical power grid is transforming into a cyber physical system, the smart grid. Distributed control algorithms for different platforms are being developed and deployed in different scales. Existing grid simulators solve the power flow of the physical aspect of the grid efficiently, but fail to address the co-simulation of distributed control algorithms, thus the CPS aspect of the smart grid. There is a need for a flexible simulator to co-simulate and test independent distributed control algorithms in order to observe their effects on both each other and the health of the system. To answer this need, we have developed S^{2}Sim. S^{2}Sim allows the co-simulation of any object connected over TCP/IP, which can represent any type and any size of grid elements, with distributed independent control strategies. S^{2}Sim takes care of communication, time synchronization and introduces an interface for multiple coordinators to construct closed loop feedback controlled system. S^{2}Sim is extensible, scalable and has low overhead. We present 3 different case studies specifically possible with our simulator, where the first case shows, why it is necessary to have closed loop control for grid stability. The second case shows that we cannot justify the performance of a control algorithm under isolated conditions alone, without testing it within the grid picture. The third case shows that we can use S^{2}Sim to compare the performance of different heuristics using our tool.

References

[1] http://www.gridlabd.org/.

[2] poweranalytics.com/paladin-software/paladin-live.

[3] sourceforge.net/projects/electricdss/.

[4] www.rtds.com.

[5] www.sdge.com.

[6] Standard load models for power flow and dynamic performance simulation. *Power Systems, IEEE Transactions on*, 10(3):1302–1313, 1995.

[7] EPRI Smart Grid Demonstration Initiative Two Year Update. Technical report, Electric Power Research Institue, 2010.

[8] B. Aksanli, A. S. Akyurek, M. Behl, et al. Distributed control of a swarm of buildings connected to a smart grid: Demo abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, BuildSys '14, pages 172–173, New York, NY, USA, 2014. ACM.

[9] B. Aksanli, J. Venkatesh, and T. Rosing. Using datacenter simulation to evaluate green energy integration. *Computer*, 45(9):56–64, Sept 2012.

[10] A. Akyurek, B. Torre, and T. Rosing. Eco-dac energy control over divide and control. In *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, pages 666–671, Oct 2013.

[11] F. Angstmann, A. Bracher, S. Bhat, and S. Ramaswamy. A smart grid simulation framework for electricity trading. In *Networking, Sensing and Control (ICNSC), 2013 10th IEEE International Conference on*, pages 609–614, 2013.

[12] J. Arrinda, J. Barrena, and M. Rodriguez. Distribution network simulation method based on a combination of dynamic power-flow simulation and electro-magnetic simulation. In *EUROCON, 2013 IEEE*, pages 1336–1343, 2013.

[13] W. Bernal, M. Behl, T. X. Nghiem, and R. Mangharam. Mle+: A tool for integrated design and deployment of energy efficient building controls. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '12, pages 123–130, New York, NY, USA, 2012. ACM.

[14] E. Dall'Anese, J. Doyle, S. Dhople, et al. Federated smart grid testbed. Technical report, March 2015.

[15] Y. Deng, S. Foo, and H. Li. Real time simulation of power flow control strategies for fuel cell vehicle with energy storage by using real time digital simulator (rtds). In *Power Electronics and Motion Control Conference, 2009. IPEMC '09. IEEE 6th International*, pages 2323–2327, 2009.

[16] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and implementation of a high-fidelity ac metering network. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, IPSN '09, pages 253–264, Washington, DC, USA, 2009. IEEE Computer Society.

[17] M. Maasoumy, C. Rosenberg, A. Sangiovanni-Vincentelli, and D. Callaway. Model predictive control approach to online computation of demand-side flexibility of commercial buildings hvac systems for supply following. In *American Control Conference (ACC), 2014*, pages 1082–1089, June 2014.

[18] V. Raman, A. Donze, M. Maasoumy, et al. Model predictive control with signal temporal logic specifications. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 81–87, Dec 2014.

[19] A. Rowe, M. Berges, G. Bhatia, et al. Sensor andrew: Large-scale campus-wide sensing and actuation. *IBM Journal of Research and Development*, 55, Jan 2011.

[20] J. Venkatesh, B. Aksanli, J.-C. Junqua, P. Morin, and T. Rosing. Homesim: Comprehensive, smart, residential electrical energy simulation and scheduling. In *Green Computing Conference (IGCC), 2013 International*, pages 1–8, June 2013.