Development of a Synchrophasor Based Power Systems Monitoring Software with a Fault Locator Application for Multi-Terminal Transmission Lines

By

Yaojie Cai

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of
MASTER OF SCIENCE

Department of Electrical and Computer Engineering
Faculty of Engineering
University of Manitoba
Winnipeg, Manitoba

12 2016

Abstract

Synchrophasor technology is widely available embedded in modern power grid protection, metering, and recording devices. Utilizing synchrophasor measurements, a novel algorithm is proposed for fault location in multi-terminal transmission lines. In order to implement real-time synchrophasor applications, a software platform called "PhasorEye" was developed in this research. PhasorEye facilitates collecting synchrophasor data stream, visualization of decoded data, and implementation of synchrophasor applications as analysis tools. A laboratory setup involving a RTDS real-time digital simulator and a synchrophasor communication network was assembled to demonstrate and validate the use of the software and the proposed new fault location technique. Tests revealed several challenges in practical application of synchrophasor data for fault location and showed that the proposed fault location algorithm can accurately identify the faulted line segment and fault location. Additionally, several other synchrophasor applications developed by other University of Manitoba researchers were implemented and integrated into the software.

Acknowledgments

I would like to express my gratitude to Dr. Athula Rajapakse for his guidance and support throughout the years. It was an honour to receive advice from such a knowledgeable person.

I would like to thank the examining committee for the time spent to review my thesis.

I must also thank the academic, technical and administrative staff from the Department of Electrical and Computer Engineering. I am also grateful to graduate classmates for all their help during these years.

Thanks to my parents for the continuous support and encouragement to achieve goals in every step of my life.

My special gratitude is to God who never give me up and gave me the strength when I needed the most.

Yaojie Cai

Dedication

To my dear parents.

Contents

List of Tables

List of Figures

List of Symbols

L	Number of terminals tapping to the transmission line
K	Number of transmission line tapping nodes
i	Index of transmission line tapping node in forward direction
j	Index of transmission line tapping node in backward direction
FDT	Fault Direction Tag for bisection search
p	Index of terminals which connected to the ith tapping node
V_p	Positive-sequence voltage measurements from pth terminal
I_p	Positive-sequence current measurements from pth terminal
$V_{i,p}$	estimated positive-sequence voltage in ith tapping node using the measurements from pth terminal
$I_{i,p}$	estimated positive-sequence current in branch connecting ith tapping node and pth terminal using the measurements from pth terminal
VD_i	positive-sequence voltage deviation
MVD_i	maximum positive-sequence voltage deviation
$R_{p,x}$	Resistance from the transmission connected between ith tapping node and pth terminal x meters away from pth terminal
$r_{p,i}$	resistance per unit length from the transmission connected between ith tapping node and pth terminal
$L_{p,x}$	inductance from the transmission connected between ith tapping node and pth terminal and x meters away from pth terminal

$l_{p,i}$	inductance per unit length from the transmission connected between *i*th tapping node and *p*th terminal
$G_{p,x}$	conductance per phase from the transmission connected between *i*th tapping node and *p*th terminal *x* meters away from *p*th terminal
$g_{p,i}$	conductance per phase per unit length from the transmission connected between *i*th tapping node and *p*th terminal
$C_{p,x}$	capacitance per phase from the transmission connected between *i*th tapping node and *p*th terminal *x* meters away from *p*th terminal
$c_{p,i}$	capacitance per phase per unit length from the transmission connected between *i*th tapping node and *p*th terminal
γ	propagation constant for the transmission line
Z_c	characteristic impedance for the transmission line
l	length for the transmission line
x	fault distance from the sending of the transmission line
$\hat{x}$	normalized fault distance from the sending of the transmission line

List of Abbreviations

WAM	**A**lternating **C**urrent
GPS	**G**lobal **P**ositioning **S**ystem
UTC	**C**oordinated **U**niversal **T**ime
PMU	**P**hasor **M**easurement **U**nit
ROCOF	**R**ate **O**f **C**hange **O**f **F**requency
PT	**P**otential **T**ransformer
CT	**C**urrent **T**ransformer
ADC	**A**nalog to **D**igital **C**onverter
DSP	**D**igital **S**ignal **P**rocessor
DFT	**D**iscrete **F**ourier **T**ransform
PDC	**P**hasor **D**ata **C**oncentrator
TCP/ IP	**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol
API	**A**pplication **P**rogramming **I**nterface
PAR	**P**roject **A**uthorization **R**equests
IED	**I**ntelligent **E**lectronic **D**evices
CFG-1	**C**onfiguration **F**rame 1
CFG-2	**C**onfiguration **F**rame 2
CFG-3	**C**onfiguration **F**rame 3
LAN	**L**ocal Area **N**etwork
MU	**M**erging **U**nit

CID	**C**onfigured **I**ED **D**escription file
PSP	**P**hysical **S**ecurity **P**erimeter
WAN	**W**ide **A**rea **N**etwork
KDC	**K**ey **D**istribution **C**enter
SV	**S**ample **V**alue
OSI	**O**pen **S**ystems **I**nterconnection model
GSF	**G**rid **S**olution **F**ramework
WPF	**W**indows **P**resentation **F**oundation
MSDN	**M**icro**S**oft **D**eveloper **N**etwork
GUI	**G**raphical **U**ser **I**nterface
XML	E**X**tensible **M**arkup **L**anguage
CRL	**C**ommon **L**anguage **R**untime
COM	**C**omponent **O**bject **M**odel

Chapter 1

Introduction

In this chapter, importance of the wide area monitoring of power systems and its monitoring applications are presented. The concept of synchrophasor and the devices used for synchrophasor measurements are introduced. After providing this background, the research motivation, objectives, and the organization of this thesis are presented.

1.1 Background

The complexity of modern power systems is continuously increasing with passage of time, especially with wide employment of long, high capacity transmission lines and interconnection of massive amounts of renewable energy generation. In addition, the modern power systems are operated close to their critical limits as power utilities strive to optimize utilization of transmission and distribution assets to improve the economic profiles. With variable generation from wind and solar resources, a power system requires to dynamically adjust to achieve generation and load balance without violating various network constraints. Also, a power system should appropriately respond to various disturbances such

as faults or malfunction of equipment to prevent cascaded events which may lead to wide-spread blackouts. A power system is spread in a large geographical area, but a system wide view of the status of power system is very important for dealing with major events. The lack of wide area situation awareness has been identified as one of the main reasons behind some large scale blackouts [1] in the last decade. Some examples include the North American blackout on August 14th, 2003, which caused some locations to be without power for nearly two weeks; the European power outage November 4th, 2006, which affected 15 million people; and the loss of 50 gigawatts of generation in the north part of India in 2012, July, causing 670 million people to live in the darkness.

The renewed attention for Wide Area Monitoring (WAM) has attracted many researchers around world to search for WAM solutions. In order to develop an effective WAM system, time synchronized measurements from different locations of power network is essential. A device for synchronized phasor measurement was developed by G. A. Phadke, and J. S. Thorp, at Virginia Tech [2] in 1980s. However due to the limitations of the technology existing at that time, the delay error in the phase angle measurements was about 40μs [2], which is too high for many of the intended applications. At the beginning of the 21st century, enhanced accuracy civilian signals from the Global Positioning System (GPS) became available from the United States Government. As a result, the accuracy of time synchronization was improved to 0.2μs [2]. High quality time source permitted the researchers to develop wide area monitoring and analysis tools which further enhance the stability and flexibility of modern complex power systems.

1.1.1 Synchrophasor

A phasor represents a periodic waveform as a vector in the complex plane [3]. If the periodic waveform is given as

$$x(t) = X_m \cos(2\pi f_0 t + \phi) \quad (1.1)$$

where X_m is the amplitude of the waveform, f_0 is the nominal angular frequency, and ϕ is the angular starting point of the waveform, its phasor representation is

$$X = (X_m/\sqrt{2})e^{j\phi} \quad (1.2)$$

According to IEEE Standard C37.118.2-2011 [3], X becomes the synchrophasor representation of the signal (t), if the value of instantaneous phase angle ϕ is expressed relative to a reference cosine function of the nominal system frequency (60/50 Hz), synchronized with the start of the current second of the Coordinated Universal Time (UTC) [3]. Unlike the standard phasors which represent steady state waveforms at a constant frequency, a synchrophasors can represents a time varying sinusoidal waveform. When the amplitude of the sinusoidal is time varying, magnitude of the synchrophasor vary with time. When the frequency of waveform deviates from its nominal value or time varying, the phase angle of the synchrophasor vary with time. Such a time varying synchrophasor can be expressed as

$$X(t) = (X_m/\sqrt{2})e^{j(\phi + 2\pi \int_0^t \Delta f(t)dt)} \quad (1.3)$$

where $\Delta f(t)$ is instantaneous deviation of frequency from its nominal value.

1.1.2 Phasor Measurement Unit (PMU)

A phasor measurement unit is a functional unit in a physical device that provides synchrophasor measurements from the input sinusoidal voltage and/or current signals. A PMU requires a standard time signal to generate the common reference cosine signal. In addition to synchrophasors, a PMU also provide time tagged measurements of the frequency, and the Rate of Change of Frequency (ROCOF). The voltage and current analog signals from the Potential Transformer (PT) and Current Transformer (CT) are filtered using an antialiasing filter and converted to digital form at the Analog to Digital Converter (ADC) (Figure 1.1). The sampling process of the ADC is usually synchronized with the GPS clock signal and a phase-locked oscillator to provide a time accuracy of ±1μs. A Digital Signal Processor (DSP) receives the data from the ADC and computes the components of each phasor typically using the Discrete Fourier Transform (DFT). However, other algorithms such as Least Square Estimation can be used to compute the phasors. The result from the phasor calculation is further filtered and attached with a time stamp. The frequency deviation is computed as the rate of change of phase angle, and ROCOF is the rate of change of frequency (or of frequency deviation). All measured data are encoded into a synchrophasor data frame according to IEEE Standard C37.118.2-2011 in order to allow the communication interface to stream out as shown in Figure 1.1 [3]. Synchrophasor data are reported at one of the standard rates that depend on the nominal frequency as specified in IEEE Standard C37.118.2-2011. The fastest reporting rate according to the current standards is one data frame per cycle at the nominal frequency (50/60 frames per second) [3].

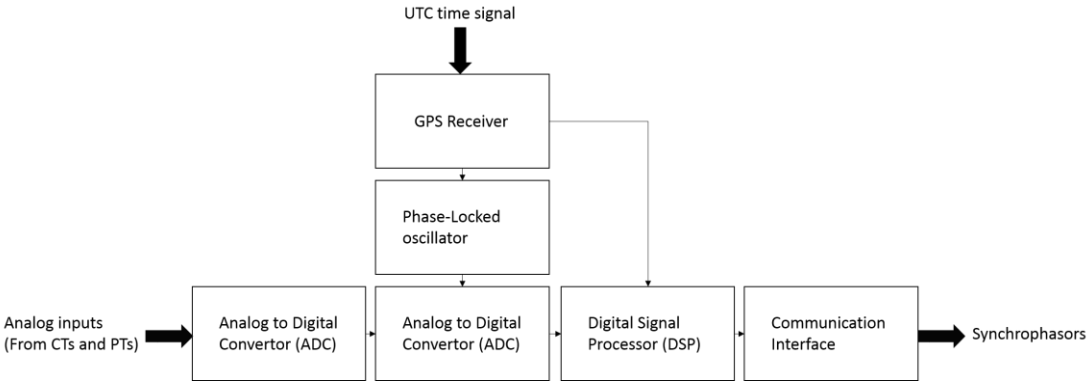


Figure 1.1 Phasor Measurement Unit (PMU) structure

1.1.3 Phasor Data Concentrator (PDC)

A Phasor Data Concentrator is a hardware device or a software application which allows the user to gather data from different PMU output streams. Its functions also includes verification of the integrity of the data, time alignment of measurements according to the time stamps, and creating coherent output streams for other application to use [3]. The structure of a PDC is not fixed: from the platform perspective, it can be a set of cloud servers, or a simple digital data storage with the network connection and a microprocessor; from the software perspective it can be multithread desktop application or a browser application which interface to the hardware server.

1.1.4 Synchrophasor applications

The ability of collecting synchrophasor data from a large number of PMUs (which are located at different geographical locations) to a central location at a fast rate and the availability of powerful modern multiprocessor computers capable of analysing massive

amounts of data are inspiring the researchers to develop advanced synchrophasor applications that can support more reliable, economical and secure operation of electricity grids. The synchrophasor applications include various monitoring, control and protection functions. Applications such as phase angle difference and frequency monitoring, oscillation and damping monitoring, line parameter monitoring, voltage stability monitoring, etc [6], improve the situational awareness of the system operators and aid decision making. Closed loop control applications such as synchrophasor based damping control are emerging, while research on wide area protection and emergency control are promising [6]. One of the useful but simple application of synchrophasors is fault location in complex networks. Although most modern protection relays have the ability of providing fault locations of two or three terminals lines, they are unable to locate faults in transmission lines with multiple terminals. Time synchronized phasor measurements at multiple terminals may allow location of faults in such complex transmission lines.

1.1.5 Fault location in multi-terminal transmission lines

Data obtained at the time of fault occurrence is essential for accurate fault location. In this respect, synchrophasor measurements, which are time-synchronized measurements of voltages and currents with time stamps, can provide high-resolution snapshots of a power system. PMUs and associated communication networks allow collection of data and transmit them to the control room at a high rate (some PMUs are capable of streaming up to 240 frames per second [2]) through Ethernet. There is a potential to develop a reliable fault

detector and locator using synchrophasor measurements to give highly accurate fault location estimates. Accurate fault location helps minimizing the down time and saving millions of dollars to power utilities.

Multi-terminal transmission line are the transmission lines with three or more terminals. It is used mainly in distribution level. However, some transmission level networks, especially where renewable energy generation is interconnected, also have multi-terminal structure. Multi-terminal transmission line includes one main transmission line and a number of branches tapped out to supply remote loads or bring in remote generation. Often, there are no measuring devices or protection equipment located at such tapping nodes. Although most distance relays are equipped with fault location functions, they are not directly applicable to multi-terminal transmission lines. Identification of the faulted line segment is a major challenge when measurements are not available at tapping points. The existing techniques for fault location in multi-terminal situations are extensions of two-terminal fault locations algorithms. Considerable effort and offline analysis is required for setting various thresholds required by these algorithms [7]-[11]. Some of those setting are heavily dependent on the engineer's knowledge and experience in studying the power system for which the fault location method is applied.

1.2 Motivation

IEEE standards C37.118.1 [4] &2 [3] -2011 establish the synchrophasor measurement requirements, including the measurement accuracy requirements under dynamic conditions, and the methods for synchrophasor information exchange. Scientists and engineers around

world are working together to develop applications that use synchronized phasor measurements. However, use of synchrophasors for fault location in complex multi-terminal transmission lines has not been well investigated. Furthermore, most of the research on synchrophasor applications does not go beyond the point of offline analysis, or a command line based console applications with proposed algorithms. Thus, there is still a gap in the research process between the abstract theoretical algorithms or the command line programs and the off the shelf applications which can be used in the real world. The specific problem that can arise in real-time implementation of various algorithms are not properly investigated. Nonetheless, with the help of the developer community such as Microsoft Developer Network, Java community, or some new self-learning and online streaming websites such as Coursera, Livecoding.tv, the learning curve for software engineering is shifting from a positively skewed distribution which has a high requirement for the beginner, to a negatively skewed distribution which makes the researchers in different areas to build a medium size software without being through too much troubles. In addition, more and more developing environment tools for building software applications is becoming free. Example are Unity [12] and Unreal [13] 3D engines in the computer game developing area or Visual Studio [14] in the business application building area for Windows operating systems. Thus, the main motivation of this research is to develop a software application which is using the free resources, to fulfil the gap between the abstract algorithms and the real-time software applications. This software should facilitate synchrophasor communication as per IEEE C37.118.2-2011 and provide a sample application to illustrate the software structure. The ability of easily adding new application modules to the software based on new algorithms would be a highly beneficial feature for future synchrophasor application related research.

1.3 Objectives

The object of this research is to develop a synchrophasor based algorithm for fault location in a transmission line with multiple taps, and to build a software with real time monitoring and offline analysis capabilities. The software application should be able to gather data from a synchrophasor data stream communicated according to IEEE standard C37.118.2-2011 [3]. In addition, the research also aims to develop a laboratory setup to demonstrate proposed fault location algorithm and synchrophasor software. The specific objectives that are essential to achieve these goals are to:

- Review of relevant standards and some of the existing open source or commercial synchrophasor software.
- Create a reliable program structure and easy developing environment for real-time implementation of synchrophasor applications.
- Build a synchrophasor based power system monitoring program with good user interface and easy configuration steps based on the proposed program structure.
- Develop a laboratory setup centred on RTDS real-time digital simulator and existing laboratory equipment for testing the developed software application.
- Develop, implement and validate an algorithm for location of faults on a transmission or distribution line with multiple taps
- Implement several other power system analysis algorithms on the developed software platform.

1.4 Scope and limitations

The software and experimental setup will be developed at the Intelligent Power Grid Laboratory, of University of Manitoba, Canada. The research will exploit the available real-time simulation equipment, software within the laboratory, and the open source or free software online. The laboratory currently possesses one RTDS real time simulator with two GTNET cards [40], an internal communication network, and one computer with SEL-5073 Phasor Data Concentrator software [24].

1.5 Organization of the thesis

After this chapter's introduction of the research, Chapter 2 of this thesis presents a literature survey with outlines of different synchrophasor standards, review of some of the present date synchrophasor software, and assessment of different synchrophasor software layouts and the software requirements for the research. In addition, some of the current fault location algorithms for multi-terminal transmission lines is also discussed. In Chapter 3, development of the software structure is presented with the detail of software layers and supporting classes. In Chapter 4, the proposed algorithm for the searching of fault location in a multi-terminal transmission is explained. The results of testing the fault location algorithm are presented and discussed. In Chapter 5, other applications which are implemented inside the synchrophasor software are presented. Finally, the conclusions and recommendations are presented in Chapter 6.

Chapter 2

Literature Survey

In this chapter, an introduction to synchrophasor technology and standards is presented. Some existing synchrophasor application programs are reviewed. In addition, a brief review of synchrophasor based fault location algorithms is given.

2.1 Synchrophasor technology and standards

The concepts of the synchrophasor measurement and Phasor Measurement Unit (PMU) were proposed in the early 1980s [2]. The technology was gradually advanced and commercial PMUs were demonstrated in the early 1990s. In 1995, the first synchrophasor standard IEEE standard 1344-1995 [5] was introduced. For the first time, a systematic introduction of synchrophasor technology including the definition of synchrophasor was provided in this standard. In addition, synchronized sampling, source of time synchronization, and three types of the message frames, the header frame, the configuration frame and the data frame were introduced.

Like all other IEEE standards, after ten years since the first synchrophasor standard was published, a revision of synchrophasor standard was introduced in 2005 with the new standard number, IEEE C37.118-2005 [16], to reflect the state of the art. One of the major changes in the standard is counting second changes from the midnight of January first 1970

to UTC [16]. In addition, message format was specified as a packet structure, and numerous message formats examples in hexadecimal representation were given. These examples helped the developers to test their synchrophasor decoding functions [16].

In the late 2008 and early 2009, due to a request from the Institute of Electrical and Electronics Engineers (IEEE) to the International Electrotechnical Commission (IEC) for a joint logo regarding IEEE C37.118 synchrophasor standard, two of the IEC Technical Committees (TC95 - standardization of measuring and protection equipment, TC57 - standardization of exchange for real-time and non-real-time information) worked together to explore and implement a standard to allow the synchrophasor measurement to be exchanged via communication services provided in IEC 61850 substation automation standard [17]. To facilitate this, IEEE issued a Project Authorization Request (PAR) in 2010 to separate the IEEE C37.118 2005 standard into two parts: IEEE C37.118.1 standard for the synchrophasor measurement and time synchronization, and IEEE C37.118.2 standard for the synchrophasor data transfer [17]. The purpose of the two standards is to provide the backward compatibility with the old IEEE synchrophasor standards and the forward compatibility for coordinating with other standards such as IEC 61850 substation automation standard [17].

2.1.1 IEEE Synchrophasor Standard C37.118.2 2011

Communication network for the synchrophasor communication is defined in IEEE C37.118.2 [3] as shown in Figure 2.1. Different Intelligent Electronic Devices (IEDs) with the PMU function which are located in substations carry out synchrophasor measurements according to IEEE C37.118.1-2011 [4] synchrophasor measurement standard. Synchrophasor measurements streamed by different PMUs are collected by the local PDC in the

substation [3]. The collected synchrophasor data are collated and send to the corporate PDC of the utility, usually located in the utility's system control center [3]. A common central regional PDC can gather the synchrophasor data from the corporate PDCs from different utilities in order to provide a wide area snapshot of the power grid to various operating, planning or analyzing functions which include situational awareness, islanding strategy, protection strategy and others [3].

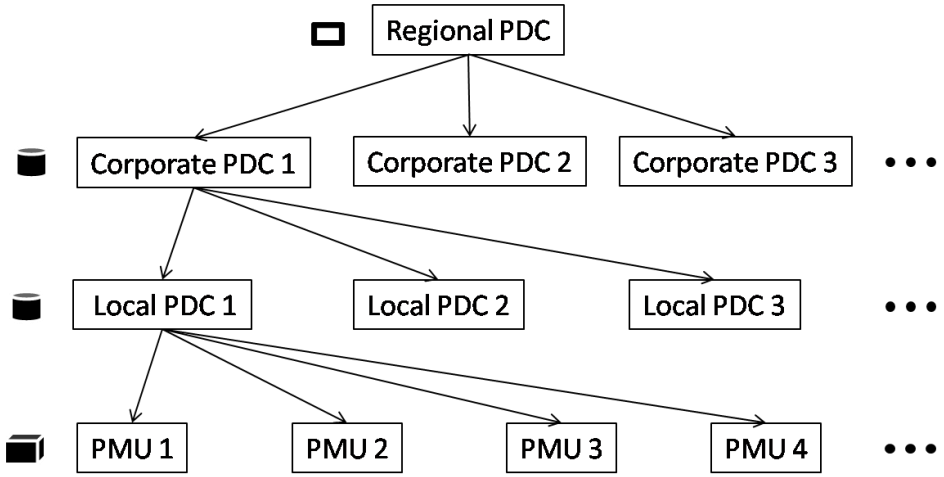


Figure 2.1 Synchrophasor network structure

IEEE C37.118.2 standard for synchrophasor data transfer defines four types of message frames which are used in the synchrophasor communication [3]. The command frame is used as starting point for hand shaking process and message control for any synchrophasor communication as illustrated in Figure 2.2. At the beginning of the communication, the first command frame is proposed by destination PDC or other synchrophasor applications to request the configuration frame from the source PDC or PMUs [3].

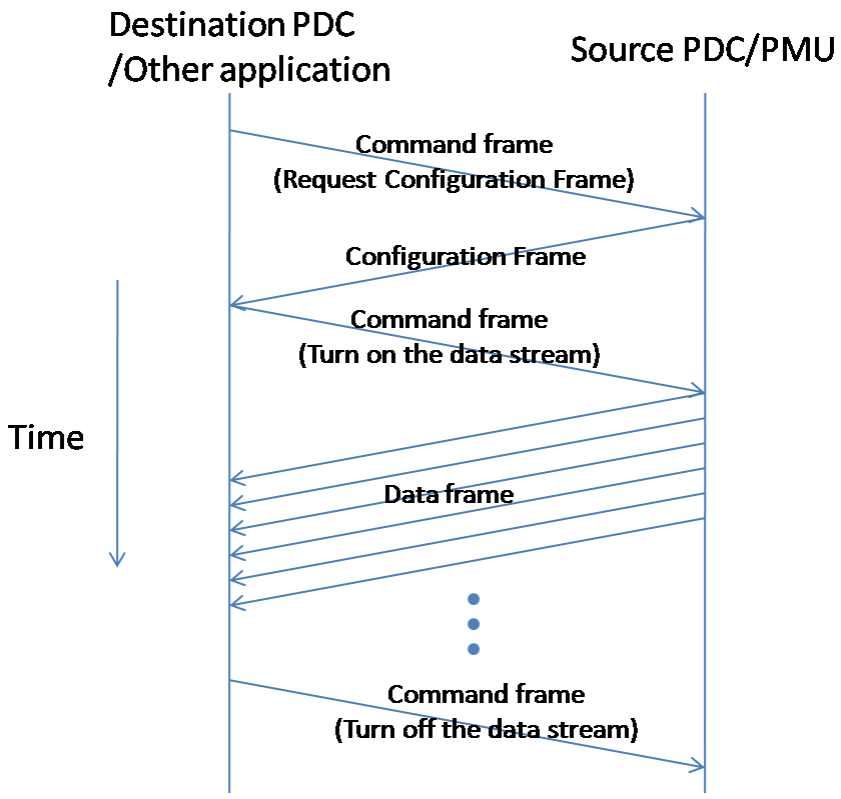


Figure 2.2 Synchrophasor stream

Configuration frame contains the information about data frame provided by source PDC or PMUs which includes station name of the source, number of PMUs which are hosted by the source, phasor name for each phasor from each PMU and its conversion factor, name for each analog signal from each PMU and its conversion factor, name for each digital signal from each PMUs and its mask words. The three types of configuration frames are defined in IEEE C37.118.2 standard [3]: (i) Configuration Frame 1 (CFG-1), (ii) Configuration Frame 2 (CFG-2) and (iii) Configuration Frame 3 (CFG-3). Both CFG-1 and CFG-2 describe the source PDC or PMU capability. In addition, CFG-2 is used to indicate the information which is being reported currently, which may be a subset of the capabilities described in CFG-1. CFG-3 has the same information contained in other configuration

frames and additional information such as the geographic location of the source PDC or PMU [3].

After correct decoding of the configuration frame, a second command frame is proposed by destination PDC or other applications to request source PDC or PMUs to turn on data stream as shown in Figure 2.2. In synchrophasor data stream, synchrophasor data frames are reported periodically according to the reporting rate set in the source PDC or PMUs [3]. A data frame contains phasor information which may be a set of single phase or three phases positive, negative or zero sequence measurements, frequency data, and rate of change of frequency in fix 16 bits or floating-point format. In addition, other analog and digital signals can also be appended into data frame to allow the transportation of the custom measurements from the PMU [3].

When it is required to end the communication process, a third command frame is proposed by the destination PDC or other applications to request the source PDC or PMUs to turn off the data streams and dispose the communication channel [3].

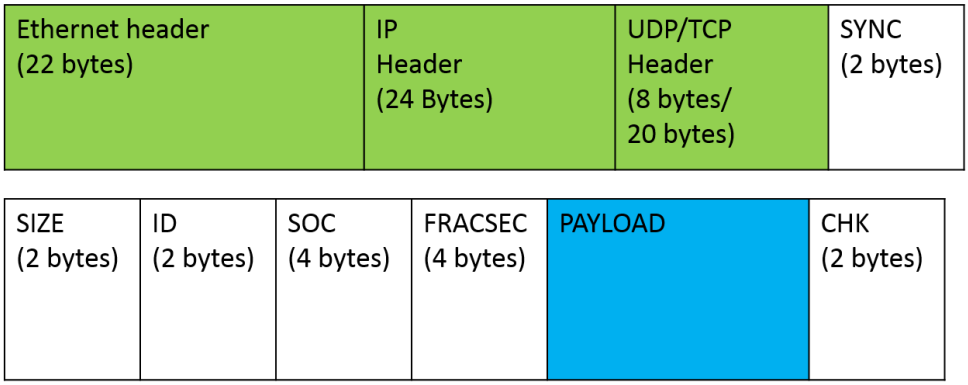


Figure 2.3 Synchrophasor data package

Frames in synchrophasor communication have a common packing layer as shown in Figure 2.3. The head of a frame is the communication layer which includes Ethernet header, IP header and UDP or TCP header. “SYNC” stands for frame synchronization word [3]. Due to the traffic and other interference, frame may be damaged; checking of leading byte “AA” in frame synchronization word provides one of the ways to estimate the integrity of the frame. In addition, frame synchronization word contains information about frame, such as the type of the frame, and the standard used to encode the frame. “SIZE” field of a frame provides the total size of the frame [3]. “ID” field indicates ID code for the source (each device which is inside the synchrophasor network should obtain a unique ID code) [3]. “SOC” field provides time stamps for the measurement [3]. Time stamp is a 32 bits unsigned number, which is the number of seconds elapsed from the midnight of January first 1970 (starting point of UTC Time) [3]. “FRACSEC” field provides the fraction of the second and the quality of the time source [3]. At the end of the frame is “CHK” field, which is a cyclic redundancy code to verify that the set of the data has not been compromised [3].

2.1.2 Synchrophasor data transfer according to IEC 61850-90-5

One of the leading standards in power system automation communication is IEC 61850 standard [17]. IEC 61850 standard is a standard for the communication inside a substation [17]. One of the key ideas for IEC 61850 standard is to replace hard wired signal wiring with a fiber-optic cable based high-speed Ethernet Local Area Network (LAN) [17]. However, the good situational awareness that wide area monitoring provides urges the IEC 61850 standard to develop its own part for wide area measurement transmission. In 2009, a joint task force between IEEE and IEC was assembled to change IEEE C37.118 2005

standard into two parts as previously mentioned, and create a new part, namely IEC 61850-90-5, in the IEC 61850 standard [17]. The idea is to use the existing IEC 61850 standard based IEDs to transfer the synchrophasor measurements [17]. The idea split into two parts. First part is using the existing IED to publish the synchrophasor data. The second part is transferring synchrophasor or other data to locations outside of the substation with the security and other features.

According to IEC 61850-90-5 standard draft edition; "Due to the structure of the IEC 61850 standard it was not possible to create one part of IEC 61850 dealing with synchrophasor communication" [17]. Thus, the easiest way to accomplish the first part is mapping the synchrophasor frames into the existing IEC 61850 standard object frame. Unlike IEEE C37.118 synchrophasor standard, raw data in IEC 61850 is laid in a highly object-oriented archive [17]. Figure 2.4 shows a basic archive structure of the IEC 61850, on the top of the archive is physical device which represents an actual protection or measurement unit in substation [17]. Below IED are logical devices which represent function goals or features that physical device possesses [17]. Under logical device is logical nodes which represent the functions or classes to support the function goal that logic device proposed [17]. In the end is the data object which is used by each logical node to support the logic node [17].

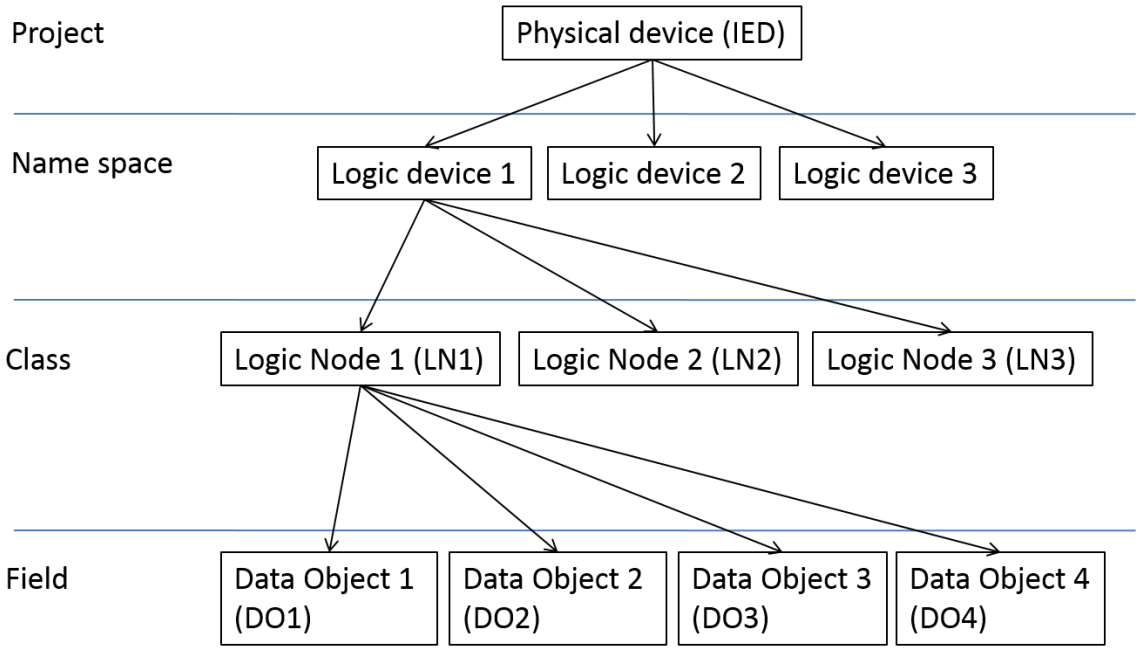


Figure 2.4 Archive for IEC 61850 standard data structure

In order to clarify the correct mapping method for publishing synchrophasor measurements by using IEC 61850 standard, a number of use cases are listed in the IEC 61850-90-5 [17]. These use cases include wide area synchro-check, adaptive relaying, out-of-step protection and other wide area control or protection schemes which are developed for the synchro-phasor technology. Based on the studies involving synchrophasor application use cases, Merging Unit (MU) is selected for publishing synchrophasor data [17].

MU is an IED which transmit current and voltage sampled values according to IEC 61850-90-5 from CT and PT to substation process bus [17]. MU has sampling, re-sampling and time measurement merging functions built inside its logic nodes that include Measurement information (MMXU), Sequence Imbalance (MSQI), Current Transformer (TCTR) and Potential Transformer (TVTR) logical nodes (refer to Figure 2.5) [19]. An MMXU logical node in merging unit can be made capable of functioning as a PMU as illustrated in Figure

2.5 by adding of new data objects HzRate (calculating the Rate Of Change Of Frequency, ROCOF) and ClcMth (the synchrophasor backend filtering class), and properly configuring the other data fields [20]. To be compatible with synchrophasor measurements, the data field ClcMode (calculating mode) should be set to periodical mode, ClcIntvTyp (MU calculating interval time unit) should be set to milliseconds, and ClcIntvPer (reporting period) should fit with the selected synchrophasor data reporting rate [20]. In addition, the configuration frame can be replaced by using the MU's Configured IED Description file (CID). CID file can be sent to the receiving PDC and synchrophasor application instead of configuration frame [20]. More information is given in Chapter 7 of reference [17].

In summary, a merging unit IED can be used as a synchrophasor publisher. In order to achieve this, MU's logical node is remapped to fit the synchrophasor data frame. In addition, CID file of MU is reconfigured to satisfy the synchrophasor configuration frame.

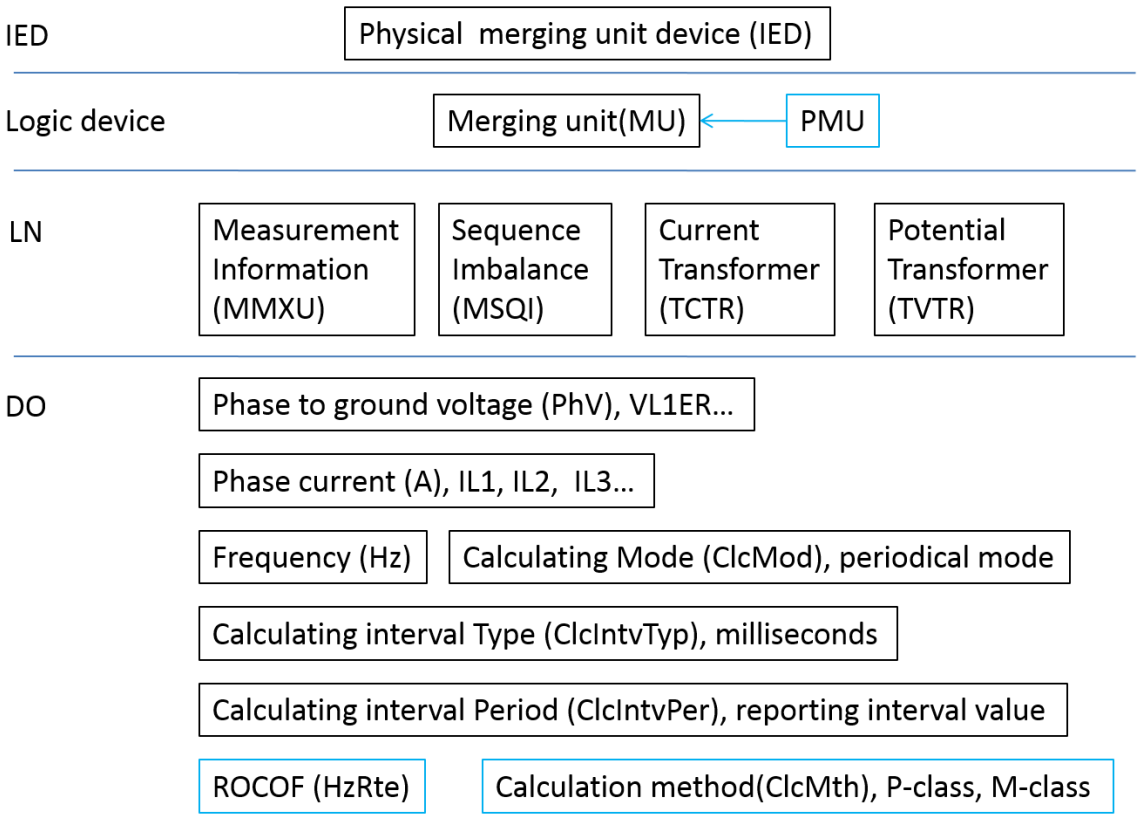


Figure 2.5 Archive for merging unit

The second part is focusing on mimicking data packing process and other features of synchrophasor data transfer through an IEC 61850 standard based network. In hardware level, in order to have a multicasting feature, speed up the data processing time and ease the process bus traffic, the UDP protocol is assigned as the only protocol which is used in IEC 61850 standard [17]. However, since IEEE C37.118.2 support both TCP/IP and UDP protocols [3], it is transformable in between standards (Figure 2.6, Figure 2.7 green section).

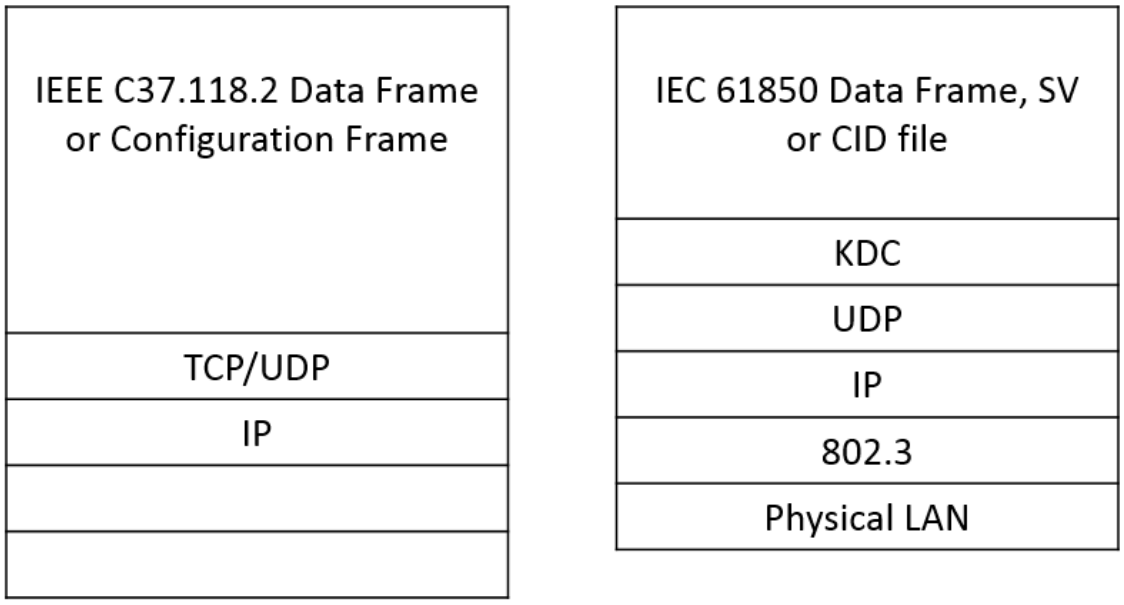


Figure 2.6 Communication layer for the synchrophasor data transfer, IEEE C37.118.2 (left), IEC 61850-90-5 (right)

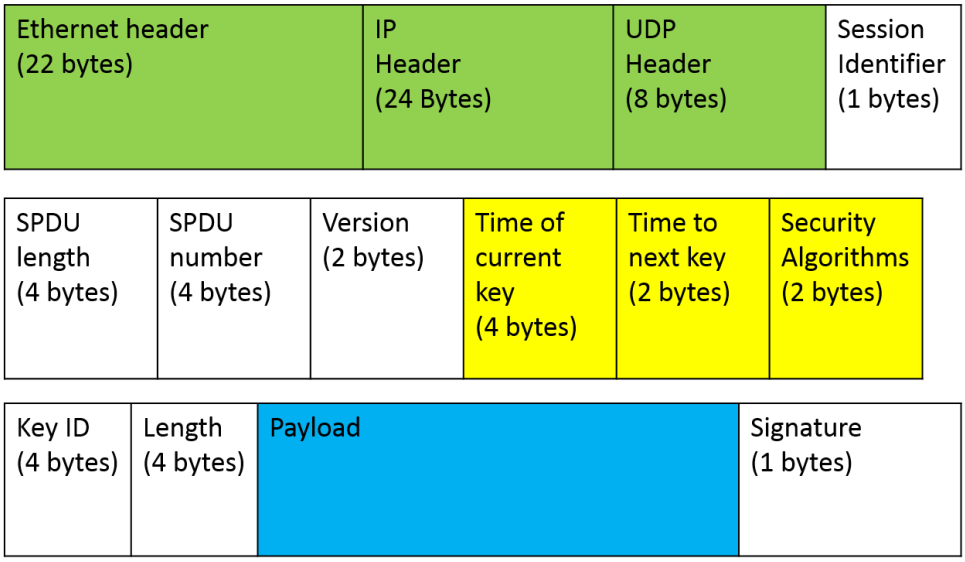


Figure 2.7 Data structure for IEC 61850-90-5

UDP protocol is a connectionless protocol, in other words, there may be corruption while transferring a package through UDP connection. However, since IEC 61850 standard data (GOOSE message or sample value) is used only inside substation, all data packages are

surrounding by the firewall or other protection method which is deployed inside the IEC 62351-6 Physical Security Perimeter (PSP) model for the cyber protection of the whole substation [17]. Data package is considered as secure package when it is inside substation's LAN. Nonetheless the integrity of the data package issues is amplified if data package is routing through Wide Area Network (WAN). In addition, even when data is transmitted using a non-connectionless protocol as TCP/IP, due to the confidentiality and availability of the transmission, connection is still asking for more security enforcement other than just using a non-connectionless protocol.

In addition, "The specification of asymmetric digital signature technology, within IEC 62351-6, has been found to be too CPU resource intensive for the current set of hard-ware/software that is being used for current protective relays, PMUs, etc." [17]. To solve above issue, reference [17] defines two traffic classes: Class A for the transmission speeds of 7.2-34.6 Mbits per second and Class B for 72-288 Kbits per second. Wide area synchro-phasor transmission between a substation and the control center is defined as class B traffic [17]. Gateways for the substation and control center are defined as the edge security nodes to provide the encryption for the class B traffic. The security method which is widely used in the industry for edge security nodes is "perfect-forward" security and encryption key rotation between each edge security nodes [17]. The Key Distribution Center (KDC) gives the encryption key for both connection ends adding a security layer into communication layers as shown in the yellow sections of Figure 2.6 and Figure 2.7.

In addition, as the payload of the package, a Sample Value (SV) from a MU can re-encoded into C37.118.2 synchrophasor standard data package and put into payload as traditional synchrophasor technology [17], or sent out as IEC 61850 SV measurement by MU along

with the high quality time stamp and other information such as the ROCOF to conduct a synchrophasor technology like environment. The second method is becoming more and more popular over time [20].

As shown in Figure 2.7, other than communication layers, security enforcement, and payload, the rest of the data package is similar to IEC 61850 protocol for sending GOOSE and SV, over Open Systems Interconnection model (OSI) which is layout in IEC 61850-90-1, IEC 61850-90-5 and IEC 61850-90-3 [17], specific data objects (such as the phase current from TCTR and phase angle from TANG) are selected to a dataset, then this dataset is assigned to a "synchrophasor" control block, user can configure transmitting configuration (such as reporting rate) by using "synchrophasor" control block [20]. By enable and disable the control block, a more secure IEC 61850-90-5 based synchrophasor stream transition is transmitted [17].

In summary, it is very important to recognize the differences in the standards. For a synchrophasor software, the overhead security enforcement or other features means it needs to include more encryption libraries or other supporting tools. This requires any synchrophasor application software to have a reliable structure for good scalability.

2.2 Analysis of Synchrophasor applications

In this section, a review of the existing applications is presented. Source code for the most of the existing software applications are not available, however, based on the analysis of some open source projects, and features of the non-open source software, some of the core design ideas for a synchrophasor software can be revealed.

2.2.1 PMU Connection Tester

"PMU connection tester" is a widely used open source tool in the field of synchrophasor technology. It is used to verify that a data stream from a synchrophasor measurement device is being successfully sent or received [23]. The version used in this thesis is 4.5.5 which was released in 2014 [23]. The analysis of "PMU connection tester" is an excellent starting point to develop a synchrophasor software.

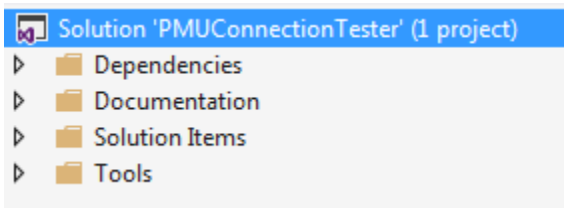


Figure 2.8 PMUConnectionTester project file

"PMUConnectionTester" is the project for "PMU connection tester", it is written in Visual Basic, with .Net framework. Project contains four folders as shown in Figure 2.8.

"Dependencies" folder contains all dynamic libraries to support the project which includes the core service library Grid Solution Framework (GSF) for supporting the decoding/encoding function for utility protocol, "Infragistics" commercial .NET user interface controls and components library, and "WIX" library for creating installation package. "Documentation" folder contains a help document for the software. "Solution Items" folder contains MIT license and other notice for the software. The core part of the software is located inside "Tools" folder. Main window and its associated class and functions are inside "PMUConnectionTester.vb" file which is under "Tools" folder.

The main window for "PMU connection tester" is spited into two parts, configuration area and display area as illustrated in Figure 2.9. The operation of "PMU connection tester" during runtime is very simple: User enters the setting (IP address and port number) for

synchrophasor device that the software wish to connect to in the configuration area, then clicks the "Connect/Disconnect" button. Synchrophasor stream from the synchrophasor device is displayed in the display area. The most important events in the operation are the event happened when user clicked the "Connect/Disconnect" button. Before the clicking event, the software is acting like a data sheet which allow user to write the setting in its data field. After the event, the software disables the data access from the user, and connect the data steam according to settings in data sheet. Analysis of the steps in "connect()" function in the main window background code grants a good insight for the program's main window.

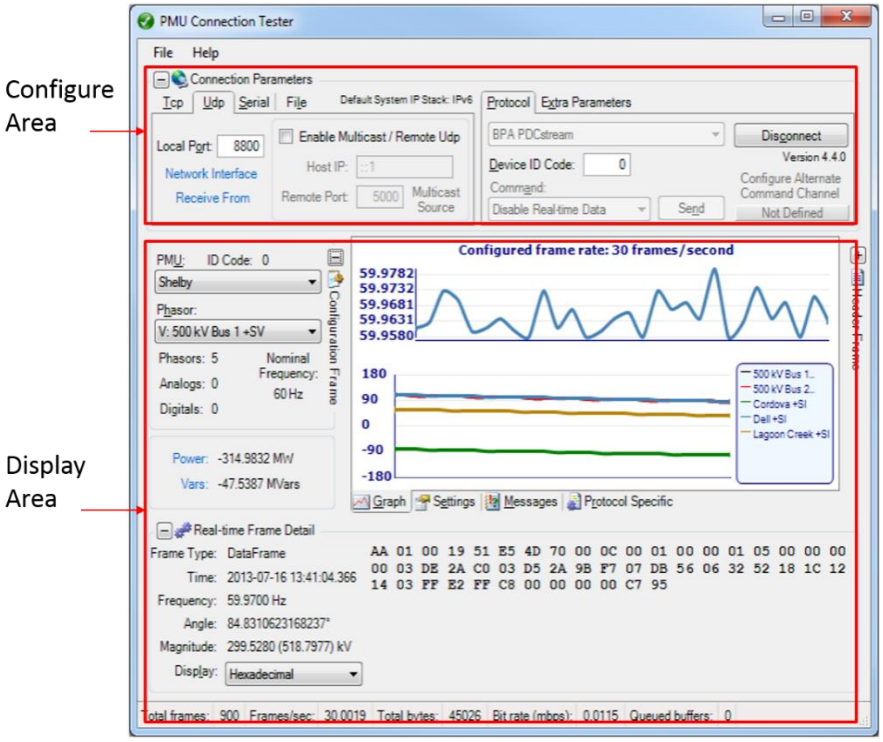


Figure 2.9 PMU Connection Tester window

Function “connect()” contains the following steps. Firstly, “connect()” validates and setup the connection parameters based on the settings the user enters. Secondly, it launches a communication socket and connect synchrophasor device via the protocol that user selected. Thirdly, the application updates the visual elements in the display area. In the end, function “connect()” setups a listener to monitor the “Connect /Disconnect” button click event again, in order to trigger the disconnect event function “Disconnect()”.

For each steps inside function “connect()”, numerous supporting functions are called: These include the user interface change, encoding or decoding function from GSF library. Thus, the runtime functions of the “PMU connection tester” can be represented by the flow chart in Figure 2.10. After the program starts, user can configure the program. Once the configuration completes, “PMU connection tester” program call the “connect()” function to construct the connection. After the connection is established, “PMU connection tester” program display and wait for the disconnect command from the user.

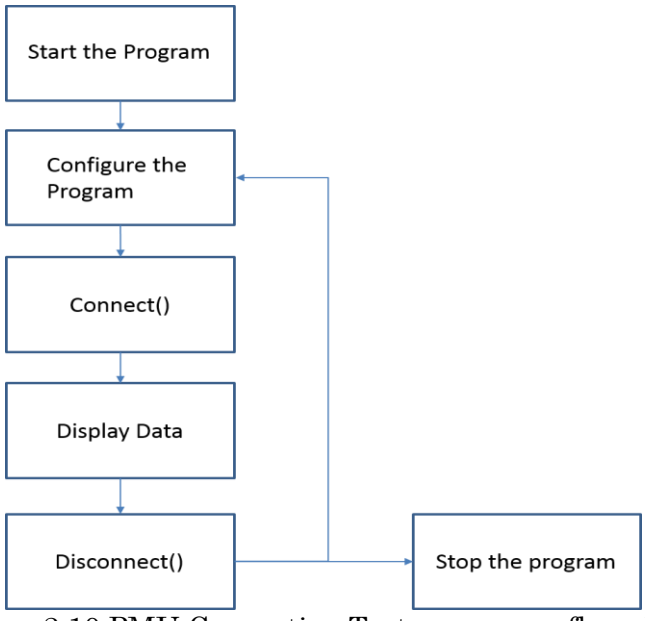


Figure 2.10 PMU Connection Tester program flow chart

The analysis of "connect()" function gives the basic structure for "PMU connection tester" software application, however "PMU connection tester" only has one single task which is receiving data in runtime. In addition, the structure of the program is complicated, the boundaries between program structure layers for the task are not very clear (Main window contains three thousands lines, and the author thinks it functions as both a business layer and a data layer). The use of GSF library is way too complicated, and no clear API or example case can be found. Those disadvantage makes it hard to learn, understand and make further development based on that. However, ignore the disadvantages, "PMU connection tester" is still one of the most valuable software for any synchrophasor software developer.

2.2.2 SEL 5037 PDC

SEL 5037 PDC software is one of the high performance, reliable commercial synchrophasor technology software in the market and its configuration window is shown in Figure 2.11 [24].

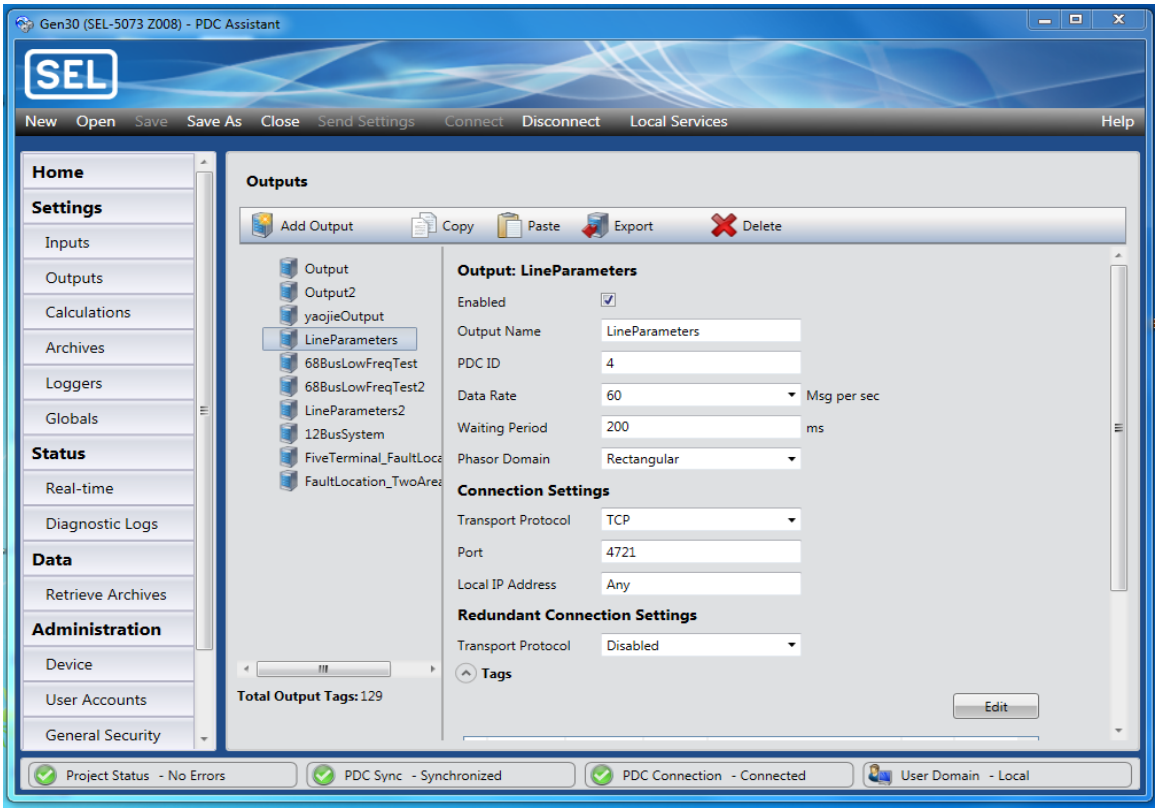


Figure 2.11 SEL-5073 PDC software configuration window

Some of the main features includes use of windows presentation foundation and page navigation in the user interface, the powerful database support and good threading operations. These positive features allow the software to connect with up to 500 phasor measurement unit inputs with up to 240 frames per second reporting rates [24]. The windows presentation foundation uses hardware acceleration for drawing the graphic user interface in order to provide better performance.

The using of individual databases improves the data sharing and security for SEL 5037 PDC software. However, the disadvantage is it requires the developer a good knowledge in database management system and its Query language.

When SEL 5073 PDC is operating and receiving data, user can reconfigure the PDC to connect with another device without stopping its connection. This feature shows that the multi-thread operation can significantly increase the performance of the software.

2.2.3 Other synchrophasor applications

Figure 2.12 shows a snap shot of "PhasorPoint" software from GE grid solutions. "PhasorPoint" software uses synchrophasor data stream to monitor the disturbances such as low frequency oscillations and to provide stability assessments [26].

Figure 2.13 shows a snap shot of a synchrophasor technology based frequency monitoring application in US Department of Energy website [27].

Figure 2.14 and Figure 2.15 show two browser-based visualization applications developed by Grid Protection Alliance. "OpenSEE" software is a browser-based waveform display tool, and it is based on "PQDashboard" software shown in Figure 2.16. This was released in 2015 March by Stephen C. Wills from Grid Protection Alliance [27].

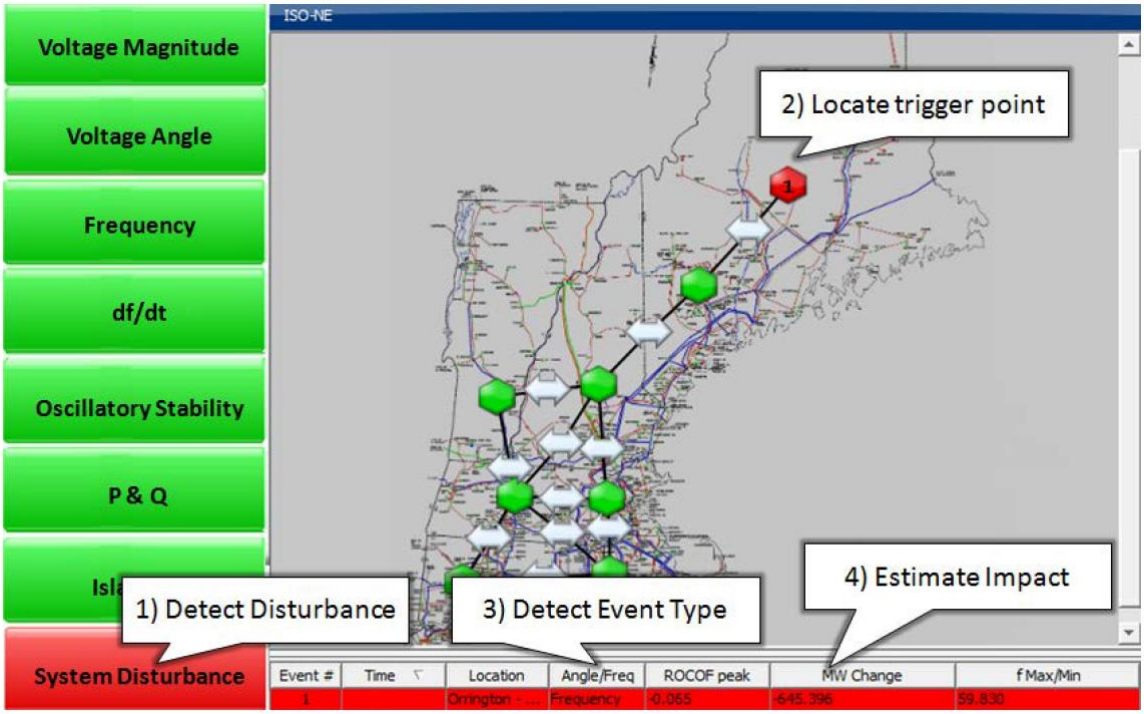


Figure 2.12 Snap shot for "PhasorPoint" software

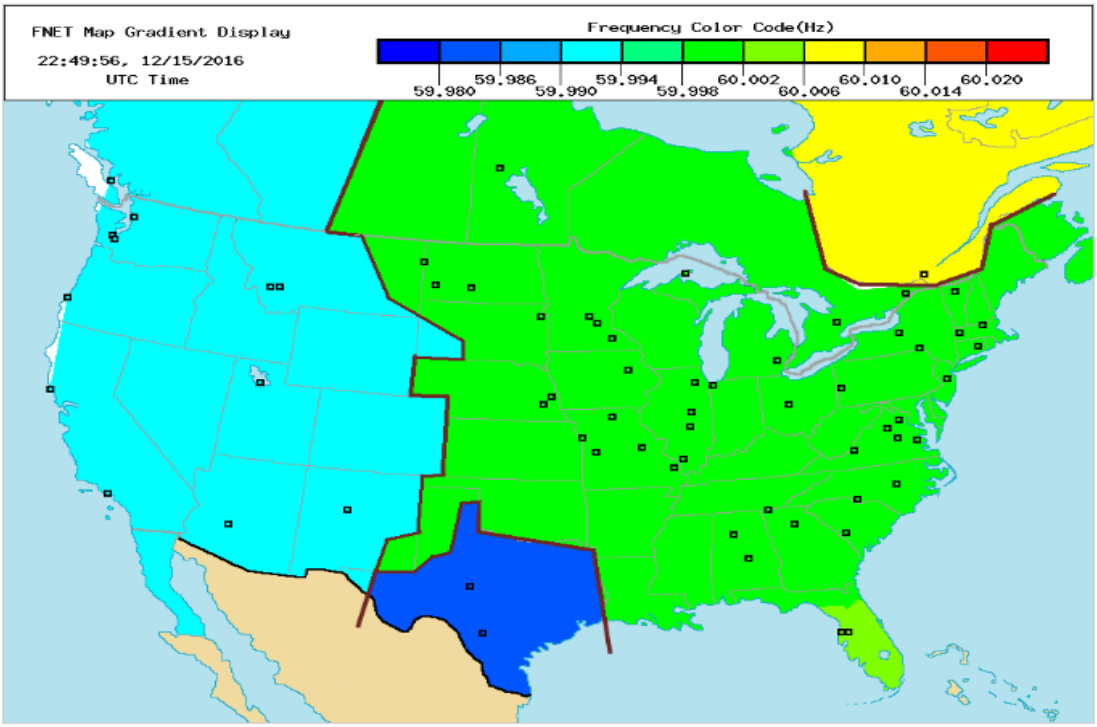


Figure 2.13 Frequency monitoring application in US Department of Energy website

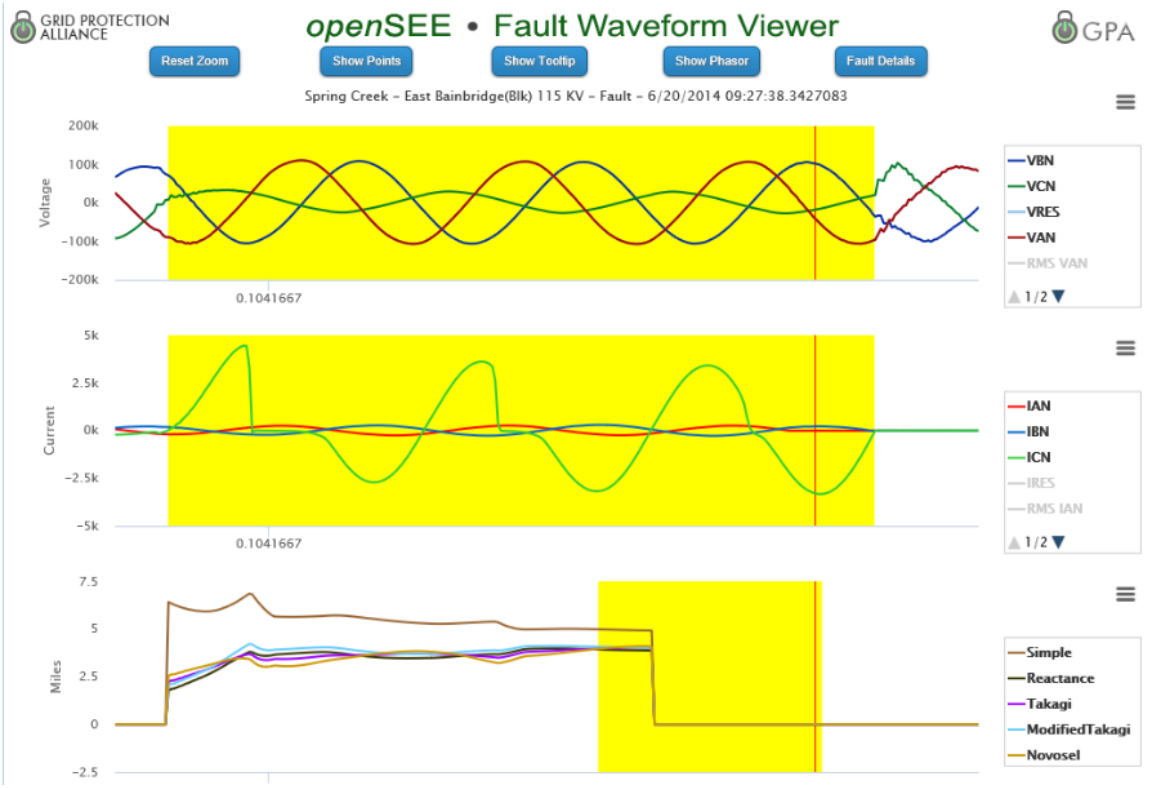


Figure 2.14 Snap shot for "OpenSEE" software

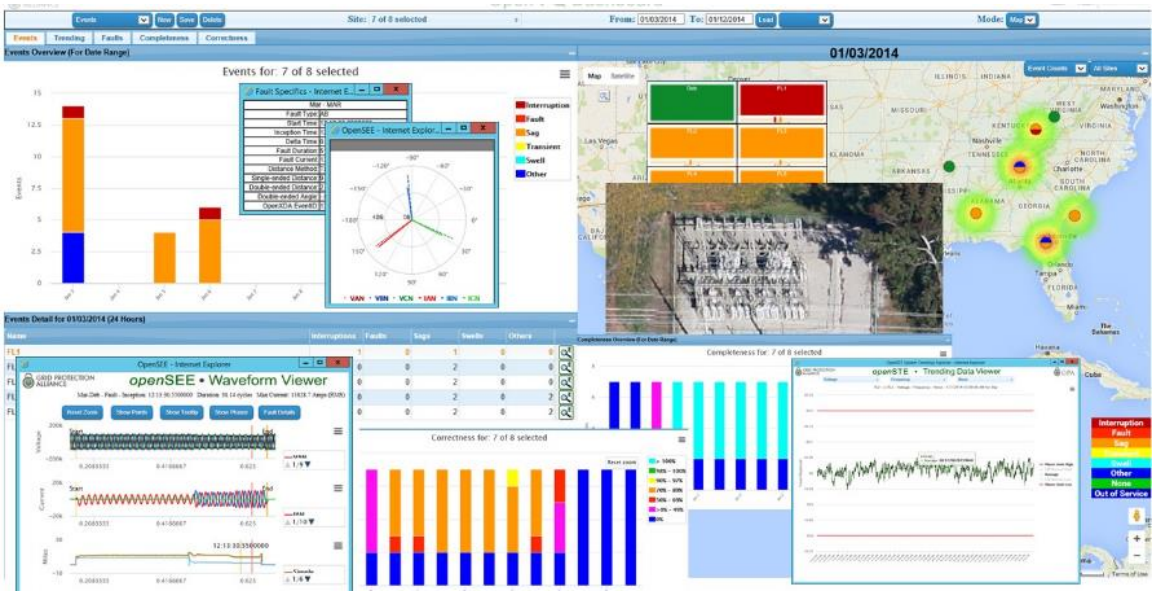


Figure 2.15 Snap shot for "PQDashboard" software

2.2.4 Summary

From the review of above applications, the key to design a high performance synchrophasor software application includes following three parts. First, windows presentation foundation can be used to develop the graphic user interface to increase the overall software performance. Second, a clear layered software structure can help the user and developer to understand the program and develop the program easily. In the end, an efficient thread pool is needed for multi-task software.

2.3 Fault location algorithms for multi-terminal transmission lines

Synchronized measurement-based fault location algorithms for multi-terminal transmission lines have been presented in the references [7]-[11]. A Thevenin equivalent impedance based scheme is presented in [7] by Brahma, only voltage measurement from all terminals are needed for this scheme. In addition, the effects of current transformer errors are avoided. Nonetheless, this scheme needs accurate source impedances to proceed.

A fault location method for a three terminal transmission line is proposed in [9]. Only the remote currents and local voltage is required for the algorithm. In addition, the shunt capacitance impact is avoided. Similar to the algorithm in [8], the faulted section is identified by using the knowledge of Thevenin impedance at each section. In addition, the three terminal transmission line algorithm in [9] is difficult to expand to a four or more terminal

system, because it becomes complex to determine the Thevenin equivalent for each section of the applied system.

The faulty branch selection algorithms for multi-terminal transmission lines presented in [10] and [11] use either some fault location index or unbalance in nodal currents. Nonetheless, the all of the above algorithms [7]-[11] are not designed to handle the fault location problem in non-homogeneous multi-terminal transmission lines.

A new scheme to locate a fault for power system with a multi-terminal, multi-section, non-homogeneous transmission line is presented in [11]. The algorithm estimates total vector errors in the tapping node voltage by using the tapping node or the terminal data next to it to select the faulty branch. In addition, the normalised fault distance is calculated by using the two terminal fault location algorithm. Yet, the difficulty in above multi-terminal line fault location algorithm is the selection of thresholds for the total vector errors comparison for the faulty branch selection [11]. Because the threshold is varying for different transmission line systems, use of Monte Carlo simulation is proposed for selecting the proper thresholds. But this procedure or use of other machine learning algorithms to identify the correct threshold needs a certain amount of data sets and time to proceed. The procedure has to be repeated when the algorithm is applied for a new system, or if the system is changed.

In this thesis, a simple bisection faulty branch searching algorithm is proposed to narrow down the fault into one of the section in a multi-terminal transmission line without using the threshold. Then, the two terminal fault location calculations are applied to the fault section for the normalised fault distance. Another aspect that has not been well investigated is the practicality of using synchrophasor measurements obtained immediately after the

fault for fault location. Synchrophasors are not well defined under transient conditions. Furthermore, clearing of faults by the action of protection systems within three or four cycles leaves only one or two valid data points (depending on PMU reporting rates) available for fault location calculations. Thus the accuracy of synchrophasor based fault location under practical conditions need to be properly investigated.

2.4 Chapter summary

In this chapter, standards for synchrophasor technology are reviewed. Several existing synchrophasor software applications are examined. The literature on the fault location in multi-terminal transmission lines is reviewed.

Chapter 3

Design of Synchrophasor software

In this chapter, details about the design and development of "PhasorEye" software are presented. Two aspects are covered; first part presents the software structure which includes the components inside "PhasorEye" software, their purpose and functionality inside "PhasorEye" software's presentation layer. Second part presents the details of "PhasorEye" software's internal interaction during online operation, which includes the network connection process, buffer reading/writing process, decoding and display process for IEEE C37.118.2 synchrophasor data stream, and the data storage process inside the data layer.

3.1 Program structure

Software architecture is a software solution which meets all of the technical and operational specifications. It can be categorized in many ways: from communication perspective, the structure can be arranged as service-oriented architecture or message bus based architecture. From the deployment perspective, the structure can be arranged as client/server, 2-tier, 3-tier, or N-tier architecture. From style of the programming, the structure can be arranged as component-based, object-oriented or layered architecture. Moreover, multiple perspectives can be selected to create hybrid architecture. To simplify the selection process,

a Venn diagram between user experience, business and system goal is used for "PhasorEye" development. Figure 3.1, shows the Venn diagram for the program goals.

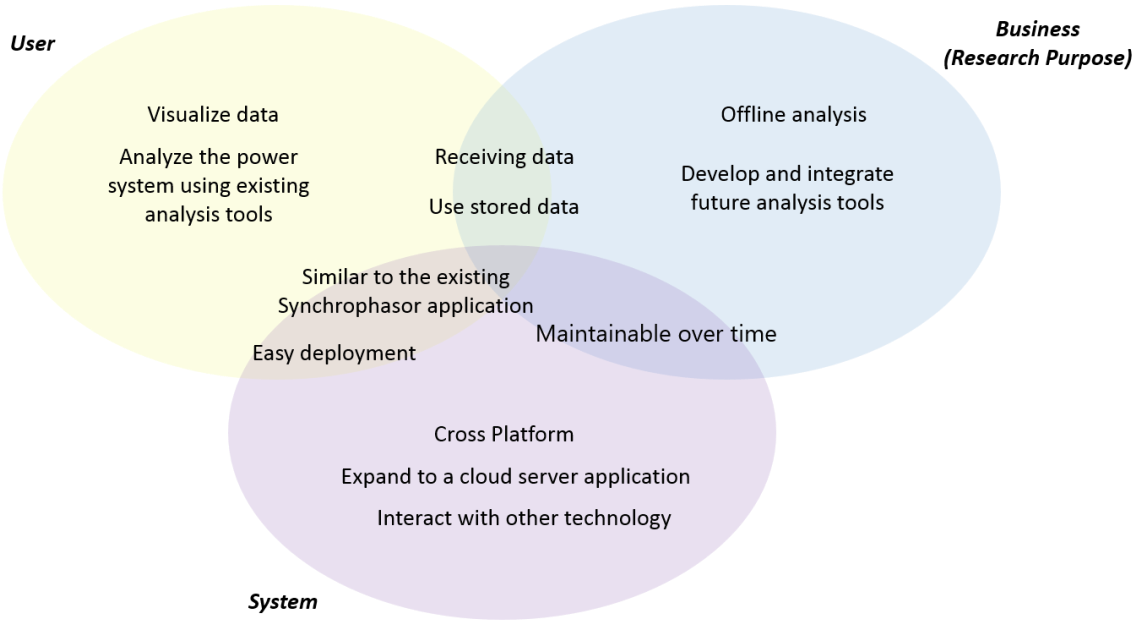


Figure 3.1 Venn diagram for the program goals

The main goal for "PhasorEye" software is user experience and research purpose. For the user experience, a simple, reliable and fast data visualization system for users to check and analysis real time data stream is one of the main features of the program. With the easy access to the real time data stream, other power system analysis tools can be added to the program to provide the user with enhanced insight to the status of the power system.

Function of a synchrophasor application begins with collecting a reliable data set. The "PhasorEye" software has to collect, decode and store the real-time synchrophasor data into a real-time database for which the user or researcher is granted access for implementing real-time algorithms or write into a file which will be accessed by offline analysis tools. This data collection, decoding and storing process has to be easy to proceed. Those goals are address in the area between the user experience and research purpose circle.

For the goal in research purpose circle, “PhasorEye” software has to deliver offline analysis tool set which allow the user to explorer and debug their analysis tool or solution in frame by frame and iteration by iteration manner. Also the architecture structure for the “PhasorEye” has to allow the user or researcher to integrate their analysis tool or solution into the main program, and start real-time power system analysis.

Between the user experience and system requirement circle, there are two important goals. “PhasorEye” software has to be easy to deploy and also its user interface needs to be similar to existing synchrophasor software. This feature can smooth the learning curve and allow users to master the program as soon as possible.

Between the research purpose and system requirement circle, easy maintainability over-time goal is proposed. Since the program is expected to grow with new power system analysis tools and other new features, software architecture needs a coherent strategy to cope with the growth and facilitate future researchers to develop new features and tools without interference from the existing features and tools.

In the system requirement circle, more goals are proposed, which includes “PhasorEye” software’s architecture structure to have a cross platform ability which it can be deployed from a desktop application to a mobile application or web application. Moreover, to handle a power system with a large amount of PMUs, a cloud server support is also allowed for “PhasorEye” software. In addition, with the new technology developed around world, continuous evolution is also a good goal for “PhasorEye” software.

However, due to time and resource limitations, the user experience and research purpose goals are more focused in the “PhasorEye” design and developing research. Figure 3.2, shows the reduced Venn diagram for the program goals.

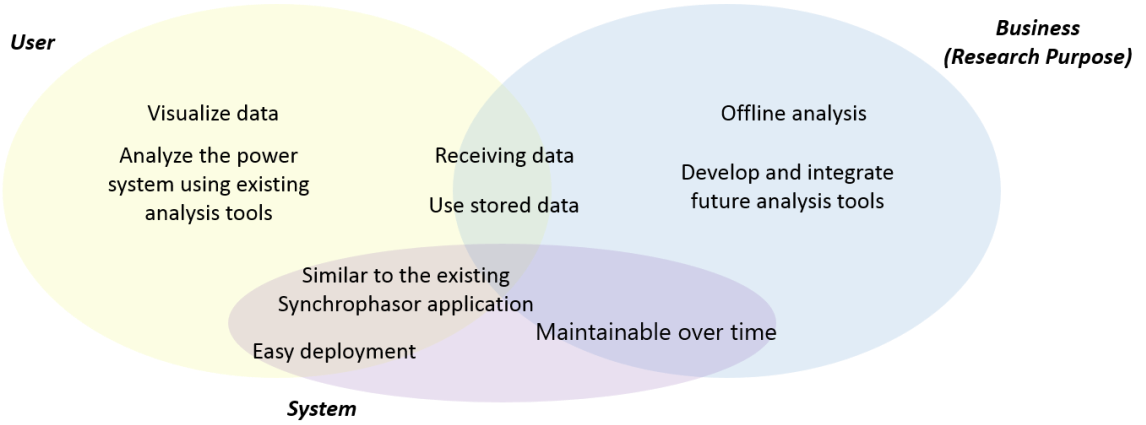


Figure 3.2 Reduced Venn diagram for the program goals

Based on the Venn diagram, for "PhasorEye" software, a 2-tier, component and layer - based hybrid architecture with .Net framework is selected to establish the foundation of the program structure. The main reason for selecting a 2-tier (Client / Server) desktop application approach for "PhasorEye" software is that most of the PDC software choose to use TCP/IP or other client and server type communication, and the communication network is considered as an independent close network. The obstacles between the client which is "PhasorEye" software and the server which is PDC is ignored. And since "PhasorEye" software is expected to be installed in a power system control room or other critical position, the cross platform problem can be ignored as well.

Four layers are included in "PhasorEye" software structure. On the top is the presentation layer which includes the entire user interface. In the middle is the business layer which contains the power system analysis algorithm, one peer layer in the same level is external system which allows external library to place. In the bottom is the data layer, which is handling the data receiving, reading, decoding and storing process. Figure 3.3, shows the four layer infrastructure for "PhasorEye" software.

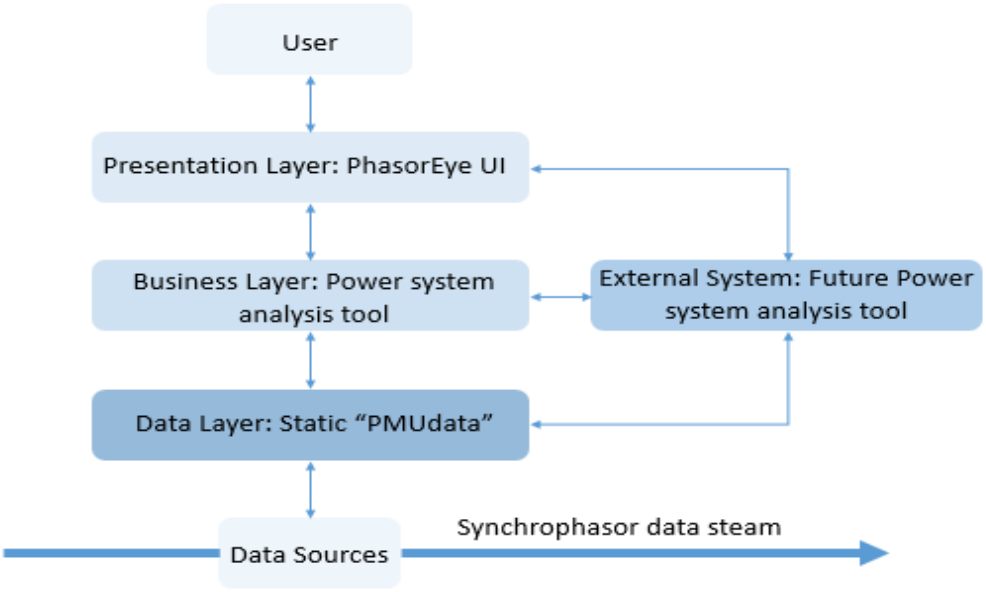


Figure 3.3 Four layer infrastructure for "PhasorEye" software

"PhasorEye" software chooses this architecture because the number of program layers are minimized; it divides the application into distinct layers with as little overlap in functionality as possible. In addition, Figure 3.4, presents the tools which are needed for the development of "PhasorEye" 2-tier program by using Microsoft development platform technologies. The .Net framework and C# programming language was chosen to develop "PhasorEye" software. Under this structure, two approaches can be adopted to start creating the program. They are Windows Forms and Windows Presentation Foundation and the details of implantation in both approaches for "PhasorEye" software is discussed in following sections.

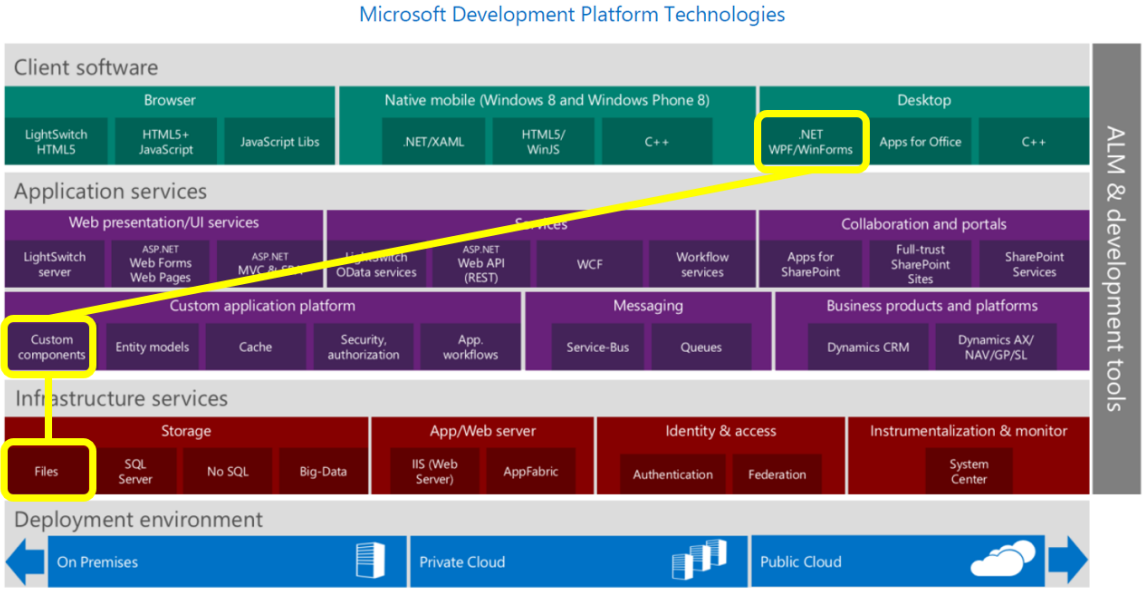


Figure 3.4 Microsoft development platform tools used by "PhasorEye" software

3.1.1 Parent child window and page navigation structure

For a Windows desktop application, Microsoft development platform technologies offer two approaches to build a program with the .Net framework which are Windows Forms and Windows Presentation Foundation. The main difference between Windows Forms and Windows Presentation Foundation is the flexibility of the user interface and thread operation. In this section, the details of developing "PhasorEye" software in both approaches are presented.

Windows Form is one of the oldest Windows desktop application approaches under the .Net framework. With a rich online example collection and recourse, Windows Form is an easy way for new developer to develop their application. One of the common structures which are used in Windows Form is the multiple-document Interface. The multiple-document Interface allows user to work in a parent window with more than one child window

and different documents at the same time. Figure 3.5, shows an example for such structure. The Multiple-document Interface is excellent choice for “PhasorEye” software, because it allows different power system analysis algorithm running at the same time. This approach provides a complete user interface library to support the structure, which means minimum code alternation for the auto generated codes is required for the program construction.

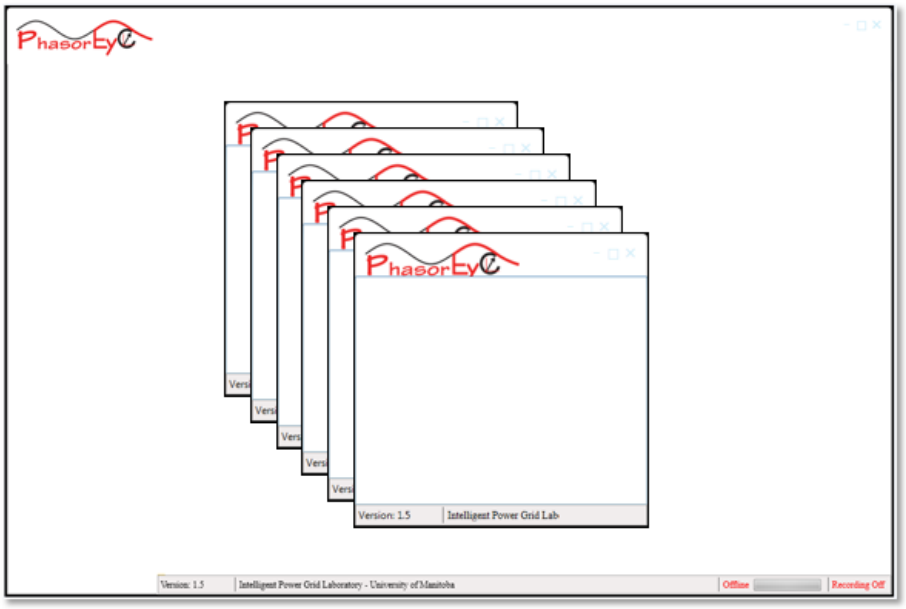


Figure 3.5 A Multiple-document Interface example by using “PhasorEye” software

An example of “PhasorEye” software using Windows Form and the multiple-document interface is shown in Figure 3.6. As in Figure 3.6, the configuration and display window are contained in the main window. Figure 3.7, shows the layer presentation for the program, inside the parent window “MIDparent”. Different child windows which contain different methods such as display, configuration and fault detection are constructed. Figure 3.8, shows the code map for the program, under the executable file, one domain contain twenty-one classes, since most of the functions and classes are called from a supporting library, the size of the program is small.

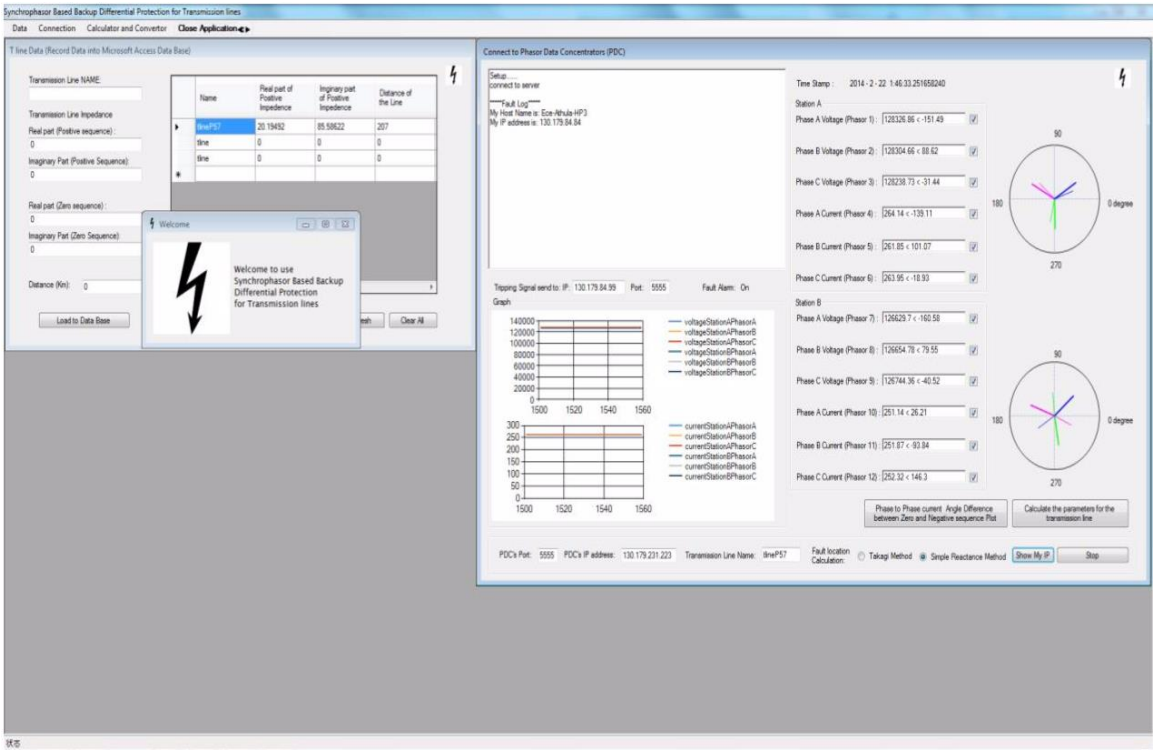


Figure 3.6 Snapshot of "PhasorEye" software by using Windows Form and multiple-document interface

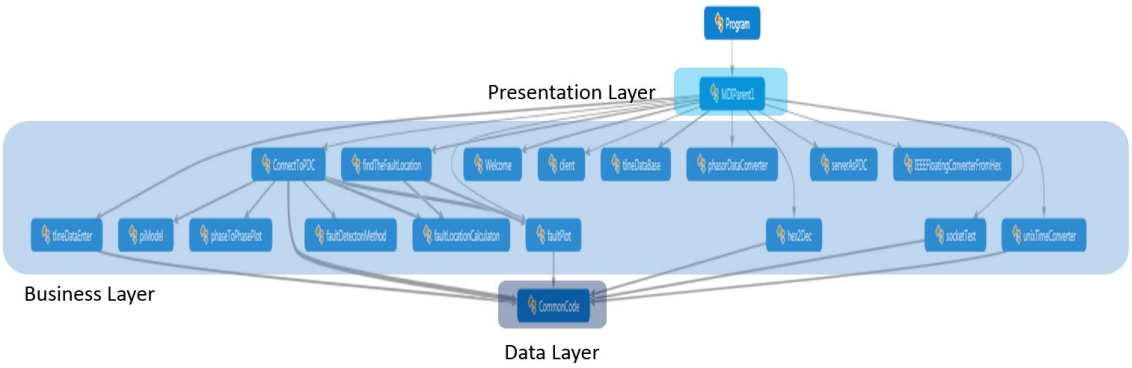


Figure 3.7 Layer presentation for "PhasorEye" software by using Windows Form and multiple-document interface

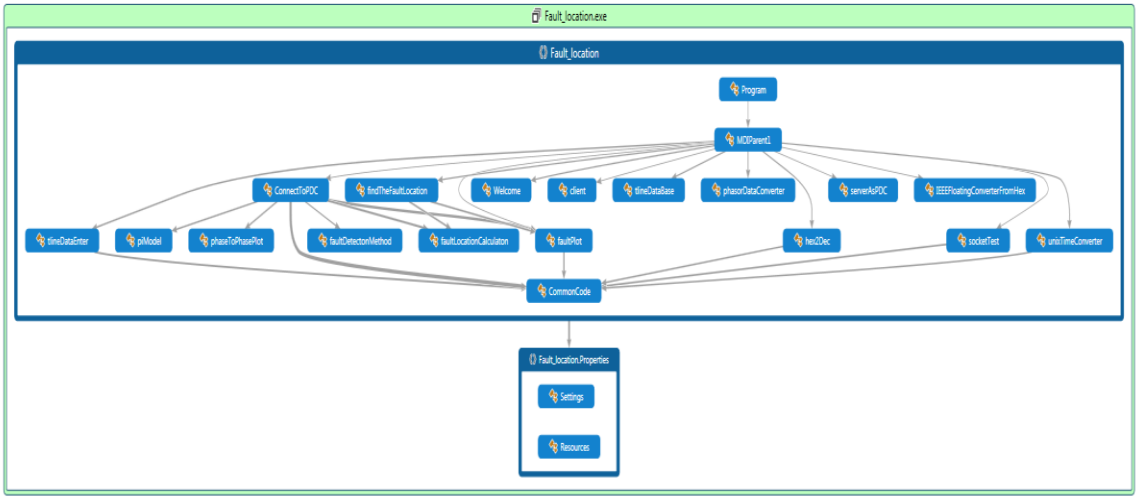


Figure 3.8 Implicit code map of "PhasorEye" software by using Windows Form and multiple-document interface

The advantage for Windows Form and the multiple-document interface is it offers easy and simple program skeleton. However, its drawbacks cannot be overlooked. Each child window is a separate unit; it does not depend on other child window, and only save one index to the parent window for the window searching. So if one algorithm produces a result which is required by other algorithm, since both algorithms are in separate child windows, the communication has to go through the upper level which is the parent window. Meanwhile most of the codes in the parent are generated by the library from Microsoft development platform, and the functions which are needed for the communication are packed with large overhead. Thus, the time delay associated with passing information between different child windows is a major concern, especially "PhasorEye" because software is intended to support real-time monitoring and analysis tools. In addition, another issue that can cause time delays is the user interface and algorithm window's thread control. The priority thread queue is hard to configure since the full access for all threads requires a substantial amount

of research for understanding the mechanism behind all the components in the user interface. In addition, Windows Form type application has challenges in program scaling and platform crossing. Initially, attempt was made to develop the software by using Windows Form, but the above problems could not be solved satisfactorily without affecting the online operation. Therefore, it was decided to develop a new software structure by using Windows Presentation Foundation. The development process and the advantages of Windows Presentation Foundation based software is explained in the next section.

Alternative approach for using Windows Form is building the program by using Windows Presentation Foundation. As mentioned before, one of the major differences between Windows Form and Windows Presentation Foundation developing environment is the flexibility, which is arising from the complete separation between the user interface and the background algorithm. Different algorithms can be run on the same window, messages for the user interface are handled by its user interface delegate and the messages between the algorithms are handled by the data layer. Overall time delay is minimized by customizing your own user interface component and sending to delegate to take care of the changes from the user. However, this feature means more codes need to be written by the developer. The program scaling and platform crossing is also solved by using Windows Presentation Foundation.

One of the common structures which are used in Windows Presentation Foundation is a navigation window with multiple pages. The main window which contains all algorithms is the navigation window, and it is using a navigation bar to switch the content inside. Figure 3.9, shows the main navigation window for “PhasorEye” software. Furthermore, in order to grant full access to any content inside the window, a separate menu bar is created

as in Figure 3.10. The left hand side menu controls the content shown to the user and this content is presented in the right hand side pages. Thus, the left hand side menu bar is operating in the program structure level, and the left hand side page is operating in the algorithm and application level. Figure 3.11, shows the connections between menu bar (left), pages (middle), and main window (right) for “PhasorEye” software.

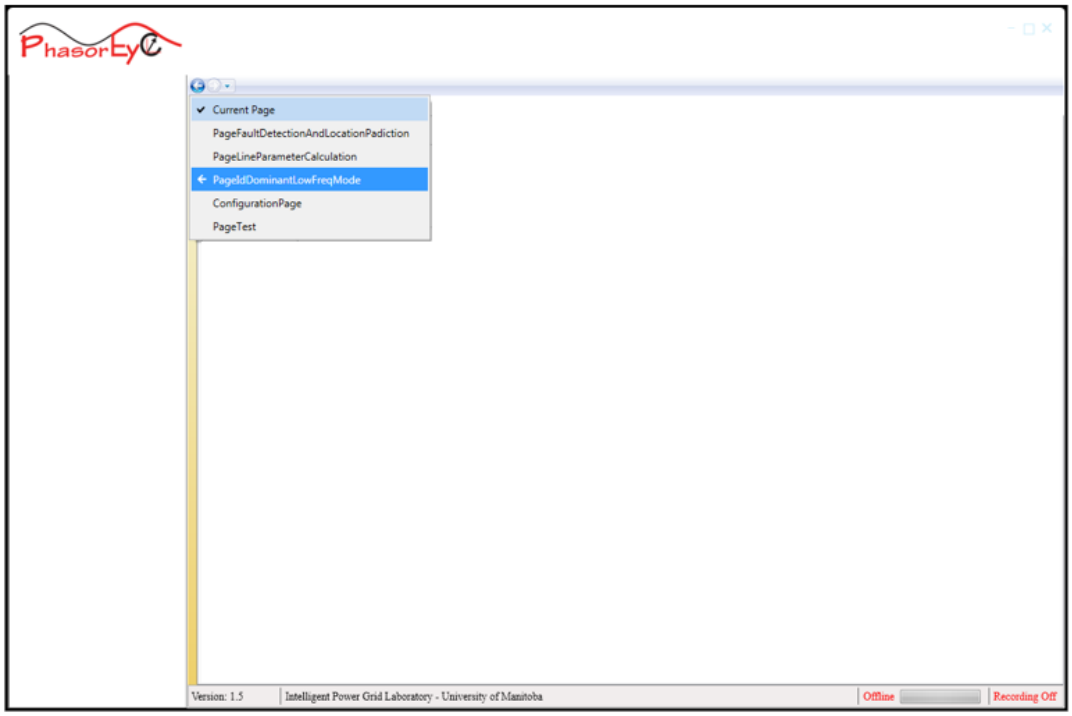


Figure 3.9 Main navigation window for “PhasorEye” software

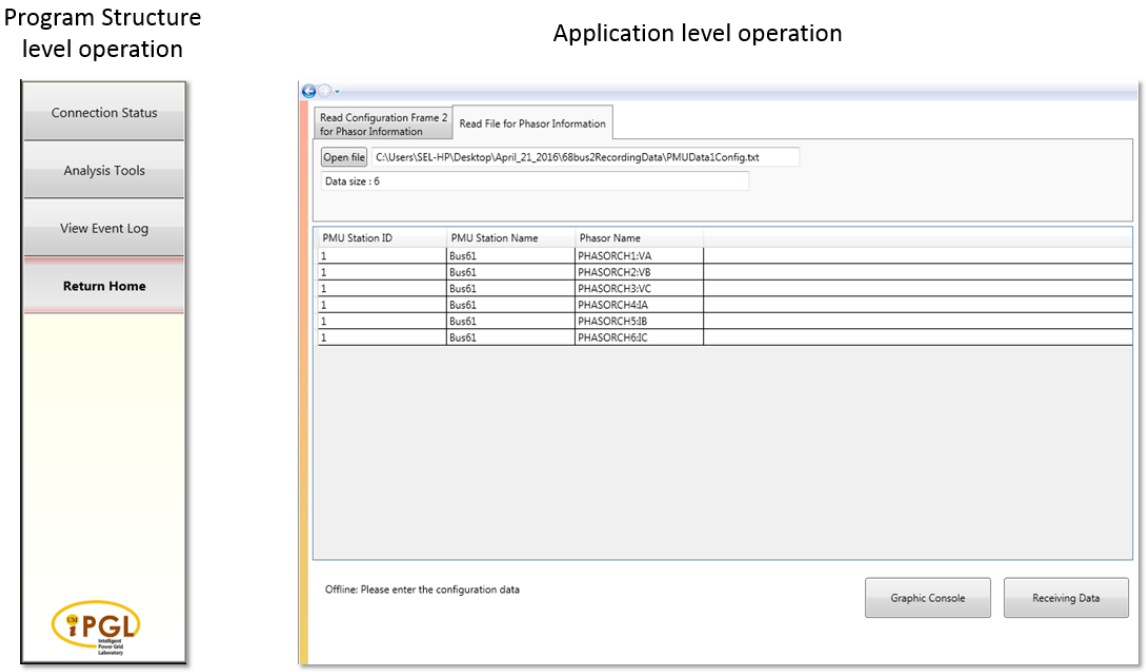


Figure 3.10 UI operation in different levels

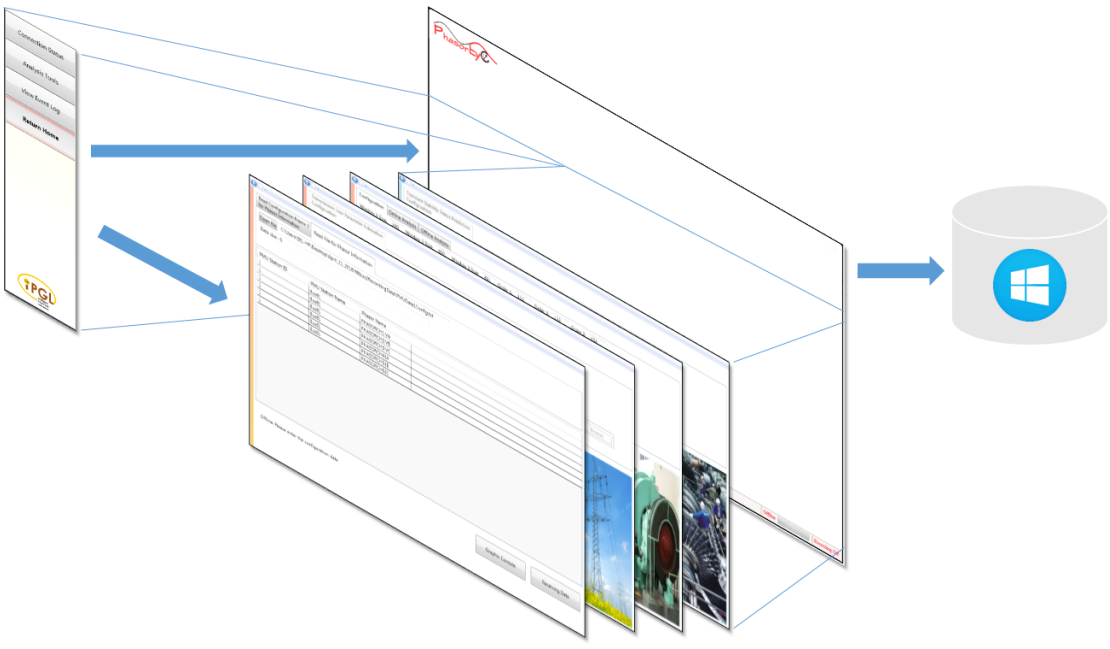


Figure 3.11 Connections between menu bar (left), pages (middle), and main window (right) for "PhasorEye" software

In Figure 3.12, the layer presentation for the program is shown, the main architecture is still using, the 2-tier, component and layer-based hybrid architecture. Figure 3.13 shows the code map for the program, under the executable file are ten domains containing all the classes with the variables and functions associated with the user interface and application algorithm. Since most of the functions and classes are written by the developer, most of the code are filed in the individual domain to have a more manageable access instead of placing in one window class. Therefore, the size of the application with Windows Presentation Foundation page and the navigation window structure is bigger than the application with the Windows Form and parent child window structure.

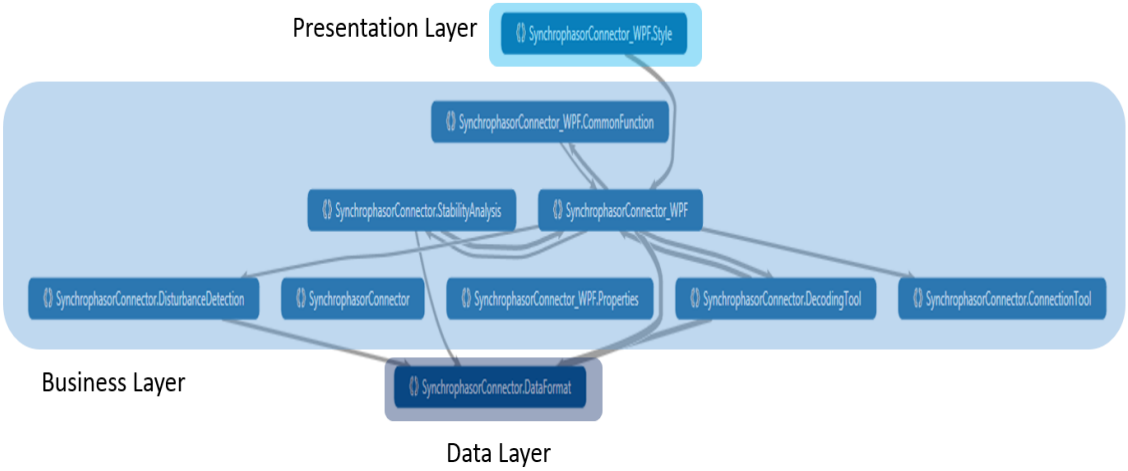


Figure 3.12 Layer arrangement for "PhasorEye" software by using Windows Presentation Foundation and the navigation window structure

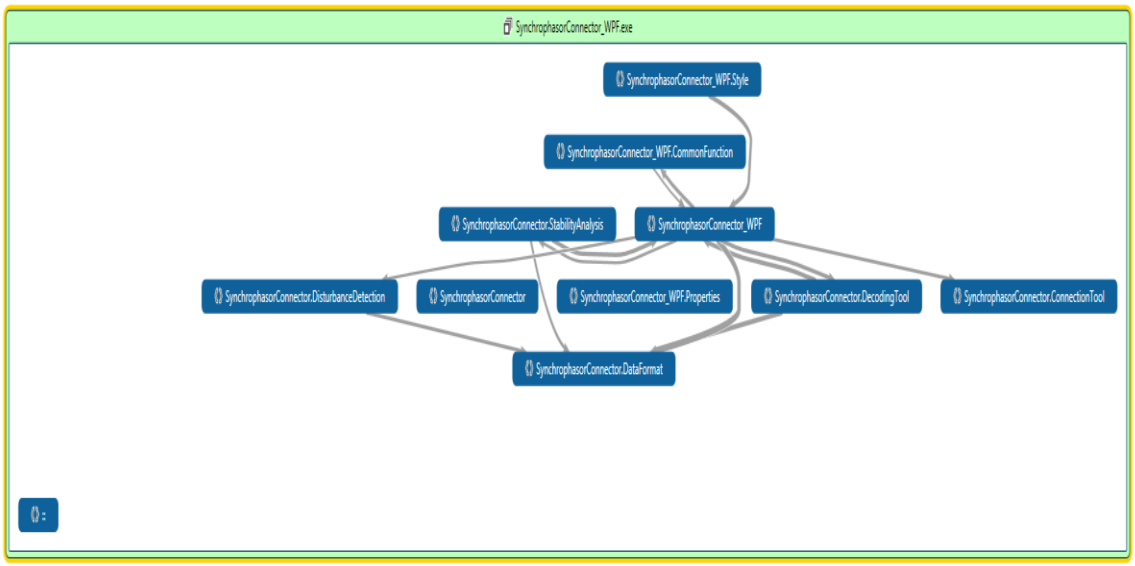


Figure 3.13 Implicit code map of "PhasorEye" software using Windows Presentation Foundation and the navigation window structure

3.1.2 Program components

In this section, the main components and the details of the presentation layer in "PhasorEye" software are presented. Figure 3.14 is a class diagram for "PhasorEye" software from Visual Studio.

Figure 3.14 "PhasorEye" software's class diagram

In the presentation layer, the main window is a container for all the content which includes the left hand side menu, right hand side page display and bottom status bar. It is the first window to create after the user clicked “PhasorEye” software icon. Figure 3.15 is a snap-shot of main window with the starting home page. The “App”, “WindowMain”, “StartingWindow” and “Setting” class creates the core part of the main window. To enforce the style influence from main window to other application pages, the “resource” class, and other style components is used in the background. During the creation of main window, the “StaticItem” class is also created by the program. Figure 3.16, shows the main window and its classes.

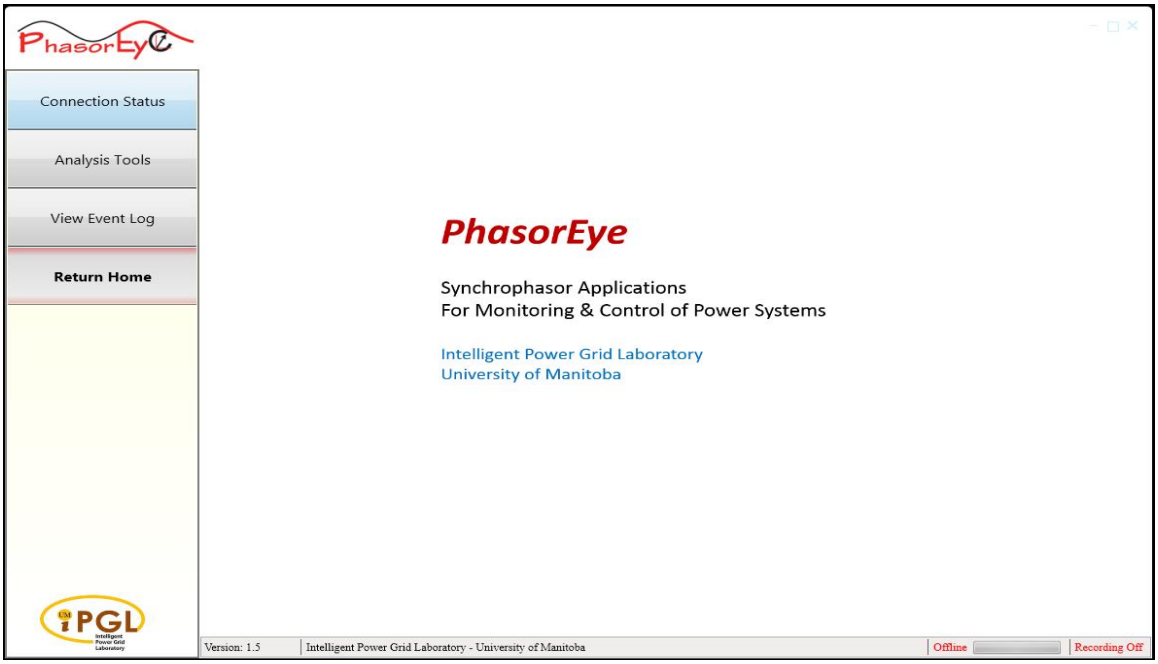


Figure 3.15 Snapshot of main window with the starting home page

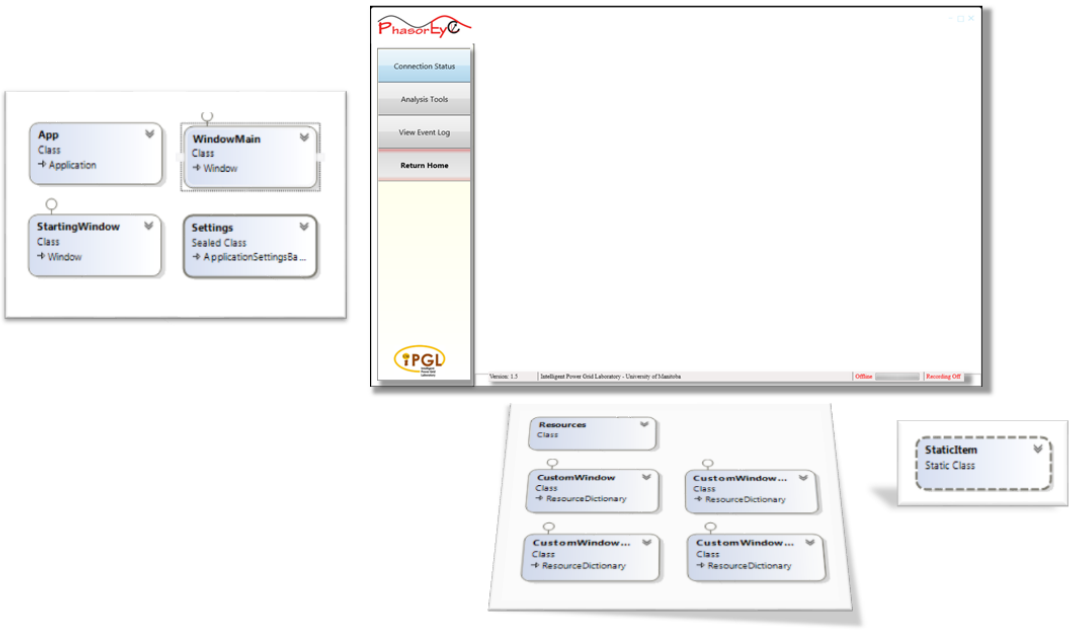


Figure 3.16 Main window and its classes

One of the advantages for the Windows Presentation Foundation is the free access and customization for the user interface. The components in the user interface are not defined, for example the developer can create a component with the appearance as a button but with the event logic as a drop down list. This unique ability allows "PhasorEye" software to have a set of unified user interfaces, however for each user interface the background algorithm can be different depending on the different applications in the business layer.

In Windows Presentation Foundation, graphical user interface is written in Extensible Application Markup Language and it is stored in a file with the extension ".xaml". One of the ways to have a connection between the graphical user interface and its background logic is to create a user interface component is by using a resource dictionary class. A resource dictionary class is a keyed dictionary of objects that can be used both in the ".xaml" file

and in the code. “PhasorEye” software is using a resource dictionary to create a template window or style components for other windows inside the software to follow.

The style components are stored in the folder “Style” in a file named as “CustomWindow_special_request.xaml”. The head of the style components are a traditional “.xaml” file which the XML namespaces is included to avoid the name conflicts. The body of the file is separated into three parts. First part is the property for the window menu, such as the foreground and opacity. Second part is the other component which is also needed in the window template, such as a side bar or other special requests from other components such as the Microsoft Bing map module. In the end, is the property for the window background such as the transparency of the window and resize ability. Figure 3.17 presents a sample of the “.xaml” file structure for the component styles.

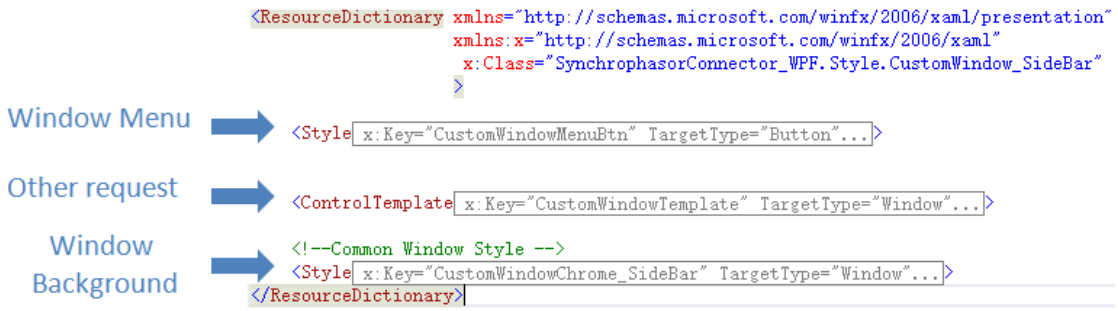


Figure 3.17 Simple structure for the style component

The background logic for style component is saved in a file named “CustomWindow_special_request.xaml.cs”. Usually there are only three event handlers inside; namely the minimize window event, resize window event and close window event. But more events can be introduced in the style component, if such events are requested by the business layer components.

The style component is not a standard class from the Visual Studio user interface library. Thus the style component needs to be registered in the starting point of the application which is the "App.xaml" file. Figure 3.18 presents the "App.xaml" with the style components in "PhasorEye" software highlighted.

```
<Application x:Class="SynchrophasorConnector.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="StartingWindow.xaml">
    <Application.Resources>

        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="Style/CustomWindow.xaml"/>
                <ResourceDictionary Source="Style/CustomWindow_SideBar.xaml"/>
                <ResourceDictionary Source="Style/CustomWindow_NoClose.xaml"/>
                <ResourceDictionary Source="Style/CustomWindow_NoClose_Map.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>

    </Application.Resources>
</Application>
```

Figure 3.18 "App.xaml" with the style components used in PhasorEye highlighted

The style component is added to the header of the windows which are inside "PhasorEye" software. Figure 3.19, presents an example from the information window from "PhasorEye" software. Figure 3.20 presents the appearance of the same window with and without using the style component.

```
<Window x:Class="SynchrophasorConnector_WPF.WindowInformation"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="InfoWindow" Height="300" Width="300"
        Style="{StaticResource ResourceKey=CustomWindowChrome_NoClose}"
        WindowStartupLocation ="Manual"  Icon="phasoreyelogo2_vpH_5.ico"
        >

    <Grid>

        <TextBox x:Name="TextBoxLog" Height="Auto" Margin="0,0,0,0" TextWrapping

    </Grid>
</Window>
```

Figure 3.19 Information window with the style components

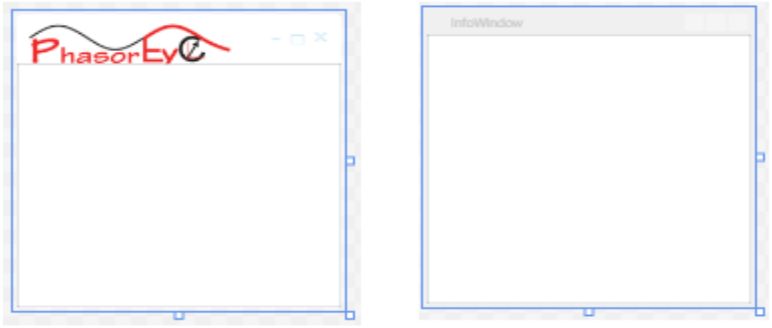


Figure 3.20 A window using the style component (left) and without using the style component (right)

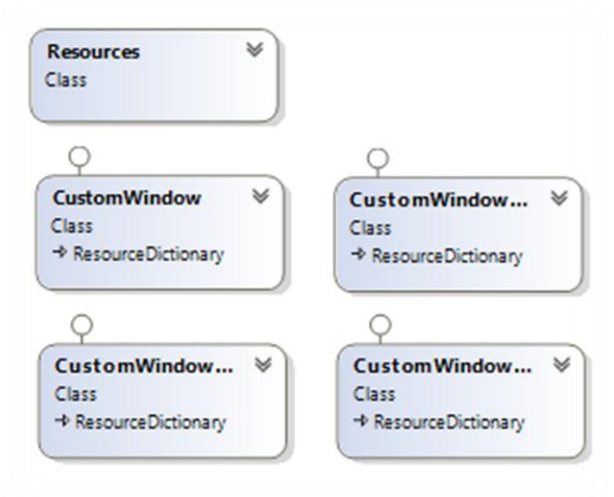


Figure 3.21 Classes for the style components

Figure 3.21 presents the class diagram for the class which is relating or serving as the style components from Visual Studio. The “Resources” file is an auto generated file by visual studio, which contains the list of the pictures which are used in the program, such as icons or button styles. “CustomWindow.xaml” file contains the basic visual aids and the functionalities such as the icon, font style, minimize, resize and close events. “CustomWindow_NoClose.xaml” file is for the real time applications in the business layer which cannot be closed. For example the log window of “PhasorEye” software is created after software is started by user. However if user double clicks the log window’s close button, user interface of the window is hided from the user, but the thread for the log window is still alive to allow other application in “PhasorEye” software to write into program’s log. “CustomWindow_NoClose_Map.xaml” is a special window template for the windows using the Microsoft Bing map module. Microsoft Bing Map components require a connection to internet to allow them download the map from Microsoft Bing map website after the window is created, so the thread for such window cannot be closed in order to avoid multiple downloading. “CustomWindow_SideBar.xaml” is for the main window, which has a left hand side sidebar granting an access to all the applications and algorithms in the business layer.

3.2 Connection and display

In this section, the main components of the data layer in “PhasorEye” software for the synchrophasor network connection and data processing are presented.

3.2.1 TCP connection and receiving buffer

In “PhasorEye” software, connection process and receiving buffer process is handled in the business layer classes “PageConfiguration”, “PageDisplay”, “CommandFrame” and “DataFrame”, and the data layer classes “StaticItem” and “PMUdata”. Detail of the configuration frame and data frame decoding is presented in the section titled “Configuration local data structure and data frame decoding”.

The main connection protocol which is used in “PhasorEye” software is the Transmission Control Protocol and the Internet Protocol (TCP/IP). In the default synchrophasor network connection, client server network is commonly used. PDC is acting as the server and other application is acting as the client to request the data from PDC. Flow chart of the process for connection in “PhasorEye” software is presented in Figure 3.22.

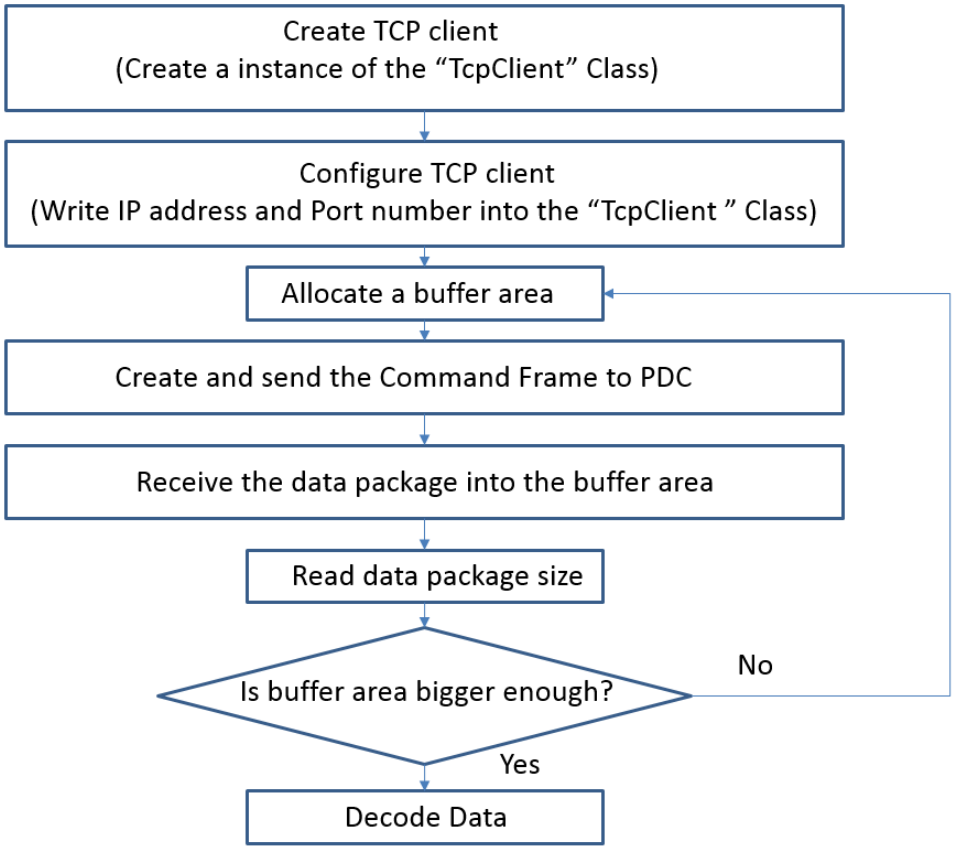


Figure 3.22 Flow chart of the process of the connection in "PhasorEye" software

Under the .Net framework, namespace "System.Net.Sockets" provides a set of tools which is from the windows socket interface to allow the developers who need to access the network to manage the internet connection. Under the namespace "System.Net.Sockets", "TcpClient" class provides client connections for the Transmission Control Protocol.

Two sets of instances of the "TcpClient" class in created in algorithm which is running in the "PageConfiguration" and the "PageDisplay" connection thread. IP address and port number for the data stream from the PDC is provided by the user from the user interface, the time span for the waiting is set to one second. If "PhasorEye" software does not receive

the data package from the server, one exception is thrown by the "result.AsyncWaitHandle.WaitOne" function, and that exception is catch by the operation thread and report to the log window to do the exception recording. In Figure 3.23, the code for the first two steps of the connection process is presented.

```
try
{
    int port = int.Parse(TextBoxPort.Text);
    string ip = TextBoxIp.Text;
    StaticItem.IpAddress = ip;
    StaticItem.Port = port;
    StaticItem.Id = int.Parse(TextBoxId.Text);

    tcpClinet = new TcpClient();
    var result = tcpClinet.BeginConnect(ip, port, null, null);
    var success = result.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));
    TextBlockConsole.Text = TextBlockConsole.Text + "\nconnecting\n";
}
catch (Exception ex)
{
    TextBlockConsole.Text = ex.ToString();

    StaticItem.logWindow.WriteToAppLog(ex.ToString());
    StaticItem.logWindow.Update();
}
```

Figure 3.23 Code for TCP/IP connection in "PhasorEye" software

In addition, a small buffer area is created for the receiving data package. The minimum size of the buffer area is four bytes, because the information for the size of the data package is in the third and fourth byte of the data package. In order to retrieve all of the data in the data package, the third and fourth bytes are combined into a 32 bits integer number and cross checked with the size of the buffer area before any decoding process. If the buffer area is not big enough to contain all the data inside the data package, a new buffer area with the correct size is created. "PhasorEye" software will resend the command frame, and the new receive data will be written into the new buffer area. In Figure 3.24 the code for the above of the connection process is presented.

```
//first communication get the size of the frame
networkStream.Write(msguploadstop, 0, msguploadstop.Length); //sending sto
networkStream.Write(msguploaddata, 0, msguploaddata.Length); //sending com
networkStream.Read(dummyBuff, 0, dummyBuff.Length);

//Re communicate with PDC again and use the currect size buff (msgReceived)
int frameSize = dummyCommandFrame.twoByteToInt(dummyBuff[2], dummyBuff[3]);
msgReceived = new byte[frameSize];

networkStream = tcpClinet.GetStream();
// Set a 10 millisecond timeout for reading.
networkStream.ReadTimeout = 10;

networkStream.Read(msgReceived, 0, msgReceived.Length);
```

Figure 3.24 Code for reconnection process

3.2.2 Static item and Phasor measurement data format

In this section, two of the major components of the data layer, the “StaticItem” and “PMUdata” class are presented.

The “StaticItem” class is a special class inside the “PhasorEye” software. All fields inside the “StaticItem” are the static variables. A static variable has properties that hold their value beyond any scope. The “StaticItem” class is acting as a global static variable holder in “PhasorEye” software. The “StaticItem” is separated into two parts. The first part is holding the data in user interface of “PhasorEye” software for the presentation layer. The second part is holding the data structure which includes low level bytes data buffer and high level data structure class “PMUdata” for “PhasorEye” software’s data layer.

Figure 3.25 presents an abstract program structure of the user interface in “PhasorEye” software. The “StaticItem” class is created after the main window is created. It holds all the information which are included in the page which is hosted in the main window and other pages which are hiding in the background. For each page, one or more tab controls are inserted to host the algorithm or application tabs. Inside the algorithm or application

tab, one thread is operating a power system analysis algorithm in the background. Figure 3.26 is a snap shot of "PhasorEye" software which illustrates "Page", "Tab" and "Thread". More details about the "StaticItem" class, page, tab and algorithm and application threads for the presentation layer of "PhasorEye" software are presented in Chapter 5.

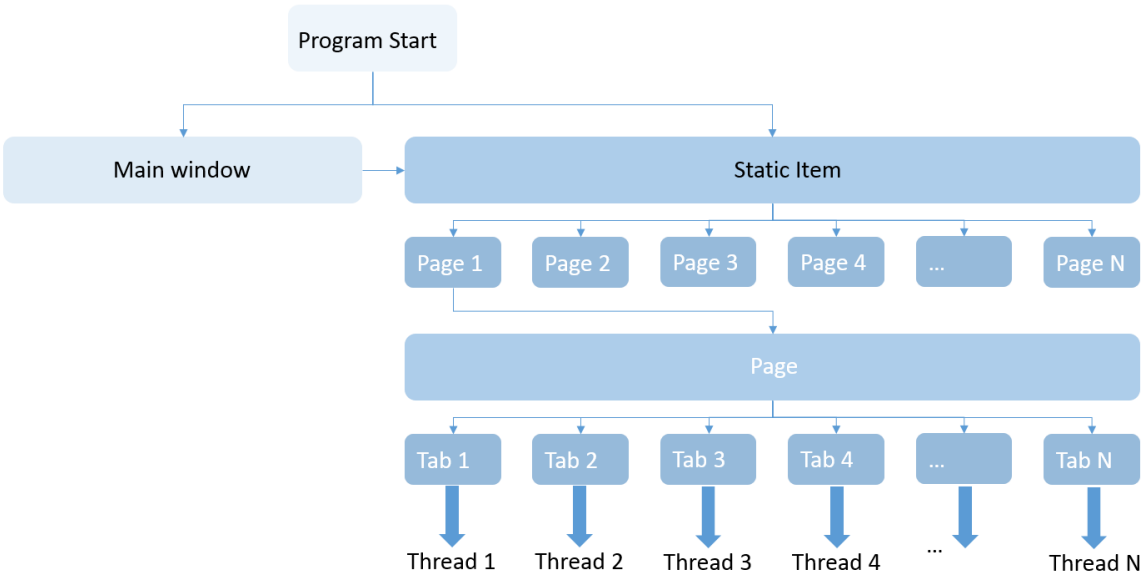


Figure 3.25 Abstract structure of the user interface

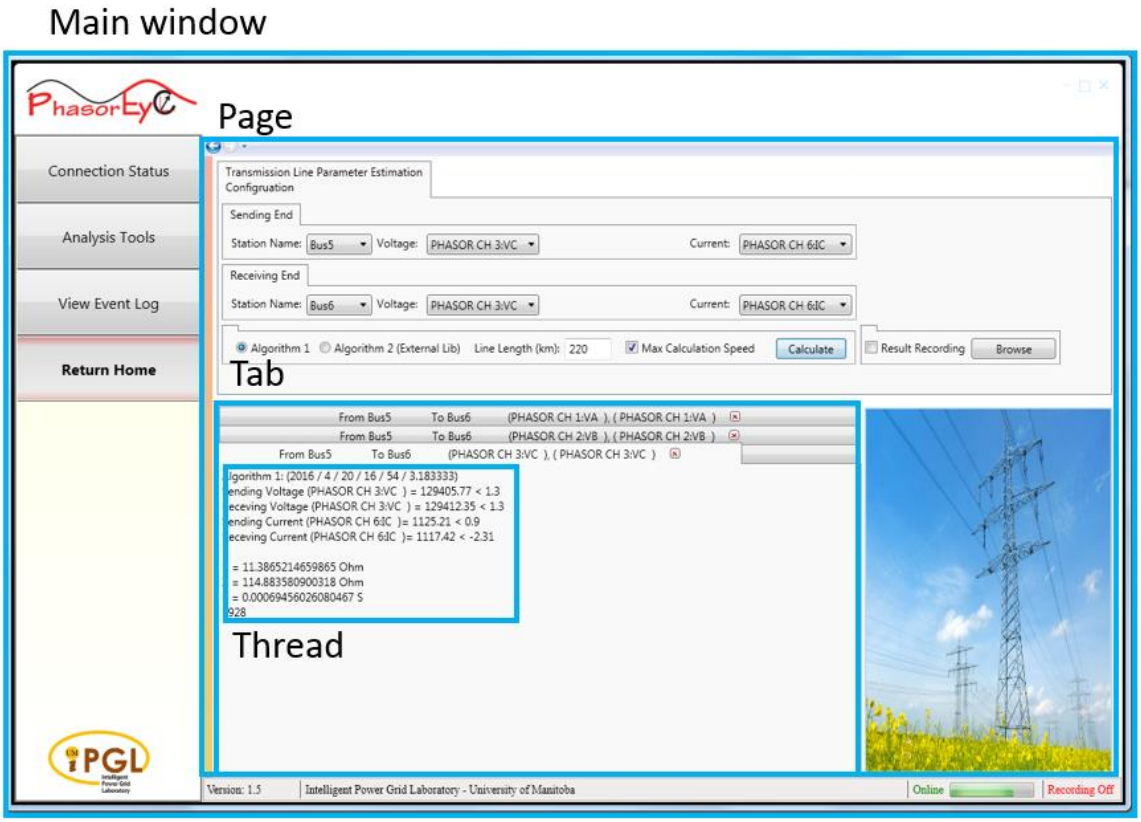


Figure 3.26 Structure for the page, tab and thread

"StaticItem", "PageConfiguration" and "PageDisplay" classes contain a set of connection process threads, which allows them to communicate with PDC and receive configuration and data package. After the connection is established in the "PageConfiguration" or "PageDisplay" class, "isOnline" field in "StaticItem" class is marked as true by the connection thread. All other online application pages require to check the "isOnline" field to confirm whether all the connections are well established before proceed with their respective application algorithms. Figure 3.27 presents the "isOnline" field inside the "StaticItem" class's class diagram.

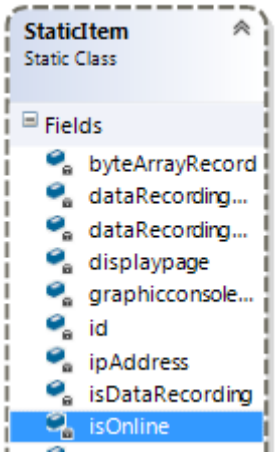


Figure 3.27 The "StaticItem" class's "isOnline" field

The second part of the "StaticItem" class is holding the data structure of "PhasorEye" software. For each synchrophasor data package received from the data stream, two copies of its instance is generated. The first copy is in raw hex bytes array format and stored in the buffer area. The second copy is an object array of the high level data structure class "PMUdata" in "StaticItem" class. The connection between two copies of the instance is the decoding process. Figure 3.28 presents the abstract structure for the connection between the data layer and the business layer in "PhasorEye" software.

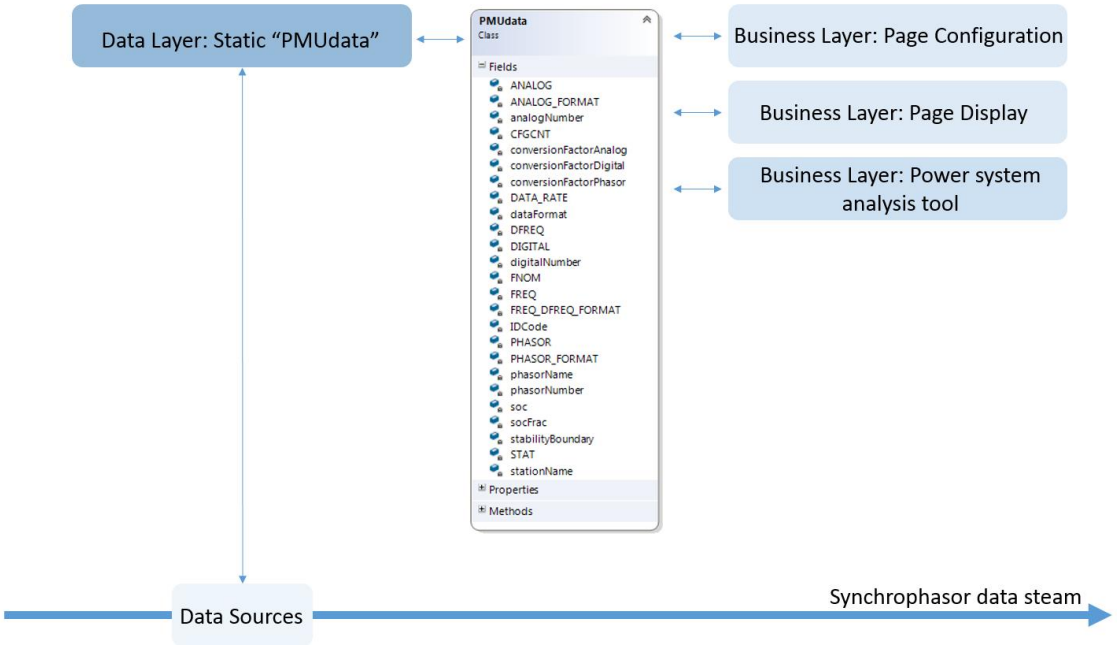


Figure 3.28 Abstract structure for the connection between the data layer and the business layer

The file for the data structure class "PMUdata" is inside "DataFormat" folder. It is a template for the synchrophasor data which is used in "PhasorEye" software under the namespace "SynchrophasorConnector.DataFormat". However, other applications should not use the data template directly; instead they should call the data structure from the "StaticItem" in order to access the data template.

```
Chart chart = this.FindName("ChartVoltage") as Chart;
chart.ChartAreas["ChartArea"].AxisX.LabelStyle.Enabled = false; //Disable x axis
chart.Series["SeriesSeries"].Points.AddXY(StaticItem.Pmu[dataSelection].getTimeStamp(),
    StaticItem.Pmu[dataSelection].getPhasor()[dataTypeSelection].Magnitude);
```

Figure 3.29 Codes for accessing time stamp and voltage magnitude

In general, any application and algorithm in the business layer can obtain the real-time data from the data layer "PMUdata" class array inside the "StaticItem" class. Figure 3.29 shows the code for updating the real time plot for the "x" axis as the time stamp and "y" axis as the magnitude of the phasor voltage in the data display page.

3.2.3 Configuration of data structure and data frame decoding

In this section, decoding of the data package into the data structure class “PMUdata” is presented.

Inside “PhasorEye” software’s “StaticItem” class, “PageConfiguration” and “PageDisplay” classes have a connection process thread, which allow them to communicate with the PDC and receive configuration and data frames. After the TCP client connection and the size of the buffer area is determined, the hand shake process is followed by sending the command frame from “PhasorEye” software to the PDC asking to send the synchrophasor configuration frame or start sending data frames.

Command frame is an eighteen bytes array, which contain the information about the sync bytes, number of the bytes in the frame, ID code for the PDC or PMU, time stamp and type of the data is required. Command frame ends with two cyclic redundancy check bytes. In “PhasorEye” software, command frame is created by using the function “CreateCommandFrame” under “ConnectionTool” folder. Figure 3.30 presents the flow chart for “CreateCommandFrame” function and connection thread.

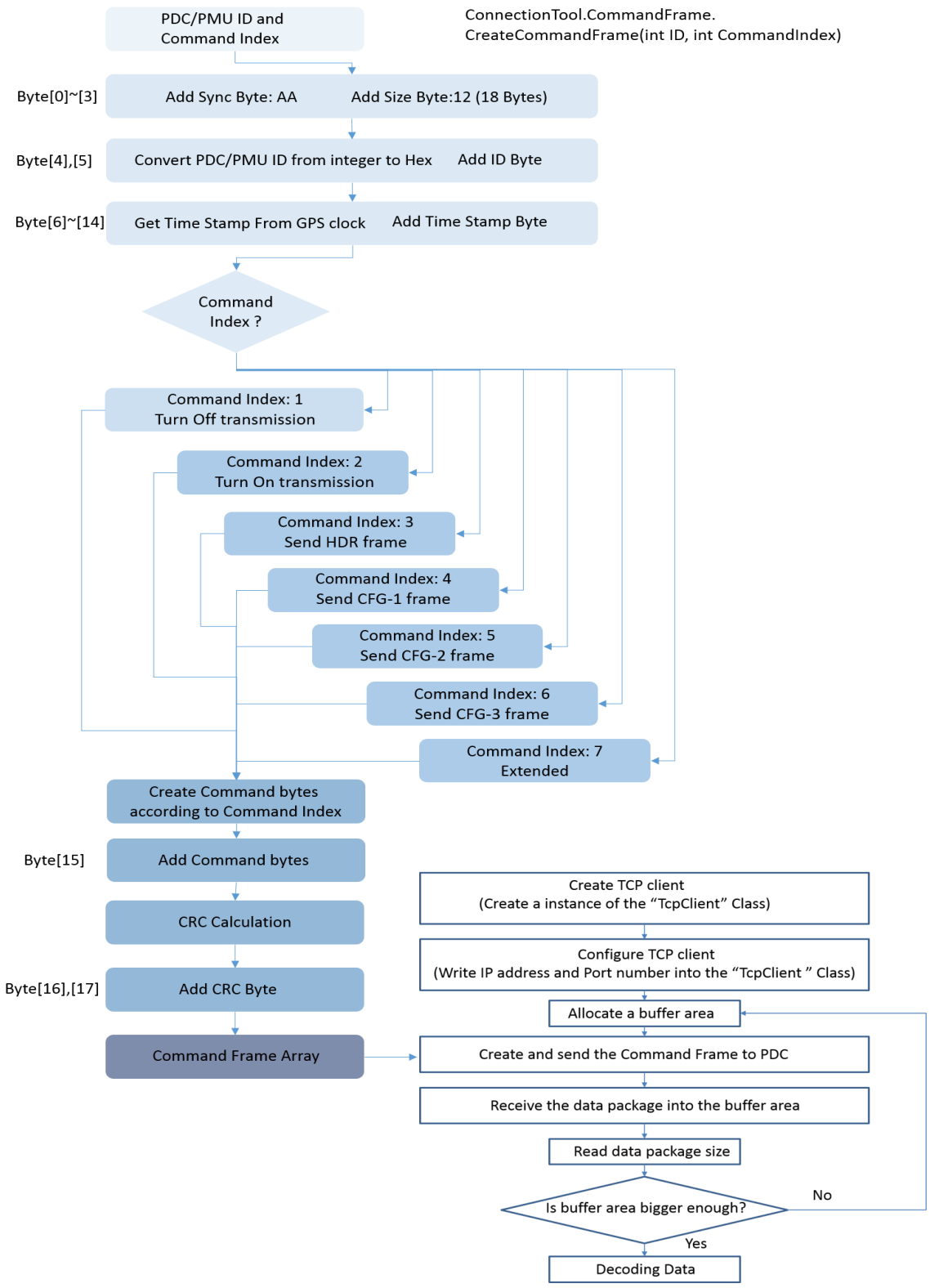


Figure 3.30 Flow charts for "CreateCommandFrame" function and connection thread

As in the flow chart, after the connection is established inside the connection thread, command frame with command index “1” which indicates all current transmission should stop is sent by the connection thread to PDC. The second command frame is sent by the connection thread is with command index “5” which indicate the synchrophasor configuration frame “CFG-2” frame is required or with command index “2” which indicate starts the real time synchrophasor data frame transmission.

After the communication thread confirmed the configuration, decoding process is initiated by the connection thread. In “PhasorEye” software, decoding process for the configuration frame allows “PhasorEye” software to construct an array of the data template structure “PMUdata” class inside the data layer part of “StaticItem” class. The decoding process for the data frame allows “PhasorEye” software to fill the real time synchrophasor data into “PMUdata” class structure array and allows other application to get access.

All decoding functions are located in the “DecodingTool” folder. Figure 3.31 shows the abstract structure of decoding process. Figure 3.32 shows “ConnectionTool”, “DataFormat”, and “DecodingTool” folders and their functions in the Visual Studio solution explorer.

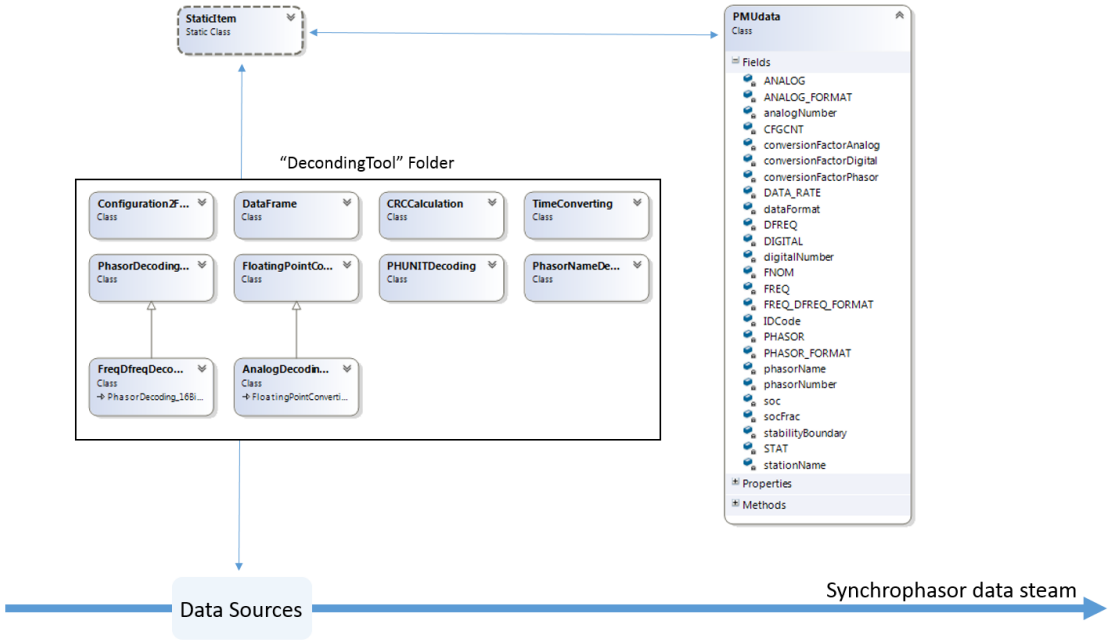


Figure 3.31 Abstract structure for the decoding process

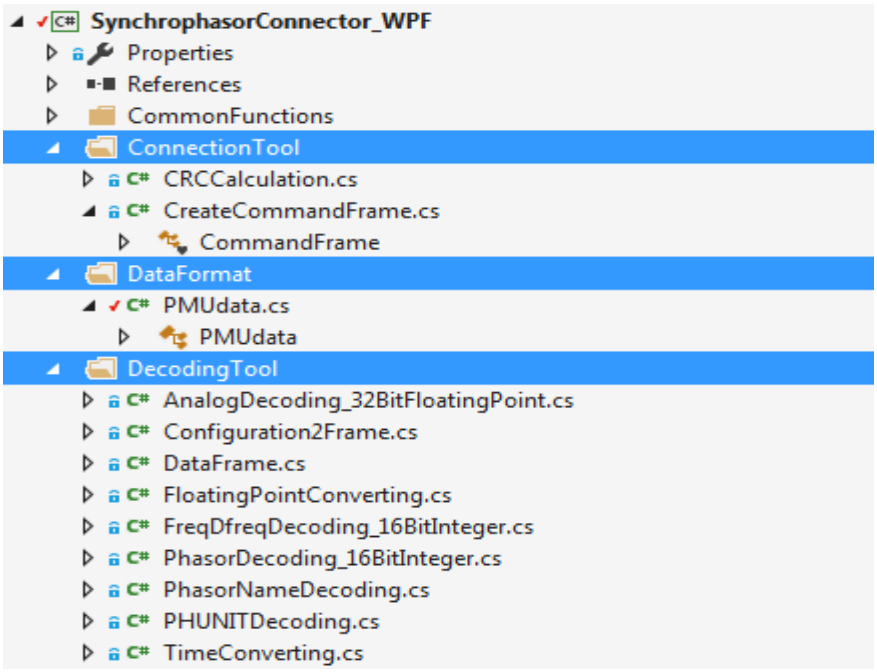


Figure 3.32 The "ConnectionTool", "DataFormat", "DecodingTool" folder and their functions

For communication thread inside the “PageConfiguration” class under the configuration page, command frame with command index “5” is sent to the PDC to request the configuration frame. After the synchrophasor configuration frame “CFG-2” is received in the buffer area, “processingConfigurationFrameTwoData” function is called by the configuration page communication thread to provide the configuration frame decoding. Figure 3.33 shows a snapshot of “PhasorEye” configuration page. Figure 3.34 shows the abstract structure of “PhasorEye” software with the configuration page and its operation threads.

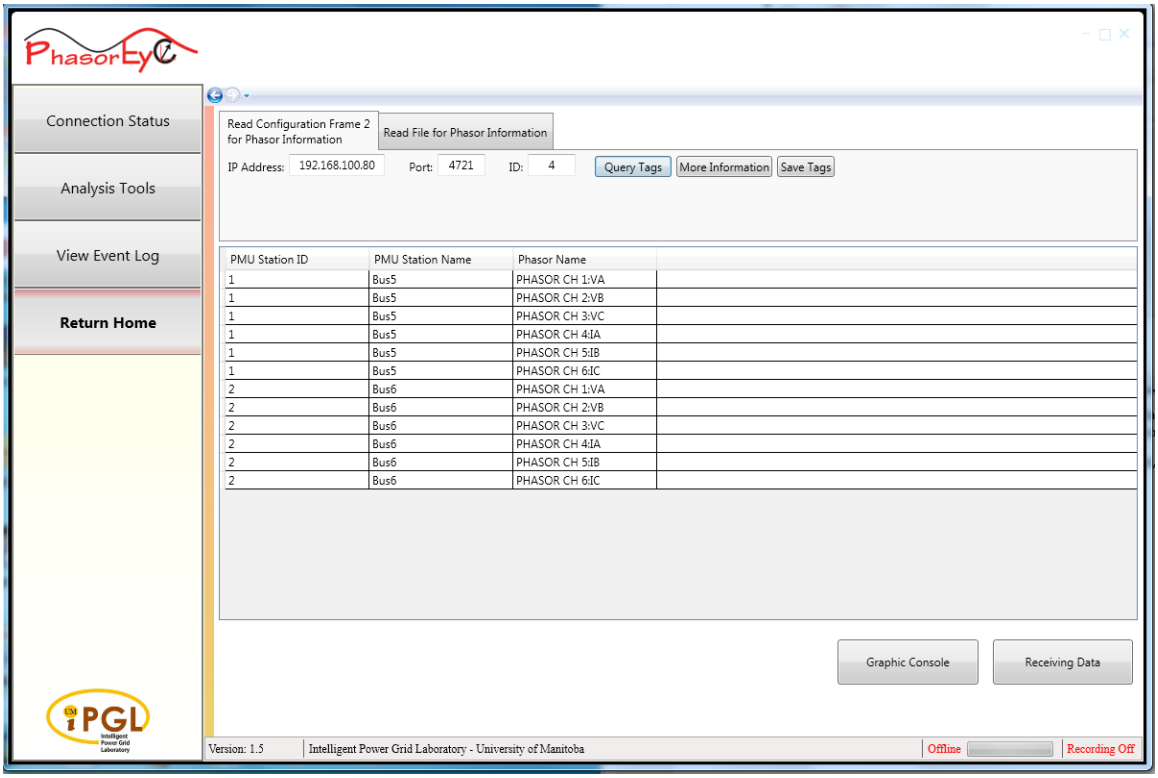


Figure 3.33 Snapshot of “PhasorEye” software configuration page

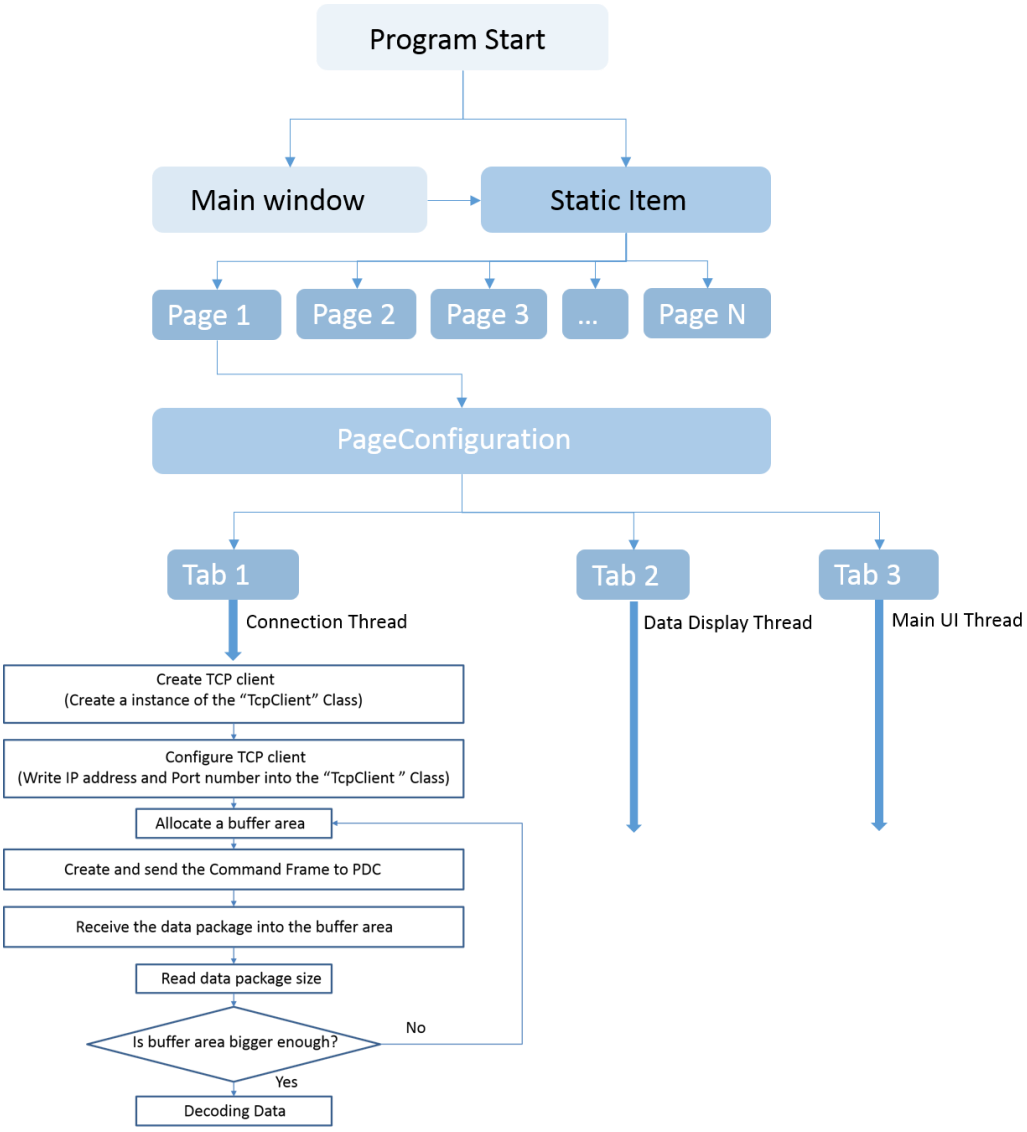


Figure 3.34 Abstract structure of "PhasorEye" software with the configuration page and its operational threads

The "processingConfigurationFrameTwoData" function is under domain "DecodingTool", and class "Configuration2Frame". Input for the "processingConfigurationFrameTwoData" function is a byte array from the TCP connection buffer area. The configuration frame is separate into two parts; first part is the header which includes the size of the frame, type of the frame, ID code for the PDC, time stamp, time format and number of the PMUs inside

the configuration frame. The second part is the body of the configuration frame, which includes station name for each PMU, phasor name for each phasor data, name of the analog signals, name of the digital signals inside each PMU and the conversion factor for each phasor data, analog signal and digital signal. Body part of the configuration frame may repeat multiple times, according to number of PMUs inside the configuration frame. "ProcessingConfigurationFrameTwoData" function decodes configuration frame, and creates an array of the "PMUdata" class. Each element inside "PMUdata" class array represents a data set from one PMU. Figure 3.35 shows the basic flow chart for the "processingConfigurationFrameTwoData" function.

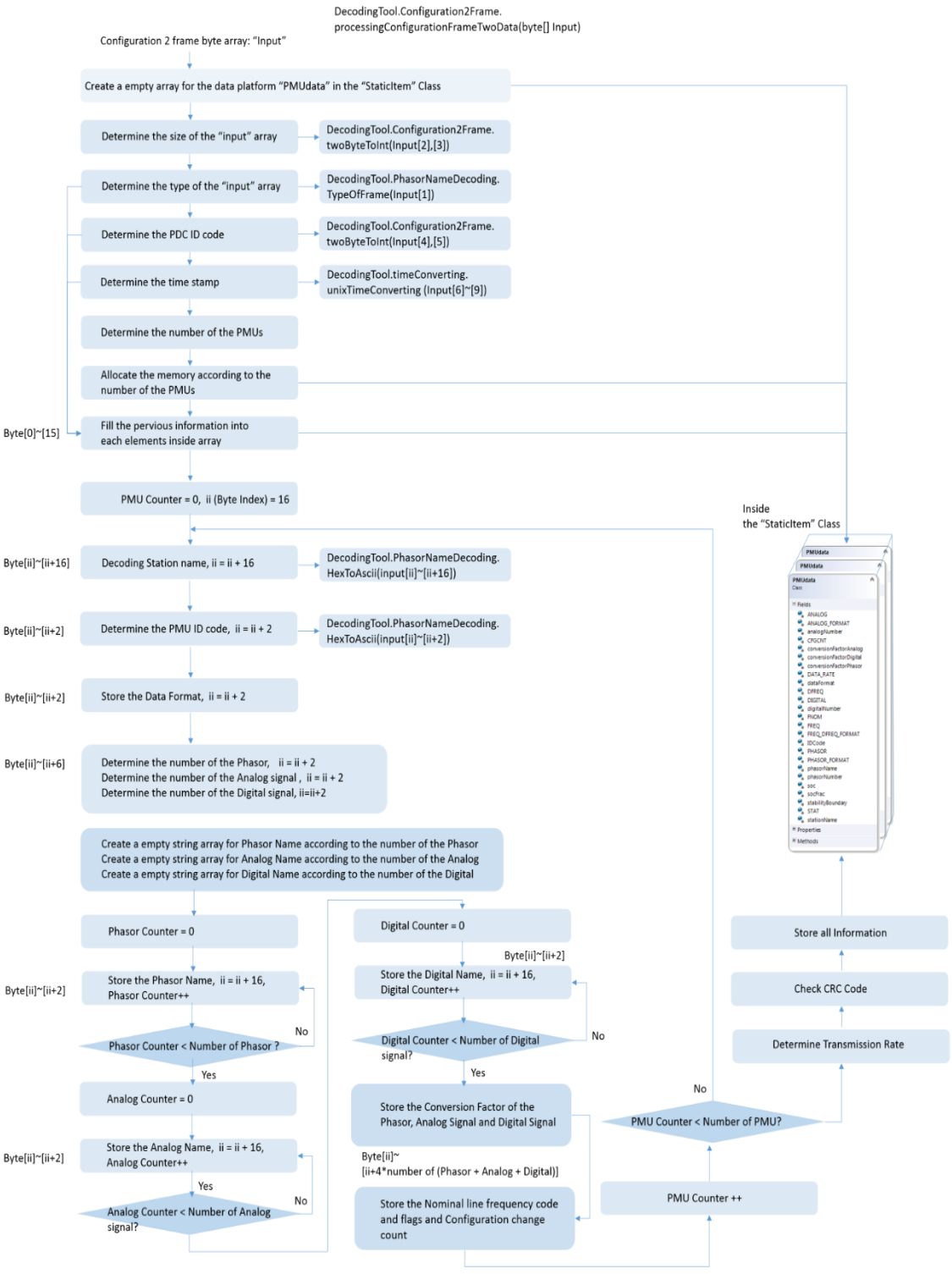


Figure 3.35 Flow chart for the "processingConfigurationFrameTwoData" function

Decoding process for the data frame is very similar to decoding process for the configuration frame. For communication thread inside “PageDisplay” class under display page, command frame with command index “1” and “2” is sent to the PDC to turn off and then turn on the real time synchrophasor data stream. After synchrophasor data frame is received in buffer area, “processingDataFrame” function is called by display page communication thread to provide data frame decoding.

“processingDataFrame” function is under domain “DecodingTool”, and class “DataFrame”. Input for the “processingDataFrame” function is also a byte array from the TCP connection buffer area. Data frame is also separated into two parts, first parts is the header which is similar to the configuration frame’s header but has no the time format field. Second part is the body of the data frame, which includes phasor data, analog signals, digital signals, frequencies and change of frequencies for each PMU inside data frame. Same as the configuration frame, the body part of the data frame may repeat multiple times, depending on the number of PMUs inside the data frame. “processingDataFrame” function decodes the data frame according to the conversion factor inside the “PMUdata” class inside “StaticItem” class’s “PMUdata” class array, and fills the “PMUdata” class with decoded real time synchrophasor measurements. Figure 3.36 shows the basic flow chart for “processingDataFrame” function.

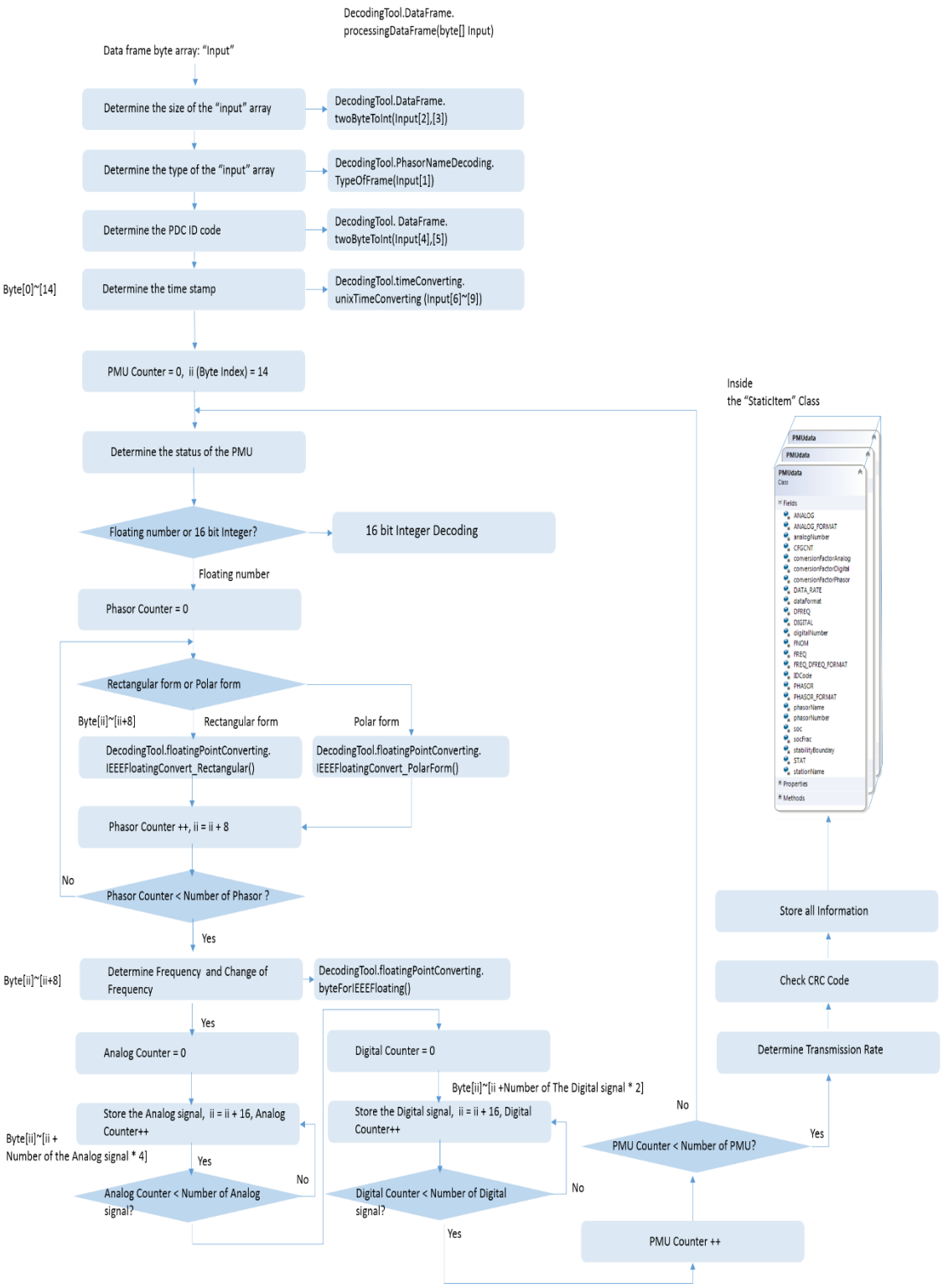


Figure 3.36 Basic flow chart for the "processingDataFrame" function

After the decoding process, other applications can call the data structure from “StaticItem” to access the data template to get real time synchrophasor measurements. Figure 3.37, shows the snapshot of the data display page. Data display thread in the display page gets the measurements from “StaticItem” class and plot data on its user interface.

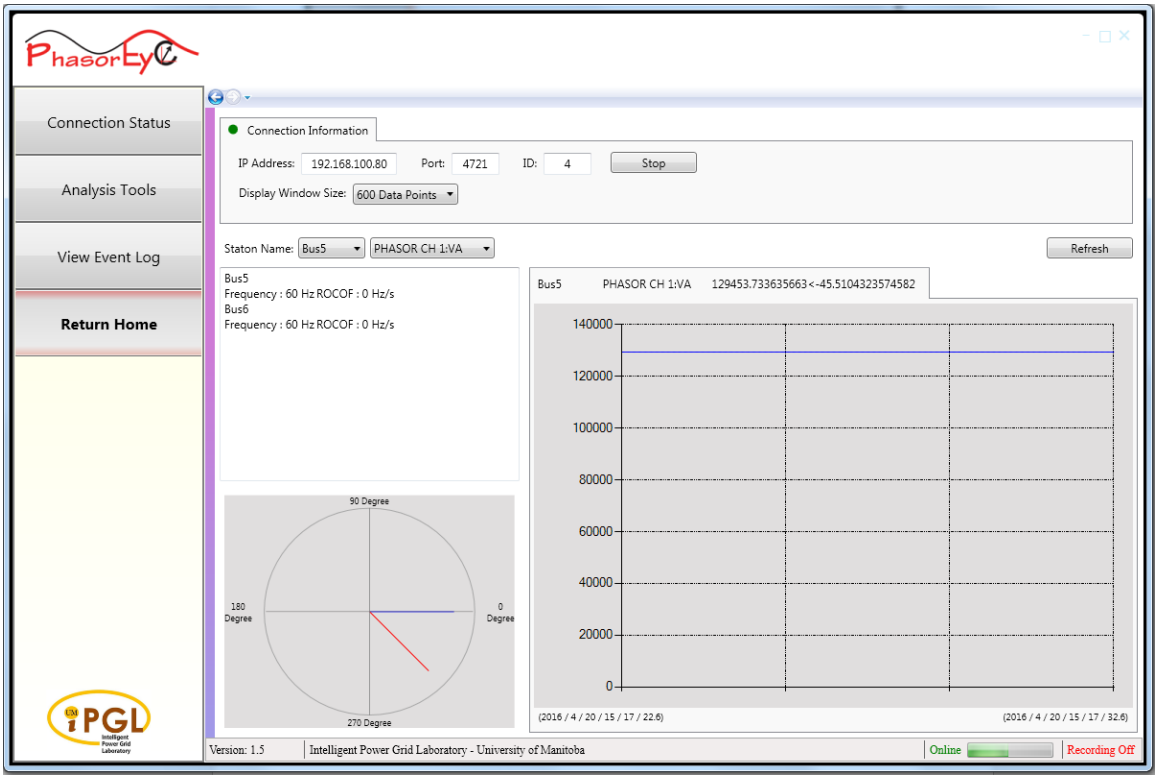


Figure 3.37 A snapshot of the data display page in “PhasorEye” software

3.3 Chapter summary

In this chapter, “PhasorEye” software’s 2-tier, component and layer -based hybrid architecture with .Net framework is presented. In addition, main window, style components and other major components for presentation layer are presented. Finally, the connection process, decoding process for data layer are presented.

Chapter 4

Fault locator for multi-terminal transmission lines

Power transmission lines play a vital role in the supply of electricity. Faults on transmission lines interrupt the power supply to the destination until the repairs are carried out. As for the line protection, it is very important to quickly detect the faults and send the tripping signals to appropriate switches to clear the fault. When a permanent fault happens, crews need to be sent to the right location to repair the faulty transmission line as soon as possible. Accurate knowledge of fault location can reduce down times as well as money, especially when a transmission line is long and passes through difficult to access terrains.

Most of the modern line protection relays are equipped with fault locators based on measurements from one end. The impedance based fault location algorithms used in the protection relays work well for double ended transmission lines and give good fault location estimates under practical conditions. However, the accuracy of fault location algorithms based on the measurements from a single end can be reduced due to fault resistance and power flow. Such errors can be eliminated by using synchrophasor based double ended fault location algorithms such as the one described in Section 4.1.5.

The fault location algorithms designed for point-to-point transmission lines cannot be directly applied to multi-terminal transmission lines. Multi-terminal transmission/distribution lines occur when a point-to-point line is tapped to supply a remote load or connect remote generation source. Generally, protection relays and circuit breakers are not employed at the tapping point; only isolator switches are used. The main challenge in fault location a multi-terminal transmission line is to determine the faulted segment. Once the faulted segment is established, the fault location can be accurately determined using the same approach used for double ended lines. Some of the previous work on multi-terminal line fault location was reviewed in Section 2.3. In this chapter, development, implementation, and testing of a novel threshold free synchrophasor based algorithm to identify the fault location when a fault occurs in a multi-terminal non-homogeneous transmission lines is presented. Furthermore, challenges in practical implementation of synchrophasor based fault location algorithms are discussed and solutions are proposed.

4.1 Algorithm for the location of faults in a multi-terminal transmission line

The presentation of the fault location algorithm for multi-terminal lines is organised as follows. Section 4.1.1 introduces the structure of the multi-terminal transmission line and the notation used in proposed algorithm. Section 4.1.2 describes the procedure for the estimation of tapping node voltage and current values from the measurements at the nodes or terminals which are connected to it. The proposed fault location detection algorithm is developed in three steps. Section 4.1.3 describes the first step in the algorithm which is the

bisection search for detecting the tapping node connected to the faulty branch. Section 4.1.4 describes the second step which is the identification of faulty branch. Section 4.1.5 describes the final step which is the calculation of the exact fault location in the faulty branch. Section 4.1.6 introduces the idea of virtual tapping node and a method to use it for non-homogeneous transmission lines.

4.1.1 Topology of multi-terminal transmission line

Multi-terminal transmission line is assumed to consist of one main line and tapping branches as shown in top section of Figure 4.1. No measurements are done at tapping nodes, but at each terminal it is assumed that a PMU is installed to provide real-time synchrophasor measurements to the PDC. Each tapping node is connected to two main branches, and one or many tapping branches. In order to provide bisection search, the proposed algorithm estimates the tapping node quantities (voltage and branch currents) in both forward and backward directions. When the calculations are done in the forward direction, the branch to the right of relevant tapping node in the main line and all other tapping branches are considered as branches in the forward direction, as shown in the middle of Figure 4.1. On the other hand, when the calculations are done in the backward direction, the branch to the left of the considered tapping node in the main line and tapping branches are considered as branches in the backward direction as illustrated in bottom of Figure 4.1.

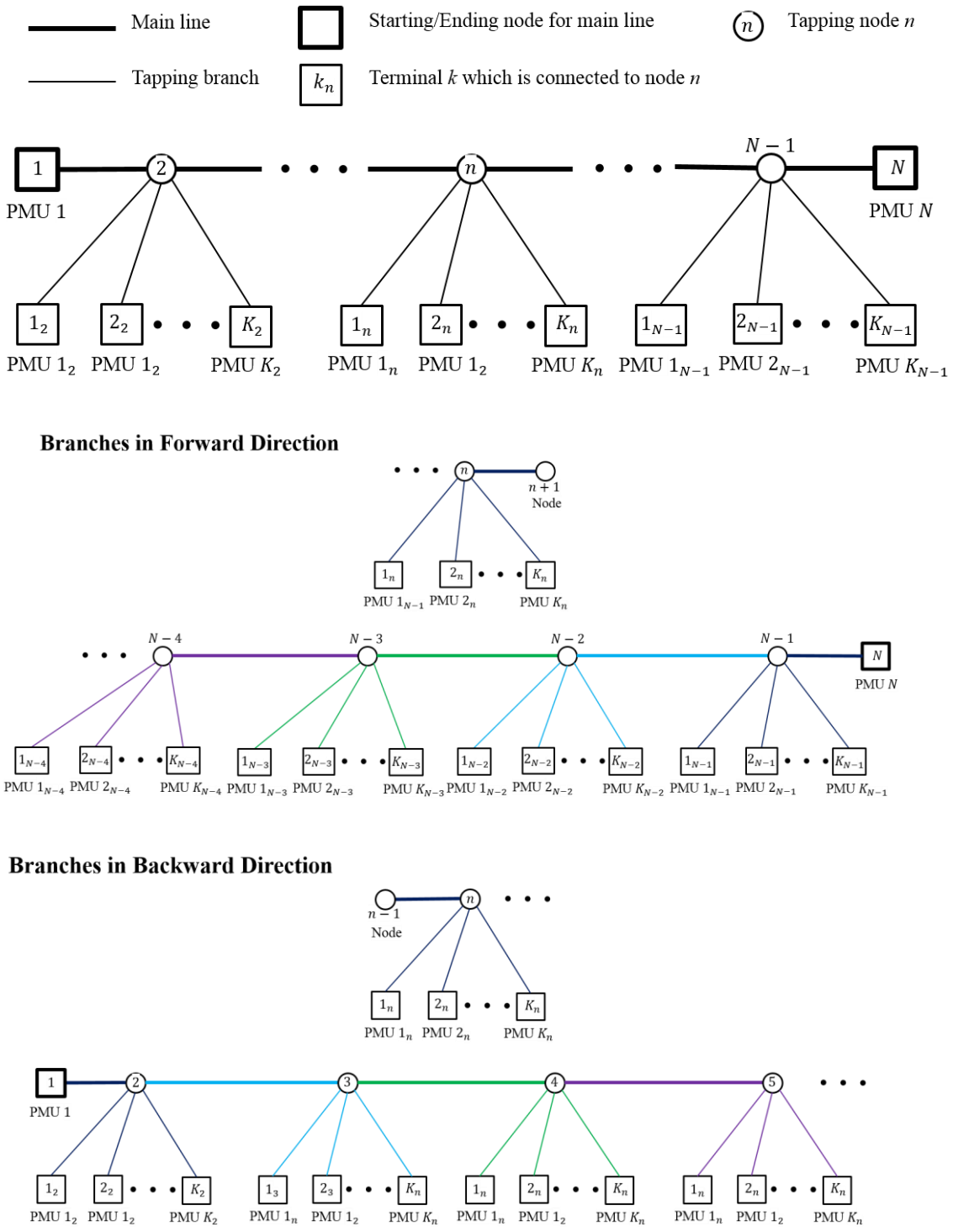


Figure 4.1 Topology for multi-terminal transmission line

4.1.2 Estimation of the tapping node voltage and currents

Consider a multi-tapping transmission line with the main branch from terminal A to terminal B as shown in Figure 4.2. The i^{th} tapping node is connected between the main branch and p^{th} terminal.

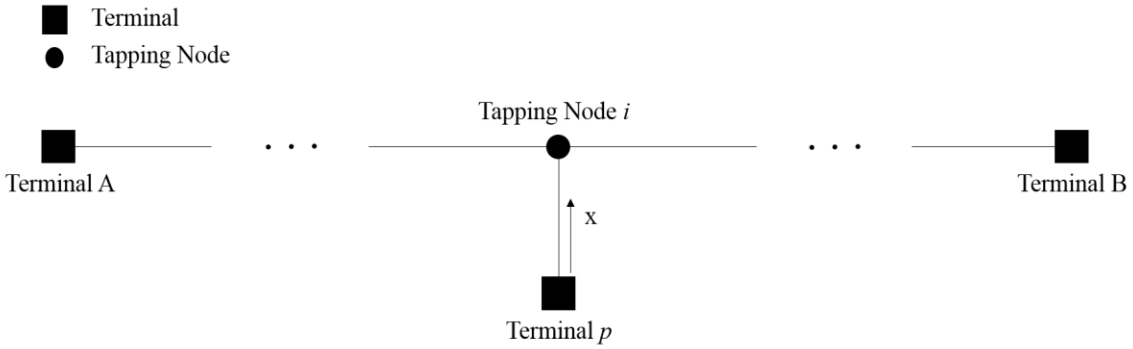


Figure 4.2 A tapping node connected with one terminal

Single phase, distributed parameter representation of the transmission line segment between the p^{th} terminal and the i^{th} tapping node is shown in Figure 4.3. The long line model equation is present in [44]. Assume that the total length of the line is l meters [44].

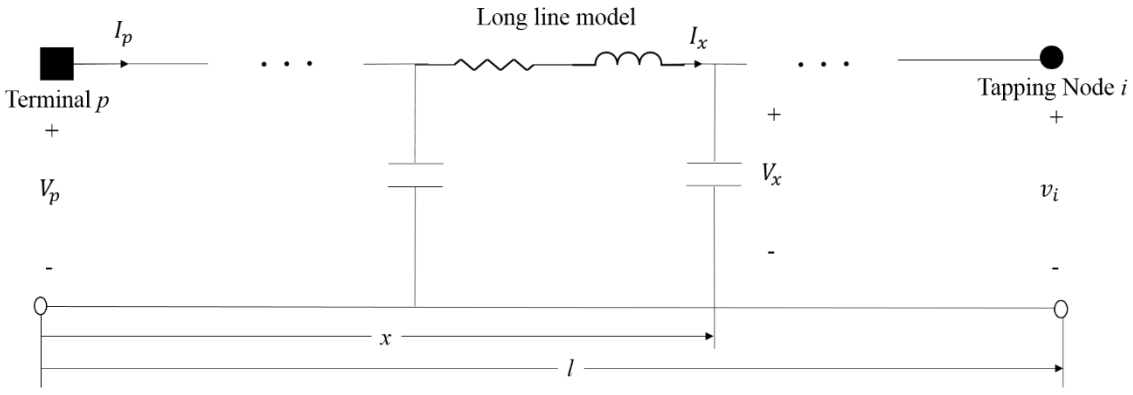


Figure 4.3 Long line modal for the transmission line segment between a tapping node and a terminal

Considering a small section of the transmission line with a length *dx* located *x* meters away from p^{th} terminal [44], transmission line equations for the voltage and current *x* meters away from the p^{th} terminal can be expressed as [44]:

$$\frac{\partial v_p}{\partial x} = R_{p,x} + L_{p,x}\frac{\partial i_p}{\partial t} \tag{4.1}$$

$$\frac{\partial i_p}{\partial x} = G_{p,x} + C_{p,x}\frac{\partial v_p}{\partial t} \tag{4.2}$$

Where $R_{p,x}$ and $L_{p,x}$ are series resistance and inductance from terminal *p* up to a distance *x*, and $G_{p,x}$ and $C_{p,x}$. are the shunt conductance and capacitance for the same distance. The solution to the transmission line equations is:

$$V_x = Ae^{\gamma x} + Be^{-\gamma x} \tag{4.3}$$

$$I_x = \frac{Ae^{\gamma x} - Be^{-\gamma x}}{Z_c} \tag{4.4}$$

Series resistance and inductance per unit length for the line connecting the p^{th} terminal and the *i*th tapping node are $r_{p,i}$ and $l_{p,i}$. The shunt conductance and capacitance per phase per unit length are $g_{p,i}$ and $c_{p,i}$. Then the characteristic impedance Z_c is given by

$$Z_c = \sqrt{\frac{r_{p,i} + jwl_{p,i}}{g_{p,i} + jwc_{p,i}}} \tag{4.5}$$

The propagation constant γ is given by

$$\gamma = \sqrt{(r_{p,i} + jwl_{p,i}) * (g_{p,i} + jwc_{p,i})} \tag{4.6}$$

Suppose voltage V_p and current I_p represents one set of measurements taken at the p^{th} terminal. Then, the unknown A and B can be solved as

$$A = \frac{V_p - I_p Z_c}{2} \tag{4.7}$$

$$B = \frac{V_s + I_s Z_c}{2} \tag{4.8}$$

Solutions for the second order (4.1) and (4.2), given in (4.3) and (4.4) becomes

$$V_x = \frac{V_p - I_p Z_c}{2} e^{\gamma x} + \frac{V_p + I_p Z_c}{2} e^{-\gamma x} \tag{4.9}$$

$$I_x = \frac{\frac{V_p - I_p Z_c}{2} e^{\gamma x} - \frac{V_p + I_p Z_c}{2} e^{-\gamma x}}{Z_c} \tag{4.10}$$

(4.9) and (4.10) can be rearranged and expressed in the form of

$$V_x = \cosh(\gamma x) * V_p - \sinh(\gamma x) * Z_c I_p \tag{4.11}$$

$$I_x = \frac{-\sinh(\gamma x) * V_p + \cosh(\gamma x) * Z_c I_p}{Z_c} \tag{4.12}$$

If the length l of the transmission is used as distance between the i^{th} tapping node and the p^{th} terminal, the voltage V_i at the tapping node and the current along transmission line I_i can be expressed as:

$$V_i = \cosh(\gamma l) * V_p - \sinh(\gamma l) * Z_c I_p \tag{4.13}$$

$$I_i = \frac{-\sinh(\gamma l) * V_s + \cosh(\gamma l) * Z_c I_s}{Z_c} \quad (4.14)$$

4.1.3 Step 1: Bisection search for determining the tapping node connected to the faulty branch

After a fault, PMUs capture the synchrophasor data from all terminals and provided to the fault location algorithm. Gathered data is organized into a data structure class which is created according to the topology of the network. The bisection search process for determining the tapping node(s) connected to the faulty branch is started at the first and the last tapping nodes of the main line, and moved along the forward and backward directions respectively.

The positive-sequence voltage and the positive-sequence current at the first tapping node (connected to terminal A) are estimated by using the measurements at each of terminals connected to the node by using (4.13) and (4.14). If the tapping node is connected to *m* number of terminals, *m* estimations can be found for the positive-sequence voltage. From the available estimations, the maximum and minimum positive-sequence of the voltage estimations are selected based on the magnitudes of the phasor values. Thereafter, the maximum deviation in the estimated voltages computed as the absolute value of the difference between the magnitudes of the maximum and minimum voltage estimates. The tapping node with the highest maximum voltage deviation is consider as the tapping node connected with the faulted branch. The above procedure is repeated for the last tapping node

(connected to terminal B) to calculate the maximum deviation between estimated voltages (using the measurements at terminals connected to last tapping node).

The proposed algorithm then compares two maximum voltage deviations computed at the first and last tapping nodes, and set Fault Direction Tag (FDT) as "forward" if the maximum voltage deviation observed at the first tapping node is lower than that observed at the last tapping node. The Fault Direction Tag (FDT) is set as "backward" if the opposite is true. This is because, a larger deviation between the estimated voltages results at the tapping node closer to the faulted branch.

Then, the proposed algorithm proceeds in the direction indicated by FDT along the main line. This ensures that calculation always move to the next tapping node which is connected to the tapping node with the lower maximum voltage deviation. When proceeding along the main branch, the estimated values of the voltage and branch currents (at the first or the last tapping node) are used. The current at i^{th} tapping node along the branch between i^{th} and $(i+1)^{\text{th}}$ nodes can be evaluated using Kirchhoff's current law for the i^{th} tapping node with estimated currents estimated with (4.20). This process is continued until the algorithm proceeds to the same tapping node from the both directions. This final tapping node is considered as the tapping node connected with the faulty branch.

The algorithm is best illustrated with an example. Consider the five terminals transmission line shown in Figure 4.4. The main transmission line is extends from the first terminal (*T1*) to the last terminal (*T5*). Rest of the terminals (*T2*, *T3* and *T4*) are at the ends of tapping branches connected to the main transmission line through the tapping nodes *N1*, *N2* and *N3* respectively. Suppose that the positive-sequence measurements (voltage and current)

obtained from the PMUs located at each of the terminals *T1, T2, T3, T4, T5* are V^{+}_{T1}, I^{+}_{T1}, V^{+}_{T2}, I^{+}_{T2}, V^{+}_{T3}, I^{+}_{T3}, V^{+}_{T4}, I^{+}_{T4}, V^{+}_{T5}, I^{+}_{T5}. These voltages are indicated on Figure 4.4.

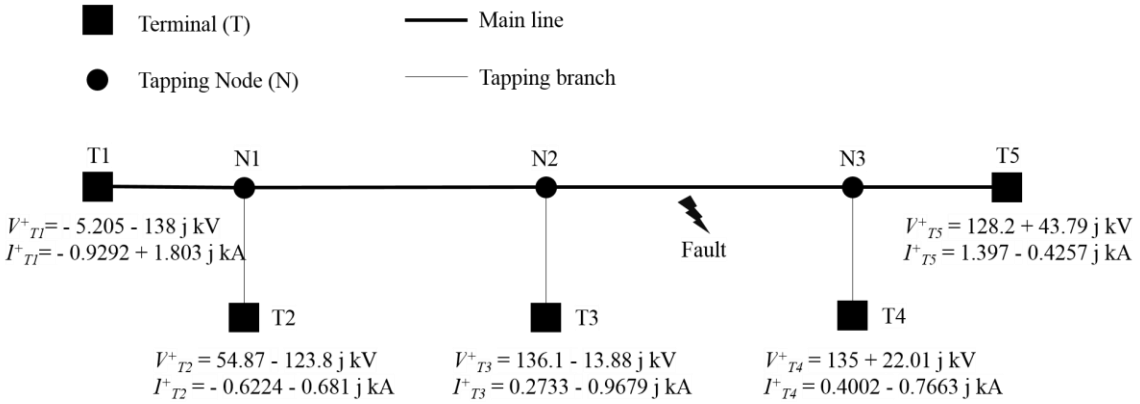


Figure 4.4 Five terminal transmission line

Consider a fault on the main transmission line between tapping nodes *N2* and *N3*. The algorithm starts to estimate the tapping node voltages and branch currents for first and last tapping node *N1* and *N3.* For the tapping node *N1,* two sets of positive-sequence voltage and branch current estimations are computed by using the terminal measurement sets (V^{+}_{T1}, I^{+}_{T1},) and (V^{+}_{T2} , I^{+}_{T2}) and equations (3.19) and (3.20) as ($V^{+}_{N1,T1}$, $I^{+}_{N1,T1}$) and ($V^{+}_{N1,T2}$, $I^{+}_{N1,T2}$). Similarly, for the tapping node *N3* two sets of positive-sequence voltage and branch current estimations are computed by using the terminal measurement sets (V^{+}_{T4}, I^{+}_{T4}) and (V^{+}_{T5} and I^{+}_{T5}) as ($V^{+}_{N3,T4}$, $I^{+}_{N3,T4}$) and ($V^{+}_{N3,T5}$, $I^{+}_{N3,T5}$). The results of these calculations are shown in Figure 4.5.

The maximum voltage deviation at tapping node *N1* (*MVD1*) is calculated using the absolute values of the maximum positive-sequence voltage ($V^{+}_{N1,\,T1}$) and the minimum positive-sequence voltage ($V^{+}_{N1,\,T2}$). Similarly, the maximum voltage deviation at tapping node *N3* (*MVD2*) is calculated by using the absolute values of the estimated maximum positive-

sequence voltage ($V^{+}_{N3,\,T4}$) and the estimated minimum positive-sequence voltage ($V^{+}_{N3,T5}$)

.

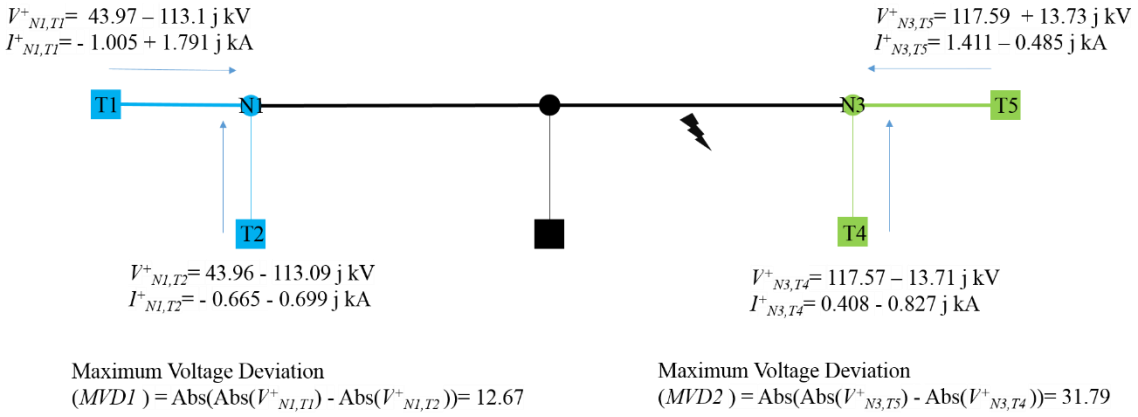


Figure 4.5 Bisection search along the main line (1)

Since a larger maximum voltage deviation is observed at tapping node *N3* (*MVD2* > *MVD1*), as indicated on Figure 4.5, according to the proposed algorithm FDT is set "forward" and computation is moved to tapping node *N2* (which is next to tapping node *N1* in forward direction). This is based on the assumption that branches (*N1-T1*), (*N1-T2*) are without fault. When moving in the forward direction, tapping node *N1* positive-sequence voltage (V^{+}_{N1}) is assumed equal to the positive-sequence voltage estimated by using measurements from terminal *T1* on the transmission line ($V^{+}_{N1,T1}$). Positive-sequence current from the tapping node *N1* towards to the tapping node *N2* (I^{+}_{N1}) is estimated as the summation of all the positive-sequence branch currents estimated in the previous step. For this case, it is the summation of $I^{+}_{N1,T1}$ and $I^{+}_{N1,T2}$ as indicated on Figure 4.6.

When proceeding in forward direction, the branches connected to tapping node *N2* are the line segment (*N1-N2)* along the main line, and tapping branch (*T3-N2)*. Then for tapping node *N2*, two sets of positive-sequence voltage and current estimations can be computed

considering the estimated voltage and branch current set (V^{+}_{N1}, I^{+}_{N1}) and the measurement set (V^{+}_{T3} and I^{+}_{T3}) and using equations (4.13) and (4.14) as ($V^{+}_{N2,\,N1}$, $I^{+}_{N2,N1}$) and ($V^{+}_{N2,T3}$, $I^{+}_{N2,T3}$). The maximum voltage deviation at tapping node *N2* (*MVD3*) is calculated by using the absolute values the maximum positive-sequence voltage ($V^{+}_{N2,\,N1}$) and the minimum positive-sequence voltage ($V^{+}_{N1,\,T3}$).

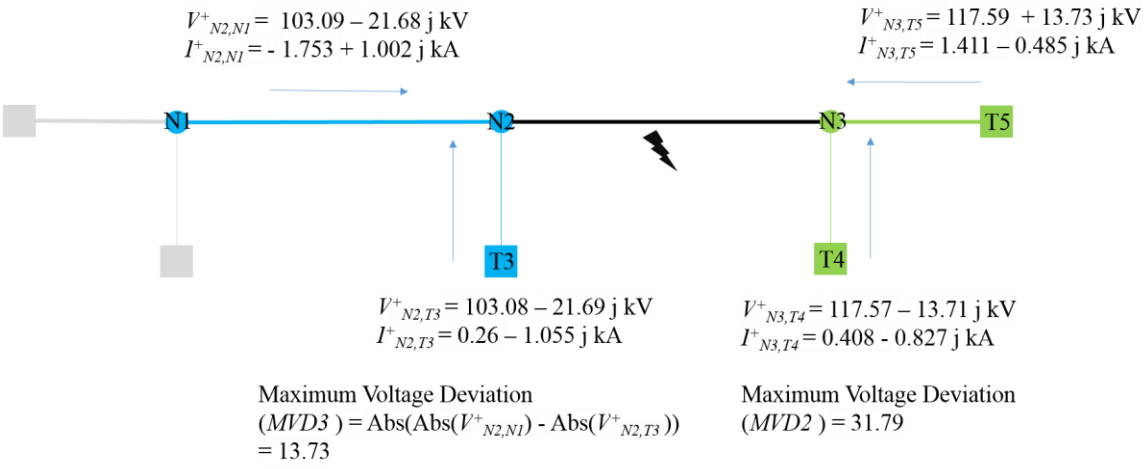


Figure 4.6 Bisection search in the main line (2)

Once more, the maximum voltage deviation observed at tapping node *N3* (*MVD2*) is higher than the maximum voltage deviation observed at tapping node *N2* (*MVD3*) as indicated on Figure 4.6. The algorithm then sets FDT as “forward” and move to tapping node *N3* which is next to the tapping node *N2* in the forward direction as shown in Figure 4.7. This movement indicates that the branches connected to tapping node *N2* in forward direction (*N2*, *N1*) and (*N2*, *T3*) are without fault. Tapping node *N2* positive-sequence voltage (V^{+}_{N2}) is equal to the positive-sequence voltage estimated by using the quantities from tapping node *N1* on the main line ($V^{+}_{N2,\,N1}$). Positive-sequence current from tapping node *N2* towards tapping node *N3* (I^{+}_{N2}) is the summation of $I^{+}_{N2,N1}$ and $I^{+}_{N2,T3}$.

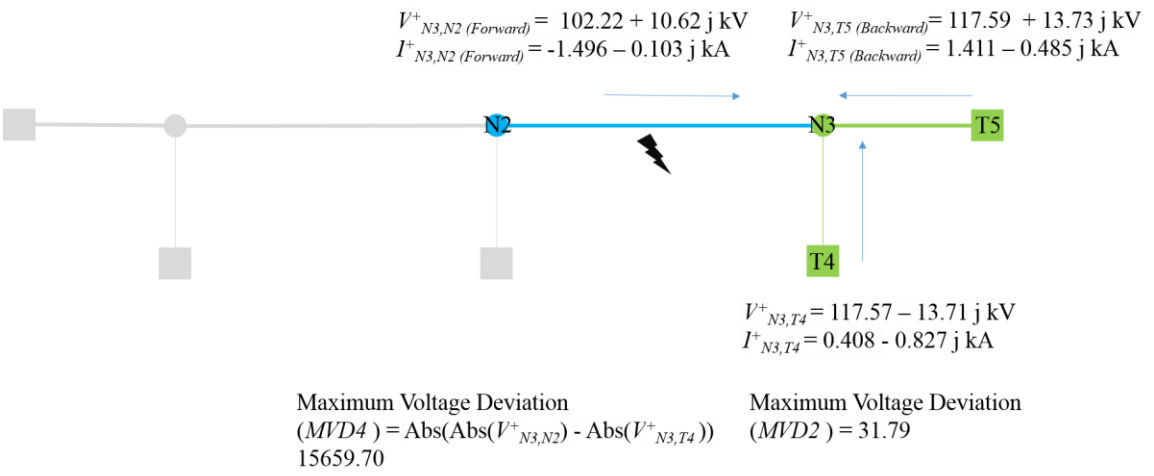


Figure 4.7 Bisection search in the main line (3)

At the end, the computations from both directions are met at the same tapping node *N3* as shown Figure 4.7, indicating that the fault is on one of the branches connected to tapping node *N3*. The positive-sequence voltage and current estimations for tapping node *N3* by using the tapping node measurements (V^{+}_{N2} and I^{+}_{N2}) are ($V^{+}_{N3,\,N2}$ and $I^{+}_{N3,\,N2}$). The maximum voltage deviation in the tapping node *N3* (*MVD4*) in the forward direction is calculated by using the absolute values of the maximum estimated positive-sequence voltage ($V^{+}_{N3,\,N2}$) and the minimum estimated positive-sequence voltage ($V^{+}_{N3,\,T4}$) . The maximum voltage deviation in backward direction, which was computed earlier is equal to *MVD2*.

Since *MVD4* is higher than *MVD2*, FDT is set to "*backward*". This situation indicates that the one of the branches connected to the tapping node *N3* when coming from the forward direction, which is either branch (*N3*, *N2*) or branch (*N3*, *T4*) is faulty. (see Figure 4.8). In addition, it can be deduced that the correct estimation for tapping node *N3* positive-sequence voltage (V^{+}_{N3}) is equal to the positive-sequence voltage estimated along the remaining branch. In this example, it is the estimation made by using the data from the terminal *T5* on the main line ($V^{+}_{N3,\,T5}$).

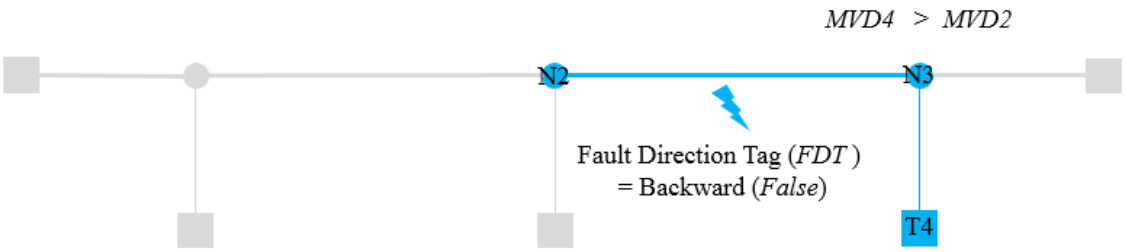


Figure 4.8 Bisection search in the main line (4)

4.1.4 Step 2: Determination of the faulty branch

Once the tapping node connected with the faulty branch is determined, and its correct positive-sequence voltage is estimated, the differences between the correctly estimated voltage and the voltages estimated along the other branches (using the measured terminal voltages and currents or estimated node voltages and currents), referred to as voltage deviations (VDs), can be evaluated. The branch that results in the highest value of VD is considered as the faulty branch.

For the five terminal transmission line example, after the first step of proposed algorithm, tapping node *N3* is identified as the tapping node connected with the faulty branch. Considering each branch connected to the tapping node *N3* in forward direction, two sets of positive-sequence data is computed. By using the estimated voltages and currents at tapping node *N2*, ($V^{+}_{N2}$and I^{+}_{N2}), voltage at *N3* is estimated as ($V^{+}_{N3,N2}$, $I^{+}_{N3,N2}$); By using the measurements at terminal *T4*, (V^{+}_{T4}, I^{+}_{T4}) voltage at *N3* is estimated as ($V^{+}_{N3,T4}$, $I^{+}_{N3,T4}$). The best (correct) estimation of tapping node *N3* positive-sequence voltage (V^{+}_{N3}) is equal to the positive-sequence voltage estimated by using the data from terminal *T5* ($V^{+}_{N3,\,T5}$) as indicated in Figure 4.9.

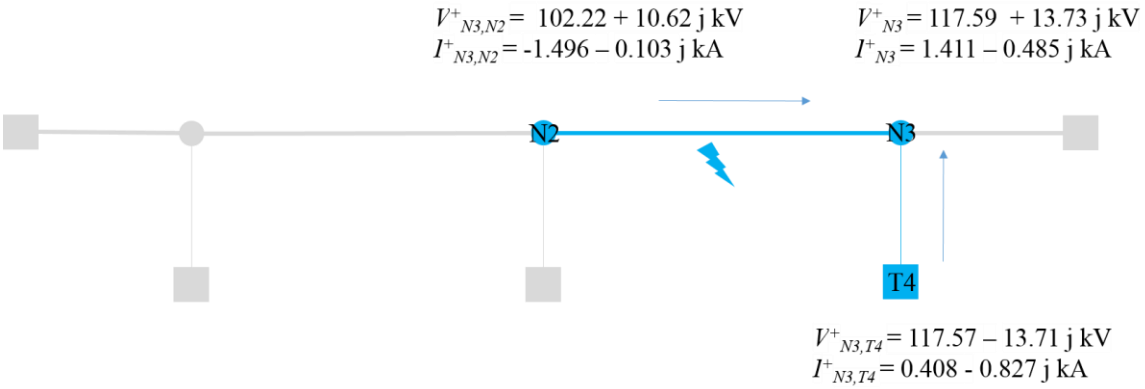


Figure 4.9 Faulty branch search (1)

The deviations of voltages estimated for tapping node *N3* ($V^{+}_{N3,N2}$ and $V^{+}_{N3,T4}$) with reference to its correct estimate ($V^{+}_{N3,\,T5}$), say *VD1* and *VD2*, are calculated next. The results of this calculation are indicated on Figure 4.10. Since, the estimations along the branch between the tapping nodes *N2* and *N3* result in the highest voltage deviation, this branch is declared by the proposed algorithm as the faulty branch as illustrated in Figure 4.11.

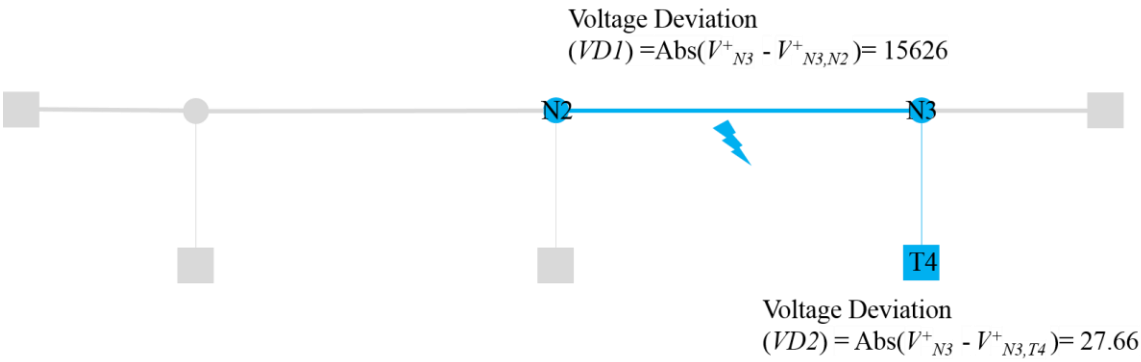


Figure 4.10 Faulty branch search (2)

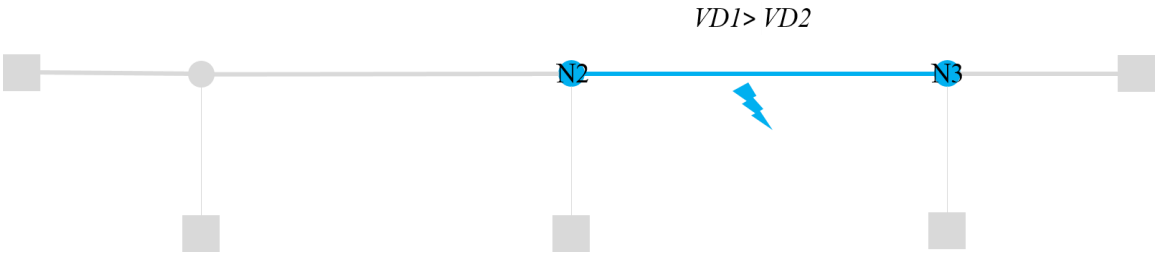


Figure 4.11 Faulty branch search (3)

4.1.5 Step 3: Calculation of the fault location

Once the faulted segment is determined and the measured or estimated voltages and currents at two ends of the segment are available, a two terminal fault location algorithm can be applied. One of the earliest two-terminal fault algorithms that independent of source impedances and fault resistance is presented in [45]. Assume that the fault happened in *x* meters away from the sending end. The fault splices the transmission line into two sections: each section is a line which connect with either the sending or receiving end of the faulted line segment. Thus, the voltage at the fault point can be estimated from two ways. By using the sending end data (V_S, I_S) to calculate the voltage at the fault location $V_{F,Sending}$ is obtained as:

$$V_{F,Sending} = \cosh(\gamma x) * V_S - \sinh(\gamma x) * Z_c I_S \tag{4.15}$$

$$V_{F,Sending} = \frac{V_S - I_S Z_c}{2} e^{\gamma x} + \frac{V_S + I_S Z_c}{2} e^{-\gamma x} \tag{4.16}$$

$$V_{F,Sending} = \frac{V_S}{2} e^{\gamma x} + \frac{V_S}{2} e^{-\gamma x} - \frac{I_S Z_c}{2} e^{\gamma x} + \frac{I_S Z_c}{2} e^{-\gamma x} \tag{4.17}$$

Using the receiving end data (V_R, I_R) to calculate the voltage at the fault location, $V_{F,Receiving}$ is obtained.

$$V_{F,Receiving} = \cosh(\gamma l - \gamma x) * V_R - \sinh(\gamma l - \gamma x) * Z_c I_R \tag{4.18}$$

$$V_{F,Receiving} = \frac{V_R - I_R Z_c}{2} e^{\gamma l - \gamma x} + \frac{V_R + I_R Z_c}{2} e^{-\gamma l + \gamma x} \tag{4.19}$$

$$\begin{aligned} V_{F,Receiving} = {} & \frac{V_R}{2} e^{\gamma l - \gamma x} + \frac{V_R}{2} e^{-\gamma l + \gamma x} - \frac{I_R Z_c}{2} e^{\gamma l - \gamma x} \\ & + \frac{I_R Z_c}{2} e^{-\gamma l + \gamma x} \end{aligned} \tag{4.20}$$

Since both quantities point the same voltage,

$$V_{F,Sending} = V_{F,Receiving} \tag{4.21}$$

(4.21) can be rewritten as

$$\begin{aligned} & \frac{V_S}{2} e^{\gamma x} + \frac{V_S}{2} e^{-\gamma x} - \frac{I_S Z_c}{2} e^{\gamma x} + \frac{I_S Z_c}{2} e^{-\gamma x} \\ & = \frac{V_R}{2} e^{\gamma l - \gamma x} + \frac{V_R}{2} e^{-\gamma l + \gamma x} - \frac{I_R Z_c}{2} e^{\gamma l - \gamma x} \\ & + \frac{I_R Z_c}{2} e^{-\gamma l + \gamma x} \end{aligned} \tag{4.22}$$

By multiplying both the left and right hand sides of (4.22) by the term$(2 * e^{\gamma x})$, it can rewritten as

$$\begin{aligned} & V_S e^{\gamma 2x} + V_S - I_S Z_c e^{\gamma 2x} + I_S Z_c \\ & = V_R e^{\gamma l} + V_R e^{-\gamma l} e^{\gamma 2x} - I_R Z_c e^{\gamma l} + I_R Z_c e^{-\gamma l} e^{\gamma 2x} \end{aligned} \tag{4.23}$$

Manipulations lead to,

$$(V_S - I_S Z_c - V_R e^{-\gamma l} - I_R Z_c e^{-\gamma l}) * e^{\gamma 2x} = -V_S - I_S Z_c + V_R e^{\gamma l} - I_R Z_c e^{\gamma l} \quad (4.24)$$

$$e^{\gamma 2x} = \frac{-V_S - I_S Z_c + V_R e^{\gamma l} - I_R Z_c e^{\gamma l}}{V_S - I_S Z_c - V_R e^{-\gamma l} - I_R Z_c e^{-\gamma l}} \quad (4.25)$$

$$x = \frac{1}{2} * \frac{1}{\gamma} * ln(\frac{-V_S - I_S Z_c + V_R e^{\gamma l} - I_R Z_c e^{\gamma l}}{V_S - I_S Z_c - V_R e^{-\gamma l} - I_R Z_c e^{-\gamma l}}) \quad (4.26)$$

The unknown terms *A* and *B* can be written as following

$$A + B = -V_S - I_S Z_c + V_R e^{\gamma l} - I_R Z_c e^{\gamma l} \quad (4.27)$$

$$A - B = V_S - I_S Z_c - V_R e^{-\gamma l} - I_R Z_c e^{-\gamma l} \quad (4.28)$$

Then, the unknown terms can be calculated as following

$$2A = -V_S - I_S Z_c + V_R e^{\gamma l} - I_R Z_c e^{\gamma l} + V_S - I_S Z_c - V_R e^{-\gamma l} - I_R Z_c e^{-\gamma l} \quad (4.29)$$

$$2A = -2 * I_S Z_c + V_R (e^{\gamma l} - e^{-\gamma l}) - Z_c I_R (e^{-\gamma l} + e^{\gamma l}) \quad (4.30)$$

$$A = -I_S Z_c + V_R \frac{(e^{\gamma l} - e^{-\gamma l})}{2} - Z_c I_R \frac{(e^{-\gamma l} + e^{\gamma l})}{2} \quad (4.31)$$

$$A = -I_S Z_c + V_R \sinh(e^{\gamma l}) - Z_c I_R \cosh(e^{\gamma l}) \quad (4.32)$$

$$B = -V_S + V_R\cosh(e^{\gamma l}) - Z_c I_R \sinh(e^{\gamma l}) \quad (4.33)$$

Then the fault distance *x* can be calculated as

$$x = \frac{1}{2} * \frac{1}{\gamma} * ln(\frac{A+B}{A-B}) \quad (4.34)$$

$$x = \frac{1}{2} * \frac{1}{\gamma} * ln(\frac{1+\frac{B}{A}}{1-\frac{B}{A}}) \quad (4.35)$$

$$x = \frac{1}{2} * \frac{1}{\gamma} * atanh(\frac{B}{A}) \quad (4.36)$$

The normalized fault distance $\hat{x}$ is calculated as

$$\hat{x} = \frac{1}{2} * \frac{1}{\gamma l} * \text{atanh}\left(\frac{B}{A}\right) \quad (4.37)$$

For five terminal transmission line examples, positive-sequence voltage and current in both ends of branch are given as (V^+_{N2}, I^+_{N2}) and (V^+_{N3}, I^+_{N3}) in Figure 4.12. The normalized fault location is calculated by using (4.32), (4.33), and (4.36) and the result is 49.72% of the length of the section between the tapping nodes *N2* and *N3*. It is very close to the actual fault location (50% of the section length) in this example.

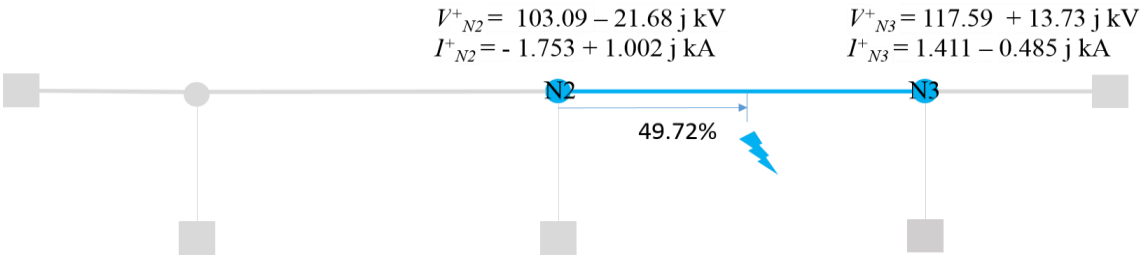


Figure 4.12 Determine the normalized fault distance

Proposed three step bisection search algorithm for fault location in multi-terminal multi-section transmission lines is summarised in the flow chart shown in Figure 4.13.

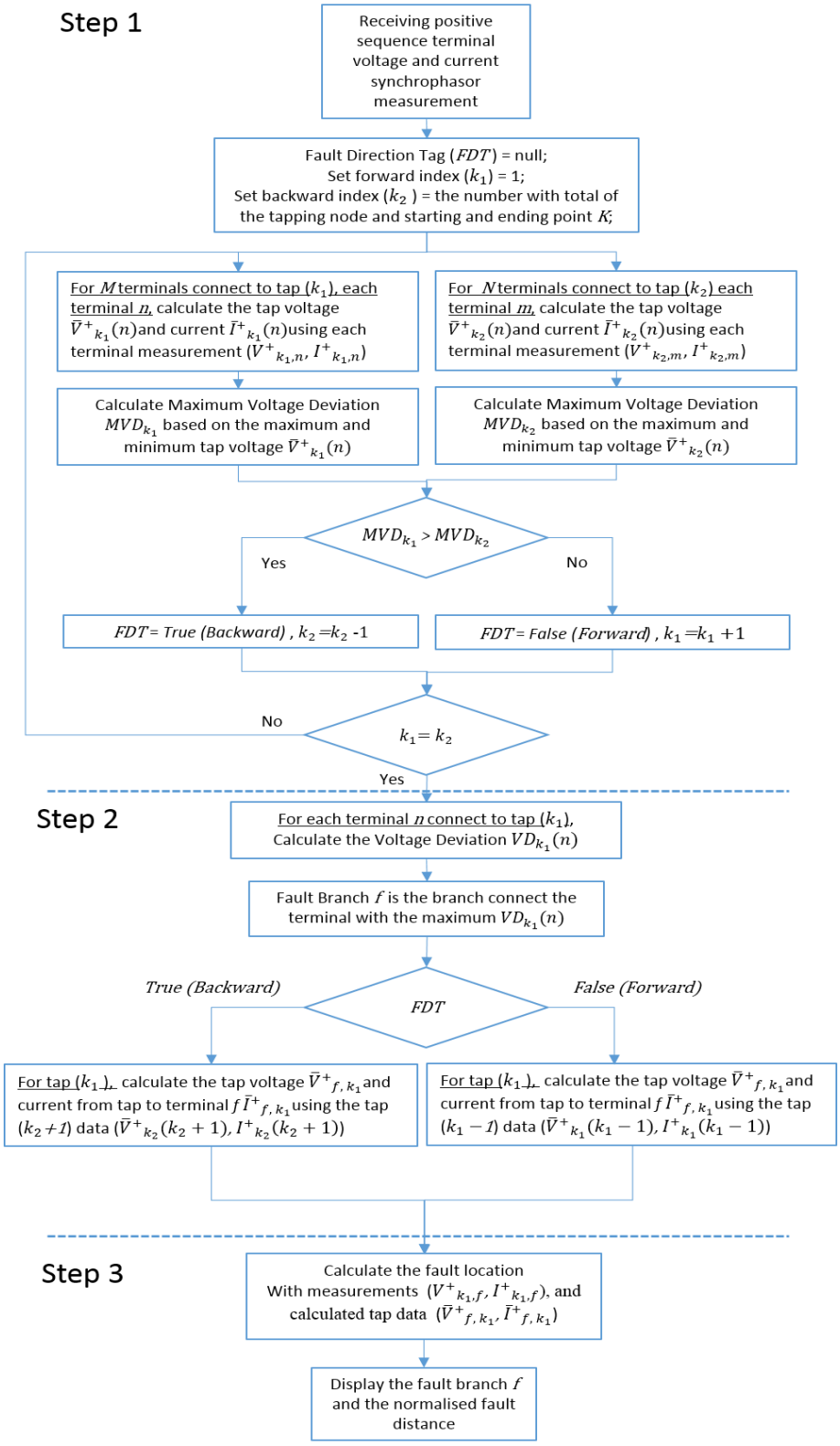


Figure 4.13 Flow chart of the fault location technique for multi-terminal line

4.1.6 Fault location in non-homogeneous transmission line

A transmission line is non-homogeneous when it consists of two or more sub-section with different tower and conductor configurations. The result is non-identical line parameters (resistance, inductance, conductance and capacitance) along the line. Non-homogeneous transmission lines are not uncommon in practical transmission systems. In order to locate the faults in a non-homogeneous transmission line, Virtual tapping Node (VN) is introduced into the proposed algorithm. VN is inherited from a normal tapping node or from a terminal, which allows estimation of the voltage and currents at the VN. A virtual node has only two branches, each branch represent one of the sections in the non-homogeneous transmission line. Both branches are either sections of a segment on the main line or a tapping branch.

When a fault happens, the proposed algorithm starts with the bisection search in both forward and backward direction of the transmission line. When the algorithm hits a virtual node, it estimates the virtual node measurements and move to the next node without calculating and comparing the maximum positive-sequence voltage deviation. In general, the first step of the proposed algorithm treats a non- homogeneous transmission line as two separate lines during the taping node data estimation, and as one line during the bisection searching process.

If non-homogeneous line section or tapping branch is identified as the faulty branch, in Step 3 of the proposed algorithm, the two ended fault location equations (4.32), (4.33), and (4.36) are applied to every section of the homogeneous transmission line. The section with

a valid result for the normalised fault distance (within the range [0-1]) is identified as the faulty section.

The following example illustrates the above procedure for a five terminals transmission line with two non-homogeneous branches between the tapping nodes *N1*, *N2* and *N1*, *T2*. This configuration results in two virtual tapping nodes *VN1* and *VN2* as shown in Figure 4.14.

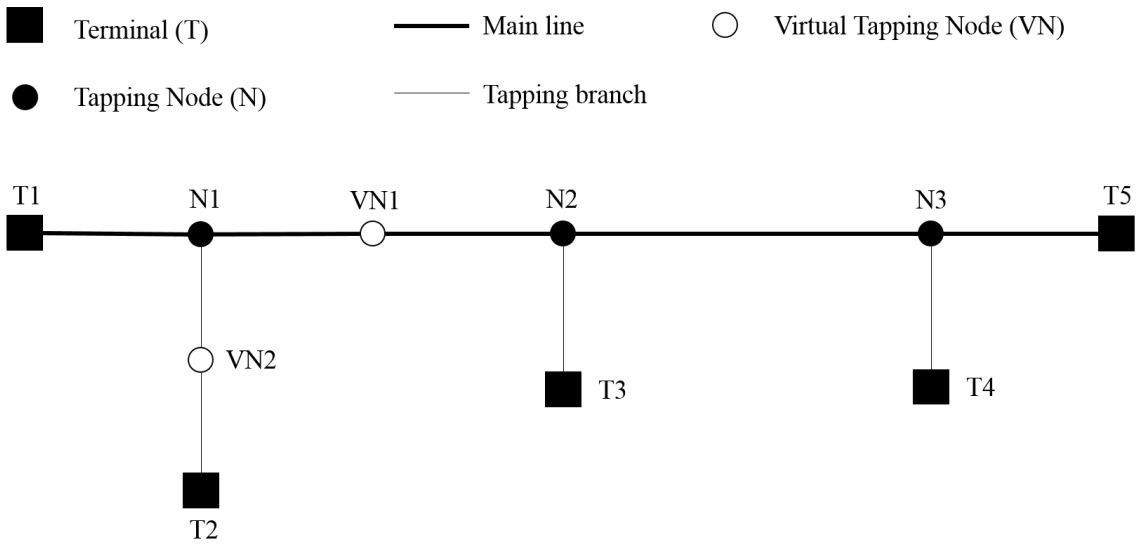


Figure 4.14 Five terminal transmission line with non-homogeneous section

Virtual node voltages and currents sets ($V^{+}_{N1,\,VN1}, I^{+}_{N1,\,VN1}$), ($V^{+}_{N2,\,VN1}, I^{+}_{N2,\,VN1}$) and ($V^{+}_{T2,\,VN2}, I^{+}_{T2,\,VN1}$) are estimated by using the data from the nodes adjacent to the respective VNs as indicated on Figure 4.15.

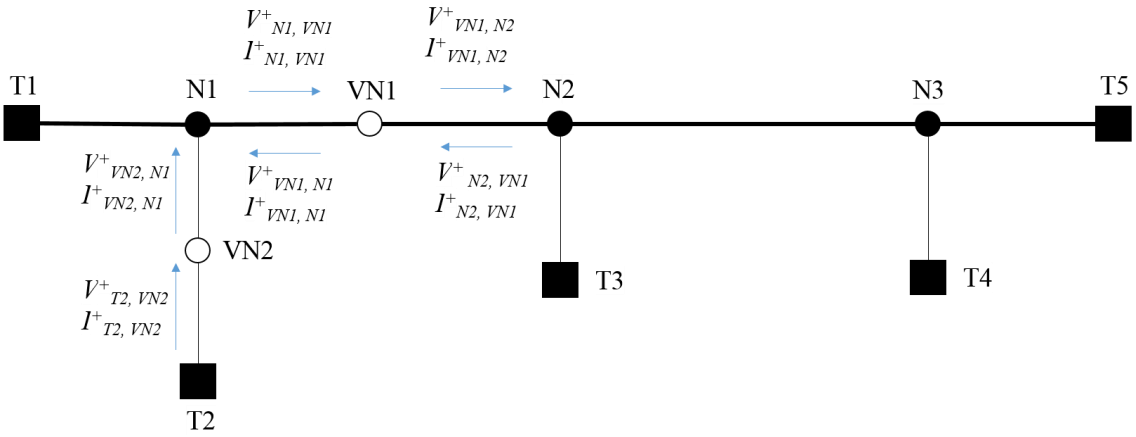


Figure 4.15 Fault location detection for non-homogeneous transmission line (1)

If the fault is in the section between the tapping node *N1* and the virtual tapping node *VN1*, after the bisection search, non-homogeneous branch between *N1* and *N2* will be identified as the faulty branch as shown in Figure 4.16.

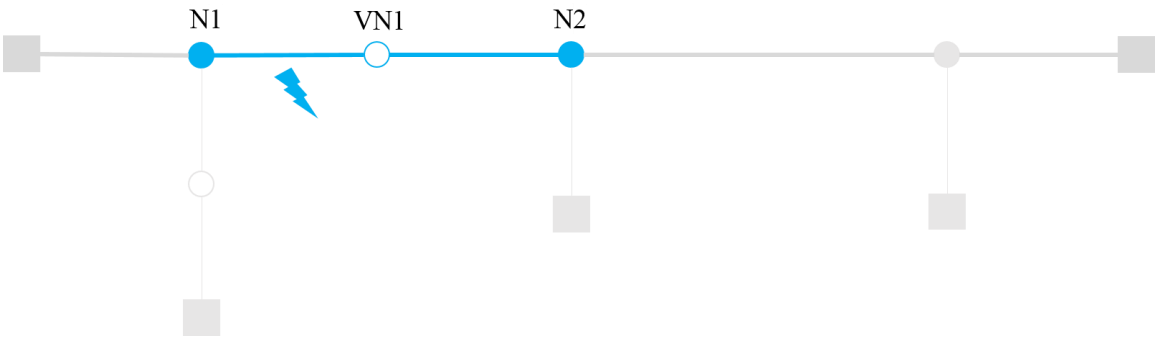


Figure 4.16 Fault location for non-homogeneous transmission line (2)

The algorithm then applies two ended fault location equations for every section of the non-homogeneous transmission line. Section between tapping node *N1* and virtual tapping node *VN1* returns a valid result which is in the range between zero and one as shown in Figure 4.17. When the fault is not actually on a given section, the calculation should return an invalid result outside of the range [0-1]. One notable feature of this algorithm is that it does not depend on any thresholds.

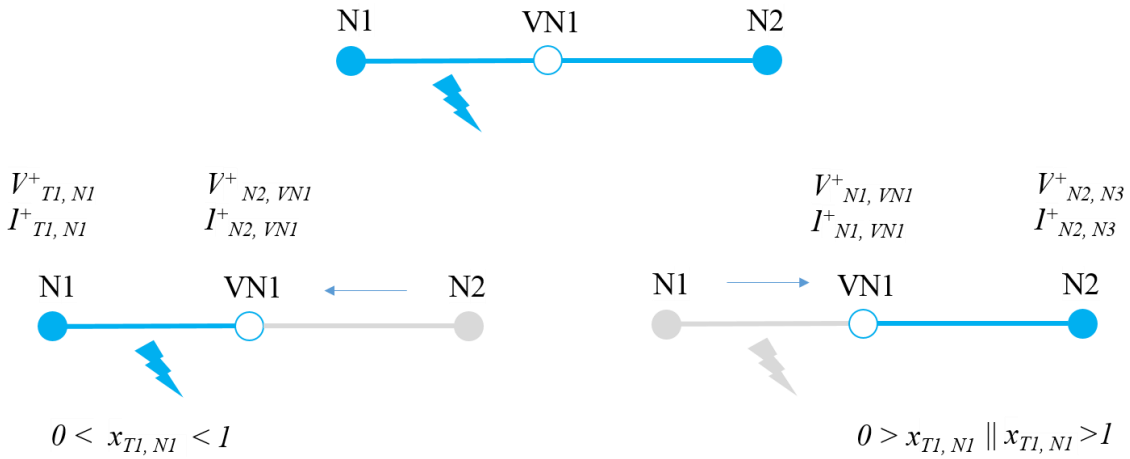


Figure 4.17 Fault location detection for non-homogeneous transmission line (3)

4.2 Experimental setup

To test the proposed fault location algorithm, a 230kV, 60Hz, 5-terminal line is simulated in the RTDS real time digital simulator [11]. The power system shown in Figure 4.18 consists of four equivalent sources, seven transmission line segments and one loads. In the simulation case, two GTNET-PMU8 modules [40] are included to provide real-time PMU measurements made at all terminals. The relevant parameters used in the simulation case are provided in Table 4.1, Table 4.2, and Table 4.3. In addition, a 66kV, 60Hz, 6-terminal distribution feeder is also built and simulated to test proposed fault location algorithm in disturbution level feeder with multiple taps. This power system is shown in Figure 4.19. The related parameters are depicted in Table 4.4, Table 4.5, and Table 4.6.

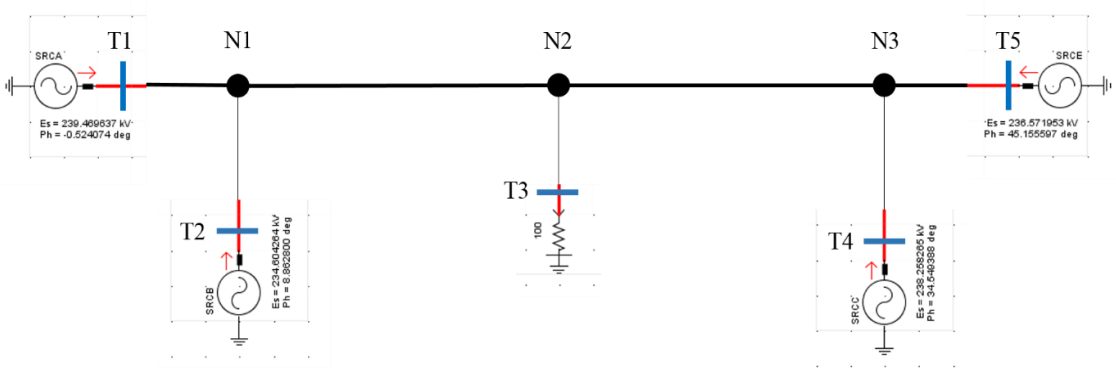


Figure 4.18 230kV 60Hz 5-terminal system

Table 4.1 Parameters for the sources in 230kV 60Hz 5-terminal system

Generator	**Terminal number**	**T1**	**T2**	**T4**	**T5**
AC source initial values	**Initial source mag (L-L, RMS) (kV)**	**239.46**	**234.6**	**238.25**	**236.57**
	Initial frequency (Hz)	**60**	**60**	**60**	**60**
	Initial phase (Degree)	**-0.523**	**8.86**	**34.53**	**45.14**
AC source initial power output	**Initial real power (MW)**	**500**	**100**	**100**	**500**
	Initial reactive power (MVAR)	**100**	**50**	**50**	**100**
Positive sequence impedance	**Base frequency (Hz)**	**60**	**60**	**60**	**60**
	Positive sequence impedance (Ohms)	**2**	**2**	**3**	**3**
	Positive sequence impedance phase angle (Degree)	**86**	**86**	**86**	**86**

Table 4.2 Parameters for the loads in 230kV 60Hz 5-terminal system

Load	**550 MVA**
Impedance	**100 Ohms**

Table 4.3 Parameters for the transmission lines in in 230kV 60Hz 5-terminal system

Location		**Positive sequence**		**Zero sequence**	
Section	**Length (km)**	**Impedance (Ohms)**	**Shunt Capacitive Reactance (Ohms)**	**impedance (Ohms)**	**Shunt Capacitive Reactance (Ohms)**
T1-N1	**100**	**0.66 + j26.66**	**1658.37**	**34.64 + j111.00**	**2843.74**
T4-N3	**80**	**0.52 + j21.27**	**2078.21**	**27.65 + j88.58**	**3563.67**
T2-N1	**60**	**0.39 + j15.89**	**2782.69**	**20.65 + j66.15**	**4771.7**
T5-N3	**80**	**0.52 + j21.28**	**2078.21**	**27.65 + j88.59**	**3563.67**
N1-N2	**200**	**1.32 + j53.58**	**825.02**	**69.64 + j223.13**	**1414.72**
T3-N2	**120**	**0.79 + j32.04**	**1379.65**	**41.64 + j133.43**	**2365.8**
N2-N3	**80**	**0.52 + j21.27**	**2078.21**	**27.65 + j88.58**	**3563.67**

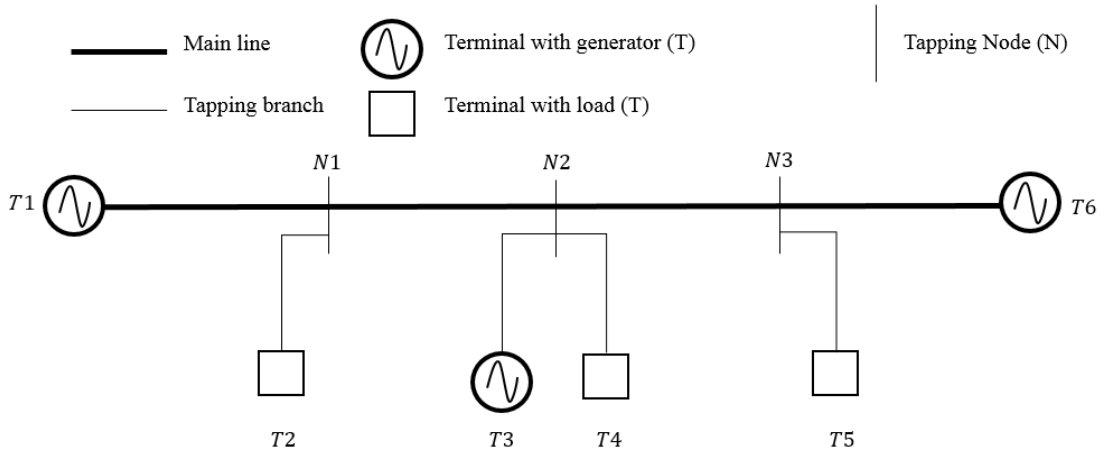


Figure 4.19 66kV 60Hz 6 terminals distribution feeder

Table 4.4 Parameters for the sources in 66kV 60Hz 6-terminal distribution feeder

Generator	Terminal index	T1	T3	T6
AC source initial values	Initial source mag (L-L, RMS) (kV)	68.66	68.39	67.34
	Initial frequency (Hz)	60.00	60.00	60.00
	Initial phase (Degree)	0.14	-5.58	2.78
AC source initial power output	Load flow result: Real power (MW)	13.93	2.00	15.00
	Load flow result: Reactive power (MVAR)	1.065	23.52	-2.62
Positive sequence impedance	Base frequency (Hz)	60.00	60.00	60.00
	Positive sequence impedance (Ohms)	0.85	1.20	1.00
	Positive sequence impedance phase angle (Degree)	86.00	80.00	75.00

Table 4.5 Parameters for the transmission lines in in 66kV 60Hz 6-terminal distribution feeder

Location		Positive sequence		Zero sequence	
Section	Length (km)	Impedance (Ohms)	Shunt Capacitive Reactance (Ohms)	Impedance (Ohms)	Shunt Capacitive Reactance (Ohms)
T1-N1	40	7.80 + j23.06	8582.11	12.18 + j73.55	15216.42
T2-N1	5	0.97 + j2.88	68656.85	1.52 + j9.19	121731.33
TN1-N2	20	3.86 + j11.32	17011.07	6.52 + j36.91	31315.98
T3-N2	5	0.97 + j2.88	68656.85	1.52 + j9.19	121731.33
T4-N2	10	1.95 + j5.76	34328.42	3.04 + j18.39	60865.66
T5-N3	5	0.96 + j2.82	70058.01	1.49 + j9.01	124215.64
TN2-N3	10	1.95 + j5.76	34328.42	3.04 + j18.39	60865.66
T6-N3	60	11.70 + j34.59	5721.40	18.27 + j110.32	10144.28

Table 4.6 Parameters for the loads in 66kV 60Hz 6-terminal distribution feeder

Load terminal index	**T2**	**T4**	**T5**
Connection type	**Y**	**Y**	**Y**
Shunt resistance per phase (Ohms)	**874.80**	**218.70**	**874.80**
Shunt inductance per phase (H)	**3.093**	**0.93**	**3.093**

In addition, a synchrophasor network shown in Figure 4.20 is assembled for the test setup. The analog currents and voltages obtained from the simulation are input to GTNET PMU models in the RTDS simulator. All PMUs are configured as P-class and they report synchrophasors encoded according to IEEE C37.118.2 standard via network sockets in GTNET card. The laboratory internal communication network transports synchrophasor data packages to a computer with SEL PDC software (SEL-5073) via a TCP/IP connection. Furthermore, SEL PDC software creates another TCP/IP connection to PhasorEye monitoring software to send its output synchrophasor data stream. In order to provide offline analysis, PhasorEye monitoring software records synchrophasor data package by writing the original hexadecimal digits into a text file. An example is shown in Figure 4.21. PhasorEye data player is used to decipher and replay the saved synchrophasor text file as shown in Figure 4.22. PhasorEye data player is an interface to provide the decoded synchrophasor data to any analysis tool. Visual studio software is used to create an application to analysis the recoded data by using proposed synchrophasor measurement-based fault location algorithm (Figure 4.23).

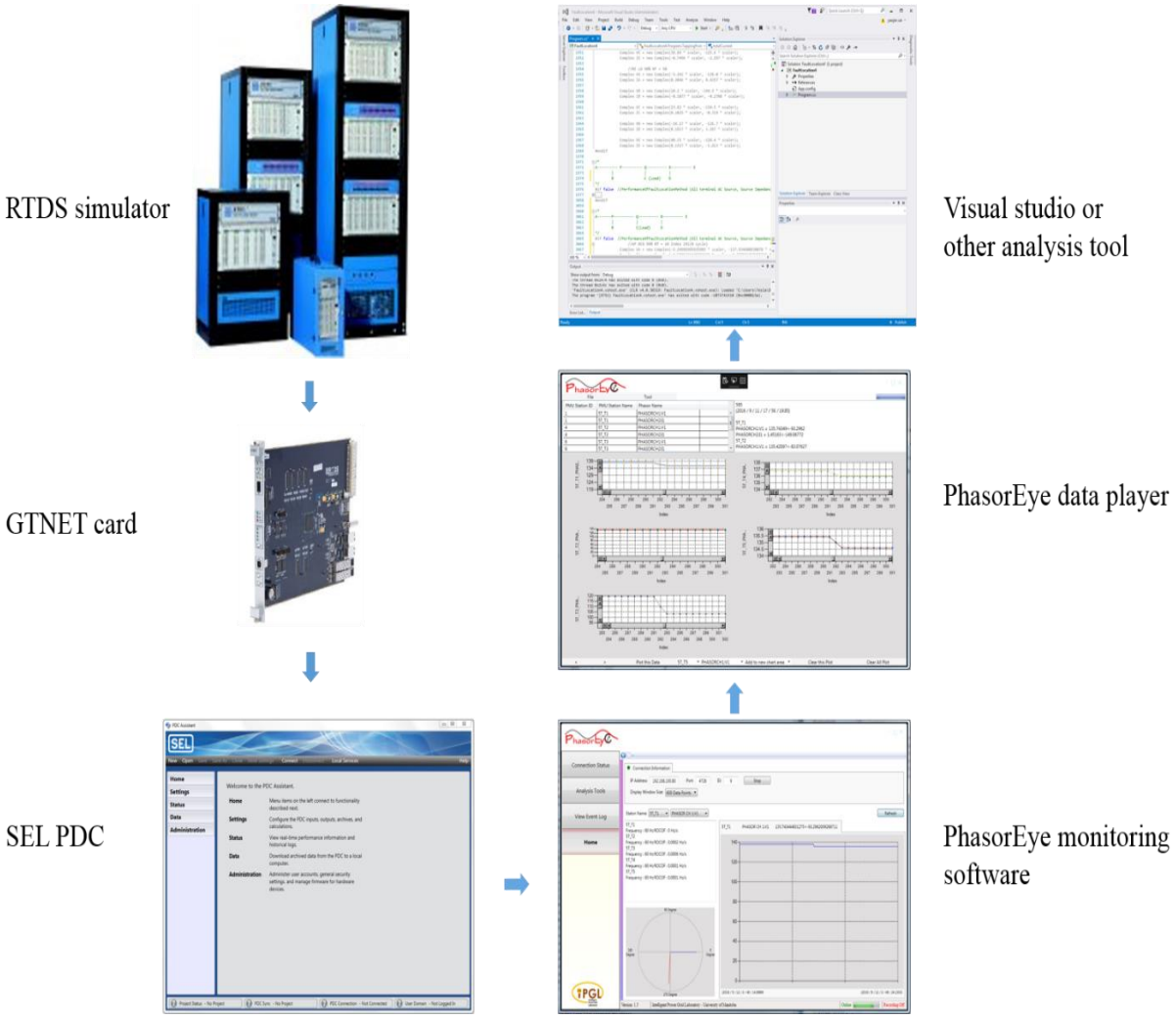
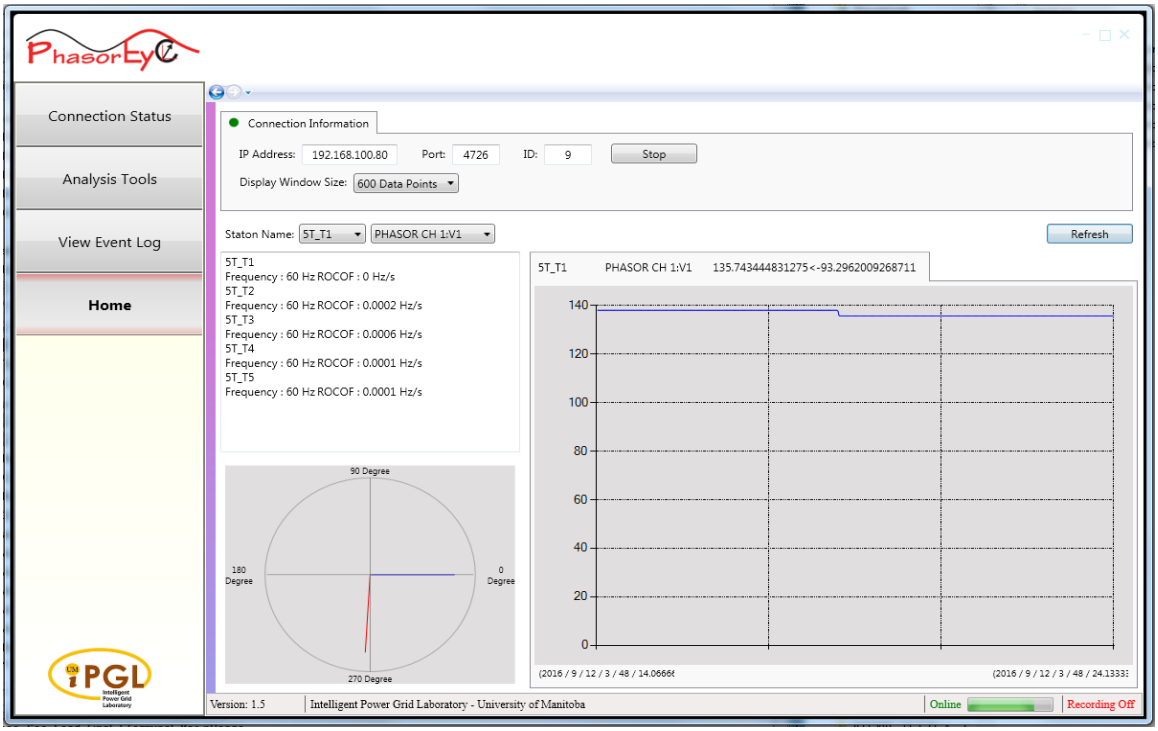


Figure 4.20 Lab testing hardware and software set up



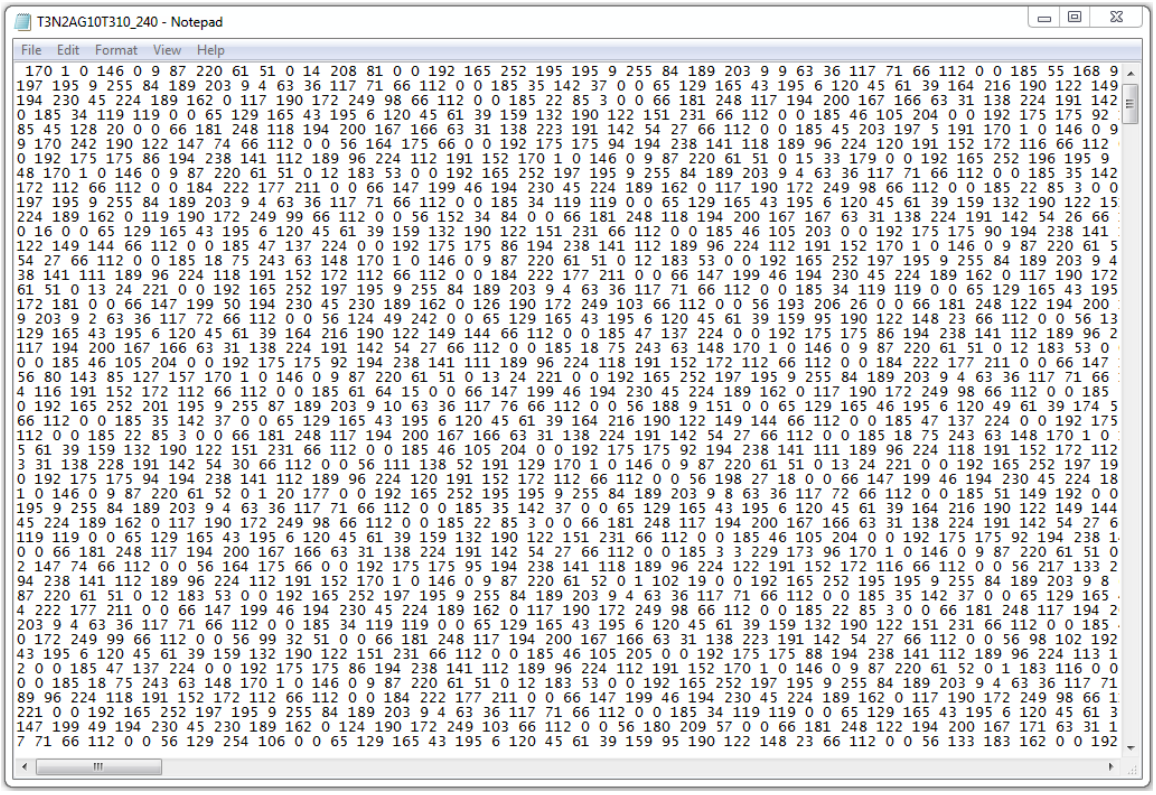


Figure 4.21 Snapshot of PhasorEye program (top) and its recorded data (bottom)

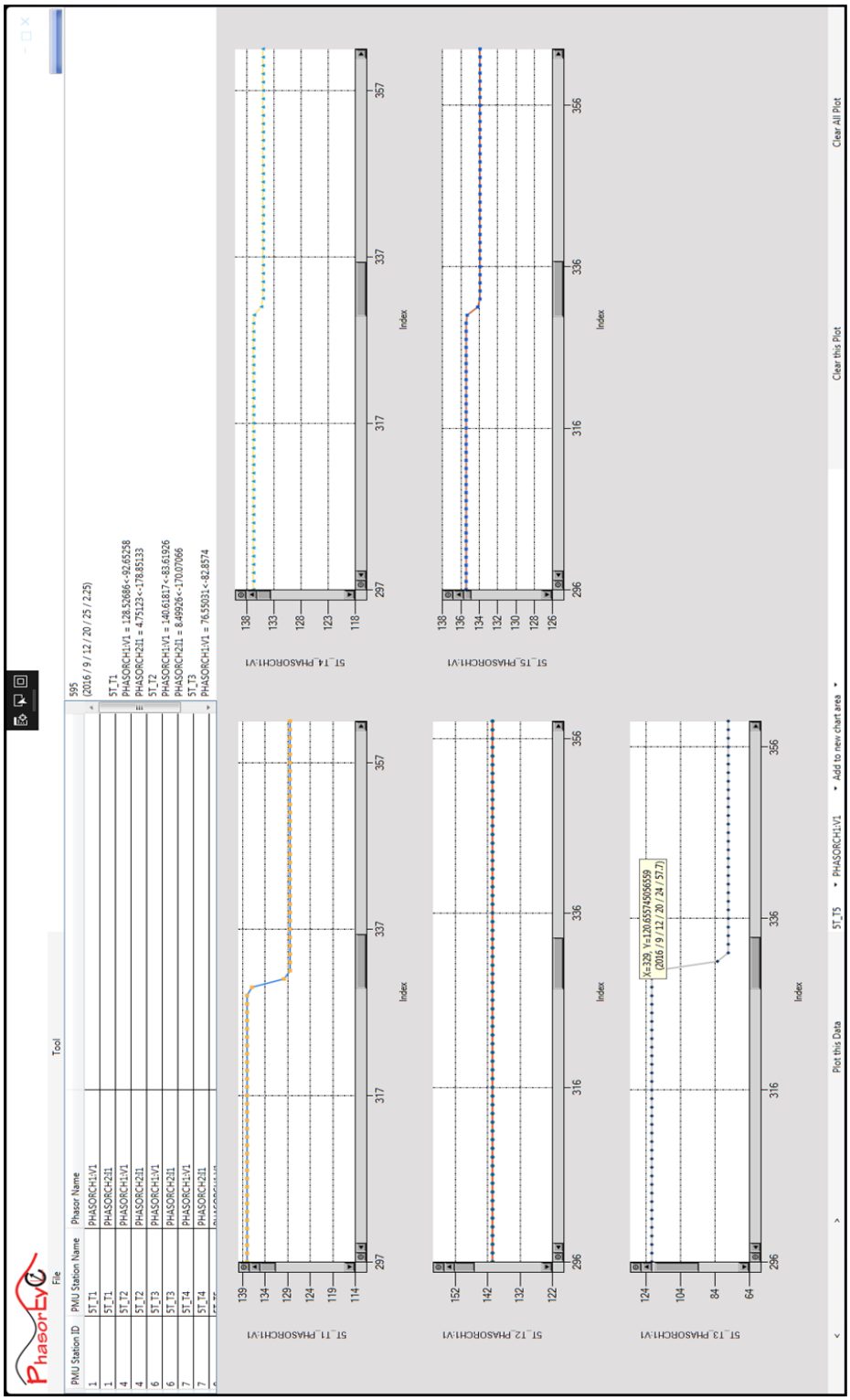


Figure 4.22 Snapshot of PhasorEye data player program with 60 frames per second reporting rate

Figure 4.23 Feeding data from data player (left) into synchrophasor measurement-based fault location algorithm (right)

4.3 Real time simulation results

Within the period from the occurrence of fault and the clearance of fault by protection, several sets of synchrophasor data can be captured. However, data captured immediately after the fault are not accurate as the voltage and current waveforms contain transients. On the other hand, data obtained after the clearance of fault are not useful for fault location. Thus, in order to determine the appropriate time delay after the fault for applying the proposed algorithm, several tests are conducted.

Different types of faults with various values of fault resistances are created at many locations in each section of the 230 kV 60Hz 5-terminal line system (Figure 4.18). These tests are conducted on line sections representing three typical connections. Branch *N1* to *N2* represents a connection between two tapping nodes. Branch *T3* to *N2* represents a connection between a load terminal and a tapping node. Branch *T1* to *N1* represents a connection between an equivalent source terminal and a tapping node. Different types of faults are tested: they are one phase to ground faults, three phase to ground faults, phase to phase faults and three phase faults. Moreover, five different locations are selected for each section: they are 1 percent, 10 percent, 50 percent, 90 percent or 99 percent of the branch length. Fault resistance for the test is selected as 10 ohms. Reporting rate for the synchrophasor network is selected as 240 frames per second (time delay between each data frame is 4.1667 microseconds) which is the highest reporting rate that hardware and software in the laboratory setup supports (note that maximum standard reporting rate is 60 fps as per IEEE C37.118.1-2011. The first 20 frames after the fault are fed into the algorithm. This measurement period roughly covers the 5 power frequency cycles after the fault. In total,

1200 tests are conducted, and the results are presented in 12 figures ranging from Figure 4.24 to Figure 4.35. Each figure contributes one type of faults in one of the three sections. List of figures is presented in Table 4.7.

Table 4.7 Performance tests for selecting the appropriate time delay

Location / Type	**Tapping node to tapping node (*N1* to *N2*)**	**Load to tapping node (*T3* to *N2*)**	**Source to tapping node (*T1* to *N1*)**
AG	**Figure 4.24**	**Figure 4.25**	**Figure 4.26**
ABCG	**Figure 4.27**	**Figure 4.28**	**Figure 4.29**
AB	**Figure 4.30**	**Figure 4.31**	**Figure 4.32**
ABC	**Figure 4.33**	**Figure 4.34**	**Figure 4.35**

X-axis of each figure represents time stamps from the first frame to the twentieth frame after the fault. Y-axis of the figure represents the location from 1 percent to 99 percent of faulted line segment length. Z-axis represents error from 0 percent to 100 percent. The error for the normalized fault distance is calculated as

$$\text{Error } \% = \frac{|\text{Actual location} - \text{Estimated location}|}{\text{Total seciton length}} * 100 \qquad (3.44)$$

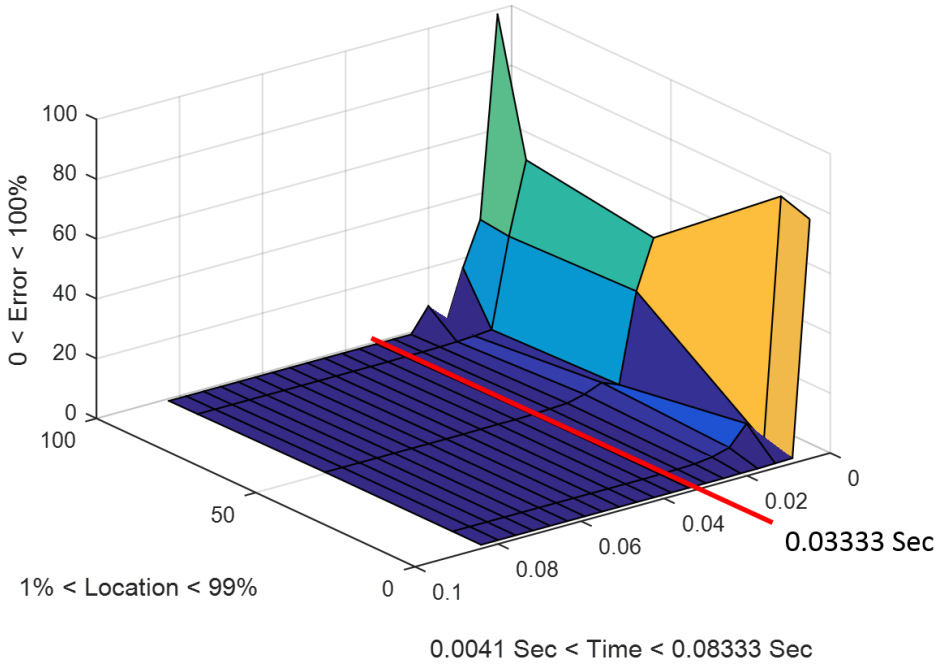


Figure 4.24 Phase A to ground faults in section between tapping nodes N1 and N2

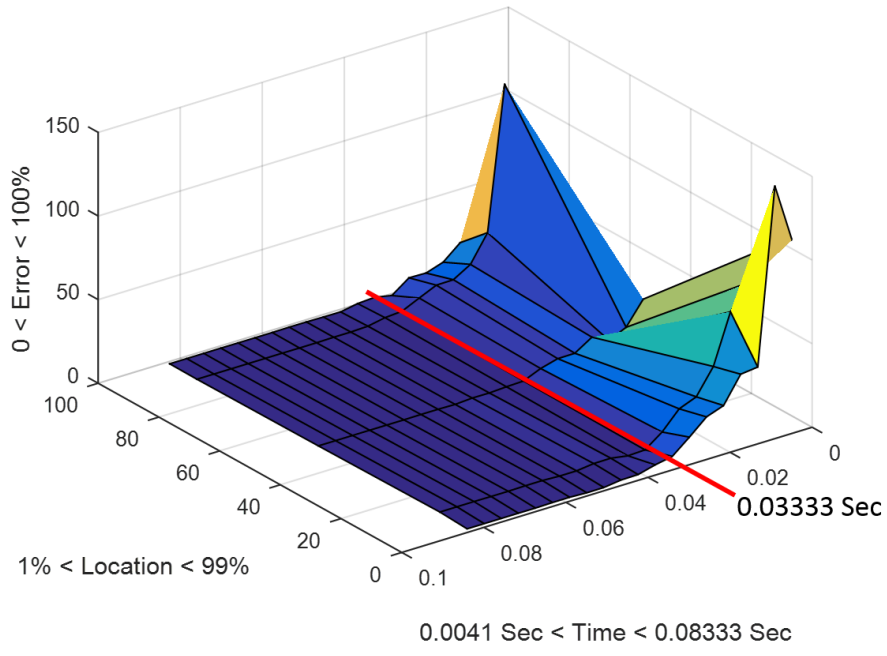


Figure 4.25 Phase A to ground faults in section from terminal T3 to tapping node N2

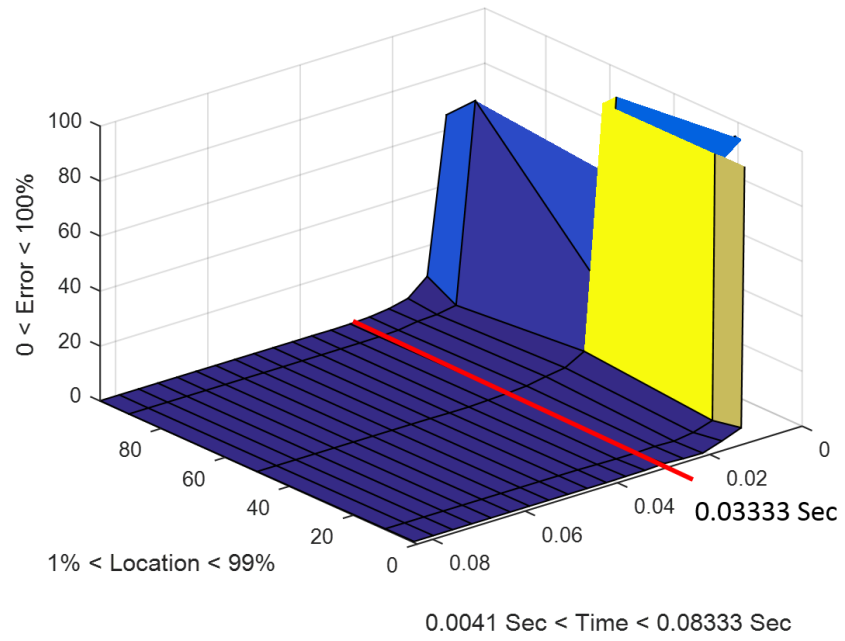


Figure 4.26 Phase A to ground faults in section from terminal T1 to tapping node N1

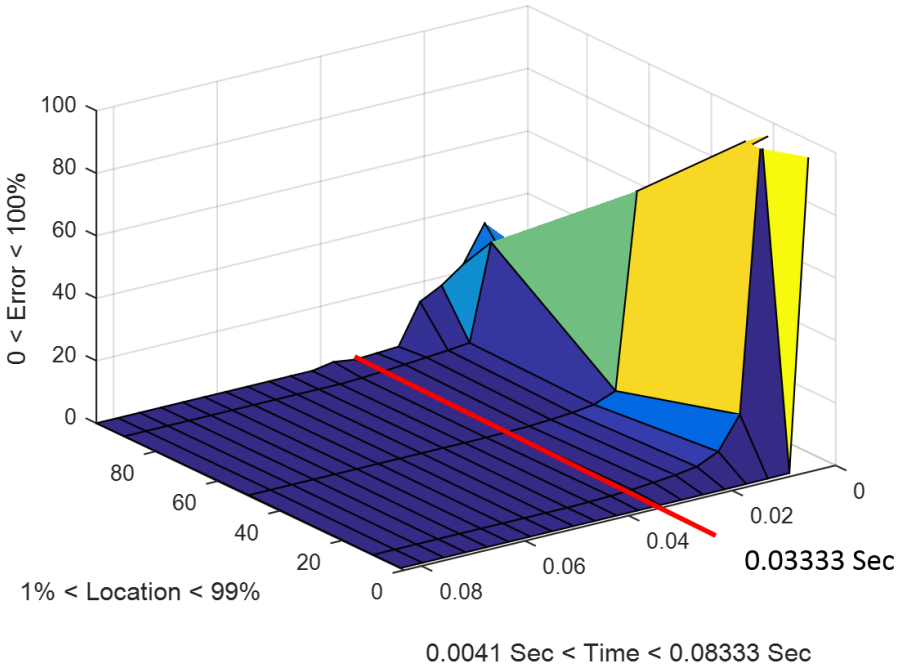


Figure 4.27 3-phase to ground faults in section between tapping nodes N1 and N2

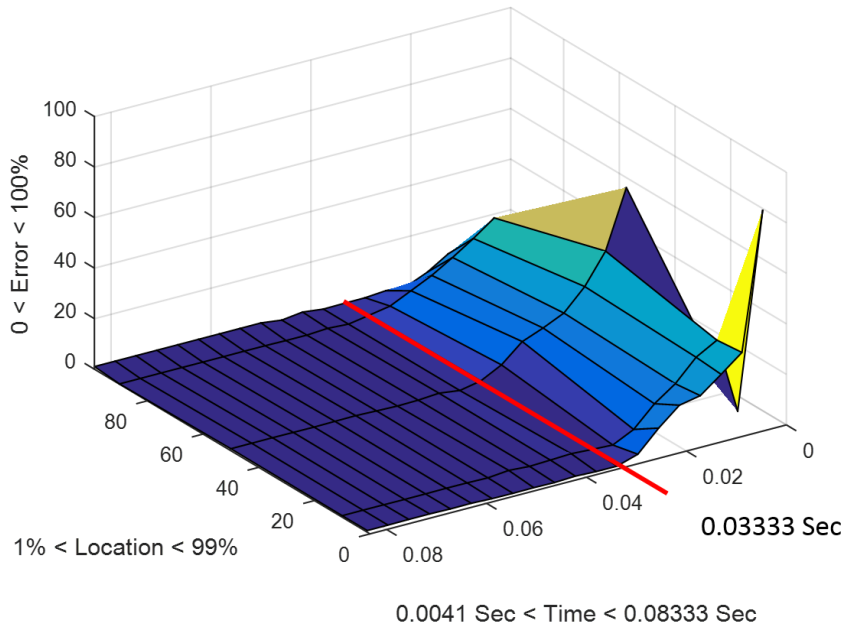


Figure 4.28 3-phase to ground faults in section from terminal T3 to tapping node N2

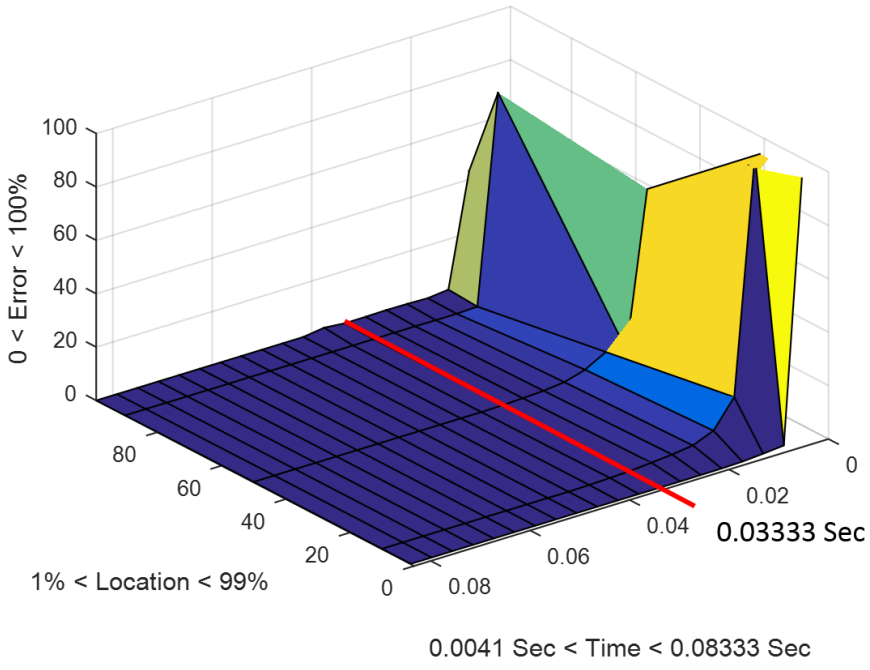


Figure 4.29 3-phase to ground faults in section from terminal T1 to tapping node N1

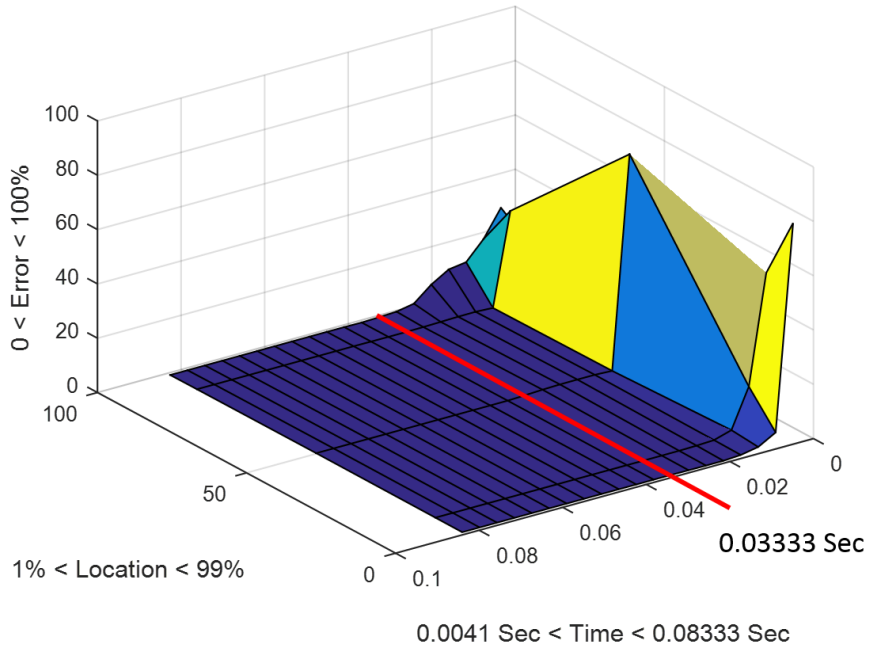


Figure 4.30 Phase A to B faults in section from tapping node N1 to tapping node N2

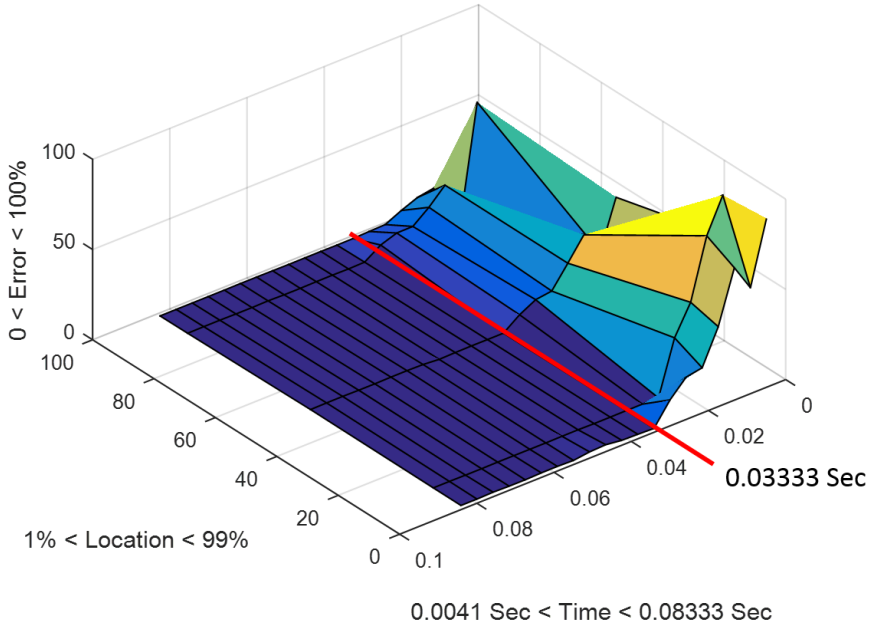


Figure 4.31 Phase A to B faults in section from terminal T3 to tapping node N2

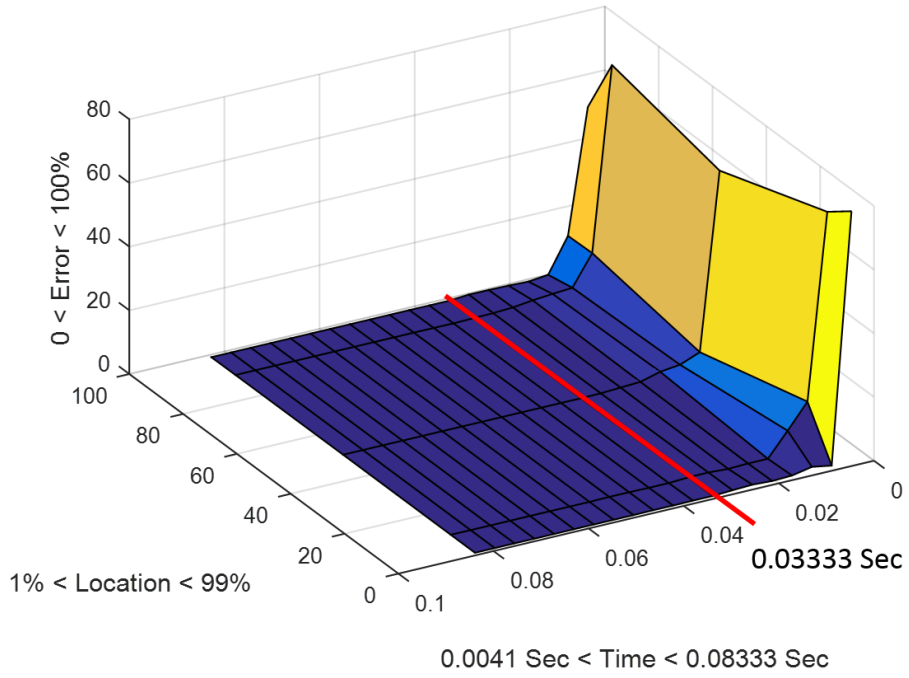


Figure 4.32 Phase A to B faults in section from terminal T1 to tapping node N1

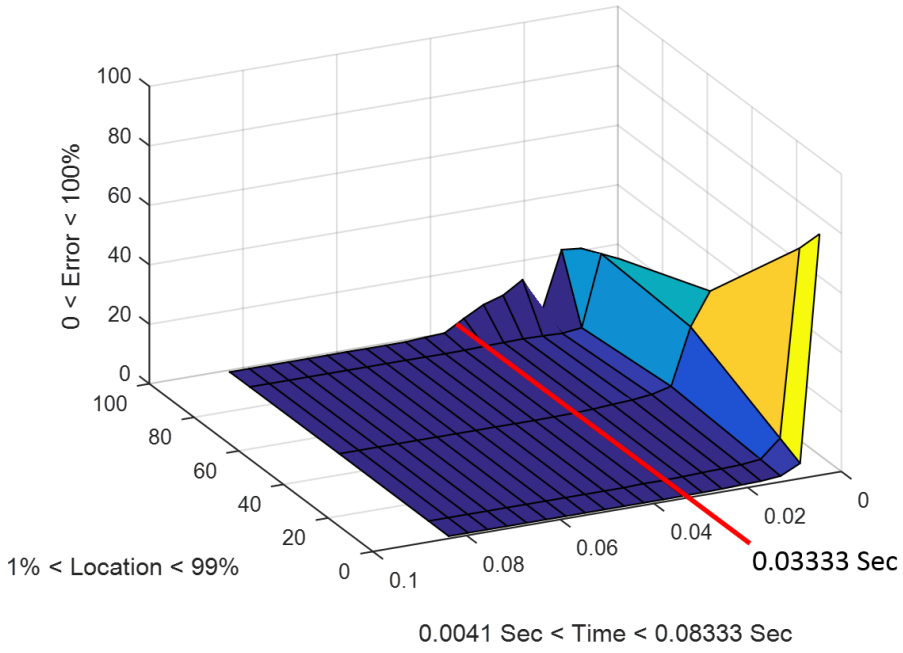


Figure 4.33 3-phase faults in section between tapping node N1 and N2

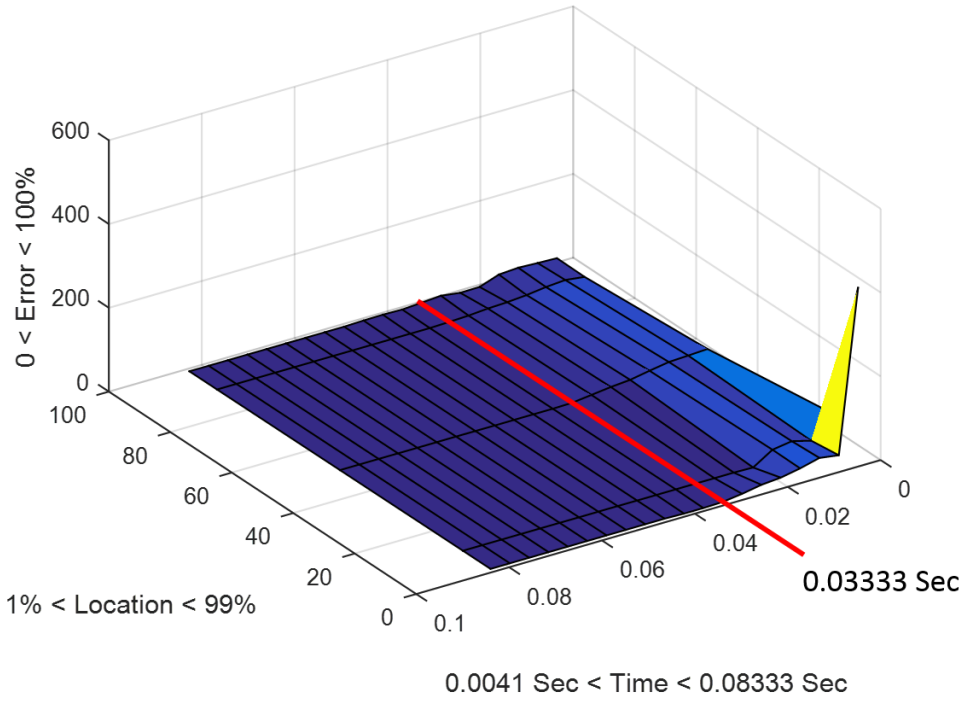


Figure 4.34 3-phase faults in section from terminal T3 to tapping node N2

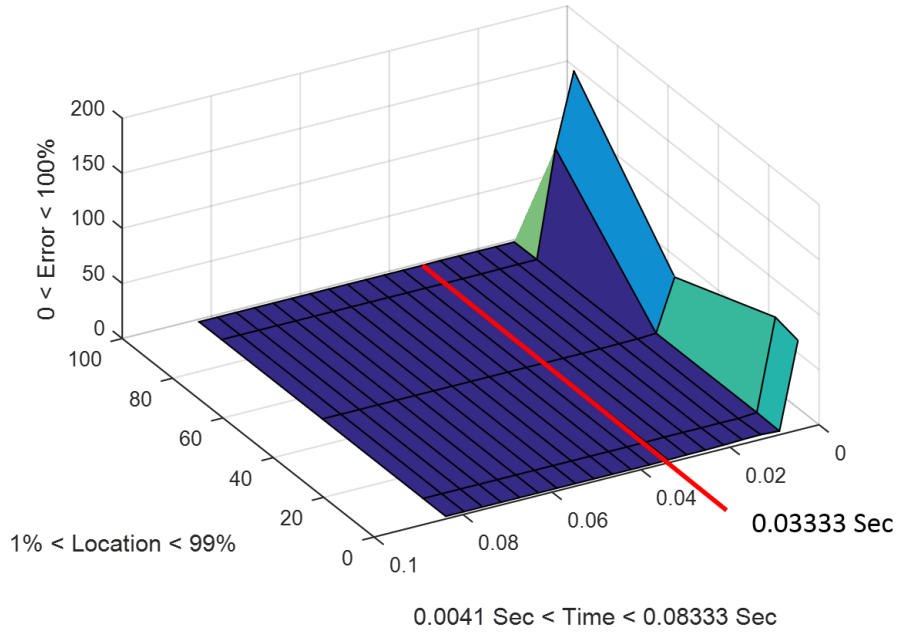


Figure 4.35 3-phase faults in section from terminal T1 to tapping node N1

As seen from Figures 4.24 to 4.35, fault location errors are very high when using the first two frames after the fault. In some cases, errors are beyond hundred percent. Errors when the subsequent four frames are used also large, although not as high as in the case of first two frames. The best time to apply the algorithm seems to be 0.0333 seconds (approximately the eight frames) after the fault. In this case, the highest error is close to 1%, which happens for a single phase to ground fault. Average error is around 0.6%.

In order to investigate the impact due to different reporting rates, another set of tests are created at 50 percent of the branch between *N1* to *N2* for 66kV, 60Hz, 6-terminal distribution feeder shown in Figure 4.19, under different fault conditions (one phase to ground faults, three phase to ground faults, phase to phase faults and three phase faults) and with different reporting rates (240 Frame/s, 120 Frame/s, 60 Frame/s and 30 Frame/s). Branch *N1* to *N2* is selected because its location is in the middle of the transmission line and the data for the tapping points require multiple times of estimation from the starting and ending point of the transmission line, which leads to the highest estimation error.

Two circuit breakers are placed at each end of the transmission line and they are set to operate four cycles after the fault (once cycle detection time + three cycles breaker operating time). The waveforms of the three phase voltage and current measurements under a Phase A to ground fault are shown in Figure 4.36. The magnitudes and phase angles of the corresponding positive sequence voltages and currents as reported by the PMUs are shown in Figures 4.37 and 4.38. As seen from figures, after the fault, there is 0.08 seconds window for the algorithm to calculate the fault location before the breaker operates. The results from the fault location calculations within this time window are shown in Figure 4.39.

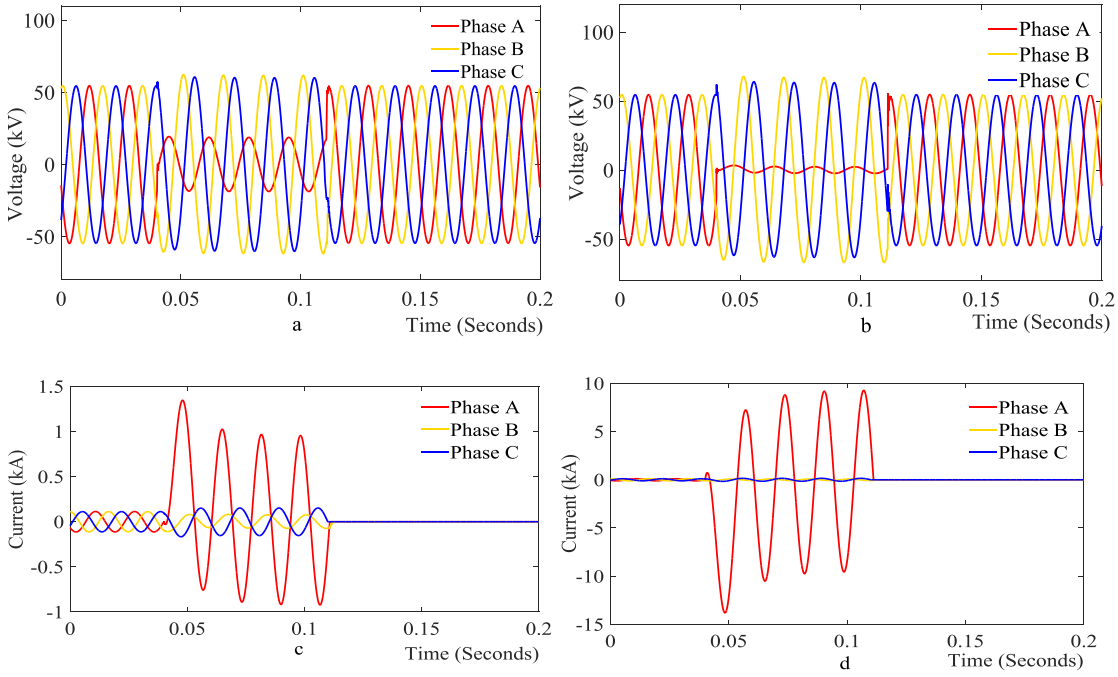


Figure 4.36 Phase voltages and current measurement at two ends (a. Voltage at *N1*, b. Voltage at *N2*, c. Current at *N1*, d. Current at *N2*)

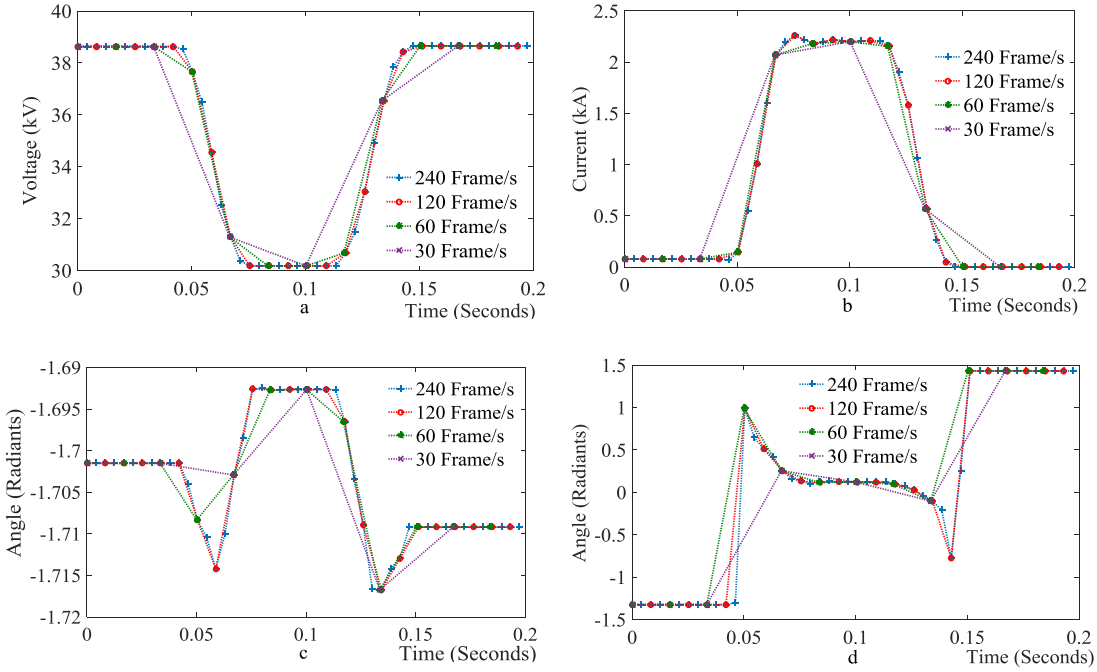


Figure 4.37 Positive sequence synchrophasor measurements at *N1* (a. Voltage magnitude, b. Voltage angle, c. Current magnitude, d. Current angle)

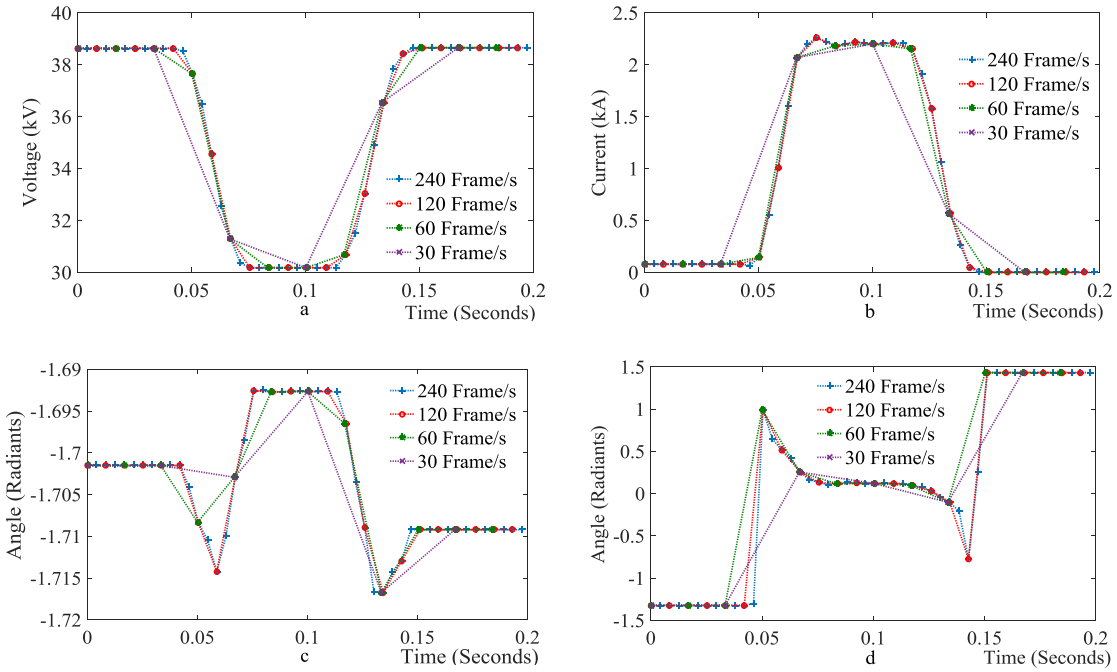


Figure 4.38 Positive sequence synchrophasor measurements at *N2* (a. Voltage magnitude, b. Voltage angle, c. Current magnitude, d. Current angle)

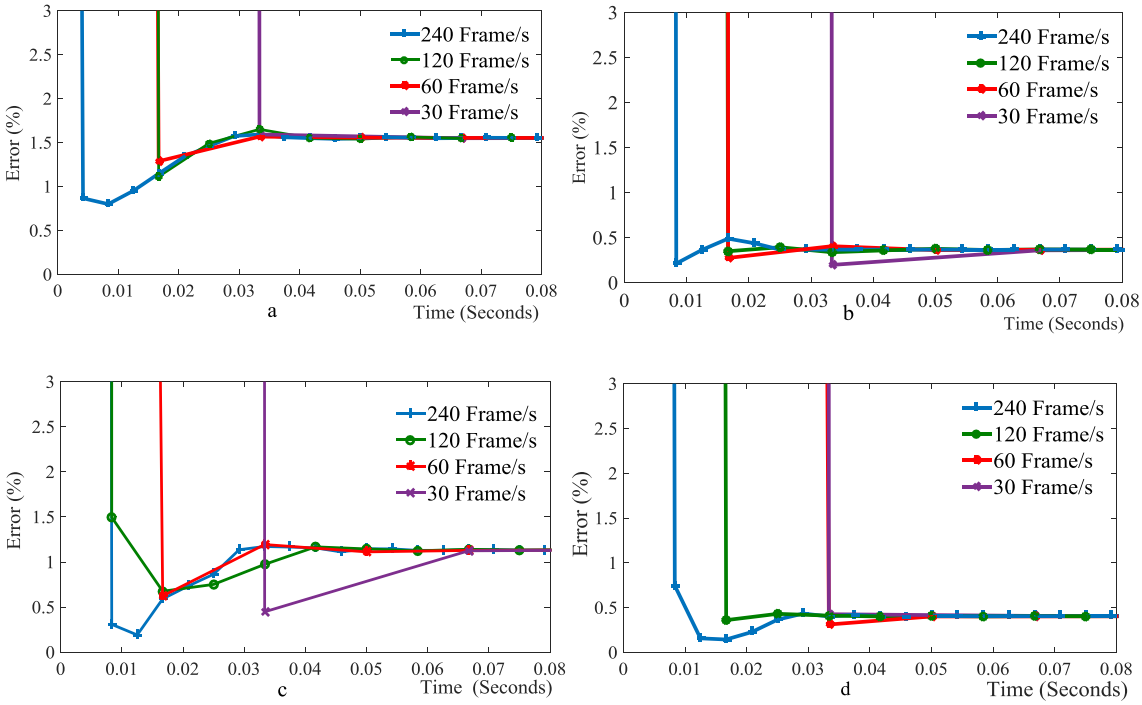


Figure 4.39 Calculated fault location under different reporting rates (a. AG, b. ABCG, c. AB, d. ABC)

As seen from Figure 4.39, errors for the algorithm by using the first two frames after fault are very high. This is because of the latency in phasor measurements: P-class (M-class) filters in PMUs need slightly over 2 (6) cycles of data, and the time-tag is usually corresponding to the center of this data window. The complete effect of the fault will not be reflected in the synchrophasors until two cycles after the fault, if P-class filtering is used. Measurements with M-class filtering may not be useable if fast circuit breakers are used. According to observations, the best time to apply the algorithm is 0.0333 seconds after the fault, which allows use of reporting rates down to 30 frames per second to calculate the correct fault location before the breaker opens. In addition, 0.0333 seconds is two cycle time length. This time length allows P-Class filter to have a complete post fault data set to process and provide synchrophasor measurements corresponding to post-fault period. Allowing this period after the fault decreases the error caused by the non-fault data inside the data window.

In order to cover all possibilities, a set of random tests are conducted with combinations containing all major types of faults on different locations in all line sections. Fault resistance for ground faults can be 10 ohms, 50 ohms or 100 ohms. This range covers most of the high resistance faults [7]. Fault resistances for the non-ground fault is changing between 1 ohm and 10 ohms; this range extends above the average arc resistance from [39]. All measurements input to the algorithm are taken 0.0333 seconds after the fault happens. The testing results for 230kV, 60Hz, 5-terminal power system (Figure 4.18) are tabulated in Table 4.8.

Table 4.8 Fault location performance using data obtained from RTDS simulator for 230kV, 60Hz, 5-terminal transmission line.

Fault				**Result**				
Faulted Section	**Type**	**Location**	**Re-sistance**	**Tapping node**	**Direction**	**Branch**	**Location %**	**Error %**
T1-N1	**ACG**	**95% from T1**	**10**	**N1**	**Backward**	**T1-N1**	**95.01**	**0.01**
			50	**N1**	**Backward**	**T1-N1**	**95.14**	**0.14**
			100	**N1**	**Backward**	**T1-N1**	**95.19**	**0.19**
	ABC	**5% from T1**	**1**	**N1**	**Backward**	**T1-N1**	**4.96**	**0.04**
			10	**N1**	**Backward**	**T1-N1**	**4.96**	**0.04**
			100	**N1**	**Backward**	**T1-N1**	**4.99**	**0.01**
	BG	**50% from T1**	**10**	**N1**	**Backward**	**T1-N1**	**49.89**	**0.11**
			50	**N1**	**Backward**	**T1-N1**	**49.04**	**0.96**
			100	**N1**	**Backward**	**T1-N1**	**48.17**	**1.83**
	AB	**95% from T1**	**1**	**N1**	**Backward**	**T1-N1**	**94.96**	**0.04**
			10	**N1**	**Backward**	**T1-N1**	**94.97**	**0.03**
T2-N1	**AB**	**95% from T2**	**1**	**N1**	**Backward**	**T2-N1**	**94.93**	**0.07**
			10	**N1**	**Backward**	**T2-N1**	**94.96**	**0.04**
	BCG	**5% from T2**	**10**	**N1**	**Backward**	**T2-N1**	**4.72**	**0.28**
			50	**N1**	**Backward**	**T2-N1**	**3.99**	**1.01**
			100	**N1**	**Backward**	**T2-N1**	**2.76**	**2.24**
	CG	**50% from T2**	**10**	**N1**	**Backward**	**T2-N1**	**49.77**	**0.23**
			50	**N1**	**Backward**	**T2-N1**	**48.9**	**1.1**
			100	**N1**	**Backward**	**T2-N1**	**48.54**	**1.46**
	ABC	**95% from T2**	**1**	**N1**	**Backward**	**T2-N1**	**95.18**	**0.18**
			10	**N1**	**Backward**	**T2-N1**	**94.96**	**0.04**

N1-N2	**ABC**	**95% from N1**	**1**	**N1**	**Backward**	**N1-N2**	**94.93**	**0.07**
			10	**N1**	**Backward**	**N1-N2**	**94.94**	**0.06**
	BC	**5% from N1**	**1**	**N1**	**Backward**	**N1-N2**	**4.91**	**0.09**
			10	**N1**	**Backward**	**N1-N2**	**4.91**	**0.09**
	ABG	**50% from N1**	**10**	**N1**	**Backward**	**N1-N2**	**49.74**	**0.26**
			50	**N1**	**Backward**	**N1-N2**	**49.36**	**0.64**
			100	**N1**	**Backward**	**N1-N2**	**49.13**	**0.87**
	CG	**95% from N1**	**10**	**N1**	**Backward**	**N1-N2**	**95.16**	**0.16**
			50	**N1**	**Backward**	**N1-N2**	**95.28**	**0.28**
			100	**N1**	**Backward**	**N1-N2**	**95.4**	**0.4**
T3-N2	**BC**	**95% from T3**	**1**	**N2**	**Forward**	**T3-N2**	**94.99**	**0.01**
			10	**N2**	**Forward**	**T3-N2**	**94.98**	**0.02**
	AG	**95% from T3**	**10**	**N2**	**Backward**	**T3-N2**	**95.34**	**0.34**
			50	**N2**	**Backward**	**T3-N2**	**95.59**	**0.59**
			100	**N2**	**Backward**	**T3-N2**	**95.81**	**0.81**
	ABC	**50% from T3**	**1**	**N2**	**Forward**	**T3-N2**	**49.03**	**0.03**
			10	**N2**	**Backward**	**T3-N2**	**49.76**	**0.76**
	ABG	**5% from T3**	**10**	**N2**	**Backward**	**T3-N2**	**4.21**	**0.78**
			50	**N2**	**Backward**	**T3-N2**	**3.56**	**1.44**
			100	**N2**	**Backward**	**T3-N2**	**2.85**	**2.15**
N2-N3	**BG**	**5% from N2**	**10**	**N2**	**Forward**	**N2-N3**	**94.57**	**0.43**
			50	**N2**	**Forward**	**N2-N3**	**94.12**	**0.88**
			100	**N2**	**Forward**	**N2-N3**	**93.97**	**1.03**
	ACG	**5% from N2**	**10**	**N2**	**Forward**	**N2-N3**	**94.71**	**0.29**
			50	**N2**	**Forward**	**N2-N3**	**94.54**	**0.46**
			100	**N2**	**Forward**	**N2-N3**	**94.25**	**0.75**

	ABC	**50% from N2**	**1**	**N2**	**Forward**	**N2-N3**	**49.21**	**0.79**
			10	**N2**	**Forward**	**N2-N3**	**49.23**	**0.77**
	AC	**95% from N2**	**1**	**N2**	**Forward**	**N2-N3**	**4.09**	**0.91**
			10	**N2**	**Forward**	**N2-N3**	**4.1**	**0.9**
T4-N3	**ABC**	**95% from T4**	**1**	**N3**	**Backward**	**T4-N3**	**94.98**	**0.02**
			10	**N3**	**Backward**	**T4-N3**	**95.25**	**0.25**
	BC	**5% from T4**	**1**	**N3**	**Backward**	**T4-N3**	**4.93**	**0.07**
			10	**N3**	**Backward**	**T4-N3**	**4.92**	**0.08**
	AG	**50% from T4**	**10**	**N3**	**Backward**	**T4-N3**	**49.66**	**0.34**
			50	**N3**	**Backward**	**T4-N3**	**49.11**	**0.89**
			100	**N3**	**Backward**	**T4-N3**	**48.37**	**1.63**
	ACG	**95% from T4**	**10**	**N3**	**Backward**	**T4-N3**	**94.96**	**0.04**
			50	**N3**	**Backward**	**T4-N3**	**94.97**	**0.03**
			100	**N3**	**Backward**	**T4-N3**	**95.04**	**0.04**
T5-N3	**ABG**	**5% from T5**	**10**	**N3**	**Forward**	**T5-N3**	**5.11**	**0.11**
			50	**N3**	**Forward**	**T5-N3**	**5.43**	**0.43**
			100	**N3**	**Forward**	**T5-N3**	**6.02**	**1.02**
	ABC	**5% from T5**	**1**	**N3**	**Forward**	**T5-N3**	**5.01**	**0.01**
			10	**N3**	**Forward**	**T5-N3**	**4.99**	**0.01**
	BC	**50% from T5**	**1**	**N3**	**Forward**	**T5-N3**	**49.97**	**0.03**
			10	**N3**	**Forward**	**T5-N3**	**49.96**	**0.04**
	CG	**95% from T5**	**10**	**N3**	**Forward**	**T5-N3**	**94.83**	**0.17**
			50	**N3**	**Forward**	**T5-N3**	**94.64**	**0.36**
			100	**N3**	**Forward**	**T5-N3**	**94.26**	**0.74**

As seen from Table 4.8, the faulty branch can be efficiently identified using the first and second steps of the proposed algorithm. The error of fault location obtained from the third step of algorithm is reasonable: Over 90% of the 71 cases tested reported errors lower than 1%. Errors for unbalance phase-to-ground faults with 100 ohms fault impedance are higher, but still remained around 2%.

In addition, the structure of the second test system, 66kV, 60Hz, 6-terminal distribution feeder, is given in Figure 4.19. The test results for 66kV, 60Hz, 6-terminal distribution feeder are tabulated in Table 4.9.

Table 4.9 Fault location performance using data obtained from RTDS simulator for 66kV, 60Hz, 6-terminal distribution feeder.

Fault				**Result**				
Faulted Section	**Type**	**Location**	**Resistance**	**Tapping node**	**Direction**	**Branch**	**Location %**	**Error %**
T1-N1	**ACG**	**95% from T1**	**10**	**N1**	**Backward**	**T1-N1**	**95.79**	**0.79**
			50	**N1**	**Backward**	**T1-N1**	**95.82**	**0.82**
			100	**N1**	**Backward**	**T1-N1**	**95.81**	**0.81**
	ABC	**5% from T1**	**1**	**N1**	**Backward**	**T1-N1**	**5.06**	**0.06**
			10	**N1**	**Backward**	**T1-N1**	**5.08**	**0.08**
			100	**N1**	**Backward**	**T1-N1**	**5.03**	**0.03**
	BG	**50% from T1**	**10**	**N1**	**Backward**	**T1-N1**	**50.12**	**0.12**
			50	**N1**	**Backward**	**T1-N1**	**50.12**	**0.12**
			100	**N1**	**Backward**	**T1-N1**	**50.13**	**0.13**
	AB	**95% from T1**	**1**	**N1**	**Backward**	**T1-N1**	**96.24**	**1.24**
			10	**N1**	**Backward**	**T1-N1**	**96.20**	**1.20**

T2-N1	**AB**	**5% from T2**	**1**	**N1**	**Backward**	**T2-N1**	**6.18**	**1.18**
			10	**N1**	**Backward**	**T2-N1**	**6.67**	**1.67**
	BCG	**5% from T2**	**10**	**N1**	**Forward**	**T2-N1**	**6.01**	**1.01**
			50	**N1**	**Forward**	**T2-N1**	**6.24**	**1.24**
			100	**N1**	**Forward**	**T2-N1**	**6.53**	**1.53**
	CG	**50% from T2**	**10**	**N1**	**Backward**	**T2-N1**	**52.71**	**2.71**
			50	**N1**	**Backward**	**T2-N1**	**52.08**	**2.08**
			100	**N1**	**Backward**	**T2-N1**	**52.44**	**2.44**
	ABC	**95% from T2**	**1**	**N1**	**Backward**	**T2-N1**	**97.88**	**2.88**
			10	**N1**	**Backward**	**T2-N1**	**97.34**	**2.34**
N1-N2	**ABC**	**95% from N1**	**1**	**N2**	**Backward**	**N1-N2**	**96.81**	**1.81**
			10	**N2**	**Backward**	**N1-N2**	**95.15**	**0.15**
	BC	**5% from N1**	**1**	**N1**	**Forward**	**N1-N2**	**95.51**	**0.51**
			10	**N1**	**Forward**	**N1-N2**	**95.52**	**0.52**
	ABG	**50% from N1**	**10**	**N1**	**Forward**	**N1-N2**	**51.67**	**1.67**
			50	**N1**	**Forward**	**N1-N2**	**51.59**	**1.59**
			100	**N1**	**Forward**	**N1-N2**	**51.6**	**1.6**
	CG	**95% from N1**	**10**	**N2**	**Backward**	**N1-N2**	**95.08**	**0.08**
			50	**N2**	**Backward**	**N1-N2**	**95.09**	**0.09**
			100	**N2**	**Backward**	**N1-N2**	**95.09**	**0.09**
T3-N2	**ABC**	**95% from T3**	**1**	**N2**	**Forward**	**T3-N2**	**95.45**	**0.45**
			10	**N2**	**Backward**	**T3-N2**	**95.81**	**0.81**
	AG	**95% from T3**	**10**	**N2**	**Forward**	**T3-N2**	**96.27**	**1.27**
			50	**N2**	**Forward**	**T3-N2**	**96.26**	**1.26**
			100	**N2**	**Forward**	**T3-N2**	**96.15**	**1.15**
	ABC		**1**	**N2**	**Backward**	**T3-N2**	**50.08**	**0.08**

		50% from T3	**10**	**N2**	**Backward**	**T3-N2**	**50.14**	**0.14**
	ABG	**5% from T3**	**10**	**N2**	**Backward**	**T3-N2**	**5.64**	**0.64**
			50	**N2**	**Backward**	**T3-N2**	**5.45**	**0.45**
			100	**N2**	**Backward**	**T3-N2**	**5.58**	**0.58**
N2-N3	**BG**	**5% from N2**	**10**	**N3**	**Backward**	**N2-N3**	**5.63**	**0.63**
			50	**N3**	**Backward**	**N2-N3**	**5.62**	**0.62**
			100	**N3**	**Backward**	**N2-N3**	**5.62**	**0.62**
	ACG	**5% from N2**	**10**	**N3**	**Backward**	**N2-N3**	**5.27**	**0.27**
			50	**N3**	**Backward**	**N2-N3**	**5.28**	**0.28**
			100	**N3**	**Backward**	**N2-N3**	**5.27**	**0.27**
	ABC	**50% from N2**	**1**	**N3**	**Backward**	**N2-N3**	**50.29**	**0.29**
			10	**N3**	**Backward**	**N2-N3**	**50.34**	**0.34**
	AC	**95% from N2**	**1**	**N3**	**Backward**	**N2-N3**	**95.45**	**0.45**
			10	**N3**	**Backward**	**N2-N3**	**95.51**	**0.51**
T5-N3	**ABC**	**95% from T5**	**1**	**N3**	**Forward**	**T5-N3**	**95.53**	**0.53**
			10	**N3**	**Forward**	**T5-N3**	**95.84**	**0.84**
	ABC	**5% from T5**	**1**	**N3**	**Forward**	**T5-N3**	**5.3**	**0.30**
			10	**N3**	**Forward**	**T5-N3**	**6.43**	**1.43**
	AG	**50% from T5**	**10**	**N3**	**Forward**	**T5-N3**	**55.79**	**5.79**
			50	**N3**	**Forward**	**T5-N3**	**54.92**	**4.92**
			100	**N3**	**Forward**	**T5-N3**	**55.63**	**5.63**
	ACG	**95% from T5**	**10**	**N3**	**Forward**	**T5-N3**	**95.23**	**0.23**
			50	**N3**	**Forward**	**T5-N3**	**95.59**	**0.59**
			100	**N3**	**Forward**	**T5-N3**	**95.1**	**0.10**
T6-N3	**ABG**		**10**	**N3**	**Forward**	**T6-N3**	**5.04**	**0.04**

		5% from T6	50	N3	Forward	T6-N3	5.05	0.05
			100	N3	Forward	T6-N3	5.03	0.03
	ABC	5% from T6	1	N3	Forward	T6-N3	5.01	0.01
			10	N3	Forward	T6-N3	5.01	0.01
	BC	50% from T6	1	N3	Forward	T6-N3	50.03	0.03
			10	N3	Forward	T6-N3	50.01	0.01
	CG	95% from T6	10	N3	Forward	T6-N3	95.05	0.05
			50	N3	Forward	T6-N3	95.06	0.06
			100	N3	Forward	T6-N3	95.05	0.05
T4-N2	AB	95% from T4	1	N2	Forward	T4-N2	95.47	0.47
			10	N2	Forward	T4-N2	95.51	0.51
	BCG	5% from T4	10	N2	Backward	T4-N2	2.79	2.21
			50	N2	Backward	T4-N2	2.77	2.23
			100	N2	Backward	T4-N2	3.06	1.94
	CG	50% from T4	10	N2	Backward	T4-N2	51.43	1.43
			50	N2	Backward	T4-N2	51.48	1.48
			100	N2	Backward	T4-N2	51.42	1.42
	ABC	95% from T4	1	N2	Forward	T4-N2	95.05	0.05
			10	N2	Forward	T4-N2	95.03	0.03

As seen from Table 4.9, the algorithm's second and third steps for fault branch selection process can efficiently identify the faulty branch. The error of fault location is reasonable. Over 70% of the 81 test cases reported errors lower than 2%. The highest errors are less than 5%.

In addition, some the bus faults at tapping nodes, and the fault locations which are very close to bus (1 percent of total length) are also tested using real time simulation. In both cases, the correct tapping node is reported, and a near zero normalized fault distance is given. However, some time if the fault is happened very close to the one of the transmission line ends (50 meters away from the bus), the third step reports a wrong branch. Again, a warning can be issued to the user under such extreme conditions.

To sum up, the results from above tests show that the proposed algorithm is a fast and straightforward algorithm with a good accuracy. However, the possible sources of errors for the proposed fault location cannot be overlooked. First, measurements from PMU during fault can be damaged due to current transformer saturation. In addition, modern breakers operates very fast which leaves a small time window for the algorithm to apply. The missing data situation can cause the algorithm to miss suitable post-fault data.

4.4 Chapter summary

In this chapter, a synchrophasor measurement-based fault location algorithm for multi-terminal, non-homogeneous transmission lines was presented. Proposed algorithm can track down the faulty branch and its normalized fault distance without using thresholds or other predetermined aids. The experimental setup consisted of an actual synchrophasor network and a PDC, and thus practical implementation aspects are taken into account in the tests. The algorithm was implemented as an application launched from “PhasorEye” software. Numerous real time simulations were conducted to validate the algorithm. The results confirm the accuracy of the algorithm.

Chapter 5

Other real time applications

On the "PhasorEye" software platform developed in this thesis, new synchrophasor based real-time power system analysis tools can be implemented. In this chapter, implementation of several synchrophasor applications, some developed by other University of Manitoba researchers [41], [42] and [46], on "PhasorEye" software platform is presented. The details of the implemented algorithms, which have been published elsewhere, are not given here to limit the length of the chapter. However, very significant effort was devoted for the implementations presented in this chapter. It required reviewing relevant literature, understanding algorithms, consulting the original authors for clarifications, coding of algorithms, developing suitable test cases in real-time simulator, debugging code, and testing the applications. Despite the amount of work and challenges involved, a number of synchrophasor applications were successfully implemented to demonstrate the utility of "PhasorEye" platform.

5.1 Multi-thread real-time application

In the previous chapter, the application proposed for fault location utilizes recorded synchrophasor measurements in offline mode, although the calculations are done immediately after a fault. Such an application with a single feeding source can be easily written as a

black box function with a single thread program. However, if an application has many feeding sources and requires multiple copies of black box functions, multiple threads programming is one of the solutions available to increase the responsiveness and limit the delays and interferences.

In Section 3.2, titled “Connection and display”, the idea and advantages of multiple thread operation is pitched for the communication connection and data display in “PhasorEye” software. However, the main potential for such structure is not just limited in the communication aspect; it is also expanded to deploy analysis methods which are packed as black box functions. For instance, the simultaneous access to multiple threads grants the developer freedom to write control mechanisms to abandon an analysis method which is broken or using the out of date data. The developer doesn't have to worry about the step that is being currently processed inside of black box function and return the result from recursive call of functions. Instead, the method thread can be terminated and the problem can be logged in order to avoid the interference for real-time operation. The problem can be studied later.

Figure 5.1 shows the structure implementation inside “PhasorEye” software. The “StaticItem” starts to create the two parts of the program which are the data structures and the content pages hosted in the main window at the beginning of runtime. The data structure contains the instances of the flags such as the connection status flag “isOnline”, the recording status flag “isRecording” and declares the data fields such as the connection information, the null array for the processed data (Figure 5.2) which are needed for future configurations.

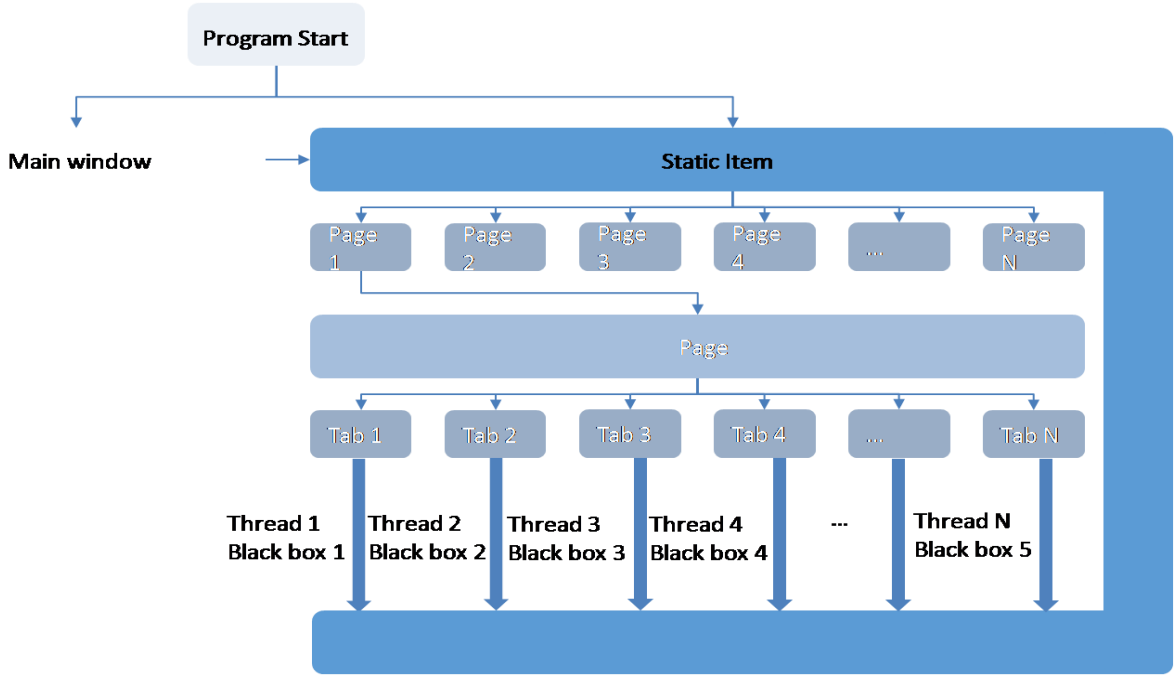


Figure 5.1 Structure implementation inside "PhasorEye" software

```
namespace SynchrophasorConnector_WPF
{
    public static class StaticItem
    {
        private static string organization = "Intelligent Power Grid Laboratory - University of Manitoba";
        public static string Organization...

        private static string versionNumber = "Version: 1.5";
        public static string VersionNumber...

        private static string windowMainStatusBarField3 = "Offline";
        public static string WindowMainStatusBarField3...

        private static bool isOnline = false;
        public static bool IsOnline...


        private static PMUdata[] pmu = null;
        public static PMUdata[] Pmu...

        Connection Information

        Window and Pages

        Transient Stability Status Padiction

        Data Recording

    }
}
```

Figure 5.2 Code for structure implementation inside "PhasorEye" software

For the content page, initially only the starting logo page and the connection configuration page are created. However, when a user selects a program through the menu bar at the left hand side of the main window, the pervious page will be hidden into the background, and a new page will be created or brought from the background to the main window. The reference for the movement of the windows and pages creations or rising and hiding is recorded in the "StaticItem" layer. Additionally, the results of each thread is also reported into "StaticItem" layer data structure. However, tab controls and threads which are operating inside content page are owned by the content page itself. In other word tab controls and threads do not belong to the data layer "Static Item", they belong to the business layer with the pages. The details and examples for the page implementation are discussed in Section 5.1.1. To sum up, for content pages, two parts are controlled by the "StaticItem" layer. First part is content page's reference and UI visual status. Second part is reference for the data structures which are used for the thread inside pages.

5.1.1 Real-Time monitoring of transmission line parameters

In this section, implementation of an application for real-time monitoring of transmission line parameters is presented. The relationship between the content pages, tab control and operation thread are explained.

The application implemented for real-time monitoring of transmission line parameters is based on an algorithm proposed by D. R. Gurusinghe and A. D. Rajapakse [46]. A series of synchrophasor measurements, which are selected according to the settings input by the user through the graphical user interface window are packed into a matrix. The measurements should include synchrophasor measurements of the voltages and currents at two ends

of the transmission line. Using the algorithm proposed in [46], the unknown parameters of the equivalent π-model of the transmission line or characteristic impedance and propagation constant can be estimated using the measurements. This algorithm formulates the problem as a linear least square problem and therefore directly solved. If the line is series compensated, the problem becomes nonlinear and the solution can be found through nonlinear least square estimation using an iterative process based on the Newton method. Once the solution updates reach within a specified tolerance, the iteration process is terminated. The algorithm is built as a class inside "PhasorEye" program, and is used inside "PhasorEye" program page class "PageLineParameterCalculation" for the line parameter calculation page as shown in Figure 5.3.

```
namespace SynchrophasorConnector_WPF
{
    /// <summary> Interaction logic for PageLineParameterCalculation.xaml
    public partial class PageLineParameterCalculation : Page
    {
        private List<TabItem_Thread> tabItems;
        private TabItem_Thread tabAdd;


        public PageLineParameterCalculation()...

        public void UIChange()...
        private void ButtonAddTlineParameterCalculationDisplayTab_Click(object sender, RoutedEventArgs e)...

        TabControlTlineParameterCalculationDisplay

        ComboBox

        ButtonTlineParameter

        recording


    }
    public class TabItem_Thread : TabItem
    {


        UI

        Result Saving


        control

        Calculation

    }
}
```

Figure 5.3 Class "PageLineParameterCalculation"

Like all pages inside “PhasorEye” program, the line parameter calculation page has three components which are the page class, the tab class and the thread. The page component contains the configuration tools for user to customize the setting for algorithm such as the souce of the measuremts and the recording of the results (Figure 5.4 top). After the user changes one of the setting in the graphical user interface, the setting data is loaded into the page class “PageLineParameterCalculation”. After entering all the settings, the user can click the “Calculate” button on the user interface to create a thread to start the algorithm and a tab which is showing the results from the algorithm (Figure 5.4 top). In addition, the object for the tab in “PhasorEye” program is the “TabItem_Thread” class, it is inheriting from “TabItem” class to have all the graphic feature to function as a tab, and also bonded with one or more threads to run the algorithm without interference with the user interface changes. Shown in Figure 5.4 is the code for the line parameter calculation page, and the three components for the page, page class “PageLineParameterCalculation”, tab class “TabItem_Thread” and thread lists “tabItems” and “tabAdd”. Figure 5.5 is the online testing result, this online result is agree with the offline analysis result presented in [46].

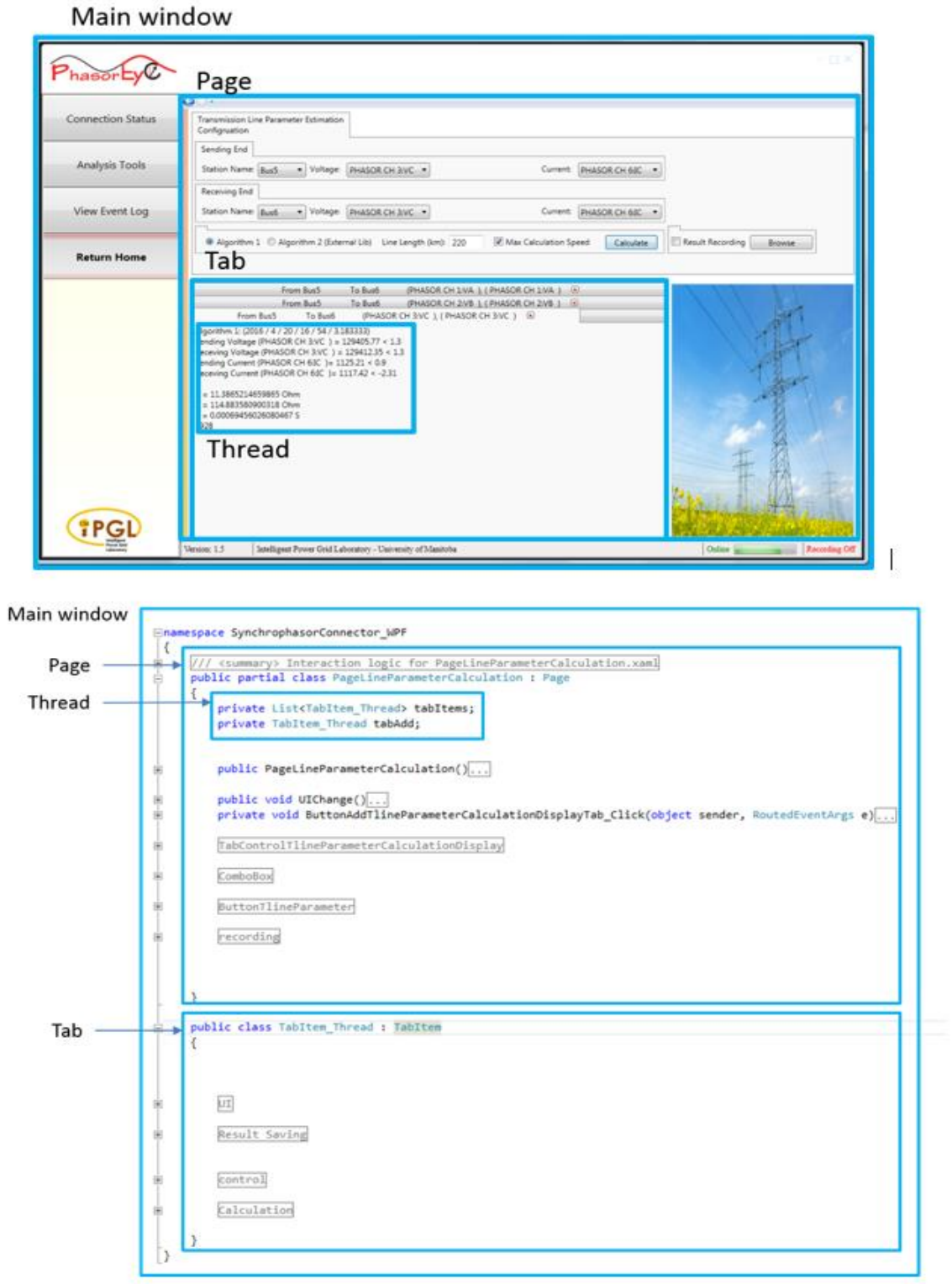


Figure 5.4 Line parameter calculation page (Top) and Class "PageLineParameter-Calculation" (Bottom)

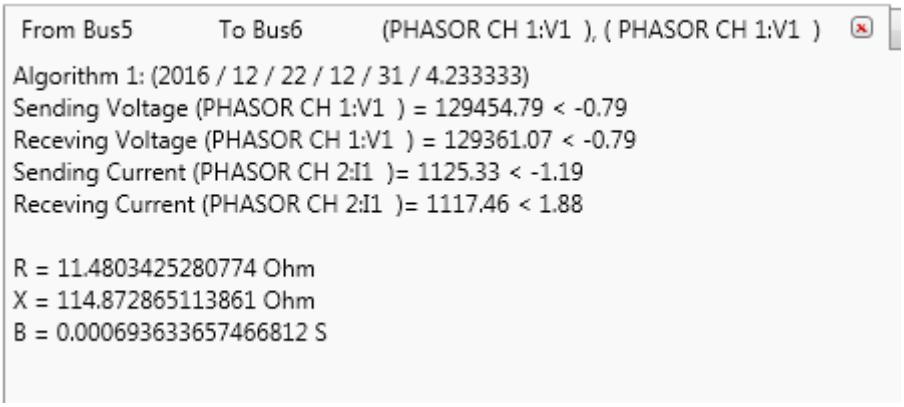


Figure 5.5 Online line parameters monitoring result

5.1.2 Post-disturbance transient stability status prediction

The application implemented on "PhasorEye" software for the prediction of post-disturbance transient stability status of a power system is based on an algorithm proposed by D. R. Gurusinghe and A. D. Rajapakse in [41]. The transient swings in post-disturbance stage are classified as a stable swings or unstable swings by examining trajectories of generator operating points on the rate of change of voltage (ROCOV) versus the deviations of the voltage magnitudes (ΔV) plane. Voltage measurements are obtained from the PMUs which are installed at the generator terminals.

This algorithm is integrated into "PhasorEye" software, and tested in IEEE 39 bus system shown in Figure 5.6. Figure 5.7 shows test result for a case where the system is unstable after a fault. The snapshot at top of Figure 5.7 shows the variation of the voltage of generator 38 after the fault as indicated by the synchrophasor measurements; the bottom snapshot shows the trajectory of generator 38 and the stability boundary on ROCOV and ΔV plane.

In addition, the class diagram for both implementations are shown in Figure 5.8. When compared the class sizes of the two implementations, class size of the algorithm class with multi-thread operation is halved compared to that of the single thread implementation. This application shows that the code reusability in the multi-thread application is increased, leading to a small and manageable code that is easily debugged and maintained.

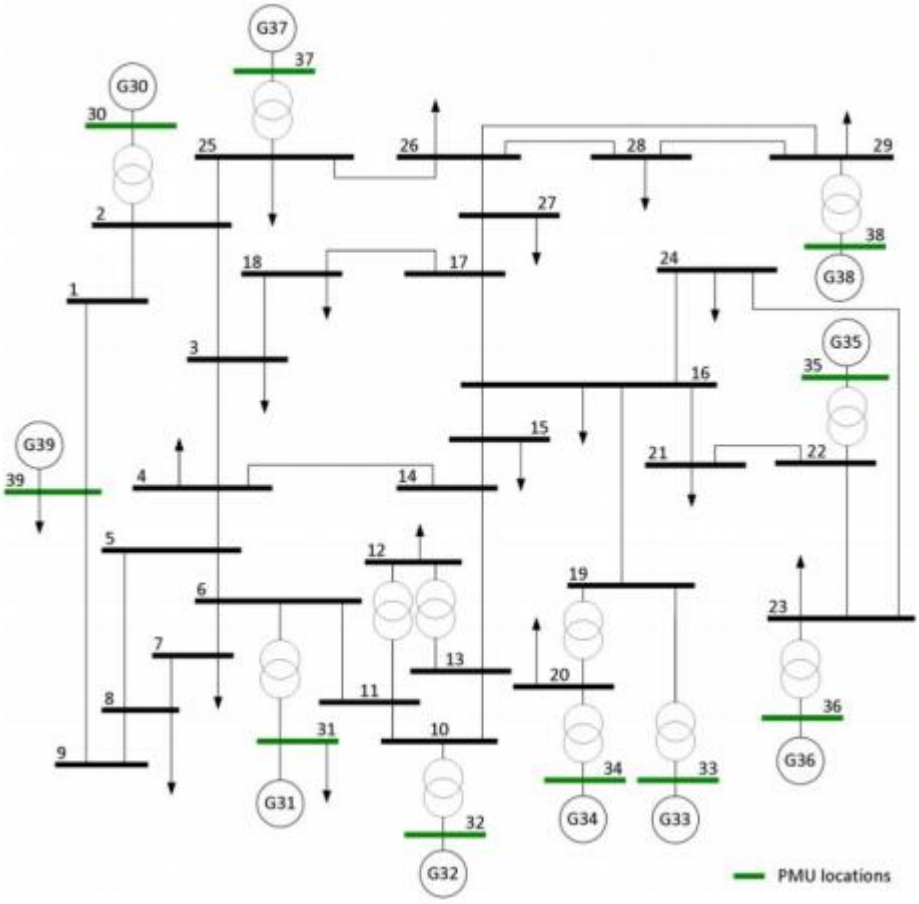


Figure 5.6 IEEE 39 bus system [41]

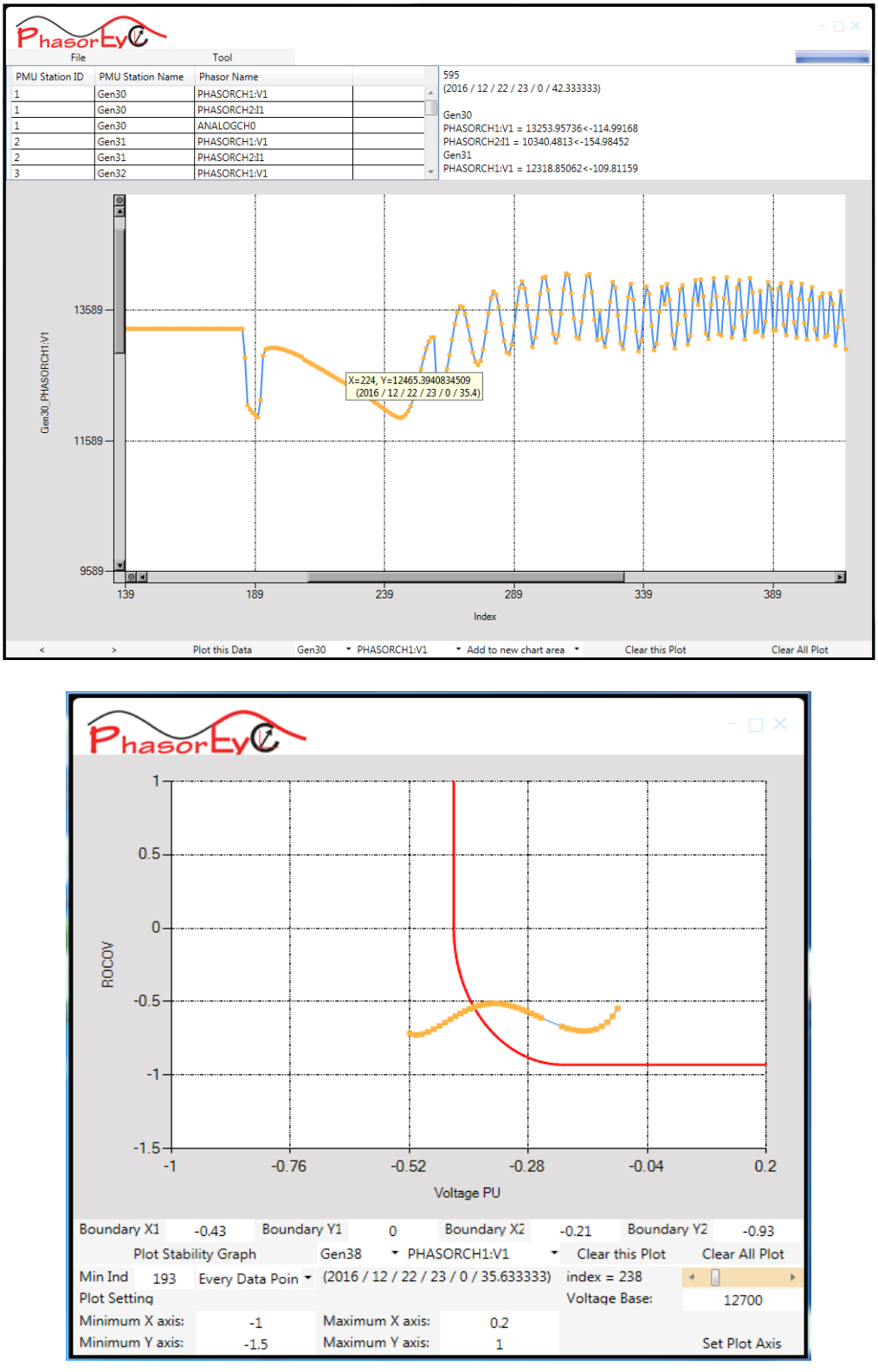


Figure 5.7 Post-disturbance transient stability status prediction for unstable case.

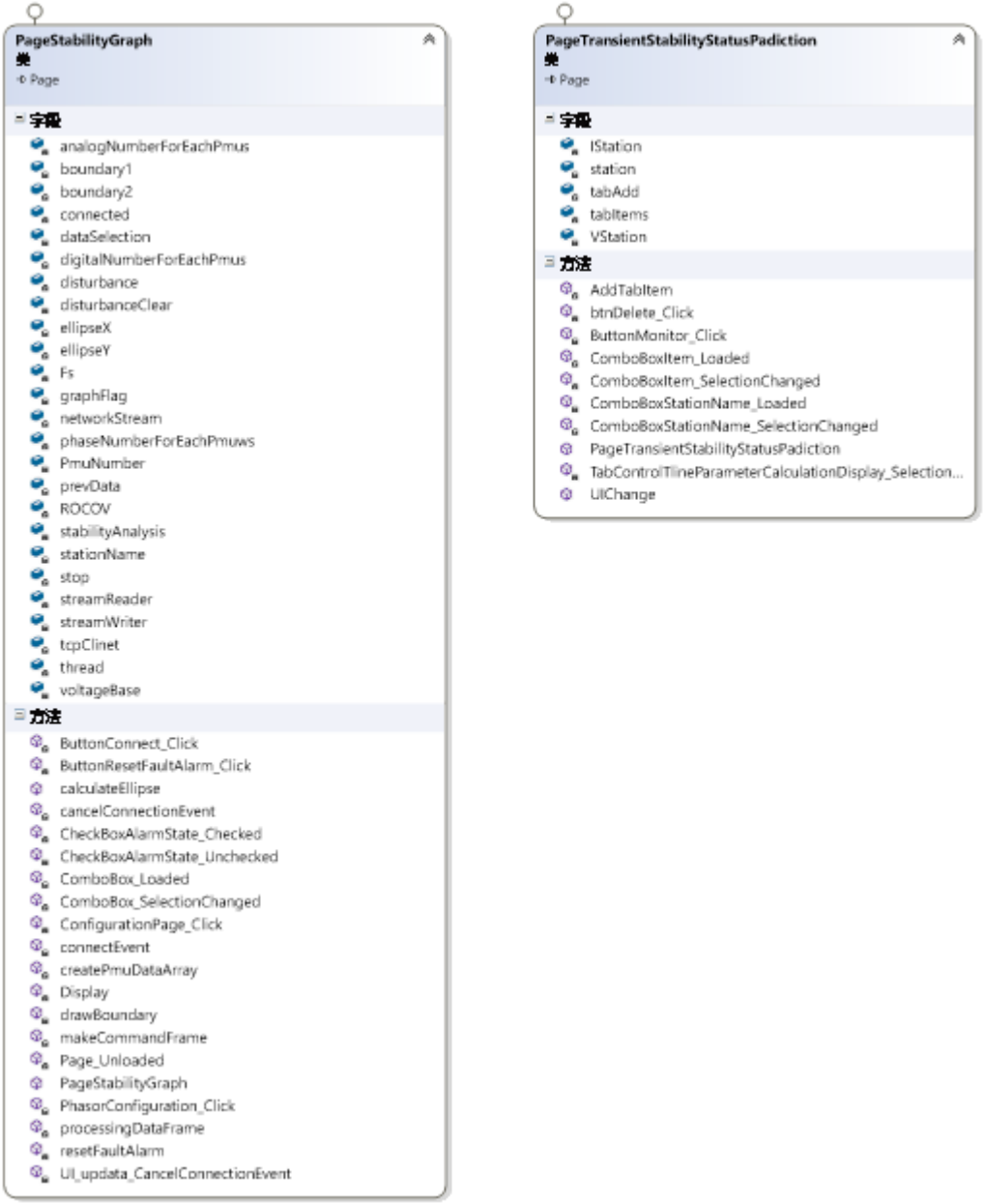


Figure 5.8 Class diagram for Post-disturbance transient stability status prediction in the first edition of "PhasorEye" program (Left) and in the fifth edition of "PhasorEye" program (Right)

5.1.3 Identification of dominant low-frequency modes in ring-down oscillations

The third application implemented on PhasorEye" is a monitoring tool for the identification of dominant low-frequency modes in ring-down oscillations. This application is based on an algorithm proposed by D. P. Wadduwage, U. D. Annakkage and K. Narendra in [42]. The modes and damping of ring-down oscillations appearing in a power system, similar to the one illustrated in Figure 5.9, can be extracted using Prony analysis with varying orders [42]. In order to change the order of Prony analysis, a shrinking window approach with multiple sampling is used. Flow chart for this algorithm is shown in Figure 5.10.

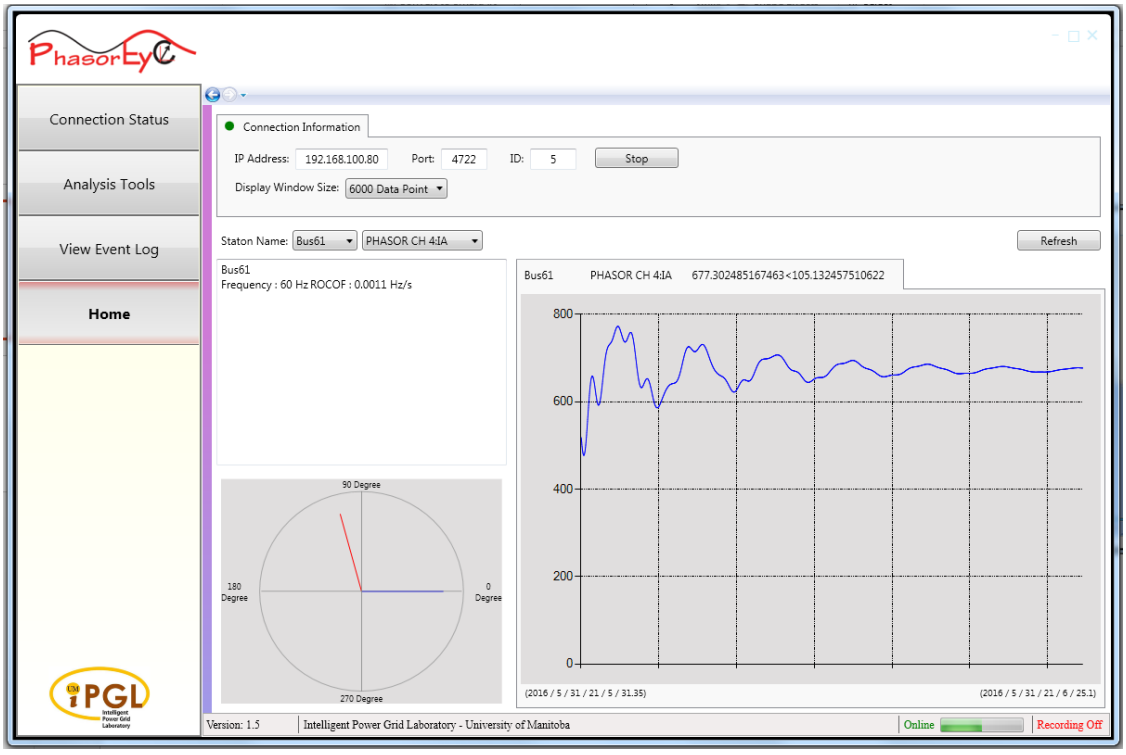


Figure 5.9 Real time monitoring of ring-down oscillations

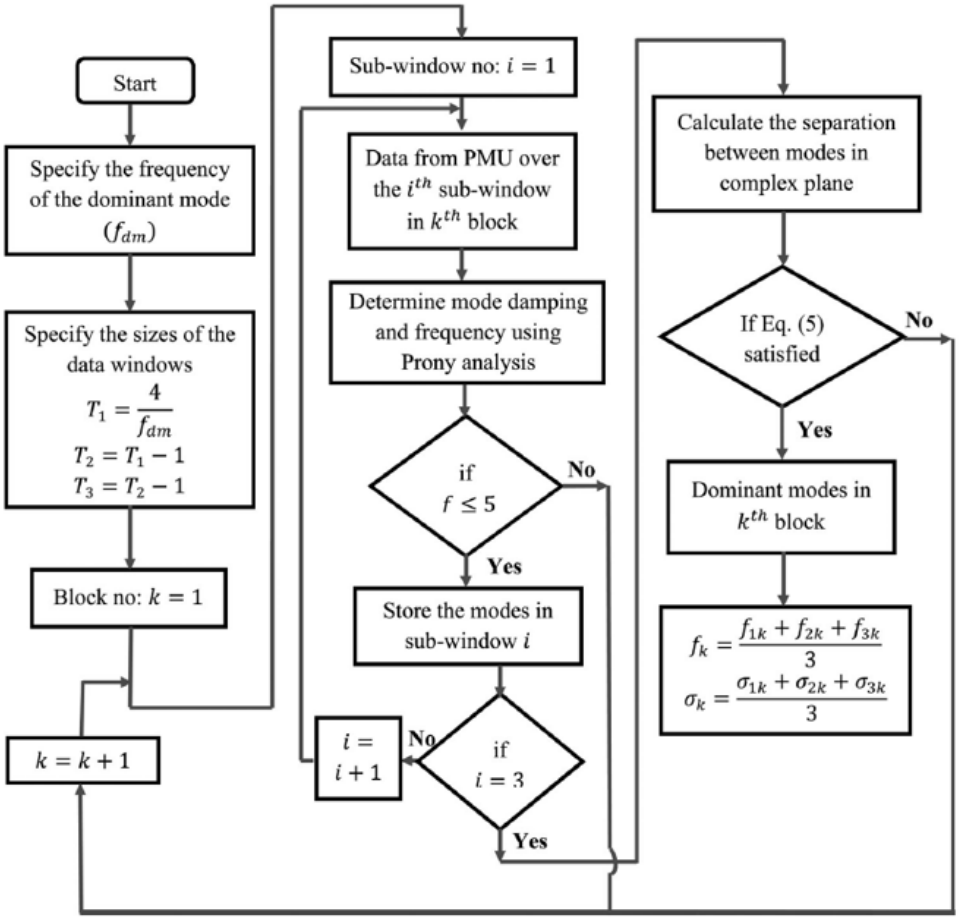


Figure 5.10 Flow chart of the algorithm for monitoring of ring-down oscillations [41]

The difficulty for implementing the above algorithm as a real-time application is the long processing times required for solving high order equations, and necessity for repeated calculation for different window sizes. However, with the aid of "PhasorEye" multi-thread structure, the processing time could be improved. Thread diagram for monitoring of ring-down oscillations algorithm in "PhasorEye" software is shown in Figure 5.11.

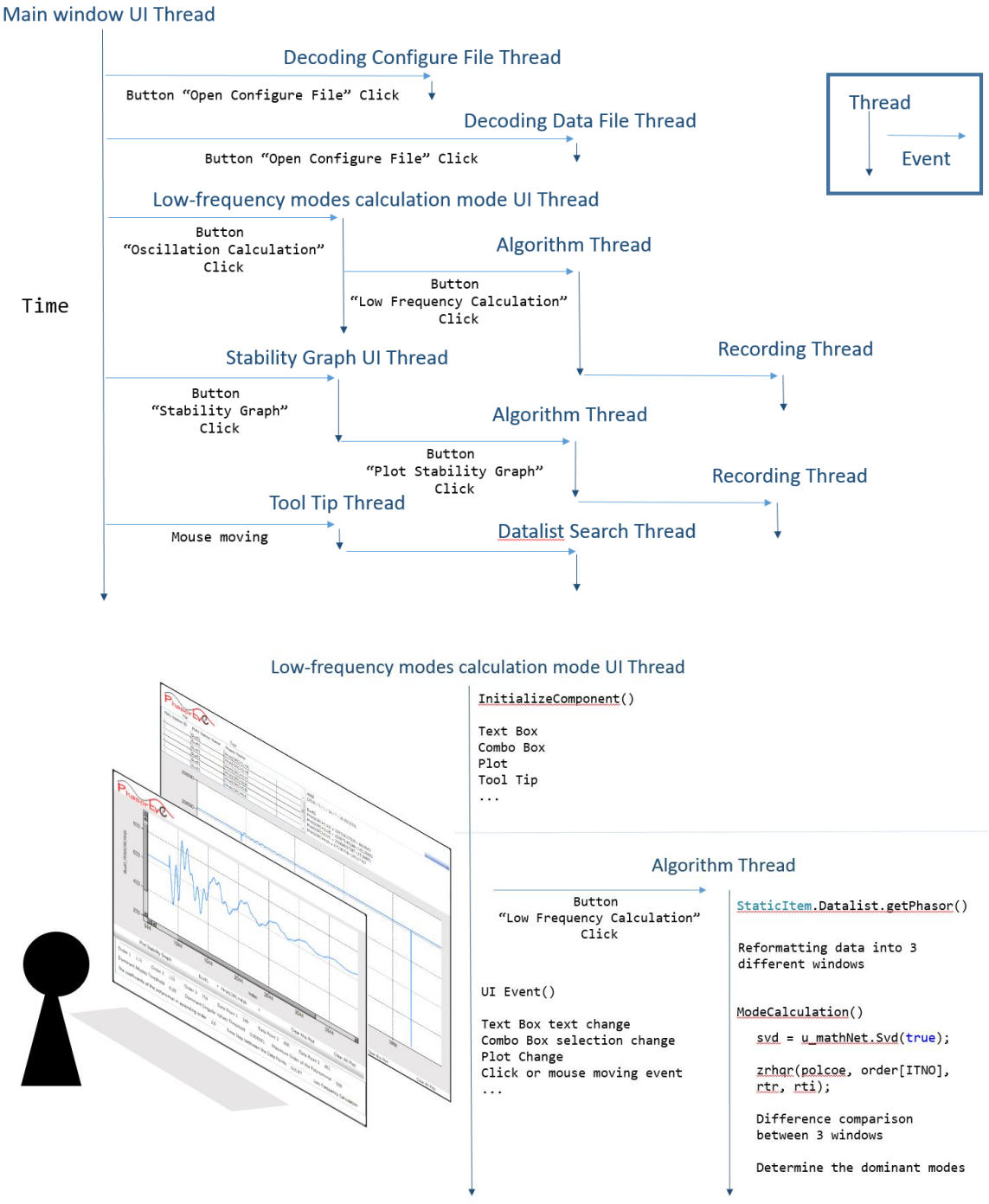


Figure 5.11 Thread diagram for monitoring of ring-down oscillations [42]

Figure 5.12 left is the diagnostic tools window with a single thread program for the above algorithm, and it shows processing time for 4990 time points with 4068 iterations is around three hours. The processing time for one iteration is around 2.65 seconds. Figure 5.12 right is the diagnostic tools window when algorithm is implemented inside "PhasorEye" application, and it shows processing time for same data set with the same number of iterations is decreased to around 20 minutes. Processing time for each iteration for a complete result shrinks to 0.294 seconds.

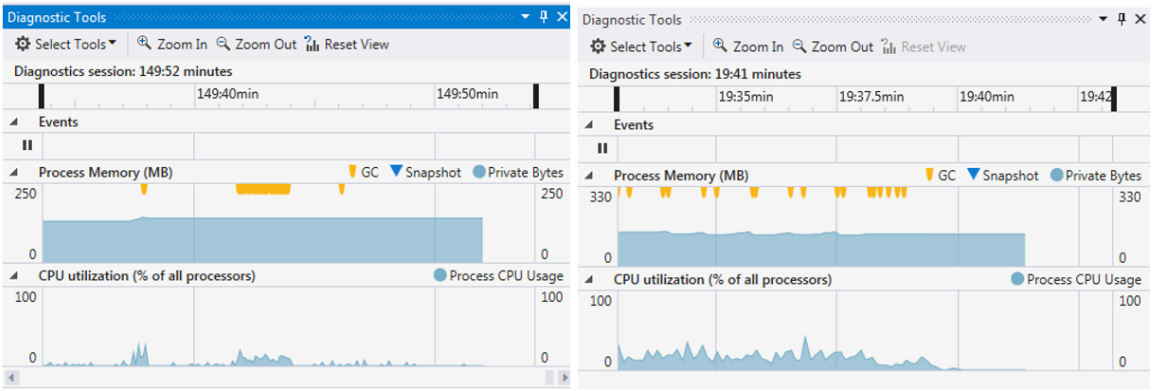


Figure 5.12 Performance for single thread implementation (left) and the multi-thread implementation (right)

This improvement gives only a very small jitter in real time low-frequency monitoring for "PhasorEye" program and offers a method implementing applications that require a massive amount of mathematical calculations. Figure 5.15 is the online testing result for low-frequency oscillation in IEEE 68 bus system [42] shown in Figure 5.13, low frequency 0.11Hz and 0.5Hz is observed. This result agrees with the offline analysis results shown in Figure 5.14, obtained with recorded data from the simulated system

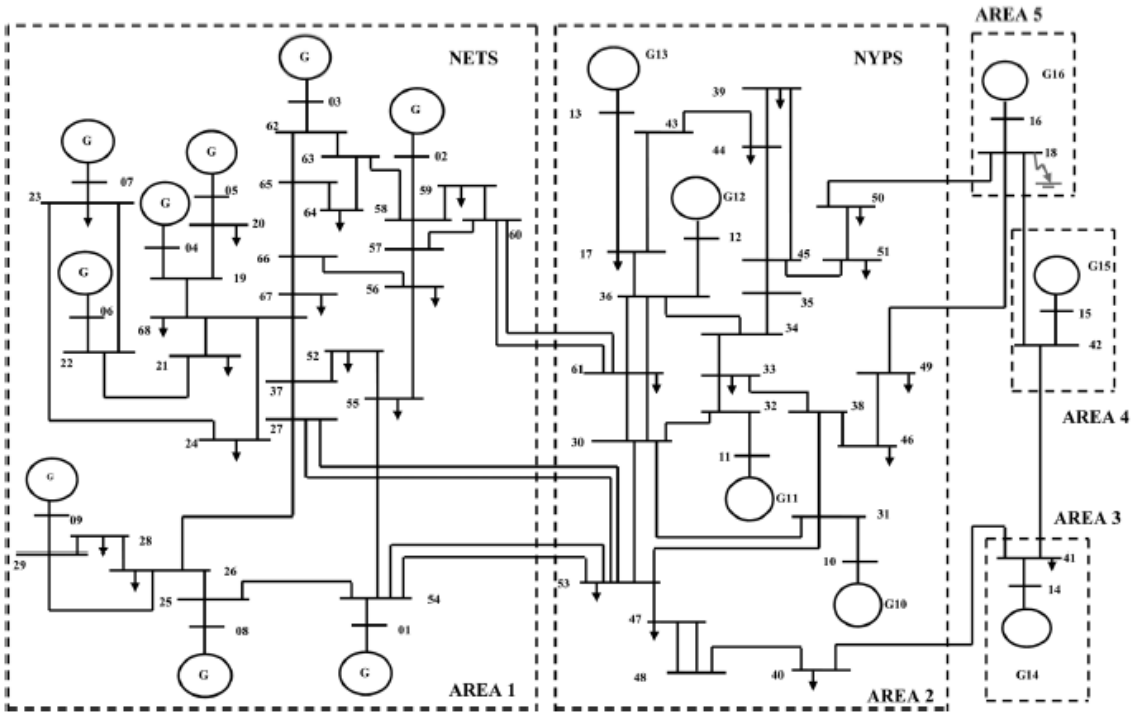


Figure 5.13 IEEE 68 bus system [42]

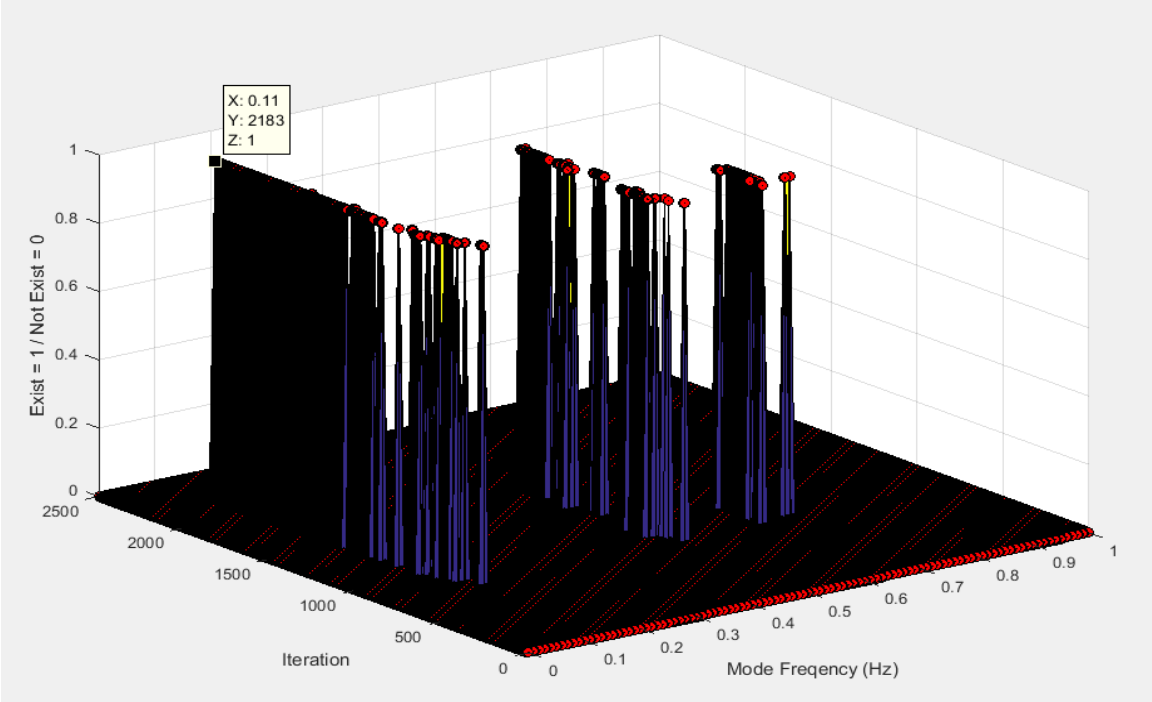


Figure 5.14 Low-frequency oscillations during ring-down oscillations for IEEE 68 bus system (Offline analysis)

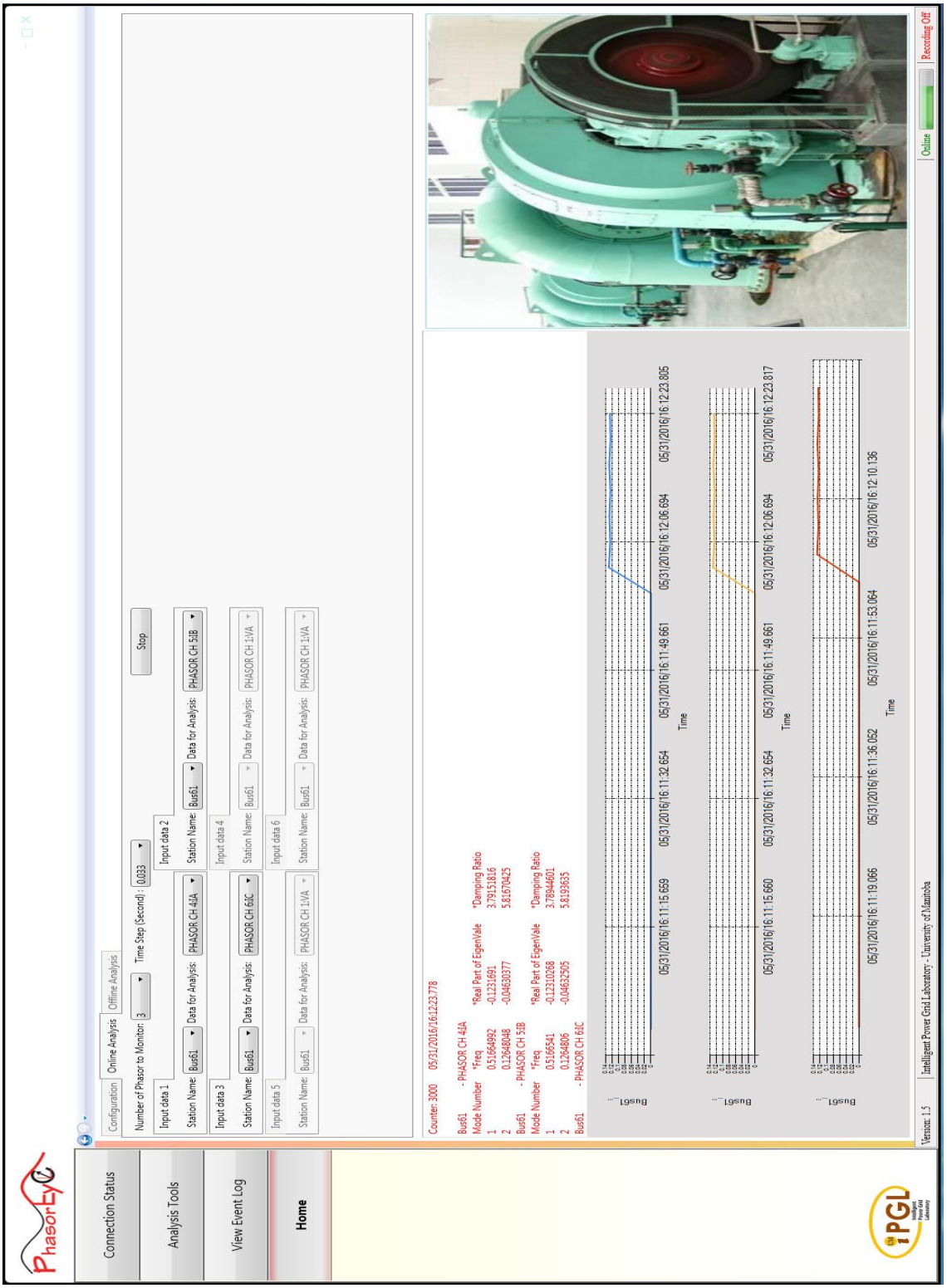


Figure 5.15 Low-frequency oscillations monitoring page during ring-down oscillations for IEEE 68 bus system (Online analysis)

5.1.4 Graphic console

Graphic console is another application developed in “PhasorEye” program. User can create a graphic console to access and observe the phasor measurements made by different PMUs. With the help from Microsoft Bing map library, the geographic information can be easily obtained for “PhasorEye” software as shown in Figure 5.16. The PMUs streaming data to “PhasorEye” are represented by PMU icons located at the top left hand side corner of the window. A user can drag an icon representing a particular PMU and place it at the correct geographical location on the map. In addition, a connection matrix is created, after the user specifies the connecting information between different PMUs as illustrated in Figure 5.17. This feature can be extended to generate an admittance matrix for power system, if the user can provide the branch information such as the transmission line parameters for the transmission lines between the buses with PMUs.

In conclusion, this application shows that external library can be easily imported into “PhasorEye” program. Many research or development platforms such as Matlab [43] have integrated methods to export functions (based on other programming languages) into components which can be used in a C# project. One example is the Matlab Component Object Model (COM) objects. The ability to integrate external library components increases the possibility of easily integrating new algorithms developed by different researchers to “PhasorEye” software. This is very important feature for using “PhasorEye” as a research tool.

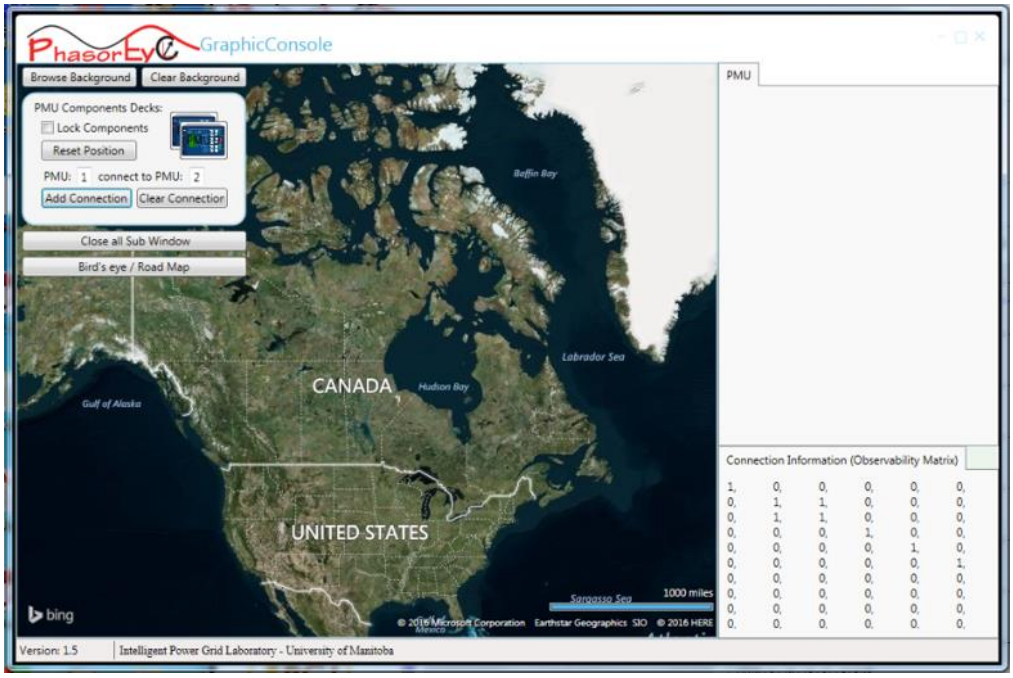


Figure 5.16 Graphic console window

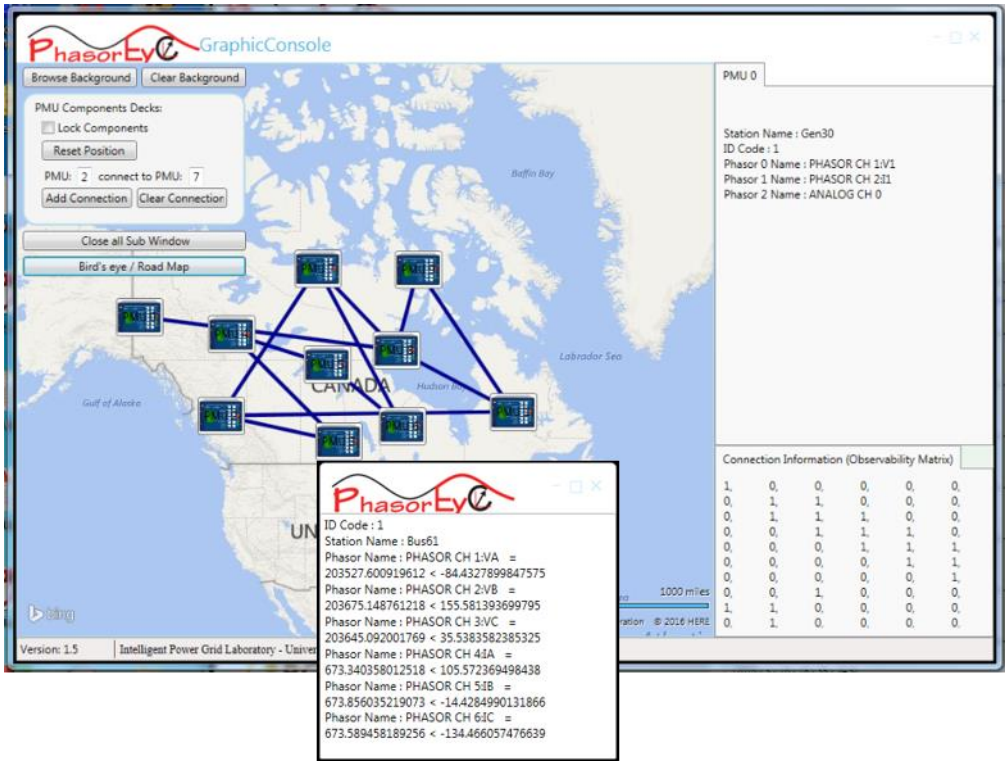


Figure 5.17 Graphic console window with the line connections and connection matrix

5.1.5 Data recording and replay

The last application to describe in this chapter is the recording and replay function in "PhasorEye" program. Figure 5.18 shows the procedure for the data recording. As a part of "PhasorEye" program, the recording feature has its own content page and operation thread. In the content page, the user can select the path for the recording data file, and size of the time window for the data file as illustrated in Figure 5.18. The recording file is a text file with the phasor measurements, and is recorded as hexadecimal numbers as shown in top right of Figure 5.18. The recorded data can be decoded by using the data frame decoding library of "PhasorEye" software.

The data player software shown in bottom right of Figure 5.18 can replay the recorded data. It is created by using the same software structure as "PhasorEye". However, instead of the page navigation graphical user interface used in "PhasorEye" program, the data player created a traditional window based application which is close to applications such as "PMU connection tester" software. Nevertheless, the decoding library, style component, "StaticItem" layer and other supporting functions or libraries and main structure used in the data player are the same as in "PhasorEye" software. The data player shows the portability of "PhasorEye" program's structure, the developer can use the same structure and modify the code to fit for deferent scenarios to create different real-time or offline applications.

Figure 5.18 Data recording page (left), the recorded data file (top right), "PhasorEye" data player (bottom right)

5.2 Chapter summary

In this chapter, five different synchrophasor application or tools, including three real-time monitoring applications are implemented on "PhasorEye" software platform. In addition, the advantages of "PhasorEye" program structure such as high code reusability, processing efficiency, easy importation of external libraries and the portability for expanded "PhasorEye" software are highlighted.

Chapter 6

Conclusions

6.1 Summary of work and conclusions

The main goal of this research is to build a software application for using the synchrophasors measured and communicated according to IEEE standard C37.118.1 and .2, with real time monitoring and offline analysis features. In order to achieve the main goal, following essential objectives are fulfilled during the research.

The first objective was to review the relevant literature and standards, and explorer the existing open source or commercial synchrophasor software packages. Both IEEE and IEC standards related to synchrophasor communications were reviewed. The fundamental structure of a synchrophasor network, the related devices, and the data exchange protocol within the synchrophasor technology were examined in detail. In addition, the open source software “PMU Connection Tester” and the commercial software SEL 5037 PDC software were examined to understand the essential features required for a synchrophasor application program. As a conclusion from the above review, Windows Presentation Foundation and an efficient thread pool were chosen as the best approach for implementing a multi-task synchrophasor application software.

The second objective was to use the chosen approach and IDE for creating the foundation of the multi-thread software "PhasorEye".

For this research, a 2-tier, component and layered-based hybrid architecture was used as the foundation of "PhasorEye" software. The main components of the program are the page, tab and multi-thread operation. A style component was developed for unifying the user interface makeup and enforce the system functions. A connection procedure with the decoding function was developed for capturing an output synchrophasor data stream from a PDC and decoding it for using in application functions. A clear four layered software structure was built for the "PhasorEye" software, facilitating easy understanding of the software by future users and developers.

In order to test the performance of the developed "PhasorEye" software with both online and offline applications, several different synchrophasor applications including line parameter monitoring, oscillation and damping monitoring, and transient stability monitoring algorithms were developed and integrated into "PhasorEye" software based on the algorithms developed by other researchers at the University of Manitoba. In addition, tools for synchrophasor data recording and offline reading were developed. All algorithms were tested for correct functioning using the synchrophasor data obtained from the power system models simulated in RTDS real time simulator.

A novel algorithm for locating faults in a multi-terminal, non-homogeneous, transmission line was developed. The proposed fault location algorithm is very general as long as the structure of transmission line consists of a main line and tapped branches, and functions without using any predetermined thresholds. Development of the fault location algorithm and the steps of implementation were presented in detail. The algorithm was implemented

and integrated into “PhasorEye” software. A laboratory setup that includes a power system simulated in a real time simulator and a practical synchrophasor network was assembled for testing of the algorithm. Issues related to practical application of the algorithm with practical synchrophasor data were analyzed. Best estimates are obtained with phasor measurements reported at least after 0.0333 seconds after the fault, when P-class PMU measurements are used. The measurements obtained from M-class PMUs may need slightly longer delay, and this need to be further investigated. The fault location algorithm performance under different conditions were tested and discussed. Fault location accuracies below 2% could be obtained under most conditions.

The multiple power system real time analysis tools implemented on “PhasorEye” software platform illustrates that the selected software structure has a decent scalability and the solid performance under the real time operation.

6.2 Recommendations for future research

In this thesis, the “PhasorEye” was developed and demonstrated with a few synchrophasor applications. However, more and more new protection and monitoring applications can be developed and integrated into “PhasorEye” in the future.

Currently, “PhasorEye” application is a WPF desktop application. However, WPF supports cross platform developing. With some proper changes, “PhasorEye” can be expanded as a mobile application for smart phones. In addition, cloud servers are becoming more and more popular in recent years due to the data processing ability. Thus, with the help from ASP service inside .NET framework, “PhasorEye” desktop application can be expanded to

an ASP web application using the cloud server. This will be a good choice for a utility level power network.

Bibliography

[1] Y. Liu, R. Fan and V. Terzija, “Power system restoration: a literature review from 2006 to 2016,” *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 3, pp 332–341, Jul, 2016.

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

[3] *IEEE Standard for Synchrophasor Data Transfer for Power Systems*, IEEE Standard C37.118.2-2011 (Revision of IEEE Standard C37.118-2005), 2011.

[4] *IEEE Standard for Synchrophasor Measurement for Power Systems*, IEEE Standard C37.118.1-2011 (Revision of IEEE Standard C37.118-2005), 2011.

[5] *IEEE Standard for Synchrophasors for Power Systems*, IEEE Standard 1344-1995, 1995.

[6] V. Terzija and Y. Liu, “Guest editorial: special issue on wide area monitoring, protection and control in smart grid,” *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 3, pp 317–318, Jul, 2016.

[7] S. M. Brahma, "Fault location scheme for a multi-terminal transmission line using synchronized voltage measurements," *IEEE Transactions on Power Delivery*, vol. 20, no. 2, pp. 1325-1331, Apr, 2005.

[8] J. Izykowski, E. Rosolowski, M. M. Saha, M. Fulczyk and P. Balcerek, "A Fault-Location Method for Application with Current Differential Relays of Three-Terminal Lines," *IEEE Transactions on Power Delivery*, vol. 22, no. 4, pp. 2099–2107, Oct, 2007.

[9] C. W. Liu, K. P. Lien and C. S. Chen, "A Universal Fault Location Algorithm for N-Terminal Transmission Lines," *IEEE Transactions on Power Delivery*, vol. 23, no. 4, pp. 1366–1373, 2008.

[10] Q. Jiang, B. Wang and X. Li, "An Efficient PMU-Based Fault-Location Algorithm for Multiterminal Transmission Lines," *IEEE Transactions on Power Delivery*, vol. 29, no. 4, pp. 1675–1682, 2014.

[11] T. Wu, C. Y. Chung, I. Kamwa, J. Y. Li and M. W. Qin, "Synchrophasor Measurement-Based Fault Location Algorithm for Multi-Terminal Multi-Section Non-Homogeneous Transmission Lines," *IET Generation, Transmission & Distribution*, vol. 10, no. 8, pp. 1815–1824, May, 2016.

[12] Unity3d. (2005). *Unity Game Engine* [Online]. Available: https://unity3d.com/ca/ [Oct. 10, 2016].

[13] Unrealengine. (1998). *What is Unreal Engine 4* [Online]. Available: https://www.unrealengine.com [Oct. 10, 2016].

[14] Visualstudio. (1997). *Free IDE and Tools | Visual Studio Community* [Online]. Available: https://www.visualstudio.com/vs/community/ [Oct. 10, 2016].

[15] A.G. Phadke, P. Wall and L. Ding. "Improving the Performance of Power System Protection using Wide Area Monitoring Systems," *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 3, pp 319–331, Jul, 2016.

[16] *IEEE Standard for Synchrophasors for Power Systems*, IEEE Standard C37.118-2005 (Revision of IEEE Standard 1344-1995), 2006.

[17] *Use of IEC 61850 to transmit synchrophasor information according to IEEE C37.118*, IEC TR 61850-90-5 draft edition, 2010.

[18] *IEEE Standard Profile for Use of IEEE 1588 Precision Time Protocol in Power System Applications*, IEEE Standard C37.238-2011, 2011.

[19] H. Falk, M. Adamiak, D. Baigent and V. Madani, "An Overview of the New IEC 61850 Synchrophasor Publish-Subscribe Profile", in *2013 66th Annual Conference for Protective Relay Engineers*, College Station, TX, USA, 2013 pp. 309-321.

[20] I. Ali, M. Aftab and S. M. S. Hussain, "Performance comparison of IEC 61850-90-5 and IEEE C37.118.2 based wide area PMU communication networks," *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 3, pp 487-495, Jul, 2016.

[21] R. E. A. Bonnin, "Development of a Laboratory Synchrophasor Network and an Application to Estimate Transmission Line Parameters in Real Time," M. S. thesis, Dept. Elect. Eng., Manitoba, Winnipeg, MB, 2014.

[22] Y. Cai, K. Jin, G. Ngantian and J. Wang, "Synchrophasor-Based Backup Differential Protection Design of Transmission Lines," B.S. capstone project, Dept. Elect. Eng., Manitoba, Winnipeg, MB, 2014.

[23] Grid Protection Alliance. (2006). *PMU Connection Tester* [Online]. Available: https://pmuconnectiontester.codeplex.com/stats, May. 22, 2014 [Oct. 10, 2016].

[24] Schweitzer Engineering Laboratories. (2011). *SynchroWAVe Phasor Data Concentrator (PDC) Software* [Online]. Available: https://selinc.com/products/5073/ [Oct. 10, 2016].

[25] GE Grid Solution. (2009). *Phasorpoint Services* [Online]. Available: http://www.gegridsolutions.com/alstomenergy/grid/products-services/product-catalogue/electrical-grid-new/control-room/wams/phasorpoint/phasorpoint-services/index.html [Oct. 10, 2016].

[26] US Department. (2007). *Synchrophasor Applications in Transmission Systems* [Online]. Available: https://www.smartgrid.gov/recovery_act/program_impacts/applications_synchrophasor_technology.html [Oct. 10, 2016].

[27] Grid Protection Alliance. (2009). *Open Source Software & Services for Electric Utilities* [Online]. Available: https://www.gridprotectionalliance.org/products.asp#PDC [Oct. 10, 2016].

[28] Mathworks. (2007). *Run MATLAB on multicore and multiprocessor machines* [Online]. Available: http://www.mathworks.com/discovery/matlab-multicore.html?requestedDomain=www.mathworks.com [Oct. 10, 2016].

[29] Mathworks. (2007). *MATLAB GUI – MATLAB* [Online]. Available: http://www.mathworks.com/discovery/matlab-gui.html [Oct. 10, 2016].

[30] Mathworks. (2007). *The Growth of MATLAB and the MathWorks over Two Decades* [Online]. Available: http://www.mathworks.com/company/newsletters/articles/the-growth-of-matlab-and-the-mathmathw-over-two-decades.html, Jan [Oct. 10, 2016].

[31] Gnu.org. (2005). *GSL - GNU Scientific Library* [Online]. Available: https://www.gnu.org/software/gsl/ [Oct. 10, 2016].

[32] Gnu.org. (2005). *GCC, the GNU Compiler Collection* [Online]. Available: https://gcc.gnu.org/ [Oct. 10, 2016].

[33] MSDN.microsoft. (2007). *C# Reference* [Online]. Available: https://msdn.microsoft.com/en-us/library/618ayhy6.aspx [Oct. 10, 2016].

[34] MSDN.microsoft. (2007). *Introduction to the C# Language and the .NET Framework* [Online]. Available: https://msdn.microsoft.com/en-CA/library/z1zx9t92.aspx [Oct. 10, 2016].

[35] MSDN.microsoft. (2007). *System.Numerics Namespace* [Online]. Available: https://msdn.microsoft.com/en-us/library/system.numerics(v=vs.111).aspx [Oct. 10, 2016].

[36] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. *Numerical Recipes in C*, 2nd ed., Cambridge, MA: Cambridge University Press, 1992.

[37] ILNumerics. (2013). *ILNumerics-Technical Computing* [Online]. Available: https://ilnumerics.net/ [Oct. 10, 2016].

[38] Math.Net. (2007). *Math .NET* [Online]. Available: http://www.mathdot-net.com/ [Oct. 10, 2016].

[39] J. L. Blackburn, *Symmetrical Components for Power System Engineering*. New York, NY: Marcel Dekker, 1993.

[40] RTDS.Inc. (2013). *RTDS GTNET Card* [Online]. Available: https://www.rtds.com/the-simulator/our-hardware/gtnet-card/ [Oct. 10, 2016].

[41] D. R. Gurusinghe and A. D. Rajapakse. “Post-Disturbance Transient Stability Status Prediction using Synchrophasor Measurements”, *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3656 - 3664, 2016.

[42] D. P. Wadduwage, U. D. Annakkage and K. Narendra, "Identification of Dominant Low-Frequency Modes in Ring-Down Oscillations using Multiple Prony Models". *IET Generation, Transmission & Distribution*, vol. 9, pp. 2206-2214, Nov. 2015.

[43] Math.Net. (2007). *Call MATLAB Function from C# Client* [Online]. Available: https://www.mathworks.com/help/matlab/matlab_external/call-matlab-function-from-c-client.html [Oct. 10, 2016].

[44] H. Saadat. *Power System Analysis*, 3rd ed, United States of America: PSA Publishing, 2010.

[45] A. T. Johns and S. Jamali, "Accurate fault location technique for power transmission lines," *IEE Proceedings C - Generation, Transmission and Distribution*, vol. 137, no. 6, pp. 395-402, Nov 1990.

[46] D. R. Gurusinghe and A. D. Rajapakse. “Efficient Algorithms for Real-Time Monitoring of Transmission Line Parameters and Their Performance with Practical Synchrophasors,” Accepted for publication in *IET Generation, Transmission & Distribution* in Dec 2016.