5-2017

Wide-Area Synchrophasor Data Server System and Data Analytics Platform

Dao Zhou
University of Tennessee, Knoxville, dzhou3@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Dao Zhou entitled "Wide-Area Synchrophasor Data Server System and Data Analytics Platform." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Electrical Engineering.

Yilu Liu, Major Professor

We have read this dissertation and recommend its acceptance:

Hairong Qi, Wei Gao, James Ostrowski

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Wide-Area Synchrophasor Data Server System and Data Analytics Platform

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Dao Zhou
May 2017

Acknowledgements

I would like to express my deepest gratitude and appreciation to my advisor, Dr. Yilu Liu, for her patient guidance and consistent encouragement through my PhD study. She is not only an academic advisor, but also a mentor for my life who has passed on wisdom and inspiration.

In addition, I would like to give special thanks to mycommittee members, Dr. Hairong Qi, Dr. Wei Gao, Dr. James Ostrowski for their precious time, guidance and suggestions.

Next, I would like to thank all my colleagues: Dr. Yanzhu Ye, Dr. Jingyuan Dong, Dr. Ye Zhang, Dr. Yin Lei, Dr. Lingwei Zhan, Dr. Yong Liu, Dr. Jiahui Guo, Dr. Gefei Kou, Dr. Jidong Chai, Dr. Lin Zhu, Dr. Jiecheng Zhao, Ling Wu, Shutang You, Micah Till, Wenxuan Yao and Hesen Liu. It was a great pleasure to work them. I would also like to thank staff members in EECS department, Ms. Dana Bryson and Mr. Markus Iturriaga for their generous help in many ways.

Last but not least, I would like to appreciate my parents, my wife and daughter, for their love and support.

Abstract

As synchrophasor data start to play a significant role in power system operation and dynamic study, data processing and data analysis capability are critical to Wide-area measurement systems (WAMS). The Frequency Monitoring Network (FNET/GridEye) is a WAMS network that collects data from hundreds of Frequency Disturbance Recorders (FDRs) at the distribution level. The previous FNET/GridEye data center is limited by its data storage capability and computation power. Targeting scalability, extensibility, concurrency and robustness, a distributed data analytics platform is proposed to process large volume, high velocity dataset. A variety of real-time and non-real-time synchrophasor data analytics applications are hosted by this platform. The computation load is shared with balance by multiple nodes of the analytics cluster, and big data analytics tools such as Apache Spark are adopted to manage large volume data and to boost the data processing speed. Multiple power system disturbance detection and analysis applications are redesigned to take advantage of this platform. Data quality and data security are monitored in real-time. Future data analytics applications can be easily developed and plugged into the system with simple configuration.

Table of Contents

List of Tables

List of Figures

Chapter 1 Overview of FNET System

1.1 Introduction

With the increasing loads of power grids and massive inter-area power transfers enabled by the deregulation, it is very important to improve operators' situational awareness. A Wide-area measurement systems (WAMS) consists of advanced measurement technology, information tools, and operational infrastructure that facilitate the understanding and management of the increasingly complex behavior exhibited by large power systems [1]. The Synchrophasor technologies, as the major components of the WAMS, provide significant information about the bulk power grid captured by Phasor Measurement Units (PMUs). Large volumes of data are streamed into a central data server in real-time. However, unless the server system is designed with the capability to efficiently process and analysis the data, system operators cannot exploit the information hidden in the data to assist the system operation or control.

Big data technologies are increasingly boosting the performance to handle data with large volume, high velocity and variety [2]. Introduction of the technologies like Hadoop and Spark could bring the smart grid data analytics to a new era, in which comprehensive analysis algorithms can be applied in real-time, and close-loop data solutions can be implemented for situation awareness, asset management, planning, fault detection and protection [3]. Some works has been done to study the availability using big data technologies for grid data storage [4], data processing [5], and data analysis [6]–[8]. However, most of these studies are based on simulation or offline data. More challenges have to be solved to build an infrastructure for real-time data collection, analysis and visualization.

1.2 Frequency Monitoring Network (FNET/GridEye)

The FNET/GridEye system is a wide-area synchrophasor measurement network, which measures synchrophasor information at distribution level, as shown in Figure 1-1.Frequency and voltage phase angle information about system dynamics can be obtained using low-cost, high-accuracy Frequency Disturbance Recorder (FDR), which is a single phase version of PMU [9], [10].

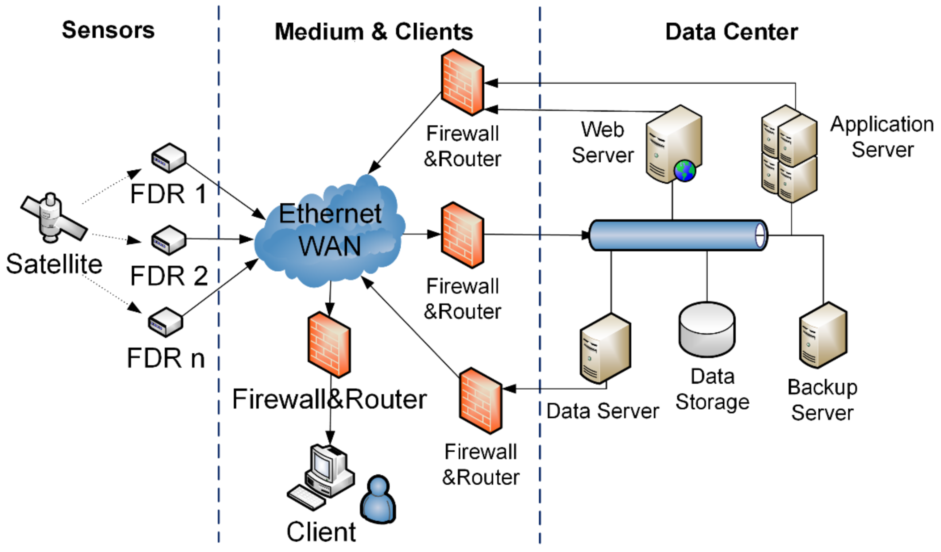


Figure 1-1 FNET/GridEye structure

Since 2004, more than 150 FDRs have been deployed in United States and about 50 are deployed worldwide, as shown in Figure 1-2 and Figure 1-3. These measurements are then time-stamped and transmitted to the FNET/GridEye data center at the University of Tennessee, Knoxville for data processing and long-term storage. A variety of

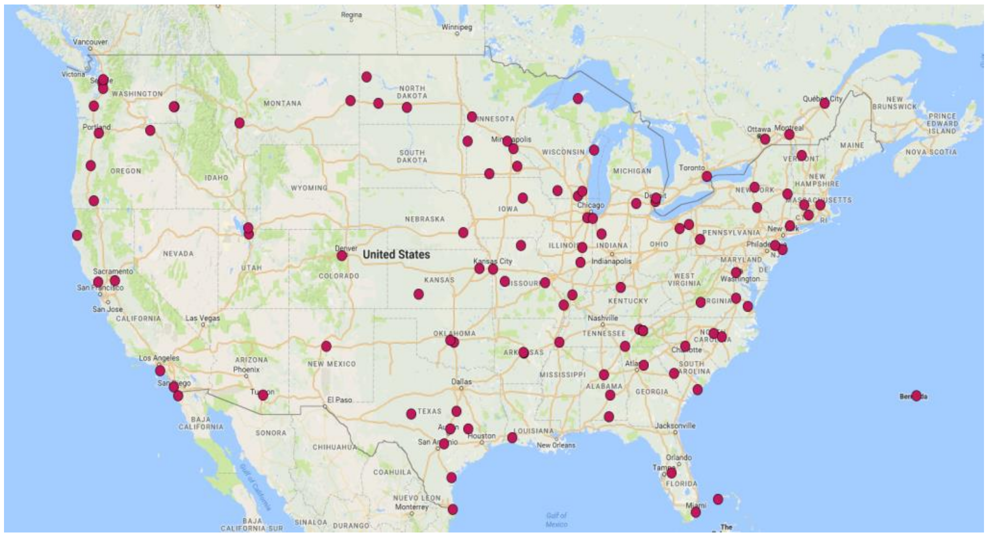


Figure 1-2 Map of FDR locations in North America.

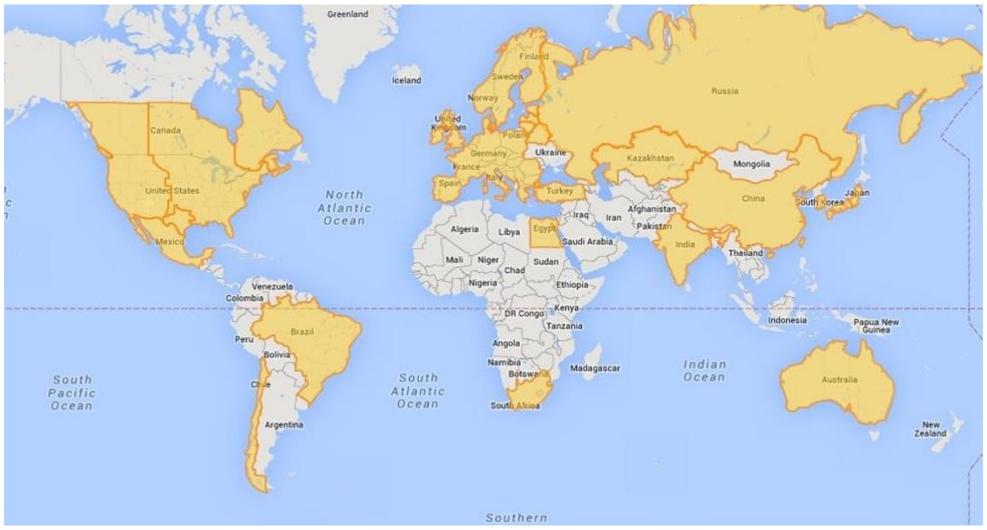


Figure 1-3 Map of worldwide FDR coverage.

synchrophasor applications have been developed over FNET/GridEye situational awareness system, including real-time event detection and location estimation, oscillation detection and modal analysis, line trip detection, off-grid/islanding detection, and forensic authentication of digital evidence [9]–[19].

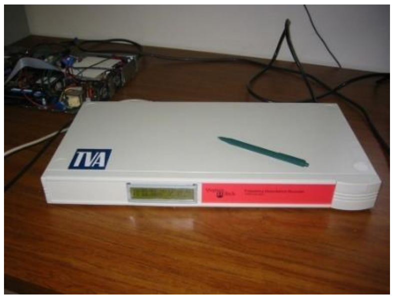
1.3 Frequency Disturbance Recorders

FDR is a GPS-synchronized single-phase PMU, which measures frequency, voltage magnitude and angle at distribution level. The first generation of FDRs was built in 2003, and it has expanded to three generations and several relative devices as shown in Figure 1-4.

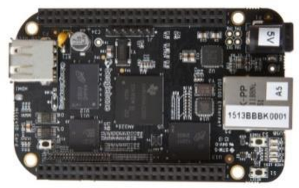
The measurement data are transmitted to MCU and sent out to FNET servers through the network module by TCP/IP protocol. Timing synchronization of the measurement data is important for FDRs and FNET. Discrete Fourier Transform (DFT) is one of the common algorithms used in synchrophasor measurement area, and it is adopted as the basic framework of the FDR algorithm. The angle and frequency error of the most updated version of FDR is less than 0.005 degree and 0.00015Hz respectively at 60Hz nominal frequency.

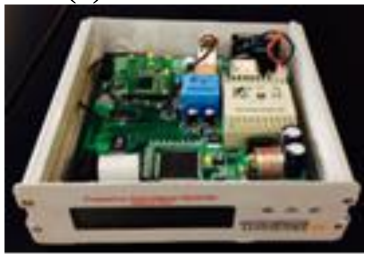
1.4 Organization of Materials

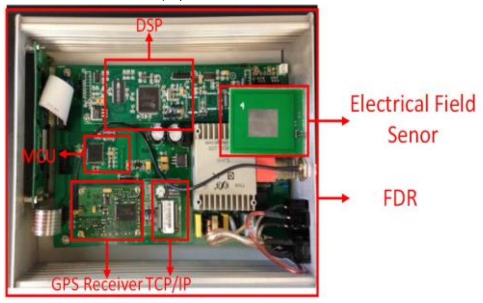
The data collection server system and improvements are discussed in Chapter 2. The challenges and methods to develop real-time synchrophasor applications based on FNET/GridEye system is discussed in Chapter 3. An example application, real-time line trip detection, is illustrated in Chapter 4. The data quality and data security issues about synchrophasor applications are discussed in Chapter 5. A data analytics platform is


(a)

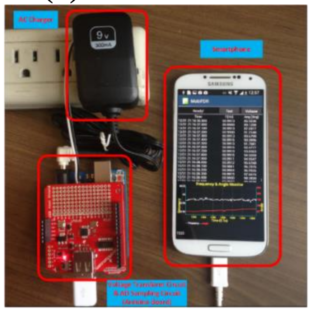
(b)


(c)


(d)



(e)



(f)

Figure 1-4 Frequency Disturbance Recorder
(a) Generation I, (b) Generation II (c) Generation III (d)Universal Grid Analyzer (e) Wireless FDR (f) FDR on Smartphone

proposed to for big data applications in Chapter 6. The conclusions and future works are summarized in Chapter 7.

Chapter 2 Data Collection

2.1 Data Collection Server System

The FNET/GridEye server system collect data from FDRs via internet connection. The FDRs are configured to set up TCP connections between the FDR data transmission interface and the FNET servers. Two servers in the system are programmed to communicate with FDRs directly, including a main data collection server and a backup server. Each of the FDRs are configured to create independent connections to the two servers using their server IP addresses as destination. (The backup server IP is set up using Google domain name service, so FDRs are configured to set connection to a domain name URL.)

Both main data collection server and backup server are programmed to handle three major functions, receiving data from internet, store received data locally, and forward the data stream to other servers for other application purposes. These functions are implemented using TCP sockets.

The TCP sockets are designed with client-server models, supporting guaranteed connection with three-way handshakes. A brief illustration of TCP socket model is shown in Figure 2-1. The TCP client is defined as the connection end which initiate the connection, while the TCP server is defined as the end which waits for connection to be requested. A basic TCP connection is built within a few steps: 1) The server creates a TCP socket, bind with its local IP address and port number, and start to listen to incoming connection requests; 2) the clients creates a TCP socket, and request a connection using the server IP and port number; 3) the server observes the connection request, it can accept and process the request to setup the connection; 4) once the connection is configured, either the server side or the client side can send and receive data packets; 5) if the data transmission is finished, either the server or the client can close and destroy the socket and notify the other side.

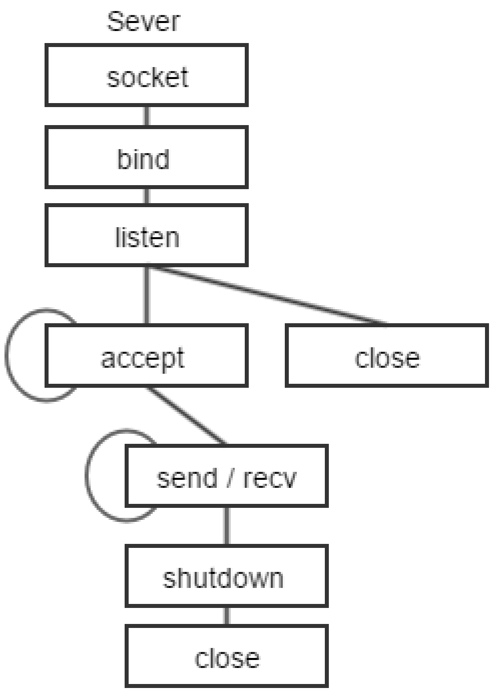
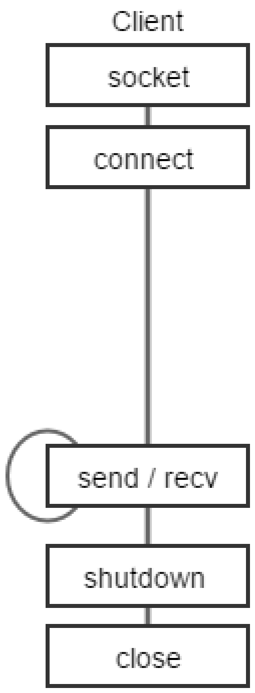


Figure 2-1 TCP sockets connection model

In the data collection server, FDRs are the TCP clients and the server host is the server. However, when the data collection server forwards data to other servers, the sending server is the TCP client, and the receiving servers are considered as TCP server. In this case, the data is always transmitted from the TCP client to the TCP server. The data collection server is used as a proxy to forward data, as shown in Figure 2-2. Since each FDR is set to send data to a specific port, the number of listening and receiving socket depends on the number of FDRs. For each of the FDR, the sever will create a list of forwarding sockets to connect to other servers. Therefore, the number of forwarding sockets depends on not only the number of FDRs, but also the number the remote servers that the data collection server forwards data to.

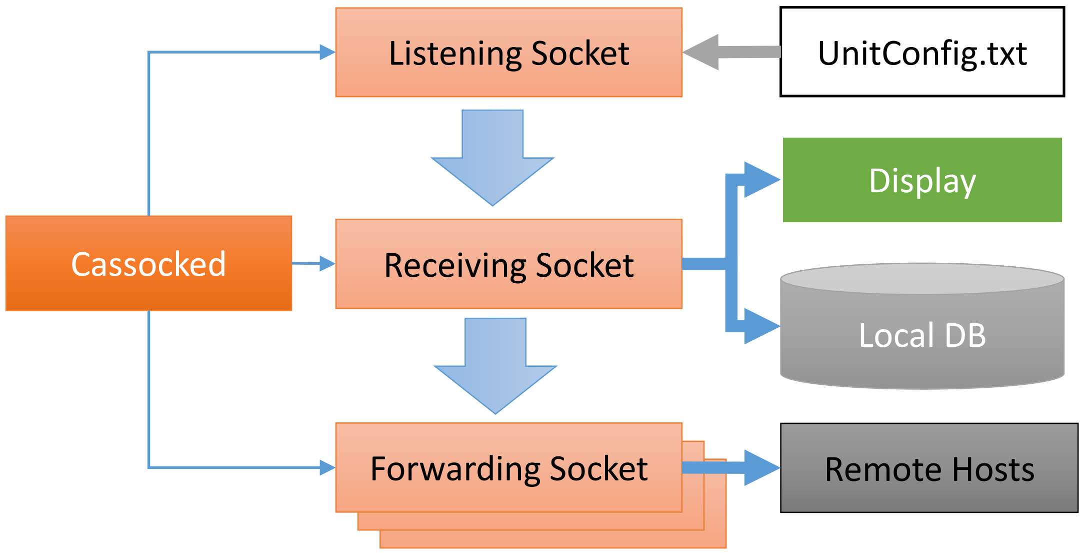


Figure 2-2 Functionality of data collection server

Once the FDR data is received, the serve will verify the received data packet, and store them in a local MS Access database. Also, the data will be send to screen display module if the unit is selected on the server program user interface. MS Access database is a light weight database system that has a capacity limit of 2GB. The server overcome this issue by automatically backup the Access files at certain time of the day.

2.2 FDR Configuration Synchronization

The problem of storing the FDR unit list using a text file is that each server has its own unit list, which makes it difficult to maintain. Instead of manually update the text file on each server, a new scheme to synchronized the FDR unit configuration file is implemented using MySQL database.

All the servers in the FNET system query this MySQL database for the FDR information. Therefore, the FDR information are always synchronized among the servers. However, there are chances that the server hosting the MySQL database is not accessible, because of network failure or server failure. In this case, to ensure the other servers operate normally, each server also maintains its own local configuration files, as shown in Figure 2-3.

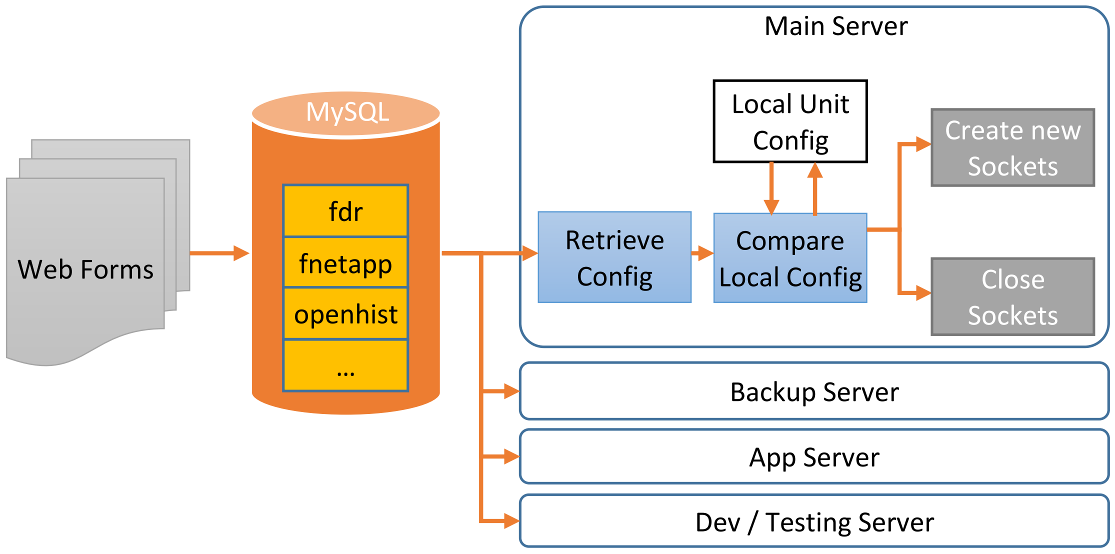


Figure 2-3 FDR configuration synchronization

The FDR information, including their hardware information, IP address, operation status and the information about their hosts, are stored in a MySQL database, as shown in Figure 2-4.

Whenever a server needs to update its FDR configuration information, it trys to query the most updated information from the MySQL database. If the FDR information is

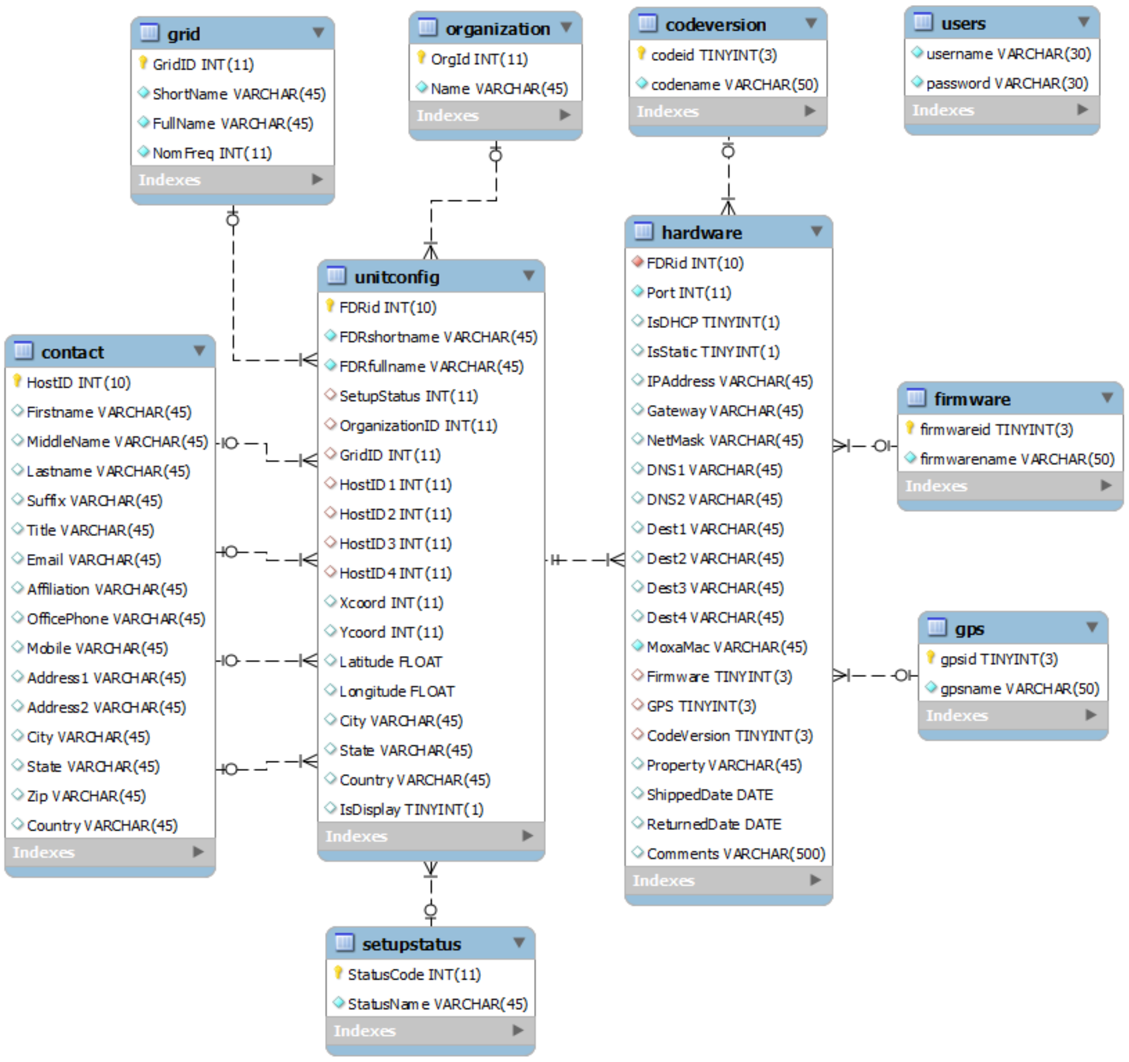


Figure 2-4 Data Model of FDR Schema in the FNET Configuration Database

retrived, it compares the updated information with its local configuration files. The server will then decide which units are added to or removed from the system. Since the sockets has high resource consumption and depends on the number of FDRs and number of forwarding destinations, the serve can dynamically create and destroy sockets based on the diffence between the updated FDR list and the existing FDRs and sockets running in the server. In this way, the server resource consumption, including networking, computing and memory is significantly reduced.

To further simplify the process to update the FDR information for FDR distribution team, a web interface is developed to update the FDR list and the host information. With this scheme, the FDR information is easily updated, syncronized among difference servers, while the server performance is improved and the maintenance job is simplified.

2.3 Timestamp Correction

The early versions of FDRs, including the first generation and some second generation, have been observed with random timestamp shifts. The shift is caused by the mismatch of FDR local clock and the GPS synchronized timing system.

The FDR clock is driving by its local oscillator. However, this clock is affected by many factors like temperature. Thus, a universal clock is required to provide accurate clock synchronized across the system. Current FDRs and PMUs are using GPS signal to synchronize the clocks. FDRs synchronize its local clock by receiving a Pulse Per Second (PPS) signal from the onboard GPS module, and the timestamp is requested from the GPS module at the beginning of every second.

However, even though the PPS signal can be considered accurate, the timestamp can be shifted. For example, if the local clock runs faster than the GPS clock, the FDR will have a shorter length of "second" comparing the the GPS. It may request the timestamp twice at the beginning and the end of one local second, while both request are received within one GPS second. That will result a same timestamp obtained twice by the FDR. This phenomenon is called "overlapping second". On the other hand, if the local clock runs slower than GPS clock, a "missing second" may occur, as shown in Figure 2-5

By investigating the data of all FDR units, most of the timestamp shifts occur in paris, which means a missing second is usually observed before or after a overlapping second. The time slot between a missing second and a overlapping second is usually within a few seconds, and most of them are one second. This is because the local clock is corrected by the PPS signal quickly after it shift away, then the shifted second will shift back whin in one second.

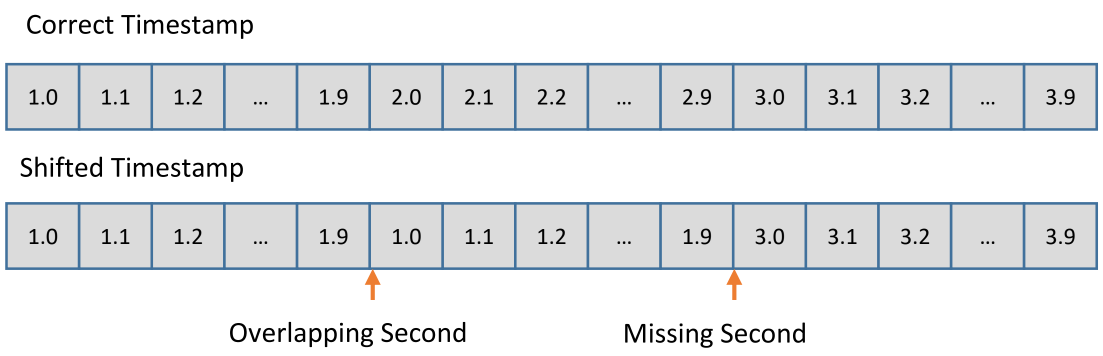


Figure 2-5 Timestamp shift caused by clock mismatch

To correct the shifted timestamp, a correction algorithm is implemented on the data collection server, as shown in Figure 2-6. Once the TCP data stream is received by the receiving socket, and parsed by the data parsing function, the timestamp of the current data packet is buffered in a two-second processing queue. The timestamp processing queue store the timestamps received in the past to seconds, and the head of the queue is the timestamp received two second earlier. The "second" part of the timestamps are extracted and compared. If the timestamps are not increamentally consecutive, the timestamps in queue will be corrected based on the head timestamp of the queue, then the data in the queue will be pushed to the next processing module for further steps, including display, data forwarding and store to MS Access database. Otherwise, if the timestamps are verified to be correct, the data in the head of the queue are sent to the processing module directly.

By implementing this algorithm, the timestamps within two seconds are ensured to be increamentally consecutive. If the timeshift slot between the missing timestamp and the overlapping timestamp is greater than two second, this algorithm can not guarantee the timeshift over two second can shift back. However, since most of timestamp shift are within one second, this is a efficient solution. Test results show over 90% of the timestamp shifts can be corrected.

The timestamps shifts issue is compensated in the later version of FDR hardware. However, many of the old FDRs can not be replaced in a short time. The algorithm can be used to correct timestamp for either online data, or the stored historian data.

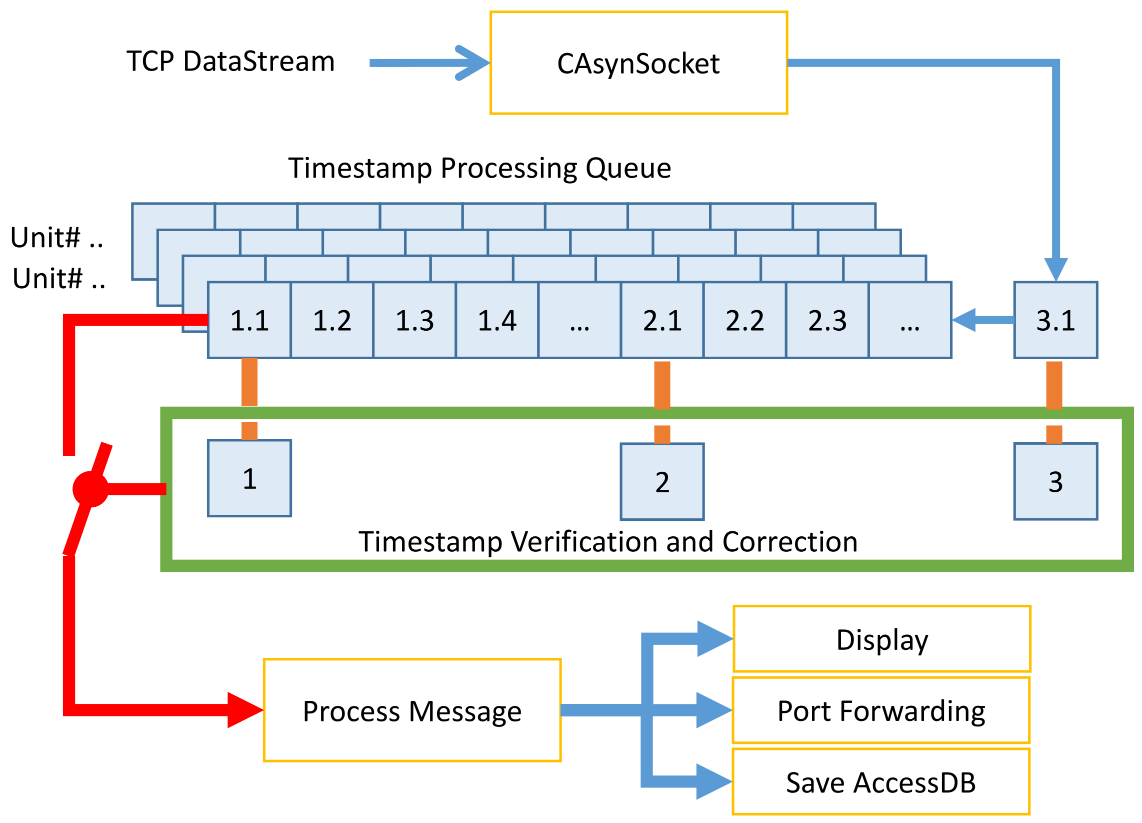


Figure 2-6 Timestamp correction algorithm

2.4 Conclusion

The data collection server received data from FDRs through TCP sockets. It store data in local database and forward them to other server systems. By synchronizing the FDR configuration information, the information updates and maintenance are simplified. The timestamp correction algorithm solves over 90% timestamp shift caused be clock mismatch.

Chapter 3 Real-time Synchrophasor Applications

3.1 Motivations

As synchrophasor measurements and applications play a growingly critical role in the modern power systems, the number of FDRs has expanded to approach the limit of the originally designed system capacity. Also, an increasing number of real-time applications and data analytics algorithms have pushed the system to reach its computation limit. Thus, the FNET/GridEye system is proposed to be redesigned to incorporate the following features:

Scalability: The system is required to collect, store and process the data without delay as the number of FDRs continuously increases.

Extensibility: A variety of technologies developed in recent years to solve power system stability issues also requires the server system to include easy plug-in interface for adapting new applications.

Concurrency: More complex data analytical algorithms are introduced to the system, which requires higher computation capability. To ensure the data are processed in real-time, the server system should be able to execute applications in parallel by distributing the computation stress to multiple nodes.

Robustness: The system should be able to operate under stress and tolerate unpredictable or invalid input.

3.2 Synchrophasor Applications

Since 2006, the FNET/GridEye system has been developed to host a series of synchrophasor applications for power system monitoring and visualization. A list of currently implemented online applications is shown in Table 3-2.

Several other online applications are being developed or under test, including Fault-Induced Delayed Voltage Recovery (FIDVR) detection, dynamic Beta-value estimation, etc.

Beyond these online application, FNET/GridEye system also provides the platform supporting a variety of offline data analytics, including post-event analysis, long-term statistics, data quality analysis etc. [9].

3.2.1 Real-time Online Applications

Synchrophasor applications are usually time sensitive. In the FNET/GridEye system, the online applications are divided into two tiers based on their response time, real-time application and near real-time applications, as shown in Table 3-1.

Table 3-1 Two tiers of Real-time Applications

	Tier 1	**Tier 2**
Time constrains	Real-time	Near-real-time
Response Time	<10s	<2 min
Server Platform	Data Server	Application Server
Data Source	Memory	Historian
Sensor Units	All units	Selected units
Includes (ID as shown in Table 3-2)	1, 2, 3, 4, 5, 6	7, 8, 9, 10, 11,12*

*The response time of Video Replay depends on the video resolution and time length. It usually takes 5 to 10 minutes to render a video.

Table 3-2 Online Applications hosted by FNET/GridEye

ID	Application	Description
1	Event Trigger	Detects generator trip and load shedding by continuously monitoring the ROCOF of the incoming frequency data [11]
2	Oscillation Trigger	Detects inter-area oscillation by monitoring the relative phasor angle [12]
3	Islanding Trigger	Detects the situation in which a part of the grid becomes electrically isolated from the remainder of the power system [13]
4	Line Trip Trigger	Detects a line outage event by monitoring sudden change but quickly damped frequency feature on local units [14]
5	Ambient Mode Analysis	Continuously analyzes system ambient oscillation frequency and damping ratio in real-time
6	Real-time Data Visualization	Web-based real-time data display in table, trend, and map format [15]
7	Event Location	Estimates generation trip or load shedding event location using TDOA algorithm [16]
8	Frequency Response Analysis	Analyzes frequency excursion during events, estimates events MW amount, point A,B,C frequency etc. [17]
9	Oscillation Mode Analysis	Estimates inter-area oscillation frequency, magnitude and damping ratio for dominate modes [12], [19]
10	Online Report	Generates web-based online analytics reports for detected disturbances [15]
11	Email Alert	Sends Email alert messages to registered customers about detected disturbances in real-time [10]
12	Event Video Replay	Automatically generates animation video for disturbances [15]

The real-time applications require fast reaction, usually in seconds. Thus the computation is designed to stay close to the data source. Trigger algorithms with minimum computation complexity are designed to achieve this goal with data read from buffered memory in the data server. In contrast, the Tier 2 applications obtained data from historian. This enables the applications to process time-aligned data with assured data quality. Tier 2 applications are developed to handle more complicated algorithms like location triangulation and oscillation mode analysis.

Although the time requirement of the Tier 2 applications are not as critical as those of Tier 1 applications, some data processing algorithms are still intensive for a computation node, especially when multiple applications request the CPU simultaneously.

3.2.2 Online Applications Dependency and Priority

In many cases, the online applications cannot be isolated from each other. It is intuitional to hold the report and visualization for the results from other applications, and the Event Location executes only when the Event Trigger fires. However, there are some hidden links between certain applications. For instance, both the Event Trigger and the Oscillation Trigger analyze the incoming data streams of all FDR/PMU units, unless one or a few units are isolated from the rest of the power grid, which is detected by the Islanding Trigger. In this case, the Islanding Trigger has to execute in advance to provide the unit status for the other two triggers, even though they are all considered as Tier 1 application. Figure 3-1 demonstrates the dependency of the current online applications on FNET/GridEye system.

The dependency between applications purposes significant challenge for application distribution. Each of the applications has to be ranked with priority, and queued in the system to execute.

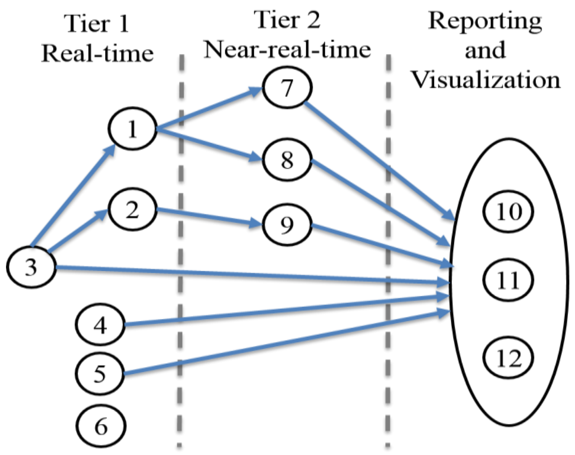


Figure 3-1 Online application dependency

3.2.3 Offline Data Analytics

The offline analytics normally refers to applications which are not time sensitive, including but not limited to post-event analysis, statistical analysis, data quality analysis, system model validation, model reduction, social impact and forensic studies, etc. [9], [18]. These applications usually require extraction of large volume of data from historian, analysis data in temporal or spatial domain, and compare or integrate with data from other sources.

An example application could be a project to summarizing the GPS synchronization losses of all the FDR units in the past five years, and finding their relationship with local weather conditions. In this case, large volume data has to be processed, tabulated with location coordinates, and matched with data from historical weather report. For statistical analysis like the example, big data analytics techniques and parallel processing are introduced to accelerate the computation speed.

3.3 System Architecture Design

To solve the challenges brought by the real-time phasor applications, as well as to ensure the system scalability, extensibility, concurrency and robustness, a distributed computing architecture is proposed to host the server system and data analytics platform, as shown in Figure 3-2 .

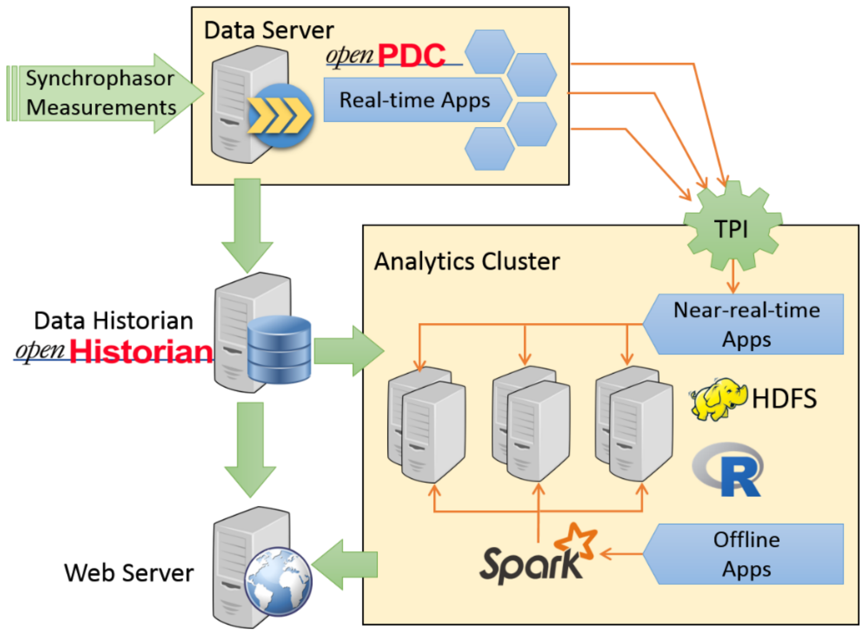


Figure 3-2 Distributed data analytics system architecture

The proposed architecture consists of a data server, a data historian, a web server, and a computing cluster for data analytics. The synchrophasor data are collected by the data server through internet. Tier 1 applications are executed in the data server, processing real-time data from all income phasor measurements. The real-time phasor data are concentrated and aligned with timestamps and dumped to historian. Also, the phasor data are scanned by the real-time triggers, which detect disturbances and send trigger messages to the analytics clusters.

The analytics cluster contains a series of computation nodes. Whenever the analytics cluster receives the trigger message, the trigger processing interface (TPI) finds its corresponding Tier 2 applications, and dispatches them to the computation nodes. Also, the TPI is responsible for loading the data from historian according to the requested time range. The applications are distributed into different node based on their computation complexity, so that the computation load is balanced.

For offline applications, the analytics cluster can be reused if there is no Tier 2 application being processed or waiting in queue. Multiple software packages, including Spark, Hadoop and R, are hosted on the cluster to support big data analytics.

The web server requests down-sampled data from the historian for real-time display, as well as publish disturbance analysis reports whenever it receives the processing results from analytics cluster.

By utilizing this design, the Tier 1 and Tier 2 applications are decoupled. Tier 1 applications stays close to the data source for its time-sensitivity, while Tier 2 applications are capable to process complex computation using the clusters. Both near-real-time applications and offline analytics can scale up easily by adding more computation nodes. Failure of any one application does not impact the other applications, or the performance of the whole system.

3.4 System Implementation

Following the proposed architecture, the distributed data collection and analytics system is implemented with each component elaborated in this section.

3.4.1 Data Concentrator

The FNET/GridEye data server plays the role as a data concentrator that collects data from hundreds of FDRs around the world. Synchrophasor data are transferred to the server through TCP connections. Since the FDRs are developed as distributed level PMUs, the FDR data format is designed as a simplified version of IEEE C37.118 standard format [20], which only contains timestamp, frequency, angle, voltage information measured at the distribution level, as well as the GPS coordinates of each FDR unit. The openPDC developed by Grid Protection Alliance (GPA) is tailored to adopt FNET/GridEye data stream format [21], and serves as the server-side concentrator software for FNET/GridEye system. Once a connection is built, the data server continuously receives the data packet, parses data to structured format, and verifies data validity.

Even though the FDRs are synchronized with GPS time, and their data packets are labeled with timestamp, the network latency from a FDR to the data server is not guaranteed. This requires the data server to be synchronized with an accurate clock and to be able of tolerate minor delay from different FDR units.

The data server is synchronized with a NTP server clock to compensate the time drift effect caused by the inaccuracy of its local clock. Other clock sources, like atomic clock or eLoarn clock are optional to improve the timing accuracy of the server system [22].

The openPDC allows operator to customize the Lead Time and Lag Time parameters for the input data streams. The Lead Time represents the maximum time intervals that the server can accept if an arriving data packet has a timestamp ahead of local server clock. The Lag Time defines the maximum time that the server can wait for the data to arrive with a timestamp later than the local server clock [21]. The data server buffers all the incoming data within the Lead Time and Lag Time range, and then publish the data with the same timestamp as a collective measurement frame. The phasor data from all channels are sorted and aligned with the correct timestamps before they are dumped to the historian or transferred to other servers.

3.4.2 Data Storage

The capability to write and read large volume of data with high speed is critical for the data historian. After exploration of a several relational databases, including MS Access and MySQL, and a variety of NoSQL databases, including MongoDB and Cassandra, the openHistorian 2.0 developed by GPA is selected to fulfill the requirements.

The openHistorian 2.0 is a file based storage system designed to efficiently integrate and archive SCADA, synchrophasor, digital fault recorder and other process control data to support real-time grid operations and post-disturbance analysis. It supports indexed data retrieval interface and lossless data compression, which largely saves the storage capacity of the historian. The archive files produced by the openHistorian are ACID Compliant [23], which create a very durable and consistent file structure that is resistant to data corruption. Internally the data structure is based on a B+ Tree that allows out-of-order data insertion.

It also provides high-speed APIs that can be customized for visualization of real-time and historical data, web-based data access and remote historian data extraction [24].

3.4.3 Real-time Disturbance Detection

The major objective of real-time applications is to detect disturbance instantaneously. All the detection triggers are hosted in the data server, which has fast access to the data as soon as they are parsed and cached into the memory. The disturbance detection triggers are implemented by customizing the Input/Action/Output Interface (IAON) Adapters of the Grid Solutions Framework (GSF), which is the fundamental library collection of openPDC and openHistorian [21].

The real-time triggers, include event trigger, oscillation trigger, islanding trigger and line trip trigger etc., are extended from the action adapter class. The functions to process the measurement data are implemented as modular methods within the action adapter. Each of the triggers consists a series of modules to be executed in order. For instance, Figure 3-3 shows the implementation of the line trip trigger as an action adapter with series of function modules [14]. An input adapter and the historian output adapter are also included for testing purpose.

The triggers are fired if the final processing result exceeds the predefined threshold. The thresholds for these triggers are different for each power grid, thus the configuration parameters and thresholds are open to user for initialization and modification. Some of the commonly used functions, like the average or median filter are packaged as libraries to share across multiple triggers.

The triggers of each power grid interconnection are defined as a thread, so they can be executed in parallel. In practice, however, since the real-time trigger applications may have dependencies, they cannot be processed in the form of “embarrassingly parallelism”, but rather in a pipelined structure, as shown in Figure 3-4. In this case, Trigger 2 and 3 rely on the decision result of Trigger 1, thus they cannot make final decision before Trigger 1 thread publishes its results. Even if Trigger 2 completes the computation before Tigger 1, it holds the result until Trigger 1 finalizes the decision.

An example is the dependency between the islanding trigger and the others, as shown in Figure 3-1. If an FDR unit detects islanding from the rest of the system, it should not be counted for the event trigger or oscillation trigger. However, regional islanding is rare case in the power system. To ensure the real-time features of the trigger computation, event trigger and oscillation trigger calculate the preliminary results, and wait for the islanding trigger to confirm that islanding is not detected, then fire the alarm trigger. Otherwise, the two trigger have to remove the islanded unit and recalculate the results.

3.4.4 Trigger Processing Interface

Since the real-time triggers are designed to detect disturbances as early as possible, the triggers are not capable of confirming the case without complete analysis. Some phasor features may create false alarms, and some may invoke multiple types of triggers. It is critical to retrieve more detailed data and perform more complicated analysis using the near-real-time applications. Detailed information including event location and oscillation modes can be estimated by retrieving data from the data historian and applying algorithms like Location Triangulation and Matrix Pencil methods.

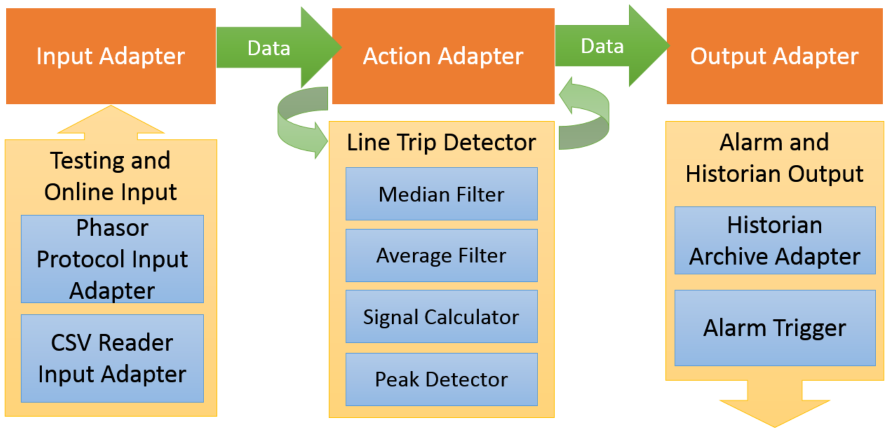


Figure 3-3 IAON Adapter layer design of Line Trip Trigger

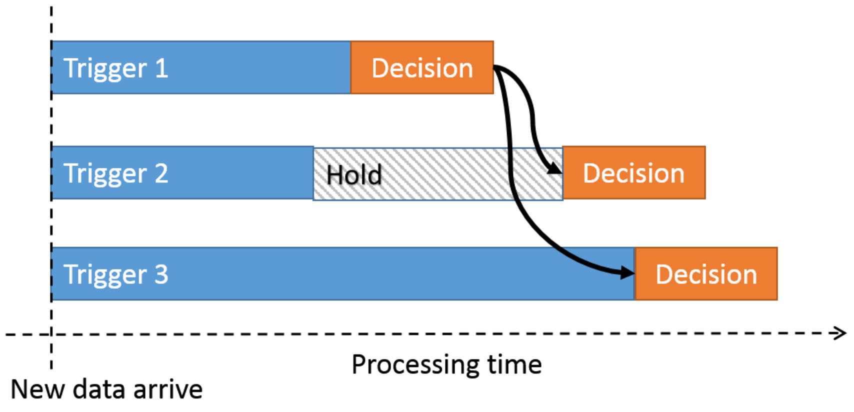


Figure 3-4 Trigger threads with dependencies

To implement the distributed data analysis platform as designed in Section III, the TPI is developed to facilitate the communication between the data server and the analytics cluster nodes, and to dispatch the analysis tasks and historian data to the proper applications.

The TPI is developed as a shell library with the functionality as shown in Figure 3-5. Whenever a real-time trigger is fired, the TPI sender broadcasts a trigger message to the analytics cluster though a UDP connection. The message is constructed by metadata of the trigger information, including the trigger type and the time range of relevant historian data.

The analytics cluster hosts a list of data analysis applications. The received trigger messages are queued in buffer, and checked with priority. Some disturbances may fire multiple triggers, for instance, a generator trip mixed with oscillation. It is necessary to analyze both of the generator trip and oscillation. However, in some other cases, like the measurement from an islanded unit shows a line trip like feature. Only the islanding should be considered in this case, while the line trip trigger can be treated as a false alarm. With the priority check, system operators can configure a ranked order that the data analytics applications should follow.

The TPI also provides interfaces to load historian data from openHistorian and dispatch the data to the suitable applications to process. In case any error occurs, the TPI has authority to kill the application process.

With the implementation of TPI, the analysis applications are fully decoupled and distributed to multiple nodes. The computation load is shared in balance, and future applications can be tested and deployed easily.

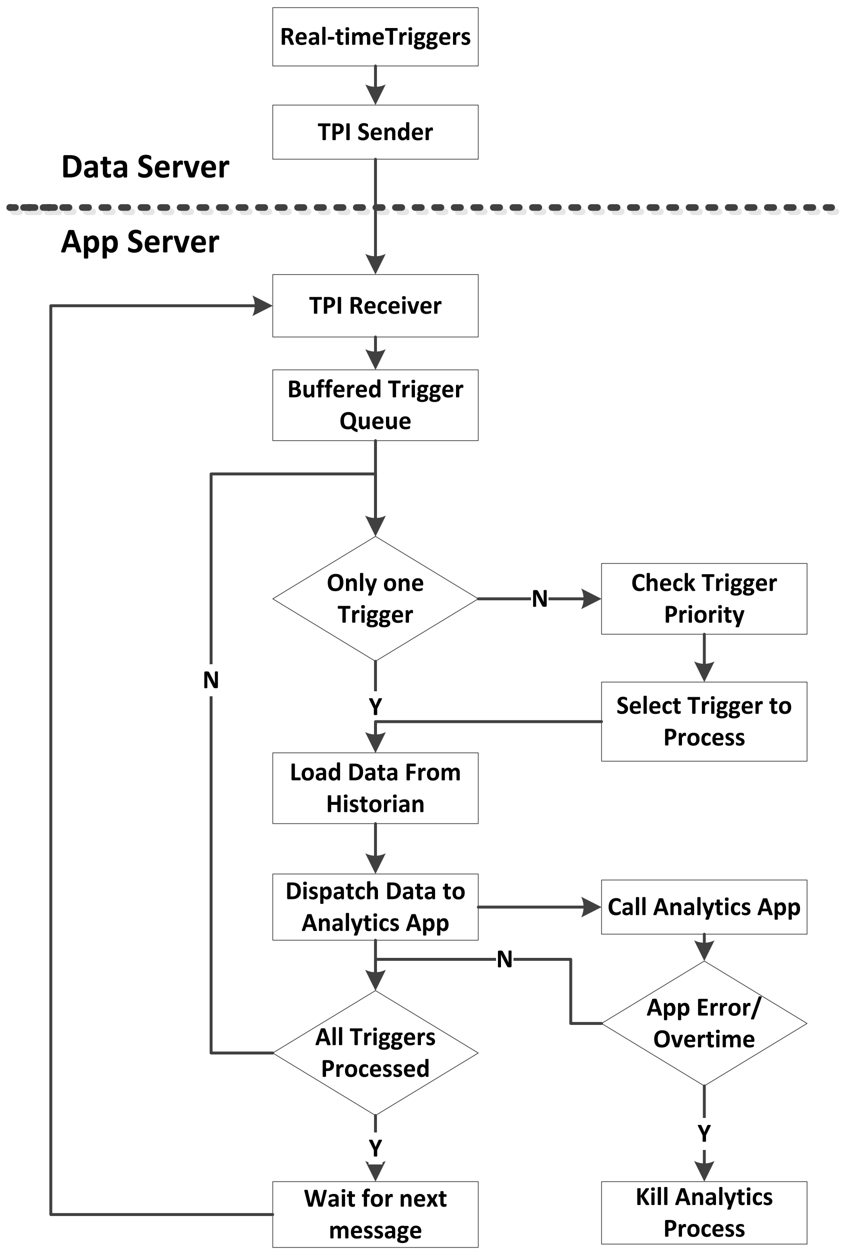


Figure 3-5 Workflow of TPI

3.4.5 Distributed Near-real-time analytics

The near-real-time analytics involves more comprehensive algorithms comparing to the real-time applications. The modules that take most of computation burden in these algorithms are usually the ones responsible for processing of the data from all FDRs. Because of this feature, the algorithms are naturally parallelizable. For instance, analysis of the oscillation frequency and damping ratio of each FDR using Matrix Pencil algorithm can be shared by multiple nodes on the cluster.

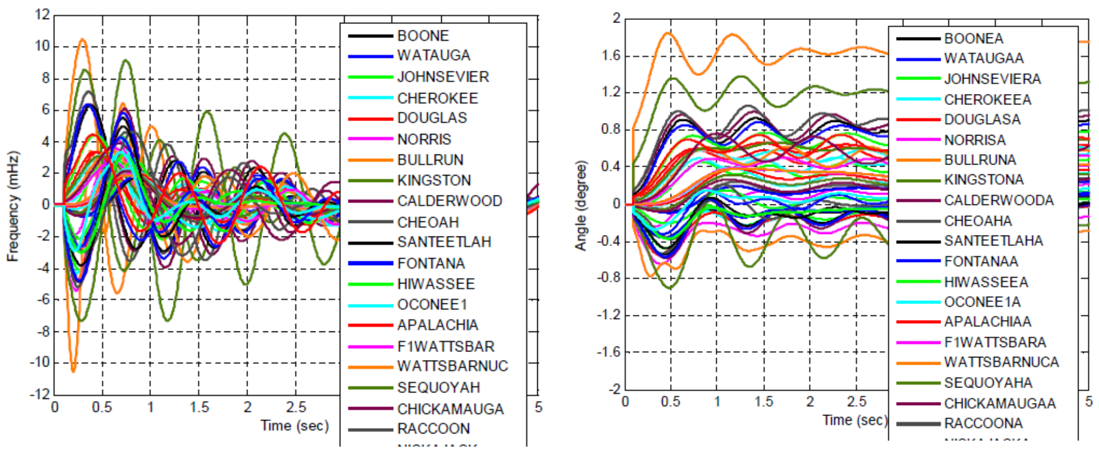
Unlike the statistical analysis of long term historian data, the input data size of the near-real-time applications is limited within a few minutes. Thus, even without utilizing the installed Hadoop HDFS file system, in-memory processing with multi-threads or MPI shows good performance for these applications. Other real-time processing techniques like Apache Storm or Spark Stream are being investigated to further boost the performance.

Chapter 4 Real-time Line Trip Detection

4.1 Algorithm Design

Previous study shows that both frequency and phase angle are informative for line outage detection, as frequency variations directly reflect the machine rotational speed thus related to the generation and load, and voltage phase angle is related to the power flow distribution in the power network [3].

From simulation and observed measurements, line trip events impact only the nearby system on both sides of the tripped line: frequency rises suddenly on the power sending side and drops on the power receiving side. At the same time, voltage angles shift rapidly from their original value. Figure 4-1 shows the simulation results of tripping a 500kV line in the TVA system [3].



(a) (b)

Figure 4-1 Line trip caused (a) frequency variation (b) voltage angle shift

FNET/GridEye sensors are able to capture high precision frequency measurements.

However, since they are widely deployed at distribution level, their angle measurements may not reflect the transmission level angle change caused by line trip. It is practical to develop the line trip detector based on frequency variations for higher accuracy.

Based on the characteristics of frequency variation, an algorithm is developed to detect such events. The block diagram of this algorithm is shown in Figure 4-2.

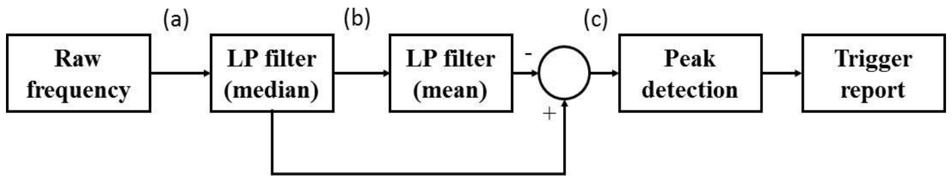
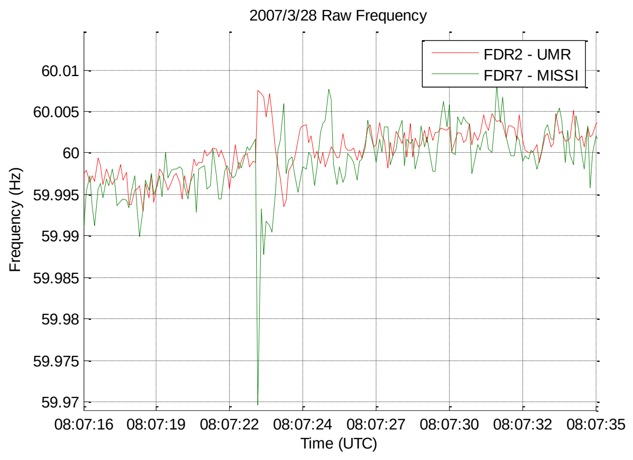


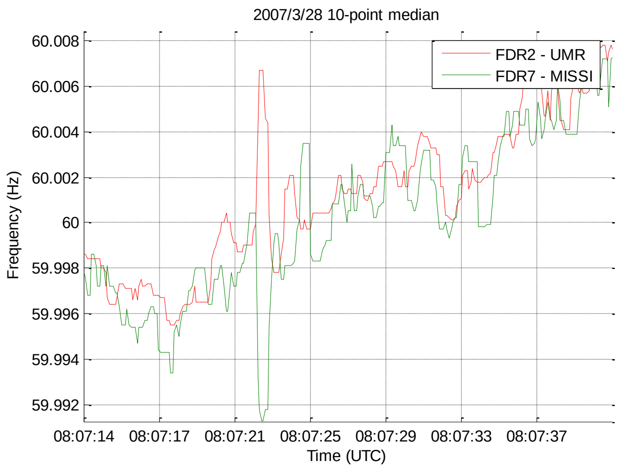
Figure 4-2 Line trip detection block diagram

A sample line trip event is presented in Figure 4-3 to explain the algorithm. The signals from two FDRS at the sending side and the receiving side respectively are shown in this figure. Figure 4-3(a) shows the raw frequency data where the line trip introduced oscillations are observable together with other high-frequency noises and possible local dynamics in the distribution network. The first low-pass filter in the block diagram in Figure 4-2 is to remove the high-frequency noises in the raw frequency and a moving median filter is used. The filtered frequency data is shown in Figure 4-3(b).

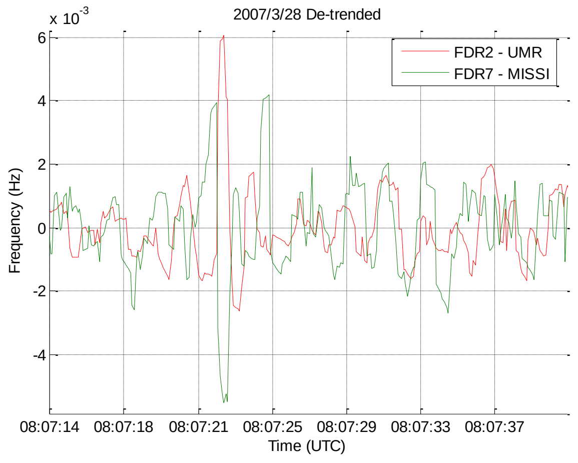
Then the filtered frequency is fed into the second low-pass filter, a moving average filter, to get the trend of frequency data which is then subtracted from the filtered data. The signal of interest in the line trip detection is the oscillation caused by the line outage, so de-trending is necessary to remove the effects of the data trend and maintain the oscillatory



(a)



(b)



(c)

Figure 4-3 Signal outputs of each block of line trip detector
(a) Raw input (b) Median filtered frequency (c) De-trended frequency

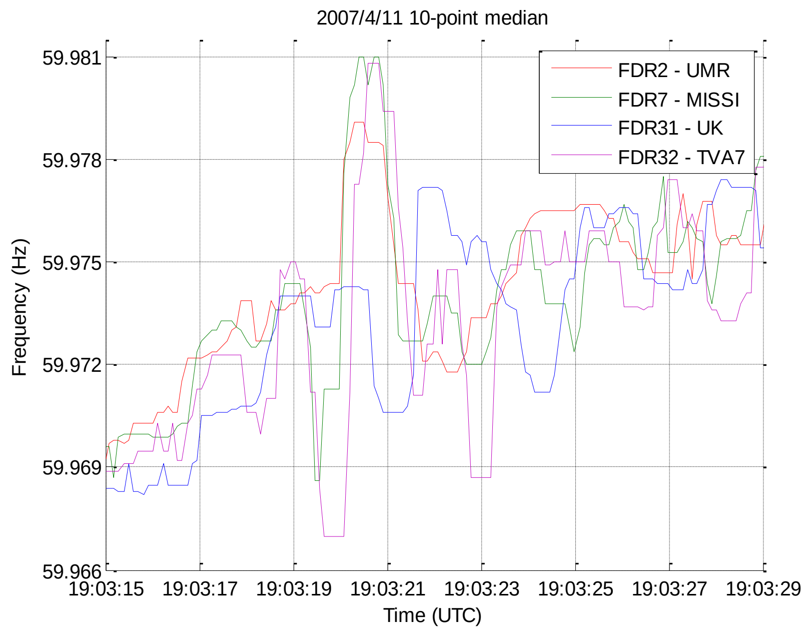
part at the same time. Figure 4-3(c) shows the de-trended data which becomes zero-mean signals and the rapidly varying components of the original frequency are clearly presented.

Two thresholds are set to detect the initial frequency peaks during the first swing of the de-trended signal. It has been observed that the oscillations caused by line trips are usually well-damped, which means that the peak values of the oscillations are descending. Within a two-second time window, if the first peak is larger than the higher threshold and the second peak exceeds the lower threshold, the algorithm will throw a trigger flag and generate reports. The two-second time window is decided based on observations from measured line trip events as well as simulation results that the period of the first swing is usually less than two seconds.

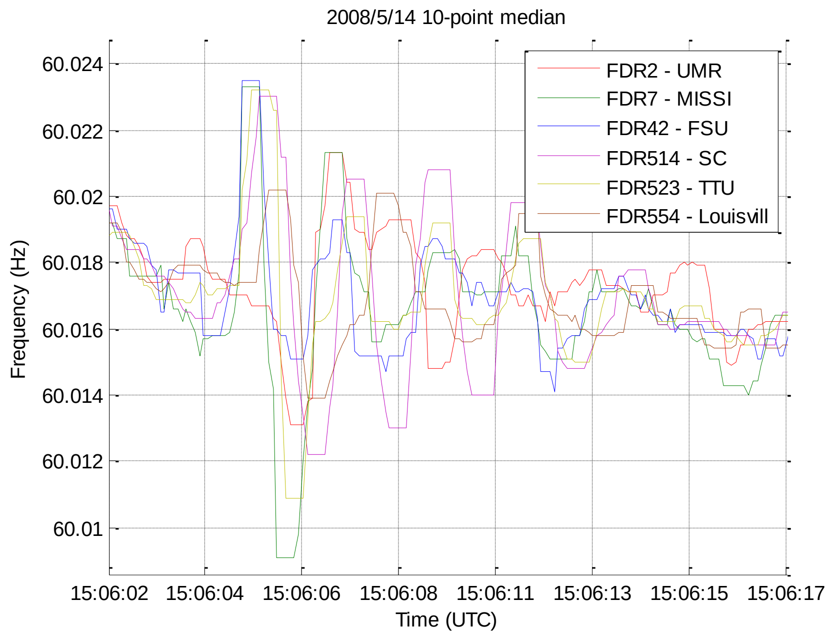
4.2 Simulation Results

The frequency based detection algorithm is programmed in MATLAB to verify its performance. The program is tested with confirmed historical line trip event data observed by FNET/GridEye. Some selected test results are demonstrated below.

Figure 4-4(a) shows the results of line a trip event that occurred on April 11, 2007, observed by four FDR units, while Figure 4-4(b) shows another event on May 14, 2008, seen by six FDR units. From the above tested results, the de-trended frequency is effective to detect line trip events if the thresholds are chosen properly. To summarize the simulation results, 48 line trip events out of 63 are detected.



(a)



(b)

Figure 4-4 Sample cases tested by Matlab simulation
(a)Line trip event on 2007/4/11 (b) Line trip event on 2008/5/14

4.3 Application Development

4.3.1 Application Architecture

This application serves as a part of the FNET/GridEye trigger application framework. The new generation of the FNET/GridEye system is based on openPDC platform, as shown in Figure 4-5.

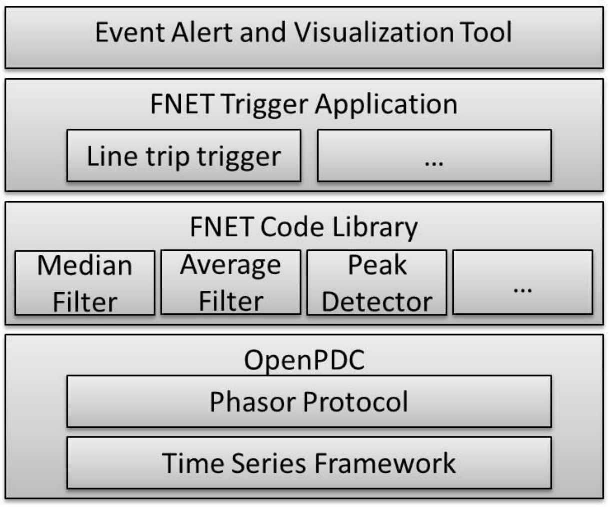


Figure 4-5 openPDC based FNET application architecture

In the architecture diagram, the openPDC provides the fundamental infrastructure for the historian database and data connections. The Time Series Framework assembles real-time data stream through TCP or UDP ports. It also provides the interface to communicate with OpenHistorian, the file based real-time database. Phasor Protocol Layer provides varies protocols to parse received data, such as BPA, IEEE C37.118 and FNET data format.

On top of the openPDC platform, a code library is developed for the FNET/GridEye with some commonly used program modules, such as median filter, average filter and peak detector. The real-time trigger applications, including line-trip trigger, are developed by assembling modules from the code library. FNET/GridEye system provides rich alert system and visualization interface, which are shared by real-time triggers.

4.3.2 Adapter Design

To implement line trip trigger application on openPDC platform, the concept of Input/Action/Output Interface (IAON) Adapter is introduced by the Time Series Framework. Each of these adapters is a base class: Input Adapters handle the functions of reading and parsing data stream, Action Adapters are in charge of data manipulation, and Output Adapters format the output data, as shown in Figure 4-6.

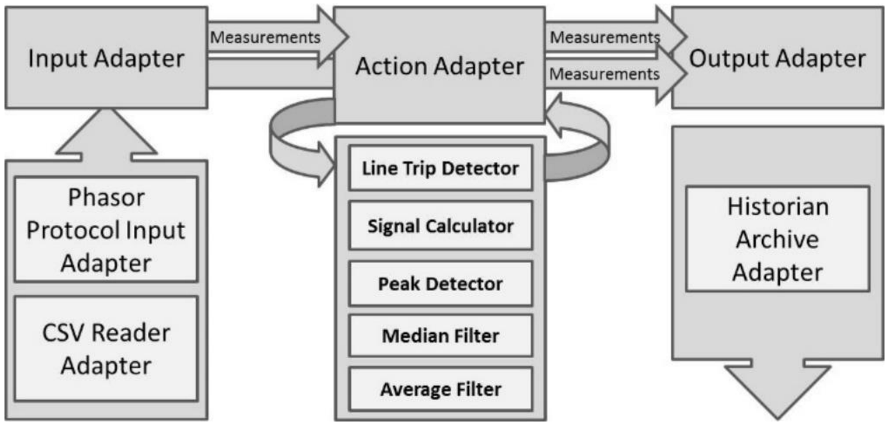


Figure 4-6 IAON Adapter layer design

The FNET data format is included in the Phasor Protocol library. For real-time trigger applications, a Phasor Protocol Input Adapter is used to parse input data stream from TCP port. It forwards the parsed data to both the Action Adapter and the Output Adapter. The Historian Archive Adapter packs the parsed raw data to the OpenHistorian format and stores it in the historian files. The CSV Reader Adapter is developed to replay historical FNET line trip event for testing purpose.

Signal processing functions mentioned in the previous section, such as peak detector and low pass filters, are implemented as Action Adapter, which is the major part of application development. The Line Trip Detector plays the role of the main function that calls the other modules and triggers the alarm. The Signal Calculator is a pre-developed adapter in openPDC library to perform simple mathematical calculation of signal measurements, such as addition and multiplication. Signal processing modules are triggered in the order given by the algorithm. Each of them processes a short interval of measurement data and sends its calculation results to the next module. Once Line Trip Detector collects the final results, it compares the results to the predefined threshold and check if the alarm should be triggered.

To enhance the flexibility of adapters, the openPDC platform enables users to define parameters as Connection Strings, in which users can specify the input signal, output signal and other critical arguments, such as threshold values, for each adapter.

Error! Reference source not found. lists some of the parameters defined for line t rip detection. Line Trip Detector class reads these parameters from the user interface, and forwards them to their destination modules. Different parameters may apply in various conditions. For instance, the thresholds may change in different interconnections. By tuning these parameters based on historical data, the accuracy of the line trip trigger can be improved such that it can reject false alarms efficiently.

Table 4-1 Connection String Properties

Name	**Type**	**Description**
detectionWindowSize	INT	Number of frames for detection (buffer size)
averageFilterSize	INT	Window length of mean filter
medianFilterSize	INT	Window length of median filter
firstPeakThreshold	FLOAT	Threshold value of the first peak during line trip
secondPeakThreshold	FLOAT	Threshold value of the second peak during line trip
minValidNodes	INT	Number of node seeing line trip event to trigger alarm

4.3.3 Alarming

Once a line trip event is detected by the peak detector module, it reports to the Line Trip Detector to trigger an alarm signal. The alarm is designed to be a binary output signal, which is obtuse to sudden change of input with a user defined delay to avoid false alarm, and hysteresis condition is considered when the alarm is cleared. As shown in Figure 4-7, the alarm is triggered when the value exceeds the threshold and stays above the threshold for a certain amount of time. The alarm is cleared when the value drops to a hysteresis point, which is lower than the alarm trigger threshold [2].

The alarm threshold is determined based on historical line trip events. The threshold Sthreshold is calculated by the following equation.

$$S_{threshold} = [\ max\ (\ |\ S_{nonLT}\ |\ ) + min\ (\ |\ S_{LT}|\ )\ ]\ /\ 2$$

where SLT is empirical signal magnitude when line trip occurs, and SnonLT is empirical signal magnitude when there is no fault event. Thus, the threshold is set to be the mean point between the minimum peak value when events occur and the maximum signal noise when there is no event.

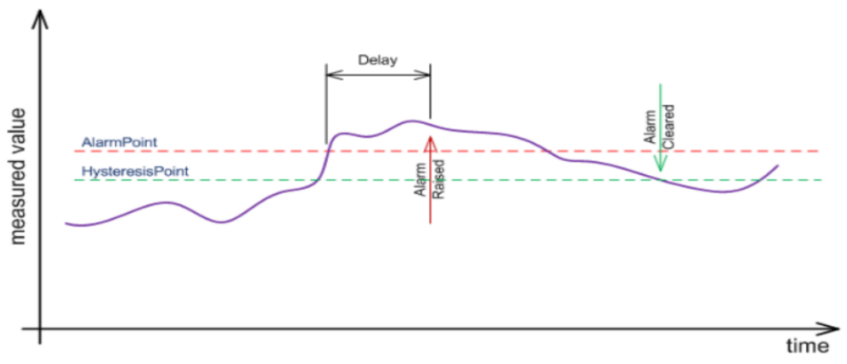


Figure 4-7 Alarm trigger mechanism

4.4 Test Results

The Application is tested with confirmed historical line trip events. Based on these historical data, some critical parameters are decided. The median filter window size is set to 7 points, while the average filter size is chosen to be 31 points. Figure 4-8 and Figure 4-9 show a sample test result on openPDC Manager, based on a line trip event observed by an FDR device near Rolla, Missouri, on March 28, 2007.

It can be seen from Figure 4-8 that the median filter effectively removed the high frequency noise, but preserved the frequency characteristics at the edges (blue trace). By subtracting the trend signal detected by the average filter (green trace), the frequency pulse change caused by the line trip event was significantly boosted from the ambient frequency (red trace). Note that the de-trended frequency (red trace) is scaled to fit in the display.

Figure 4-9 demonstrates the peak detection mechanism of the above line trip event. With the detection window set to two seconds (20 points), the peak detector continuously

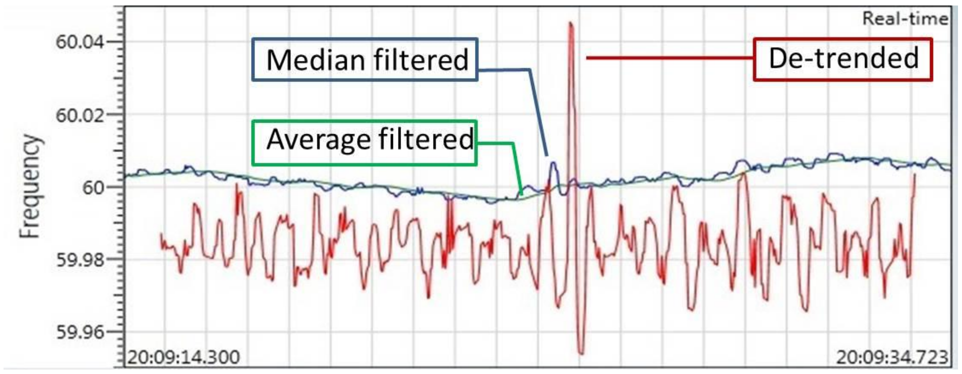


Figure 4-8 Display of filtered and de-trended frequency

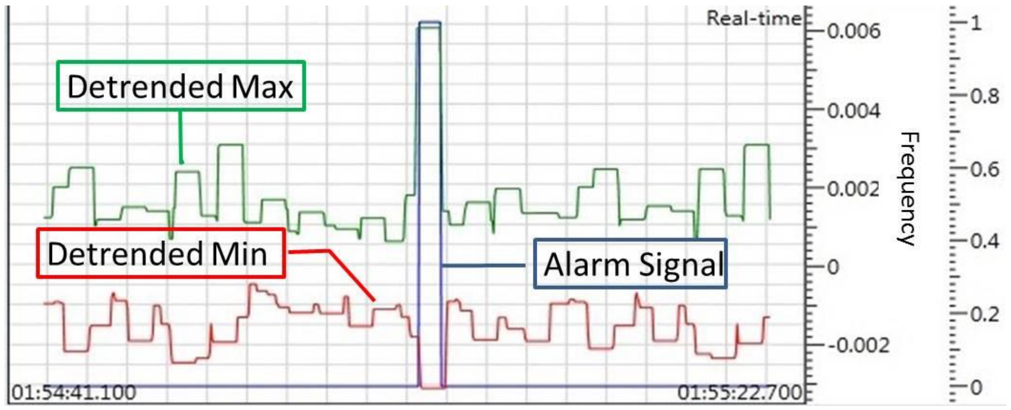


Figure 4-9 Display of alarm trigger

calculates the maximum and minimum values of the de-trended frequency within the last 2 seconds. The threshold of the first peak is set to 0.0045 Hz and that of the second peak is 0.0025 Hz based on empirical values. Once the peaks of the de-trended frequency exceed the thresholds, the binary alarm signal is triggered to 1. In this case, the maximum value reaches the first peak threshold first, and the minimum value hits the negative second peak threshold immediately after. The alarm signal is cleared after two seconds when the detection window shifts to upcoming data stream.

A warning message pops up to openPDC console with the event time stamp as soon as a line trip event is detected. Further reporting formats such as email reminder and graphic report are being developed.

4.5 Conclusion

This paper proposed a frequency based line trip detection algorithms and it is verified through actual line trip data. Utilizing the openPDC platform, the frequency-based detection algorithm is implemented as a real-time online application, which consists of median filter, average filter, signal calculator and peak detector. These modules are included in FNET/GridEye application library for other applications. Test results shows that the system effectively detects line trip events and set the proper alarms.

The angle based algorithm will be tested and implemented for line trip detection as the next step. Multi-channel detection will be considered to improve the detection accuracy and to provide the possibility to locate the event. Since line trip events often occur in companion with other faults such as generation trips or load changes. Further study will show the sensitivity of the line trip trigger and possible approaches to separate it from other events.

Chapter 5 Synchrophasor Data Quality and Security

5.1 Introduction

Synchrophasor data quality are affected by many factors. One of the major reasons for low quality data is GPS signal losses. Accurate timing source is vitally important for PMUs. Correct operation of a PMU requires a common and accurate timing reference. The timing reference is described in IEEE Std. C37.118.1-2011. To achieve a common timing reference for the PMU acquisition process, it is essential to have a source of accurate timing signals (i.e., synchronizing source) that may be internal or external to the PMU. For internal, the synchronization source is integrated into the PMU but external global positioning system (GPS) antenna still required. In the latter case, the timing signal is provided to the PMU by means of an external source, which may be local or global, and a distribution infrastructure (based on broadcast or direct connections). Within a PMU, a phase-locked oscillator is used to generate the time tags within the second. The time tag is sent out with the phasors. Thus if a phasor information packet arrives out of order to a phasor data concentrator (PDC), the phasor time response can still be assembled correctly. If the GPS pulse is not received for a while, the time tagging error may result in significant phase error.

The time system of a PMU based is depicted in Figure 5-1. The standard temporal reference of this system is generated with a signal of one pulse per second (one PPS) from a GPS. This pulse as received by any receiver on earth is coincident with all other received pulses to within 1 microsecond. PPS signal is used for sampling the analog data. The GPS time does not take into account the earth's rotation. Corrections to the GPS time are made in the GPS receivers so that they provide UTC clock time.

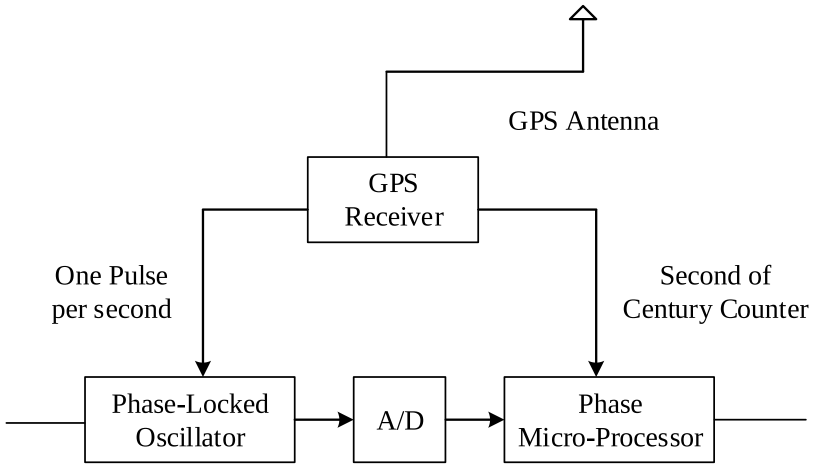


Figure 5-1 Indicator of the GPS loss and recovery flag.

This project studies the GPS loss by processing the historical PMU data from 2009 to 2012. The statistics of four-year GPS loss is implemented, in particular, the name of units, starting and ending time, and duration are recorded. Yearly, monthly and hourly loss are the selected statistical quantity. But comparison of the above quantity, we can have a clear picture of GPS loss in PMUs.

5.2 Methodology

The latest PMU/PDC protocol is the IEEE C37.118-2011 that was developed in the last few years. By monitoring the value of the 13th bit of the STATUS flag defined in the IEEE Standard C37.188.1-2011, we can get the number and duration of GPS loss, which flow chart is shown in the Figure 5-2.

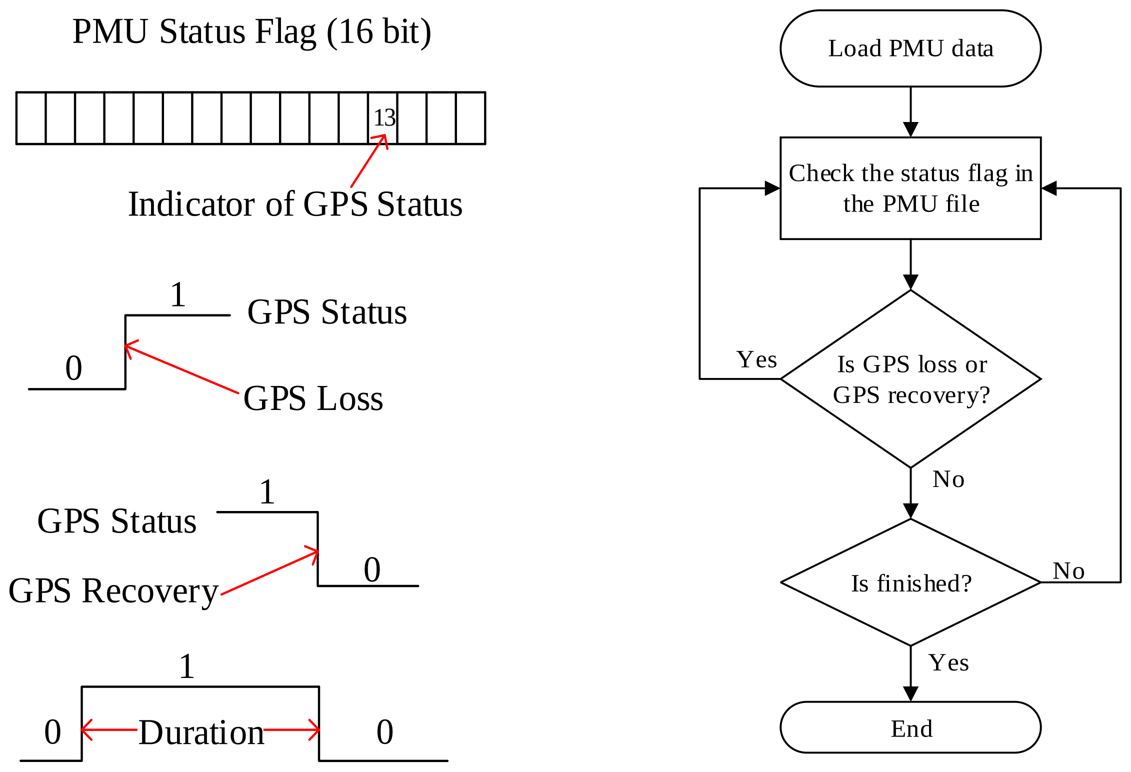


Figure 5-2 Indicator of GPS loss and how to detect a GPS loss.
(a) Indicator of the GPS loss and recovery flag.
(b) Flow chart of GPS loss detection for PMU data.

The one-bit GPS status flag is used as the indicator of GPS loss: “0”-status indicates that the PMU is with GPS timing signal whereas “0”-status means without GPS signal. So by monitoring the variance of the GPS flag, we can see when the GPS loss started and ended, and how long the loss was.

On the algorithm shown in Figure 5-2(a), a tool for detecting the GPS loss in PMU data is developed by Microsoft Visual Studio C# 2012, whose user interface is shown in Figure 5-2(b). Thus, the tool is used to obtain the statistical results of PMU GPS loss from 2009 to 2012.

FDR data does not include GPS information in every data frame. Instead, it updates the number of locked GPS satellites every 1 minute, which can be considered as the strength of the timing signal. Thus, the PMU data give the GPS loss information in higher resolution, while FDR data provide more information about GPS strength.

5.3 Temporal Patterns of GPS Losses

As the more PMUs and FDRs are deployed over the past years, more GPS loss are observed. Figure 5-4 shows the number of monthly GPS losses from 2010 to 2012. The total number of FDRs increased from 53 in Jan 2010 to 131 in Dec 2012. Over 50% of the FDRs suffer from frequent GPS Timing losses. Similarly, over 50% of the sampled PMUs detected GPS loss.

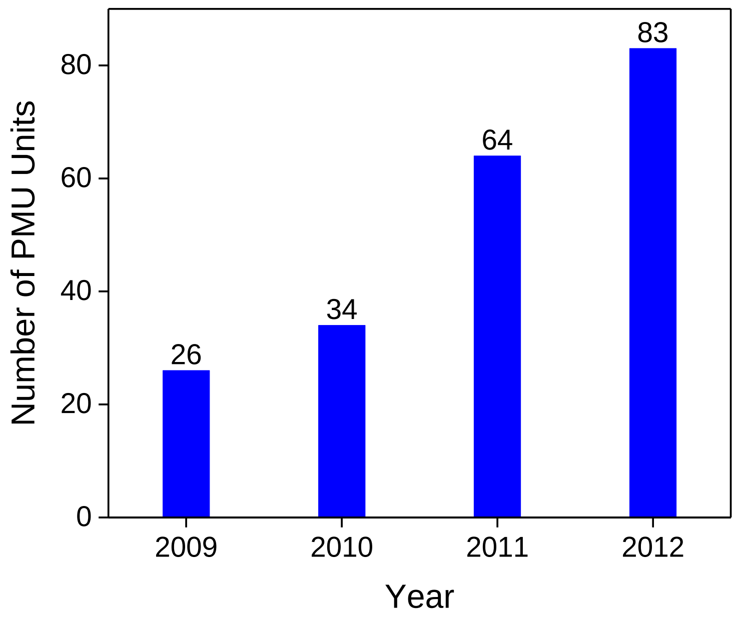


Figure 5-3 Number of PMU GPS losses from 2009 to 2012

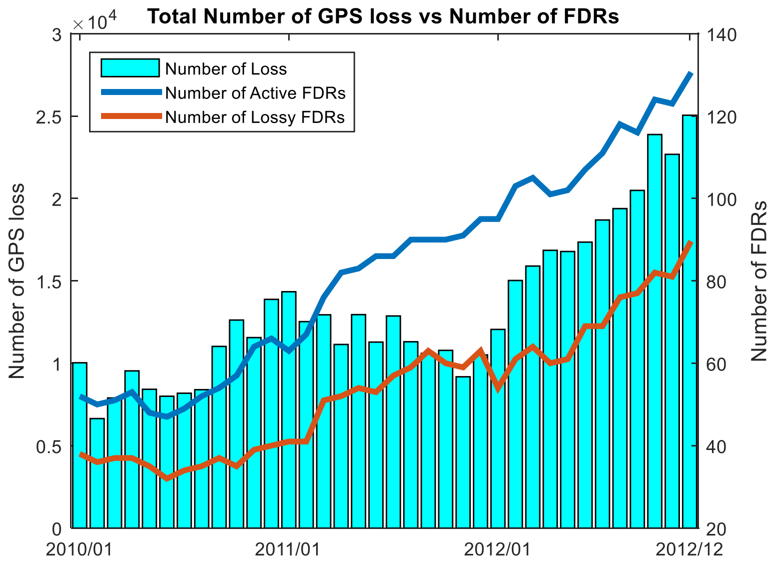


Figure 5-4 Number of FDR GPS losses from 2010 to 2012

5.3.1 Loss-recovery time

Both PMU and FDR data shows that the number of GPS losses decreases exponentially as recovery time increases. Most of GPS losses recover within a short period of time. For FDR data, since there is a forced coasting period of 1 or 2 hours, (FDR stops sending data if lose GPS timing over 1 or 2 hours), there are high count values at 60 minutes and 120 minutes.

5.3.2 Monthly Trend

Figure 5-7 and Figure 5-8 shows the trend of average monthly loss per unit of FDRs and PMUs. No obvious seasonal or monthly trend was found.

5.3.3 Daily Trend

From PMU or FDR data, we observed that some unit lost GPS more frequently in certain hours in a day. However, there is no obvious pattern detected that matches for all the units.

With the FDR data include the number of GPS satellites locked per minute, we are able to exam more details of GPS signal strength. Figure 5-11 shows the number of satellites of an FDR in days.

The four figures show four FDRs which (a) has always strong GPS signal;(b) has always weak GPS signal; (c) has a daily pattern, that tend to lost GPS at around 9AM every day; (d) has no obvious daily pattern, but never loss GPS timing.

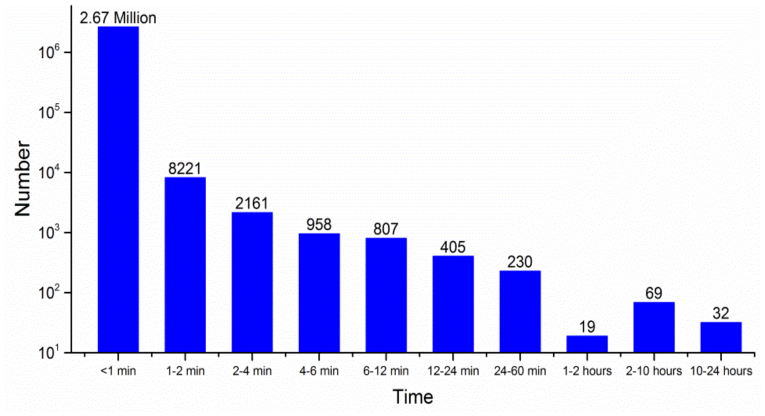


Figure 5-5 Count of GPS loss recovery time of PMU data

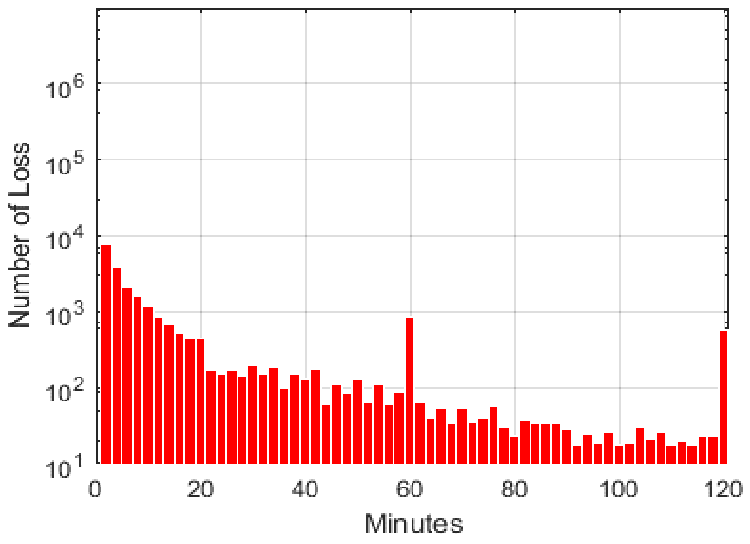


Figure 5-6 Count of GPS loss recovery time of FDR data

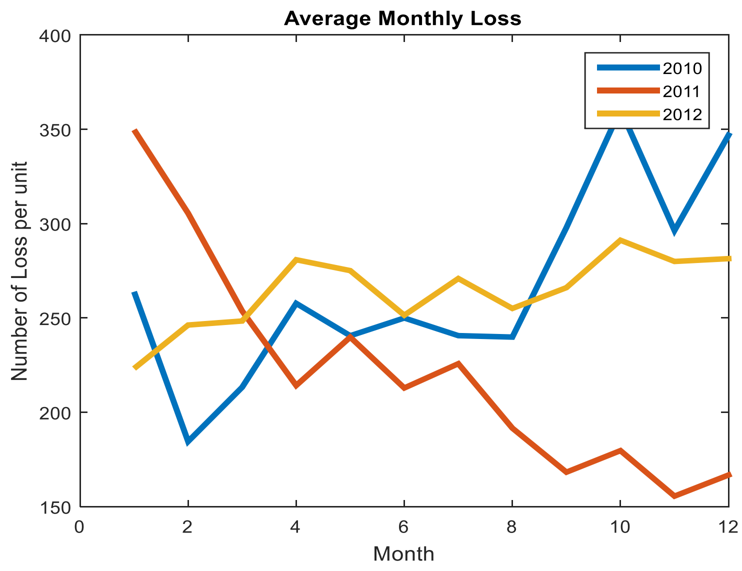


Figure 5-7 Average monthly loss per unit of FDR

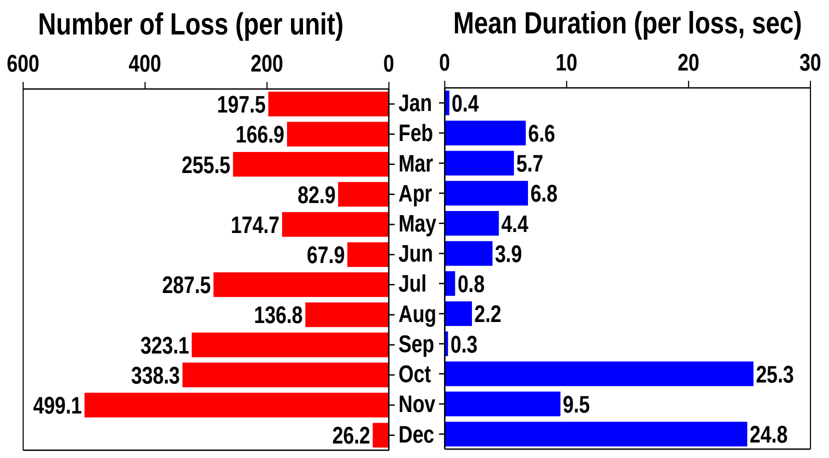


Figure 5-8 Average monthly loss per unit of PMU

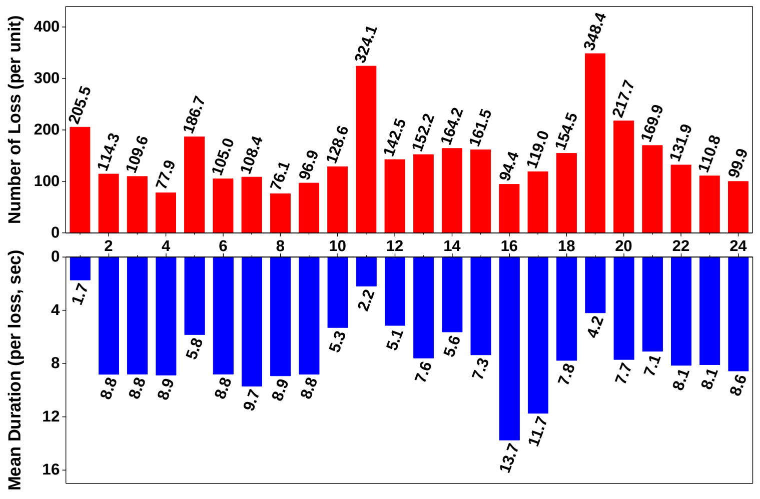


Figure 5-9 Daily GPS loss pattern per PMU

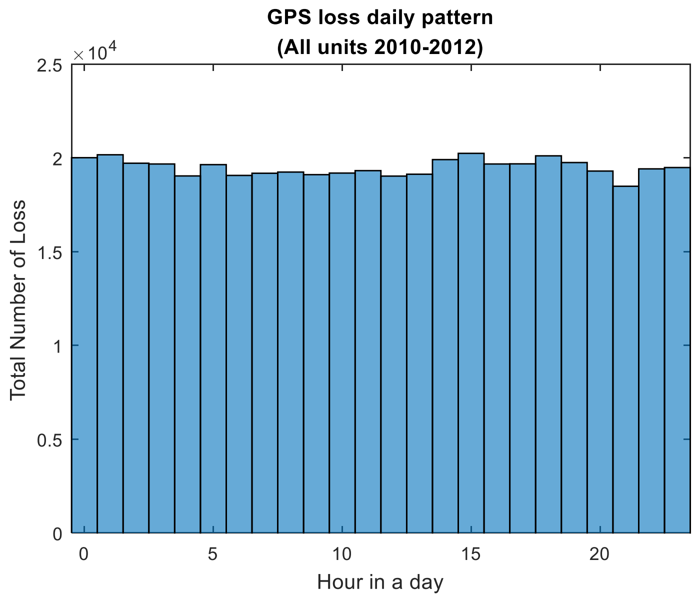


Figure 5-10 Daily GPS loss pattern of all FDRs

Since FDR need at least one GPS satellite locked to synchronize timing, the number of GPS locked can be represented as the number of backups the timing signal has, named timing signal strength. If a FDR is not locked with any GPS satellite, it losses timing synchronization. With good GPS signal reception, a receiver should be able to see 4 to 12 GPS satellites [2]. However, FDRs are usually installed indoor with a directional GPS antenna instead of an omnidirectional antenna, the GPS reception may be affected by whether the antenna is installed at a window with open view to the sky. The GPS signal may be blocked or reflected by buildings or other obstacles in certain times of the day, which can cause a daily pattern as shown above.

More analytics result from PMU data are included in Appendix.

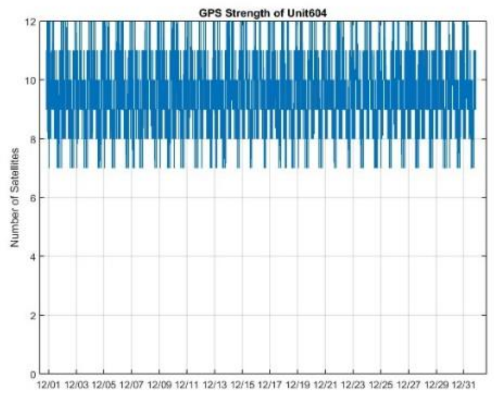
5.4 Spatial Patterns of GPS Losses

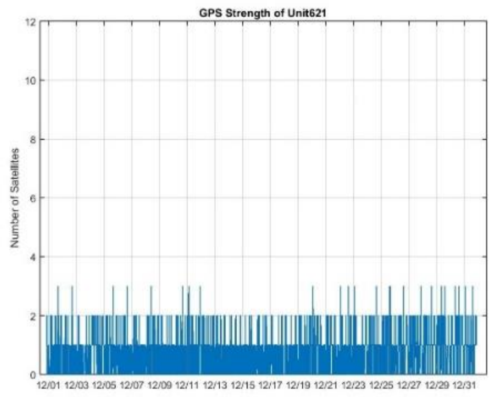
We counted the number of GPS losses of each unit across the United States, as shown in Figure 5-12. Theoretically, the GPS signal strength is affected by the latitude of the sensor location [2]. However, no significant geological pattern was observed in real data.

5.5 Impact of Weather/ Space Weather

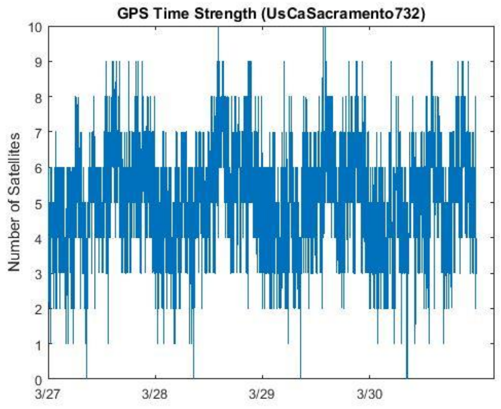
GPS signal strength is also affected by space weather, especially solar activities. [3] The largest solar activity in the last 6 years happened on March 7, 2012 00:24 UTC – the sun unleashed an X5.4-class solar flare. However, no obvious impact was detected on the overall FDR data.

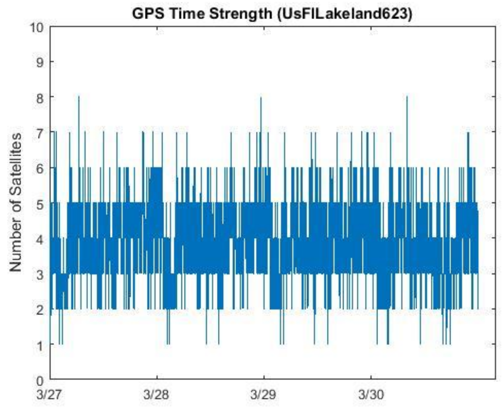
Historical weather reports are studied as well. No obvious relationship was found between GPS loss and local temperature or precipitation. Offline Experiments.





(a) (b)





(c) (d)

Figure 5-11 Daily pattern of number of satellites locked of FDRs

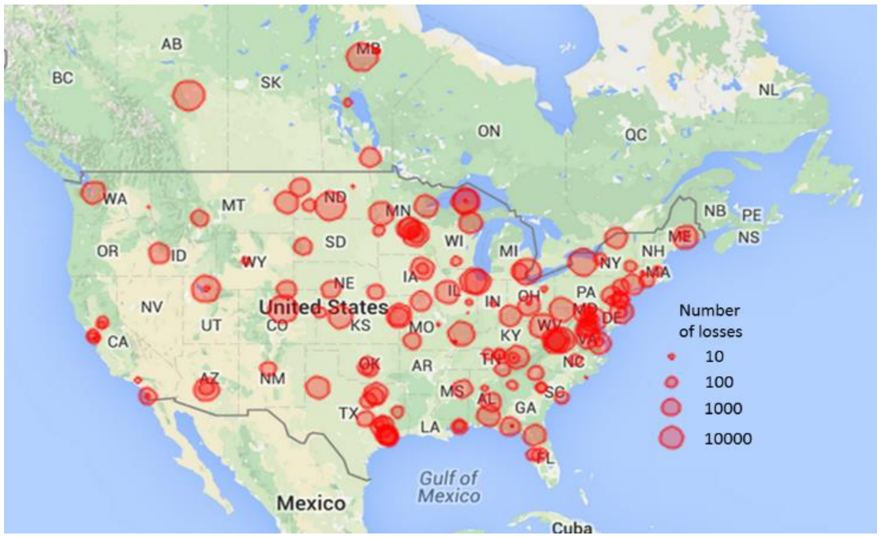


Figure 5-12 GPS losses per unit in United States

5.6 Data Security and Encryption

Data security is a critical concern in synchrophasor measurements. IEEE standard C37.118.2-2011 defines synchrophasor data transfer format and communication protocol. As the standard specifies, phasor measurement system commonly use the IP over network communication. At transport layer, either TCP or UDP or a combination of both can be used to transfer data and configuration commands [15]. However, there is no encryption method included in the standard, which exposes synchrophasor data to cyber-attacks.

To enhance the confidentiality of synchrophasor data, an encryption system is proposed by Scout Industries and tested with FNET. The system setup is demonstrated in Figure 5-15.

As shown in the Figure 5-15, a pair of hardware encryptors is added to the connection between PMU/FDR and the Phasor Data Concentrator (PDC) server. The encryptors installed for the test are Certes CEP10 modules, as shown in Figure 5-15, which enables 10 Mbps, full duplex data encryption. The CEP10 modules offer multi-layer security, including IP packet encryption for Layer 3 networks, and Layer 4 data payload encryption for IP and MPLS networks [16]. Each CEP 10 provides three ports for the network connection: PMU/FDR and PDC server are connected to Local (L) port; Remote (R) port is to transfer data over the network; Management (M) port is connected to an encryption server, TrustNet Manager. TrustNet Manager is a web based server platform to configure and manage the encryption policies, and to update encryption keys, as shown in Figure 5-15. Also the TrustNet server can be accessed from any web or command line client in the network.

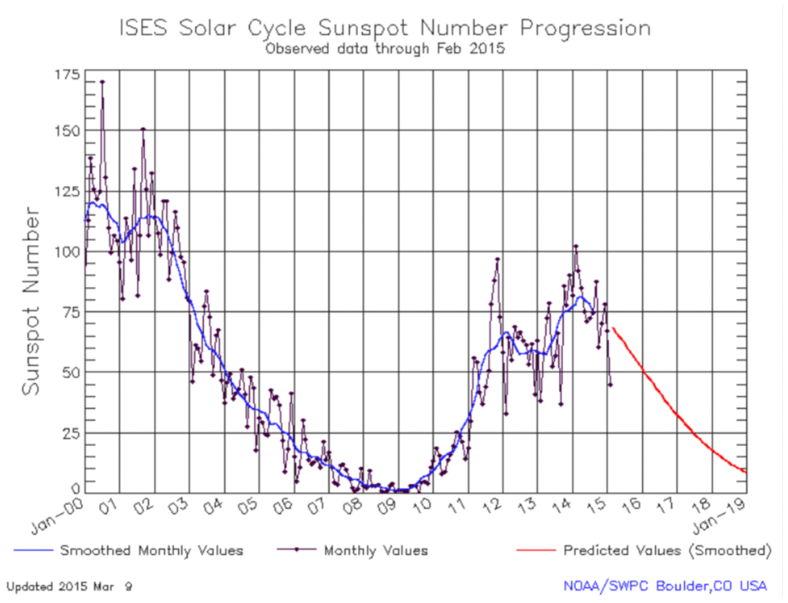


Figure 5-13 Historical trend of solar activities [5]

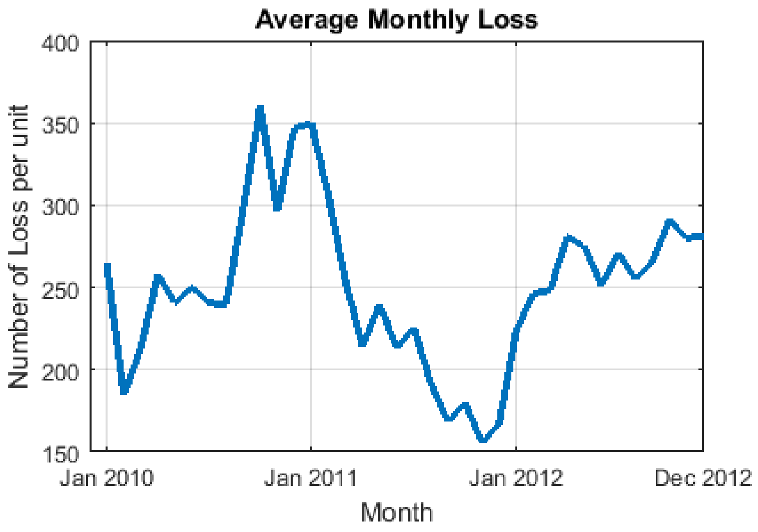


Figure 5-14 GPS loss trend from 2010 to 2012

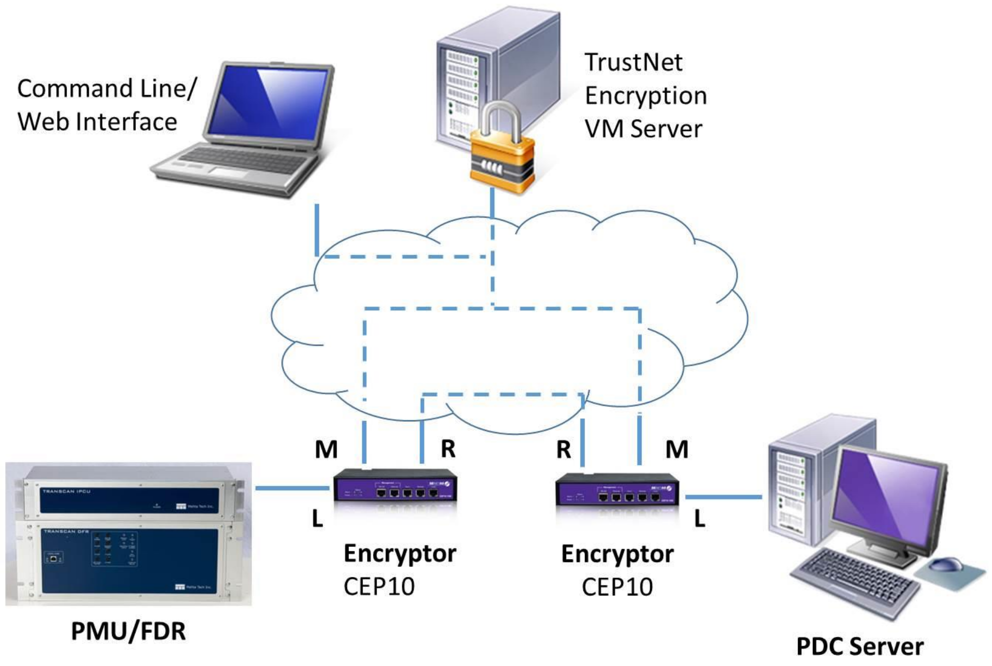


Figure 5-15 Cyber-security test system setup

The encryptors and the TrustNet server can be connected either over a local area network (LAN), or over the Internet. Phasor data is transferred without knowing the network topology or configuration. For this test, all the devices are connected over a LAN, and the encryption policy is configured to use 256-bit Advanced Encryption Standard (AES) for encryption and SHA1 for authentication. AES was announced by National Institute of Standards and Technology (NIST) in 2001 [17] and National Security Agency (NSA) approved that 256-bit AES is sufficient to protect classified information up to the Top Secret level [18]. Report shows that to brute force crack an AES 256-bit key with a state of art supercomputer, it would take 3.31×1056 years, while the age of universe is 13.75×109 years [19]. TrustNet server assigns new encryption key to the encryptors every 24 hours (this is the default value and the re-key time is configurable). With this setup, the phasor data is transferred over a secure but transparent link. The data transmission is verified by sending both single phase measurements from FDR and single channel, three phase measurements from MethaTech PMU to the OpenPDC server.

Since phasor data is intensive real-time data and sensitive to time delay, a verification test is designed to measure the encryption delay added to the communication system.

The PDC server is set to run 100 pings the PMU with 32 bytes, 64 bytes, 128 bytes messages, with and without encryption, respectively. Figure 5-16 shows the round trip latencies by pinging the PMU. The average latencies of 32 bytes, 64 bytes, 128bytes messages without encryption are 0.50ms, 0.52ms, 0.85ms respectively, while with encryption, these values increases to 0.79ms, 0.83ms and 1.56ms. The delay introduced by encryption, which is the difference between the results with and without encryption, increases as the message length grows.

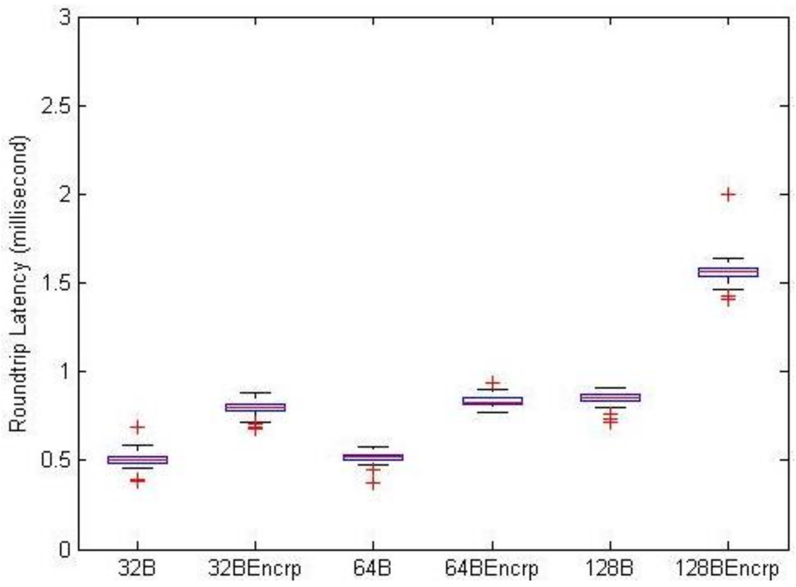


Figure 5-16 ICMP Ping latency with message length of 32, 64, 128 bytes

Since FDRs and many PMUs implements TCP protocol for data transmission, another latency test with 32 bytes TCP ping is shown in Figure 5-17. The results with encryption is 1.63ms, approximately twice of the one without encryption, 0.82ms. The TCP ping adds more latency because of its multiple-handshake scheme, which also makes the results vary in a wider range. Also, the larger TCP header, comparing to a normal ICMP ping header, contributes to further delay.

Since a single channel PMU data frame is 52 bytes long as defined in IEEE C37.118.2-2011, and a typical FDR data frame is 55 bytes in length, we expect the proposed encryption system add 1ms to 1.5ms delay in a real system. Though the total communication delay in a real system changes case by case, a typical estimation of delay from PMU to PDC is in the 20ms and 50ms range [15], which makes the encryption delay a relative small amount.

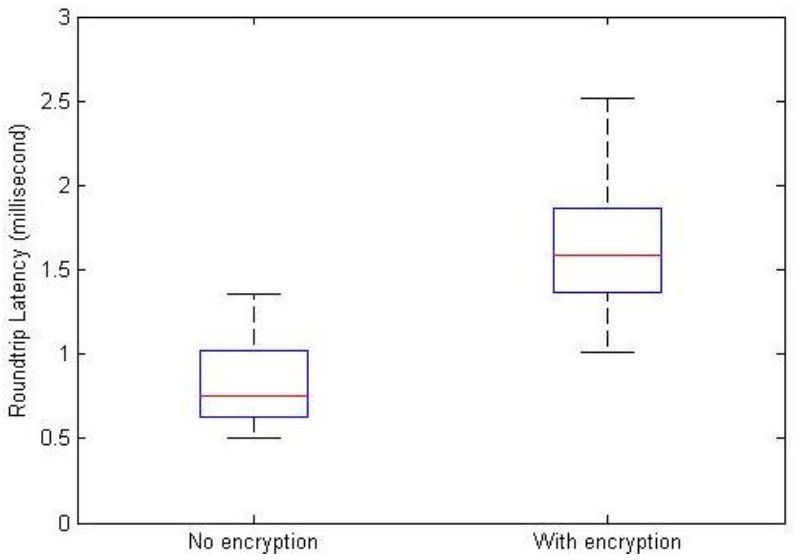


Figure 5-17 TCP Ping latency

5.7 Conclusion

By exploring the GPS loss pattern from PMU data from 2009 to 2012 and FDR data from 2010 to 2012, we found that PMUs and FDRs suffer from loss of GPS synchronization frequently, but most of the losses can recover in a few minutes. GPS loss that last longer than an hour is very rare. For FDRs, the GPS timing signal strength is most affected by antenna direction. No obvious GPS loss pattern is discovered that is related to day/month/season of the year, location, weather, or solar activity. In addition, a cyber-security solution is proposed to encrypt phasor data. Test results show that the system introduced limited encryption delay. Future work includes improvements of timing signal and data security by using additional timing source such as eLoran system as a backup for GPS system.

Chapter 6 Data Analytics Platform for Synchrophasor Data

6.1 Introduction

To enhance the capability to host large volume data for post event and statistical analytics, the Apache Spark is configured on the analytics cluster. Unlike conventional Hadoop MapReduce platform, Spark is designed to support in-memory processing, which enable it to run up to100 times faster. It also supports SQL queries, streaming data, and complex analytics such as graph algorithms and machine learning [25].

For post event and statistical data analytics, the phasor dataset are mapped to the worker nodes as Resilient Distributed Datasets (RDDs). Each of the data point can be mapped into key-value pairs, with the timestamp as the key and the data to be processed as the value. After certain processing or analysis operation, the result data are collected or reduced, first locally on each node, then globally to the master node. A sample statistical analysis of historian events is demonstrated in the next section.

Besides utilizing Spark to improve the computation speed, other data analytics tools like R and Pandas are hosted by the analytics cluster for more complicated data queries, data summary and visualization. In addition, the most commonly used data, for instance event data, are copied as snapshots in the Hadoop Distributed File System (HDFS) on the cluster nodes, to save the burden to retrieve data from historian to the analytics system.

A variety of real-time and near-real-time analytics applications hosted by the FNET/GridEye system were published since 2006. The offline applications were recently migrated to the cluster analytics platform to fully utilize the big data technologies. A sample statistical analysis using Apache Spark is demonstrated in this section.

6.2 Sample Test

The frequency disturbance event trigger detects frequency deviation caused by generation trip or load shedding. All the triggered events are logged with a timestamp and the raw event data are stored in the data historian. North American Electric Reliability Corporation (NERC) utilizes a system frequency for frequency response analysis [27]. The system frequency is calculated using the median value at each sampling time from the event data including 1 minute pre-event and 5 minutes post-event frequency. From 2012 to 2015, tens of thousands of events are captured, which makes the data extraction and conversion a challenging job.

Assuming 10000 events are to be analyzed with 100 FDR data, and the sampling frequency of FDRs is 10 per second, the total data size is 10000* 100 FDRs *10 per second * 360 second*4bytes = 24 Gigabyte. At each time point of 0.1 second, the frequency value of 100 FDRs are to be sorted to obtain the median point.

The distributed data analytics platform with Apache Spark provides a solution to process such amount of data in parallel. Each of the frequency data and is according timestamp are packed as key-value pairs, and loaded as Spark RDDs. The RDD will first map the data partitions to computation nodes for processing, and let each nodes execute the sorting function in parallel. Then the calculated median values are sorted by their timestamps, to form a frequency time series for each event. Eventually, the time series are grouped by their event id and are aggregated as final result to the master node. Unlike traditional MapReduce method, Spark handles the whole process in memory, which ensures the computation speed to be significantly faster.

6.3 Performance

The test environment is setup using a server system with two Intel Xeon E5-2470v2 processors, which includes 20 cores with hyper-threading. The test results show that for 100 events, the least computation time is about 128 milliseconds, while without the Spark implementation, the computation time is 2592 millisecond, which is over 20 times slower.

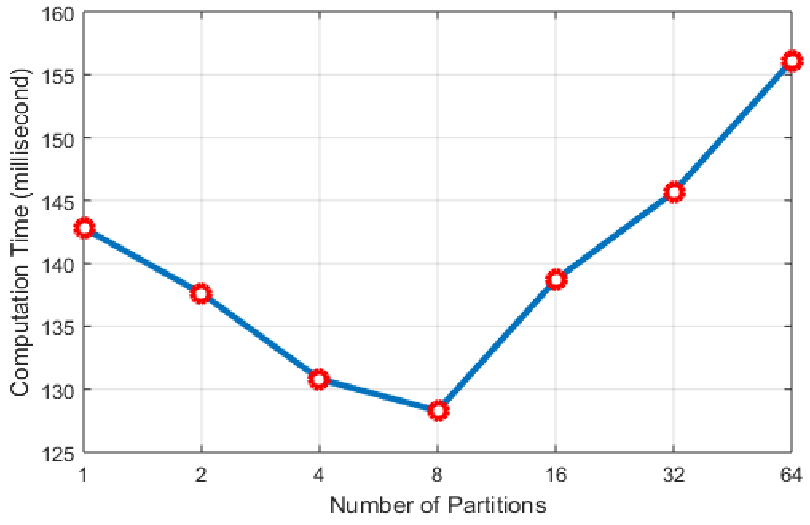


Figure 6-1 Computation Time vs. Number of Partitions.

Increasing the number of partitions of the dataset may improve the performance by processing data in a more parallel cluster. However, the communication overhead to map and collect the dataset will also increase. Figure 6-1 demonstrates the computation time using different partitions of the dataset. In this case, the least computation time is achieved with 8 partitions. Using more than 8 partitions will increase the computation time because of communication.

6.4 Test Results

To further investigate the frequency response of events, the time of occurrence of the events are calculated using the distributed platform, to learn if it matches a daily or seasonal variation pattern [28].

To count the events happened by their hour of the day in each month, the timestamps of events in Eastern Interconnection (EI) from July 2012 to July 2015 are stored in a Python list structure. The timestamps are then converted to a Spark RDD using PySpark, and grouped by their month and hour segment, as shown in the following Python program.

The number events with the same hour within each month is counted, and the results is plotted as a colored contour map in Figure 6-2, with the X axis as the months, Y axis as the hours of the day (UTC time), and the color representing the total number of events counted within the time range.

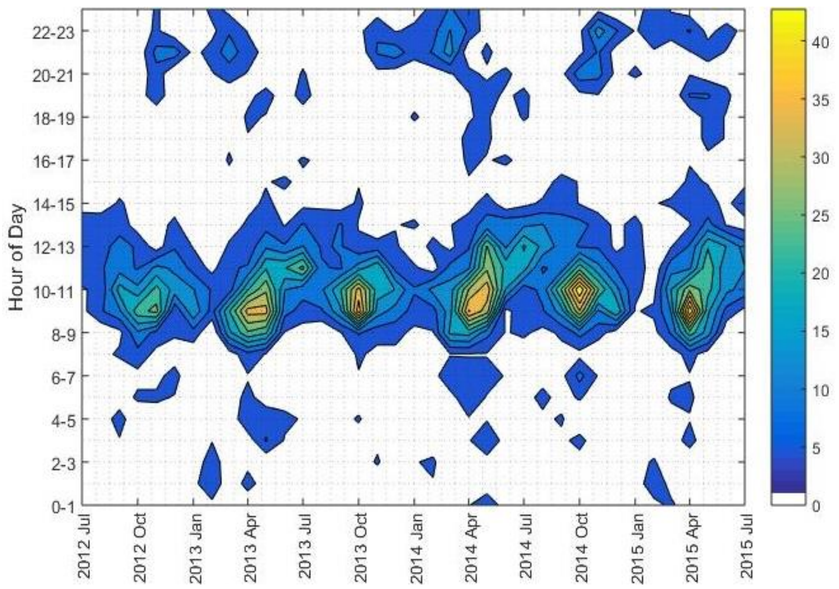


Figure 6-2 Count of EI events by month by hour of the day

Since most of the events are congregated at 7:00 to 14:00 UTC, it is suspected that some scheduled operation is one of the major causes of events. To prove this hypothesis, generation trip events and load shedding events are plotted separately in Figure 6-3 and Figure 6-4.

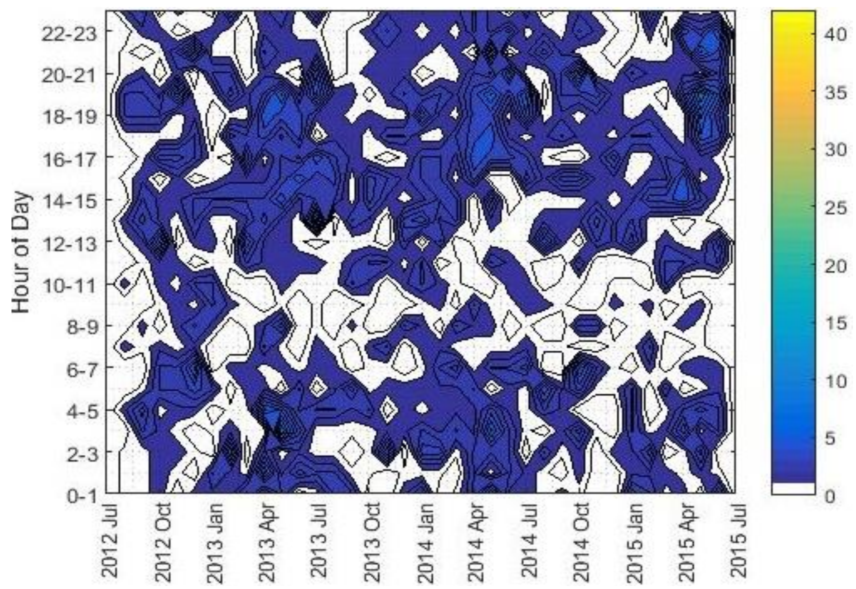


Figure 6-3 Count of EI generator trip events by month by hour of the day

The generation trip events are random distributed over a day, and there is no obvious pattern over a year. However, the load shedding events are concentrated around 9:00 and 21:00 UTC (4:00 and 16:00 EST). It confirms that most of the events are scheduled load shedding or pump storage shut down. Also, Figure 6-4 shows the peaks of the number of load shedding events appears at April and October each year, but very few load sheddings are captured in summer or winter.

To further compare events between Eastern and Western Interconnections, the events data in WECC is analyzed with the same procedure. Similarly, the generation trip events are randomly distributed. However, most the load shedding events in WECC are laid between 10:00 to 17:00 UTC, as shown in Figure 6-5. It implies that the utilities in WECC may have one scheduled time per day for load shedding or storage shut down. The peak positions of load shedding indicate that these operations are usually 2 to 3 hours later than the load sheddings in EI, possibly caused by the local time difference.

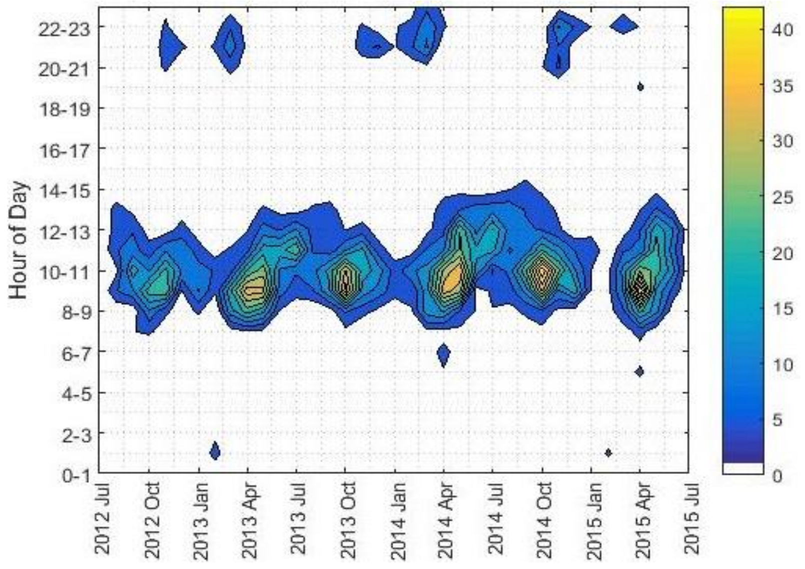


Figure 6-4 Count of EI load shedding events by month by hour of the day

6.5 Conclusion

The computation power is enhanced by parallelizing analysis algorithms, distributing the computation load across multiple nodes, and utilizing big data analysis tools like Spark. Large volume historian data are easily accessible to for applications and external clients. Based on this platform, more sophisticated data analytics algorithms and visualizations tools can be easily implemented for power system monitoring, operation and control.

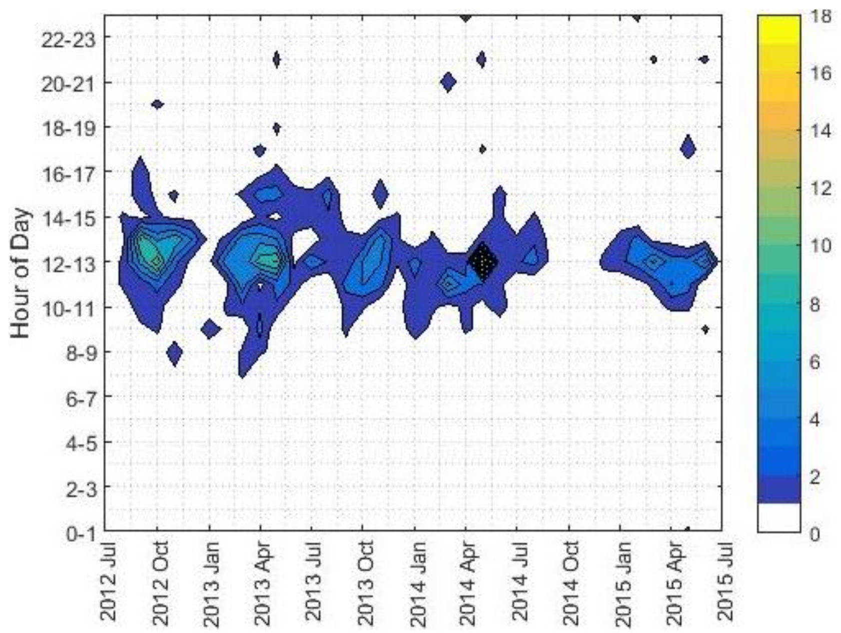


Figure 6-5 Count of WECC load shedding events by month by hour of the day

Chapter 7 Conclusions

The wide area synchrophasor data server system and data analytics platform discussed in this dissertation are designed and implemented based on FNET/GridEye system. The system provides a comprehensive solution for synchrophasor data collection, real-time data application, data quality monitoring and big data analytics. It collects distribution level synchrophasor data from FDRs, stores the data in openHistorian database, and connected with data analytics systems with Spark. Many real-time and offline applications have been developed to monitor the system status of the North American transmission system.

The data collection server system is implemented using TCP sockets. The server is utilized to receive data, store in a local database, and forward data to multiple other server systems. The FDR configuration is synchronized across servers using MySQL database. The timestamp shift error introduced by FDR clock mismatch was corrected.

The real-time applications are divided into multiple tiers based on their time sensitivity. These applications are implemented using openPDC based action adapters. Real-time data are stored in openHistorian system, and a trigger processing interface is introduced to connected real-time disturbance detection, historian data extraction and near-real-time data analysis. A line trip detection application is developed to demonstrate the process.

A data analytics platform is proposed to handle analytics jobs that involves big volume of historian data. The analytics platform is built on a virtual cluster, and Apache

Spark in implemented for data processing. Test results shows that the computation performance is increased by tens of times.

Synchrophasor data quality, including GPS losses and data security issues are also discussed in this dissertation. Both temporal and spatial pattern of GPS losses are investigated.

Based on the architecture proposed in this dissertation, more real-time applications, including but not limited to forced oscillation detection, FIDVR detection, islanding detection, can be easily implemented. Future works may also focus on data analytics using the data analytics platform.

This system improves the scalability, extensibility, concurrency and robustness of the FNET/GridEye system. It not only boosted the performance of synchrophasor applications, but also provides a reference for industrial level wide area synchrophasor monitoring system.

List of References

[1] M.D. Hadley, J.B. McBride, T.W. Edgar, L.R. O’Neil, and J.D. Johnson, “Securing Wide Area Measurement Systems,” Jun-2007.[Online]. Available: http://energy.gov/oe/downloads/securing-wide-area-measurement-systems.

[2] C. E. Otero and A. Peter, “Research Directions for Engineering Big Data Analytics Software,” IEEE Intell. Syst., vol. 30, no. 1, pp. 13–19, Jan. 2015.

[3] M. Kezunovic, L. Xie, and S. Grijalva, “The role of big data in improving power system operation and protection,” in 2013 IREP Symposium Bulk Power System Dynamics and Control - IX Optimization, Security and Control of the Emerging Power Grid, 2013, pp. 1–9.

[4] D. Wang and L. Xiao, “Storage and Query of Condition Monitoring Data in Smart Grid Based on Hadoop,” in 2012 Fourth International Conference on Computational and Information Sciences, 2012, pp. 377–380.

[5] M. Edwards, A. Rambani, Y. Zhu, and M. Musavi, “Design of Hadoop-based Framework for Analytics of Large Synchrophasor Datasets,” in Procedia Computer Science, 2012, vol. 12, pp. 254–258.

[6] J. Zheng and A. Dagnino, “An initial study of predictive machine learning analytics on large volumes of historical data for power system applications,” 2014, pp. 952–959.

[7] F. Bach, H. K. Cakmak, H. Maass, and U. Kuehnapfel, “Power Grid Time Series Data Analysis with Pig on a Hadoop Cluster Compared to Multi Core Systems,” in 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2013, pp. 208–212.

[8] M. Khan, P. M. Ashton, M. Li, G. A. Taylor, I. Pisica, and J. Liu, “Parallel Detrended Fluctuation Analysis for Fast Event Detection on Massive PMU Data,” IEEE Trans. Smart Grid, vol. 6, no. 1, pp. 360–368, Jan. 2015.

[9] Y. Liu, L. Zhan, Y. Zhang, P. N. Markham, D. Zhou, J. Guo, Y. Lei, G. Kou, W. Yao, J. Chai, and Y. Liu, “Wide-Area Measurement System Development at the Distribution Level: an FNET/GridEye Example,” IEEE Trans. Power Deliv., vol. PP, no. 99, pp. 1–1, 2015.

[10] Y. Zhang, P. Markham, T. Xia, L. Chen, Y. Ye, Z. Wu, Z. Yuan, L. Wang, J. Bank, J. Burgett, R. W. Conners, and Y. Liu, “Wide-Area Frequency Monitoring Network (FNET) Architecture and Applications,” IEEE Trans. Smart Grid, vol. 1, no. 2, pp. 159–167, 2010.

[11] J. Bank, R. Gardner, J. Wang, A. Arana, and Y. Liu, “Generator Trip Identification Using Wide-Area Measurements and Historical Data Analysis,” in 2006 IEEE PES Power Systems Conference and Exposition, 2006, pp. 1677–1681.

[12] Zhiyong Yuan, Tao Xia, Yingchen Zhang, Lang Chen, P. N. Markham, R. M. Gardner, and Yilu Liu, “Inter-area oscillation analysis using wide area voltage angle measurements from FNET,” in IEEE PES General Meeting, 2010, pp. 1–7.

[13] J. Guo, Y. Zhang, M. A. Young, M. J. Till, A. Dimitrovski, Y. Liu, P. Williging, and Y. Liu, “Design and Implementation of a Real-Time Off-Grid Operation Detection Tool from a Wide-Area Measurements Perspective,” IEEE Trans. Smart Grid, vol. 6, no. 4, pp. 2080–2087, Jul. 2015.

[14] D. Zhou, Y. Liu, and J. Dong, “Frequency-based real-time line trip detection and alarm trigger development,” in 2014 IEEE PES General Meeting, 2014, pp. 1–5.

[15] Y. Zhang, L. Chen, Y. Ye, P. Markham, J. Bank, J. Dong, Z. Yuan, Z. Lin, and Y. Liu, “Visualization of wide area measurement information from the FNET system,” in 2011 IEEE Power and Energy Society General Meeting, 2011, pp. 1–8.

[16] W. Li, J. Tang, J. Ma, and Y. Liu, “Online Detection of Start Time and Location for Hypocenter in North America Power Grid,” IEEE Trans. Smart Grid, vol. 1, no. 3, pp. 253–260, 2010.

[17] Y. Ye and Y. Liu, “Monitoring power system disturbances based on distribution-level phasor measurements,” in 2012 IEEE PES Innovative Smart Grid Technologies (ISGT), 2012, pp. 1–8.

[18] Y. Liu, Z. Yuan, P. N. Markham, R. W. Conners, and Y. Liu, “Application of Power System Frequency for Digital Audio Authentication,” IEEE Trans. Power Deliv., vol. 27, no. 4, pp. 1820–1828, 2012.

[19] T. Xia, Y. Zhang, L. Chen, Z. Yuan, P. N. Markham, Y. Ye, and Y. Liu, “Phase angle-based power system inter-area oscillation detection and modal analysis,” Eur. Trans. Electr. Power, vol. 21, no. 4, pp. 1629–1639, May 2011.

[20] IEEE Standard for Synchrophasor Data Transfer for Power Systems, in IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005) , vol., no., pp.1-53, Dec. 28 2011.

[21] Grid Protection Alliance, “openPDC Documentation.” [Online]. Available: http://openpdc.codeplex.com/. [May 7, 2015]

[22] Zhan, L.; Zhou, D.; King, T.; Liu, Y.; Johannessen, E.; Alexander, J.; Boza, B., "Improvement of timing reliability and data transfer security of synchrophasor measurements," in T&D Conference and Exposition, 2014 IEEE PES , vol., no., pp.1-5, 14-17 April 2014

[23] J. Gray, “The Transaction Concept: Virtues and Limitations (Invited Paper),” in Proceedings of the Seventh International Conference on Very Large Data Bases, 1981, pp. 144–154.

[24] Grid Protection Alliance, , “openHistorian Documentation.” [Online]. Available: http://openhistorian.codeplex.com/. [May 7, 2015]

[25] Apache Spark, “Apache Spark Documentation.” [Online]. Available: http://spark.apache.org/.[Oct. 15, 2015]

[26] FNET/GridEye, “FNET/GridEye Event Video Report.” [Online]. Available: https://goo.gl/fqCEDF. [Feb. 20, 2016]

[27] R. Quint, P. V. Etingov, D. Zhou, and D. Kosterev, “Frequency Response Analysis using Automated Tools and Synchronized Measurements,” in IEEE PES General Meeting, 2016.(Accepted)

[28] J. Bian, “Generator Frequency Response," [Online]. Available: https://goo.gl/scnsCK. [Aug. 10, 2015]GPA, “openPDC documentation” [Online]. Available: http://openpdc.codeplex.com/documentation

[29] J. Dong, “Power System Disturbance Analysis and Detection Based on Wide-Area Measurements” Ph.D. dissertation, Dept. ECE, Virginia Polytechnic Institute and State Univ. Blacksburg, VA, 2008.

[30] J. Dong, J. Zuo, L. Wang, K. S. Kook, I.-Y. Chung, Y. Liu, S. Affare, B. Rogers, and M. Ingram, “Analysis of Power System Disturbances Based on Wide-Area Frequency Measurements,” in IEEE Power Engineering Society General Meeting, 2007, 2007, pp. 1–8.

[31] J. E. Tate and T. J. Overbye, “Line Outage Detection Using Phasor Angle Measurements,” IEEE Transactions on Power Systems, vol. 23, no. 4, pp. 1644–1652, Nov. 2008.

[32] IEEE Standard for Synchrophasors Measurement for Power Systems, IEEE Std C37.118.1-2011, Dec. 2011. (Revision of IEEE Std. C37.118-2005).

[33] IEEE Standard for Synchrophasors Data Transfer for Power Systems, IEEE Std C37.118.2-2011, Dec. 2011. (Revision of IEEE Std. C37.118-2005).

[34] Certes Networks. Internet Encryption. [Online]. Available: http://certesnetworks.com/

[35] Announcing the ADVANCED ENCRYPTION STANDARD (AES). Federal Information Processing Standards Publication 197. United States National Institute of Standards and Technology (NIST). November 26, 2001. Retrieved October 2, 2012.

[36] L. Hathaway. “National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information”, 2003, Retrieved 2011.

[37] M. Arora. (2012) How secure is AES against brute force attacks? [Online], Available :http://www.eetimes.com/document.asp?doc_id=1279619&

Appendix

PMU GPS Losses from 2009 to 2012

1. Results of 2009

Table A-1 Units with GPS loss in 2009

Name	Number	Gross Duration	Mean Time (per unit per loss)
-	10384.42	123724.86	11.91
-	7935.31	1398.27	0.18
-	4772.12	1941.13	0.41
-	3655.69	1706.46	0.47
-	2772.82	2915.09	1.05
-	2362.73	3085.25	1.31
-	2267.05	2880.29	1.27
-	247.49	97.78	0.40
-	169.16	180.18	1.07
-	59.12	5650.72	95.59
-	44.24	45.72	1.03
-	41.67	43.06	1.03
-	35.77	6.54	0.18
-	30.99	4.32	0.14
-	30.15	67.35	2.23

Table A-2 Monthly GPS loss in 2009

Mont h	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
Jan	7.00	109.71	107.58	0.98
Feb	9.00	171.74	2696.46	15.70
Mar	12.00	651.41	4026.09	6.18
Apr	10.00	80.85	390.98	4.84
May	10.00	45.22	65.33	1.44
Jun	11.00	161.45	678.12	4.20
Jul	8.00	221.44	130.67	0.59
Aug	13.00	322.42	173.28	0.54
Sep	9.00	22.50	9.92	0.44
Oct	6.00	19.83	8.16	0.41
Nov	4.00	30.13	11.04	0.37
Dec	2.00	4.83	5.00	1.03

Table A-3 Hourly GPS loss in 2009

Hour (UTC)	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
1	10	441.16	984.60	2.23
2	11	208.57	668.04	3.20
3	11	185.37	701.96	3.79
4	11	89.49	55.25	0.62
5	11	80.23	52.57	0.66
6	10	138.95	595.34	4.28
7	8	217.08	820.59	3.78
8	10	114.37	130.73	1.14
9	10	99.97	86.29	0.86
10	11	92.27	51.29	0.56
11	11	298.53	109.51	0.37
12	13	111.32	459.70	4.13
13	13	183.67	642.05	3.50
14	11	228.92	618.79	2.70
15	12	264.70	1180.75	4.46
16	14	109.07	1671.95	15.33
17	10	219.00	1470.37	6.71
18	13	296.25	1509.48	5.10
19	12	235.34	2054.05	8.73
20	12	315.28	1676.57	5.32
21	13	308.73	1099.39	3.56
22	11	289.11	1434.59	4.96
23	12	192.24	720.42	3.75
24	9	244.63	859.27	3.51

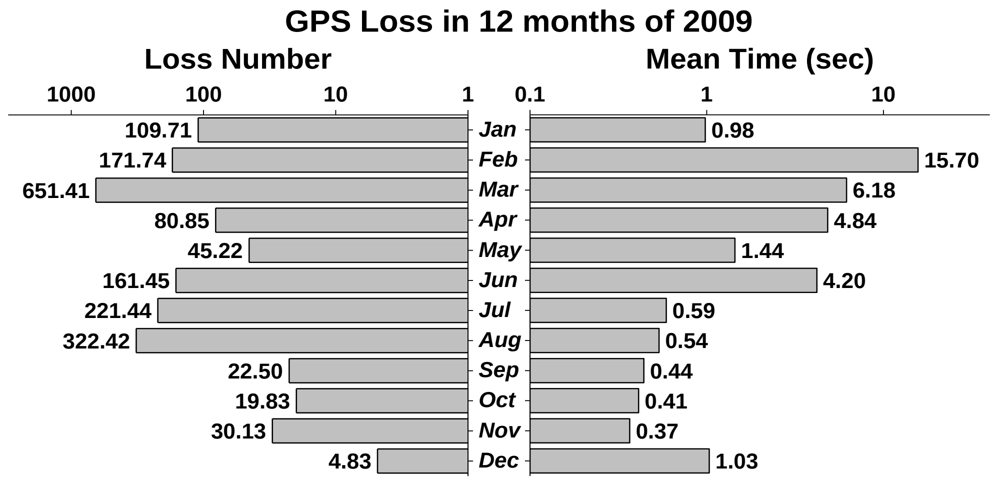


Figure A-1 Monthly GPS loss in 2009.

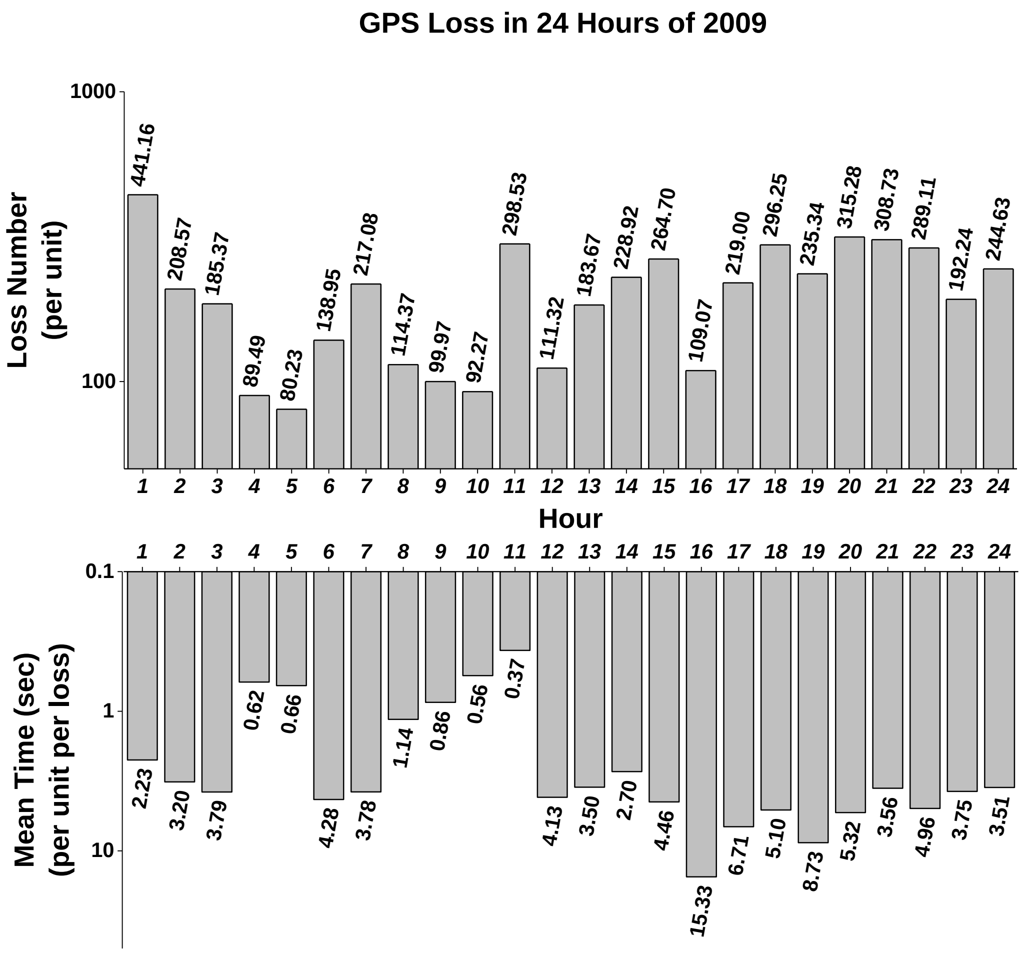


Figure A-2 Hourly GPS loss in 2009.

2. *Results of 2010*

Table A-4 Units with GPS loss in 2010

Name	Number	Gross Duration	Mean Time (per unit per loss)
-	3126.58	6418.68	2.05
-	2440.41	260.72	0.11
-	1860.44	3830.15	2.06
-	1705.83	3832.31	2.25
-	1591.33	3484.63	2.19
-	1465.80	3123.41	2.13
-	1220.73	5384.25	4.41
-	1198.87	5516.73	4.60
-	542.40	4316.89	7.96
-	508.84	20868.27	41.01
-	417.39	530.10	1.27
-	407.21	544.65	1.34
-	260.02	5368.02	20.64
-	257.72	3756.83	14.58
-	193.70	53.73	0.28
-	130.97	31.37	0.24
-	77.24	239.44	3.10
-	64.51	66.67	1.03
-	48.98	50.61	1.03
-	40.95	595.59	14.54
-	37.06	38.29	1.03

Table A-5 Monthly GPS loss in 2010

Month	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
Jan	9	23.11	28.15	1.22
Feb	8	230.65	30.91	0.13
Mar	8	27.00	33.35	1.24
Apr	19	112.93	1045.54	9.26
May	17	407.93	2013.77	4.94
Jun	6	13.93	100.75	7.23
Jul	15	110.89	158.94	1.43
Aug	16	79.67	246.59	3.10
Sep	14	132.25	130.62	0.99
Oct	15	46.57	17.60	0.38
Nov	7	16.07	18.29	1.14
Dec	6	21.95	25.02	1.14

Table A-6 Hourly GPS loss in 2010

Hour (UTC)	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
1	4	22.63	75.68	3.34
2	11	26.90	13.26	0.49
3	10	46.76	15.38	0.33
4	11	31.39	17.72	0.56
5	19	307.33	1330.77	4.33
6	11	31.56	16.85	0.53
7	9	35.94	51.57	1.43
8	14	27.88	16.14	0.58
9	9	48.43	38.13	0.79
10	11	43.56	33.38	0.77
11	16	34.19	28.06	0.82
12	15	75.28	87.74	1.17
13	15	33.27	39.46	1.19
14	13	67.95	469.67	6.91
15	14	51.47	504.01	9.79
16	15	70.82	569.38	8.04
17	12	64.53	592.55	9.18
18	16	62.36	485.82	7.79
19	15	287.89	233.25	0.81
20	19	236.15	1243.32	5.26
21	16	41.82	326.63	7.81
22	14	60.05	318.46	5.30
23	9	33.47	26.06	0.78
24	11	15.82	9.44	0.60

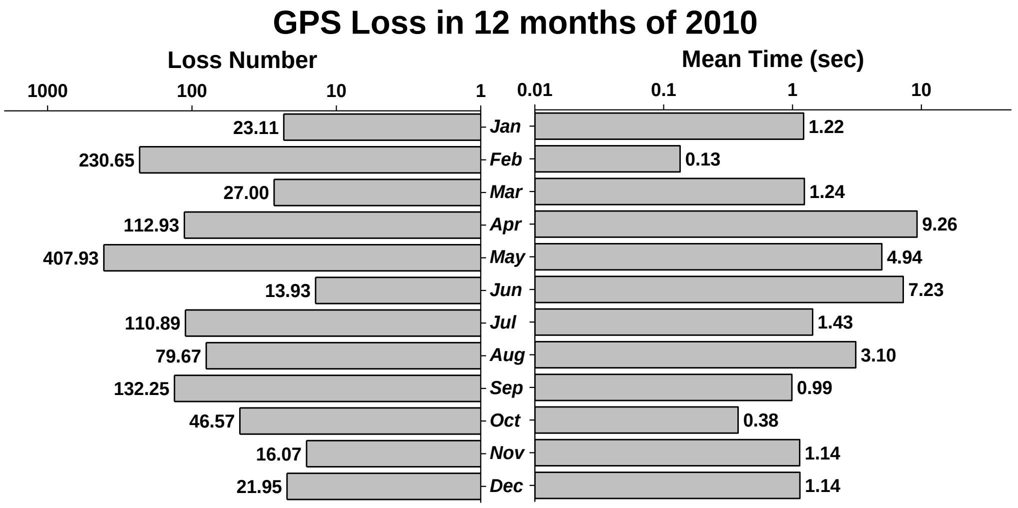


Figure A-3 Monthly GPS loss in 2010.

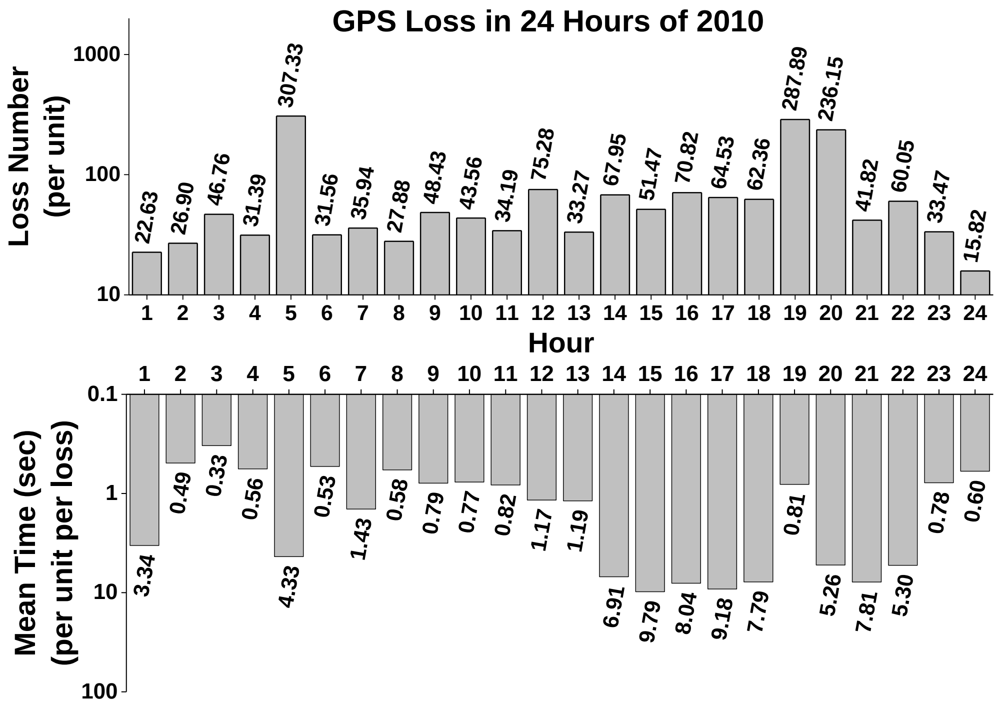


Figure A-4 Hourly GPS loss in 2010.

3. ***Results of 2011***

Table A-7 Units with GPS loss in 2011

Name	Number	Gross Duration	Mean Time (per unit per loss)
-	23447.20	466785.61	19.91
-	5078.26	5426.23	1.07
-	866.95	5515.52	6.36
-	448.66	599.69	1.34
-	447.34	598.68	1.34
-	177.61	11.27	0.06
-	151.25	8.86	0.06
-	123.04	131.31	1.07
-	109.97	5.25	0.05
-	98.01	5.66	0.06
-	92.66	6.24	0.07
-	90.85	5.52	0.06
-	88.18	5.14	0.06
-	72.94	75.54	1.04
-	59.10	62.35	1.05
-	36.40	1420.54	39.03
-	13.52	1256.90	92.97
-	2.49	0.16	0.07

Table A-8 Monthly GPS loss in 2011

Month	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
Jan	8	600.01	111.45	0.19
Feb	6	35.83	90.36	2.52
Mar	8	63.38	47.63	0.75
Apr	9	18.49	34.06	1.84
May	13	37.39	30.43	0.81
Jun	8	20.68	27.73	1.34
Jul	9	25.92	65.97	2.55
Aug	10	65.98	730.11	11.07
Sep	8	56.71	69.76	1.23
Oct	8	1240.58	35207.81	28.38
Nov	5	1844.38	16751.93	9.08
Dec	9	27.66	295.65	10.69

Table A-9 Hourly GPS loss in 2011

Hour (UTC)	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
1	6	210.42	90.57	0.43
2	7	171.93	4652.48	27.06
3	9	117.54	3283.50	27.93
4	7	167.32	3802.88	22.73
5	9	199.72	3028.00	15.16
6	7	207.70	3849.65	18.53
7	6	156.88	4579.76	29.19
8	8	128.94	3435.49	26.64
9	9	176.46	3441.33	19.50
10	7	419.40	3729.41	8.89
11	7	486.21	4010.31	8.25
12	9	391.53	3026.60	7.73
13	8	449.75	3382.12	7.52
14	9	380.80	3159.49	8.30
15	10	303.12	2784.58	9.19
16	10	154.70	3398.09	21.97
17	8	143.27	4092.45	28.56
18	15	203.96	2551.36	12.51
19	10	302.29	4087.52	13.52
20	8	240.26	4371.94	18.20
21	9	205.31	3854.53	18.77
22	9	127.34	3109.42	24.42
23	8	164.34	3455.42	21.03
24	9	104.99	2904.70	27.67

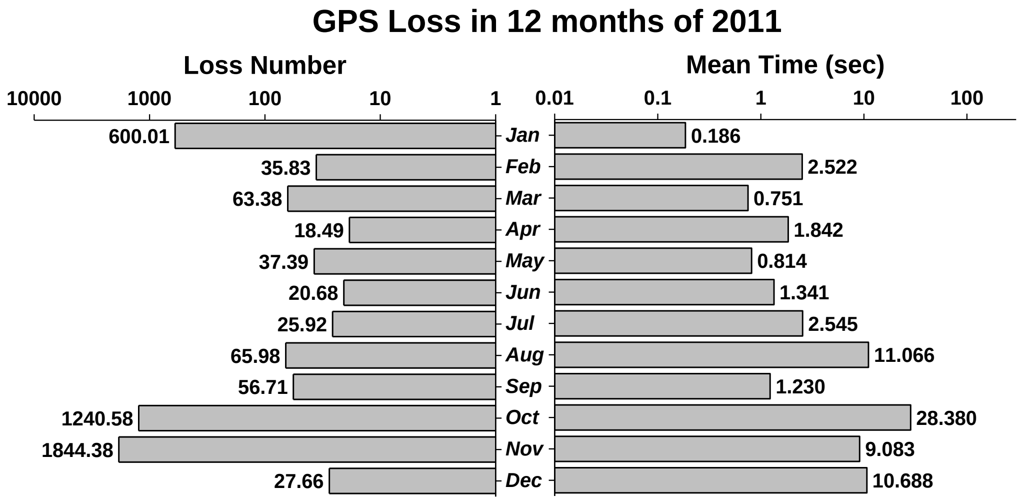


Figure A-5 Monthly GPS loss in 2011.

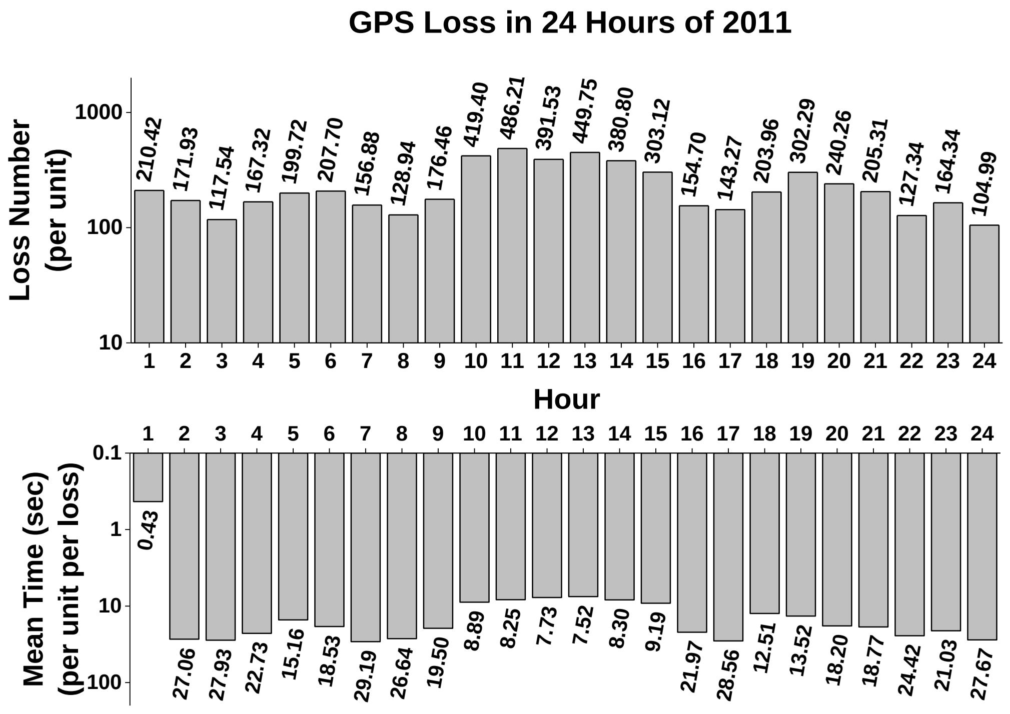


Figure A-6 Hourly GPS loss in 2011.

4. ***Results of 2012***

Table A-10 Units with GPS loss in 2012

Name	Number	Gross Duration	Mean Time (per unit per loss)
-	8125.84	1543.74	0.19
-	4311.01	4398.91	1.02
-	2620.99	92.74	0.04
-	1203.31	7699.59	6.40
-	872.14	1237.90	1.42
-	840.11	1204.42	1.43
-	639.03	138.72	0.22
-	405.43	5132.03	12.66
-	273.98	71.05	0.26
-	218.72	55.46	0.25
-	149.16	167.71	1.12
-	46.40	11595.60	249.92
-	26.42	14.68	0.56

Table A-11 Monthly GPS loss in 2012

Month	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
Jan	6	24.73	37.42	1.51
Feb	7	199.96	1157.39	5.79
Mar	6	24.43	26.94	1.10
Apr	6	87.60	126.03	1.44
May	6	27.52	40.06	1.46
Jun	7	21.05	26.26	1.25
Jul	7	1077.88	766.95	0.71
Aug	6	5.22	14.09	2.70
Sep	10	1073.85	85.78	0.08
Oct	4	105.52	141.99	1.35
Nov	3	9.26	1962.55	211.88
Dec	5	37.04	2290.14	61.83

Table A-12 Hourly GPS loss in 2012

Hour (UTC)	Unit Number	Loss Number (per unit)	Duration (sec)	Mean Time (sec, per unit per loss)
1	11	54.96	28.27	0.51
2	11	70.89	20.81	0.29
3	9	78.95	18.68	0.24
4	11	55.98	19.21	0.34
5	10	62.87	18.84	0.30
6	8	73.63	17.56	0.24
7	10	57.56	18.52	0.32
8	11	64.42	18.67	0.29
9	10	65.70	20.84	0.32
10	11	65.08	21.50	0.33
11	10	702.58	143.08	0.20
12	11	67.18	55.39	0.82
13	11	60.94	1663.84	27.30
14	12	46.64	18.13	0.39
15	11	60.24	607.58	10.09
16	12	56.39	17.83	0.32
17	11	69.87	246.30	3.52
18	12	61.92	124.64	2.01
19	12	575.54	215.36	0.37
20	13	86.73	645.16	7.44
21	12	163.77	493.48	3.01
22	12	75.07	94.19	1.25
23	12	51.55	16.59	0.32
24	11	61.33	19.60	0.32

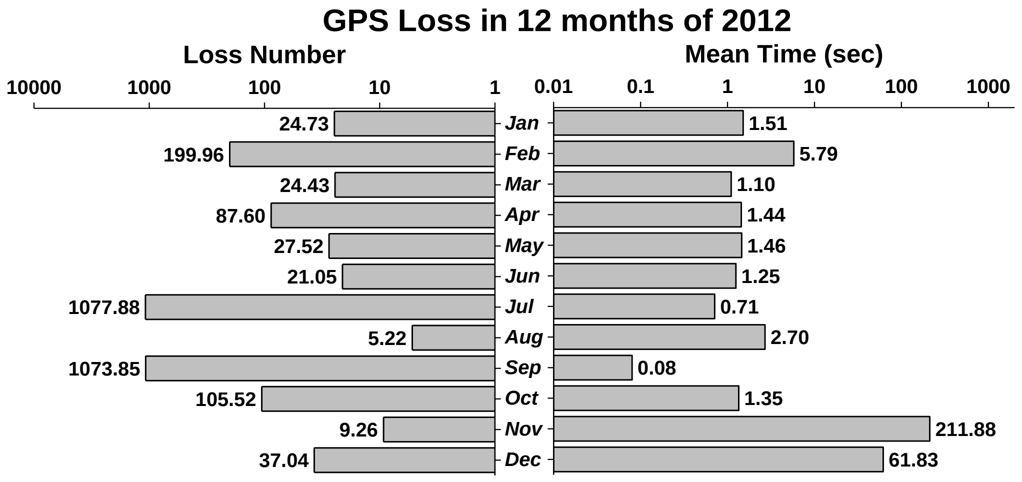


Figure A-7 Monthly GPS loss in 2012.

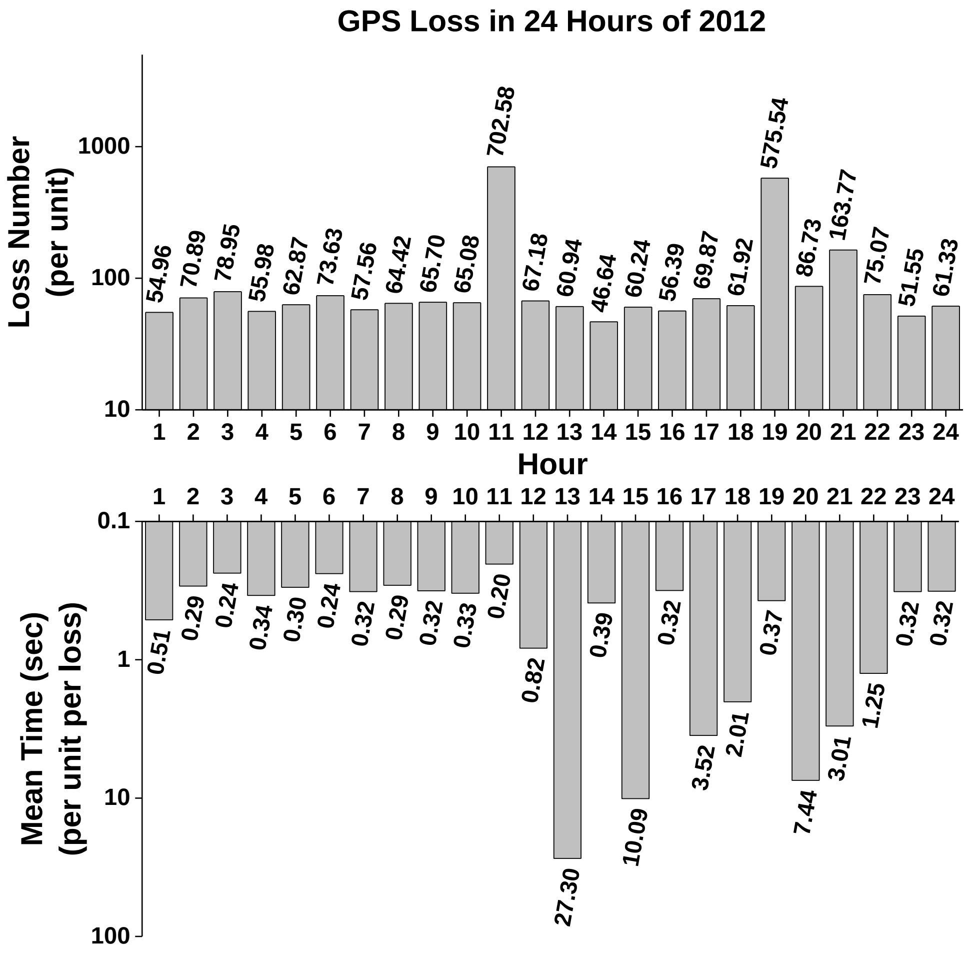


Figure A-8 Hourly GPS loss in 2012.

Vita

Dao Zhou was born in Baoding, China. He received the B.S. degree in electrical engineering from North China Electric Power University in 2007, and the M.S. degree in electrical engineering from Virginia Polytechnic Institute and State University in 2010. He is currently pursuing the Ph.D. degree in electrical engineering in University of Tennessee, Knoxville, with anticipated graduation in December, 2016. He is a recipient of Min Kao Fellowship in 2013.

His research interests include wide-area power system monitoring, situational awareness application development and big data analytics. He has published journal and conference papers on IEEE transaction on Smart Grid, IEEE Power and Energy Society General Meeting, IEEE Power and Energy Society Transmission and Distribution Conference and Exposition, Journal of Smart Structure and System, etc.