Modeling and Simulation of Network Aspects for Distributed Cyber-Physical Energy Systems*

Ilge Akkaya
UC Berkeley

Yan Liu
Concordia University

Edward A. Lee
UC Berkeley

Abstract

Electric power grids are presently being integrated with sensors that provide measurements at high rates and resolution. The abundance of sensor measurements, as well as the added complexity of applications trigger a demand for cyber-physical system (CPS) modeling and simulation for evaluating the characteristics of appropriate network fabrics, timing profiles and distributed application workflow of power applications. Although simulation aids in the pre-deployment decision making process, system models for complex CPS can quickly become impractical for the purposes of specialized evaluation of design aspects. Existing modeling techniques are inadequate for capturing the heterogeneous nature of CPS and tend to inherently couple orthogonal design concerns. To address this issue, we present an aspect-oriented modeling and simulation paradigm. The aspect-oriented approach provides a separation between functional models and cross-cutting modeling concerns such as network topology, latency profiles, security aspects, and quality of service (QoS) requirements. As a case study, we consider a three-area smart grid topology and demonstrate the aspect-oriented approach to modeling network and middleware behavior for a distributed state estimation application. We also explore how aspects leverage scalable co-simulation, fault modeling, and middleware-in-the loop simulation for complex smart grid models.

1 Introduction

Emerging cyber-physical energy systems (CPES) broadly depend on high-throughput sensor measurements to execute distributed control tasks. Integration of *smart* sensors into the grid enables high-fidelity and trustworthy data to become available at monitoring centers in real-time, which has the potential benefit of dramatically improving the accuracy of existing contingency and state estimation applications, and enabling novel grid control techniques.

Traditionally, the primary concern of power engineers has been to provide correct and efficient design of algorithms that run on power grid hardware. Given the unprecedented volumes of streaming data available on the smart grid, this requirement alone is no longer sufficient. Data volume combined with real-time aggregation and processing requirements of applications bring about a unique challenge for existing communication and data management tools. Deployment of decentralized computation resources and coordination among these become key to realizing the next generation data intensive real-time tasks on the grid [9].

The communication infrastructure for distributed application management encompasses middleware, communication networks, and software platforms. Responsiveness and scalability play an essential role within this framework not only for efficiency, but also for the correctness of the distributed applications. Such applications include distributed state estimation, contingency analysis and grid stability monitoring, which rely on real-time sensor measurements produced at geographically distributed synchrophasor devices.

The qualitative evaluation of this ubiquitous application domain requires systematic modeling formalisms that enable abstraction of system dynamics. The modeling challenges are threefold: (i) defining modeling formalisms, which provide multi-view models that enable a separation of cross-cutting concerns, such as communication fabrics, and implementation-specific middleware characterizations; (ii) composing multiple models in one heterogeneous simulation environment or equivalently, enabling determinate composition of computation models; (iii) correctly representing application-specific timing characteristics of the distributed applications by the aid of a common notion of time among

*The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-662-45928-7_1

distributed model components.

This paper studies a network modeling approach for CPES infrastructure, which leverages separation of cross-cutting concerns in the system, and consequently enables development of scalable and easy-to-analyze CPES models. We demonstrate the benefit of aspect-oriented modeling (AOM) that presents a separation of power domain applications and implementation-specific aspects such as network communication and middleware for fine-grained data coordination. We prototype the AOM methodology using the actor-oriented modeling and simulation environment Ptolemy II (4.1). Ptolemy II is a framework that specializes in addressing intrinsic CPS challenges such as timing, concurrency and heterogeneous composition and has been extensively used for industrial CPS design [11, 25].

To demonstrate the vast capabilities of the aspect-oriented CPES models, we consider a distributed state estimation (DSE) application, which operates on time-synchronized phasor data collected from the transmission grid. We perform simulation studies on a three-area distributed grid architecture, composed with a prototype network and middleware infrastructure, and explore how AOM facilitates design and analysis of network and middleware requirements for time-critical distributed applications. We present analysis and results that demonstrate a vast set of potential benefits of AOM for network and middleware design for distributed CPES applications.

The rest of the chapter is organized as follows: We survey recent developments on CPES modeling and simulation tools and methodologies in Section II, followed by the detailed analysis of application requirements for CPES applications in Section III. In Section IV, we introduce the main modeling modeling approach and explain the aspect-oriented modeling methodology for the energy systems domain. Section V presents simulation results with emphasis on temporal characterization of model execution. We finally discuss possible extensions in Section VI and provide concluding remarks in Section VII.

2 Related work

The use of computer models for power system analysis have been an actively investigated research topic for several decades. Early models and software for transmission and distribution systems emerged when computers started to become popular for business purposes. Following the rapid development in distributed and parallel computing, distributed power system modeling and analysis has become a widely studied research topic of the last decade [10, 30, 21, 20].

Existing electric power simulators provide well-proven point tools for transmission networks (e.g., Siemens PSS/E) and distribution networks (e.g., GridLab-D [8], general optimal power flow [32]). Recent work has also focused on enabling power system and network co-simulation. GridSpice [3], which composes several projects (such as GridLab-D and MatPower) in a single simulation package, provides a simulation environment that allows modeling the interactions between all parts of the electrical network including generation, transmission, distribution, storage and load models [3]. GridSpice builds upon a cloud-based architecture, in which the client user interface is accessed through the Google App Engine and the simulation package is deployed on a public cloud such as Amazon EC2. This cloud based architecture is able to handle the increasing computational demand of power system simulation even for large-scale smart grid scenarios. The network modeling aspect has not yet been part of simulation.

Co-simulation has also been addressed by several research studies recently, which have integrated continuous system simulators (e.g., OpenDSS, PSLF) with a Discrete-Event network simulator (e.g., NS2, OPNET) to simulate cyber-physical smart-grid behavior [15, 26].

Several approaches utilize the Common Information Model (CIM) as a model to combine several traditional power system analyses [4, 23]. These approaches focus on the interoperability of different energy management subsystems. CIM contains standard-based entities and attributes to present the semantics of power system entities and their organization. [23, 18].

The aspect-oriented modeling paradigm in Ptolemy II builds upon concepts that have first been introduced in the Metropolis project [5], in which the use of *Quantity Managers* have introduced a mechanism to annotate and synchronize different model resources, leading to the intrinsic separation of concerns (SoC) within the model. AOM has also been studied in the context of the Unified Modeling Language (UML) [13, 24], however, the concept of an *aspect* in Ptolemy II provides a fundamentally different approach to the AOM paradigm. The Ptolemy implementation of AOM offers an actor-oriented mechanism to associate aspects with executable models that are defined by deterministic concurrent semantics tied to one or more of many supported models of computation (MoC) [29]. The details of AOM in Ptolemy II will be discussed further in Sections 4.1-4.2.

3 Application Requirements and Modeling Challenges

In recent years, power grids have undergone dramatic changes in the fields of grid monitoring and control due to the increasing number of phasor measurement units (PMUs) deployed to produce real-time synchrophasor data that capture the power system dynamics. The swarm of synchrophasors that actively fetches real-time data, which is then utilized for supervisory control in the distributed power grid, has become the root cause of the need to reevaluate the entire data flow design of the grid. PMUs generate precisely time stamped measurements at rates that typically range between 10-60 samples per second. This high data rate enables power applications to operate at significantly higher frequencies compared to traditional Supervisory Control And Data Acquisition (SCADA) based applications.

Wide Area Monitoring and Control (WAMC) systems benefit from this dramatic increase in the rate and redundancy of grid measurements. One such WAMC application is state estimation. State estimation collects field measurements and solves the system-wide nonlinear state equations based on redundant PMU measurements to estimate the system state variables at each iteration. The results are estimates of state variables in the grid, e.g., voltage magnitude, power injection, power flow, and power factors. These estimates are critical inputs for related power system operational tools, such as contingency analysis, optimal power flow analysis, economic dispatch and automatic generation control.

Combining high-rate phasor measurements with low-rate SCADA data for distributed state estimation has been an emerging research interest of power engineers [16, 19, 22]. The volumes of available PMU data has been a major improvement to the quality of WAMC applications, however, at the expense of increased load at data centers, network fabric, and computation nodes.

In addition to the real-time requirements on data processing, phasor data has also imposed constraints on historian components. Given a control center with hundreds of PMUs installed, archived sensor data can accumulate to the order of terabytes, within only a 30-day period [14]. In effect, it becomes extremely inefficient to utilize a single centralized coordinator to collect and archive all the available data from corresponding balancing areas. One approach to alleviating the burden on computing resources has been to distribute the computation across sub-areas.

At run time, the power system dynamics require data flow to be coordinated for the implementation of algorithms that utilize the data. For instance, varying sensor data rates and imperfect clocks at the sensor end requires data from different sensors to be time-synchronized at the *middleware* layer. Hence, the underlying infrastructure (including middleware and network fabric) plays a key role in satisfying both the functional and non-functional requirements of the power application. We use the term *functional model* to describe the designed behavioral model of a system, whereas the remaining model may be interpreted as the implementation-specific configuration details. The functional requirements include coordination of the data flow to distributed state estimators by data synchronization, aggregation and coordination of multiple data source-destination pairs. The non-functional requirements include the following *aspects*:

- *Scalability*. The middleware needs to support a large number of sensor devices and their inter-communications since they become a significant factor in determining the temporal and functional properties of distributed applications that consume data streams.

- *Low latency and time-predictability*. Optimizing the worst-case latency is particularly important to meet deadlines of time-sensitive applications. Previous work has demonstrated that heavy-tailed latency behavior in middleware is directly observable in the end-to-end completion times of distributed algorithms that require synchronization of an extensive number of sensor streams [1, 2]. Even a simple rule of coordination based on enumerating the expected number of data streams can cause up to 45% overhead in the middleware layer.

- *Reconfigurability*. Power engineers are often required to revise algorithms to improve the accuracy of WAMC applications. The data flow through the power grid consequently needs to be revised to work with the reconfigurations. The distributed programming software should allow adding new computation nodes, revising current components and integrating additional sensor streams into the distributed system. It is required for the modeling methodology to take into consideration this reconfigurable behavior of distributed power applications.

Given the application requirements, a monolithic model representation of distributed applications has an immediate disadvantage of quickly leading to a complex model that blends functional models with implementation details of middleware and communication infrastructure. The monolithic model would intrinsically couple the sensor-to-node data path with the communication infrastructure, which is a poor

design choice as it entangles independent design aspects and tremendously increases model complexity. Moreover, such models develop to be neither scalable nor reconfigurable, especially when the distributed model topology is to be altered.

The modeling approach presented in Section 4.3 demonstrates an aspect-oriented CPES design methodology that separates functional application entities and communication infrastructure. Each model can further be customized with application-specific timing characteristics and an appropriate computation model in an actor-oriented environment.

4 Modeling Approach

Power system dynamics and the communication layer mutually affect one another in a closed-loop distributed sensing and control environment, constraining these two systems to be designed as a combined CPS. Nonetheless, a unified approach that encapsulates continuous dynamics, discrete-event communication models, network layer requirements, possible fault tolerance, and QoS requirements in a single model would be too complex to be useful in any form for system designers.

In the light of these requirements, we present an *aspect-oriented modeling* methodology, which enables the overall design to provide a clear separation of cross-cutting concerns from component interactions in the model. We prototype the AOM approach in Ptolemy II.

Ptolemy II is a heterogeneous actor-oriented design environment that supports hierarchical composition of computation models, which enables continuous dynamics to be integrated with the discrete-event AOM framework. Also, there exists extensive work on enabling Functional Mock-up Units (FMUs) for determinate co-simulation and various other applications that are supported by the Ptolemy framework [6, 31]. The rest of this chapter will study a promising approach to modeling complex network and middleware aspects of complex energy systems. Since the focus of this chapter is mainly the aspect-oriented composition of cross-cutting modeling concerns, integration of continuous system dynamics to the framework will not be discussed in depth, however, it will be noted that this capability has already been studied in the context of Ptolemy II environment [12, 27].

4.1 Ptolemy II

Ptolemy II is an actor-oriented modeling and simulation tool for heterogeneous system design [29]. Actors in Ptolemy II are concurrently executed components that communicate via messages called *events* sent and received through actor *ports*.

An *actor* in Ptolemy is annotated with a set of parameters, input and output ports along with an inner state representation which itself can be a graphical sub-model. Actor execution at a particular level of the model hierarchy is governed by a component named a *director*, which implements a desired MoC. Figure 1a demonstrates an example model that represents two sensor clusters that transfer data streams through the communication network. The streams are later processed at high performance computing (HPC) nodes. A middleware component connects these two nodes and facilitates intermediate data exchange. In Figure 1, the Discrete-Event (DE) Director is used in the top-level model, enforcing DE semantics on events produced by the PMUCluster actors.

Ptolemy also provides entities called *decorator*s, which are objects that can be associated with other objects as parameters and provide services to those objects they are associated with. *Aspects* in Ptolemy, which are discussed in the following section, rely on the *decorator* mechanism to provide services to the actors they are associated with.

4.2 Aspect-Oriented Modeling in Ptolemy II

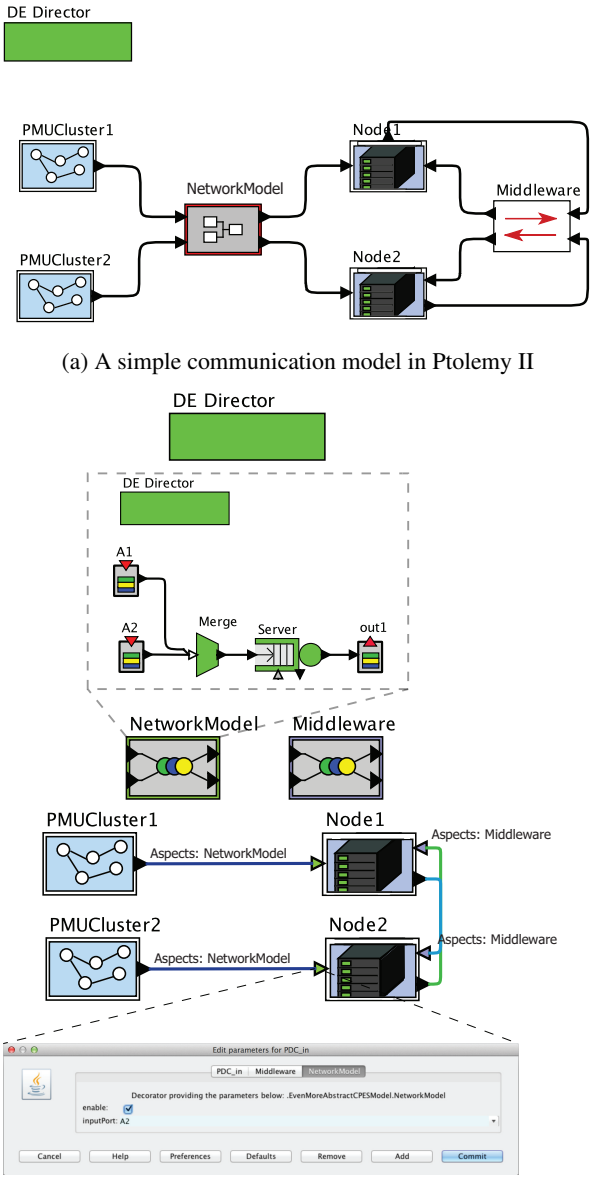
Ptolemy is an actor-oriented framework. Modeling cross-cutting concerns in an actor-oriented environment entails an unambiguous syntax for associating an aspect with an actor object. In the context of CPES models, we concentrate on *communication aspects* that provide cross-cutting refinements to the network resource contention and inter-component behavior modeling. In a Ptolemy model, a *communication aspect* is associated with other model components via parameters tied to actor ports [7].

As an example, in Figure 1b, a subsystem network model that features two network aspects have been presented. The input ports of `Node1` and `Node2` actors are associated with the `NetworkModel` aspect, as annotated on the incoming links to the regarding ports. This association implies that any event destined to these ports would first be forwarded to the `NetworkModel` for processing, before they are routed back to the original destination. In Figure 1b, any event sent from `PMUCluster1` to `Node1` are routed to port `A1` of the `NetworkModel`. Upon being merged with events from port `A2` and being processed by a `Server` actor in time stamp order, the events are routed to their original destinations in the functional model, i.e, the `Node1` actor. The `NetworkModel` aspect aims at modeling the resource contention of the single server that is processing messages from two different data sources. Note that a communication aspect itself is an actor with input and

output ports, and may have an internal composite actor representation, as in this example. It is common for a communication aspect to alter time stamps and values of incoming events before relaying these to their respective destination ports. In this example, the `Server` actor, which is part of the `NetworkModel` definition, applies a constant processing delay to incoming events, while the payloads of the events remain unaffected.

As seen in the dialogue box in Figure 1b, the association of the `Node` actors with the `NetworkModel` aspect is realized by an `enable` parameter of the input port. The process for associating the `Middleware` aspect with `Node` actors follows the same pattern.

Following the introduction of the AOM semantics, it would provide insight to compare the models given in Figures 1a and 1b. Note that these two models consist of the same set of actors, which are connected in a different topology in the two realizations. The `NetworkModel` and `Middleware` actors appear as regular actors that belong to the functional model in the first case, whereas they have been implemented as communication aspects in the second variant. Note that the internal functional representation of these two actor blocks remain identical across 1a-1b. The former approach, illustrated in in Figure 1a serially incorporates network and middleware behavior into the functional model. Specifically, Figure 1a is ambiguous as in whether `PMUCluster1` and `PMUCluster2` are both communicating with `Node1` and `Node2`, or the `NetworkModel` is providing a peer-to-peer (P2P) channel between pairs of actors. This is not a feasible design choice due to two reasons: (i) network and middleware components are typically models of helper infrastructure that are highly dependent on design choices and are usually subject to frequent alterations due to dynamic hardware requirements and availability [9], therefore it is not desirable to couple these with the core functionality of a system (ii) having a single-view model with serially connected components introduces an intrinsic impedance to the scalability of the model.



(a) A simple communication model in Ptolemy II

(b) Model refinement using Ptolemy aspects

Figure 1: Comparison of Monolithic and Aspect-Oriented Modeling Approaches

4.3 Domain-Specific Models

We represent key entities in distributed power applications including sensors, data concentrators, computing clusters and middleware in the framework of aspect-oriented modeling. In the context of the smart electric grid, it is natural to represent network topologies and middleware components as communication aspects. The top level model for a three-area distributed application topology is depicted in Figure 2. The section describes each entity of the model. The execution details of this model will be discussed further in section 5.

4.3.1 Sensor Clusters

Sensors are the main sources of data that enable applications that run on distributed power systems. The sensors considered for smart-grid applications are PMUs, also known as *synchrophasors*, due to providing synchronous phasor data at rates varying in the range of 10-60 samples per second. Due to the overwhelming number of sensors in the power grid, it is not feasible to model each sensor as an individual component. For the interest of distributed applications, one practical

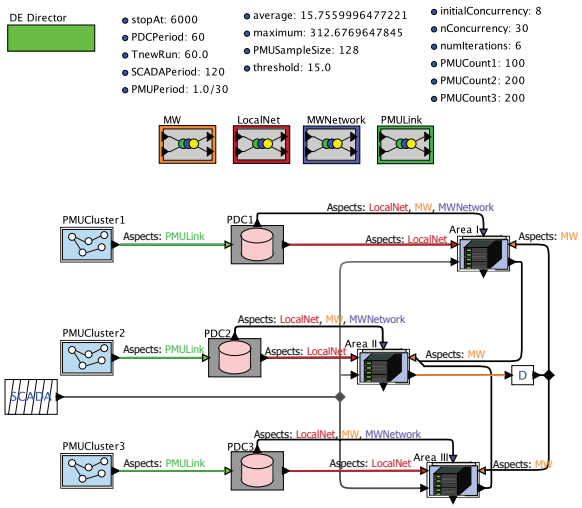


Figure 2: Aspect-Oriented communication model for a distributed grid application

abstraction is to model clusters of sensors that belong to the same area in a single component that is parameterized by the number of PMUs that it encapsulates. The `PMUCluster` actor given in Figure 2 follows this methodology and generates samples of multiple data streams at each iteration, where the number of streams corresponds to the number of PMUs in this area.

4.3.2 Data Concentrators

Data concentrators are intermediate historian components that are commonly utilized in distributed power systems architectures. In the considered model (see Figure 2), Phasor Data Concentrators (PDCs) are actors representing the data concentration units that receive data streams from PMU Clusters in a FIFO fashion and perform relaying prior to sending data to computation nodes and the middleware.

4.3.3 Computation Clusters

The computation clusters are abstract components that may correspond to hardware that process sensor data for various purposes (e.g., GPUs, HPC Clusters). In the common sense of a distributed application, the computation units cannot perform autonomously and are capable of producing local estimates of algorithm outcomes. The model for these computation clusters are named `Area I-III` in Figure 2. Each `Area` establishes communication with the neighboring computation clusters and exchanges local estimates, until global consensus is established.

4.3.4 Modeling Communication Aspects

The use of aspects for modeling inter-component communication enables different network topologies to be implemented in separate composite models in a cross-cutting way, such that multiple links between components can be mapped to a single communication aspect. In the top-level model given in Figure 2, four aspects are used to model different network fabrics:

- `LocalNet`: Network aspect that models inter-area links. Depicted in Figure 3, this aspect models each link as a server with probabilistic delay characteristics. Three types of communication links are defined in the scope:
  - PDC to Area: The local area links used for sending PMU readings to `Areas` to be used in the distributed computations
  - PDC to Middleware : The links that are used for sending PMU readings to the middleware layer for aggregation and global state estimation
  - Area to Area : P2P connection fabric between `Areas`. The connections implicitly determine the topological layout of the neighboring areas in the power grid. In the studied topology, pairwise communications are established between Areas I-II and II-III.
- `MWNetwork`: This intermediate component models the network that connects the historians (PDCs) of each area to the middleware (`MW`). In Figure 4, this network layer models a single channel that carries the PMU data coming from all three areas into the middleware.
- `PMULink`: In Figure 5, this aspect is an aggregate of parallel dedicated links that connect each Phasor Measurement Unit (PMU) to the local PDC.

In all of the listed communication models, probabilistic component delays are modeled as a function of physical system characteristics including physical link length, propagation speed of light in fiber, network packet length, link capacity and a constant queuing delay. These variables characterize the smart-grid communication latency based on the NASPINet specification [17].

4.3.5 Serial Composition of Aspects

In a general setting, links between components may be associated with more than one aspect at a time. In such condition, the processing order of events as they are pipelined through aspects is determined by the order in

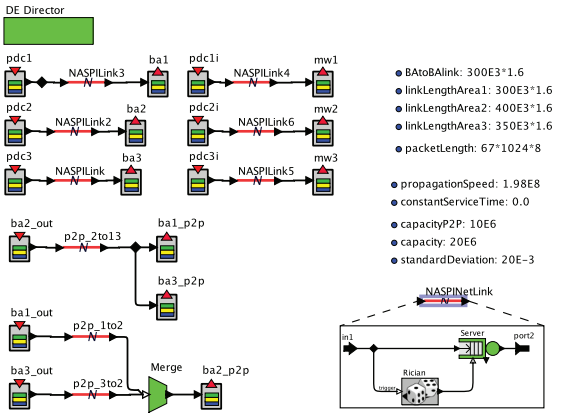


Figure 3: Sub-Model for LocalNet aspect

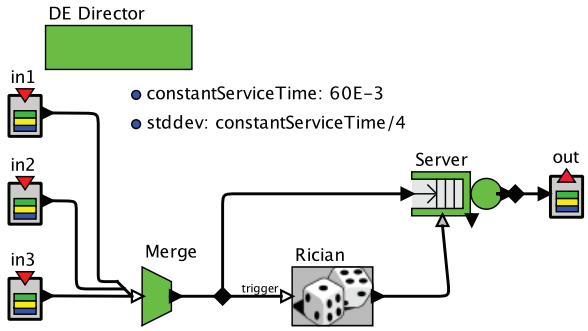


Figure 4: Sub-Model for MWNetwork aspect

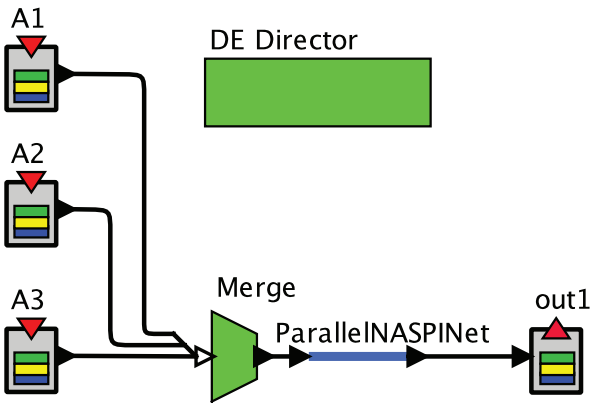


Figure 5: Sub-Model for PMULink aspect

which aspects are associated with the input port. In Figure 2, the pairwise data communication between actors {PDC1, PDC2, PDC3} and {Area I, Area II, Area III} respectively, are mediated through three communication models, specified as three composite communication aspects named LocalNet, MW, and MWNetwork, whose respective models are given in Figures 3, 7 and 4. Events generated by the PDCs are processed by the LocalNet, then are handed over to MW and finally to MWNetwork before eventually being delivered to the north input port of the computation nodes (Area I). Figure 6 demonstrates the communication traversal from PDC1 to Area I. The time stamp of the original event produced by PDC1 is being modified by the three communication actors before the event is handed over to the final destination, Area I. Note that in the case that communication aspect also addresses fault conditions, as in a packet erasure channel, the event may be *dropped* by one of the aspects and never be delivered to the destination port.

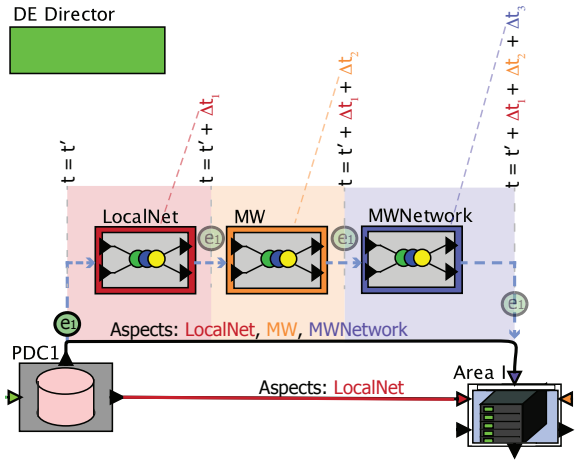


Figure 6: Sequential event handover between multiple aspects associated with the same link

4.3.6 Modeling Middleware Structure

The middleware configuration is studied separately from the network layer. The reasoning is the additional role of performing aggregation and time-alignment of packets of the middleware layer. These roles can be elaborated as follows:

- *Aggregation.* The initial task of the middleware is to time-align and aggregate the individual sensor data into a single file that only contains information of faulty or missing data from all areas of sensors. As modeled in Figure 7, lower branch, it combines extracted faulty information into a globally broadcast file. local runs eliminate data with these time stamps in order to maintain data consistency among clusters.

- *Control of global convergence.* The second task of the middleware is to receive local estimates from distributed areas and to declare global convergence

of state. As a result of this requirement, in general, P2P communication between computation nodes of different areas must be coordinated by the middleware. The model is presented in Figure 7.

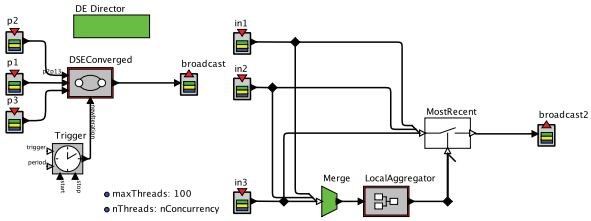


Figure 7: Sub-Model for MW aspect: [Left branch] Algorithm convergence control [Right branch] Aggregation and processing simulation at middleware level

5 Case Study: Distributed State Estimation

We consider the Distributed State Estimation (DSE) application to evaluate the use cases of aspect-oriented CPES models. Improving the accuracy and robustness of DSE has been an actively investigated research topic [19]. The algorithms are commonly exercised on centralized computation nodes for verification, nonetheless, the actual deployment performance and correctness under a distributed setting remains unexplored [10]. There is a shortfall of distributed system testbeds on which DSE can be deployed and evaluated, so that the characteristics of the run time behavior in the presence of communication, processing and middleware delays can be accounted for. In practice, the volume of data communication between state estimators until convergence of estimators is a function of underlying power system dynamics and data quality. Understanding the characteristics of communication latency in a model-based design environment leverages better design of DSE algorithms that scale in size of the system and data volume [9].

DSE is a comprehensive candidate application that demonstrates the middleware requirements for time-centric CPES applications due to its data intensive nature that forces multi-area communication and coordination. Moreover, several different network fabrics are involved in DSE architectures, for which aspect-oriented modeling proves efficient. From a distributed system perspective, the challenge is to autonomously coordinate the data flow and access within the distributed model. The core of this system architecture is a middleware that mediates data exchange between predetermined grid partitions. Local results from each area are transferred to the middleware to be aggregated and time-aligned. The middleware also coordinates the faulty PMU readings of the entire system, and broadcasts this information to all remote state estimators for global situational awareness. An additional decision component, which is also part of the middleware, receives intermediate state estimation results and notifies each area when global convergence for the estimation model has been achieved. Additional P2P communication between subsystems occur without the need of a coordinator.

We consider the aspect-oriented network model in the context of DSE using aspects and actor components defined in Section 4.3. The simulation enables *what-if analysis* for different scenarios of network delays and middleware configurations. The simulation results provide insight on temporal properties of large a class of distributed power applications at the design level.

5.1 Overview of the Top Level Model

The DSE application is triggered by two main sources of data, which are collected from (i) PMU and (ii) SCADA sensors. Since SCADA data is available at much lower frequencies compared to PMU data, we consider PMU data to be the main trigger for the application data flow.

As shown in Figure 2, the functional model consists of PMU clusters that generate sensor readings at a rate of 30 samples per second, delivered to PDCs for relaying, which are then sent to `Areas` for local executions of the DSE algorithm. In parallel to the computation task, the data is additionally sent to the middleware (`MW`) layer, where it is aggregated to identify faulty or missing data into an aggregate index file, which is then broadcast to Areas for global situational awareness.

For a single iteration of the DSE application, it is expected that the areas exchange multiple estimates of local state with the neighboring computation nodes until global convergence of state has been attained. A convergence message issued by the middleware declares the finalization of the active iteration of the DSE algorithm.

5.2 End-to-end Simulation

The use cases of the aforementioned functional model annotated with network and middleware aspects can be multiple. Initially, such functional models are essential for evaluating architectures under test for achievability of end-to-end latency and performance goals. Decoupling the functional model from the implementation-dependent network and middleware components triggers a convenient experimentation process for implementation decisions to be made over wide-area smart-

grid communication design. For instance, means of communications and protocols which are feasible to be deployed on the power grid communications, remains an active topic of debate. In this formalism, it is possible to replace each network component with a candidate communication model (e.g., GSM, PLC, IEEE 802.11) and evaluate the end-to-end performance in the presence of the candidate model.

5.3 What-If Analysis

To explore satisfiability of requirements discussed in Section 3 by the proposed model, we carry out a *what-if* analysis under corner cases of distributed system behavior. Since the middleware design has flexible choices of architectural options, such concerns to be studied include whether a certain middleware architecture can accommodate application deadlines or whether middleware scales as smart grid components increase in size and connectivity over time. The following subsections address the temporal requirement schemes for (i) network congestion, (ii) network faults, (iii) meeting strict application deadlines, and (iv) maintaining a desired mean end-to-end run time for applications.

5.3.1 Network Congestion Analysis

It is often desirable to test an end-to-end power application under worst-case scenarios in terms of latency. The complex network that interconnects distributed components in the topology is the main source of timing uncertainty in such applications. A reduced channel capacity or a burst of sensor data packets may dominate link capacity to cause local or global deadlines to be missed. We carry out what-if analysis on the model by defining custom stress tests on network components. As an example, we consider the `LocalNet` component, which is modeled to have a link capacity of 10 Mbps per area. As a stress test, we assume a 60% capacity loss on the link that connects Area I to the middleware, on the topology presented in Figure 2. Figure 8 illustrates the distributions of end-to-end run times under stress and under normal configurations. The simulation results demonstrate the effect of the local link capacity loss on the distribution of end-to-end execution times. Under the network congestion, the majority of execution times exceed the worse case run time observed in the no congestion scenario.

5.3.2 Network Fault Modeling

It is a common scenario to consider network failures and probabilistic faults during packet transmission in

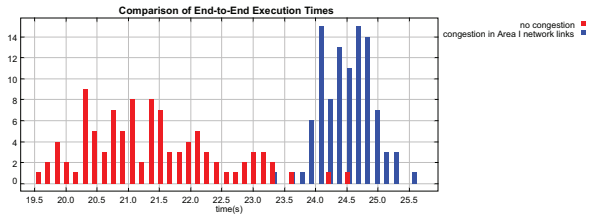


Figure 8: Effect of local link capacity loss on end-to-end DSE run times

designing a network process. When accounting for end-to-end performance, delay and failure characteristics may have a large impact on the satisfiability of application requirements given a platform of choice. Since faults of this nature are usually considered to be *external* artifacts to system behavior than being a part of the intrinsic system, aspects facilitate modeling fault behavior as cross-cutting injections to the model.

We consider a packet erasure channel scenario, where the probability of each packet being dropped at a channel is given by a Bernoulli random variable, parameterized with a packet drop probability p. In Ptolemy, a *modal model* actor defines an extended finite state automaton, in which each state may implement an arbitrary MoC internally. Modal models also support probabilistic transitions, where a probabilistic guard expression of a transition is defined by an expression `probability(`p`)`, which evaluates to *true* with probability $p \in [0,1]$. For visualization purposes, we consider a packet erasure channel with 10% loss. With this probabilistic fault model implemented as an aspect, a PMU-to-PDC link with packet loss can be realized. Figure 9 illustrates a portion of the top level model that now includes a new aspect, `PacketLossFault`, that is associated with the input ports of the PDC components. In Figure 10, a subsequence of the input and output packets are plotted in the `PacketLossFault` aspect. Note that the packet drop behavior will process the events that have already been handled by the `PMULink` aspect. This is why variable inter-arrival times are observed in the input packet stream. It can be seen that some packets that are present in the input are not relayed to the output. The empirical probability of packet loss follows the loss probability that is a parameter of the `PacketLossFault` aspect.

5.3.3 Middleware Scalability for Distributed Applications with Fixed Deadlines

We consider a time-critical application scenario, in which a distributed application has a hard deadline to

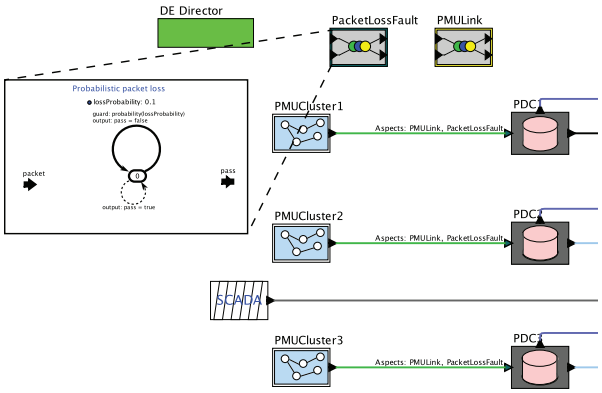


Figure 9: Sub-model with packet loss fault injection at the PMU-to-PDC link

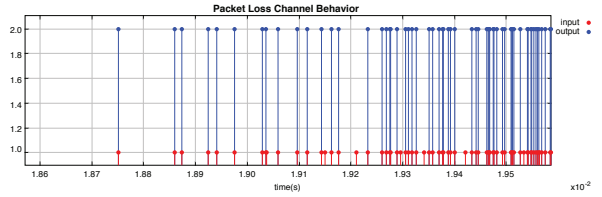
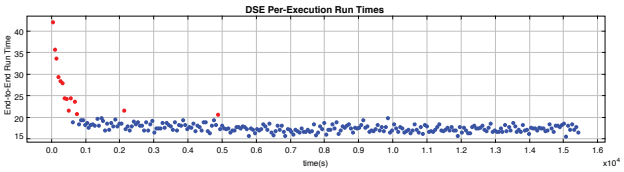


Figure 10: Packet erasure channel behavior with 10% loss probability

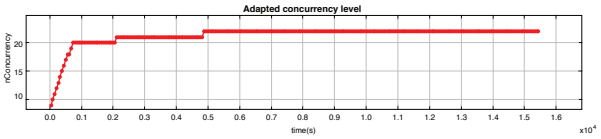
satisfy. One common class of applications with such deadline characteristics arise from integration of the middleware layer with physical grid dynamics. While the deterministic CPS integration between physical plants and the cyber layer is a major research interest, we point the reader to [28] for further details and focus on the networking aspect in this section.

The end-to-end model starts execution with a baseline middleware resource allocation scenario and requests an increase to the allocated resources in the middleware layer each time a deadline is missed, subject to a maximum middleware thread pool capacity. 500 PMU streams are considered to be available in the electrical power grid, distributed to three areas as $nPMU = \{100, 200, 200\}$, where nPMU_i denotes the number of PMUs at Area i. The assumption of the application scale already exceeds the scenario in previous research that provide commercial distributed software solutions [18]. Figure 11 depicts an an application scenario that assumes a deadline of $\tau = 20$ s. The simulation model assumes an initial concurrency level of 8 (simulated by 8 parallel middleware processing queues), that is dynamically incremented upon a missed deadline τ during execution. The simulation trace reveals that this dynamic scaling policy results in a steady-state middleware configuration, under which missed deadlines are avoided for the most part.

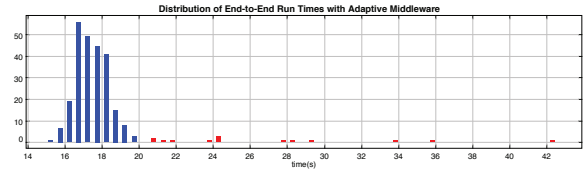
Likewise, deadlines can be applied to sub-models too, for instance, the designer could set a latency deadline on the `PMULink` perform adaptive design variations on this link.



(a) Run-Times of middleware adaptation. [Blue] Executions within deadline, [Red] Executions that missed the deadline



(b) Middleware concurrency level adaptation



(c) End-to-end run time histogram of overall execution with adaptive middleware capacity

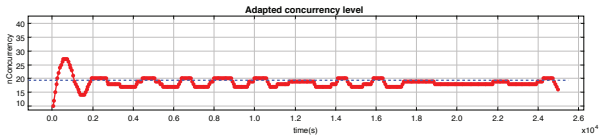
Figure 11: Middleware adaptation for fixed-deadline distributed applications

5.3.4 Middleware Scalability for Distributed Applications with Variable Deadline

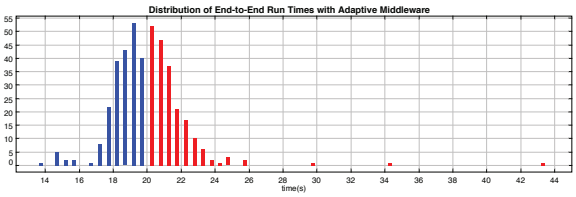
A variant of the above scenario is desired to evaluate middleware scalability for applications that don't have a fixed deadline, but require maintaining a desired average end-to-end run time. The resources in the model (network latency, middleware concurrency level) are modeled in a stochastic sense, which in turn may suggest dynamic analysis of application deadlines. To maintain a desired average run time subject to probabilistic model delay, a manually-tuned proportional-integral (PI) controller is implemented. Th PI controller is used to control the middleware concurrency level, which is modeled as a thread-pool with variable concurrency. The error signal provided to the PI controller is the difference between the the desired and current end-to-end run time. The PI output is quantized to the nearest integer to be used as the correction signal to the middleware concurrency level. Figure 12 demonstrates simulation results for a sample three-area application with a desired end-to-end run time of 20 s. As shown in Figures 12a-

12c, with the PI controller in the loop, execution time is eventually stabilized around the desired run time of 20 s, with a steady-state concurrency level of 19.

(a) Per-execution DSE run time [Blue] Below desired, [Red] Above desired



(b) Middleware concurrency level adaptation



(c) End-to-end run time histogram of overall execution with desired run time

Figure 12: Middleware adaptation for desired average run time of 20s

6 Discussions

6.1 Hardware-in-the-loop Simulation

A common method for evaluation of complex real-time embedded systems is hardware-in-the-loop (HIL) simulation. Aspect-oriented simulation models for network and middleware topologies can be used to evaluate real-time control algorithms using smart grid components such as relays and PMUs. Aspects can act as an interface in the simulation-hardware boundary, while integrating physical system components under evaluation into the control simulation. An example to such HIL study would be to replace the PMU clusters in Figure 2 with actual PMU hardware.

6.2 Middleware-in-the-Loop Simulation

The single-server network models assumed so far may fall short in demonstrating the actual complex network and middleware fabric required for grid deployment. AOM can also facilitate the software integration process for evaluating actual middleware architectures into the model, enabling middleware-in-the loop simulation. Figure 13 demonstrates an alternative implementation of the `Middleware` aspect with Java connectors (`MIFTransmitter` and `MIFReceiver`) to a proprietary middleware implementation. An application that uses this middleware has been introduced in [1], where an actual middleware is invoked within the simulation loop. The middleware is implemented using a JMS (Java Message Service) interface that connects to ActiveMQ [http://activemq.apache.org/], an open-source Apache messaging server. Hence, the run times of a three-area system performing DSE are collected as the per-packet real-time latency introduced by ActiveMQ.

In Figure 13, the Discrete-Events delivered to the `MIFTransmitter` are internally converted to JMS messages, processed through the ActiveMQ, received at the `MIFReceiver` and eventually issued as discrete-events sent to the rest of the simulation flow. To incorporate the timing characteristics of the ActiveMQ fabric, the real-time processing latency is computed by the Java connector actors and reflected to model time by altering time stamps of the outgoing events. The ActiveMQ implementation does not follow an adaptively scalable structure as assumed in Figures 11 and 12. Moreover, parallel queues in this framework are configured statically upon initialization of the ActiveMQ server. Based on simulation results, the extension to ActiveMQ should include a monitoring mechanism of collecting run times and a trigger to create new queues for missed deadlines. The conclusions suggest that this modeling approach, which enables integrating actual middleware with smart grid network models would guide the actual middleware development process.

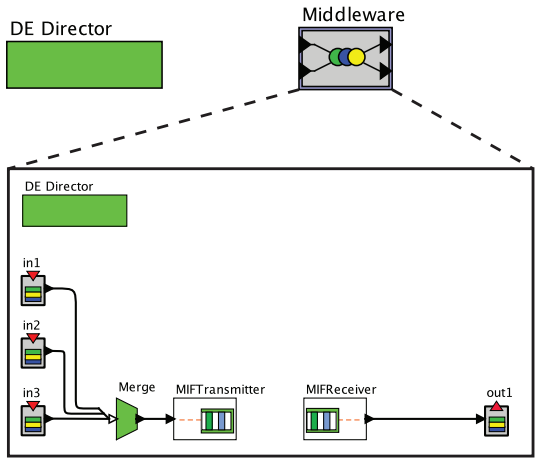


Figure 13: Middleware-in-the-Loop Simulation

7 Acknowledgments

The authors would like to thank Shuai Lu, Senior Engineer at the Pacific Northwest National Laboratory (PNNL) for his insightful guidance on the Power Engineering background aspects of the chapter.

8 Conclusion

In this chapter, we have introduced a novel design methodology for data intensive distributed CPES applications. We studied an aspect-oriented design paradigm, which focused on decoupling cross-cutting aspects in functional CPES models. A general distributed CPES architecture has been presented and was populated with network and middleware models that can be altered without any modifications to the functional model.

Following the discussed modeling paradigm, we performed an extensive simulation study using the example of the DSE application, which demonstrated how end-to-end simulation can be performed to yield analysis of algorithm execution times and to evaluate resource requirements for desired application timing profiles. We also demonstrated how AOM enables performance comparisons among different network topologies that are integrated with a fixed functional grid application. The discussions were presented on an abstract high-level application scenario to highlight that they seamlessly apply to a wide family of WAMC applications.

Further improvements of the modeling paradigm will include integration of fault and attack models and anomaly detection techniques to simulate data and network quality in the AOM setting. This will contribute to developing robustness and reliability features as part of distributed CPES models. It will also be interesting integrate the AOM paradigm followed for network modeling with a co-simulation of physical grid dynamics and demonstrate the benefit of the modeling abstractions for this integration.

References

[1] Akkaya, I., Liu, Y., Gorton, I.: Modeling and analysis of middleware design for streaming power grid applications. In: Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference, MIDDLEWARE '12, pp. 1:1–1:6. ACM, New York, NY, USA (2012). DOI 10.1145/2405146.2405147. URL http://doi.acm.org/10.1145/2405146.2405147

[2] Akkaya, I., Liu, Y., Lee, E.A., Gorton, I.: Modeling uncertainty for middleware-based streaming power grid applications. In: Proceedings of the 8th Workshop on Middleware for Next Generation Internet Computing, MW4NextGen '13, pp. 4:1–4:6. ACM, New York, NY, USA (2013). DOI 10.1145/2541608.2541612. URL http://doi.acm.org/10.1145/2541608.2541612

[3] Anderson, K., Du, J., Narayan, A., Gamal, A.E.: Gridspice: A distributed simulation platform for the smart grid. In: Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2013 Workshop on, pp. 1–5 (2013). DOI 10.1109/MSCPES.2013.6623311

[4] Andrén, F., Stifter, M., Strasser, T.: Towards a semantic driven framework for smart grid applications: Model-driven development using CIM, IEC 61850 and IEC 61499. Informatik-Spektrum **36**(1), 58–68 (2013). DOI 10.1007/s00287-012-0663-y. URL http://dx.doi.org/10.1007/s00287-012-0663-y

[5] Balarin, F., Watanabe, Y., Hsieh, H., Lavagno, L., Passerone, C., Sangiovanni-Vincentelli, A.: Metropolis: An integrated electronic system design environment. Computer **36**(4), 45–52 (2003)

[6] Broman, D., Brooks, C., Greenberg, L., Lee, E.A., Masin, M., Tripakis, S., Wetter, M.: Determinate composition of fmus for co-simulation. In: Proceedings of the Eleventh ACM International Conference on Embedded Software, p. 2. IEEE Press (2013)

[7] Cardoso, J., Derler, P., Eidson, J.C., Lee, E.A., Matic, S., Zhao, Y., Zou, J.: Modeling timed systems. In: C. Ptolemaeus (ed.) System Design, Modeling, and Simulation using Ptolemy II, pp. 355–393. Ptolemy.org, Berkeley, CA (2014). URL http://ptolemy.org/books/Systems

[8] Chassin, D., Schneider, K., Gerkensmeyer, C.: GridLAB-D: An open-source power systems modeling and simulation environment. In: Transmission and Distribution Conference and Exposition, 2008. IEEE/PES, pp. 1–5 (2008). DOI 10.1109/TDC.2008.4517260

[9] Chen, Y., Huang, Z., Liu, Y., Rice, M., Jin, S.: Computational challenges for power system operation. In: System Science (HICSS), 2012 45th Hawaii International Conference on, pp. 2141–2150 (2012). DOI 10.1109/HICSS.2012.171

[10] Chen, Y., Huang, Z., Liu, Y., Rice, M.J., Jin, S.: Computational challenges for power system operation. In: Proceedings of the 2012 45th Hawaii

International Conference on System Sciences, pp. 2141–2150. IEEE Computer Society (2012)

[11] Derler, P., Lee, E.A., Vincentelli, A.S.: Modeling cyber-physical systems. Proceedings of the IEEE **100**(1), 13 –28 (2012). DOI 10.1109/JPROC.2011.2160929

[12] Eker, J., Janneck, J.W., Lee, E.A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., Xiong, Y.: Taming heterogeneity-the Ptolemy approach. Proceedings of the IEEE **91**(1), 127–144 (2003)

[13] Elrad, T., Aldawud, O., Bader, A.: Aspect-oriented modeling: Bridging the gap between implementation and design. In: Generative Programming and Component Engineering, pp. 189–201. Springer (2002)

[14] Gibson, T., Kulkarni, A., Kleese-van Dam, K., Critchlow, T.: The feasibility of moving pmu data in the future power grid. In: CIGRE Canada Conference on Power Systems: Promoting Better Interconnected Power Systems. Hallifax,NS, Canada (2011)

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

[16] Gomez-Exposito, A., Abur, A., de la Villa Jaen, A., Gómez-Quiles, C.: A multilevel state estimation paradigm for smart grids. Proceedings of the IEEE **99**(6), 952–976 (2011)

[17] Hasan, R., Bobba, R., Khurana, H.: Analyzing naspinet data flows. In: Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES, pp. 1–6. IEEE (2009)

[18] Hazra, J., Das, K., Seetharam, D.P., Singhee, A.: Stream computing based synchrophasor application for power grids. In: Proceedings of the first international workshop on High performance computing, networking and analytics for the power grid, HiPCNA-PG '11, pp. 43–50. ACM, New York, NY, USA (2011). DOI 10.1145/2096123.2096134. URL http://doi.acm.org/10.1145/2096123.2096134

[19] Jiang, W., Vittal, V., Heydt, G.: A Distributed State Estimator Utilizing Synchronized Phasor Measurements. Power Systems, IEEE Transactions on **22**(2), 563 –571 (2007). DOI 10.1109/TPWRS.2007.894859

[20] Khaitan, S.K., McCalley, J.D.: Cyber physical system approach for design of power grids: A survey. In: Power and Energy Society General Meeting (PES), 2013 IEEE, pp. 1–5 (2013)

[21] Khaitan, S.K., McCalley, J.D.: Design techniques and applications of cyberphysical systems: A survey. Systems Journal, IEEE **PP**(99), 1–16 (2014). DOI 10.1109/JSYST.2014.2322503

[22] Korres, G.: A distributed multiarea state estimation. Power Systems, IEEE Transactions on **26**(1), 73–84 (2011)

[23] Koziolek, A., Happe, L., Avritzer, A., Suresh, S.: A common analysis framework for smart distribution networks applied to survivability analysis of distribution automation. In: Software Engineering for the Smart Grid (SE4SG), 2012 International Workshop on, pp. 23–29 (2012). DOI 10.1109/SE4SG.2012.6225713

[24] Krechetov, I., Tekinerdogan, B., Garcia, A., Chavez, C., Kulesza, U.: Towards an integrated aspect-oriented modeling approach for software architecture design. In: 8th Workshop on Aspect-Oriented Modelling (AOM. 06), AOSD, vol. 6. Citeseer (2006)

[25] Lee, E.A.: Cyber physical systems: Design challenges. In: International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC),, pp. 363 – 369. IEEE, Orlando, Florida (2008). URL http://dx.doi.org/10.1109/ISORC.2008.25

[26] Lin, H., Sambamoorthy, S., Shukla, S., Thorp, J., Mili, L.: Power system and communication network co-simulation for smart grid applications. In: Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES, pp. 1–6 (2011). DOI 10.1109/ISGT.2011.5759166

[27] Liu, J., Liu, X., Lee, E.A.: Modeling distributed hybrid systems in Ptolemy II. In: American Control Conference, 2001. Proceedings of the 2001, vol. 6, pp. 4984–4985. IEEE (2001)

[28] Matic, S., Akkaya, I., Zimmer, M., Eidson, J.C., Lee, E.A.: PTIDES model on a distributed testbed emulating smart grid real-time applications. In: Innovative Smart Grid Technologies (ISGT-EUROPE). IEEE, Manchester, UK (2011). URL http://chess.eecs.berkeley.edu/pubs/857.html

[29] Ptolemaeus, C. (ed.): System Design, Modeling and Simulation using Ptolemy II (2014). URL http://ptolemy.org/books/Systems

[30] Tomsovic, K., Bakken, D.E., Venkatasubramanian, V., Bose, A.: Designing the next generation of real-time control, communication, and computations for large power systems. Proceedings of the IEEE **93**(5), 965–979 (2005)

[31] Wetter, M.: Co-simulation of building energy and control systems with the building controls virtual test bed. Journal of Building Performance Simulation **4**(3), 185–203 (2011)

[32] Zimmerman, R., Murillo-Sanchez, C., Thomas, R.: Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. Power Systems, IEEE Transactions on **26**(1), 12–19 (2011). DOI 10.1109/TPWRS.2010.2051168