ABSTRACT

ZHANG, JIANHUA. Distributed Cyber-Physical Algorithms for Wide-Area Monitoring and Control of Power Systems using Cloud-in-the-Loop Feedback. (Under the direction of Aranya Chakrabortty.)

This dissertation work focuses on several cyber-physical algorithms to evaluate the impacts of realistic Internet communication networks on the wide-area real-time oscillation monitoring and damping control of a large power system network. In Chapter 2 of this report we recap three computational approaches to estimate electro-mechanical oscillation modes of large power systems using the measurements of voltages, currents and phasor angles available from Phasor Measurement Units (PMUs), namely decentralized least squares (DLS), its recursive implementation (RLS) and distributed Prony with alternating direction method of mulitplier (PronyADMM). In the first case of DLS, these PDCs receive data from multiple PMUs spread across the physical network, and run an independent DLS algorithm to compute the coefficients of the characteristic polynomial of the transfer function of the power system. These estimates are finally transmitted to a central server where the estimated coefficients are averaged, and the global estimates of the dominant oscillation modes are evaluated offline. In the second case, the same algorithm is executed recursively in real-time. In the third case of distributed PronyADMM, at any iteration, the local estimators receive PMU measurements from within their own respective areas, run a local Prony ADMM-based consensus algorithm, and communicate their local estimates and dual parameters to a central estimator. The central estimator calculates the global consensus variable, and broadcasts it back to each local estimator for their next iteration. It was shown that this global value converges to the global solution as the number of iterations tends to infinity.

In Chapter 3, we develop a suite of asynchronous distributed optimization algorithms for wide-area oscillation estimation in power systems using Alternating Direction Method of Multipliers (ADMM) to mitigate the perennial problem of asynchrony in wide-area communication networks. We first consider a probabilistic traffic model for modeling delays in a typical wide-area communication network, and study how the delays disturb the process of information exchange between distributed Phasor Data Concentrators (PDC) that are employed to execute this consensus algorithm in a coordinated fashion. Finally, we propose four different strategies by which the convergence rate and accuracy of this consensus algorithm can be made immune to the asynchrony resulting from the network traffic. We carry out extensive simulations to show possible numerical instabilities and sensitivities of the ADMM convergence on our proposed strategies and the delay distribution parameters under two simple averaging strategies. Our results exhibit a broad view of how the convergence of any distributed estimation algorithm in a generic cyber-physical system depends strongly on the uncertainties of the underlying

communication models.

To address the cyber security issues of these cyber-physical algorithms, we further validate the security enhancement through the distributed optimization in Chapter 4 by designing specific attack-resilient strategies for both centralized Recursive Least Squares (RLS) algorithm and distributed PronyADMM algorithm during their implementation on a federated testbed between NC State University (RTDSLab) and the Information Sciences Institute (DETERLab). In the centralized RLS, the resilient strategy makes the central PDC can continue to accommodate the rest of PMU machines once one or some of PMU machines is/are attacked. While in the distributed PronyADMM architecture, we design two redundancy-based resilient strategies: 1) standby backup server 2) mutual coordination between the local and the central PDCs. The emulation results show that the distributed algorithm is highly more resilient to communication failures than centralized.

In Chapter 5, we propose the use of cloud-computing platforms and virtual network laboratories such as GENI (Global Environment for Network Innovations), together with high-speed software defined networks such as Internet2 to combat various cyber-physical implementation challenges for wide-area oscillation control of large power systems using Synchrophasors. Experimental results from a cloud-in-the-loop testbed environment are reported to support our proposed architecture.

To better validate the implementation issues of these algorithms for wide-area oscillation monitoring problem in large power systems, in Chapter 6, we first introduce a cyber-physical testbed, named the ExoGENI-WAMS testbed. Based on the Hardware-In-Loop(HIL) concept, the physical part of this testbed consists of the Real-time Digital Simulators (RTDS), which is used to simulate high fidelity detailed models of large power systems, and the PMU-based Wide-Area Measurement System (WAMS), which can capture the dynamic responses of large power system via real hardware Phasor Measurement Units and transfer them to the Phasor Data Concentrator. To better utilize next-generation cyber-infrastructure technologies that including high-speed virtual networking, high performance cloud computing, and virtualization and data management at the cyber level, we further integrated the PMU-based WAMS framework with the US-wide ExoGENI network through a state-funded, metro-scale, multi-layered dynamic 100gbps optical network testbed called Breakable Experimental Network (BEN). Next, we first conduct the end-to-end delay evaluation for the least-squares-based algorithms in both centralized and decentralized fashions, as well as its recursive implementation on the ExoGENI-WAMS testbed. The results show how the end-to-end computational and communication delays as well as the corresponding accuracy of estimation vary for both of these decentralized algorithms with respect to their centralized counterpart. Then we implemented the distributed storage system and distributed wide-area control algorithm in this testbed.

Distributed Cyber-Physical Algorithms for Wide-Area Monitoring and Control of Power Systems using Cloud-in-the-Loop Feedback

by
Jianhua Zhang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2016

APPROVED BY:

Frank Mueller

Ning Lu

Subhashish Bhattacharya

Yufeng Xin

Aranya Chakrabortty
Chair of Advisory Committee

DEDICATION

This dissertation is dedicated to my advisor, committee members, family, friends, parent.

BIOGRAPHY

Jianhua Zhang was born in Jiangsu, China. She graduated with her bachelor and master degrees from the Jimei University and Xiamen University at Xiamen of China, respectively, with majors in Industry Automation and System Engineering. After graduation in 2005, she joined the Fujian Nokia telecommunication Ltd where she worked as a telecommunication engineer for two years and laid down the solid foundation of communication technology. She then obtained her master degree in Electrical Engineering from New Mexico Tech, Socorro, NM in 2010.

Inspired to build the future energy internet, she continued to pursue her PhD degree in the Department of Electrical and Computer Engineering of North Carolina State University, Raleigh, NC in 2011. During her PhD period, she worked as a research assistant on both National Science Foundation and Department of Energy sponsored research focusing on the distributed cyber-physical optimizations of wide-area monitoring and control by the cloud-in-loop feed-back. Jianhua will be graduating with her PhD in December 2016 with the hopes of continuing to build the future energy internet through both theoretical and practical techniques. Especially, her research interests are power system dynamics and monitoring, especially in the development and implementation of cyber-physical algorithms using the Synchrophasor technology.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Dr. Aranya Chakrabortty for imparting his knowledge of power system and control theory on me as well as allowing me venture off into unfamiliar territory and multiple research projects. I cannot thank him enough for his trust and willingness to let me work on the majority of my PhD in different state. He has helped me to improve my research skills through involving into several research projects and publishing our own research contributions. Thank you.

To my committee members, Drs. Yufeng Xin, Frank Mueller, Ning Lu, and Subhashish Bhattacharya, your advice, encouragement, and support have been an immense help over the years. I would also like to thank Dr. Alefiya Hussain from Information Science Institute, University of Southern California, Los Angeles, CA for mentoring me, sharing your wealth of knowledge of networking with me.

I would not have succeeded without the love and moral support from my family. Gaopeng, you will never know how much I appreciate all the times you were there for me when I felt run down and being right beside me to celebrate the joyous moments. Ziya, my daughter, thank you for accompanying with me and bringing me tons of laugh. I love you two forever. To my parents, I appreciate your constant love and support throughout my long educational endeavors. To the rest of my family, thank you for various words of wisdom over the years, helpfulness, delicious meals, and wonderful times when we are able to get together and chat up.

To my friends, your unwavering patience and kindness has meant the world to me. Thank you to Matthew Weiss, Seyedbehzad Nabavi, Abhishek Jain, Mang Liao, Li Wang, and Yao Meng. Matthew and Nabavi for joyfully co-working on our research projects and papers. Li, I do remember our happy time on all kinds of shows at our school theaters. Thank you to my lab mates and the graduate students I have gotten to know throughout my time at NCSU, especially in the FREEDM center. The FREEDM party times, seminars, and random conversations will be treasured memories.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Following the Northeast blackout of 2003, Wide-Area Measurement System (WAMS) technology using Phasor Measurement Units (PMUs) has largely matured for the North American grid [36]. However, as the number of PMUs scales up into the thousands in the next few years under the US Department of Energy's smart grid demonstration initiative, Independent System Operators (ISO) and utility companies are struggling to understand how the resulting gigantic volumes of real-time data can be efficiently harvested, processed, and utilized to solve wide-area monitoring and control problems for any realistic power system interconnection. It is rather intuitive that the current state-of-the-art centralized communication and information processing architecture of WAMS will no longer be sustainable under such a data explosion, and a completely distributed, self-adaptive cyber-physical architecture will need to be developed [2,3]. In the Eastern Interconnection (EI) of the US grid, for example, about 60 PMUs are currently streaming data via the Internet to a super phasor data concentrator (SPDC) which is handling about 100,000 data points per second. This architecture will no doubt become untenable as the EI scales up to 300-400 PMUs by 2015.

The North American Synchrophasor Initiative (NASPI) is currently addressing this architectural problem by developing new communication and computing protocols for WAMS through NASPInet and Phasor Gateway [38] to facilitate PMU data communication between multiple utilities and control centers. Data from PMUs owned by any specific utility is first passed through the Phasor Gateway, and the output is either sent to 'Applications' or to a 'Historian' for local use, or sent directly to a NASPInet data bus for sharing information with other utilities, which themselves may be exchanging data through the same bus from PMUs under their own local territories with the neighboring control regions. Research is currently being carried out by the Data and Network Management Task Team (DNMTT) of NASPI on the implementation of this distributed architecture with the prime research focus being - protocols, Quality-of-Service, latency, bandwidth and security [8].

However, very little attention has yet been paid to perhaps three of the most critical consequences of this envisioned distributed architecture - namely (1) formulation of traditional centralized strategies for wide-area monitoring and control as distributed algorithms, (2) investigation of numerical stability and convergence properties of these algorithms when they are implemented in a completely asynchronous environment that is bound to arise in any practical wide-area communication network, and (3) analysis of typical cyber physical challenges including end-to-end delay and attack resiliency, when these algorithms are implemented in a completely distributed cyber-physical architecture. Partly due to a lack of a cyber-physical research infrastructure and partly due to the priorities set forth by PMU installations, the NASPI community has not yet delved into investigating how the currently used centralized algorithms for wide-area monitoring [9] and control [10] can be translated into a distributed computing framework once the aforementioned decentralized WAMS architecture is realized in the next three to four years. Development of such algorithms will obviously be imperative not only for increasing reliability by eliminating single-point failures, but also for minimizing network transit. As shown in the recent work on NASPInet in [11], transmitting data across a wide-area communication network (WAN) is expensive, the links can be relatively slow, and the bandwidth per dollar will indeed grow slower than other computing resources leading to distributed PMU data processing followed by transmission of full or partially processed outputs as a natural choice [12, 13].

Motivated by this challenge, the primary focus of this research is to develop distributed cyber-physical algorithms for studying the impact of the actual communication network on the real-time wide-area monitoring of large power systems, and to validate them with extensive offline simulations, as well as implement them on the actual cyber-physical platform. The research group led by Dr. Chakrabortty is currently investigating the theory behind distributed estimation and control, and recently proposed three different distributed communication and computational architectures to address problem (1) [24]. In this report, we first develop a family of distributed asynchronous optimization algorithms to answer the fundamental question on how asynchrony in a communication network can influence the performance for one of the most critical wide-area monitoring applications, namely, modal estimation of electro-mechanical oscillations using Synchrophasors to address problem (2), and secondly address problem (3) by implementing both a suit of least-squares-based algorithms and ADMM-based Prony algorithms on the actual cyber-physical testbed.

We start our research with recap of several distributed cyber-physical algorithms for wide-area oscillation monitoring in the large power system network in Chapter 2. We first consider two computational approaches, namely Decentralized Least-Squares(DLS) and its recursive implementation (RLS). In the first case, these Phasor Data Concentrators (PDCs) receive data from multiple Phasor Measurement Units (PMU) spread across the physical network, and run

an independent DLS algorithm to compute the coefficients of the characteristic polynomial of the transfer function of the power system. These estimates are finally transmitted to a central server where the estimated coefficients are averaged, and the global estimates of the dominant oscillation modes are evaluated. In the second case, the same algorithm is executed recursively in real-time. To avoid the prior known disturbance input information, we consider the existing well-known modal estimation algorithm - Prony analysis method. It can be recast as a real-time distributed optimization problem by formulating it as a global consensus problem, and solving it using Alternating Direction Method of Multipliers (ADMM) [33]. Thus, at any iteration, the local estimators receive PMU measurements from within their own respective areas, run a local consensus algorithm, and communicate their estimates to a central estimator. The central estimator averages all estimates, and broadcasts the average back to each local estimator as the consensus variable for their next iteration. It was shown that this average value converges to the global solution as the number of iterations tends to infinity.

However, a critical assumption in synchronous ADMM was that the communication between the local PDCs/estimators and the central PDC/estimator is completely synchronized. In reality, if the data exchange is implemented on a wide-area communication network such as Internet, that perfect synchrony cannot be maintained. The key question, therefore, is how the convergence of the ADMM algorithm gets affected by such asynchronies in communication? In Chapter 3, we answer this question in the three following ways. We first impose a probability model for the communication delays between the local PDCs and the central PDC. Second, we implement two strategies of averaging at the central PDC by defining a delay threshold d^*. If any of the local estimates does not arrive within time d^*, the central PDC either skips them (Strategy 1), or uses the latest value from the previous iterations (Strategy 2). We carry out simulations to show the possible instabilities and sensitivities of ADMM convergence on the delay distribution parameters under these two averaging strategies. However, these two proposed strategies are only applied on the uni-directional delays. Thirdly, we further propose four different update rules at both local PDCs and central PDC by which the convergence rate and accuracy of this consensus algorithm can be made immune to the asynchrony resulting from the network traffic as much as possible. We also carry out extensive simulations using an IEEE prototype power system model to show possible numerical instabilities and sensitivities of the ADMM convergence on our proposed update rules.

In Chapter 4, we further verify the security enhancement through the distributed architecture. When these two cyber-physical algorithms of centralized Recursive Least Squares (RLS) algorithm and distributed PronyADMM algorithm are implemented in the actual communication network, they are prone to failures and attacks on their physical infrastructure, as well as cyber attacks on their data management and communication layer [17]. To this end, we design specific attack-resilient strategies for both centralized and distributed architectures. For the centralized

architecture of RLS, the resilient strategy makes the central PDC to continue to accommodate the rest of PMU machines once one or some of PMU machines is/are attacked. Compared to the centralized case, the distributed PronyADMM architecture has two redundancy-based resilient strategies: 1) standby backup server 2) mutual coordination between the local and the central PDCs. We emulated those two algorithms with their respective attack-resilient strategies on a federated testbed between NC State University (Phasorlab) and the Information Sciences Institute (DETERLab), and results show that the distributed algorithm is highly more resilient to communication failures than centralized.

Beside the real-time monitoring of oscillation of the large power system, the wide-area measurement system technology is also an ideal way to control instabilities caused by oscillations of the bulk power grid. However, the challenge is the bottleneck in data communication and computation, which cannot be handled by today's Internet. In Chapter 5, we propose the use of cloud-computing platforms to study the impact of data communication and computation on the performance of the damping control of the WECC transmission grid. We first formulate the reduced model of a multi-area power system following its dynamic equivalent via time-scale separation. Based on the reduced WECC system model, we design the typical distributed LQR control algorithm to combat both small-signal and large-signal disturbances. Finally, we compare the performance between local PSS controller and wide-area controller using the RSCAD based simulation and the simulation result shows that the wide-are controller has much better control performance.

The envisioned wide-area monitoring and control system for a large power system network is a typical cyber-physical system. Due to the specific privacy policy and limited location accessibility of practical PMU measurement data, our research group has started developing a Hardware-In-Loop(HIL) simulation framework at the physical level where high fidelity detailed models of large power systems can be simulated in Real-time Digital Simulators (RTDS), and the dynamic responses can be captured via real hardware Phasor Measurement Units from multiple vendors including Schweitzer Engineering Lab and ABB Inc. To use next-generation cyber-infrastructure technologies that including high-speed virtual networking, high performance cloud computing, and virtualization and data management at the cyber level, we further integrated the PMU-based WAMS framework with the US-wide ExoGENI network through a state-funded, metro-scale, multi-layered dynamic 100gbps optical network testbed called Breakable Experimental Network (BEN). The resulting testbed is referred to as the ExoGENI-WAMS testbed. Therefore, we can validate the distributed cyber-physical algorithms proposed in Chapter 2 and 3 to study the typical cyber-physical challenges (e.g. end-to-end delay and attack-resiliency). Next, we conduct the end-to-end delay evaluation for the lease-square-based algorithms in both centralized and decentralized fashions, as well as its recursive implementation on the ExoGENI-WAMS testbed. The results show how the end-to-end computational and

communication delays as well as the corresponding accuracy of estimation compare for both of these decentralized algorithms with respect to their centralized implementation. Finally, to verify the control close loop of the testbed system, we build up the particular cloud-in-the-loop test system consisting of 1) the Real-time Digital Simulators (RTDS) for simulating the WECC power system, 2) 5 hardware Phasor Measurement Units (PMU) for the measurement of power grid, 3) the ExoGENI cloud computing platform for data communication and computation. We finally implement the proposed state-feedback LQR control algorithm for small-signal oscillation damping of the five-area model of WECC system and the experimental results from such a hardware-in-the-loop test platform showed the support our proposed architecture.

1.1 Contributions

1. Convergence analysis of Asynchronous ADMM (A-ADMM) algorithms on the delay distributed parameters under two averaging strategies.

2. Development of a suit of heuristic variants of distributed Asynchronous-ADMM (A-ADMM) optimization algorithms and convergence analysis of the A-ADMM algorithms on different update strategies.

3. Security enhancement using distributed ADMM optimization.

4. Validation of cyber-physical algorithms on ExoGENI-WAMS testbed.

1.2 Future Tasks

1. Develop delay-robust algorithms using hybrid strategy and spatial correlation.

2. Validation of A-ADMM algorithms and their integration with DHT-based storage system.

Chapter 2

Wide-Area Oscillation Monitoring Algorithms for Power System using Optimization

This chapter studies the oscillation mode estimation problem of a large, geographically distributed power system network using distributed processing of Synchrophasor measurements. We consider three computational approaches, namely decentralized least squares (DLS), its recursive implementation (RLS) and distributed Prony with alternating direction method of multiplier (PronyADMM). Both decentralized LS and RLS algorithms are executed using multiple phasor data concentrators (PDC), deployed as virtual computing machines communicating over a fiber-optic communication network. In the first case, these PDCs receive data from multiple Phasor Measurement Units (PMU) spread across the physical network, and run an independent DLS algorithm to compute the coefficients of the characteristic polynomial of the transfer function of the power system. These estimate are finally transmitted to a central server where the estimated coefficients are averaged, and the global estimates of the dominant oscillation modes are evaluated. In the second case, the same algorithm is executed recursively in real-time. In contrast, the PronyADMM algorithm formulates the mode estimation problem as a global consensus problem for the coefficients of the characteristic polynomial of the system, and then solve it using Alternating Direction Method of Multipliers (ADMM) [33]. The physical grid is assumed to be divided into multiple balancing regions or *areas*, which may or may not be coherent, but belong to different utility companies. PMUs in each area communicate their data in real-time to estimator(s) or Phasor Data Concentrators (PDC) located at the local control center via a Virtual Private Network (VPN). These local PDCs can then share information between each other and also with a central PDC located at the Independent System Operator (ISO) through a wide-area communication network, examples of which can range from standard

Internet to any Software Defined Network (SDN).

2.1 Power System Oscillation Model

Consider a power system network consisting of m synchronous generators and n_l loads connected by a given topology. Without loss of generality, we assume buses 1 through m to be the generator buses and buses $m+1$ through $m+n_l$ to be the load buses. Let P_i and Q_i denote the total active and reactive powers injected to the ith bus ($i = 1, \ldots, m+n_l$) from the network that is calculated as:

$$P_i = \sum_{k=1}^{m+n_l} V_i^2 r_{ik}/z_{ik}^2 + V_i V_k \sin(\theta_{ik} - \alpha_{ik})/z_{ik}, \tag{2.1a}$$

$$Q_i = \sum_{k=1}^{m+n_l} V_i^2 x_{ik}/z_{ik}^2 - V_i V_k \cos(\theta_{ik} - \alpha_{ik})/z_{ik}, \tag{2.1b}$$

where $V_i\angle\theta_i$ is the voltage phasor at the ith bus, $\theta_{ik} = \theta_i - \theta_k$, r_{ik} and x_{ik} are the resistance and reactance of the transmission line joining buses i and k, $z_{ik} = \sqrt{r_{ik}^2 + x_{ik}^2}$, and $\alpha_{ik} = \tan^{-1}(r_{ik}/x_{ik})$. The electromechanical model of the power system can be described as a system of differential-algebraic equations (DAE) [19] as follows:

$$\dot{\delta}_i = \omega_s(\omega_i - 1) \tag{2.2a}$$

$$M_i\dot{\omega}_i = P_{m_i} - P_{e_i} - D_i(\omega_i - 1),\ i = 1, .., m \tag{2.2b}$$

with associated power balance equations given by

$$\begin{array}{cc} P_{e_i} + P_i - P_{L_i} = 0, & Q_{e_i} + Q_i - Q_{L_i} = 0, \\ P_k - P_{L_k} = 0, & Q_k - Q_{L_k} = 0, \end{array} \tag{2.3}$$

for $i = 1, \ldots, m$ and $k = m+1, \ldots, m+n_l$, where δ_i, ω_i, M_i, D_i, P_{m_i}, P_{e_i}, and Q_{e_i} denote the internal angle, speed, inertia, damping, mechanical power, active and reactive electrical powers produced by the ith generator, respectively, and P_{L_k} and Q_{L_k} denote the active and reactive powers of the loads at the kth bus. The DAE (5.2) can be converted to a system of purely differential equations by relating the algebraic variables V_i and θ_i in (5.1) to the system state variables (δ, ω) and then substituting them back in (5.2) via Kron reduction. The resulting system is a fully connected network of n second-order oscillators with $l \leq m(m-1)/2$ tie-lines. Let $\tilde{E}_i = E_i\angle\delta_i$ denote the internal voltage phasor of the ith machine. The electromechanical

dynamics of the ith generator in *Kron*'s form can be written as:

$$\dot{\delta}_i = \omega_s(\omega_i - 1), \tag{2.4a}$$

$$M_i\dot{\omega}_i = P_{mi} - P_i - D_i(\omega_i - 1), \tag{2.4b}$$

$$P_i = \sum_k E_i E_k \big(\frac{X_{ik}}{Z_{ik}^2}\sin(\delta_{ik}) - \frac{R_{ik}}{Z_{ik}^2}\cos(\delta_{ik})\big),$$

where, $i = 1, \ldots, m$, $Z_{ij}^2 = R_{ij}^2 + X_{ij}^2$, R_{ij} and X_{ij} denote the resistance and reactance of the line connecting the ith and j^{th} generator in the Kron's form, respectively, and $\delta_{ik} = \delta_i - \delta_k$. Linearizing (5.3) about the equilibrium $(\delta_{i0}, 1)$ results in the small signal state space model:

$$\left[\frac{\Delta\dot{\delta}}{\Delta\dot{\omega}}\right] = \underbrace{\left[\begin{array}{c|c} 0_{n\times n} & \omega_s I_{n\times n} \\ \hline \mathcal{M}^{-1}\mathcal{L} & \mathcal{M}^{-1}\mathcal{D} \end{array}\right]}_{\mathcal{A}} \left[\frac{\Delta\delta}{\Delta\omega}\right] + \underbrace{\left[\frac{0}{\mathcal{M}^{-1}\mathbf{e}_j}\right]}_{\mathcal{B}} u,$$

$$\mathbf{y} = \mathrm{col}(\Delta\delta_i,\ \Delta\omega_i), \quad \text{for} \quad i \in S \tag{2.5}$$

where $\Delta\delta = \begin{bmatrix}\Delta\delta_1 & \cdots & \Delta\delta_m\end{bmatrix}^T$, $\Delta\omega = \begin{bmatrix}\Delta\omega_1 & \cdots & \Delta\omega_m\end{bmatrix}^T$, $I_{m\times m}$ denote the $(m \times m)$ identity matrix, $\mathcal{M} = \mathrm{diag}(M_i)$ and $\mathcal{D} = \mathrm{diag}(D_i)$ are the $(m \times m)$ diagonal matrices of the generator inertias and damping factors, $\mathbf{e}_j$ is the jth unit vector with all elements zero but the jth element that is 1, considering that the input is modeled as a change in the mechanical power in the jth machine. However, since we are interested only in the oscillatory modes or eigenvalues of $\mathcal{A}$, this assumption is not necessary and the input can be modeled in any other feasible way such as faults and excitation inputs. The matrix $\mathcal{L}$ in (2.5) is the $(m \times m)$ Laplacian matrix of the form:

$$[\mathcal{L}]_{i,j} = \frac{E_i E_j}{Z_{ij}^2}\big(X_{ij}\cos(\delta_{i0} - \delta_{j0}) + R_{ij}\sin(\delta_{i0} - \delta_{j0})\big) \quad i \neq j,$$

$$[\mathcal{L}]_{i,i} = -\sum_{k=1}^{n}[\mathcal{L}]_{i,k}. \tag{2.6}$$

Let us denote the ith eigenvalue of the matrix $\mathcal{M}^{-1}\mathcal{L}$ by $\hat{\lambda}_i$. The largest eigenvalue of this matrix is equal to 0, and all other eigenvalues are negative, i.e. $\hat{\lambda}_m \leq \cdots \leq \hat{\lambda}_2 < \hat{\lambda}_1 = 0$. The eigenvalues of $\mathcal{A}$ are denoted by $\lambda_i = (-\sigma_i \pm j\Omega_i)$, $(j = \sqrt{-1})$, where $\Omega_i = \sqrt{|\hat{\lambda}_i|}$ is denoted as the ith osscillation frequency, $\sigma_i > 0$ is denoted as the ith damping factor. Our objective is to estimate these eigenvalues using measurements of voltage, phase angle, and frequency from PMUs in both decentralized and distributed fashions. For this purpose, we next present the DLS, RLS and distributed Prony with ADMM algorithms, and explain their cyber-physical implementation architectures.

2.2 Decentralized Least-Squares-based Algorithms

We open the problem by considering a fixed *input* bus, i.e., a node through which a disturbance input $u(t)$ enters the system, and two distinct output nodes, say bus p and bus q, (which may or may not be the same as the input bus) where PMUs are installed, i.e., nodes whose outputs $y_p(t)$ and $y_q(t)$ are known. As stated above, in practice y may refer to either voltage magnitude, or phase angle of frequency recorded by the PMU at that specific bus. Also, in reality, there may be many more than just two outputs. But for simplicity of discussion, we just restrict our discussion to two outputs measured respectively by two PMUs. Since there are m generators, each modeled by a second-order dynamic model, the total system order is $n = 2m$. The corresponding discrete-time transfer functions defined for nodes p and q from the small-signal disturbance input $u(t)$ can be expressed as, respectively,

$$G_p(z) = \frac{Y_p(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + ... + b_{m_p} z^{-m_p}}{1 + a_1 z^{-1} + ... + a_n z^{-n}} \tag{2.7}$$

$$G_q(z) = \frac{Y_q(z)}{U(z)} = \frac{c_0 + c_1 z^{-1} + ... + c_{m_q} z^{-m_q}}{1 + a_1 z^{-1} + ... + a_n z^{-n}} \tag{2.8}$$

where $m_p \leq n$ and $m_q \leq n$ are the orders of the respective zero polynomials. Taking inverse z-transform, these z-domain transfer functions (2.7-2.8) can be converted into the sequence-domain equations represented by the block-matrix at the sample index $k \in \{0, 1, ..., \infty\}$, as

$$y_p(k) = \begin{bmatrix} \phi_p(k) & U_p(k) \end{bmatrix} \begin{bmatrix} \gamma_3 \\ \gamma_1 \end{bmatrix} \tag{2.9}$$

$$y_q(k) = \begin{bmatrix} \phi_q(k) & U_q(k) \end{bmatrix} \begin{bmatrix} \gamma_3 \\ \gamma_2 \end{bmatrix} \tag{2.10}$$

where,

$$\begin{aligned}
\phi_p(k) &= \begin{bmatrix} y_p(k-1) & y_p(k-2) & \cdots & y_p(k-n) \end{bmatrix} \\
\phi_q(k) &= \begin{bmatrix} y_q(k-1) & y_q(k-2) & \cdots & y_q(k-n) \end{bmatrix} \\
U_p(k) &= \begin{bmatrix} u(k) & u(k-1) & \cdots & u(k-m_p) \end{bmatrix} \\
U_q(k) &= \begin{bmatrix} u(k) & u(k-1) & \cdots & u(k-m_q) \end{bmatrix} \\
\gamma_1 &= \begin{bmatrix} b_0 & b_1 & \cdots & b_{m_p} \end{bmatrix} \\
\gamma_2 &= \begin{bmatrix} c_0 & c_1 & \cdots & c_{m_q} \end{bmatrix} \\
\gamma_3 &= \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_n \end{bmatrix}
\end{aligned}$$

Our objective is to simply estimate the common characteristic polynomial of the two transfer functions captured by the parameter vector γ_3 from the known input sequence $u(k)$ and the output sequences $y_p(k)$ and $y_q(k)$. Once estimated, these coefficients can be used to compute the system poles, and to get an indication of the dominant frequencies of oscillation and their corresponding damping factors. To achieve this purpose, without any loss of generality, we assume the incoming disturbance $u(t)$ to be an impulsive input, and apply a batch as well as recursive decentralized least squares approach to compute a global estimate for γ_3.

2.2.1 Centralized vs Decentralized Least-Squares Algorithms

We are interested in investigating two problems: 1) how much accuracy we may have to lose, and 2) how much total time including communication and computation time we can save, if instead of considering the centralized implementation of the least-squares estimation of $\beta = -\gamma_3$, using both $y_p(k)$ and $y_q(k)$ taken together, we consider each of them individually leading to decentralized, independent local estimates β_p and β_q, and thereafter merge these local estimates to form the global estimate β. We first consider the centralized estimation of β with both $y_p(k)$ and $y_q(k)$ stacked together in the measurement vector. From (2.9)-(2.10), we can write

$$\underbrace{\begin{bmatrix} y_p(k) \\ y_q(k) \end{bmatrix}}_{A(k)} = \underbrace{\begin{bmatrix} \phi_p(k) & U_p(k) & 0 \\ \phi_q(k) & 0 & U_q(k) \end{bmatrix}}_{B(k)} \underbrace{\begin{bmatrix} \gamma_3 \\ \gamma_1 \\ \gamma_2 \end{bmatrix}}_{\Lambda} \tag{2.11}$$

By assuming that any variable with a negative sample index is zero by default, we construct matrices A and B for $k \in \{0, 1, ..., K\}$ with $K > n$ being a sufficiently large integer, as below,

$$\begin{aligned} A &= col(y_p(1), y_q(1), y_p(2), y_q(2), ..., y_p(K), y_q(K)) \\ B &= \begin{bmatrix} \phi_p(1) & U_p(1) & 0 \\ \phi_q(1) & 0 & U_q(1) \\ \phi_p(2) & U_p(2) & 0 \\ \phi_q(2) & 0 & U_q(2) \\ \vdots & \vdots & \vdots \\ \phi_p(K) & U_p(K) & 0 \\ \phi_q(K) & 0 & U_q(K) \end{bmatrix}_{2K \times (n+m_p+m_q+2)} \end{aligned} \tag{2.12}$$

The problem in the centralized case, therefore, is to generate the parameter vector Λ that solves $\Lambda = B^{-1}A$ (the matrix inverse operator here means pseudo inverse), then extract the first n entries of Λ, flip their sign to get the common parameter vector β, which can be written as

$\beta = -\left[B^{-1}A\right]^{+n}$.

On the other hand, the decentralized estimation is carried out using the two node outputs independently. The derivation process is exactly same as above, except $y_p(k)$ and $y_q(k)$ are handled respectively in equations (2.9) and (2.10) instead of equation (2.11). Taking the output $y_p(k)$ as the example, we construct the matrices A_p and B_p for $k \in \{0, 1, ..., K\}$ as

$$A_p = col(y_p(1), y_p(2), ..., y_p(K)) \in \mathbb{R}^{K\times 1}, \tag{2.13}$$

$$B_p = \begin{bmatrix} \phi_p(1) & U_p(1) \\ \phi_p(2) & U_p(2) \\ \vdots & \vdots \\ \phi_p(K) & U_p(K) \end{bmatrix}_{K\times(n+m_p+1)} . \tag{2.14}$$

Similarly, we can get the matrixes A_q, B_q for the output $y_q(k)$. Then, the respective estimates will be

$$\beta_p = -\left[B_p^{-1}A_p\right]^{+n}, \beta_q = -\left[B_q^{-1}A_q\right]^{+n}. \tag{2.15}$$

To construct a *global* estimate of β, we simply average these two decentralized (or local) estimates β_p and β_q, namely $\bar{\beta} = \frac{\beta_p+\beta_q}{2}$ as an arithmetic mean, or $\bar{\beta} = \sqrt{\beta_p\,\beta_q}$ as the geometric mean. If there are l PMU measurements, then the average will simply result to $\bar{\beta} = \frac{1}{l}\sum_{i=1}^{l}\beta_l$ for the arithmetic mean, or $\bar{\beta} = (\Pi_{i=1}^{l}\beta_i)^{1/l}$ for the geometric mean.

2.2.2 Recursive Least-Squares (RLS) Algorithm

The standard least-squares algorithms described in the above subsection, however, is more suited to be carried out in an offline mode, meaning that the algorithm is applicable only when a large chunk of PMU data from various locations is available to the operator in a batch [18]. The recursive least-square (RLS) algorithm, on the other hand, is a real-time parameter estimation method, where PMU data from various locations may stream in online to the operator central server (in the centralized case) or to the local PDCs (in the decentralized case). To formulate the RLS problem, equation (2.9) can be rewritten as $y_k = \phi_k^T\theta$ for all k, where $\theta = \begin{bmatrix} \gamma_3 \\ \gamma_1 \end{bmatrix}$ is the unknown parameter vector and $\phi_k^T = \begin{bmatrix} \phi_p(k) & U_p(k) \end{bmatrix}$ is the regressor vector at the sample index k containing the past input and output information. Following the standard least-squares algorithm, the problem in the case, therefore, is to generate θ_K that solves

$$\min_{\theta_K} \sum_{k=0}^{K-1} (y_k - \phi_k^T\theta_K)^2. \tag{2.16}$$

Differentiating (2.16) with respect to θ_K and equating to zero for the first-order necessary condition, we get

$$\theta_K = \left(\sum_{k=0}^{K-1} \phi_k \phi_k^T\right)^{-1} \left(\sum_{k=0}^{K-1} y_k \phi_k\right). \tag{2.17}$$

In the event that $\sum_{k=0}^{K-1} \phi_k \phi_k^T$ is not invertiable, we may define another cost function

$$\min_{\theta_K} \left(\sum_{k=0}^{K-1} (y_k - \phi_k^T \theta_K)^2 + (\theta_K - \theta_0)^T R_0 (\theta_K - \theta_0)\right), \tag{2.18}$$

where, R_0 is a positive-definite matrix, and θ_0 is a best initial guess. With this new cost function, the problem reduces to

$$\theta_K = \left(R_0 + \sum_{k=0}^{K-1} \phi_k)\phi_k^T\right)^{-1} \left(R_0 \theta_0 \sum_{k=0}^{K-1} y_k \phi_k\right). \tag{2.19}$$

To save computational time, the RLS step can next be applied as follows. Let

$$P_K = \left(R_0 + \sum_{k=0}^{K-1} \phi_k \phi_k^T\right)^{-1}, \tag{2.20}$$

we can get

$$P_{K+1} = \left(P_K^{-1} + \phi_K \phi_K^T\right)^{-1}. \tag{2.21}$$

By the matrix inversion leamma of $(A + BC)^{-1} = A^{-1} - A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1}$, we may rewrite Equation (2.21) as,

$$P_{K+1} = P_K - \frac{P_K \phi_K \phi_K^T P_K}{1 + \phi_K^T P_K \phi_K}, \tag{2.22}$$

where $P_0 = R_0^{-1}$. By substituting (2.22) into (2.19), the iteration for the unkown parameter vector θ_K can be written as

$$\theta_{K+1} = \theta_K + P_{K+1} \phi_K (y_K - \phi_K^T \phi_K). \tag{2.23}$$

The stop condition of this recursive algorithm is when $||\theta_{K+1} - \theta_K|| < \epsilon$ where ϵ is a chosen tolerance. The estimated common parameter vector then is $\beta = -[\theta_K]^{+n}$. The decentralized version of this algorithm follows similarly as for the batch case. Each PDC, receiving $y_p(k)$ and $y_q(k)$ respectively in real-time, may estimate decentralized estimates β_p and β_q, transmit these two estimates to another PDC, often referred to as a central server in our simulations, where the coefficients are averaged to form the global estimate $\bar{\beta}$. Using this global estimate the

characteristic polynomial can be easily constructed and the poles can be solved for. For example, considering $\bar{\beta} = \{a_1, a_2, \cdots, a_n\}$, estimates of the actual system poles can then be calculated by solving for the characteristic polynomial

$$1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n} = 0. \tag{2.24}$$

It is worth noting that depending on need it may not be necessary to estimate all poles of the system but only a specific set of poles, for example those corresponding to inter-area oscillations. The primary goal of wide-area monitoring, in face, often involves selective estimation of oscillation modes. If that is the case, least squares may not be the optimal approach as it tends to estimate the entire characteristic polynomial as well as the zeros of the transfer function. Instead input-independent methods such as Prony may be more suitable and time-saving.

2.3 Distributed Prony Algorithm with ADMM

Both standard least-squares and its recursive implementation algorithms described in above section not only require the prior known disturbance input information, but also not easy to be implemented in completely distributed architecture. However, practical disturbance events always feature as unpredicted and varying variable. To address this problem, we introduce the existing well-known modal estimation algorithm - Prony analysis method and its real-time distributed architecture in this section.

2.3.1 Prony Algorithm

Consider a set of N PMU measurements $\mathbf{y}(t) = \text{col}(y_1(t) \cdots y_N(t))$ are available at $t = 0, 1, \ldots, M$, in a given power system network described in Section II. Following the linearized state space model shown in Equation(2.5), one can write the continuous-time transfer function between the input $u(t)$ and output $y_p(t)$ as below [5]:

$$G_p(s) = \frac{Y_p(s)}{U(s)} = \sum_{i=1}^{n} \frac{r_{p,i}}{s - \lambda_i}, \tag{2.25}$$

where λ_i is the ith pair of eigenvalues, corresponding to the ith pair of oscillatory modes of the system, $r_{p,i}$ is the residue or amplitude of ith mode, and n is the total system order. If we apply an impulse as input to the system, the output $y_p(t)$ can be written as

$$y_p(t) = \sum_{i=1}^{n} r_{p,i} e^{(-\sigma_i + j\Omega_i)t} + r_{p,i}^* e^{(-\sigma_i - j\Omega_i)t}, p = 1, \cdots, N, \tag{2.26}$$

It is worth noting that regardless input is an impulse or a step unit, the linearized system response will always be a sum of exponential terms [4]. This is the form that the Prony method can be applied for modal estimation. When $y_p(t)$ is sampled at a constant sampling period, Δt, we have the following discrete form as below:

$$y_p(k) = \sum_{i=1}^{n} r_{p,i} z_i^k, \tag{2.27}$$

where $z_i = e^{(-\sigma_i \pm j\Omega_i)\Delta t}$. The modal estimation objective is to find the damping factors σ_i, the frequencies Ω_i and residues $\mathbf{r}_i = col(r_{1,i} \cdots r_{p,i} \cdots r_{N,i})$ for $i = 1, \ldots, n$. Such that, the basic idea is to fit a sum of exponential terms to the measured data. We then consider Prony method as it formulates the problem as a least squares minimization problem, and further, can be cast as a distributed optimization problem [6].

Let us consider the z-transform of the ith PMU measurement set $y_i(t)$, as

$$Y_i(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}. \tag{2.28}$$

Step 1. The first step of Prony is to find b_1 through b_n by solving

$$\underbrace{\begin{bmatrix} y_i(n) \\ y_i(n+1) \\ \vdots \\ y_i(n+\ell) \end{bmatrix}}_{\mathbf{c}_i} = \underbrace{\begin{bmatrix} y_i(n-1) & \cdots & y_i(0) \\ y_i(n) & \cdots & y_i(1) \\ \vdots & & \vdots \\ y_i(n+\ell-1) & \cdots & y_i(\ell) \end{bmatrix}}_{\mathbf{H}_i} \underbrace{\begin{bmatrix} -a_1 \\ -a_2 \\ \vdots \\ -a_n \end{bmatrix}}_{\mathbf{a}} \tag{2.29}$$

where ℓ is an integer satisfying $n + \ell \leq M - 1$, where $M - 1$ is the time index of the most recent measurement. One can find $\mathbf{a}$ by solving a least squares (LS) problem defined as

$$\min_{\mathbf{a}} \frac{1}{2} \|\mathbf{H}_i \mathbf{a} - \mathbf{c}_i\|^2. \tag{2.30}$$

Step 2. We next find the roots of the discrete-time characteristic polynomial equation as shown in denominator of (2.28) denoted by z_i, $i = 1, \ldots, n$. Then, the eigenvalues of A in (2.5) are equal to $\ln(z_i)/T$, T being the sampling period.

Step 3. The final step is to find the residues $\mathbf{r}_i$ in (2.26). This can be done by forming the

following so-called *Vandermonde* equation and solving it for $\mathbf{r}_1$ through $\mathbf{r}_n$.

$$\begin{bmatrix} y_i(0) \\ y_i(1) \\ \vdots \\ y_i(M) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ (z_1)^{1/T} & (z_2)^{1/T} & \cdots & (z_n)^{1/T} \\ \vdots & \vdots & & \vdots \\ (z_1)^{M/T} & (z_2)^{M/T} & \cdots & (z_n)^{M/T} \end{bmatrix} \begin{bmatrix} r_1 \\ r_1^* \\ \vdots \\ r_n \\ r_n^* \end{bmatrix}. \tag{2.31}$$

Note 1 The method can be easily generalized to the case of multiple output measurements $\mathbf{y}(t) = \mathrm{col}(y_1(t) \cdots y_N(t))$. Let us concatenate $\mathbf{c}_i$ and $\mathbf{H}_i$ in (2.29) for $i = 1, \ldots, p$. Then, one can solve the following LS problem for $\mathbf{a}$.

$$\min_{\mathbf{a}} \frac{1}{2} \| \begin{bmatrix} \mathbf{H}_1 \\ \vdots \\ \mathbf{H}_p \end{bmatrix} \mathbf{a} - \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_p \end{bmatrix} \|^2. \tag{2.32}$$

where $\|\cdot\|$ denotes the 2-norm of a vector.

Step 3 then will be applied for each $y_i(t)$ individually.

Note 2 The method can be performed in real-time. That is, one may solve (2.30) or (2.32) recursively, while instantly updating $\mathbf{H}_i$, and $\mathbf{c}_i$ as new PMU measurements become available [7].

2.3.2 Real-Time Distributed Prony Algorithm using ADMM

In this subsection, we show how the centralized approach delineated in above subsection for estimating the oscillation modes from PMU data can be recast as a distributed optimization problem. The LS problem (2.32) is, in fact, a *global consensus problem* over a network of N regional utility companies, and, as shown in the following, can be estimated via distributed protocols with one central independent system operator (ISO) performing a *supervisory* step to guarantee convergence. Before we describe the mathematical formulation of the algorithm, we wish to briefly describe the way this strategy may be actually implemented by an ISO. We assume that each operating zone of the power system is equipped with its own regional PMUs, and phasor data concentrator (PDC) that runs the estimation algorithm locally. In reality, there may be a cluster of PDCs or a virtual machine of cloud computing running in each area, but for convenience of analysis we only consider an aggregate regional PDC to be responsible for accepting its area-level PMU measurements, run the estimation using these measurements, and then share a set of estimated transfer function parameters with PDCs of other areas, albeit through one step of supervision through the central ISO-level PDC. This problem in general is

described as,

$$\begin{aligned} \underset{\mathbf{a}_1,\ldots,\mathbf{a}_N,\mathbf{z}}{\text{minimize}} \quad & \sum_{i=1}^{N} \frac{1}{2}\|\hat{\mathbf{H}}_i\mathbf{a}_i - \hat{\mathbf{c}}_i\|^2 \\ \text{subject to } & \mathbf{a}_i - \mathbf{z} = 0, \text{ for } i = 1,\ldots,N, \end{aligned} \tag{2.33}$$

where, $\hat{\mathbf{H}}_i =$ col$(\mathbf{H}_k)$ $\forall k \in$ Area i, $\hat{\mathbf{c}}_i =$ col$(\mathbf{c}_k)$ $\forall k \in$Area i, and the global consensus solution, denoted by $\mathbf{z}$, is the solution of (2.32) that is obtained when the local estimates of the entire N regional PDCs, denoted by $\mathbf{a}_i$, $\forall\ i = 1,\ldots,N$, reaches the same value. We next use one important class of solution approaches for (2.33), namely, alternating direction method of multipliers (ADMM), as follows.

The ADMM distributed estimation method uses the Lagrangian multiplier approach to solve (2.33). The *augmented Lagrangian* for (2.33) is defined as

$$L_\rho = \sum_{i=1}^{N}(\frac{1}{2}\|\hat{\mathbf{H}}_i\mathbf{a}_i - \hat{\mathbf{c}}_i\|^2 + \mathbf{w}_i^T(\mathbf{a}_i - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{a}_i - \mathbf{z}\|^2),$$

where $\mathbf{w}_i$ is the vector of the *dual variables*, or the Lagrange multipliers associated with (2.33), and $\rho > 0$ denotes the *penalty factor*. Using this, the resulting ADMM problem, assuming $\mathbf{z} = \bar{\mathbf{a}} \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbf{a}_i$,can be defined by the following set of recursive optimization problems [33] to solve (2.33) in a distributed way:

$$\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k-1)} + \rho(\mathbf{a}_i^{(k)} - \mathbf{z}^{(k)}), \tag{2.34a}$$

$$\mathbf{a}_i^{(k+1)} = ((\hat{\mathbf{H}}_i^{(k)})^T\hat{\mathbf{H}}_i^{(k)} + \rho I)^{-1}((\hat{\mathbf{H}}_i^{(k)})^T\hat{\mathbf{c}}_i^{(k)} - \mathbf{w}_i^{(k)} + \rho\mathbf{z}^{(k)}), \tag{2.34b}$$

$$\mathbf{z}^{(k+1)} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{a}_i^{(k+1)}. \tag{2.34c}$$

Each iteration of (2.34) consists of the following steps: 1) update the local variable $\mathbf{a}_i$ locally at PDC i (2.34b); 2) gather the values $\mathbf{a}_i$ at a central ISO-level PDC, and calculate their mean $\mathbf{z}$ for the synchronous ADMM (2.34c); 3) broadcast $\mathbf{z}$ to the other local PDCs; and 4) finally, update $\mathbf{w}_i$ at each PDC i (2.34a). It can be shown that $\mathbf{z}$ as $k \to \infty$ converges to the global minimum of (2.33) [33]. The rate of convergence also depends on the choice of ρ.

Chapter 3

Mode Estimation with Asynchronous Communication

In Chapter 2, we proposed a distributed PMU-PDC architecture for estimating power system oscillation modes by integrating a Prony algorithm with Alternating Direction Method of Multipliers (ADMM) [27]. A critical assumption behind the proposed method was that the communication between local PDCs and the central averaged is completely synchronized. In realistic wide-area networks, however, such synchronous communication may not always be possible. In this chapter we address this issue of asynchronous communication, and its impact on the convergence of the distributed estimation. We first pose the estimation problem as a real-time, iterative, distributed consensus problem. Thereafter, we consider a probabilistic traffic model for modeling delays in any typical wide-area communication network, and study how the delays enter the process of information exchange between distributed Phasor Data Concentrators (PDC) that are employed to execute this consensus algorithm in a coordinated fashion.

To investigate the ADMM convergence on delay distribution parameters, we implement two strategies of averaging at the central PDC based on a chosen delay threshold. We carry out simulations to show possible instabilities and sensitivities of the ADMM convergence on delay distribution parameters under these two averaging strategies. However, preliminary results on two of the proposed algorithms reported in our conference paper [26], consider only uni-directional delays. Next, we propose a suite of four different asynchronous distributed optimization algorithms for wide-area oscillation estimation in power systems using Alternating Direction Method of Multipliers (ADMM) to extend those initial findings to two new strategies, each with multiple sub-scenarios, with considering the combined impact of both uplink and downlink delays in asynchronous ADMM. By the proposed four different strategies, the convergence rate and accuracy of this consensus algorithm can be made immune to the asynchrony resulting from the network traffic. We also carry out extensive simulations to show possible

numerical instabilities and sensitivities of the ADMM convergence on our proposed strategies. Finally, our results exhibit a broad view of how the convergence of any distributed estimation algorithms in a generic cyber-physical system depends strongly on the uncertainties of the underlying communication models [25, 26].

3.1 Delay Model for Wide-Area Communication

The main purpose of this chapter is to study the convergence analysis of ADMM-based power system mode estimation under asynchronous wide-area communication delays. For this purpose, we first develop a probability distribution model for the network delays. Following [28], we model the stochastic end-to-end delay experienced by a message between the central PDC and local PDCs in terms of three components: the minimum deterministic delay, denoted by m; the Internet traffic delay with Probability Density Function (PDF), denoted by ϕ_1; and the router processing delay with PDF, denoted by ϕ_2. Then, the PDF of the total delay at any time t is given as

$$\phi(t) = p\phi_2(t) + (1-p)\phi_1(t) * \phi_2(t), \quad t \geq 0, \tag{3.1}$$

with $\phi_1(t) * \phi_2(t) = \int_0^t \phi_2(u)\phi_1(t-u)du$. Here p is the probability of open period of the path with no Internet traffic, and the router processing delay can be well approximated by a Gaussian density function $\phi_2(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$, where $\mu > m$. The Internet traffic delay is modeled by an alternating renewal process with exponential closure period when the Internet traffic is on, with the PDF $\phi_1(t) = \lambda e^{-\lambda t}$, where λ^{-1} models the mean length of the closure period. The benchmark value of all parameters of this model are set as: $p = 0.58, \lambda = 1.39, \mu = 5.3, \sigma = 0.078$, following [28].

We next derive the Cumulative Distribution Function (CDF) of the delay model. First, (3.1) can be rewritten as

$$\phi(t) = \frac{p}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} + \frac{\lambda(1-p)}{\sigma\sqrt{2\pi}} e^{-\lambda t} \int_0^t e^{\lambda s - \frac{(s-\mu)^2}{2\sigma^2}} ds. \tag{3.2}$$

We rewrite the integral part of (3.2) by using the error function $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$, to get

$$\begin{aligned} \phi(t) = & \frac{p}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} \\ & + \frac{\lambda(1-p)}{2} e^{(\frac{1}{2}\lambda^2\sigma^2 + \mu\lambda)} e^{-\lambda t} \mathrm{erf}(\frac{t - \lambda\sigma^2 - \mu}{\sqrt{2}\sigma}) \\ & + \frac{\lambda(1-p)}{2} e^{(\frac{1}{2}\lambda^2\sigma^2 + \mu\lambda)} \mathrm{erf}(\frac{\lambda\sigma^2 + \mu}{\sqrt{2}\sigma}) e^{-\lambda t}. \end{aligned} \tag{3.3}$$

By using the partial integral method and the first derivative of the error function, $\frac{d}{ds}\mathrm{erf}(s) =$

$\frac{2}{\sqrt{\pi}}e^{-s^2}$, we derive the CDF of the delay model as

$$P(t) = \int_{-\infty}^{t} \phi(s)ds = \frac{1}{2}[\text{erf}(\frac{\mu}{\sqrt{2}\sigma}) + \text{erf}(\frac{t-\mu}{\sqrt{2}\sigma})] + \frac{(p-1)}{2}e^{(\frac{1}{2}\lambda^2\sigma^2+\mu\lambda)}e^{-\lambda t}[\text{erf}(\frac{\lambda\sigma^2+\mu}{\sqrt{2}\sigma}) + \text{erf}(\frac{t-\lambda\sigma^2-\mu}{\sqrt{2}\sigma})]. \tag{3.4}$$

Random delays from this CDF will next be imposed on the communication links to emulate the ADMM-based power system mode estimation under asynchronous wide-area communication delays.

3.2 Convergence Analysis on Delay Distribution Parameters

3.2.1 Distributed Prony with Standard ADMM (S-ADMM)

Based on the description of the distributed Prony algorithm using ADMM in Chapter 2, we notice that $\mathbf{a}_i,\ i = 1, \ldots, N$ updates can be performed by all local PDCs in parallel. However, they have to be synchronized in that the central PDC has to wait for them from all local PDCs. In this sequal, this distributed consensus ADMM algorithm is referred as synchronous ADMM or standard ADMM (S-ADMM). The update procedures for both central PDC and local PDCs are shown in **Algorithm 1**, respectively.

3.2.2 Distributed Prony with Asynchronous ADMM (A-ADMM)

From the communication perspective, the distributed Prony with S-ADMM algorithm has three delay-sensitive steps: **Step 1)** each local PDC streams PMU measurements in real-time from multiple local PMUs. Since this communication happens in a private network, we ignore this communication delay. **Step 2)** the central PDC receives all local estimates within a certain time period, referring to as delay threshold and denoted by d^*, due to the uplink delays arising in the communication network. **Step 3)** the central PDC broadcasts the averaged estimation to each local PDC to update the Lagrange multiplier for the next iteration, which also involves the downlink communication delay.

We are interested in the reaction of the central PDC at the kth iteration when some $\mathbf{a}_i^{(k)}$ cannot be received within time d^*. We test the numerical stability and convergence of (2.34) under two different averaging strategies depending on the choice of d^* as follows.

Strategy I

In this strategy the central PDC skips all delayed local estimations at the kth iteration. In other words, if any of the local estimates $\mathbf{a}_i^{(k)}$ does not arrive at the central PDC within a time

Algorithm 1 Standard-ADMM

```
procedure CentralPDC(ϵ)
    initialize: k = 0
    repeat
        repeat
            wait
        until receive updates a_i^(k) from all local PDCs
        update z^(k) by (2.34c)
        send z^(k) to all local PDCs
        k ← k + 1
    until Δz = ||z^(k) − z^(k−1)|| ≤ ϵ
end procedure
procedure LocalPDC i
    initialize: k = 0, w_i^(0) = 0
    repeat
        update y_i(k), Ĥ_i^(k), ĉ_i^(k) by (2.29)
        update a_i^(k) by (2.34b)
        send a_i^(k) to the central PDC
        repeat
            wait
        until receive z_i^(k) from the central PDC
        update w_i^(k+1) by (2.34a)
        k ← k + 1
    until termination
end procedure
```

threshold d^* starting from the time instant when the central PDC broadcasts $\bar{\mathbf{a}}^{(k-1)}$ back, then the latter will ignore these estimates, and calculate the average based on only the non-delayed local updates as

$$\mathbf{z}^{(k)} = \frac{1}{l}\sum_{i=1}^{l}\mathbf{a}_i^{(k)}, \quad l \leq N \tag{3.5}$$

where l is the number of non-delayed local estimations.

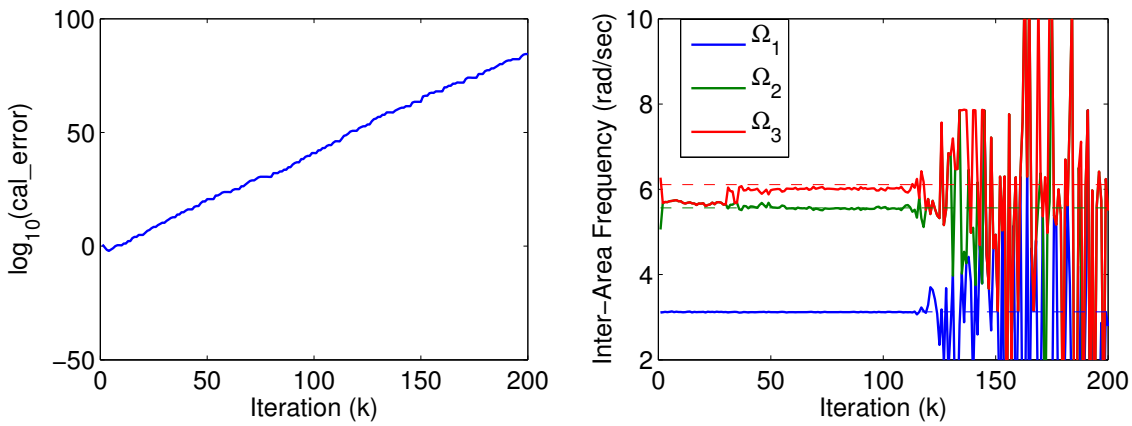


Figure 3.1: Convergence Performance of Strategy I

Simulation results shown in Fig. 3.1 indicate that this skipping strategy will cause the algorithm to diverge.

Strategy II

In this alternative strategy the central PDC uses the previous update $\mathbf{a}_i^{(k-1)}$ for all delayed local updates, assuming that $\mathbf{a}_i^{(k-1)}$ is stored at the central PDC. Then, the average is calculated as

$$\mathbf{z}^{(k)} = \frac{1}{N}(\sum_{i=1}^{l}\mathbf{a}_i^{(k)} + \sum_{i=l+1}^{N}\mathbf{a}_i^{(k-1)}), \tag{3.6}$$

where l is the number of non-delayed local updates. From this point onwards, the ADMM algorithm with the averaging step as in (3.6) will be referred to as Asynchronous ADMM (A-ADMM).

The stop condition, $\Delta\mathbf{z} = ||\mathbf{z}^{(k)} - \mathbf{z}^{(k-1)}||$ in step 10 of Algorithm 1 of S-ADMM also needs to be modified for the A-ADMM. The reason is as follows: in case every $\mathbf{a}_i^{(k)}$ arrives after time

d^*, all previous updates have to be used, $\Delta\mathbf{z}$ will then be zero, and the algorithm terminates unexpectedly. To solve this problem, we propose the following adjustments in the stop condition and the convergence tolerance. We first modify the stop condition to

$$\Delta\mathbf{z} = \frac{1}{l}\sum_{i=1}^{N} ||\mathbf{a}_i^{(k)} - \mathbf{a}_i^{(k-1)}|| \tag{3.7}$$

The convergence tolerance denoted by ϵ, is set to be 10^{-6} as usual. This adjustment is referred to as Adj. I. Another way would be to adjust ϵ adaptively according to the number of the non-delayed local updates, which is referred to as Adj. II. This adjustment follows a rule that a lower ϵ can be chosen for a smaller number of l. That is, we define a vector $\epsilon = [\epsilon_1, \ldots, \epsilon_N]$ such that for $l = 1$ we use ϵ_1, for $l = 2$, we use ϵ_2 and so on, where $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_N$. The modified update procedure for the central PDC in the A-ADMM algorithm is shown in **Algorithm 2**. The regression steps for the local PDC remains the same as **Algorithm 1**.

Algorithm 2 Asynchronous-ADMM

```
procedure CentralPDC(ϵ)
    initialize: k = 0, d*
    repeat
        repeat
            wait
            receive updates a_i^(k) from local PDC_i
        until timer ≤ d* or all updates received
        update z^(k) by (3.6), update Δz by (3.12)
        send z^(k) to all local PDCs
        k ← k + 1
    until Δz ≤ ϵ
end procedure
```

In the following subsection, we use (3.4) to generate a set of random delays and test the convergence of Strategy II for a given delay threshold d^* under different delay model parameters.

3.2.3 Numerical Verification

We implement the distributed Prony with S-ADMM algorithm as the benchmark, and the distributed Prony with A-ADMM combining the aforesaid delay model in Matlab to verify their convergence performance. We consider the IEEE 39-bus model divided into four area with four measurements from buses 32, 33, 37, and 39, and the sampling rate of PMU data being

0.01 seconds. For details of S-ADMM applied to this model please see [27]. Table 3.1 shows the convergence iteration number k^* for different ϵ for S-ADMM. The data verifies that as ϵ becomes tighter, the algorithm becomes slower.

Table 3.1: Benchmark of Convergence Iteration of Prony with S-ADMM

ϵ	10^{-8}	0.5×10^{-7}	10^{-7}	0.5×10^{-6}	10^{-6}	0.5×10^{-5}	10^{-5}	0.5×10^{-4}	10^{-4}	0.5×10^{-3}	10^{-3}
k^*	626	82	78	38	37	33	30	21	12	5	5

Convergence of A-ADMM via Algorithm 2

To verify the convergence performance of two adjustments of the A-ADMM algorithm, described in Section 3.2.2, we set the tolerance as $\epsilon = [10^{-8}, 10^{-7}, 0.5 \times 10^{-6}, 10^{-6}]$. Table 3.2 shows the values of k^* for the two adjustments. This table shows that Adj. I has better convergence performance than Adj. II. Thus, from now onwards, we implement the A-ADMM using Adj. I.

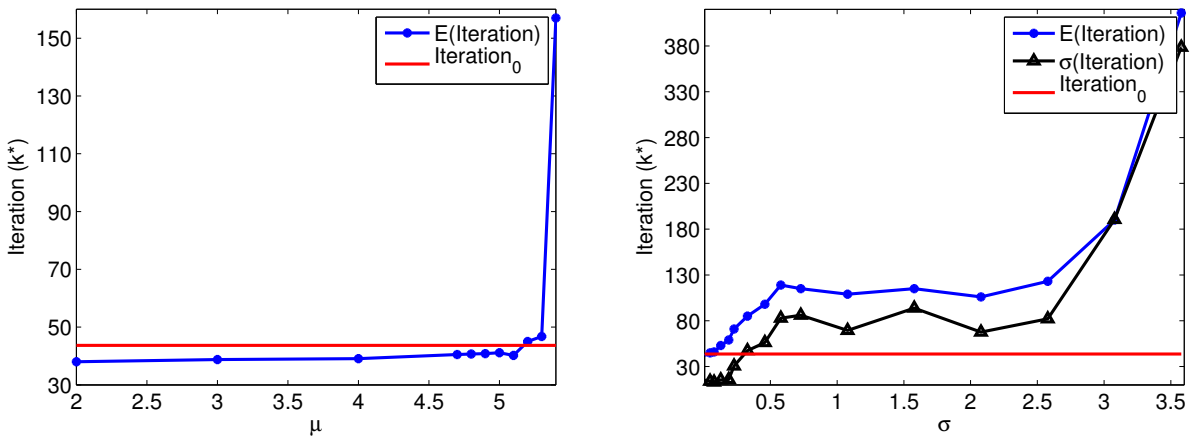
Table 3.2: Convergence Rate of Two Proposed Adjustments

$\tau = P(X \leq d^*)$	1	0.9	0.8	0.7	0.6	0.5	0.4
d^* (ms)	6.8	6	5.7	5.5	5.4	5.35	5.3
k^* (Adj. I)	37	41	42	41	43	84	330
k^* (Adj. II)	37	43	45	43	52	94	628

Delay Model Parameters vs Convergence Rate

The impact of five delay model parameters $(\mu, \sigma, \lambda, p, \tau)$ are investigated on the convergence performance of A-ADMM with Adj. I. The results are shown in Figs. 3.2, 3.3, and 3.4, in which the red line indicates the benchmark value for $k^* = 43$. The delay threshold is set to be $d^* = 5.458$ ms.

• *Effect of deterministic delay* μ: Fig. 3.2a shows the following observations: 1) when $\mu < 2$ ms, the value of k^* reaches the S-ADMM benchmark value of 38; 2) as μ increases from 2 ms to the benchmark value of 5.3 ms, k^* increases monotonically with a slow step; 3) while μ changes from 5.4 ms to 5.5 ms, i.e., it is approaching to d^*, k^* increases dramatically; 4) once $\mu > 5.5$ ms, which implies that all random generated delays will be greater than d^*, and the algorithm never converges, as we expected. Observation 2 also implies that there exists a lower bound of μ theoretically.



(a) (b)

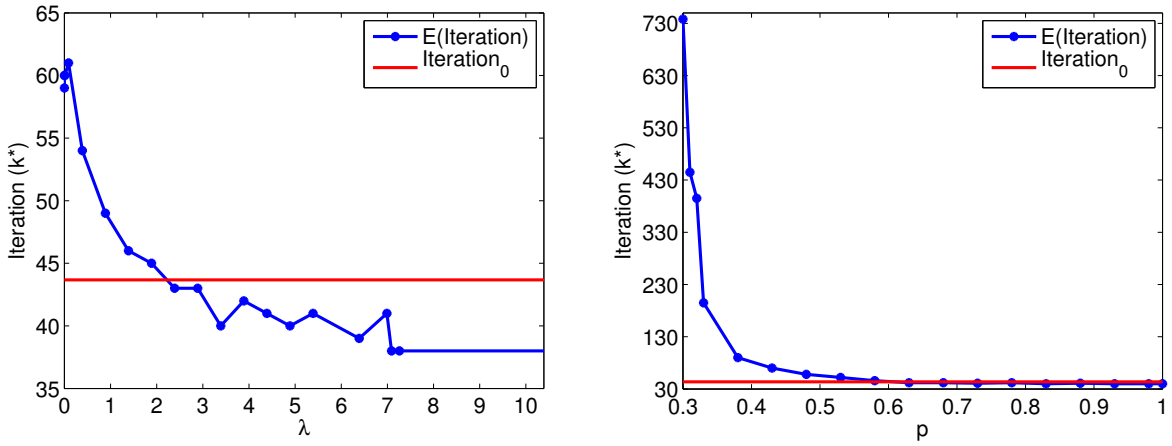
Figure 3.2: μ, σ vs convergence rate

• *Effect of standard deviation* σ: Three interesting conclusions can be drawn from Fig. 3.2b: 1) both expectation and standard deviation of k^* increases when σ varies in the range $[0.008, 3.5]$; 2) k^* rises with a constant slope as σ increases from 0.008 to 0.578, while it remains almost constant around the value of 105 as $\sigma \in [0.578, 2.5]$. As σ increases from 2.5 to 3.5, k^* goes up dramatically. 3) when σ approaches to 0, the expectation value of k^* gets close to 42 and the standard deviation of k^* gets close to 14. If $\sigma > 3.5$, the algorithm diverges; in other words, large variance of communication delays will cause the algorithm to diverge.

• *Effect of the length of closure period* λ^{-1}: Theoretically, smaller λ indicates longer closure period, at which the Internet traffic will cause longer communication delay. Thus, it is expected that the smaller λ results in larger k^*. As shown in Fig. 3.3a, when the λ increases from 0 to the benchmark value of 1.39, the convergence performance drops. While $\lambda > 1.39$, the convergence performance is more stable, until λ reaches to 7, when it approximates the S-ADMM benchmark value of 38.

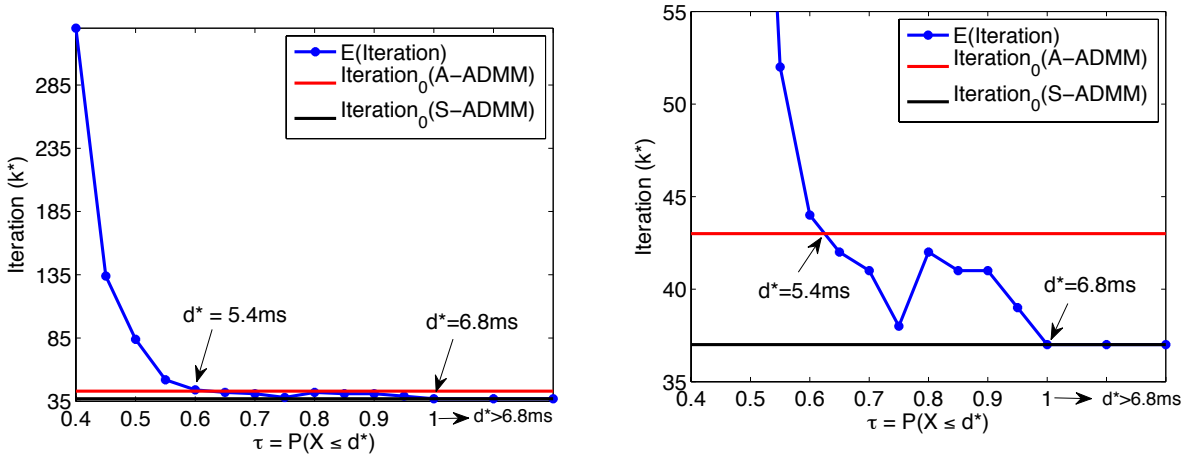
• *Effect of probability of open period* p: Note that when parameter p increases from 0.3 to 1, the convergence iteration drops as shown in Fig. 3.3b. The reason is that as p increases, the Internet traffic decreases, which indicates the smaller communication delays; thus, we can achieve a better convergence. The fact that for $p < 0.3$ the algorithm tends to diverge indicates that there exists a lower-bound of p for convergence.

• *Effect of* τ*, the CDF of delay threshold* d^*: Similar to p, when the delay threshold d^* is set to a large value, equivalently τ is larger. When $\tau < 0.4$, the algorithm also starts to diverge. This observation implies the existence of lower-bound of τ or d^*. Fig. 3.4 zooms in and shows that the convergence curve will be the expected benchmark value when $d^* \geq 6.8$ ms.



(a) (b)

Figure 3.3: λ, p vs convergence rate



(a) (b)

Figure 3.4: τ vs convergence rate

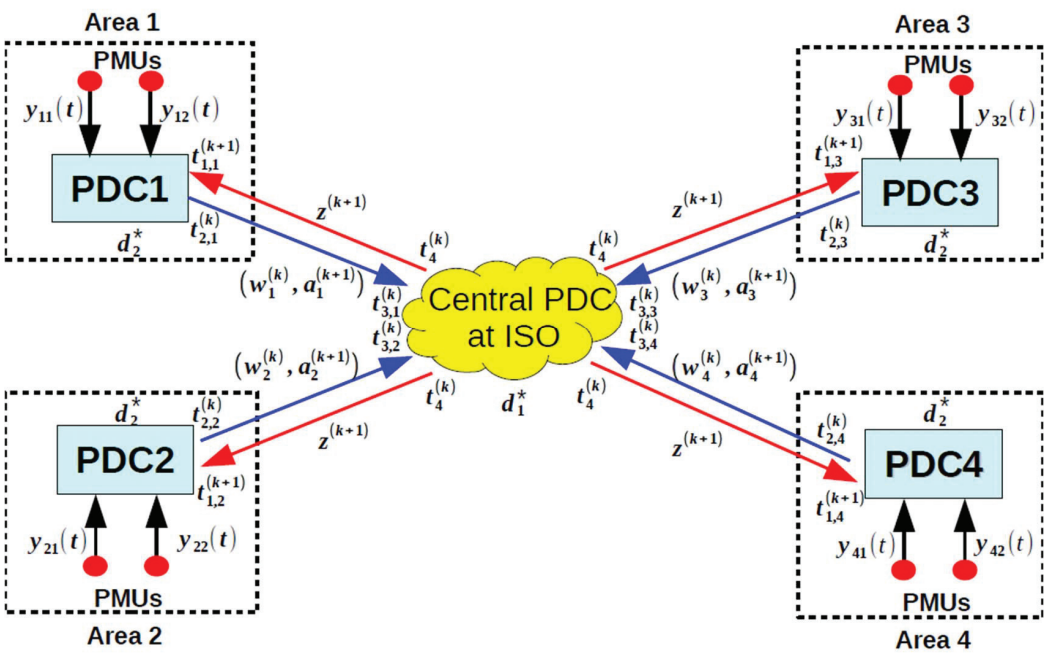


Figure 3.5: Distributed architecture for a 4-area power system network.

3.3 Convergence Analysis on A-ADMM Strategies

Preliminary results on two of the proposed algorithms in Section 3.2 only consider 1) the simple average of local estimations at the central PDC; 2) the uplink communication delay from local PDCs to the central PDC. This section extends those initial findings to 1) new average of local estimations and dual variables [24], as

$$\mathbf{z}^{(k+1)} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{a}_i^{(k+1)} + (1/\rho)\mathbf{w}_i^{(k)}), \tag{3.8}$$

2) two new strategies, each with multiple sub-scenarioes, and considers the combined impact of both uplink and downlink delays in asynchronous ADMM. We first review the A-ADMM algorithm and the asynchronous communications under these two considerations.

3.3.1 Asynchronous Communications in A-ADMM

With the new average approach, we re-summarize the distributed PMU-PDC architecture as follows. Consider the kth iteration. Referring to Fig. 3.5: **Step 1)** at time $t_{1,i}^{(k)}$, any local PDC i runs the dual-primal update to gain ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) using (2.34a), and (2.34b) after receiving the consensus variable $\mathbf{z}^{(k)}$ from the central PDC; **Step 2)** at time $t_{2,i}^{(k)}$, the local PDC i transmits

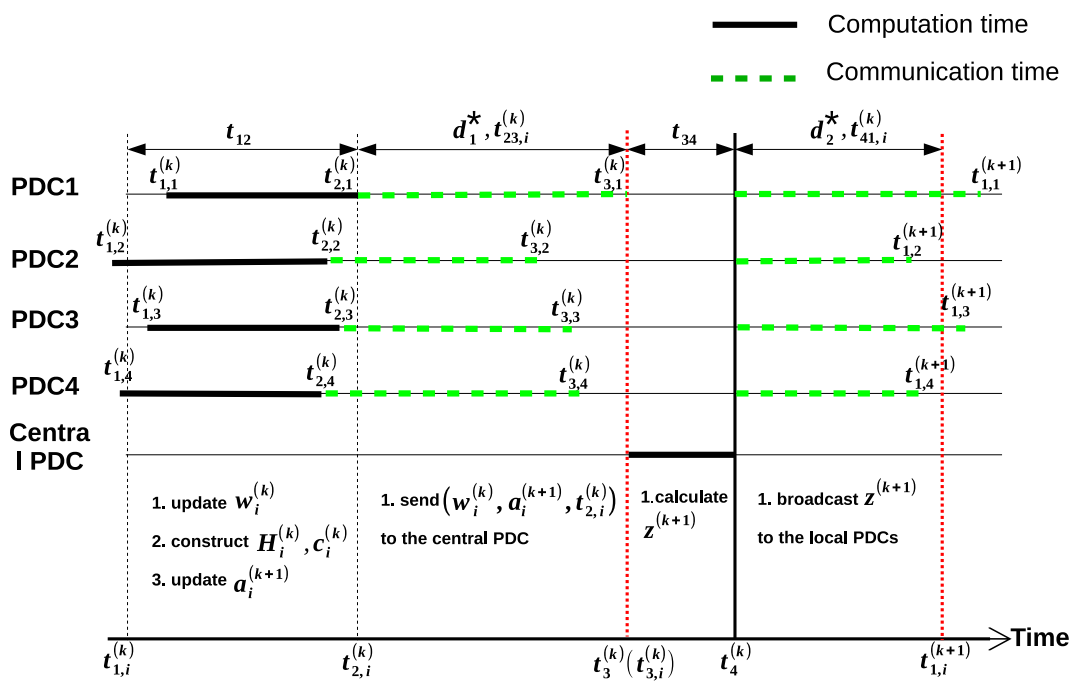


Figure 3.6: Timing diagram for Asynchronous ADMM

updated dual-primal variable set ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) to a central PDC; **Step 3)** at time $t_{3,i}^{(k)}$, the central PDC calculates the consensus variable $\mathbf{z}^{(k+1)}$ using (3.8); **Step 4)** at time $t_{4,i}^{(k)}$, the central PDC broadcasts $\mathbf{z}^{(k+1)}$ to the local PDCs in each area for their next update. One critical assumption behind this architecture is that all local PDCs are performing their respective optimization steps with equal speed, i.e., they are synchronous.

In practice, however, the communication between the local PDCs and the central PDC will always involve communication delays, thereby leading to asynchrony in message arrivals at all PDCs. The timing diagram under that condition, as shown in Fig. 3.6, will consist of three main delay-sensitive components: (1) local PMU measurements stream in real-time to the local PDCs. Since this communication happens over a private network, we ignore this communication delay throughout this paper; (2) the green dash lines between time $t_{2,i}^{(k)}$ and time $t_{3,i}^{(k)}$ at any iteration k show that the local estimates ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) arrive at the central PDC at different instants $t_{3,i}^{(k)}, i = 1, \ldots, N$ due to variable uplink delays; (3) the green dash lines between time $t_4^{(k)}$ and time $t_{1,i}^{(k+1)}$ show that the consensus variable $\mathbf{z}^{(k)}$ arrives at different instants $t_{1,i}^{(k+1)}, i = 1, \ldots, N$ at different local PDCs due to variable downlink delays. The resulting algorithm is referred to as asynchronous-ADMM or A-ADMM [29]. One trivial way to counteract the asynchrony would be to force all PDCs to wait till they receive every scheduled message at every iteration. This, however, can lead to unacceptably slow convergence times, and, depending on network

congestion, can even turn out to be very risky in case any message gets lost or delayed for an uncertain period of time. Instead, we wish to counteract asynchrony by defining a set of *flexible deadlines* for message arrival in every PDC, and accordingly by modifying the update rules in (2.34,3.8) based on only those messages that respect these deadlines. In order to understand how these deadlines should be constructed in accordance to the network traffic, we first develop a probability distribution model for the network delays.

3.3.2 Proposed A-ADMM Strategies

Strategy I

In this strategy PDCs *skip* messages that do not arrive within a chosen deadline. We define two deadlines or *delay thresholds*, namely $\mathbf{d}_1^* > 0$ and $\mathbf{d}_2^* > 0$ in millseconds, for the uplink and downlink delays, respectively. Without any loss of generality, as assume that the counting of these deadlines start from the instant at which any PDC *sends out* any message at any iteration. For simplicity of notations, we also assume that every local PDC is assigned the same threshhold $\mathbf{d}_2^*$ although same exact analyses will hold for different threshhold values at different PDCs. Following the timing diagram in Fig. 3.6, if any local update ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) does not arrive at the central PDC within time $\mathbf{d}_1^*$, the central PDC skips them, and computes the consensus variable $\mathbf{z}^{(k+1)}$ in (3.8) simply as,

$$\mathbf{z}^{(k+1)} = \frac{1}{|S_1^{(k)}|} \sum_{i \in S_1^{(k)}} (\mathbf{a}_i^{(k+1)} + (1/\rho)\mathbf{w}_i^{(k)}), \tag{3.9}$$

where $S_1^{(k)}$ is the index set of local PDCs whose messages arrive on time, and $S_2^{(k)}$ is the index set for PDCs that can receive $\mathbf{z}^{(k+1)}$ within time, at the kth iteration.

On the other hand, if any local PDC i does not receive $\mathbf{z}^{(k)}$ from the central PDC within the delay threshold $\mathbf{d}_2^*$, it skips the update (2.34a)-(2.34b) altogether, and simply returns the local estimate ($\mathbf{w}_i^{(k-1)}$, $\mathbf{a}_i^{(k)}$) (i.e., the estimate from the previous iteration) to the central PDC. For a practical Internet model, the skipping strategy is most suitable for scenarios that involve exceptionally long delays or possible packet loss. The algorithm for this strategy is listed in **Algorithm 3**.

Strategy II

In this strategy PDCs use internal memory to replace messages that do not arrive within the respective deadlines with their values from previous iteration. Memorized data from PDCs, for example, can be stored and retrieved via a distributed storage service [30]. Following the timing diagram in Fig. 3.6, if any local update ($\mathbf{w}_i^{(k)}$, $\mathbf{a}_i^{(k+1)}$) does not arrive at the central PDC within

Algorithm 3 A-ADMM with Strategy I

```
procedure CentralPDC(ϵ)
    initialize: k = 1, d_1^*
    repeat
        repeat
            wait
            receive updates (w_i^(k), a_i^(k+1))
        until timer ≤ d_1^* or all updates received
        update z^(k+1) by (3.9)
        broadcast z^(k+1) to all local PDCs
        k ← k + 1
    until Δz = ||z^(k+1) − z^(k)|| ≤ ϵ
end procedure
procedure LocalPDC i
    initialize: k = 1, w_i^(0) = 0, a_i^(0) = 1, d_2^*;
    repeat
        repeat
            wait
        until timer ≤ d_2^*, or receive z^(k)
        update y_i(k), Ĥ_i^(k), ĉ_i^(k) by (2.29)
        if global update received then
            update (w_i^(k), a_i^(k+1)) by (2.34a), (2.34b)
        else
            w_i^(k) = w_i^(k−1); a_i^(k+1) = a_i^(k)
        end if
        send (w_i^(k), a_i^(k+1)) to the central PDC
        k ← k + 1
    until termination
end procedure
```

time $\mathbf{d}_1^*$, the central PDC computes the consensus variable $\mathbf{z}^{(k+1)}$ in (3.8) as

$$\mathbf{z}^{(k+1)} = \frac{1}{N}\Big(\sum_{i\in S_1^{(k)}}(\mathbf{a}_i^{(k+1)} + \frac{\mathbf{w}_i^{(k)}}{\rho}) + \sum_{i\notin S_1^{(k)}}(\mathbf{a}_i^{(l_i+1)} + \frac{\mathbf{w}_i^{(l_i)}}{\rho})\Big), \tag{3.10}$$

where $l_i \in \{k-1, k-2, k-3, \cdots\}$ denotes the index of the *latest* message that arrived successfully at the central PDC from the ith local PDC. Similarly, if any local PDC $i, i \notin S_2^{(k)}$ does not receive $\mathbf{z}^{(k)}$ within its deadline, then it updates (2.34a)-(2.34b) as

$$\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k-1)} + \rho(\mathbf{a}_i^{(k)} - \mathbf{z}_i^{(l_i)}), \tag{3.11a}$$

$$\mathbf{a}_i^{(k+1)} = ((\hat{\mathbf{H}}_i^{(k)})^T\hat{\mathbf{H}}_i^{(k)} + \rho I)^{-1}((\hat{\mathbf{H}}_i^{(k)})^T\hat{\mathbf{c}}_i^{(k)} - \mathbf{w}_i^{(k)} + \rho\mathbf{z}_i^{(l_i)}), \tag{3.11b}$$

where $l_i \in \{k-1, k-2, k-3, \cdots\}$ denotes the index of the *latest* message that arrived successfully at the ith local PDC. Note that $l_i \neq l_j$, in general. However, since every PMU data is time-stamped by GPS, the PDCs will have the ability to stamp or decipher the iteration number corresponding to any message they send or receive.

Adjustment in Stopping Criterion: The usual practice in S-ADMM is to keep track of $\Delta\mathbf{z} = ||\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}||$, and terminate the algorithm once $\Delta\mathbf{z}$ falls below a chosen tolerance. This step needs to be modified for A-ADMM with Strategy II. The reason is as follows. In case every local update $(\mathbf{w}_i^{(k)}, \mathbf{a}_i^{(k+1)})$, $i = 1, \ldots, N$ arrive after time $\mathbf{d}_1^*$ at the local PDC, then the newest update of $\mathbf{z}^{(k+1)}$ will be computed from its stored latest update, which is exactly same as $\mathbf{z}^{(k)}$ causing $\Delta\mathbf{z}$ to be zero. This may force the algorithm to terminate unexpectedly. To alleviate impact of delayed updates, we first modify the stopping criteria to

$$\Delta\mathbf{z} = \frac{1}{|S_1^{(k)}|}\sum_{i\in S_1^{(k)}} ||(\mathbf{a}_i^{(k+1)} - \mathbf{a}_i^{(l_i)}) + (1/\rho)(\mathbf{w}_i^{(k)} - \mathbf{w}_i^{(l_i-1)})||, \tag{3.12}$$

where $(\mathbf{w}_i^{(l_i-1)}, \mathbf{a}_i^{(l_i)})$ is the latest local update received from the ith local PDC at the $(k-1)^{th}$ iteration at the central PDC. The tolerance factor, denoted by ϵ, is set to be a constant value for every iteration (10^{-6} in our simulations). Another way would be to adjust ϵ adaptively according to the number of non-delayed estimates. It chooses smaller values of ϵ for smaller values of $|S_1^{(k)}|$. That is, we define a vector $\epsilon = [\epsilon_1, \ldots, \epsilon_N]$ such that for $|S_1^{(k)}| = 1$ we use ϵ_1, for $|S_1^{(k)}| = 2$ we use ϵ_2, and so on, where $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_N$, [26]. The different steps of A-ADMM with Strategy II are shown in **Algorithm 4**.

Strategy II with Gradient Update

It is well-known that ADMM algorithms typically show steep initial convergence to a ball around the optimal point, and slow convergence towards the optimal thereafter [31]. This fact

Algorithm 4 A-ADMM with Strategy II

```
procedure CentralPDC(ϵ)
    initialize: k = 1, d_1^*
    repeat
        repeat
            wait
            receive updates (w_i^(k), a_i^(k+1))
        until timer ≤ d_1^* or all updates received
        update z^(k+1) by (3.10), update Δz by (3.12)
        broadcast z^(k+1) to all local PDCs
        k ← k + 1
    until Δz ≤ ϵ
end procedure
procedure LocalPDC i
    initialize: k = 1, w_i^(0) = 0, a_i^(0) = 1, d_2^*;
    repeat
        repeat
            wait
        until timer ≤ d_2^*, or receive z^(k)
        update y_i(k), Ĥ_i^(k), ĉ_i^(k) by (2.29)
        if global update received then
            update (w_i^(k), a_i^(k+1)) by (2.34a), (2.34b)
        else
            update (w_i^(k), a_i^(k+1)) by (3.11a), (3.11b)
        end if
        send (w_i^(k), a_i^(k+1)) to the central PDC
        k ← k + 1
    until termination
end procedure
```

Table 3.3: Comparison of A-ADMM strategies w.r.t sensitivity to network characteristics

Strategy Type	Packet Loss	Packet Delay	Delay Threshold
Strategy I - Skipping	high	high	high
Strategy II - Previous Update	high	low	medium
Strategy II with Gradient Method	medium	low	low

motivates us to add a gradient update term to (3.11) to improve convergence as:

$$\mathbf{w}_i^{(k)} = \mathbf{w}_i^{(k-1)} + \rho(\mathbf{a}_i^{(k)} - (\mathbf{z}^{(l_i)} + \gamma_i(\mathbf{z}^{(l_i)} - \mathbf{z}^{(l_i-1)}))), \tag{3.13a}$$

$$\mathbf{a}_i^{(k+1)} = ((\hat{\mathbf{H}}_i^{(k)})^T \hat{\mathbf{H}}_i^{(k)} + \rho I)^{-1}((\hat{\mathbf{H}}_i^{(k)})^T \hat{\mathbf{c}}_i^{(k)} - \mathbf{w}_i^{(k)} + \rho(\mathbf{z}^{(l_i)} + \gamma_i(\mathbf{z}^{(l_i)} - \mathbf{z}^{(l_i-1)})), \tag{3.13b}$$

where, $l_i \in \{k-1, k-2, \dots\}$, and γ_i is the step size whose value can be flexibly adjusted to expedite convergence. That is, now the local PDC $i, i \notin S_2^{(k)}$ uses $\mathbf{z}^{(l_i)} + \gamma_i(\mathbf{z}^{(l_i)} - \mathbf{z}^{(l_i-1)})$ instead of $\mathbf{z}^{(l_i)}$ in its update equations. Since (2.34) is solving consensus, this gradient information can improve the closeness of $\mathbf{z}^{(k)}$ to $\mathbf{z}^{(k+1)}$ compared to $\mathbf{z}^{(l_i)}$. The update at the central PDC remains same as in (3.10).

Table 3.3 compares the three A-ADMM strategies in terms of their sensitivity to different communication bottlenecks.

Other Strategies

Other heuristic strategies can also be used to mitigate asynchrony by taking advantage of fact that (2.34) is a consensus problem. One such strategy can be to use spatial correlation between the estimates generated by the local PDCs at every iteration. Local estimates $\mathbf{a}_i^{(k)}$ and $\mathbf{a}_j^{(k)}$ for any two PDCs i and j, located at two different spatial locations in the grid, are likely to be close in their magnitudes over iteration k, and exactly equal as $k \to \infty$. The closeness at any given k, especially for smaller values of k, of course depends on the difference between the initial conditions $\mathbf{a}_i^{(0)}$ and $\mathbf{a}_j^{(0)}$, and individual convergence rates of the two PDCs depending on their network traffic. The idea, therefore, is to keep track of the correlation factors between every pair of local estimates at the central PDC. The correlation coefficient $\rho_{(\mathbf{a}_i^{(k)}, \mathbf{a}_j^{(k)})}$ between $\mathbf{a}_i^{(k)}$ and $\mathbf{a}_j^{(k)}$ with expected values $\mu_{\mathbf{a}_i^{(k)}}$ and $\mu_{\mathbf{a}_j^{(k)}}$ and standard deviations $\sigma_{\mathbf{a}_i^{(k)}}$ and $\sigma_{\mathbf{a}_j^{(k)}}$ is defined as

$$\rho_{(\mathbf{a}_i^{(k)}, \mathbf{a}_j^{(k)})} = \frac{cov(\mathbf{a}_i^{(k)}, \mathbf{a}_j^{(k)})}{\sigma_{\mathbf{a}_i^{(k)}} \sigma_{\mathbf{a}_j^{(k)}}} = \frac{E[(\mathbf{a}_i^{(k)} - \mu_{\mathbf{a}_i^{(k)}})]E[(\mathbf{a}_j^{(k)} - \mu_{\mathbf{a}_j^{(k)}})]}{\sigma_{\mathbf{a}_i^{(k)}} \sigma_{\mathbf{a}_j^{(k)}}}. \tag{3.14}$$

The central PDC computes the correlation matrix $\mathcal{C}^k$ as

$$\mathcal{C}^k = \begin{bmatrix} \rho_{(\mathbf{a}_1^{(k)},\mathbf{a}_1^{(k)})} & \rho_{(\mathbf{a}_1^{(k)},\mathbf{a}_2^{(k)})} & \cdots & \rho_{(\mathbf{a}_1^{(k)},\mathbf{a}_N^{(k)})} \\ \rho_{(\mathbf{a}_2^{(k)},\mathbf{a}_1^{(k)})} & \rho_{(\mathbf{a}_2^{(k)},\mathbf{a}_2^{(k)})} & \cdots & \rho_{(\mathbf{a}_2^{(k)},\mathbf{a}_N^{(k)})} \\ \vdots & \vdots & \cdots & \vdots \\ \rho_{(\mathbf{a}_N^{(k)},\mathbf{a}_1^{(k)})} & \rho_{(\mathbf{a}_N^{(k)},\mathbf{a}_2^{(k)})} & \cdots & \rho_{(\mathbf{a}_N^{(k)},\mathbf{a}_N^{(k)})} \end{bmatrix}. \tag{3.15}$$

If at iteration $(k+1)$ any estimate $\mathbf{a}_i^{(k+1)}$ does not arrive on time, then the central PDC scans $\mathcal{C}^k$, and locates the index j such that $\rho_{(\mathbf{a}_i^{(k)},\mathbf{a}_j^{(k)})}$ has the highest magnitude among all entries in the ith row of $\mathcal{C}^k$. If $\mathbf{a}_j^{(k+1)}$ has arrived on time, then it substitutes the missing value $\mathbf{a}_i^{(k+1)}$ by $\mathbf{a}_j^{(k+1)}$ instead of $\mathbf{a}_i^{(l_i+1)}$ as in (3.10). In other words, (3.10) now takes the form

$$\mathbf{z}^{(k+1)} = \frac{1}{N}\Big(\sum_{i\in S_1^{(k)}} (\mathbf{a}_i^{(k+1)} + \frac{\mathbf{w}_i^{(k)}}{\rho}) + \sum_{j\notin S_1^{(k)}} (\mathbf{a}_{J_j}^{(k+1)} + \frac{\mathbf{w}_{J_j}^{(k)}}{\rho})\Big), \tag{3.16}$$

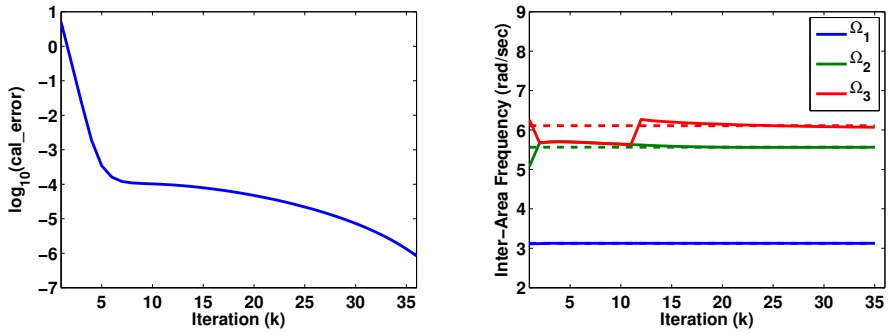
where $J_j = \arg\max_{i\in S_1^{(k)}} \mathcal{C}^k(j,i)$.

3.3.3 Simulation Results

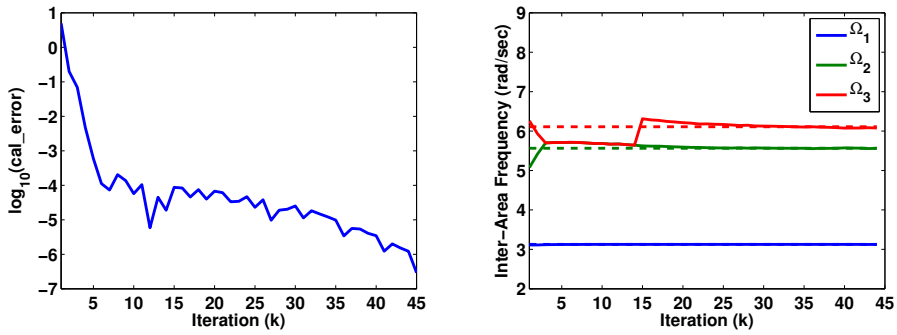
To verify the A-ADMM algorithms described in Subection 3.3.2 we consider the IEEE 68-bus system. The system is divided into 4 areas, each with one local PDC and 3 PMUs. The simulated measurements of bus frequency are obtained using the Power Systems Toolbox (PST) nonlinear dynamics simulation routine. A three-phase fault is considered occurring at the line connecting buses 1 and 2. The fault starts at $t = 0.1$ sec, clears at bus 1 at $t = 0.15$ sec and at bus 2 at $t = 0.20$ sec. The measurements are downsampled and the sampling period T is increased up to 0.2 seconds. Our objective is to estimate the post-fault inter-area oscillation modes of the system using these real-time PMU measurements of frequency. Since there are 16 generators, each with six states, our proposed algorithms should ideally solve a 96th-order polynomial. However, offline analysis of the model revealed that choosing $2n = 40$ yields a satisfactory estimate of the inter-area modes as most of the residues for the high frequency modes are practically zero. The initial 10 samples (2 seconds) of the measurements are gathered before starting the optimization iterations. We set $\rho = 10^{-9}$, and $\epsilon = 10^{-6}$.

S-ADMM v.s. A-ADMM

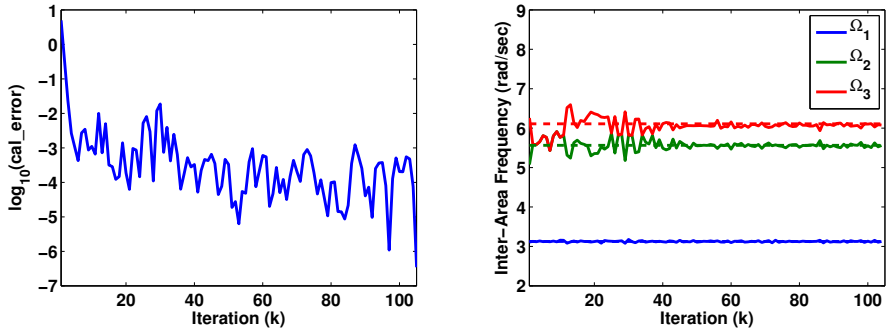
We first consider a fundamental comparison of S-ADMM and A-ADMM with Strategy I. From our delay model, we note that $P(X \leq 5.67) = 0.8$, meaning that $\mathbf{d}_1^* = \mathbf{d}_2^* = 5.67$ ms will lead to 20% of the messages be delayed. Using these values, we simulate 1000 runs of the A-ADMM



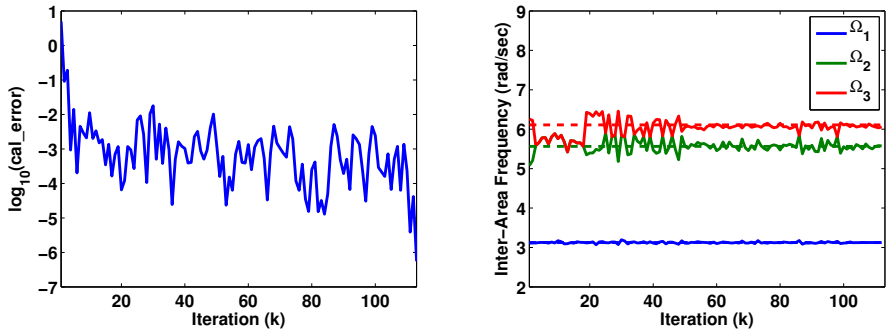
(a) S-ADMM



(b) Downlink Case of A-ADMM with Strategy I

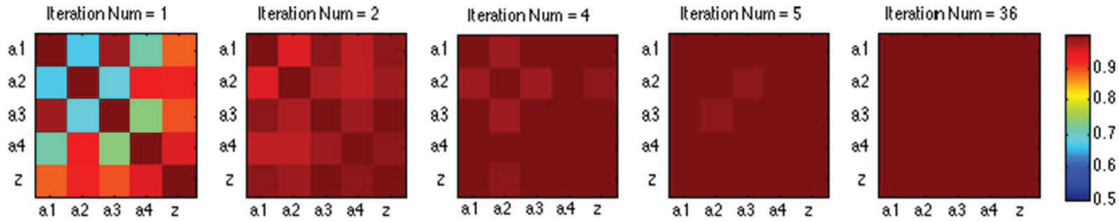


(c) Uplink Case of A-ADMM with Strategy I

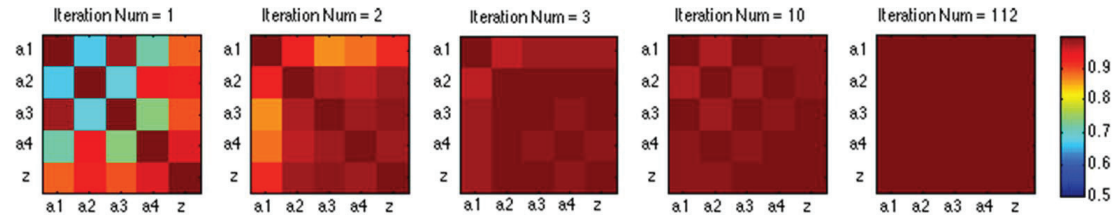


(d) Bi-link Case of A-ADMM with Strategy I

Figure 3.7: Comparison of convergence and accuracy of Strategy I and S-ADMM



(a) S-ADMM



(b) Bi-link Case of A-ADMM with Strategy I

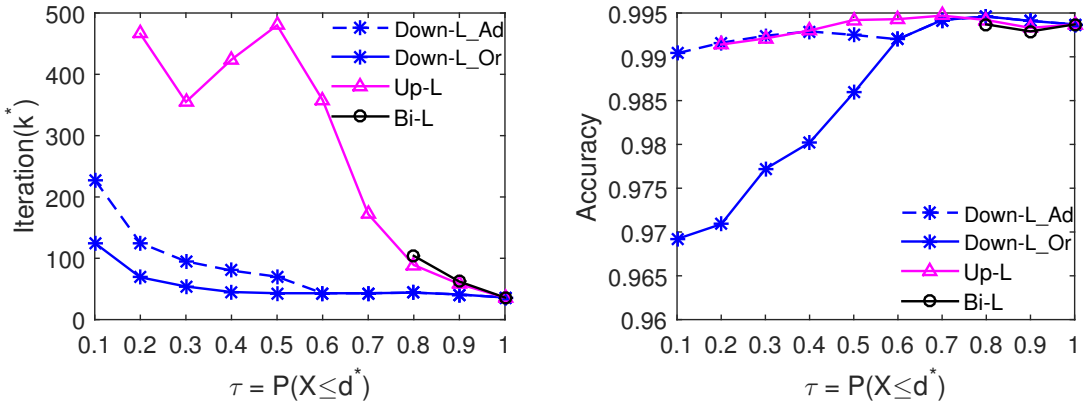
Figure 3.8: Correlation analyses for S-ADMM v.s A-ADMM with Strategy I

experiments with Strategy I, and plot the expected values of the convergence error in Fig. 3.7. The strategy is subdivided into three different cases, namely: 1) Downlink case, where we only consider those runs where downlink deadlines are missed, but uplink deadlines are always met; 2) Uplink case, where the reverse happens; and 3) Bi-link case, which is the usual A-ADMM with Strategy I. The left panel of Fig. 3.7 shows that compared to the smooth convergence of S-ADMM, the convergence of A-ADMM becomes jittery as more asynchrony is added. Setting the accuracy of estimation to be fixed at 99.5% or more, convergence rate clearly slows down from 36 iterations in S-ADMM to 45 for the Downlink case, 105 for the Uplink case, and 113 for general A-ADMM with bi-directional delay. The right panel of this figure shows the convergence and accuracy of the estimated values of the frequency of the three dominant modes. Fig. 3.8 shows the correlation plots of every local estimate and the consensus variable over different iterations, illustrating the relative convergence rates of the estimates at the five different PDCs.

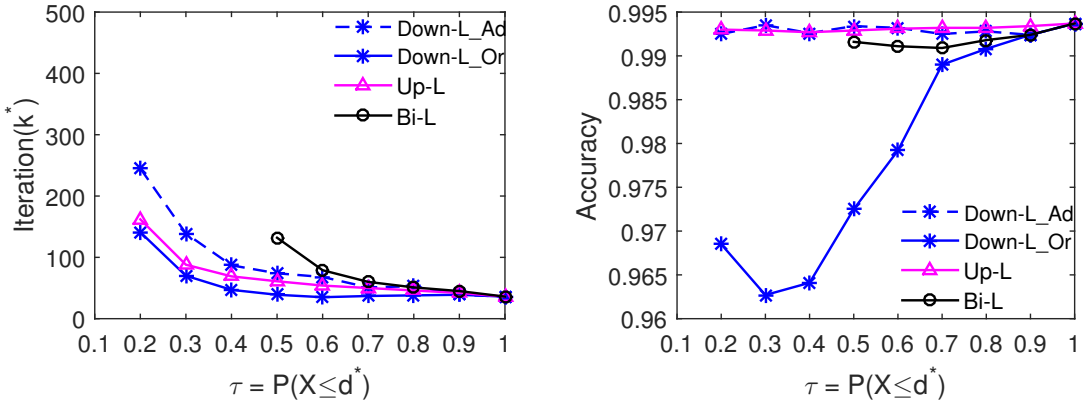
Sensitivity of A-ADMM to Delay Thresholds

We next test the impact of $\mathbf{d}_1^*$ and $\mathbf{d}_2^*$ on the convergence and accuracy of the A-ADMM strategies. For each strategy, we run 200 runs for each different choice of the delay thresholds by using 200 pre-generated sets of delays using model (3.4). Fig. 3.9 and 3.10 show the relationships between different delay thresholds and the average convergence rate and accuracy.

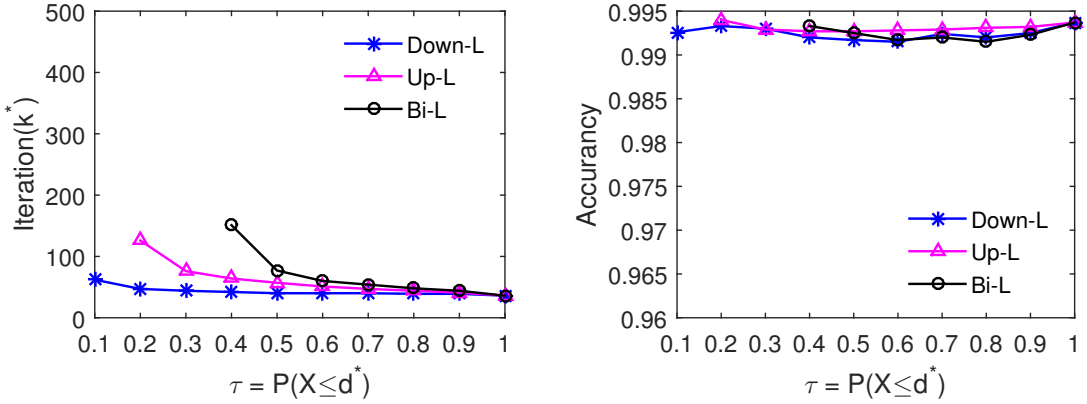
Strategy I: From the left subfigure in Fig. 3.9(a), we see that in the Downlink case, the number of iterations increases slowly when the CDF τ of the delay threshold $\mathbf{d}_2^*$ decreases from 1 to 0.2. Even when $\tau = 0.1$, meaning that $\mathbf{z}^k$ only has 10% possibility to arrive by the



(a) Strategy I - Skipping



(b) Strategy II - Lost Data



(c) Strategy II - Delayed Data

Figure 3.9: Comparison of convergence and accuracy of Strategy I and II w.r.t delay threshold

deadline, the algorithm still converges in 100 iterations. However, in Uplink case, once $\tau = 0.1$, the algorithm faces numerical instability, and therefore diverges. Also, as τ decreases from 1 to 0.5, k^* increase dramatically up to 500, while it remains around the value of 400 for $\tau \in [0.2, 0.4]$. As expected, the bi-link case has the shortest stable range of $\tau \in [0.8, 1]$, and largest value of k^*, compared to the other two cases. If $\tau \leq 0.7$, the algorithm diverges irrespective of the choice of the deadlines, indicating that Strategy I is very sensitive to bi-directional delays. Estimation accuracy of 99.1% or more is maintained by adjusting ϵ of the stop criterion from 10^{-6} to 10^{-8} or 10^{-7}, as shown in the blue dash lines in Fig. 3.9(a). After the adjustment, the Downlink case still has much better convergence than the Uplink case, indicating that message-skipping impacts the averaging step far more than the local least-square update step.

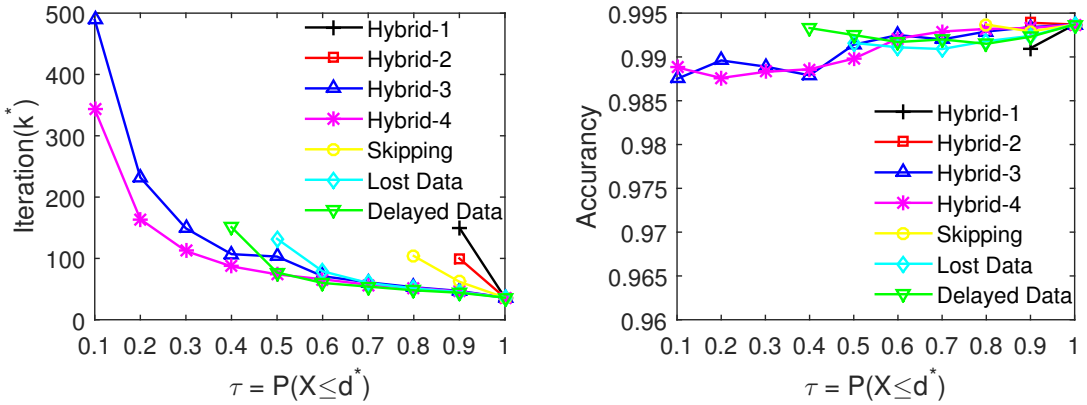
Strategy II - Lost Data: Strategy II is further divided into two subcases - namely, lost data and delayed data. Lost data essentially implies that if any message does not arrive on time at any PDC it is considered to be unusable for future iterations. Fig. 3.9(b) shows that similar to Strategy I, the accuracy of this strategy drops dramatically from 99.08% to 96.27% when τ decreases from 0.8 to 0.2 in Downlink case. Even after adjustment of ϵ, it has worse convergence than Strategy I. This observation implies that if the local PDC i does not receive any message on time it is more advisable to skip the update rather than using out-of-date information about $\mathbf{z}^{(k)}$. For the Uplink case, however, this strategy has significantly better convergence and accuracy guarantees than Strategy I, especially for $\tau \in [0.2, 1]$, implying that when the probability of data loss is high, it is more advisable for the central PDC to use stored values of local updates rather than skipping them.

Strategy II - Delayed Data: This is the usual A-ADMM with Strategy II, where delayed data are used in future iterations. It has the best convergence and accuracy guarantees among the three representative cases described so far, partly because of zero packet loss rate, and partly due to limited maximum communication delay. In Downlink case, the blue line in the left subfigure of Fig. 3.9(c), indicates that the iteration number k^* has the lowest value for each given τ, compared to the blue lines of both Strategy I and Strategy II - Lost data before adjustment. In Uplink case, the magenta line shows smallest k^* in the same effective range of $\tau \in [0.2, 1]$. For the Bi-link case, this not only converges in the smallest number of iterations k^* for a given τ, but also has the longest effective range of $\tau \in [0.4, 1]$ guaranteeing numerical stability.

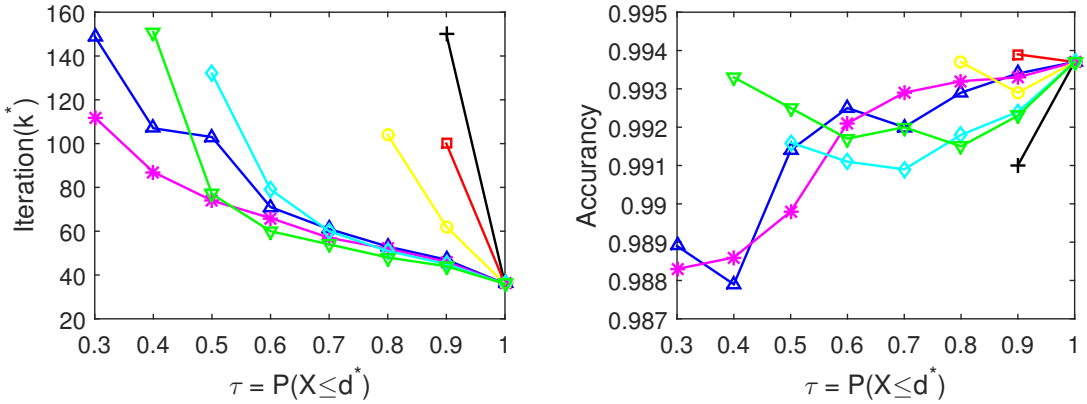
Hybrid Strategies: So far we have considered cases where Strategy I and Strategy II are employed independently for the estimation. We next consider the cases when these two strategies are used in combinations. Four possible such hybrid strategies, shown in Table 3.4, are considered. Fig. 3.10 shows the relationship between the delay threshold and the convergence and accuracy guarantees of these hybrid cases. Three main conclusions can be drawn from Fig. 3.10: (1) Hybrid-1 and Hybrid-2 have worse convergence because Strategy I is employed

Table 3.4: Hybrid Control Strategies

Strategy Type	Uplink	Downlink
Hybrid-1	Skipping	Lost Data
Hybrid-2	Skipping	Delayed Data
Hybrid-3	Lost Data	Skipping
Hybrid-4	Delayed Data	Skipping



(a) Hybrid Strategies



(b) Hybrid v.s. Standard

Figure 3.10: Comparison of convergence and accuracy of hybrid strategies w.r.t delay threshold

in the uplink communication, while Strategy II is applied to the downlink communication. When $\tau \leq 0.8$, they force the algorithm to start to diverge, as shown with black and red lines in the left subfigure of Fig. 3.10(a), respectively; (2) A magnified view of Fig. 3.10(b) shows that when the delay thresholds $(\mathbf{d}_1^*, \mathbf{d}_2^*)$ are larger than 5.35 ms, namely $\tau \geq 0.5$, the standard Strategy II-Delayed Data has best convergence with accuracy above 99.13%; (3) Finally, when $\tau \in [0.1, 0.4]$, Hybrid-3 and Hybrid-4 still guarantee convergence, while both Strategy II-Lost data and Strategy-II Delayed data start to diverge. Therefore, if we need to set the delay thresholds to small values, Hybrid-3 is the best choice.

Strategy II with Gradient Update: This strategy is most effective when the initial conditions for the local updates are noticeably different from each other. To demonstrate this method we set $\mathbf{a}_1^{(0)} = 0.01\mathbf{1}, \mathbf{a}_2^{(0)} = 0.1\mathbf{1}, \mathbf{a}_3^{(0)} = \mathbf{1}, \mathbf{a}_4^{(0)} = 10\mathbf{1}$, where $\mathbf{1}$ is a vector of ones with 40 rows (the number of unknown coefficients of the characteristic polynomial is considered to be 40). The convergence guaranteed by S-ADMM in this case is $k^* = 117$, and accuracy (on a scale of 1) is $\alpha = 0.9938$. Table 3.5 shows that the gradient method improves convergence k^* for relatively the same accuracy factor α by tuning the step size γ, considering $\gamma_i = \gamma$, $\forall i = 1, .., N$. The symbol '$-$' in the table means that the algorithm diverges. The stable range of $\tau \in [0.2, 0.9]$ for Downlink and $\tau \in [0.4, 0.9]$ for Bi-link is increased to $\tau \in [0.1, 0.9]$, and $\tau \in [0.5, 0.9]$, respectively, after the gradient update. Table 3.5 also shows the optimal value of γ for a given τ. It should be noted that when τ is large, i.e., when the delay threshold is large, then γ must be chosen to be large as well for optimal convergence. The range of stable γ, however, is increased in that condition.

The heuristic strategy involving spatial correlation analysis, as in (3.15), for this example shows comparable results to the above, especially when the initial conditions are scattered. However, we defer the details of those results to future for the sake of brevity and space limitation.

3.4 Conclusion

In this chapter, we presented four cyber-physical estimation algorithms for wide-area oscillation monitoring using Synchrophasors. Our algorithms demonstrate how multitudes of geographically dispersed PMUs and PDCs can communicate with each other, and how the various binding factors in the network protocols can pose bottlenecks for their communication. The results, thereby provide valuable insights and guidance in deploying future PMU and PDC infrastructures, not only for power systems but for any generic cyber-physical sensor network. Next, we will like to design the attack-resilient strategies for the distributed PronyADMM algorithm which might be able to show how the distributed architecture is more resilient to communication failures as compared to alternative centralized methods.

Table 3.5: Sensitivity of Gradient Update to Delay Threshold

$\tau = P(X \leq d^*)$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.9
Lost Data - Downlink (k^*/α)	-	355/0.9914	218/0.9914	160/0.9916	132/0.9917	118/0.9916	108/0.9916	83/0.9913
Gradient Method (k^*/α)	1117/0.9897	317/0.9913	218/0.9914	162/0.9916	130/0.9916	113/0.9917	102/0.9918	77/0.9914
Step Size γ	-0.4	-0.2	0.001	0.015	0.01	0.9	0.8	1.9
Effective γ Range	-	[-0.7 0.1]	[-0.9 0.5]	[-1.2 0.9]	[-1.6 1]	[-2.2 1.5]	[-2.5 2]	[-6 5]
Lost Data - Bi-link (k^*/α)	-	-	-	-	239/0.9917	170/0.9912	148/0.9918	100/0.9917
Gradient Method (k^*/α)	-	-	-	628/0.9911	230/0.9918	168/0.9912	133/0.9918	94/0.9917
Step Size γ	-	-	-	-0.3	-0.3	-0.3	0.5	1
Effective γ Range	-	-	-	-	[-1 0.3]	[-1.7 0.6]	[-1.8 1]	[-4.6 3.5]
Delayed Data - Downlink (k^*/α)	113/0.9915	116/0.9914	114/0.9914	118/0.9915	118/0.9914	111/0.9915	102/0.9912	85/0.9913
Gradient Method (k^*/α)	112/0.9916	107/0.9914	100/0.9915	89/0.9915	84/0.9914	77/0.9915	75/0.9913	78/0.9917
Step Size γ	0.01	0.2	0.35	0.6	0.8	0.9	1	1.8
Effective γ Range	[-0.3 0.12]	[-0.6 0.25]	[-1 0.45]	[-1.4 0.7]	[-1.8 0.8]	[-2 1.2]	[-2.5 1.7]	[-5 5.2]
Delayed Data - Bi-link (k^*/α)	-	-	-	-	130/0.9913	128/0.9913	119/0.9914	93/0.9913
Gradient Method (k^*/α)	-	-	-	196/0.9913	123/0.9914	105/0.9913	101/0.9914	85/0.9914
Step Size γ	-	-	-	-0.02	0.3	0.6	0.8	2.3
Effective γ Range	-	-	-	[-0.3 0.2]	[-1.2 0.5]	[-2 1]	[-2.5 1.5]	[-5.5 3.5]

Chapter 4

Security Enhancement through Distributed Optimization

In Chapter 3 we have shown the convergency analysis of distributed ADMM-based optimization through the impact of delay model parameters and different asynchronous control strategies on convergency and accuracy of the optimization algorithms. In this chapter, we further verify the security enhancement through the distributed optimization. Cyber-physical systems integrate physical processes, computational resources and communication capabilities. The wide-area monitoring system of power systems is a typical representative of cyber-physical architecture, which is prone to failures and attacks on their physical infrastructure, as well as cyber attacks on their data management and communication layer [17]. Motivated by this challenge, we address the problem of implementing wide-area monitoring algorithms over a distributed communication infrastructure using massive volumes of real-time PMU data. Our goal is to establish how distributing a monitoring functionality over multiple estimators can guarantee significantly more resiliency against extreme events. Such events may result from both malicious attacks on the cyber and physical assets as well as due to natural calamities such as storms and earthquakes. The specific monitoring algorithm that we study is the estimation of the frequency and damping of the electro-mechanical oscillations seen in the power flows in the grid after any disturbance. If the system size is small then it is straightforward to estimate these oscillation modes, or equivalently the eigenvalues and the eigenvectors of its state matrix, in a centralized way. Algorithms such as Eigenvalue Realization Algorithm (ERA), Prony analysis, and mode metering [32], for example, have been widely used by the WAMS community over the past decade for this purpose. However, as the system size and the number of PMUs scale up, the computational costs of these algorithms explode, and they completely fail to provide the required resiliency. As a solution, in this chapter we employ a distributed Prony-based algorithm combined with Alternating Direction Method of Multipliers [33] to estimate the frequency, damping and

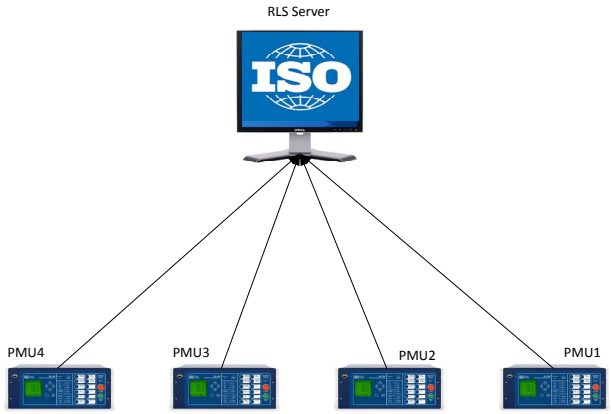


Figure 4.1: Diagram of Centralized RLS Algorithm Implementation

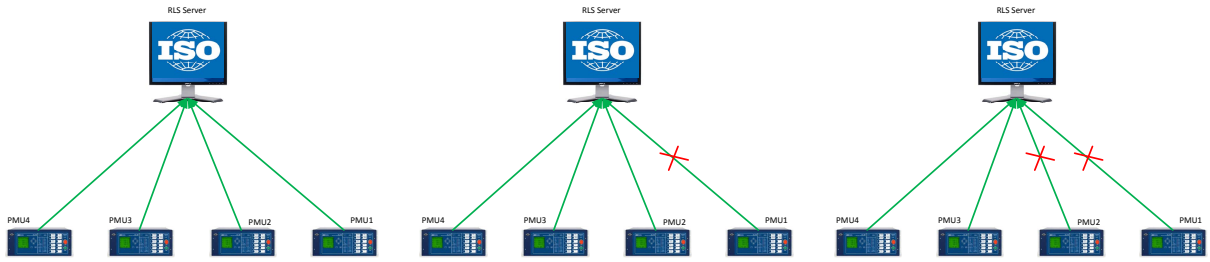
residue of each oscillation mode via distributed consensus. We partition the network topology into multiple clusters, with each cluster equipped with a local estimator at the local control center. At any iteration, the local estimators receive PMU measurements from within their own respective areas, run a local consensus algorithm, and communicate their estimates to a central estimator. The central estimator averages all estimates, and broadcasts the average back to each local estimator as the consensus variable for their next iteration. We also use Recursive Lease Squares (RLS) for the centralized estimation. By imposing a redundancy strategy between the local and the global estimators via mutual coordination, we show that the distributed algorithm is highly more resilient to communication failures than centralized. We also show that in case of an attack the local estimators only need to exchange a certain set of parameter estimates, and not actual PMU data, because of which the proposed distributed algorithm is *privacy preserving*. We illustrate our results using a IEEE 39-bus system model emulated via a federated testbed between NC State University (Phasorlab) and the Information Sciences Institute (DETERLab) [34].

4.1 Attack Senarioes and Resiliency Mechanism Design

We consider three different types of attacks using DETERLab, namely (1) a malware attack that disrupts the operation of a PDC resulting in abnormal termination of an estimation algorithm, (2) flooding attack, where an attacker can flood a targeted network link with malicious traffic resulting in the reduction of the estimation traffic due to overload, and (3) malfunctioning of physical infrastructure or hardware, eg. shutdowns due to power outages caused by earthquakes and other natural disasters.

4.1.1 Resilient Strategy of Centralized RLS

For the centralized RLS algorithm shown in Figure (4.1), the RLS server stream the PMU data from all PMU machines, and then run the centralized RLS algorithm. It is very intuitive that if the RLS server is attacked due to one of aforesaid attack types, the algorithm will be stopped immediately, which is the worse attack scenario. Nevertheless, we still design this attack-resiliency mechanism that once one or several PMU machines is/are attacked, the resilient version of centralized RLS algorithm can continue to accommodate the rest of PMU machines. The progress of this resilient strategy for the centralized RLS is shown in Figure (4.2).



(a) Normal Operation (b) PMU1 got attacked (c) PMU2 got attacked

Figure 4.2: Resiliency Strategy of Centralized RLS

4.1.2 Resilient Strategies of Distributed PronyADMM

For the distributed PronyADMM algorithm shown in Figure (4.3), the local Prony virtual machine is employed to stream local PMU data and run local parameter estimation algorithm for each area. At each iteration, the server machine collects all parameters locally estimated and takes average, which is broadcasted to each local Prony virtual machine to continue the local estimation for the next iteration. Compared to the centralized case, the distributed PronyADMM architecture has two alternative resilient strategies.

Strategy I is called the standby backup server strategy, in which, when the main server is attacked and out of service, each local Prony virtual machine creates a new connection with the backup server and continues the estimation algorithm. The reason for the feasibility of backup server strategy in the distributed Prony ADMM is that the server only needs to collect local estimations at the current iteration. However, this strategy does not work for the centralized RLS algorithm, where the server demands the historical PMU data to construct the current matrices $\hat{\mathbf{H}}_i^{(k)}, \hat{\mathbf{c}}_i^{(k)}$, but the backup server does not contain such data. The progress of this

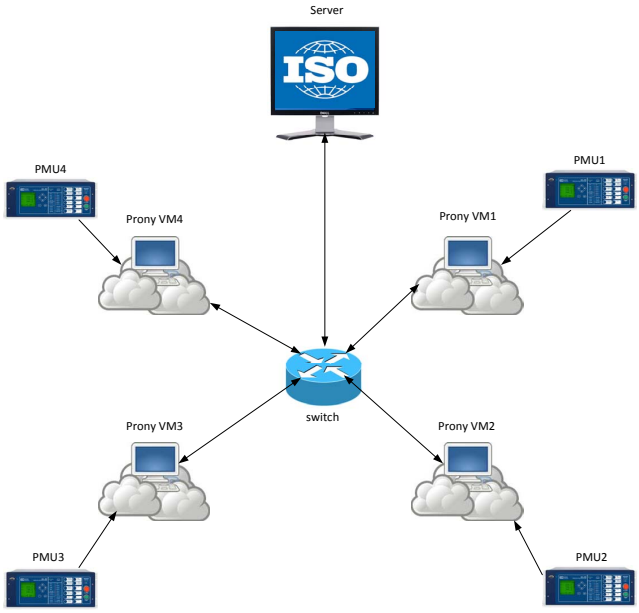
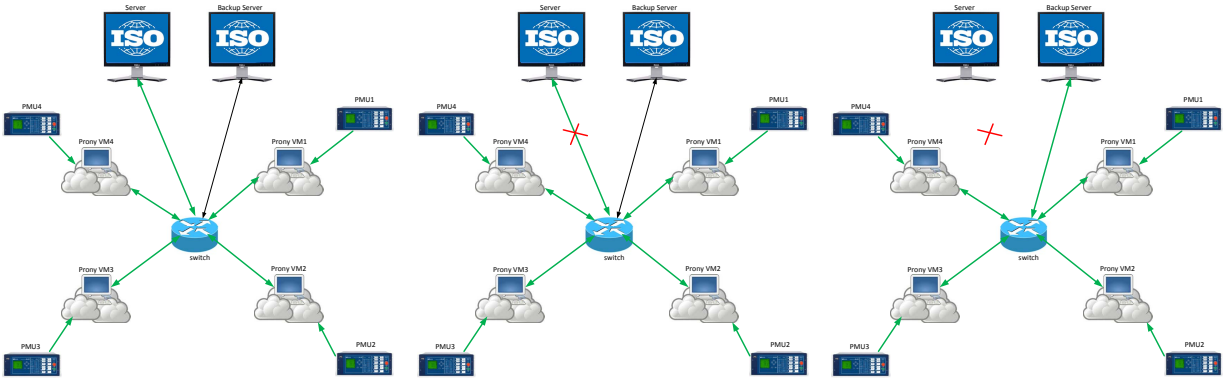


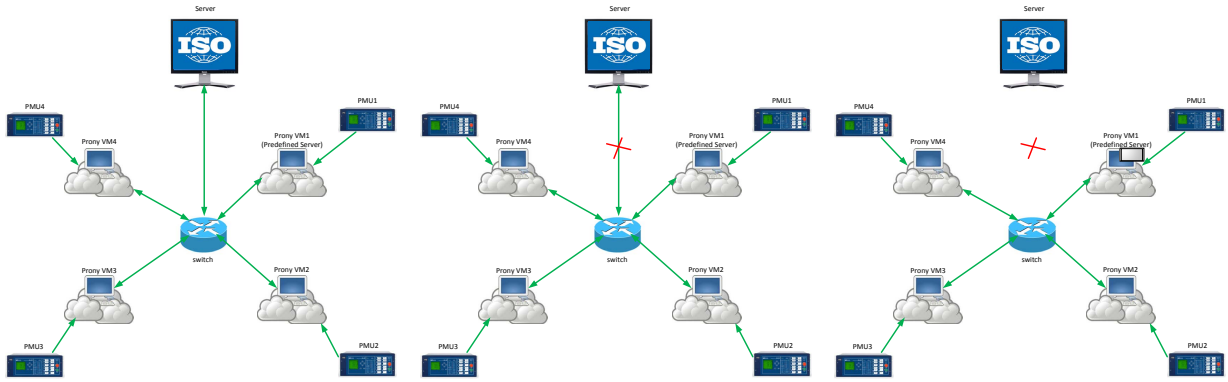
Figure 4.3: Diagram of Distributed PronyADMM Algorithm Impementation

strategy is shown in Figure (4.4). Alternatively, the distributed PronyADMM architecture can choose one of local Prony virtual machines as the backup server to continue running the estimation algorithm even at the worse attack scenario namely the Sever is attacked, shown in Figure (4.5). This attack-resiliency strategy is called as Strategy II. Such two attack-resiliency strategies are essentially the redundancy strategy.



(a) Normal Operation (b) Server got attacked (c) Backup server works

Figure 4.4: Resiliency Strategy I of Distributed PronyADMM



(a) Normal Operation (b) Server got attacked (c) Backup server works

Figure 4.5: Resiliency Strategy II of Distributed PronyADMM

4.2 Experimental Verification via Federated Testbeds

We first implement the centralized RLS and distritubed PronyADMM algorithms without considering attack-resiliency mechanisms using C programming language, and the flow charts of implementation are shown Appendix (A.1). Next, we implement the RLS and Prony-ADMM algorithms with aforesaid resilient mechamisms, and the flowcharts of resiliency implementation for both centralized RLS and distributed PronyADMM cases are shown in Appendix (A.2).

We next experiment with them under various realistic attack scenarios. Specifically, we create representative smart-grid topologies in the networking and cyber security testbed at USC/ISI called DETERLab [34], and integrate them with real-time Synchrophasor data generated from a power system testbed at NCSU called Phasorlab. We simulate a IEEE 39-bus system at Phasorlab using Real-time Digital Simulators (RTDS), and excite the model with a three-phase fault lasting for 0.3 mins. The system consists of 10 synchronous generators, partitioned into 4 coherent clusters with one PMU in each cluster. Traces of the frequency measurements from these four PMUs are shown in Figure 4.6. We represent the communication topology for RLS in DETER with four PMU nodes, generating the frequencies $y_1(k)$, $y_2(k)$, $y_3(k)$, and, $y_4(k)$, $k = 0, \ldots, 1800$, with a sampling rate of $T = 0.01$ seconds, and sending them to a central server node that executes (2.34). To facilitate the estimation speed, for this case we down-sample the data at $T = 0.2$ second. The server node receives data packets with size of 250 bytes, which usually contains around 25 samples from total four PMU/PDC nodes in parallel with each other.

For distributed estimation, each partitioned cluster has an additional Prony node representing a local PDC. The PMU nodes stream their local measurements $y_i(k)$, $i = 1, 2, 3, 4$ to the local

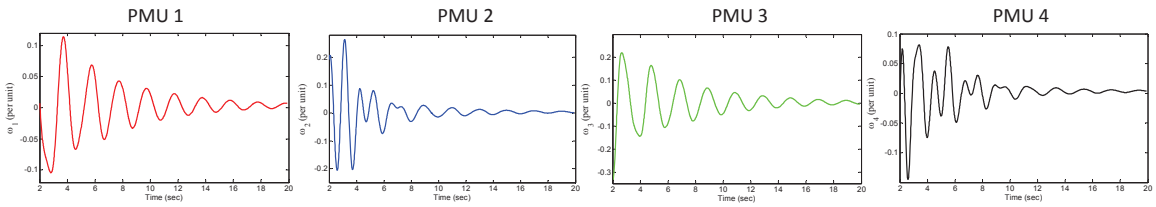


Figure 4.6: PMU measurements of frequency from buses 1, 3, 4, 8 (in per unit)

Prony node. The Prony nodes run (2.34), and send their individual estimate $\mathbf{a}_i$ to the server node. The server nodes computes $\bar{\mathbf{a}}$, and broadcasts it back to the Prony nodes for the next iteration. Since every node exchanges only a parameter vector with the server, and vice versa, and not any actual PMU measurement, complete 'data privacy' is maintained between the clusters.

The estimates for the three slow or *inter-area* modes for the 4-area system are shown in the second and third columns of Table 6.2. The actual values of the modes are shown in the first column. It can be seen that both RLS and distributed Prony yield reasonably accurate estimates. The slight mismatches in each from the actual values are mostly attributed to the sensitivity of the root finding step to small errors in b. The accuracy of RLS improves with more educated guesses for Θ_0.

Table 4.1: Estimated Slow Eigenvalues for the 39-bus Power System

Actual Eigenvalues	Centralized RLS	Distributed Prony-ADMM
$-0.1993 \pm j3.1255$	$-0.197 \pm j3.125$	$-0.1966 \pm j3.1258$
$-0.6239 \pm j5.5644$	$-0.5113 \pm j5.5889$	$-0.5111 \pm j5.5773$
$-0.5112 \pm j6.1090$	$-0.3598 \pm j6.0868$	$-0.4932 \pm j6.0926$

Table 4.2: Accuracy Evaluation for Attack Scenarios

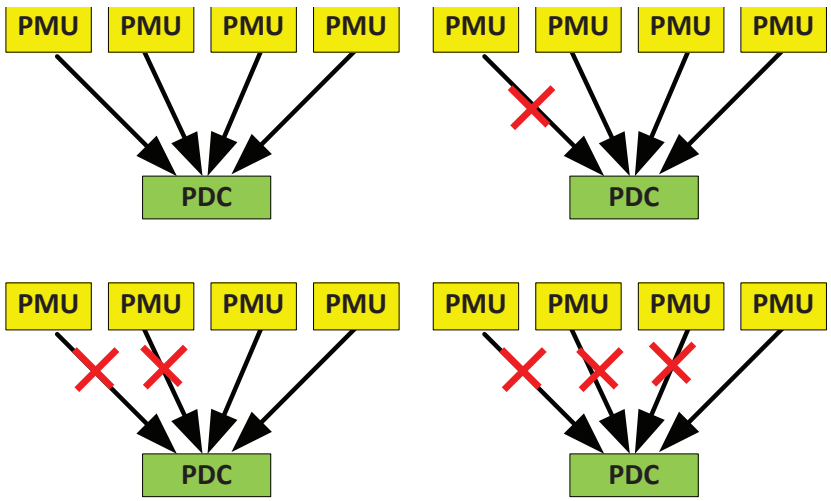
Actual Eigenvalues	Centralized RLS			Prony-ADMM
	1 PMU attacked	2 PMUs attacked	3 PMUs attacked	Server attacked
$-0.1993 \pm j3.1255$	$-0.1800 \pm j3.1258$	$-0.2615 \pm j3.1548$	$-0.3926 \pm j3.3219$	$-0.1963 \pm j3.1255$
$-0.6239 \pm j5.5644$	$-0.6519 \pm j5.6453$	$-0.7269 \pm j5.5093$		$-0.5137 \pm j5.5872$
$-0.5112 \pm j6.1090$	$-0.2213 \pm j5.8828$	$-0.0682 \pm j6.4957$	$-0.6565 \pm j6.6205$	$-0.4944 \pm j6.0843$

4.2.1 Case Study 1: Centralized RLS

We implement the flooding attack sequentially on each of the four communication links in the RLS topology, as shown in Figure 4.7. Once the algorithm detects an attack, it takes three immediate actions: 1) update the list of live PMUs, 2) resize the computation matrices A and B in (2.12), and 3) reset the initial guess for Θ to its value immediately before the attack. However, even with these accommodations, columns 2, 3 and 4 of Table 4.2 show that the accuracy of RLS estimation degrades significantly as more PMUs get disconnected by the attacks.

In fact, as shown in the fourth column of Table 4.2, when three PMUs are disconnected RLS cannot identify the second slow mode at all. Additionally, the worse-case scenario is if the RLS server itself gets attacked by a malware or physical malfunction resulting in the server source code to terminate abnormally. In such a scenario, the RLS algorithm can not be resumed anymore.

Figure 4.7: Attacks on RLS links



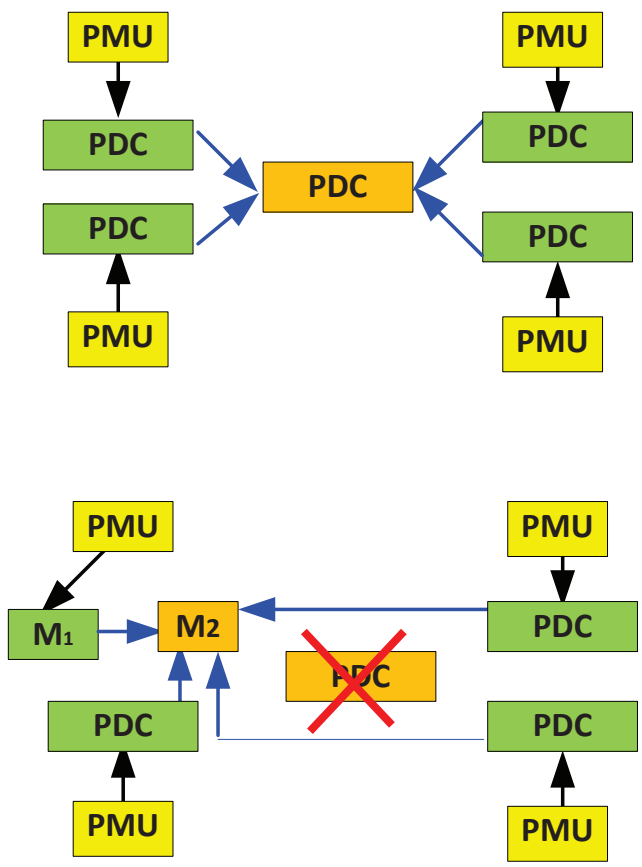
4.2.2 Case Study 2: Distributed PronyADMM

In contrast, a unique feature of the proposed distributed PronyADMM algorithm is that each local PDC can adopt the dual roles of *local estimation* and *central averaging*. For example, as shown in Figure 4.8, if the central server is deactivated by a malware, then the local PDCs can halt their estimation updates immediately, and coordinate with each other to select a candidate PDC among themselves that can act as a *pseudo* central server. If, for instance, local PDC 1 is selected as the pseudo server, then one *module*, say $\mathcal{M}_1$, inside this PDC will continue to

implement the local optimization (2.34a)-(2.34b), while another module $\mathcal{M}_2$ will implement the averaging of $\mathbf{b}_i$ communicated to it from $\mathcal{M}_1$ as well as every other local PDC, to generate $\bar{\mathbf{b}}$, and the algorithm can continue uninterruptedly as soon as all the PDCs agree on the choice of the pseudo server.

For the next round, we assume that $\mathcal{M}_1$ has very high security firewalls and, therefore, is not *allowed* to be compromised, but the hacker has the choice of deactivating $\mathcal{M}_2$ similarly as the central server in the previous round. In that case, the PDCs will halt their estimation again, choose a different candidate for the pseudo-server among themselves, and reassign $\mathcal{M}_2$ in that chosen PDC to resume (2.34). Since none of the steps of the original proposed PronyADMM algorithm changes in these scenarios, the final estimate of β, and therefore of the modes, in this case is almost same as that for the unattacked case shown in the third column of Table 6.2. The fifth column of Table 4.2 testifies this fact.

Figure 4.8: Resiliency Strategy of PronyADMM



4.3 Conclusion

In this chapter we consider a distributed estimation algorithm for computing oscillation modes of power systems from Synchrophasor data, and illustrated its resiliency against component failures compared to conventional centralized techniques. Our future work will include the extension of these methods to closed-loop oscillation damping using delay-tolerant distributed Model Predictive Control, and to validate their resiliency properties using the two federated testbeds.

Chapter 5

Wide-Area Control of Power Systems using Cloud-in-the-Loop Feedback

5.1 Introduction

The Wide-area Measurement Systems (WAMS) technology using GPS-synchronized Synchrophasor measurements is an ideal way to control instabilities in large power systems [36--38]. The challenge, however, is the bottleneck in data communication and computation, which cannot be handled by today's Internet [39--42]. In this chapter, we propose the use of cloud-computing platforms such as GENI (Global Environment for Network Innovations) [44] together with high-speed software defined networks such as Internet2 to combat this challenge. We consider a five-area reduced-order model of the Western Electricity Coordinating Council (WECC), i.e. transmission grid of the US west coast as our test system. We first formulate the reduced model of a multi-area power system following its dynamic equivalent via time-scale separation. Based on the reduced WECC system model, we design the typical distributed LQR control algorithm to combat both small-signal and large-signal disturbances. Finally, we compare the performance between local PSS controller and wide-area controller using the RSCAD based simulation and the simulation result shows that the wide-are controller has much better control performance.

5.2 Problem Formulation

We first recall the general ideas on how a multi-area power system model can be reduced to its dynamic equivalent via time-scale separation. For this, let us consider a power system network consisting of n synchronous generators and n_l loads connected by a given topology. Without

loss of generality, we assume buses 1 through n to be the generator buses and buses $n+1$ through $n+n_l$ to be the load buses. Let P_m denote the vector of the mechanical power injection at generator buses, P_L be the vector of total active power consumed by the loads, and P_i^N be the total active power injected to the ith bus of the network ($i = 1, \ldots, n+n_l$), where the superscript N indicates that this power is flowing in the network as opposed to the loads. This power is calculated as:

$$P_i^N = \sum_{k=1}^{n+n_l} \left(V_i^2 r_{ik}/y_{ik}^2 + V_i V_k \sin(\theta_{ik} - \alpha_{ik})/y_{ik} \right), \tag{5.1}$$

where, $V_i\angle\theta_i$ is the voltage phasor at the ith bus, $\theta_{ik} = \theta_i - \theta_k$, r_{ik} and x_{ik} are the resistance and reactance of the transmission line joining buses i and k, $y_{ik} = \sqrt{r_{ik}^2 + x_{ik}^2}$, and $\alpha_{ik} = \tan^{-1}(r_{ik}/x_{ik})$. Let P_G^N and P_L^N denote the vectors of P_i^N calculated for generators and loads, respectively. The electromechanical model of the power system can be described as a system of differential-algebraic equations (DAE) [46]:

$$\mathcal{M}\ddot{\delta} = P_m - P_G^N - \mathcal{D}\omega, \tag{5.2a}$$

$$P_L - P_L^N = 0, \tag{5.2b}$$

where δ is the vector of generator angles, ω is the vector of the speed deviation of the generators from synchronous speed, and $\mathcal{M} = \text{diag}(M_i)$ and $\mathcal{D} = \text{diag}(D_i)$ are $n \times n$ diagonal matrices of the generator inertias and damping factors, respectively. The DAE (5.2) can be converted to a system of pure differential equations by relating the algebraic variables V_i and θ_i to the system state variables (δ, ω, E) from (5.2b), and then substituting them back in (5.2a) via Kron reduction. The resulting system is a fully connected network of n third-order oscillators with excitation dynamics with $l \leq n(n-1)/2$ tie-lines. Let the internal voltage phasor of the ith machine be denoted as $\widetilde{E}_i = E_i\angle\delta_i$. The electromechanical dynamics of the ith generator in the Kron's form, neglecting line resistances, can be written as:

$$\dot{\delta}_i = \omega_i \tag{5.3a}$$

$$M_i\dot{\omega}_i = P_{m_i} - E_i^2 G_{L_i} - D_i\omega_i - \sum_{k=1, k\neq i}^{n} E_i E_k (G_{ik}\cos(\delta_i - \delta_k) - B_{ik}\sin(\delta_i - \delta_k)) \tag{5.3b}$$

$$\tau_i \dot{E}_i = -\frac{x_{d_i}}{x'_{d_i}} E_i + U_i + \frac{x_{d_i} - x'_{d_i}}{x'_{d_i}} \left(\sum_{k=1}^{n} K_{i,k} E_k \cos(\delta_i - \delta_k)\right) \tag{5.3c}$$

where $\omega_s = 120\pi$ (rad/sec) is the synchronous speed for the 60 Hz system, for $i = 1, \ldots, n$.

Linearizing (5.3) about the equilibrium $(\delta_{i0}, 0, E_{i0})$ results in the small signal model:

$$\begin{bmatrix} \Delta\dot{\delta}(t) \\ \mathcal{M}\Delta\dot{\omega}(t) \\ \tau\Delta\dot{E}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{d\dot{\delta}}{d\delta} & \frac{d\dot{\delta}}{d\omega} & \frac{d\dot{\delta}}{dE} \\ \frac{d\dot{\omega}}{d\delta} & \frac{d\dot{\omega}}{d\omega} & \frac{d\dot{\omega}}{dE} \\ \frac{d\dot{E}}{d\delta} & \frac{d\dot{E}}{d\omega} & \frac{d\dot{E}}{d\dot{E}} \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \Delta\delta(t) \\ \Delta\omega(t) \\ \Delta E(t) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}}_{\mathcal{B}} u \tag{5.4}$$

where, $\Delta\dot{\delta} = [\Delta\dot{\delta}_1 \cdots \Delta\dot{\delta}_n]^T$, $\Delta\dot{\omega} = [\Delta\dot{\omega}_1 \cdots \Delta\dot{\omega}_n]^T$, , $\Delta\dot{E} = [\Delta\dot{E}_1 \cdots \Delta\dot{E}_n]^T$, and I_n is the n-dimensional identity matrix. The matrix $\mathcal{A}$ in (5.4) is composed of partial differential equations taken from (5.3), the derivation of which are shown in.

5.3 Five-Area Model of the WECC System

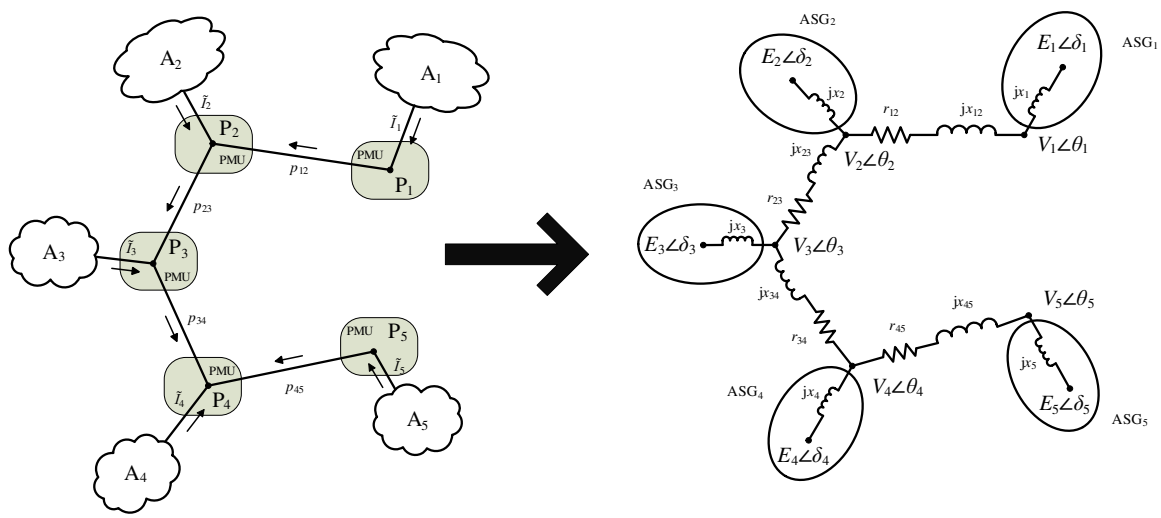


Figure 5.1: Electrical topology of WECC's 500kV network

Recently, a reduced-order five-machine dynamic equivalent model of the WECC has been developed by the authors to represent its interarea oscillation patterns [43]. The WECC system is divided into five separate areas which are connected in a linear topology through long 500 kV transmission lines. These five areas were represented by five aggregated synchronous generators (ASG) with the interconnecting 500 kV lines between any two areas can be reduced to a single equivalent transmission line between those two areas. This reduction is shown in Figure 5.1. All parameters, including intra and inter-area impedance, machine inertia, machine damping, and power flow were computed in [43] and then an RSCAD model was created using

these parameters to simulate WECC. Although the equivalent transmission lines are reduced versions of real-world transmission lines, they connect five real-world sub-stations, namely, Area 1: Colstrip in Montana, Area 2: Grand Coulee in Washington, Area 3: Malin in Northern California, Area 4: Vincent in Southern California, and Area 5: Palo Verde nuclear plant near Phoenix, Arizona. These sub-stations are referred to as pilot buses. They are selected from each area based on the following criteria and are shown in Figure 5.1:

-- The sub-station must have a PMU installed at it's location

-- All generators within that area lie behind this sub-station

The voltage phasor, $V_i\angle\theta_i$ is known at each pilot bus owing to availability of PMU data at that bus. Furthermore, the current $I_i\angle\alpha_i$ being injected at each pilot bus can be calculated from the difference in line currents flowing in and out of that pilot bus, which are known quantities from PMU data.

$$\tilde{I}_i = \tilde{I}_{ik} - \tilde{I}_{ji} \tag{5.5}$$

The pilot bus of a particular area also acts as the terminal bus for the aggregated synchronous generator which represents that area. Looking from the pilot bus into the area, this generator is modeled as a Thevenin voltage source with internal EMF $E_i\angle\delta_i$ and Thevenin reactance jx_i. Due to non-identifiability, it is not possible to model it as an impedance of $r_i + jx_i$. This is further elaborated upon in [47]. Each aggregated synchronous generator is modeled as a second order damped oscillator described by the swing equation (5.2a)-(5.2b). Since these ASGs are fictitious generators, their model parameters are not known, and need to be identified using PMU measurements of voltage and phase angles measured at the corresponding pilot bus. Thus, the parameter identification for the five-area model is a three step process:

-- Identification of long tie line impedance $r_{ij} + jx_{ij}$

-- Identification of Thevenin reactance jx_i of ASG

-- Identification of inertia M_i and damping D_i of ASG

Also, as indicated in Section 3, since the ASG is obtained by collapsing coherent areas of the network, the non-coherent modes or local modes must be removed from the PMU measurements before they can be used to identify the above three quantities for each ASG. In other words, the raw PMU data contains both fast local modes as well as slow inter-area modes, and must be passed through a band-pass filter to remove the faster modes. However, since such filtering typically adds a phase shift to the slow modes, distorting the data, an alternative time-domain

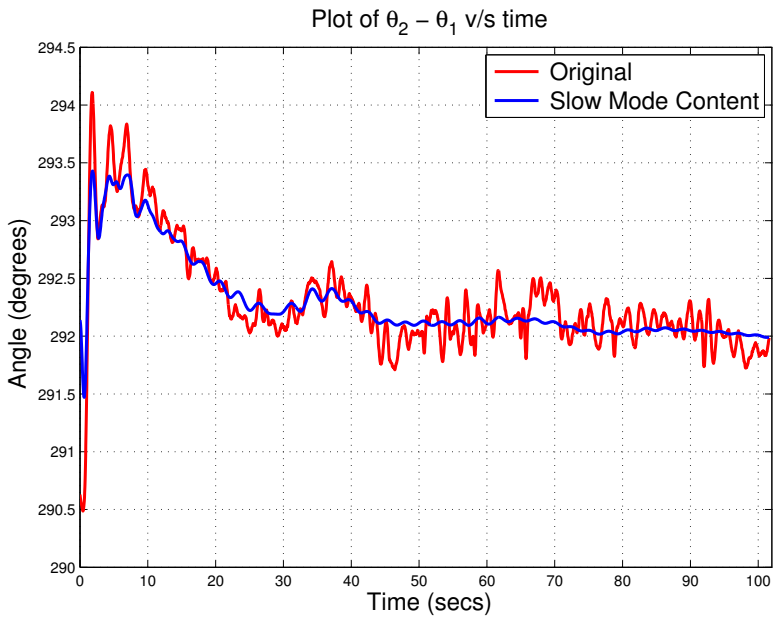


Figure 5.2: Slow mode component of Voltage angle difference between Station 2 and Station 1

approach of using modal decomposition method is applied, as described in Section 3. We next briefly describe the method adopted for this particular case, namely the Prony algorithm. Prony analysis, which is a time-domain based curve fitting technique, is used to determine frequency, amplitude, phase and damping components of the 'equivalent' PMU measurement from the pilot buses. Essentially, any stream of PMU data can be formulated as

$$y(t) = x(t) + n(t) \approx n(t) + \sum_{i=1}^{M} R_i e^{\lambda_i t} \tag{5.6}$$

where $y(t)$ is the observed time response
$x(t)$ is the reconstructed signal
$n(t)$ is noise
R_i is the complex amplitude of ith component
λ_i $= \sigma_i + j\omega_i$
$-\sigma_i$ is the damping coefficient of ith component
ω_i is the angular frequency of ith component

Prony's method returns a set of exponential and damped oscillatory components, which when combined create $x(t)$ that provides the best possible least-squares fit to $y(t)$. The least-squares fit is constrained by the value of M. The obtained list of components are then subject to the constraint, $\omega_i < 2\pi$ rad/s to obtain the slow modes. The voltage angle difference between pilot

bus 1 and 2, and its slow mode component obtained from the Prony algorithm are shown in Figure 5.2. We present a detailed description of the modal extraction results for the various event dates provided to us in the Appendix. For the sake of completeness, we compare the results of the Prony algorithm with that of Matrix Pencil method. In [47] the estimation process of the three essential model parameters listed above using the extracted slow mode components of the PMU measurements is outlined.

5.4 Design of LQR Controller

In face of both small-signal and large-signal disturbances, these five areas oscillate with respect of each other thereby giving rise to inter-area oscillation modes. For the reduced-order system, these modes will be reflected in the inter-machine oscillations as every area now is represented by a unique generator. To damp these oscillations, we next design a linear quadratic regulator (LQR) state feedback controller $u(t) = K\,x(t)$ using excitation voltage as the control signal. The controller designed was a simple feedback controller where $u = -Kx$ to transform the state-space from $\dot{x} = \mathcal{A}x + \mathcal{B}u$ into $\dot{x} = (\mathcal{A} - \mathcal{B}K)x$. Observing (5.4), $\mathcal{B}$ includes control inputs to just the excitation state variables E. Careful selection of the matrix K is also done such that the new system behaves more desirably, with more damped slow-mode oscillations. To calculate the matrix, K, use of a Linear Quadratic Regulator was implemented. An LQR choice for K will minimize:

$$\int_0^{\inf} (x^T Q x + u^T R u) dt \tag{5.7}$$

With LQR control design, R and Q can prioritize certain state variables over others, allowing favoritism of reducing deviation of certain states over others during transience. For the purposes here, $R = I_{15}$ such that all state variables are equally considered. Selection of Q, however, was a bit more challenging. x contains five state variables representing each angle δ of each machine i, five state variables representing each frequency δ for each machine i, and five state variables for each machine voltage E for each machine i, represented by (5.8).

$$x = \begin{bmatrix} \Delta\delta(t) \\ \Delta\omega(t) \\ \Delta E(t) \end{bmatrix} \tag{5.8}$$

This gives fifteen state variables total but angles are only relevant and useful in relation to one another and are never considered as absolute quantities as a control input. This is because power flow feedback is desired. For this reason, the product $x^T Q x$ must be composed entirely of relative phase angles to represent power flows. This product can be expanded, shown in (5.9).

$$x^T Qx = [\Delta\delta(t)|\Delta\omega(t)|\Delta E(t)] \begin{bmatrix} Q_\delta & 0 & 0 \\ 0 & Q_\omega & 0 \\ 0 & 0 & Q_E \end{bmatrix} \begin{bmatrix} \Delta\delta(t) \\ \Delta\omega(t) \\ \Delta E(t) \end{bmatrix} \tag{5.9}$$

Note only the submatrix Q_δ, must be altered. Q_ω and Q_E remain I_{5x5}. To ensure the product only takes relative phase angles into account, the following matrix was used for Q_δ:

$$Q_\delta = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{5.10}$$

The computation of K minimizing (5.7) can now be done with Q and R, but another hurdle must be also dealt with. The resulting feedback matrix for the LQR controller contains feedback constants for each state variable, including the absolute phase angle from each bus. In reality, again for the same reason as explained above, relative angles must be used for feedback. If the product Kx is observed and only terms with δ state variables are considered, the following series of feedback terms is observed in (5.11).

$$u_i = \sum_{j=1}^{5} K_{ij}\delta j + \sum_{j=1}^{5} K_{i,j+5}\omega j + \sum_{j=1}^{5} K_{i,j+10} Ej \tag{5.11}$$

Let's now assume we only observe the first feedback variable, u_1.

$$u_1 = K_{11}\delta_1 + K_{12}\delta_2 + K_{13}\delta_3 + K_{14}\delta_4 + K_{15}\delta_5 \tag{5.12}$$

(5.12) can be rewritten as:

$$u_1 = M_1(\delta_1 - \delta_2) + M_2(\delta_2 - \delta_3) + M_3(\delta_3 - \delta_4) + M_4(\delta_4 - \delta_5) \tag{5.13}$$

If we define:

$$K_{11} = M_1 \tag{5.14a}$$
$$K_{12} = M_2 - M_1 \tag{5.14b}$$
$$K_{13} = M_3 - M_2 \tag{5.14c}$$
$$K_{14} = M_4 - M_3 \tag{5.14d}$$
$$K_{15} = -M_4 \tag{5.14e}$$

Rearranging (5.14,

$$M_1 = K_{11} \tag{5.15a}$$
$$M_2 = K_{12} + K_{11} \tag{5.15b}$$
$$M_3 = K_{13} + K_{12} + K_{11} \tag{5.15c}$$
$$M_4 = K_{14} + K_{13} + K_{12} + K_{11} = -K_{15} \tag{5.15d}$$

(5.13) is only true if (5.15) is true, meaning:

$$\sum_{j=1}^{5} K_{1j} = 0 \tag{5.16}$$

Of course (5.16) will not be true. It is possible, however, to force this condition by changing the K matrix the LQR algorithm has given. The alteration of K is given in (5.17).

$$K_{ij}^{'} = K_{ij} - \frac{1}{5}\sum_{j=1}^{5} K_{ij} \tag{5.17}$$

The reader may be wondering, however, how altering the feedback matrix K has changed performance of the system. A very brief test was conducted in MATLAB to observe if controller performance would degrade, and no noticeable alteration in impulse response was observed by altering K to meet the criteria of (5.16).

Now that a proper controller has been designed, it would be of interest to describe the network that will be used to implement this controller.

5.5 Experimental Results of RTDS-WAMS testbed

5.5.1 Unstable System without Wide-Area Control

Table 5.1: Alteration of Load Reference Sliders, in per-unit

	Gen 1	Gen 2	Gen 3	Gen 4	Gen 5
Before	0.4926	0.8646	0.4687	0.4722	0.5856
After	0.4926	0.8646	0.3687	0.4722	0.6856

First, the steady-state condition of the model is altered by adjusting the load reference sliders on the aggregate machines, thus altering power flow throughout the system. The new load reference values are shown in Table 5.1. These references were found through a series of simulations in RSCAD, and adjusting the sliders accordingly until the system went unstable. The new steady-state operating point was observed by running RSCAD without the LQR controller, and K was recomputed and reentered accordingly.

5.5.2 Controller Comparison Tests

A comparison of the transient response of WECC was conducted with both the LQR feedback controller on and off. Note that the controller is implemented directly in RSCAD for this test, not in ExoGENI. A three phase fault of duration four cycles was applied to Area 3, and the resulting phase angle differences between each aggregate machine are recorded. First, only local frequency feedback using power system stabilizers (PSS) is actuated for every machine, without the wide-area controller being ON. The plots of the phase angle differences between the pilot buses are shown in Figure 5.3. It can be seen that local PSS feedback does guarantee the closed-loop system to be stable, but the dynamic performances are still notably degraded.

Next, the wide-area controller is switched on together with the PSSs. The resulting transient responses of the angle differences are shown in Figure 5.4. The system performance improves drastically in this case.

5.6 Conclusion

In this chapter, we recall the general ideas on how a multi-area power system model can be reduced to its dynamic equivalent via time-scale separation. Based on the reduced model of the WECC system, we proposed the distributed LQR control algorithm for the wide-area damping

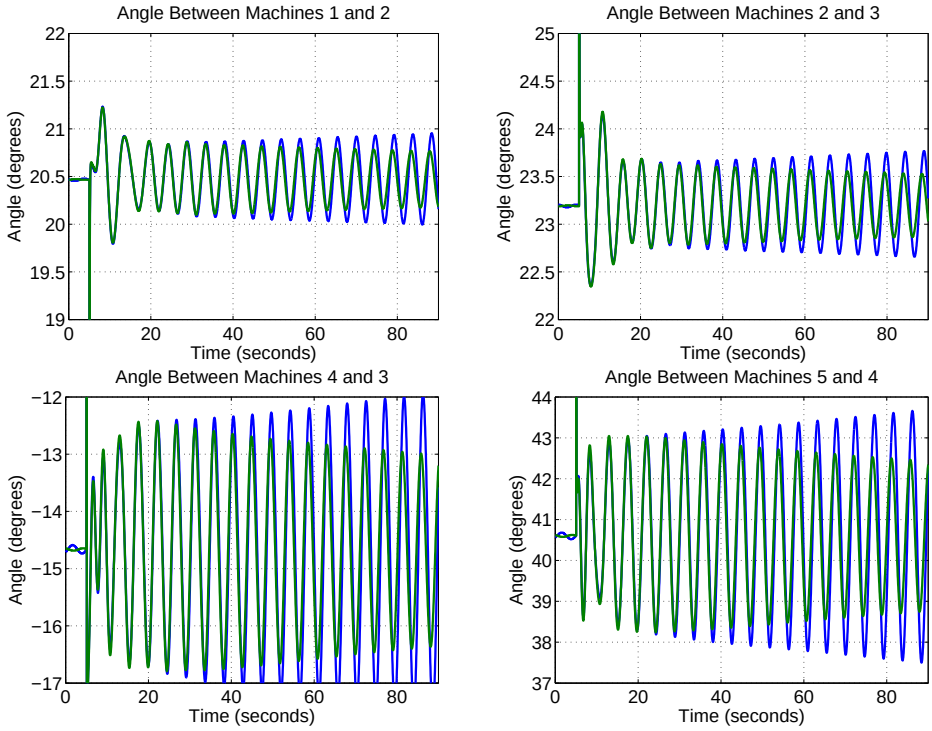


Figure 5.3: WECC Transient Response, Local Frequency Feedback with no WAC

control of the large power system. By the comparing between the local PSS controller and wide-area controller through simulation, the later controller has better performance.

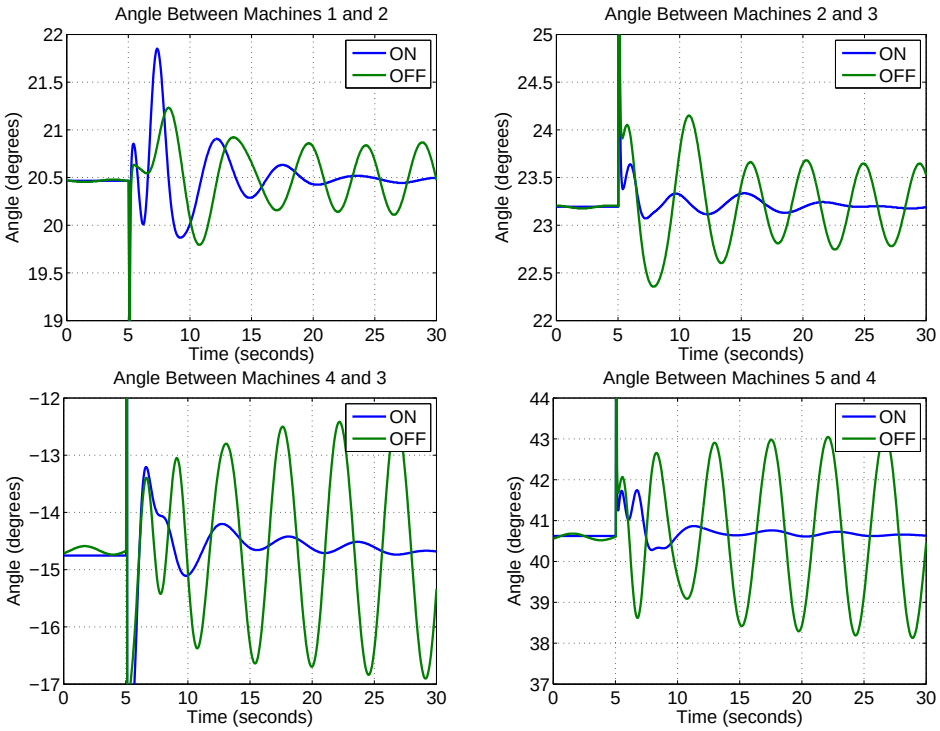


Figure 5.4: WECC Transient Response With and Without WAC

Chapter 6

Validation on ExoGENI-WAMS Testbed

In this Chapter we validate all cyber-physical algorithms on the typical cyber-physical testbed called ExoGENI-WAMS testbed. In current state-of-art, it is contingent to use PMU data for research purposes on accessing the real data from different utility companies who own the PMUs at the locations of interest. Gaining access to such data may not always be an easy task due to privacy and non-disclosure issues. More importantly, in many circumstances even if real PMU data are obtained they may not be sufficient for studying the operation and control of the large-scale power system because of their limited coverage. To circumvent this problem, over the past two year our research group has developed a Hardware-In-Loop(HIL) simulation framework where high fidelity component models of large power systems can be simulated in Real-time Digital Simulators (RTDS), and the dynamic responses can be captured via real hardware Phasor Measurement Units from multiple vendors including Schweitzer Engineering Lab and ABB Inc. This RTDS-PMU testbed has recently been federated with the US-wide ExoGENI network through a state-funded, metro-scale, multi-layered dynamic 100gbps optical network testbed called Breakable Experimental Network (BEN). The resulting testbed is referred to as the ExoGENI-WAMS testbed. Such prototype testbed could be used to not only collect massive volumes of power system data from thousands of PMUs deployed across a large power grid, but also shave and analyze that data using next-generation cyber-infrastructure technologies that including high-speed virtual networking, high performance cloud computing, and virtualization and data management. Thus, we next focus to describe the details of the ExoGEN-WAMS testbed that including the architecture, components, connection, and operational procedures, then to show how it is used for testing the distributed cyber-physical algorithms proposed in Chapter 2, 3, 4 and 5 to study the typical cyber-physical challenges (e.g. end-to-end delay and attack-resiliency) when they are practically implemented in the real wide-area communication

network.

6.1 ExoGENI-WAMS Testbed

ExoGENI-WAMS testbed is a distributed cloud computing cyber-physical testbed, here, the physical layer consists of the hardware-in-loop RTDS and the PMU-based Wide Area Measurement System (WAMS), while the cyber layer consists of BEN and ExoGENI, where hundreds of virtual phasor data concentrators (v-PDC) receive real-time PMU data streaming in from the physical layer, run local monitoring and control algorithms, and communicate with each other to execute a system-wide control action in a completely distributed way. Figure 6.1 shows the architecture of ExoGENI-WAMS testbed and its components. The physical layer developed in the RTDS lab of the FREEDM System Center of NC State uses real-time synchronized phasor measurements, streamed from PMUs and time synchronized by GPS, to precisely measure currents and voltages and capture detailed oscillations of the power flows in different parts of the grid, which has been simulated in the RTDS testbed. That data can be critical for analyzing distributed such as blackouts and can alert engineers to risks of blackouts and other problems so that appropriate control actions can be taken in time. The architecture and components in RTDS and WAMS have been described in [22, 23]. In this section, we mainly introduce the cyber layer of ExoGENI and BEN in details.

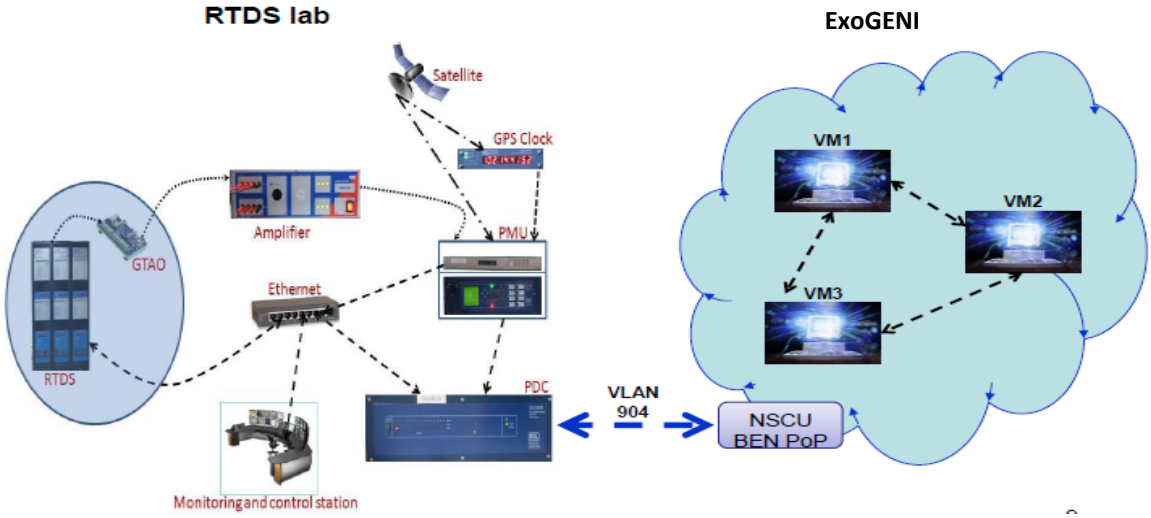


Figure 6.1: ExoGENI-WAMS Testbed Physical Architecture

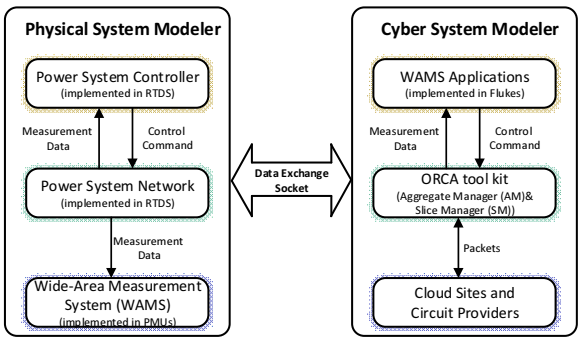


Figure 6.2: Concept Architecture of the ExoGENI-WAMS Testbed

6.1.1 Cloud-in-Loop Testbed Architecture

The architecture of the hardware-in-loop power system testbed with the cloud platform in the feedback, also referred to as the ExoGENI-WAMS cyber-physical testbed, for testing our wide-area LQR control is shown in Figure 6.2. The data exchange between cyber and physical layers is time-critical, and needs be done in real-time at every sampling instant. The various blocks in Figure 6.2 are described as follows:

1) **Physical System Modeler**: The Physical System Modeler consists of the WECC power system model, its associated controllers, and hardware PMUs with phasor data concentrators (PDC) and GPS clocks. The power system model implemented in RTDS includes all the modeling components that represent a physical power system, such as generators, transformers, lines, loads and wind turbines. The controller (e.g. exciter, governor) collects the measurement data from the power system model and sends back control commands. The GTNET card, GTAO card, and PMUs enable the RTDS to communicate with the cloud through ethernet or analogue/digital I/O ports.

2) **Cyber System Modeler**: The Cyber system modeler refers to the ExoGENI networked cloud computing platform, which interacts with the physical system modeler by emulating the reconfigurable virtual communication and computation network. The ExoGENI platform consists of 5 circuit providers and 14 cloud sites at the physical layer, and the corresponding Open Resource Control Architecture (ORCA) control software including Aggregate Manager (AM) and Slice Manager (SM) at the software layer. The ORCA AM configured at each site, i.e., either cloud site or circuit provider, comprises of a cloud handler plug-in to invoke the cloud service and an image proxy server to obtain VM's images through the URL address. A slice refers to an experiment or application run in ExoGENI. The ORCA SM is used to create, maintain, and delete an experiment or slice. The ExoGENI sites and control software enable the software-defined networking using OpenFlow. ExoGENI thus offers a powerful unified hosting platform for deeply networked, data-intensive cloud computing applications such as

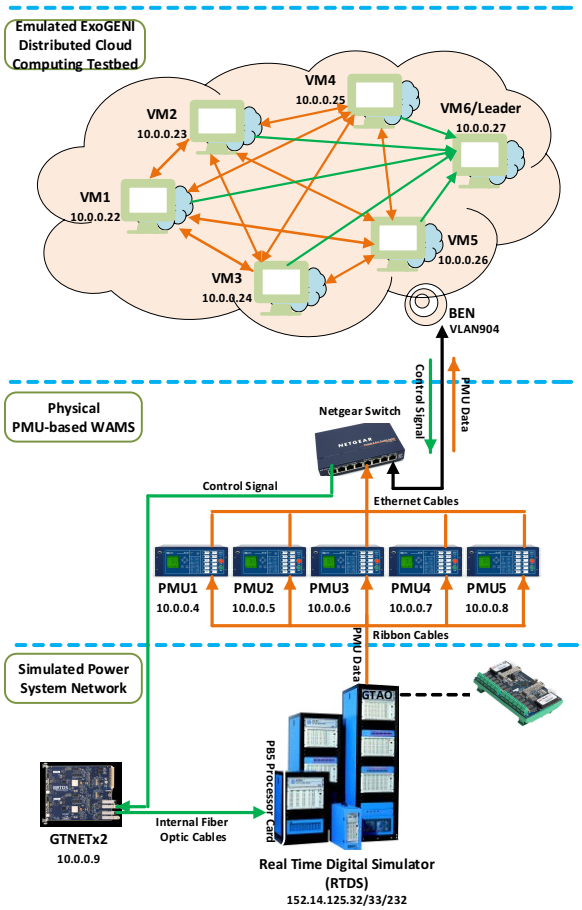


Figure 6.3: Setup of the ExoGENI-WAMS Testbed

distributed WAMS applications. Flukes, a web-start GUI application, is used to run these WAMS applications on ExoGENI.

3) **Data Exchange Socket** Data exchange socket enables PMU data to be exchanged between the cyber system modeler and the physical system modeler. The analogue and the digital signals generated by the GTAO card are converted into packets that are streamed by the hardware PMUs, and further transferred into the cyber system modeler. Packets from the cyber system modeler are formatted and converted to analogue or digital signals through the GTNET card of RTDS before using by the physical system modeler. The data exchange socket runs in real-time, thereby introducing minimum communication delays.

6.1.2 ExoGENI - Networked Cloud Computing Platform

ExoGENI is a new testbed at the intersection of networking and cloud computing, funded through NSF's Global Environment for Network Innovation (GENI) project. The ExoGENI

testbed combines computation, storage, and network capabilities with two advances in virtual infrastructure services outside of GENI: open cloud computing and dynamic circuit fabrics. It is, in effect, a widely distributed networked infrastructures-as-a-service (NIaaS) platform heard towards experimentation and computational tasks by employing sophisticated topology embedding algorithms. Consisting of could site "racks" on host campus within the US, linked with national research networks through programmable exchange point, It enables real-world deployment of innovative distributed services and new version of a Future Internet, as well as new version of smart grids. The ExoGENI testbed is a national testbed of networked cloud resources to support GENI experiments, and the available GENI rack deployment locations is shown in Figure6.4.

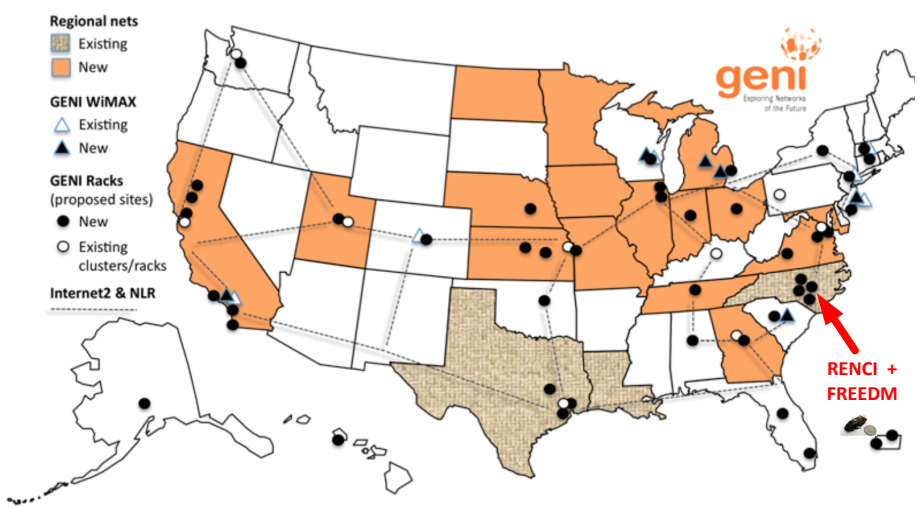


Figure 6.4: Available Rack Locations in GENI

ExoGENI allows users to create custom topologies using resources from multiple federated providers via a control and management software called the Open Resource Control Architecture (ORCA) to orchestrate the networked cloud resource provisioning. We argue that the current design practice based on the centralized servers and IP-based Internet architecture is not an economical and efficient solution to satisfy the real-time requirement of processing large volume of Synchrophasor data. We envision a IaaS based solution would be more suitable. In fact, ExoGENI service allows dynamic provisioning of virtual machines of different CPU and memory capacities with customized software images. With this capability, the WAMS communication network can automatically request for the right virtual machine to run the

best real-time algorithm, for example, for the least-squares based oscillation mode estimation problem, among other monitoring, state estimation and control problems. With this capability, one can experiment various WAMS applications under real-time constraints, and answer critical questions such as: where to deploy the computing facilities, how to design better communication topologies, and what data transport protocols to use for more efficient control actions.

6.1.3 Optical Communication Network - BEN

The Layer2 Breakable Experimental Network (BEN) is the primary platform for experimental network research. It consists of several segments of NCNI dark fiber, a time-shared resource that is available to the Triangle research community. BEN is a research facility created for researchers and scientists in order to promote scientific discovery by providing the Universities with world-class infrastructure and resources for experimentation with disruptive technologies. BEN has 10 Gbps linkages to the Internet2 network which allows us to set up a Software Defined Network (SDN), referred to as ExoGENI, with dynamic high-speed connections to other universities in the nation, as shown in Figure 6.4 [44]. We are currently using ExoGENI to access computing resources via Layer 2 networks. BEN connects distributed cloud resources in local universities, with four PoPs located at RENCI, UNC Chapel Hill, NC State Centennial campus, and Duke University, using CISCO and Juniper routers on top of WDM/TDM bandwidth virtualization technology from Infinera. These allow us to set up dynamic multi-layer connections of up to 10 Gbps between the sites.

Figure 6.5 shows how the RTDS lab of NC State including RTDS and PMU-based WAMS is connected with ExoGENI testbed through BEN using VLAN technology. VLAN 11 is created to embrace all ExoGENI racks and connected to BEN through Juniper QFX3500 switch. Similarly, VLAN 904 serves for the RTDS lab at the NC State site, and it is connected to BEN through Cisco 3560. Then a network rule has been manually created to translates traffic from VLAN 11 of ExoGENI to VLAN 904 of RTDS lab at BEN PoP located at the RENCI site. The necessary software named IEEE C37.118 Message Reader for the Linux platform has been developed to extract streaming PMU data from PMUs and PDCs at the RTDS lab to the ExoGENI testbed.

It is worth to note that we have two important steps to run the experiment in the ExoGENI testbed: 1) Create the network topology for the experiment through the ORCA network editor named Flukes, which is a Java-based graphical tool developed specifically for the ExoGENI testbed that allows an experimenter to inspect the state of ORCA substrates, create request topologies, submit requests to ORCA and inspect the returned substrate information. 2) Create the customized image of operation system for each virtual machine. The detailed procedures for these two steps are described online [35].

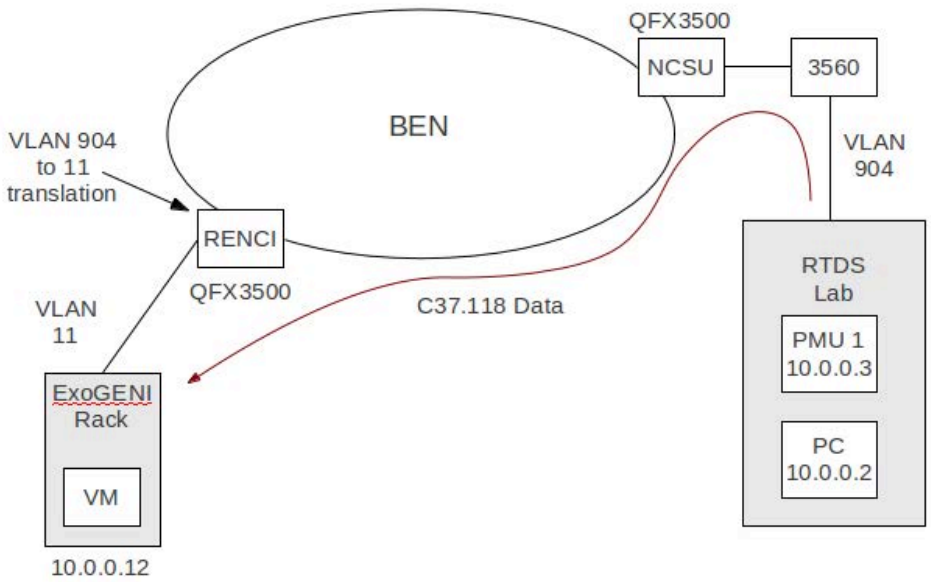


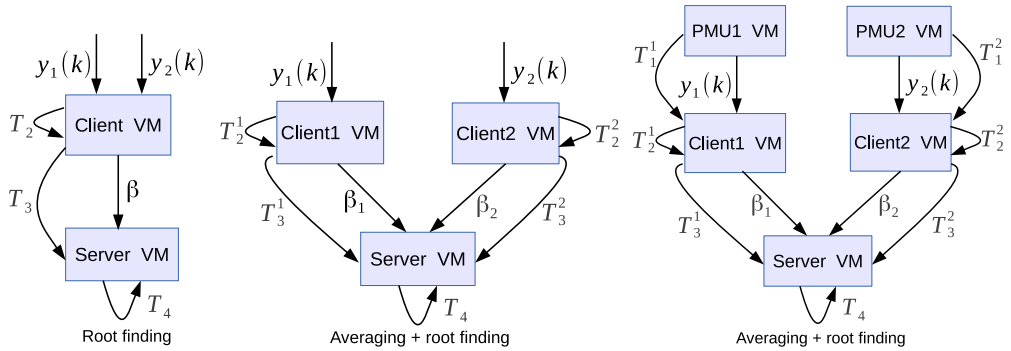
Figure 6.5: Communication Architecture of WAMS-ExoGENI testbed through BEN

6.2 Delay Evaluation

In this section, we use the ExoGENI-WAMS testbed to address one of the typical cyber-physical challenges - *end-to-end delay evaluation*, in implementing eigenvalue estimation algorithms for large power system networks using distributed processing of Synchrophasor measurements. We first design three experimental network topologies to investigate the end-to-end delay of the centralized (CLS), decentralized (DLS), and recursive (RLS) least-squares estimation, next the implementation process of both DLS and RLS is followed, finally introduce the experiment setup and analyze results.

6.2.1 Design of Experimental Topology

To test the least-squares-based estimation algorithms described in Chapter 2, and compare their accuracy and end-to-end delay including communication time and computation time, we design three experimental network topologies. The first topology is for the Centralized Least-Squares (CLS) algorithm, shown in Figure 6.6(a). It consists of one client Virtual Machine (VM), being charge of generating the simulated PMU measured data $y_1(k)$ and $y_2(k)$, executing the centralized least-squares algorithm, and sending out the estimated parameter vector β, and one server VM, which is responsible for receiving the estimates and solving for the roots of the continuous-time transfer function of the power system. This can, for example, be done



(a) Centralized Least-Squares (b) Decentralized Least-Squares (c) Decenralized LS vs Recursive LS

Figure 6.6: Experimental Network Topologies.

by first estimating the roots of the discrete-time polynomial and converting them to their continuous-time counterparts by the relationship $s = ln(z)/T$ where T is the sampling time. Table 6.1 shows each component of the end-to-end delay. As obvious from the formulation, the averaging step for β is not included in the case of centralized LS algorithm. The second topology for the Decentralized Least-Squares (DLS) algorithm is shown in Figure 6.6(b). Compared with first one, the decentralized topology employs two client VMs to generate simulated PMU data $y_1(k)$ or $y_2(k)$, run the standard least-squares algorithm and send the estimate β_1 or β_2 to server VM, respectively. The server VM averages the respective coefficients contained in β_1 or β_2, constructs the global characteristic equation, solves the root finding problem, and converts the discrete-time poles to continuous-time poles. For both first and second topologies, the end-to-end delay consists of: (a) T_2, which is the computation time of least-squares algorithm; (b) T_3 which is the communication time between the client VM and the central server; and (c) T_4, which is the computation time for root-finding with or without averaging the elements of β.

Table 6.1: Components of End-to-End Delay

Component	Description
T_1	PMU source data communication time
T_2	Least-squares algorithm computation time
T_3	Result communication time
T_4	Computation time for root finding and averaging β

To further verify the performance of end-to-end day between DLS and RLS algorithms, the third topology shown in Figure 6.6(c), is designed by adding two PMU VMs, which generate

simulated PMU source data $y_1(k)$ and $y_2(k)$, respectively. Thus, each client VM needs to stream the PMU source data from the corresponding PMU VM, except for the execution of DLS or RLS algorithm and communication of the estimated sets. In this case, the end-to-end delay contains all four components shown in Table 6.1. The step-by-step architectural diagram for executing the different steps of computing these delays is shown in Figure (6.7). The flowchart of implementation for third topology is shown in Appendix A.3.

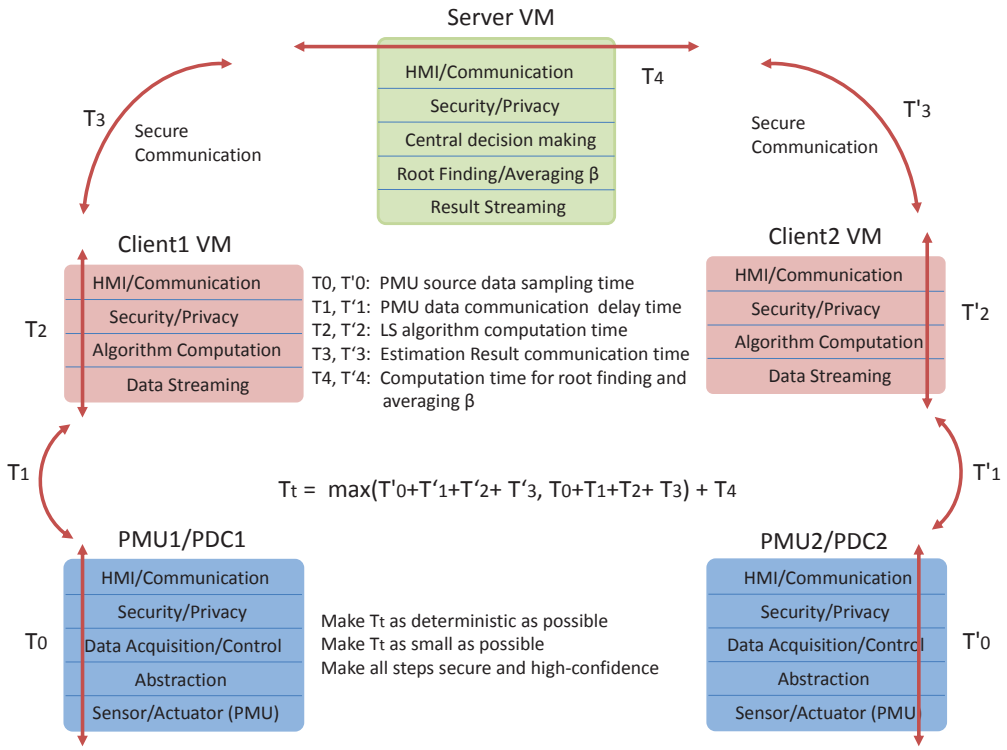


Figure 6.7: Step-by-Step Delay Evaluation

6.2.2 Experimental Setup

We carry out two experiments to verify the performance of CLS, DLS and RLS. Experiment I is to investigate the performance of standard least-squares algorithm in centralized and decentralized fashions. Experiment II is to compare the performance of DLS and RLS algorithms. In both experiments, we choose a 4th-order power system model, consisting of two pairs of complex conjugate poles. The system is a two-machine power system with a single tie-line connecting the two, excited by an impulsive input disturbance at machine 1, with two PMUs measuring their terminal bus frequencies. These measured outputs are sampled once every 0.01 second. In this

case study, the parameter averaging and root-finding are also implemented in the client VM. Thus, T_3 component in this section means total computation time of algorithm, plus averaging (if needed) and root finding.

6.2.3 Analysis of Results

Accuracy: The final estimates of the coefficients of the characteristic polynomial for the three algorithms are shown in Table 6.2. It can be seen that DLS with averaging preserves the accuracy of the CLS solution, which may be treated as the *ideal* solution. RLS, however, loses accuracy from the ideal solution to certain extent, mainly due to wrong guesses for the initial guess. More educated guesses for the initial estimate will improve the accuracy of RLS.

Table 6.2: Accuracy of CLS, DLS, RLS estimates

Algorithm	b_1	b_2	b_3	b_4
CLS	-2.2272	0.7047	1.2757	-0.7533
DLS	-2.2273	0.7049	1.2756	-0.7532
RLS	-2.4171	0.7870	1.4393	-0.8136

End-to-End Delay: Five different scenarios are implemented to investigate the performance of the end-to-end delay. Table 6.3 shows the result of Experiment I for comparing the end-to-end delay between CLS and DLS. Note that the RENCI rack is located in the triangle area of North Carolina while the UvA rack is at Netherlands. Also, each PMU source data file has 1001 samples. Two conclusions can be drawn from Experiment I: 1) Scenario 1 shows that the end-to-end delay of DLS decreases to about 54 ms from 147 ms of CLS, clearly pointing that DLS saves computational time T_2. 2) When the communication time is dominated with the long distance between the VMs in Scenario 2&3, the advantage of DLS is not as much impressive, but still clear. The result of Experiment II is shown in Table 6.4. In this case, each PMU source data file only has 200 samples. From Scenario 1&2, the end-to-end delay of RLS is only slightly less that of CLS, because the iteration number of VM1 is 144, while of VM2 2 is 76, which means that depending on the initial guess the RLS algorithm may take significant number of iterations to converge. Another point worth noticing is that the end-to-end delay is also determined by the type of PMU source data, meaning that if phase angle is used instead of frequency then the algorithm may converge faster, which clearly follows from the participation factor of the modes in the outputs.

Table 6.3: End-to-End Delay of Experiment I: CLS vs DLS

Algorithm	$T_2(us)$	$T_3(us)$	Total (us)
Scenario 1: 3 VMs at RENCI rack			
CLS	134,466	13,054	147,520
DLS	22,088	19,763	54,150
Scenario 2: 2 Clients at RENCI, Server at UvA			
CLS	169,301	3,178,939	3,348,240
DLS	23,752	3,187,137	3,229,170
Scenario 3: Client1 at RENCI, Client2 at Houston, Server at UvA			
CLS	179,913	3,267,583	3,447,497
DLS	26,079	3,191,082	3,274,337

Table 6.4: End-to-End Delay of Experiment II: DLS vs RLS

Algorithm	$T_1 + T_2(us)$	$T_3(us)$	Total (us)
Scenario 1: PMU1 at Boston, Client1 at RENCI, Server at Houston			
DLS	56,417	65,975	122,392
RLS with pmu1	53,517	59,858	113,375
RLS with pmu2	47,665	60,376	108,041
Scenario 2: 2 PMUs at Boston, 2 Client at RENCI, Server at Houston			
DLS	55,974	60,765	121,608
RLS	52,854	61,066	114,924

6.3 Distributed Storage System

6.3.1 Design of Distributed Storage System

The real-time distributed storage system is the core component of resilient wide-area measurement systems. As shown in Fig. 6.8, the storage system acts as a middleware between local PDCs and the central PDC to cache the estimation data.

Externally, our storage system provides two fundamental cloud services: *get* and *put*. PDCs and WAMS applications utilize the *put* service to store monitoring data or intermediate power grid estimation results. They can further use the *get* service to obtain relevant data from the storage system. Thus, this distributed storage system provides an additional protocol layer between data providers (PMUs/PDCs) and data consumers (PDCs and WAMS applications).

Internally, we utilize a distributed hash table (DHT) over a set of storage nodes (chosen to also be the PDCs) so that the power grid data can be disseminated in a distributed manner and subsequently accessed by monitoring and control applications. Fig. 6.9 illustrates how the DHT protocol organizes PDCs as storage nodes in a virtual Chord-like ring structure [**?**]. This ring structure provides a natural way to orchestrate power estimates and, optionally, actuation

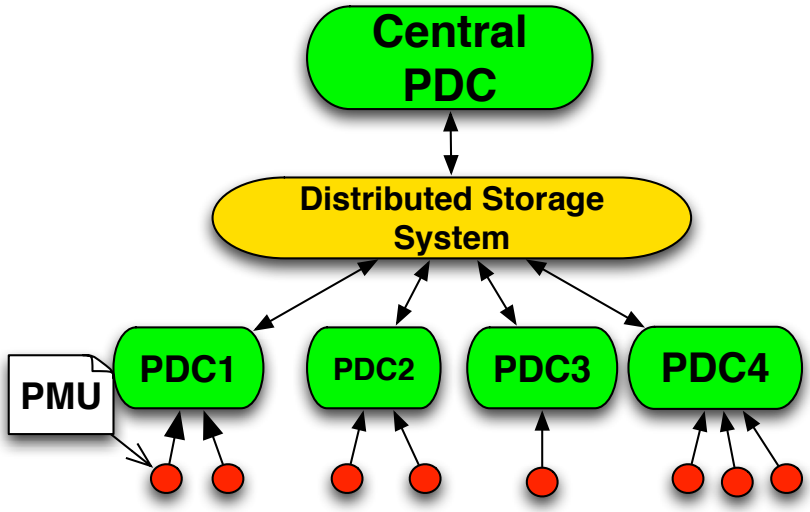


Figure 6.8: Measurement System with Real-time Storage System

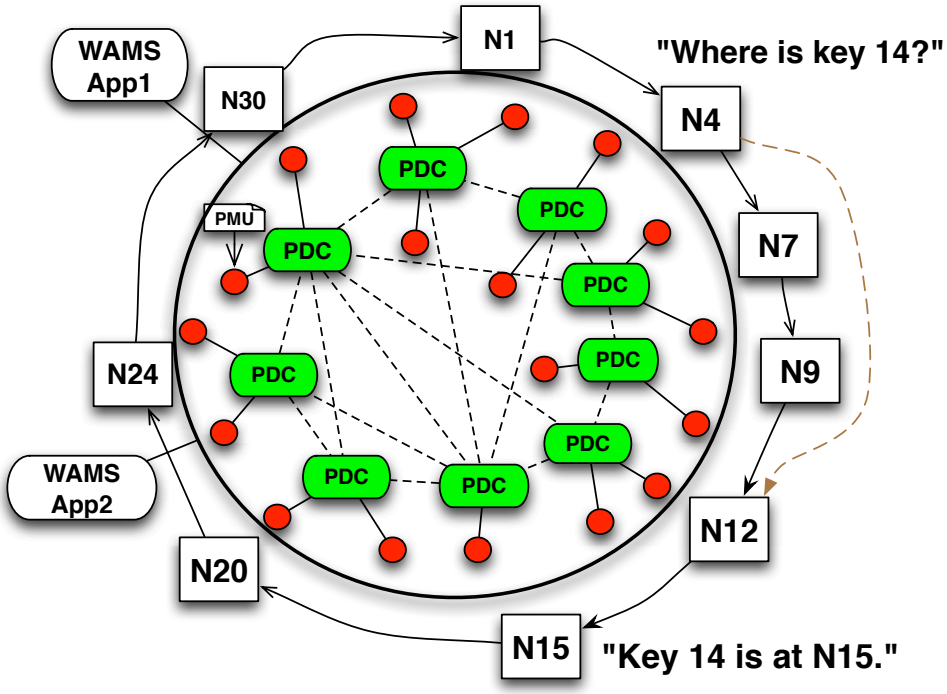


Figure 6.9: PDC, PMU, and Chord Ring Mapping Example

tasks of disjoint PDCs based on key/value pairs. The storage system can store raw PMU data, memorize state and parameter estimates, and even actuation intentions. In Fig. 6.9, PMUs, PDCs, and WAMS applications logically access our distributed storage system and coordinate actions with one another. In this example, 9 PDCs are mapped onto the Chord ring and utilized as DHT storage nodes. A PMU sends raw monitoring data to the storage system via a connection

with its local PDC. However, the raw PMU data are not necessarily stored in the local PDC. Our storage system utilizes a consistent hashing algorithm [**?**] to map data to virtual nodes. For example, the data with key 14 is located on virtual node 15 according to the Chord algorithm. The WAMS application may send requests to any PDC in the ring to fetch the data on demand. It is sufficient to locate any data by maintaining the nodes in a wrap-around circular list such that each node has a reference to its successor node, i.e., a ring traversal can always locate the data. However, this linear search algorithm is not scalable with increasing numbers of PDCs. Chord utilizes finger tables to reduce the number of intermediate nodes to $log(N)$ level, where N denotes the number of DHT nodes (see our previous work [**?**] for details).

This storage system improves the stability of the overall system in three ways. First, since the data are disseminated in a distributed manner on these storage nodes, any network link failure between a PDC and a storage node will not result in the data loss that a centralized measurement system experiences. Instead, the PDC chooses an alternative node to put/get data. We further adopt a real-time task scheduler on storage nodes so that the response time of data transmission is predictable. Third, since these storage nodes run autonomously and create replicas of data in different storage nodes, a single storage node failure will not result in data loss.

Data security needs to be considered in the storage system since the data are disseminated among different storage nodes. Since the *put* and *get* provided by our storage system are general cloud services without any requirement for the format of the data content, PDCs can easily integrate a public-key cryptography with the storage system to secure data. For example, PDCs can utilize RSA, an asymmetric encryption algorithm, to encode data before transmitting it to storage nodes. Then only the central PDC, which has the corresponding private key, can decode the data.

One important part of our distributed storage system is to provide QoS for the response time of any data access. To this end, we adopt a hybrid EDF scheduler so that urgent data requests are served at higher priority, i.e., ahead of lower priority requests that were issued earlier but have not yet been processed. This hybrid EDF scheduler includes two components: the EDF task scheduler, which schedules data requests in EDF order, and the EDF packet scheduler, which transmits IP packets that carry data request messages in EDF order. As a result, the deadlines carried in data messages are considered at the application layer as well as the network layer of the storage abstraction.

We have extended the Linux kernel to implement this hybrid EDF scheduler. Since the Linux traffic control layer does not support task deadlines embedded in data messages, which are encapsulated by the application layer, we extended the data structure for IP packets in the Linux network stack by adding new fields to store timestamps. We also extended the *setsockopt* system call so that it supplies these timestamps upon request. The most significant changes to

Linux to support this functionality are as follows:

(1) We added a new field in the kernel data structure to store the deadline of a socket transmission.

(2) We extended the kernel function *sock_setsockopt* with option $SO\_DEADLINE$, so that applications can specify message deadlines associated with messages via *setsockopt* in user mode.

(3) When the application transmits a message, the kernel stores the message data including the *deadline* of the socket. After this, the deadline of the message is passed down to the transport layer.

(4) We implemented an EDF packet scheduler, which provides the standard interface of Linux traffic control queue disciplines. The EDF packet scheduler utilizes a prioritized queue to maintain messages in a min-heap data structure as a linked list.

These novel extensions provide the capability of specifying message deadlines for real-time tasks (applications). With these provision, our EDF packet scheduler utilizes the message deadlines to transmit packets in EDF order.

6.3.2 Prony and ADMM using a Distributed Storage System

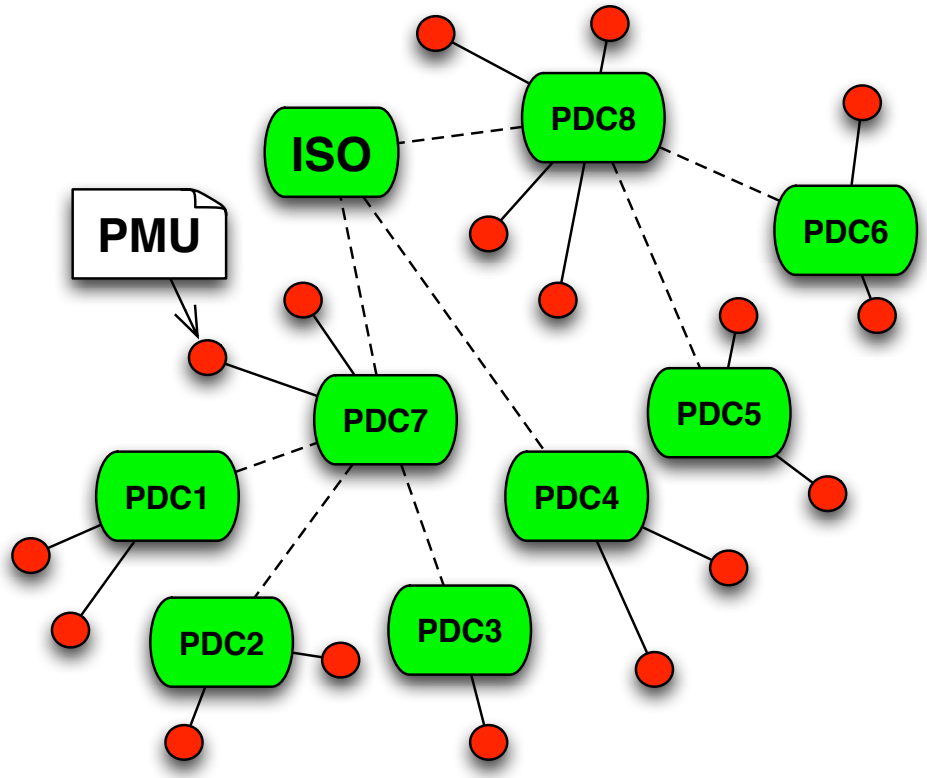


Figure 6.10: PDC Tree Topology Example

The Prony and ADMM algorithms described in Chapter 2 do not tolerate network failures. For example, estimation results in the region monitored by $PDC7$ in Fig. 6.10, which includes sub-regions monitored by $PDC1$, $PDC2$, and $PDC3$, could not be sent to the central PDC (C-PDC) at the ISO if the network channel between $PDC7$ and C-PDC fails. The coupling of PMUs and PDCs in a tree topology reduces its resilience. Thus, we enhance the algorithm by utilizing our distributed storage system as a middleware to transmit data. Fig. 6.9 illustrates the new architecture, where PDCs are used as storage nodes and organized as a Chord ring to construct a cloud storage service.

This distributed storage system decouples the strong dependency between PDCs in the tree topology. In the new architecture, each PDC maintains the IP addresses of a list of storage nodes. In case of a network channel failure, another storage node in the list can be picked as the gateway for requests to the cloud storage service. Since data in the storage system are distributed among storage nodes and redundant data are stored in different nodes, the storage system can increase the resilience of the Prony and ADMM algorithms.

In this scenario, the different steps of the ADMM algorithm are modified as follows: 1) PMUs send raw data to their local PDCs. 2) Local PDCs update the local coefficients $\mathbf{b}_i$ and send the estimation results to the storage system. 3) The C-PDC gathers the values $\mathbf{b}_i$ from the storage and sends their mean $\bar{\mathbf{b}}$ back to the storage. The C-PDC sets up a delay threshold (i.e., a deadline for data requests) when gathering the $\mathbf{b_i}$'s of all local PDCs. If the deadline expires before the C-PDC gets a $\mathbf{b_i}$ of a particular local PDC from the storage system, the previous $\mathbf{b_i}$ of that local PDC is used to calculate the mean $\bar{\mathbf{b}}$. (4) The mean coefficients $\bar{\mathbf{b}}$ are fetched by local PDCs to engage in the next iteration of coefficient estimations. We update the convergence condition in the centralized system to ignore the impact of using previous data, which always has a difference of zero. Equation 6.1 depicts the convergence condition. $\bar{\mathbf{b}'}$ is the average coefficient estimate of the previous iteration. Z is the number of new local PDC data that are used to calculate $\bar{\mathbf{b}}$. δ is the convergence threshold.

$$\frac{1}{Z}||\bar{\mathbf{b}} - \bar{\mathbf{b}'}||^{\mathbf{2}} \leq \delta \tag{6.1}$$

In addition, if the C-PDC fails, any one of the other PDCs can turn into a C-PDC to perform the supervisory step to guarantee convergence (as long as all PDCs agree on the same new C-PDC, which is known as the leader-election problem in Computer Science).

6.3.3 Evaluation

In our experiment, 4 nodes simulate local PDCs and 1 node simulates the central PDC. Each local PDC is associated with one PMU, which provides real-time data (sample rate 60Hz). This section provides the result of simulations running on Linux on bare metal machines.

In the direct channel mode, these five PDCs communicate directly via TCP (described in

Section **??**). In contrast, in the distributed storage mode, our storage system utilizes the PDC nodes as the storage nodes to provide the cloud storage service (see Section III-B).

We inject two types of network failures. For the first type, we inject intermittent failures by probabilistically dropping the data on the TCP channel between the PDCs and the C-PDC for the direct mode and between the PDCs and the access points of the storage system for the distributed mode. For the second type, we inject long-term link failures by dropping the data on the link after certain iterations. In the direct link mode, the C-PDC has to use the historical data to calculate the new coefficients while in the storage mode, an alternative access point to the storage system provides the correct data. For distributed storage, data becomes unavailable only if all channels between a PDC and all access points of the storage system are broken.

We compare the convergence speed and, more importantly, the estimation accuracy of both modes when we inject probabilistic failures. The convergence speed is represented by the number of iterations that the algorithm performs to calculate the global minimum coefficients $\bar{\mathbf{b}}$. Fewer iterations indicate faster convergence. Then, we use the global minimum coefficients to calculate the eigenvalues of the state matrix with the Prony algorithm. The estimation accuracy is calculated as the relative error between the calculated eigenvalues of the state matrix and the actual eigenvalues (known as a priori in our evaluation). Since the injected network failures may cause the PDCs to use historical data instead of the recent data, the ADMM algorithm can converge at different minimum coefficients, which result in different estimation accuracies.

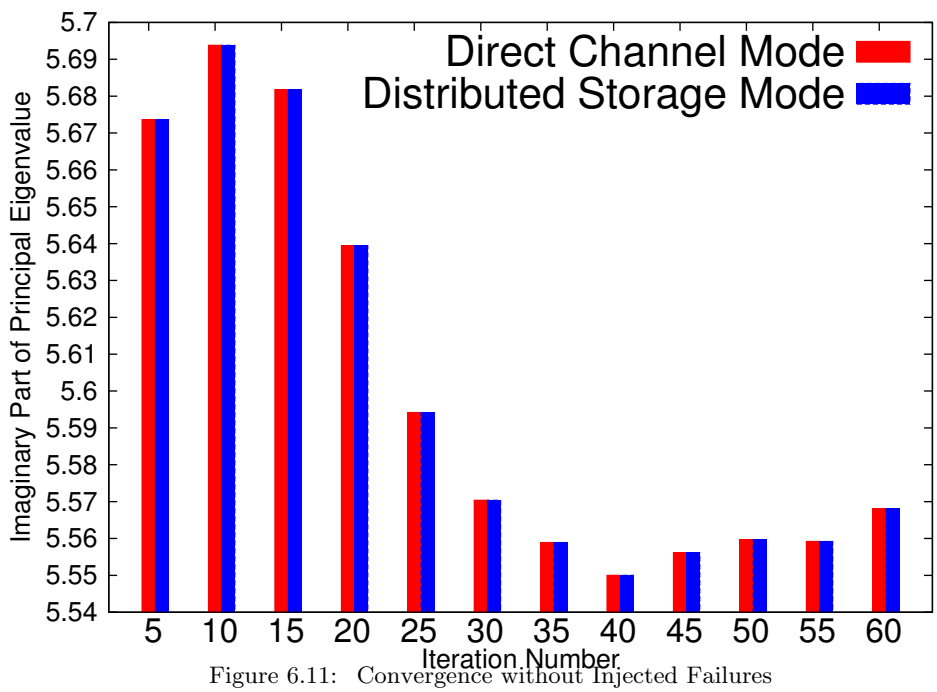


Figure 6.11: Convergence without Injected Failures

Fig. 6.11 depicts the imaginary part of one principal eigenvalue in every 5 iterations when no failures were injected in the experiments. The red line (for direct channel mode) matches the blue line (for storage mode) and $\bar{\mathbf{b}}$ converged in both modes after 63 iterations. When it converges, the imaginary part of the principal eigenvalue is 5.5684. The relative error is 0.072% (the actual value is 5.5644). In this experiment, we used a threshold value of 0.00012 to stop the iteration (see Eq. 6.1). When we injected failures (10% probability to break a channel in each communication), the convergence speed and accuracy in the storage mode, as shown in Fig. 6.12, stayed the same since the PDCs can utilize another entry points to the storage system to access data. By comparison, the direct channel mode converged earlier (after iteration 15) with the same stopping threshold as depicted by the black line. However, it resulted in an inaccurate estimation (with imaginary part of the principal eigenvalue 5.6974), which has a relative error of 2.390%. The accuracy of the direct channel mode decreases when failures where injected because of the data loss. When we decrease the stopping threshold for the direct channel mode, the number of iterations increases and the accuracy increases. However, as shown in the red line, it resulted in a less accurate estimation (relative error 3.127%) than the storage mode even when it required more iterations.

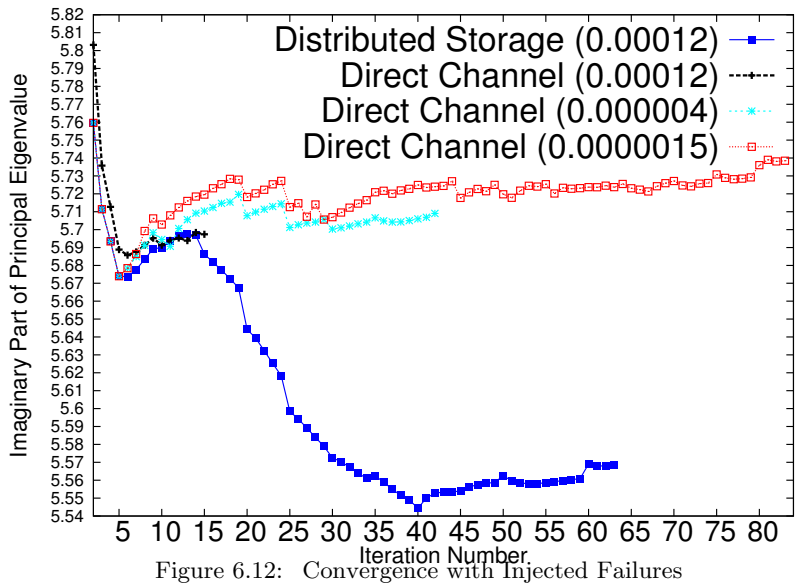


Figure 6.12: Convergence with Injected Failures

Table. 6.5 presents the estimation accuracy when we injected long-term failures to different numbers of PDCs (stop threshold 0.00012). We calculate the relative error of the principal eigenvalue and the L_2 norm of all eigenvalues. The accuracy of the storage mode is not impacted while the direct channel mode has larger errors for long-term failures.

Table 6.5: Comparison of Accuracy with Long-term Failures

Mode	# Failures	Principal Eigenvalue Error	L_2-norm Error
Storage Mode	1	0.072%	0.105%
	2	0.072%	0.105%
	3	0.072%	0.105%
Direct Channel	1	2.237%	1.961%
	2	2.346%	2.071%
	3	2.448%	2.144%

6.4 Distributed Wide-Area Control

For implementation of the distributed wide-area controller, the reduced WECC system is first implemented in Real-time Digital Simulators (RTDS), and integrated with hardware Phasor Measurement Units (PMU), which are further connected to the ExoGENI cloud. PMUs are assigned to the terminal bus of every area of the WECC model. The states measured by the PMUs are then communicated to unique sets of virtual controllers constructed in the cloud. The virtual machines (VMs) share the state information between each other in the cloud over Internet2, and cooperatively compute a state-feedback LQR control algorithm for small-signal oscillation damping of the five-area model. The control signals from each VM are thereafter communicated back to the excitation system of every generator in the RTDS for actuation. A cloud-in-the-loop control system is thereby created. Experimental results from a hardware-in-the-loop testbed environment are reported to support our proposed architecture.

6.4.1 ExoGENI-WAMS Testbed Setup

The setup of ExoGENI-WAMS testbed for the wide-area control loop is described in Figure 6.13. At the physical power grid layer, one PC is used for hosting RTDS and the RSCAD for modeling the 5-machine reduced-order model of the WECC system described in the previous section. RTDS can run at a time-step of 50 micro-seconds while showing the required system response on the host PC. At the information layer, the WAMS consists of five PMUs connected to the Gigabit Transceiver Analog Output (GTAO) card of RTDS connected through ribbon cables. With the GTAO card, RTDS sends out the generated voltage and current waveforms from the model to the PMUs. The PMUs record the phasor values of voltages, frequencies, and phase angles from these voltage and current waveforms, time-stamp them using the GPS clock, and output the data at a rate of 30-60 samples per second.

The RTDS is integrated to ExoGENI through an intermediate physical fiber network called Breakable Experimental Network (BEN) using VLAN technology. All PMUs and GTNETx2 cards that are used to receive control commands are configured into the VLAN 904. All cloud sites at ExoGENI are configured into VLAN 11. Then a network rule is manually created to

translate traffic between VLAN 11 of ExoGENI and VLAN 904 of the RTDS connected to BEN.

At the WAMS application layer, the data packets generated by PMUs are captured and pushed through a virtual communication and computation network developed in ExoGENI. The packet capture capability is enabled by the software of the IEEE C37.118 *message reader* using C++ over a Linux image. The designed LQR wide-area state-feedback controller is emulated in ExoGENI by creating a 6-VM slice. The network topology between the VMs, as shown in Figure 6.13, in this case is a full connected topology since K, in general, is a dense matrix. VM1 to VM5, the so-called *regular VMs*, execute the following functions at every sampling instant: 1) receive streaming PMU data from their corresponding PMUs using C37.118 protocol in real-time; 2) exchange PMU measurements with other VMs; 3) calculates the inter-area damping control signal for the controller; 4) sends the control command to the leader VM. The VM6 serves as a leader VM, and is used to collect control commands from all five regular VMs, convert them into one message in IEEE 754 format, and finally send this packet back to the GTNETx2 card which is connected to PB5 processor card of RTDS through internal fiber optic cables, as shown in Figure 6.13. The flow chart of implementing the wide-area controller is shown in Figure 6.14. It is worthy to note that the data packets, whether they be measurement data or control commands, will go through the virtual network and experience a network delay that is determined by both different VM topologies and their physical locations, as well as the network traffic.

6.4.2 ExoGENI Implementation

We implement the LQR control law in ExoGENI using the VM concept described in Section 6.1 instead of RSCAD. Figure 6.15 displays a comparison of transient responses for the four inter-area angles when the WECC is faulted with the same three phase line-to-ground fault at bus 3 for four cycles. The closed-loop responses are almost same as the ideal responses when the control was implemented in RSCAD. However, a slight degradation in performance is noted due to network delays and use of hardware-in-loop compared to the purely software implementation. Such degradations can be avoided by designing more robust network-layer controllers using SDN principles [45], which is part of our ongoing work.

6.5 Conclusion

In this chapter we described a typical cyber-physical system testbed called ExoGENI-WAMS testbed, and presented comparative study of three well-known modal estimation algorithms for wide area mornitoring of power system using Synchrophasors implemented via such advanced networking and distributed application testbed called ExoGENI. Our results indicate that

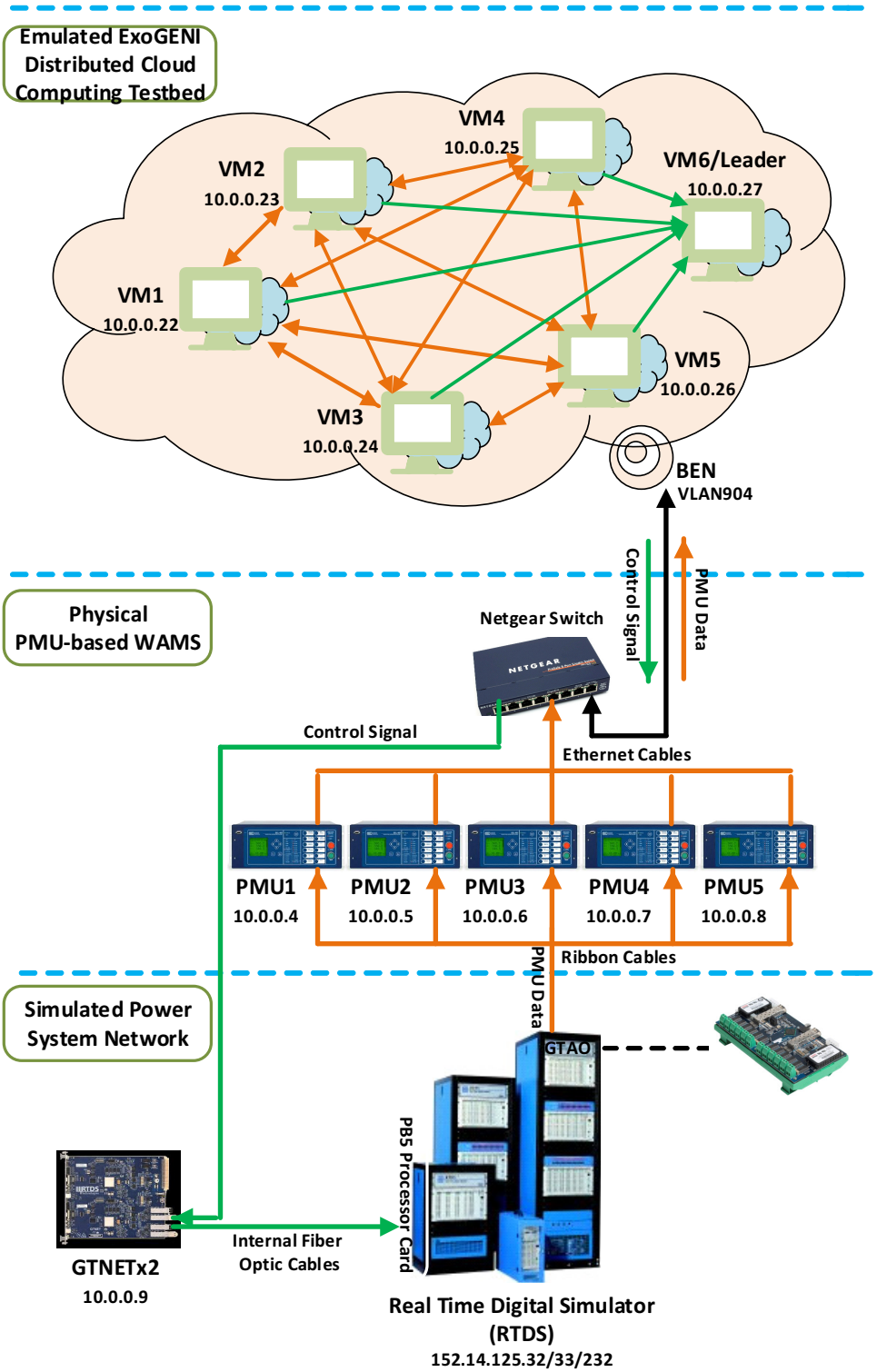


Figure 6.13: Setup of the ExoGENI-WAMS Testbed

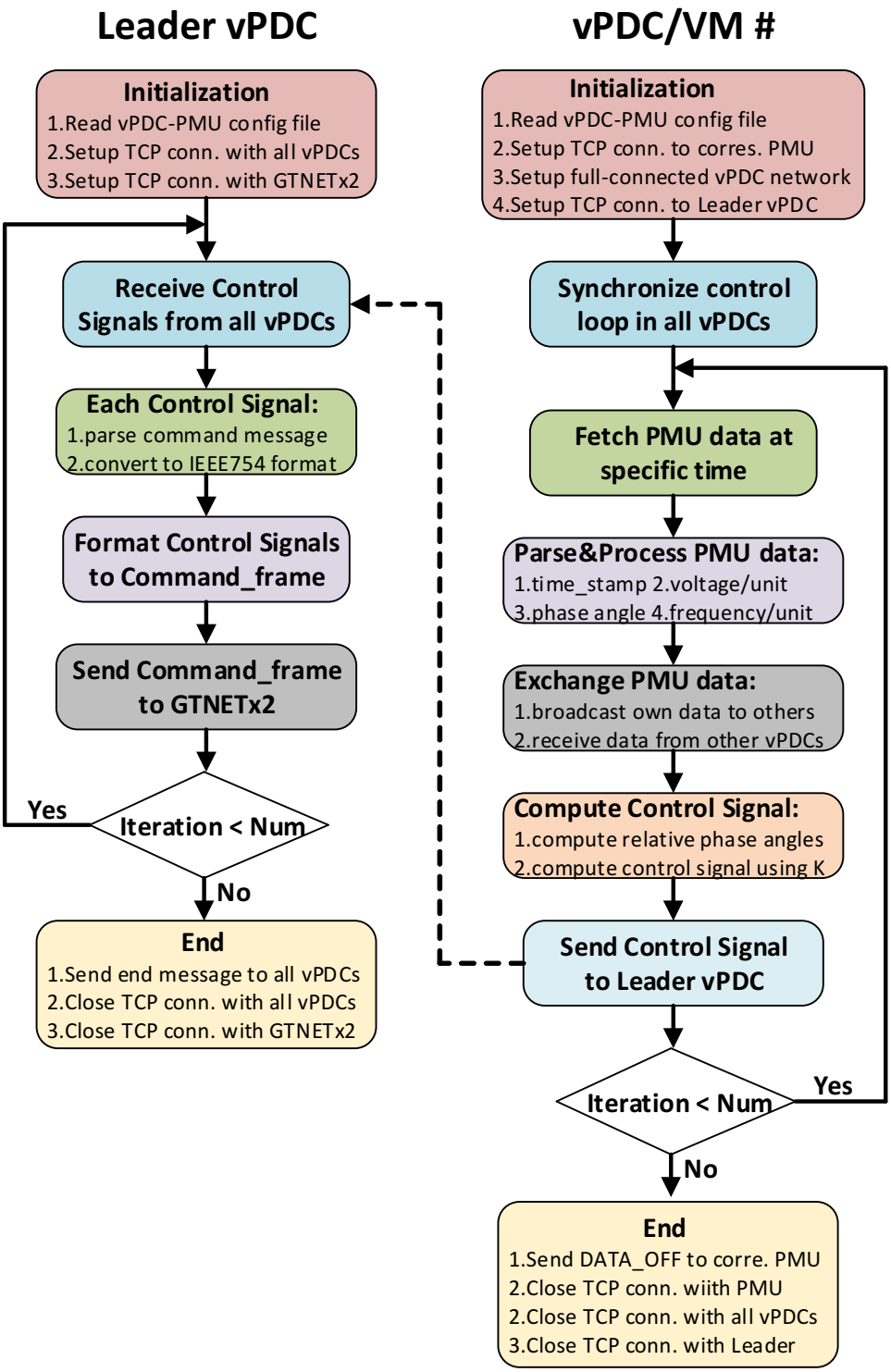


Figure 6.14: Flowchart of Wide-Area Controller

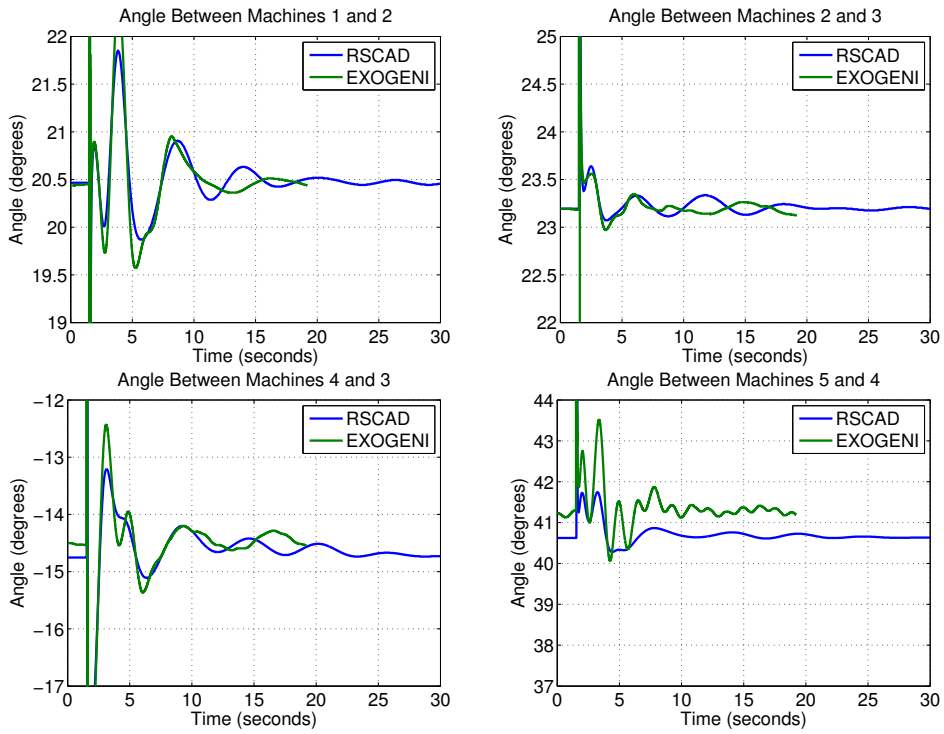


Figure 6.15: WECC Transient Response, RSCAD vs ExoGENI Controller Implementation

decentralized least-squares algorithm can significantly improve the end-to-end delay without losing accuracy compared to the centralized solution. The recursive least-squares algorithm shows similar time-saving characteristics but loses accuracy if the initial guess is far from the actual solution. Additionally, the distributed architecture has higher attack-resiliency as we expected. In particular, we described an experimental set-up for implementing wide-area control of power systems using cloud computing as a computational resource for the feedback path. Our experiments show that the controller performs close to ideal when implemented in the cloud under nominal network traffic conditions. The proposed architecture provides a feasible way to control any cyber-physical system using distributed state or output feedback control.

Chapter 7

Concluding Remarks and Future Works

We have proposed several cyber-physical algorithms to evaluate the impacts of actual communication network on the wide-area real-time oscillation monitoring and damping of a large power system network. We have studied the convergence analysis of the distributed ADMM-based optimization algorithms on the delay distribution parameters and different variants of asynchronous update strategies, and the results exhibit a broad view of how the convergence of any distributed estimation algorithm in a generic cyber-physical system depends strongly on the uncertainties of the underlying communication models. Also we investigated two of the most important and typical cyber physical challenges: 1) end-to-end delay, and 2) attack resiliency, by implementing these cyber-physical algorithms on the ExoGENI-WAMS testbed. Finally, we validate the distributed LQR controller using cloud-in-the-loop.

As for the remaining part of the dissertation work I propose to work on 1) development of hybrid and spatial correlation-based asynchronous update strategies to address the asynchronies of communication networks, 2) validation of A-ADMM algorithms and their integration with DHT-based storage system on the ExoGENI-WAMS testbed, 3) investigation of scalability of distributed delay-robust algorithms.

REFERENCES

[1] W. A. Mittelstadt, P. E. Krause, P. N. Overholt, D. J. Sobajic, J. F. Hauer, R. E. Wilson, and D. T. Rizy, ''The DOE Wide Area Measurement System (WAMS) Project Demonstration of Dynamic Information Technology for the Future Power System,'' *EPRI Conference on the Future of Power Delivery*, Washington, D.C., Apr. 1996.

[2] J. E. Dagle, ''Data Management Issues Associated with the August 14, 2003 Blackout Investigation,'' *IEEE PES General Meeting* - Panel on Major Grid Blackouts of 2003 in North America and Europe, 2004.

[3] A. Bose, ''Smart Transmission Grid Applications and Their Supporting Infrastructure,'' *IEEE Transactions on Smart Grid*, vol. 1(1), pp. 11-19, June 2010.

[4] G. Liu, J. Quintero, and V. Venkatasubramanian, ''Oscillation Monitoring System Based on Wide Area Synchrophasors in Power Systems,'' *Bulk Power System Dynamics and Control, iREP Symposium - iREP*, 2007.

[5] P. Kundur, *Power System Stability and Control*, McGraw-Hill Inc., New York, 1994.

[6] Nabavi, S. and Chakrabortty, A., ''A real-time distributed prony-based algorithm for modal estimation of power system oscillations,'' *2014 American Control Conference*,.

[7] N. Zhou, Z. Huang, F. Tuffner, J. Pierre, and S. Jin, ''Automatic Implementation of Prony Analysis for Electromechanical Mode Identification from Phasor Measurements,'' *IEEE Power and Energy Society (PES) General Meeting,* 2010.

[8] Y. Hu, 'NASPInet Specification Project', *presentation in 2008 DOE V&C Peer Review Meeting*, Washington DC, Oct. 2008.

[9] Y. Liu *et al.*, ''A US-Wide Power Systems Frequency Monitoring Network,'' *Proceedings of the IEEE PES General Meeting*, Montreal, QC, Canada, June 2006.

[10] C.W. Taylor, D.C. Erickson, K.E. Martin, R.W. Wilson, and V. Venkatasubramanian, ''Wide-Area Stability and Voltage Control System: R & D and Online Demonstration,'' *Proceedings of the IEEE*, vol. 93(5), pp 892-906, May 2005.

[11] R. Hasan, R. Bobba and H. Khurana, ''Analyzing NASPInet Data Flows,'' *IEEE Power Systems Conference and Exposition* (PSCE '09), Seattle, WA, March 2009.

[12] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, ''Above the Clouds: A Berkeley View of Cloud Computing,'' *Technical Report* UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

[13] D. Bakken, A. Bose, C. Hauser, D.Whitehead, and G. Zweigle, ''Smart Generation and Transmission with Coherent, Real-Time Data,'' *Proceedings of the IEEE*, June 2011.

[14] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena-Scientific, 1997.

[15] J. F. Hauer, C. J. Demeure, and L. L. Scharf, ‘‘Initial Results in Prony Analysis of Power System Response Signals,” *IEEE Transactions on Power Systems*, vol. 5(1), pp. 80-89, 1990.

[16] N. Zhou, J. W. Pierre, and J. F. Hauer, ‘‘Initial Results in Power System Identification From Injected Probing Signals Using a Subspace Method,” *IEEE Transactions on Power Systems*, vol. 21(3), pp. 1296--1302, Aug. 2006.

[17] Pasqualetti, F. and Dorfler, F. and Bullo, F., ‘‘Cyber-physical security via geometric control: Distributed monitoring and malicious attacks,” *2012 IEEE 51st Annual Conference on Decision and Control (CDC)*, pp. 3418-3425, Dec. 2012.

[18] L. Ljung, *System Identification: Theory for the User*, Prentice Hall, NJ, 1999.

[19] P. M. Anderson and A. A. Fouad, *Power System Stability and Control*, IEEE Press, NJ, 2002.

[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, ‘‘Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” *Foundations and Trends in Machine Learning*, vol. 3(1), pp. 1-122, 2011.

[21] T. Athay, R. Podmore, and S. Virmani, ‘‘A Practical Method for the Direct Analysis of Transient Stability,” *IEEE Transactions on Power Apparatus and Systems,* vol. 98(2), pp. 573-584, 1979.

[22] Chakrabortty, A. and Yufeng Xin, ‘‘Hardware-in-the-Loop Simulations and Verifications of Smart Power Systems over an Exo-GENI Testbed,” *Second GENI Research and Educational Experiment Workshop (GREE)*, pp. 16-19, 2013.

[23] Weiss, M, ‘‘Wide-Area Monitoring and Control of Power Systems using Real-Time Hardware-in-the-Loop Simulation,” *Prelimary Exam Report*, 2015

[24] S. Nabavi, J. Zhang, and A. Chakrabortty, ‘‘Distributed Optimization Algorithms for Wide-Area Oscillation Monitoring in Power Systems Using an Inter-Regional PMU-PDC Architecture”, *IEEE Transactions on Smart Grid*, vol.6(5), pp.2529-2538, Sept. 2015.

[25] J. Zhang, S. Nabavi, A. Chakrabortty, and Yufeng Xin. ‘‘ADMM Optimization Strategies for Wide-Area Oscillation Monitoring in Power Systems under Asynchronous Communication Delays,” *submitted to IEEE Transactions on Smart Grid*, 2015

[26] J. Zhang, S. Nabavi, A. Chakrabortty, and Y. Xin, ‘‘Convergence Analysis of ADMM-based Power System Mode Estimation Under Asynchronous Wide-Area Communication Delays,” *in proceedings of IEEE PES General Meeting*, Denver, CO, Jul. 2015.

[27] S. Nabavi and A. Chakrabortty, ‘‘Distributed Estimation of Inter-area Oscillation Modes in Large Power Systems Using Alternating Direction Multiplier Method,” *IEEE Power & Energy Society General Meeting (PES)*, 2014.

[28] G. Hooghiemstra and P. Van Mieghem, ‘‘Delay Distributions on Fixed Internet Paths,” *Delft, The Netherlands, Tech. Rep.*, 2001.

[29] T. Chang, M. Hong, and X. Wang, ‘‘Multi-Agent Distributed Optimization via Inexact Consensus ADMM,” *IEEE Transactions on Signal Processing*, vol. 63(2), pp. 482-497, Jan. 2015.

[30] T. Qian, A. Chakrabortty, F. Mueller, and Y. Xin, ‘‘A Distributed Storage System for Multi-Resolution Virtual Synchrophasors,” *IEEE PES General Meeting*, Washington DC, July 2014.

[31] E. Wei and A. Ozgladar, ‘‘On the $O(1/k)$ Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers,” 2013 [online] Available: *http://arxiv.org/abs/1307.8254*.

[32] N. Zhou, J. W. Pierre, and J. F. Hauer, ‘‘Initial Results in Power System Identification From Injected Probing Signals Using a Subspace Method,” *IEEE Transactions on Power Systems*, vol. 21(3), pp. 1296--1302, Aug. 2006.

[33] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, ‘‘Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” *Foundations and Trends in Machine Learning*, vol. 3(1), pp. 1-122, 2011.

[34] The DETERLab, *http://http://www.deter-project.org*.

[35] Using Flukes, *http://https://geni-orca.renci.org/trac/wiki/flukes*.

[36] A. G. Phadke and J. S. Thorp *Synchronized Phasor Measurements and Their Applications,* Springer, 2008.

[37] A. Chakrabortty and P. Khargonekar, ‘‘Introduction to Wide-Area Control of Power Systems,” *American Control Conference*, Washington DC, 2013.

[38] North American Synchrophasor Initiative (NASPI) - *www.naspi.org*

[39] S. Nabavi, J. Zhang, and A. Chakrabortty, ‘‘Distributed Optimization Algorithms for Wide-Area Oscillation Monitoring in Power Systems Using an Inter-Regional PMU-PDC Architecture,” *IEEE Transactions on Smart Grid*, vol. 6(5), pp. 2551-2559, Sep. 2015.

[40] P. Myrda and K. Koellner, ‘‘NASPI-net: The Internet for Synchrophasors,” *Hawaii International Conference on System Sciences*, (HICCS), 2010.

[41] D. Bakken, A. Bose, C. Hauser, D.Whitehead, and G. Zweigle, ‘‘Smart Generation and Transmission with Coherent, Real-Time Data,” *Proceedings of the IEEE*, 99(6), June 2011.

[42] K. Zhu, M. Chenine, and L. Nordstrom, ‘‘Design Requirements of Wide-Area Damping Systems-Using Empirical Data From a Utility IP Network,” *IEEE Transactions on Smart Grid,* vol. 5(2), 2014.

[43] G. Chavan, M. Weiss, A. Chakrabortty, S. Bhattacharya, A. Salazar, and F. Ashrafi, "Identification and Predictive Analysis of a Multi-Area WECC Power System Model Using Synchrophasors," *to appear in IEEE Transactions on Smart Grid*, 2016.

[44] Global Environment Network Innovations (GENI): *wwww.geni.net*

[45] F. Hu, Q. Hao, and K. Bao, " A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation, *IEEE Communication Surveys and Tutorials,* vol. 16(4), 2014.

[46] A. Fouad and M. Anderson, *Power System Control and Stability,* IEEE Press, 2003.

[47] A. Chakrabortty and S. Bhattacharya,"Wide-Area Monitoring and Control of WECC Transfer Paths Using Real-Time Digital Simulations", Oct 2013.

APPENDIX

Appendix A

Flow Charts

A.1 Centralized RLS and Distributed PronyADMM

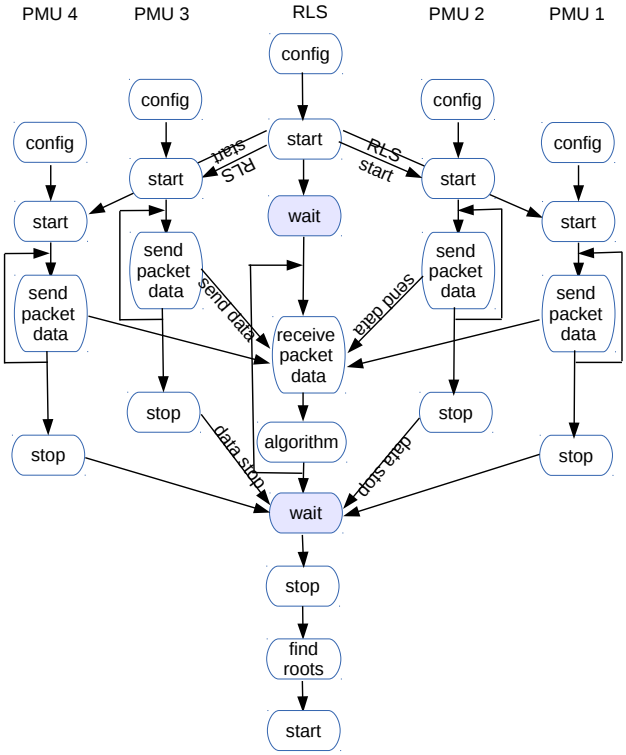


Figure A.1: Flow Chart of Centralized Recursive Least-Squares Algorithm

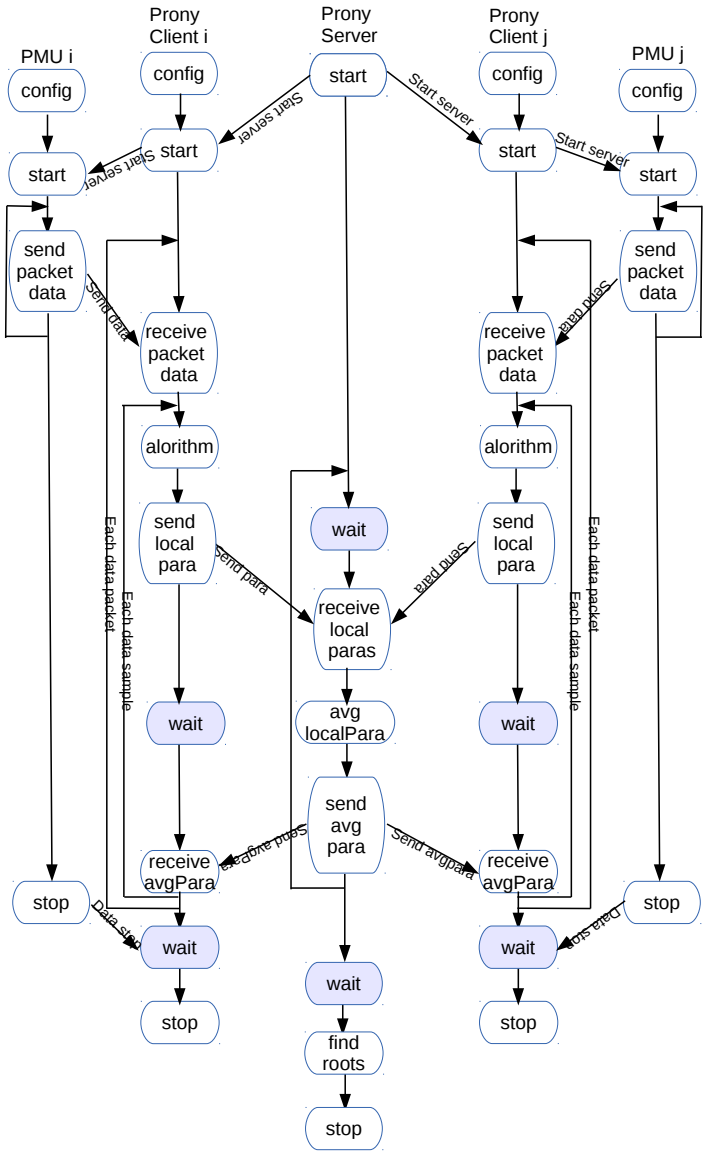


Figure A.2: Flow Chart of Distributed PronyADMM Algorithm

A.2 Resiliency Implementation of Centralized RLS and Distributed PronyADMM

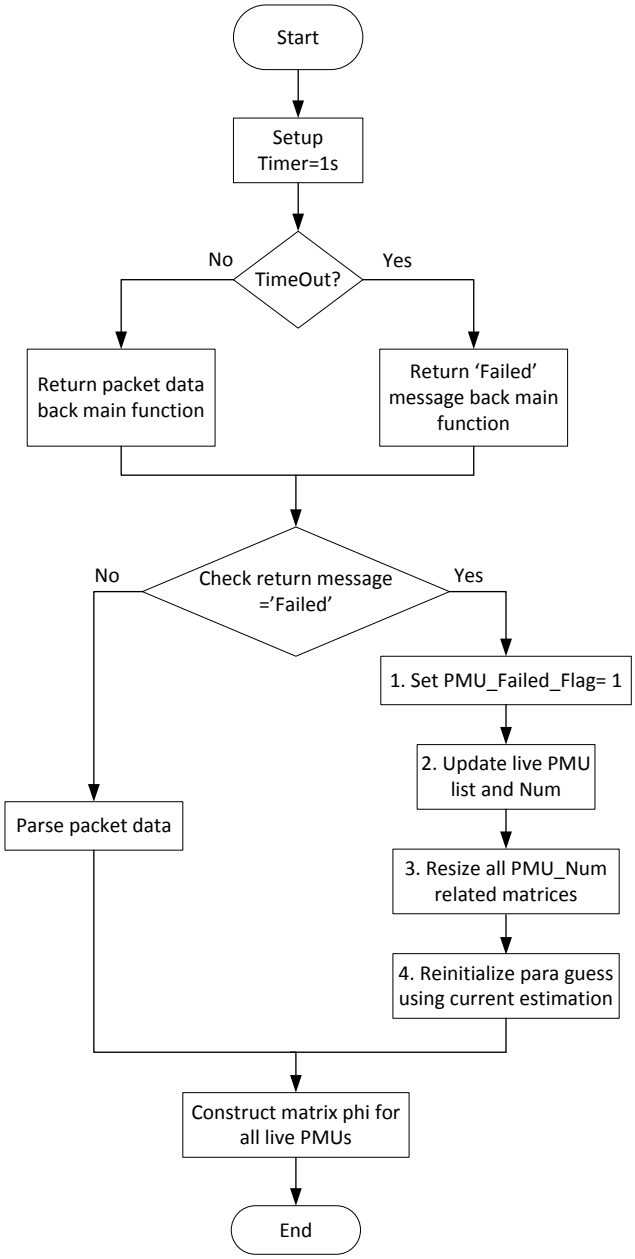


Figure A.3: Flow Chart of Resiliency Mechanism for Centralized RLS Algorithm

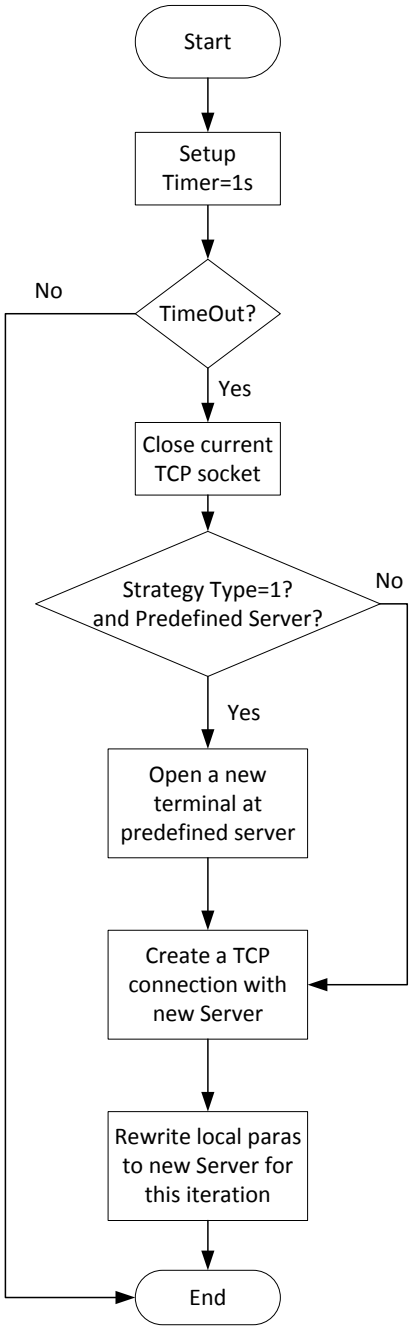


Figure A.4: Flow Chart of Resiliency Mechanism for Distributed PronyADMM Algorithm

A.3 Decentralized LS and RLS Algorithms

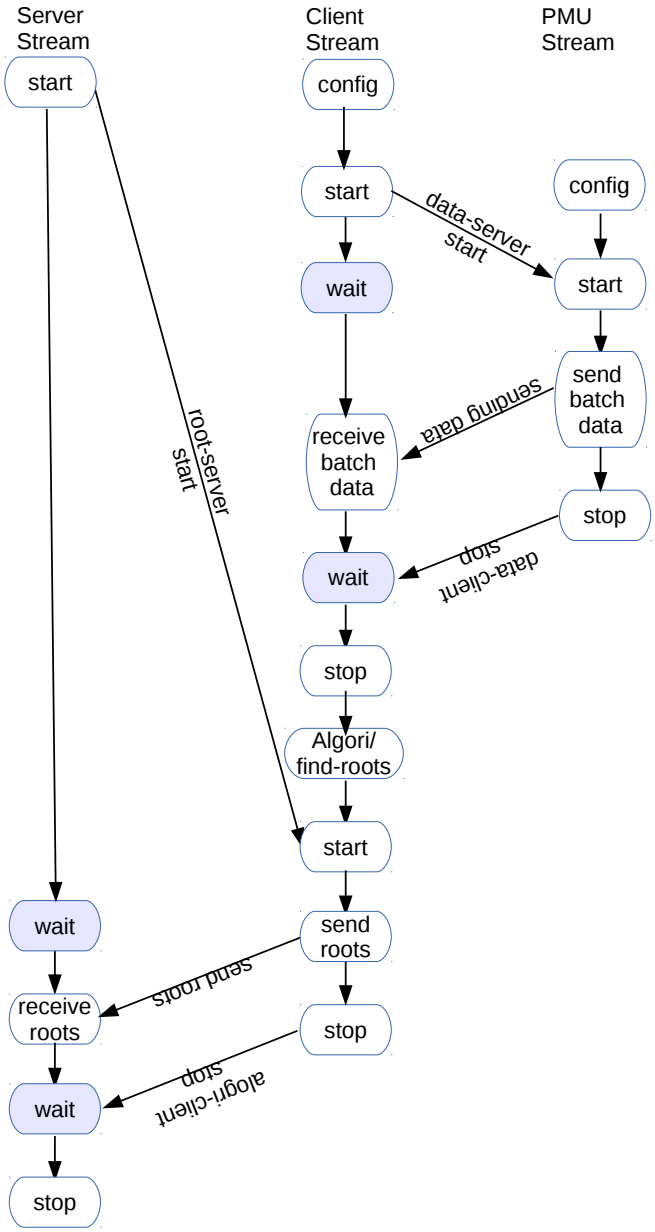


Figure A.5: Flow Chart of Decentralized Least-Squares Algorithm

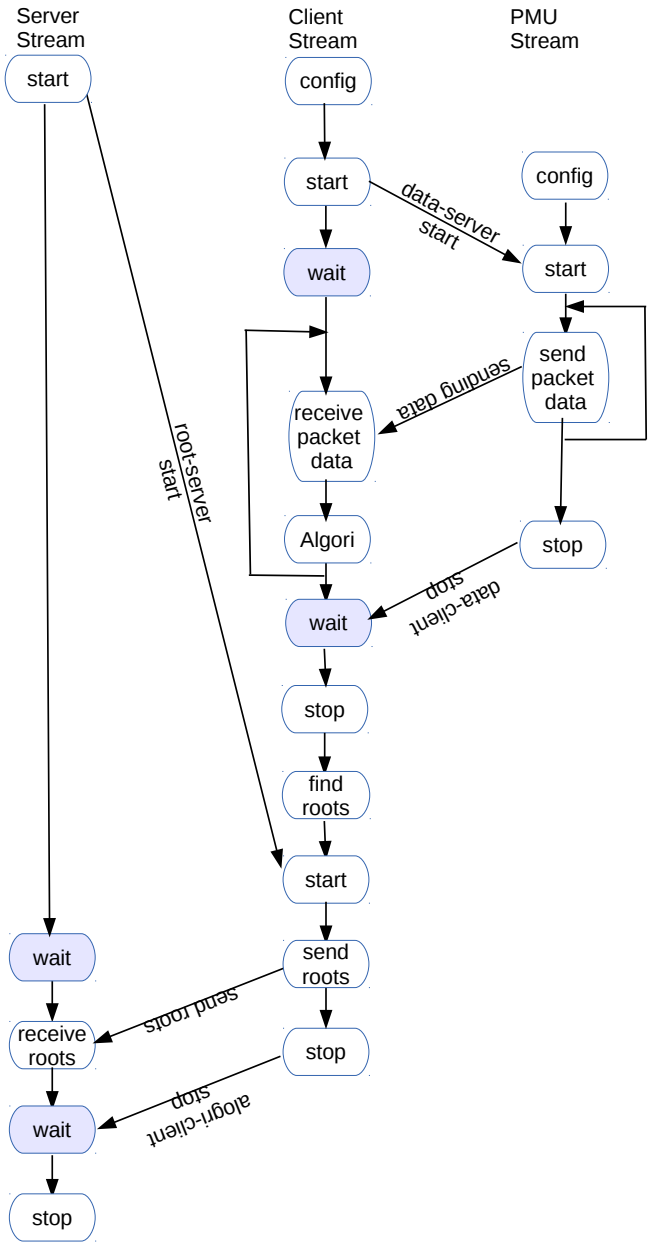


Figure A.6: Flow Chart of Decentralized Recursive Least-Squares Algorithm