

# Modeling and Analysis of Middleware Design for Streaming Power Grid Applications

Ilge Akkaya  
University of California  
Berkeley  
ilgea@eecs.berkeley.edu

Yan Liu  
Pacific Northwest National  
Laboratory  
yan.liu@pnnl.gov

Ian Gorton  
Pacific Northwest National  
Laboratory  
ian.gorton@pnnl.gov

## ABSTRACT

High quality, high throughput sensor devices in the power distribution network are driving an increase in the volume and the rate of data streams available to monitor and control the power grid. Middleware support is essential to coordinate data streams with distributed power models and adapt to situations with data communication failures and errors in the sensor measurements. One challenge in designing this middleware support is scalability. In particular, the number of sensor devices and their intercommunications is a significant factor in determining temporal and functional properties of power models such as distributed state estimation. In this paper, we present our experience modeling the entire data flow from sensor devices to distributed state estimators using middleware. This model helps to analyze the middleware's behavior and its scalability in coordinating data streams.

## Categories and Subject Descriptors

I.6.3 [Simulation and Modeling]: Applications; H.3.4 [Information Storage and Retrieval]: Systems and Software—*Distributed systems*

## General Terms

Performance, Design

## Keywords

Middleware, distributed system, smart grids, modeling, performance

## 1. INTRODUCTION

Electrical power grids are increasingly incorporating high quality, high throughput sensor devices in the power distribution network. These devices are driving an unprecedented increase in the volume and the rate of information available to utilities. For example, new Phasor Measurement Unit

(PMU) sensors produce up to 60 samples per second, in contrast to existing conventional Supervisory Control And Data Acquisition (SCADA) measurements that are generated every five seconds. This dramatic growth of these high quality sensors demands new distributed power grid models for real-time predictions, which can eliminate the bottleneck of a centralized location for large-scale data management and monolithic computation. However, a distributed power grid model needs to exchange intermediate states with its neighboring regions for global situational awareness.

Compared to conventional centralized models, the delays and order of data exchanged and quality of measurement data in streams can all cause inaccurate modeling results. Therefore middleware support is essential to synchronize multiple data streams to/from distributed power models and adapt to situations of causality of data communication and errors in the data.

One challenge in designing power grid middleware is scalability. In particular, the middleware needs to support a large number of sensor devices and their intercommunications since they become a significant factor in determining the temporal and functional properties of distributed power models. For example, state estimation at a local balancing authority typically receives sensor readings from transmission substations via Supervisory Control And Data Acquisition (SCADA) system every 4 seconds. It then extrapolates a full set of grid conditions based on the grid's current configuration and a theoretically based power flow model.

More recently, synchrophasor measurement devices (such as PMUs) are being deployed aggressively in the US and globally [4]. Each PMU provides up to 60 measurement samples of voltage, current and frequency each second. These high frequency measurements aligned with SCADA measurements enable the creation of real-time, wide area monitoring systems, requiring a scalable infrastructure to disseminate and coordinate multiple sources of sensor data. Hence the design of such a middleware infrastructure needs to consider the entire data flow that passes through several networked hardware and software components including sensor devices, software data concentrators, software data readers, distributed state estimators and subsequent data exchanges amongst the state estimators. We therefore need a solution to answer questions such as whether distributed state estimation can complete within one minute if the state estimation follows every minute cycle, given a number of PMUs in an area.

Model-based design has many benefits in system development [12], since models allow tracking the evolution of a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*Middleware 2012 Industry Track*, December 3-7, 2012, Montreal, Quebec, Canada.

Copyright 2012 ACM 978-1-4503-1613-2/12/12 ...\$15.00.

system design, and enable simulation and analysis. Having a simulation model of the entire network of a distributed power operations infrastructure helps to answer questions regarding pre-deployment projections of the middleware performance and scalability.

In this work, we use the Ptolemy II [6] framework for modeling and simulation purposes. Ptolemy has a long history in modeling Cyber-Physical Systems (CPS) [5][9][13]. A CPS model consists of physical processes that interact with models of network and computation platforms, usually including feedback relations. Some general elements that need to be modeled in particular are sensors, actuators, network fabrics and middleware components. Communication delays, software scheduling and timing properties are some common considerations of the design process.

Ptolemy II is an actor-oriented design tool. Actors are components that communicate via ports. It is possible to build hierarchical executable models with this framework, where at each level of hierarchy the execution of the model is governed by a component called a *director*, which implements a model of computation. Passage of time at each compositional level is governed by a local clock, which allows the user to define different clock rates, clock drifts and each local clock's relation to real time, if desired.

In this paper, we initially discuss power grid specific system modeling requirements in Section 3. In Section 4, we explain the Ptolemy II modeling approach and describe model details of system components as well as the middleware structure. We continue with an application study in Section 5, where we study end-to-end modeling and simulation for distributed state estimation for a power distribution network. We present results and analyze the future applications of the design approach in Section 6.

## 2. RELATED WORK

The reliability of the power grid is basically defined by the ability to deliver electric power from generation to load without disruption [2]. To manage this power delivery, fast state estimation is increasingly important in providing rapid response times to support wide-area control of power grid systems. This requires broad observability at individual control centers, typically known as balancing areas, for measurements in its neighborhood of control, and near-real time responsiveness to these observations in coordination with its neighbors.

Recent research on decentralized power grid applications has focused on solutions for simplifying the state estimation process. Bose et al. evaluate the effects of the network topology on the accuracy of the hierarchical state estimation [3]. A set of experiments are devised that cover different scenarios of the type of data communicated, failure at the network connection, and partition of the network topology. The hierarchical state estimation model is presented with sampling data preloaded to the model, which simplifies the issues arising from the distributed architecture.

Existing related work on network infrastructure focuses on enabling power grid status dissemination. With in-situ processing, the raw data from the sensors are processed and summarized on the device and then sent to the utility. A conceptual level architecture design provides useful references [2], however it lacks concrete measures to help understand impacts such as end-to-end latencies that the architecture may incur. Volttron [1] is an agent-oriented execution

platform developed by PNNL to embed sensors with general purpose and programmable computing units. Volttron enables complicated procedures such as an agent that can collect data from one device for a certain amount of time and, then, move to the next without user intervention. Integrating Volttron and middleware is promising to enable customizable data collection and dissemination.

Our previous work produced a distributed state estimation implementation that could work on HPC computers. We have been focusing on increasing the computational speed of solving a large and sparse system of linear equations, which is the time-consuming part of the state estimation process. The developed algorithm is able to solve a large connected system in a solution time comparable with today's SCADA cycle [10].

Looking ahead at the future needs for distributed systems architectures to accommodate new computations in the power grid, it is imperative to have systematic design methods and tools to facilitate the design decisions.

## 3. MODELING SCENARIO

In our modeling approach, we consider a power system that is statically partitioned into several areas, each area having a Balancing Authority (BA) responsible for the local state estimation for its portion of the grid. PMUs associated with a BA generate and send data to the local BA through a dedicated link at frequencies ranging from 1-60 Hz. The proposed NASPInet architecture in [8] considers PMUs that forward their data to a Phasor Data Concentrator (PDC) or a Phasor Gateway (PGW). In this work, we solely model interactions between PMUs and PDCs following the network structure and properties characterized in [8], without loss of generality.

One essential component we model is the middleware that connects the distributed state estimators for each BA. As shown in Figure 1 (colored in green), this component mediates data exchange between the BAs in the pre-determined partitions of the grid. Local results from PDCs are primarily transferred to the middleware to be aggregated and time-aligned, and later multicast to all BAs. Global aggregation is necessary for time-aligning the faulty PMU readings of the entire system, and broadcasting this information to all the local state estimators for global situational awareness.

The distributed state estimation application involves multiple rounds of data exchange between models that are distributed both in the sense of computation and the power grid topology that they manage. At each estimation step, PMU sensor readings from the previous time period are obtained from the local Phasor Data Concentrator (PDC). PDCs from neighboring operational areas send the PMU data to a middleware infrastructure, which upon receiving data from all the PMUs, generates a single global index file, containing relative timestamps of the faulty readings received.

This error index file is then multicast to the distributed state estimators in the topology. The first iteration of distributed state estimation takes place locally at each local operational area. Upon completion, intermediate tie-line states (tie lines connect individual operational areas and are thus shared model elements) are exchanged between each area using the middleware. An additional decision component, which is also part of the middleware, receives these intermediate values and notifies each area when global convergence for the state estimation model has been achieved.

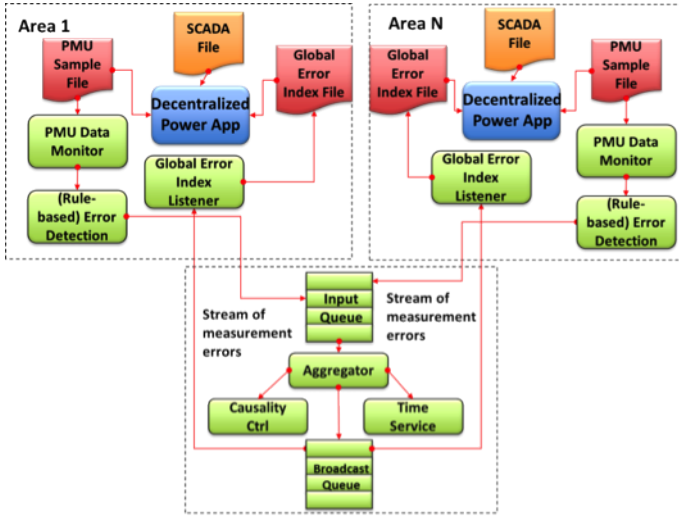


Figure 1: Overview of the middleware architecture for distributed state estimation

The number of iterations required for tie-line states to converge depends on several factors including PMU data quality, number of PMUs and the grid partitioning. Therefore, the number of iterations until global convergence remains dynamic throughout operation, even for a fixed grid topology.

## 4. MODELING APPROACH

We model the entire data flow from sensor devices to state estimators through the middleware components using the Ptolemy II design and simulation framework. The overall structure of the model is shown in Figure 2. In the following section, we present the modeling of critical elements in details.

### 4.1 Modeling PMUs

We model the PMUs of each area as an actor, called PMU Cluster. This actor is parameterized by the PMUCount parameter and generates the corresponding number of events every iteration interval (parameterized by the PMUPeriod parameter). This actor produces events at a given data frequency, typically 30 samples per second. The number of PMUs connected to each area can be specified as a parameter that determines the number of events produced by the cluster at each token firing during the simulation.

### 4.2 Modeling PDCs

A PDC is responsible for receiving data from multiple PMUs and produces an aggregated data packet for each PMU at an application specific rate. We only model the relaying function of a PDC, where it produces data packets for each PMU and processes the packets in a FIFO pattern.

### 4.3 Modeling State Estimators

The distributed state estimation runtime in practice is highly correlated with the number of PMUs associated with the BA running the state estimation. We model the state estimator at a BA as a node that receives SCADA data and PDC packets and produces results upon being triggered by

an incoming signal. We simplify the BA behavior by assuming that only the state estimation algorithm runs on the node. We model the state estimation algorithm as a composite actor that upon being triggered takes a specific amount of time to complete. The value to characterize the compute time of the state estimation has been calibrated by running the state estimation algorithm on a testbed deployed at Pacific Northwest National Laboratory [10].

The triggering mechanism for the distributed state estimation process is controlled by the *DSEControl* actor that produces a trigger signal whenever an index file or a file with intermediate results has arrived from the neighboring state estimators. Upon reception of the convergence signal, the distributed state estimation completes the computation for one time step, such as at the top of a minute. The distributed state estimator will then be triggered for the next time step.

## 4.4 Modeling Network

We model the network infrastructure connecting each PMU to the local PDC as a dedicated NASPInet link, as outlined in [8]. We then use another NASPInet model to describe the communication between PDC and the local BA. In this application, we solely consider the state estimation algorithm run on the BA components.

The network components we utilize are based on random-delay server topologies. Communication links are parameterized by their bandwidth, physical length and propagation speed. We assume fixed-length packets traveling through each link. We also assume an additional constant latency parameter for network characteristics that are not explicitly parameterized. The packet length and link characteristics are chosen to be consistent with NASPInet assumptions given in [8].

The mean delay is determined by link bandwidth, packet size, propagation speed and link length; and a Rician distribution around this mean is assumed for the network behavior. The choice of Rician distribution as opposed to the Gaussian assumption in [8] is to ensure mathematical soundness and physical consistency, since the network latency values generated by a Rician distribution are ensured to be strictly non-negative.

## 4.5 Modeling Middleware

The middleware plays an important role in time aligning and aggregating data from physically distant BAs. To simulate middleware behavior accurately, we abstract the behavior with a random-delay queuing structure with per-packet delays. The essential functions of this component are as follows:

**Aggregation.** Upon reception of all PMU data, the middleware aggregates time-stamps of all faulty data from the PMUs into a single index file and broadcasts the file to all balancing authorities. A PMU file is immediately processed upon reception, with the processing time modeled by a random delay, in model time. The overall completion of aggregation is determined by the last expected PMU file to complete processing. The stochastic delay is characterized by benchmarking results as discussed in section 5.

**Global Consistency of State.** One of the essential functionalities of the middleware is to moderate global consistency of state. In our scenario, BAs of neighboring areas exchange state information through peer-to-peer communi-

cation. A middleware component annotated as DSEConverged in the Ptolemy model monitors intermediate states to determine and declare global convergence. For simulation purposes, we assume a stable convergence pattern with a maximum number of iterations. The state estimators stop data exchange and work on the next round state estimation after the maximum number of iterations.

## 4.6 Model Simulation

The execution of the model is based on discrete event simulation. In Ptolemy, discrete event models are based on time-stamped events composed of a *token* (value) and a *tag* that denotes the time exchanged between actors. The notion of *time* discussed here may or may not be related to wall clock time. The discrete event scheduler guarantees that events are processed in timestamp order. More precisely, actors that have the earliest timestamp input events are executed first. In a discrete event model, all actors share a global notion of time, namely the model time at the particular level of hierarchy. This imposes a global ordering of events within the model and therefore ensures determinacy. A more formal explanation of the operational and denotational semantics can be found in [11].

Discrete event simulation is suitable for the scenario described in section 3. This is because the high-throughput devices communicating over the network with packets can be abstracted to discrete event components communicating via time-stamped events, where each network packet is represented by a discrete event in the Ptolemy execution.

## 5. APPLICATION STUDY

### 5.1 Experimental Setup

We explore a three-area power grid topology, where the high-level specifications of the distributed grid model conform to the IEEE 118-bus test case, as described in [10]. The study has three partitions shown in Figure 3, each representing one BA, such that Area 2 neighbors Area 1 and Area 3, and 1 and 3 do not share any direct bus connections [14]. In this topology, we assume the inter-area communications take place between two pairs of balancing authorities: BA1-BA2 and BA2-BA3.

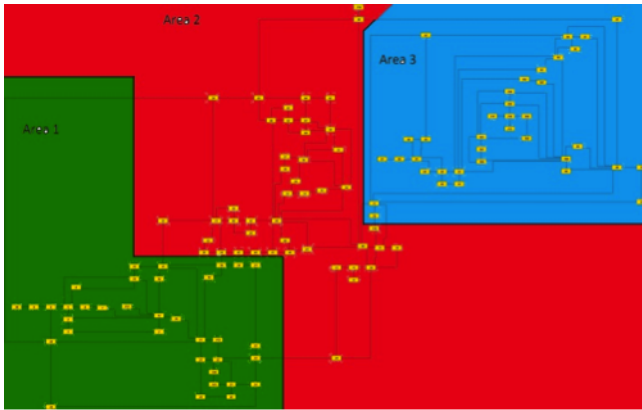


Figure 3: Topology graph of the partitioned IEEE 118-bus system (Green: Area 1, Red: Area 2, Blue: Area 3)

Table 1: Link specifications for the power grid network model

	Bandwidth	Length (mi)	Packet Size (bytes)	Average Latency (ms)
PMU to PDC	56 Kbps	50	128	19.3
PDC to BA	10 Mbps	300	67K	55.9
PDC to MW*	10 Mbps	350	67K	55.9
BA to BA	10 Mbps	300	134K	110.8
MW* to BA	-	-	-	60.0

\*MW: Middleware

The PMUs of each area produce data at 30 samples per second that is transmitted to the local PDC through a modeled NASPInet connection. The data are relayed at the PDC and sent to the local BA, as well as to the middleware component. Both of these deliveries use NASPInet connections.

In the case study, each PMU is connected to the local PDC by a 56Kbps communication line. The packet size of each PMU message is assumed to be 128 bytes in compliance with the data format assumed in [8]. Each of these links is chosen to be 50 miles long, representing an average physical distance from a bus to the nearest PDC. We assume each PDC is placed 300 miles away from the local BA and the middleware component is symmetrically located, 350 miles from the PDCs. Table 1 summarizes the experimental setup used to parameterize the model.

Upon reception of all PMU data, the middleware generates a global index file and concurrently broadcasts data through dedicated links to all three balancing authorities. We have developed a middleware prototype following the structure shown in Figure 1. The aggregation and queuing behavior is based on ActiveMQ [<http://activemq.apache.org/>], an Apache open source messaging server. The data communication between state estimators has been implemented using MeDiCi [7], which is built upon an open source enterprise service bus. This middleware implements a file connector that detects any arriving files and then automatically invokes a software component to process the file stream. This middleware prototype is integrated with the modeling and simulation environment in Ptolemy II. This means the delay of the middleware is characterized by actually running and measuring the middleware prototype, providing realistic latencies that are incorporated into our modeling results.

The delivery of the global index file to a BA triggers a state estimator. The run time of the state estimation algorithm is modeled based on benchmarking results obtained by running the state estimator algorithm on a cluster machine at PNNL [10].

Our model aims to establish simulation-based conclusions on the scalability requirement on the middleware design. For the distributed state estimation, the entire computation is expected to be completed in one minute intervals; in which the entire grid data collected for the past minute is used for state estimation at the BAs. Successful completion of this requirement depends on factors that can be outlined as network latency, number of distributed state estimation iterations until convergence, middleware latency and state estimation algorithm overhead.

### 5.2 Experimental Results

We evaluate the end-to-end delay of the distributed state estimation under two sets of benchmark results, both of

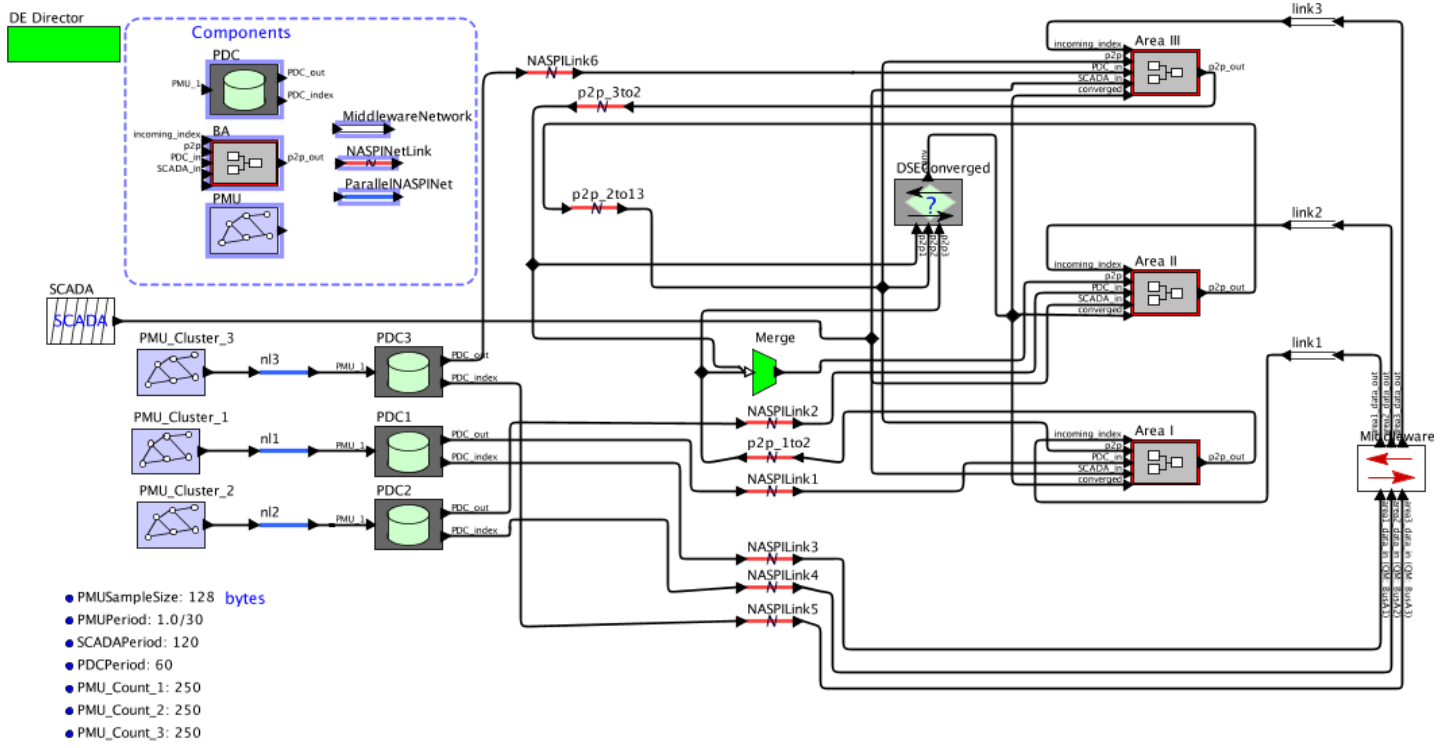


Figure 2: Overall Structure of Distributed Power Models

which are obtained using the middleware prototype and deployment discussed in section 5.1.

The first set of results is obtained on a per-file basis, where we benchmark per-file latency through the middleware queuing structure. 250 PMU files are sent simultaneously through the middleware aggregator. Figure 4 shows the distribution of the completion time of the distributed state estimation from the moment the first PMU file arrives to the moment all the state estimators stop. This benchmark results represent non-aggregating middleware behavior and produces an upper-bound for the actual middleware response. The non-aggregating middleware model applies per-file delays to each PMU being received. Under this scenario, upon reception and processing of all PMU files, the index file is produced immediately, without additional model delays.

The second set of results involves obtaining a single latency value for the entire iteration cycle. A timer measures the time gap between the first PMU file received at the middleware queue and the final time when the last PMU file has been processed. For the model, no per-file delays are applied. Instead, upon reception of all PMU files from all areas, a single file is produced and an overall delay is applied to the end product file.

The histogram given in Figure 5 shows the distribution of end-to-end completion time of the distributed state estimation for a 3-area topology with 250 PMUs in each area. The state estimators converge after a maximum of 4 iterations at each BA. The non-aggregating middleware model produces an end-to-end completion time within 16.06 seconds, while the aggregating middleware model has the end-to-end completion time of 23.27 seconds. The contrast of these results could be interpreted as an empirical aggrega-

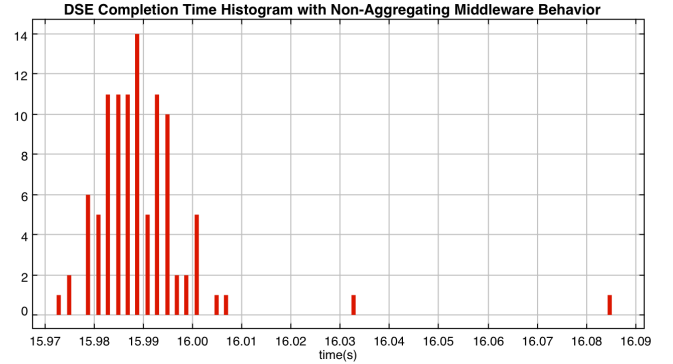
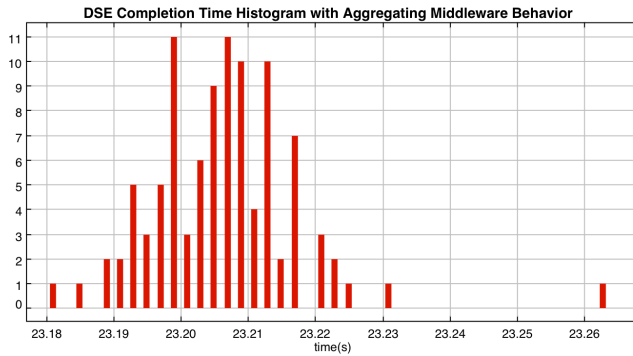


Figure 4: End-to-end delay of distributed state estimation

tion overhead close to 7 seconds. Still, even the worst case of the aggregating middleware is well within the requirement of 1 minute (the frequency interval for performing state estimation). The results demonstrate the middleware can scale to handle 750 PMU streams, and still satisfy the temporal requirements.

Both Figure 4 and Figure 5 demonstrate long-tailed middleware overhead behavior that is directly observed in the end-to-end completion times. These worst-case specifications particularly gain importance for the verification of temporal guarantees to meet deadlines in time-sensitive distributed applications on the grid.





**Figure 5: End-to-end delay of distributed state estimation**

## 6. CONCLUSION

In this paper, we present a model-based analysis of the end-to-end data flow of distributed power grid applications with a focus on the middleware scalability. Benchmark results were used to approximate middleware delay parameters under aggregating and non-aggregating scenarios. Using this modeling approach, we were able to quantify the impact of the middleware behavior on the end-to-end completion time of distributed state estimation. 750 PMU data streams were simulated and passed by the middleware to state estimators. The simulation results highlight the long-tail behavior of the middleware needs to be improved to reduce the end-to-end completion time and increase predictability.

Future work will focus on capturing additional error handling mechanisms in the model such as timeout mechanisms for dropped/delayed PMU packets. Design state exploration remains an important future work to improve scalability of power network models for more general grid topologies and dynamic partitions. Specifically, we would like to model arbitrary partitions of a given grid topology with minor user intervention to the power grid model.

Additional work in progress aims at replacing the data source structure; namely the PMU and PDC network, with a high-throughput, intensive data storage middleware [15]. The compositional model-based approach will also be useful in this work, since the end-results will reflect temporal system properties when a certain middleware is chosen and will provide important insights on worst-case execution times of distributed power grid applications.

## 7. REFERENCES

- [1] B. Akyol, J. Haack, B. Carpenter, S. Ciraci, M. Vlachopoulou, and C. Tews. Volttron: An agent execution platform for the electric power system. In *Third International Workshop on Agent Technologies for Energy Systems*, June 2012.
- [2] R. Bobba, E. Heine, H. Khurana, and T. Yardley. Exploring a tiered architecture for naspinet. In *Innovative Smart Grid Technologies (ISGT), 2010*, pages 1–8, jan. 2010.
- [3] A. Bose, K. Poon, and R. Emami. Implementation issues for hierarchical state estimators. Technical report, Department of Computer Science, Michigan State University, 2010. [Online; accessed 27-September-2012].
- [4] J. Cai, Z. Huang, J. Hauer, and K. Martin. Current status and experience of wams implementation in north america. In *Transmission and Distribution Conference and Exhibition: Asia and Pacific*, pages 1–7, 2005.
- [5] P. Derler, E. Lee, and A. Vincentelli. Modeling cyber-physical systems. *Proceedings of the IEEE*, 100(1):13–28, Jan. 2012.
- [6] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity - the ptolemy approach. In *Proceedings of the IEEE*, pages 127–144, 2003.
- [7] I. Gorton, A. Wynne, Y. Liu, and J. Yin. Components in the pipeline. *Software, IEEE*, 28(3):34–40, may-june 2011.
- [8] R. Hasan, R. Bobba, and H. Khurana. Analyzing naspinet data flows. In *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pages 1–6, march 2009.
- [9] J. C. Jensen, D. Chang, and E. A. Lee. A model-based design methodology for cyber-physical systems. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1666–1671, July 2011.
- [10] S. Jin, Y. Chen, M. Rice, Y. Liu, and I. Gorton. A testbed for deploying distributed state estimation in power grid. In *IEEE Power & Energy Society General Meeting*, July 2012.
- [11] E. A. Lee. Modeling concurrent real-time processes using discrete events. *Ann. Softw. Eng.*, 7(1-4):25–45, June 1999.
- [12] E. A. Lee. Cyber physical systems: Design challenges. In *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, May 2008. Invited Paper.
- [13] E. A. Lee. Cps foundations. In *Proc. of the 47th Design Automation Conference (DAC)*, pages 737–742. ACM, June 2010.
- [14] Y. Liu, W. Jiang, S. Jin, M. Rice, and Y. Chen. Distributing power grid state estimation on hpc clusters - a system architecture prototype. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 1467–1476, may 2012.
- [15] J. Yin, A. Kulkarni, S. Purohit, I. Gorton, and B. Akyol. Scalable real time data management for smart grid. In *Proceedings of the Middleware 2011 Industry Track Workshop, Middleware '11*, pages 1:1–1:6, New York, NY, USA, 2011. ACM.