UNIVERSITY OF VAASA
FACULTY OF TECHNOLOGY
COMMUNICATIONS AND SYSTEMS ENGINEERING

Damilola Simeon Adesina

INTERFACING IEC 61850–9–2 PROCESS BUS DATA TO A SIMULATION ENVIRONMENT

Master's thesis for the degree of Master of Science in Technology submitted for inspection, Vaasa, 28 October, 2015.

Supervisor Mohammed Salem Elmustrati

Instructors Kimmo Kauhaniemi

Reino Virrankoski

ACKNOWLEDGMENT

I am most grateful to the Almighty God who is the source of life, wisdom and knowledge for the successful completion of my Master's degree program. My appreciation goes to my supervisor – Professor Mohammed Elmustrati, The head of the Sundom Smart Grid research project and instructor – Professor Kimmo Kauhaniemi and all members of the project team. I appreciate the enabling environment provided by Reino Virrankoski and the guidance by Mike Mekkenen throughout the thesis period. I also thank Professor Timo Mantere, Tobias Glocker and Ruifeng Duan for their contribution to knowledge during the course of my study. I am indebted to Veli–Matti Eskonen and Juha Miettinen for their assistance during the period of my thesis.

To my wife and friend of many years; Olutobi Adesina, I say a big thank you for the love, understanding and encouragement at all times. Special thanks to my parents, Professor and Mrs. S.K. Adesina for their prayers and continued support. I am also thankful to my siblings; Tope, Simeon, Mayowa and Jumoke and their families for the love we share and the encouragement throughout my studies. I appreciate my friends here in Vaasa for the support system you have become to me.

I am really grateful to the Finnish government, the University of Vaasa and indeed the Communication and Systems Engineering Group for the enabling environment to learn and apply my knowledge to solving real life problem.

Vaasa, Finland, 26 October 2015

Damilola Simeon Adesina

TABLE OF CONTENT PAGE

TITLE PAGE

ACKNOWLEDGMENT ……………………………………………………………..…... 2

TABLE OF CONTENT …………………………………………………………..……. 3

ABBREVIATIONS …………………………………………………………….……..... 5

LIST OF FIGURES …………………………………………………………….……. 7

LIST OF TABLES ……………………………………………………………………….. 9

ABSTRACT ……………………………………………………………………….. 10

1. INTRODUCTION

1.1. The Smart Grid concept ……………………..…….…….…………………… 12

1.2. Objective of the Thesis ……………………….…….…….…………………….. 13

1.3. Motivation for the Thesis ………………………..…….…….………….……. 13

1.4. Scope of the Thesis ………………………………….….…………………….… 14

1.5. Structure of the Thesis ……………………………..….….……….……….. 15

1.6. Related Work ……………………………………..……………………….. 16

2. THEORY AND BACKGROUND INFORMATION

2.1. Communication Solution for the Smart Grid …………..…………….……… 18

2.1.1. Open System Interconnection ……………………………….………. 19

2.1.2. TCP/IP Protocol Suite ……………………………………………..... 20

2.1.3. Ethernet Protocol ……………………………………………………. 22

2.1.4. Internet Protocol …………………………………………………….. 24

2.1.5. User Datagram Protocol ……………………………………………….. 25

2.1.6. Transmission Control Protocol …………………………….………….... 26

2.1.7. Manufacturing Message Specification ……………………………..…. 27

2.2. Substation Automation System ……………………………………….…... 28

2.2.1. Substation Automation System Architecture ………………………….. 29

3. COMMUNICATION AND AUTOMATION STANDARD – IEC 61850

3.1. IEC 61850 Parts ………………………………………………..…………….. 31
3.2. IEC 61850 Application and Communication views ……………...………….. 33
3.2.1. IEC 61850 Application view and Data Model ……………..…..……... 36
3.2.2. IEC 61850 Communication / Information Exchange Model ……….…...38
3.2.3. IEC 61850 Substation Configuration Language …….…………..……. 42
3.3. IEC 61850-9-2 Process Bus and benefits …………………………….…… 43
3.4. Sampled Value Communications …………………………….………….... 45
3.4.1. Sampled Value packet ……………………………...……………….. 45

4. DATA EXTRACTION AND FORMATTING

4.1. How to compile PCAP files ……………….…………….……………...… 51
4.2. Data sources …………………………………..………………………… 55
4.3. Computer program to convert data to suitable format for PSCAD……….….... 58
4.4. Result and analysis ……………..……………………………………… 62
4.4.1. Extracted data and sample measurement plot ……………………..… 62
4.4.2. Software validation ………………………………………….……. 64

5. CONCLUSION AND FUTURE WORK ……………………………….…. 68

REFERENCES ……………………………………………………………….. 69

APPENDICIES
APPENDIX 1. United States Energy Independence and Security Act …..…..…. 74
APPENDIX 2. File read and Data file format in PSCAD……………...……... 76
APPENDIX 3. Tools used to capture and analyze Sampled Value packet………. 78
APPENDIX 4. The computer program ……………………………….…….. 85

ABBREVIATIONS

ACSI	Abstract Communication Service Interface
AMI	Advanced Metering Infrastructure
ADC	Analog–to–Digital Conversion
API	Application Programming Interface
ARFF	Attribute –Relation File Format
CID	Configured IED Description
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CRC	Cyclic Redundancy Check
CTs	Current Transformers
DO	Data Objects
DAS	Data Acquisition System
DR	Demand Response
DSP	Digital Signal Processing
DG	Distributed Generation
DA	Distribution Automation
DMS	Distribution Management System
EMTDC	Electromagnetic Transient simulation software
XML	Extensible Markup Language
FACTS	Flexible Alternating Current Transmission System
GOOSE	Generic Object Oriented Substation Event
GSSE	Generic Substation Status Event
GUI	Graphical user interface
HVDC	High Voltage Direct Current
HMI	Human Machine Interface
IEEE	Institute of Electrical and Electronics Engineers
IEDs	Intelligent Electric Devices
IEC	International Electrotechnical Commission
ICD	IED Capability Description
ISO	International Standard Organization
ISR	Interrupt Service Routine

LN	Logical node
MAC	Medium Access Control
MMS	Manufacturing Messaging Specification
MSB	Most Significant Bit
MU	Merging Unit
NPF	Net group Packet Filter
NIC	Network Interface Card
NCIT	Non – Conventional Instrument Transformer
OSI	Open System Interconnection
OS	Operating System
PCAP	Packet CAPture
PSCAD	Power System Computer Aided Design
PDUs	Protocol data units
RTU	Remote Terminal Units
SCSM	Specific Communication Service Mappings
SFD	Start Frame Delimiter
SCL	Subscription Communication Language
SA	Substation Automation
SAS	Substation Automation System
SCD	Substation Configuration Description
SSD	System Specification Description
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
UCA	Utility Communication Architecture
VLAN	Virtual Local Area Network
VMD	Virtual Manufacturing Device
VT	Voltage Transformer
WAN	Wide Area Network

LIST OF FIGURES PAGE

Figure 1. Structure of the Thesis …………………………………………….……….. 15

Figure 2. The structure of a packet …………………………………………………... 18

Figure 3. The OSI Model ……………………………………………..………… 19

Figure 4. Flow of data within the OSI Layers…………………………….……….…. 20

Figure 5. Layers in TCP/IP Protocol Suite ……………………………………….... 21

Figure 6. Relationship between OSI Model and TCP/IP protocol suite …………….….. 22

Figure 7. Ethernet frame protocol stack …………………………….……….…. 23

Figure 8. The Ethernet frame …………………………………………………....…... 24

Figure 9. IP datagram ……………………………………………………..……..... 25

Figure 10. IP datagram Header format …………………………………..…... 25

Figure 11. UDP User datagram ……………………………………………………… 26

Figure 12. UDP User datagram header format ……………………………….…… 26

Figure 13. A segment ……………………………………………...…...…..…… 27

Figure 14. The header of a segment …………………………………………….…. 27

Figure 15. MMS Client/Server Model showing Virtual Manufacturing Device …...... 28

Figure 16. Substation architecture based IEC 61850 standard ………………………. 30

Figure 17. Application and communication views of IEC 61850 …………………… 34

Figure 18. IEC 61850 view from application to communication ……………………. 35

Figure 19. Substation and virtual model ……………………………….………. 36

Figure 20. Elements in the data model defined by IEC 61850 …….………….…... 37

Figure 21. The communication profiles defined in IEC 61850 ………………….…... 39

Figure 22. Client/Server communication …………………………….……….…… 40

Figure 23. Publisher–subscriber communication ………………………...…...…... 40

Figure 24. Conceptual substation engineering process using SCL ……………...…… 43

Figure 25. IEC 61850-9-2 Process Bus concept …………………………………... 44

Figure 26. Frame format of IEC 61850-9-2 in ISO/IEC 8802-3 …………………….. 46

Figure 27. Structure of modelled SV APDU/ASDU (ASN.1/BER TLV triplets) …… 46

Figure 28. Encapsulation of SV APDU as it goes down the protocol stack ...……….. 47

Figure 29. An example of frame format for one current sampled Value …..………... 49

Figure 30. Add supplementary include directories …………………..……………. 52

Figure 31. Add Preprocessor Definitions ……….…………………...…..………... 53
Figure 32. Add additional Library directories ……………..………..…….……….. 54
Figure 33. Add additional dependencies ……………….……………….…………… 55
Figure 34. Outputs from Wireshark capture showing various sources ……………..... 57
Figure 35. The output from source with MAC address 00:21:c1:25:de:b8 ………….. 58
Figure 36. Sequence of operation of the computer program ………………………… 59
Figure 37. Data points identified on Wireshark capture ……………………………... 61
Figure 38. Real data values written to output file ……………………………………. 63
Figure 39. Plot of real current measurement ………………………………….….. 63
Figure 40. Plot real voltage measurement ……...…… 64
Figure 41. Waveform of Sampled Value generated from Omicron CMC 850 ……... 65
Figure 42. Output obtained by extracting and formatting data generated by the Omicron CMC 850 device ………………………………………………………...………. 65
Figure 43. Current waveform of the output data from Omicron CMC850 …….……. 66
Figure 44. Voltage waveform of the output data from Omicron CMC850 …….…… 66

LIST OF TABLES PAGE

Table 1. Parts of IEC 61850 standard documents …………………………………….... 32

Table 2. Data sources and MAC addresses …………………………………...…… 56

Table 3. Bytes representing currents and voltages in frame 5 of **Figure 35** ………… 60

UNIVERSITY OF VAASA
Faculty of technology

Author:	Damilola Simeon Adesina
Topic of the Thesis:	Interfacing IEC 61850–9–2 Process Bus Data to a Simulation Environment
Supervisor:	Mohammed Salem Elmustrati
Instructors:	Kimmo Kauhaniemi Reino Virrankoski
Degree:	Master of Science in Technology
Degree Programme:	Degree Programme in Telecommunication Engineering
Major of Subject:	Telecommunication Engineering
Year of Entering the University:	2013
Year of Completing the Thesis:	2015 **Pages:** 90

ABSTRACT

IEC 61850 – Communication and networks in substations is the standard for building communication infrastructure between the different Intelligent Electronic devices (IEDs) in the substation automation system. It consists of several parts which include Specific Communication and Service Mapping for the transmission of sampled values (defined in part 9–2 of the standard). The Sampled value communication is a high speed, time critical Ethernet based communication for the transfer of data over the network. It defines the sampling rate and time synchronization requirement of the system.

The main purpose of this thesis is to extract sampled value data (four voltages, four currents) from a PCAP data file captured over the network in the ‘Sundom Smart Grid’ environment and convert the data into the format needed for analysis on PSCAD simulation tool. This thesis serves as an interface between the real Smart Grid environment and the test environment in the University of Vaasa.

This thesis explains fundamental concepts that relate to IEC 61850, and the Sampled Value in particular. It describes the frame structure of sampled value and a software application has been developed based on WinPcap Application Program Interface (API) to extract the data points needed and fulfill the data format requirement of the PSCAD which is adaptable for use in MATLAB.

KEYWORDS: IEC 61850–9–2, Sampled Values, WinPcap, Wireshark, PSCAD

1. INTRODUCTION

The recent advances in information and communication technology have brought about great development in the way we live. The basic communication need is no more limited to humans but also involves machine–to–machine as well as human–to–machine communication. This has pushed the boundaries from just wired or cellular communication to data communication which makes available a huge volume of data to be processed. This enormous amount of data has triggered the problem of “information overload” (Fowler & Hammell II 2014: 1) which suggests that there is large amount of data in different sizes, probably different formats and one has to sift through a large amount of information given proper consideration to other factors such as time.

The development and application of information and communication technology in the electric power grid has changed the way operations are carried out and thus, has become the enabler to achieving the smart grid concept. The smart grid is a system with an enormous amount of data because its objectives warrant the interplay of many devices and a fully automated system. A key section of the electric power system is the Substation. Communication and networking in this section has been standardized in IEC 61850 – “Communication networks and systems in substations”. This makes it possible for the substation devices to communicate seamlessly and has provided a means of understanding and interpreting the data captured over the network.

There are standard tools that can capture the traffic on a network but these tools generate huge amount of data within minutes which becomes unusable unless it is properly processed so that follow–on devices or tools can deduce, analyze and interpret it. The Wireshark is one of such tools; free and readily available for data capture in networks. It is built on libPCAP/WinPcap application program interface (API) and is been used by many because it accurately captures the traffic moving through a network. Other capture tools include Microsoft Network Monitor, Snort, and Ettercap.

1.1. The Smart Grid concept

The term “Smart Grid” is envisioned to be an electric power system which is intelligent, more secure, and more reliable. According to the United States Department of Energy, the Smart Grid is expected to be a fully automated system with bidirectional flow of information and electricity which are key features in ensuring real-time management of the grid. It has distributed intelligence, automated control systems and broadband communications which facilitate real-time exchange of data and seamless interfaces among all the units involved in the grid such as buildings, generation facilities, and the electric network. (United States Department of Energy *GRID 2030*, 2003: 17). The smart grid include components such as Advanced Metering Infrastructure (AMI), Distribution Automation (DA), Distributed Generation (DG), Substation Automation (SA), Flexible Alternating Current Transmission System (FACTS) and Demand Response (DR).

> According to Andres Carvallo, The smart grid is the integration of an electric grid, a communication network, software and hardware to monitor, control and manage the creation, distribution, storage and consumption of energy. The smart grid of the future will be distributed, it will be interactive, will be self-healing, and will communicate with every device. (Carvallo & Cooper 2011:1
>
> Also, Fereidoon defined the smart grid as any combination of enabling technologies, hardware, software, or practices that collectively make the delivery infrastructure or grid more reliable, more versatile, more secure, more accommodating, more resilient, and ultimately more useful to consumers. (Fereidoon 2011: xxix).

The concept of the smart grid does not have a particular definition; in fact the previous definitions have tried to define it from a very broad perspective to encompass what it means. Therefore, it is best suited to describe the smart grid in terms of its objectives. These objectives have been listed by the United States Energy Independence and Security Act (EISA) of 2007 (see Appendix 1). (Budka, Deshpande & Thottan 2014:4).

The scope of the Smart Grid development is wide-ranging; this includes the deployment of renewable energy sources, automated demand response, peak power reduction, increased energy efficiency, and consumer participation in energy management. This development is expected to affect every section of the power grid which has seen little changes since its inception, and will also include modernization of the components of the grid as well as the introduction of new monitoring and control technologies to achieve automation, metering, fault recording and reporting.

An important development needed to achieve the smart grid goal is in the communication network. This is an integrated network that allows exchange of measurements and control data from all applications (home, substations, and generation centers) in real–time and fulfills the demand for performance, reliability and security.

1.2. Objective of the Thesis

This thesis bridges the gap between the Smart Grid environment and the test environment at the University of Vaasa. It seeks to integrate the data from the Wireshark Packet CAPture (PCAP) file to the test environment by extracting the part of the data fields required and convert it in the format acceptable by the follow–on device or tool. See **Appendix 3** for details of Wireshark

More specifically, the thesis aims at interfacing the data from the Smart Grid environment to Power System Computer Aided Design (PSCAD) simulation tool. Refer to introduction of chapter 4 for a discussion on PSCAD.

1.3. Motivation for the Thesis

A PCAP file usually has numerous fields/value and Wireshark/tshark have standard data exporting formats which might not necessarily be in the format needed by follow–on tools, devices for analysis or it may deliver only few of the numerous potential field or data value. The output module is usually a percentage of the fields and their values. Thus, it becomes necessary to develop a custom tool to extract the particular fields and values needed from the PCAP file and adapt it to a desired format. The needs to be able to select different field’s data for analysis, and have access to perform tests on different platforms are the key factors for undergoing this thesis. The computer program and different interfacing option represents a part of the Sundom Smart Grid project.

The Sundom Smart Grid project is a pilot project in Finland with the objective of improving electricity supply and instituting the essential requirements for solar and wind power usage in homes in the Vaasa region. The development of a Smart Grid research and demonstrating platform are the core of the task which involves the collaborative effort of Anvia; Information and Communication Technology Company, ABB, University of Vaasa, electricity retail company – Vaasan Sähkö and the distribution network operator – Vaasan Sähköverkko.

In the power network which comprises of both overhead lines and underground cables, Anvia provides a means of transferring the digital measurement data within the network in real time; ABB tests the latest automatic fault management system while the University of Vaasa studies the effects of the underground cables and network automation (Eero Lukin 2014; European Union Smart Cities 2015) in addition to the integration of available data to the smart grid research environment at the University of Vaasa which is the topic of this thesis.

1.4. Scope of the Thesis

The thesis work involves a study on IEC 61850 Standard especially IEC 61850–9–2; ‘Specific Communication Service Mapping–Sampled values over ISO/IEC 8802–3’. It further develops a computer program to extract specific data fields from a PCAP file captured by Wireshark over a network for importing into the PSCAD simulation tool in the standard format acceptable by the tool.

1.5. Structure of the Thesis

The thesis is composed of five chapters. Chapter one explains the concept of smart grid, the motivation, scope as well as structure of the thesis. It further gives an overview of related work that has been done in this area of study. Chapter two focuses on theoretical background details and explanation of concepts that are important in understanding the thesis such as the communication solutions for the Smart Grid and substation automation system.

Also, the details of IEC 61850 Standard and structure of the Sample Value packet is presented in the chapter three while the details of the methodology/ implementation of the thesis and how the data integration is achieved are discussed in chapter four. It further presents the results. The conclusion of the work and suggestion on future work are presented in chapter five. The summary of the structure is as shown in **Figure 1.**

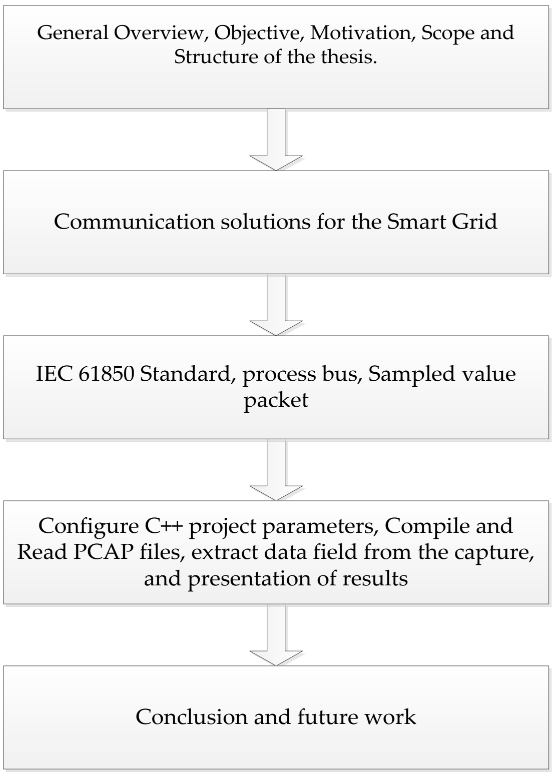


Figure 1. Structure of the Thesis.

1.6. Related Work

With the increase in the use of process bus for communication in substations, there is an attendant improvement in the research on process bus and sampled value communications. However, information about the real life implementation of the process bus is still in its infancy, given that it was introduced recently. Most of the work reported so far is based on computer simulations to generate data based on IEC 61850–9–2, test such data in simulation environments and evaluate process bus in the laboratory.

From this view, Baranov et al. developed a software for emulating the Sampled Values transmission in accordance with IEC 61850–9–2 Standard. The virtual Instrument, "IED Emulator" developed by LABVIEW makes it possible to configure the sample value transmission parameters on it for transmission. Also in the work of Konka et al. a model of SV traffic generator was developed. Similar work has been done by Liang and Campbell as well as Kanabar et al.

Further to the sampled value generation via simulations, research has been dedicated to connecting the design of power systems and communication networks to permit the study of the system as an integrated entity, thereby reducing the network effects such as loss of data, delay. Progress in this area is reported in the work of Lin et al. – "Power System and Communication Network Co-simulation for Smart Grid Applications ", Nutaro et al. – "Integrated modeling of the Electric Grid, Communications, and Control", and Ragnamay–Naeini et al. – Impacts of Control and Communication System Vulnerabilities on Power Systems Under Contingencies."

There is attendant increase in research to optimize data transmission, formatting, and analysis. A search for literature shows that there were no readily available tools to extract the fields needed from a PCAP file to be used on the PSCAD thus creating a need to develop such tool.

A closely related work is that done by Charles A Fowler and Robert J. Hammell II. They developed a tool that converts PCAPs into Attribute –Relation File Format (ARFF) that is a Weka minable data. Other tools that have been developed in this category are tcp2d and fullstats.

The novelty of this thesis is in that the data in the PCAP file is from a real life scenario of an electric substation, and the developed software can extract the specific fields of information needed for analysis in electric power system models developed in the PSCAD.

2. THEORY AND BACKGROUND INFORMATION

2.1. Communication solutions for the Smart Grid

A communication network is necessary to transmit information, data as well as enable other communication services between endpoints connected to it. These services can be connection–oriented; a link is first established and networking resources are used only for the exchange of data between the two endpoints for the period of time that the connection exists or connectionless; an endpoint is permitted to send out data to the other endpoint without first establishing a link. There are overheads associated with both the connection–oriented and connectionless services. These overheads which accompany any unit of data transfer vary based on the type of connection and can be those associated with connection, disconnection, addressing in connection–oriented services and addressing in connectionless services. Such data units are called *protocol data units* (PDUs). The PDU sent between endpoints over a communication network is called a *Packet* and it is made up of overhead information (in the Packet Header and Packet Trailer) and the real data (known as *payload*). The structure of a packet is shown in **Figure 2.** (Budka et al. 2014: 47–55)

Figure 2. Structure of a Packet (Budka et al. 2014: 51).

The rules that govern the exchange of data and communication between devices, systems or networks are collectively called *protocol*. Such rules include header and trailer format, size of packet, data delivery assurance and error detection. The following sub–sections briefly describe core communications solutions, communication protocols and protocol layers that are important to IEC 61850 and the Smart Grid.

2.1.1. Open System Interconnection (OSI)

The OSI model covers all aspect of network communication. It presents a set of protocols that permits various systems to interconnect irrespective of their underlying design.

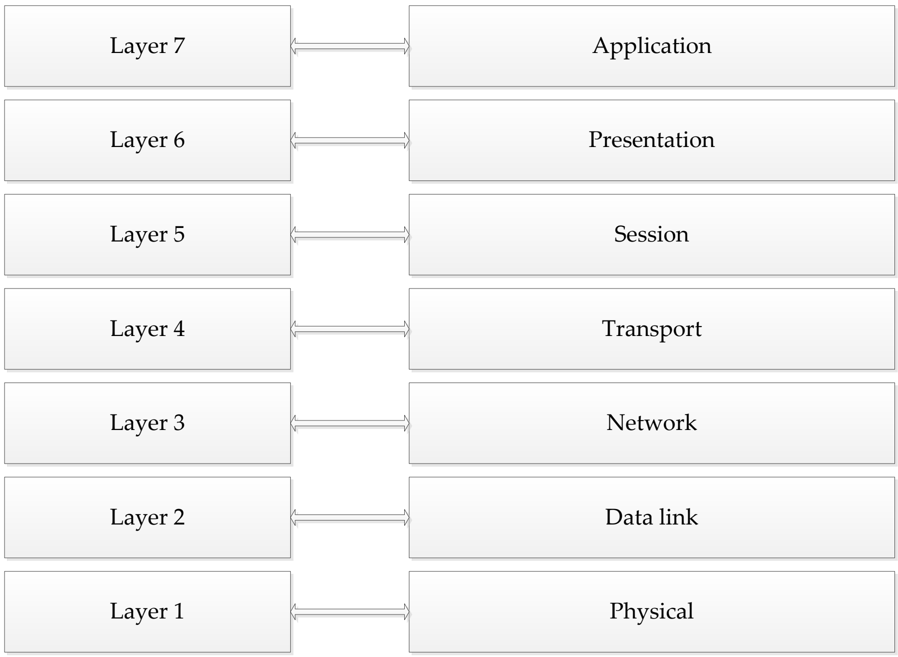


Figure 3. The OSI Model.

The OSI model is made up of seven different but related layers as seen in **Figure 3.** It makes communication between various systems possible without the need to alter the logic of the original software and hardware of the system. In the design of the model, the data transmission process is broken down to its most basic elements. The connectivity function that have related uses are grouped together to form the layers; each layer defining a group of functions different from those of the other layers (Forouzan 2010: 20–23). The flow of data from one device, network or system to another is also defined by the OSI Model. **Figure 4** shows the flow of data between two devices from the application layer to the physical layer through well-defined interfaces between the layers and from the physical layer to the application layer.

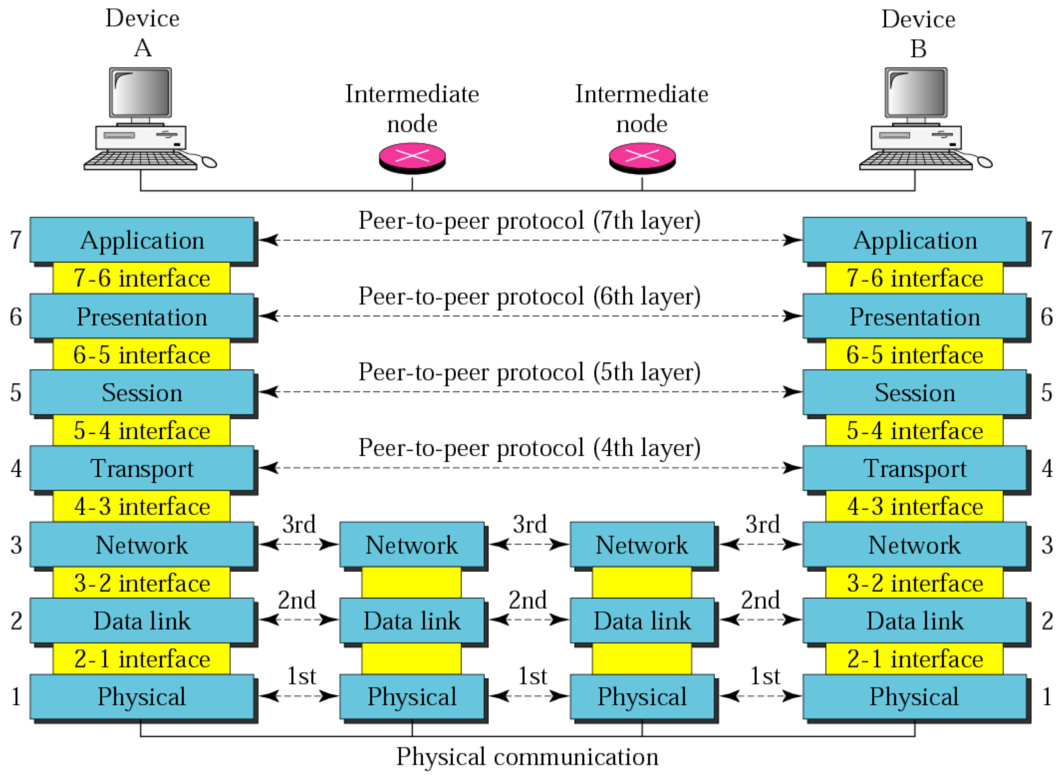


Figure 4. Flow of data within the OSI Layers (Forouzan 2010: 22).

2.1.2. TCP/IP Protocol Suite

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite consists of four layers as represented in **Figure 5**. It is considered a very significant protocol suite because IP in layer 2 makes addressing and routing of datagram possible while layer 3 TCP ensures reliable transmission of data. It is hierarchical in design and composed of modules which are interactive, carry out definite functions, but not necessarily autonomous (Forouzan 2010: 30). The layers are made up of fairly independent protocols which can be combined and harmonized based on the requirements of the system.

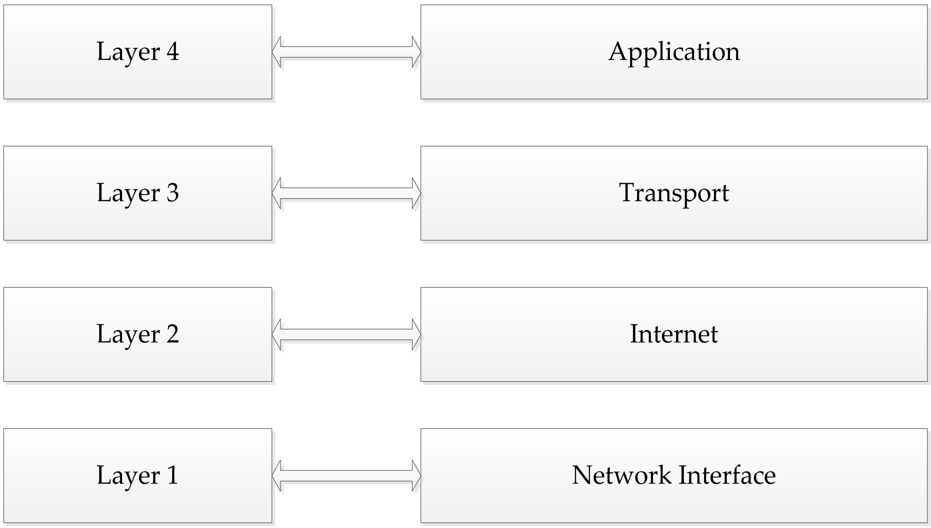


Figure 5. Layers in TCP/IP Protocol Suite.

The application layer is designed to allow a user to access the communication services. Numerous protocols have been developed for services such as file transfer, accessing the internet and e–mail. The details of the services provided by the Internet layer; IP are given in section 2.1.4 while transport layer services; UDP and TCP are discussed in sections 2.1.5 and 2.1.6 respectively. The network interface layer is responsible for features of packet transfer that deals with the network. It provides several interfaces for connecting computer end systems to networks such as Ethernet used in Sampled Values and discussed in section 2.1.3, token ring and frame relay (Leon–Garcia & Widjaja 2000: 57–58).

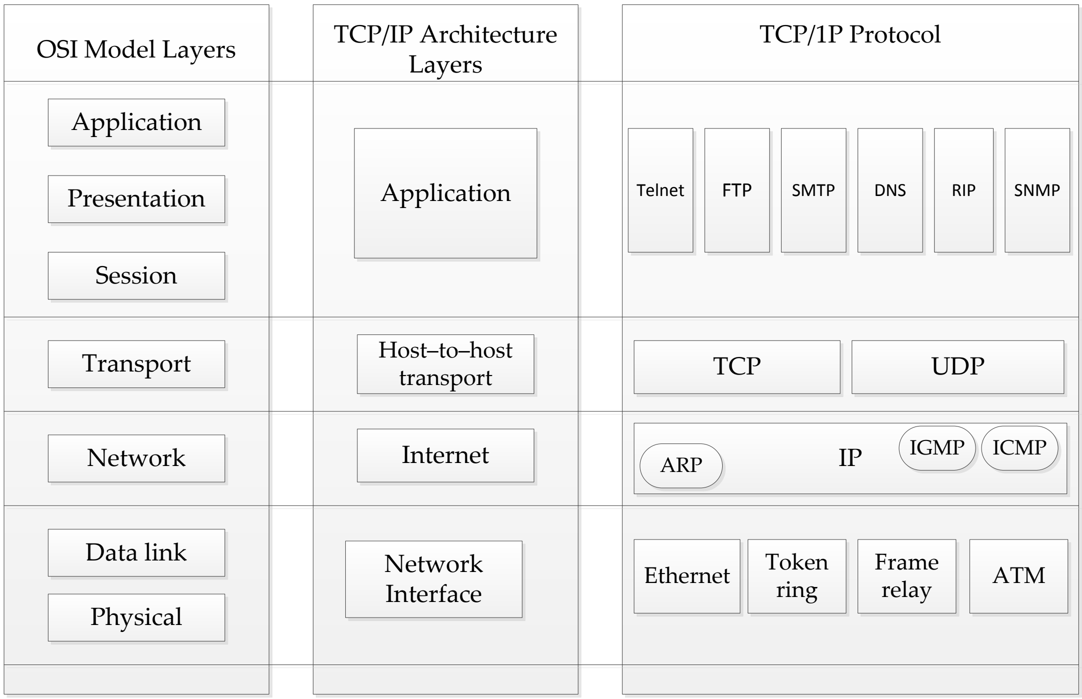


Figure 6. Relationship between OSI Model and TCP/IP protocol suite.

Comparing the OSI model and the TCP/IP Suite layers as illustrated in **Figure 6**, the application layer of TCP/IP combines the tasks of the application, presentation and session layers of the OSI model. The transport layer maps each other while the network layer of OSI model is the internet layer in TCP/IP Suite. Network interface layer combines the functions of the data link layer and the physical layer of the OSI model.

2.1.3. Ethernet Protocol

The Ethernet is a part of the project 802 by the Institute of Electrical and Electronics Engineers (IEEE) Computer Society to formalize standards to facilitate intercommunication between devices from various manufacturers. It is a data link layer protocol described in IEEE 802.3 standard with Carrier Sense Multiple Access/Collision Detection (CSMA/CD) as the access method.

CSMA/CD access method describes a process whereby a station that wants to transmit constantly checks the network cable to know if any other station in the network is transmitting data on the cable. This process is known as *carrier sensing*. Stations on the network send data only when it does not detect a signal on the cable. Every station on the network receives the transmission (*multiple access*) but only the station whose Medium Access Control (MAC) address corresponds to the destination address in the frame header keeps the frame for further processing (Budka et al. 2014: 70; Forouzan 2010: 47). **Figure 7** shows the protocol stack of the Ethernet frame.

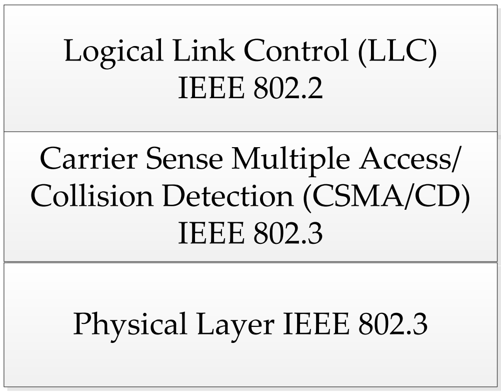


Figure 7. Ethernet frame protocol stack.

The packets sent over the Ethernet network are called *frames*. Each Ethernet frame as shown in **Figure 8** consists of seven fields: preamble and start frame delimiter (SFD) (which are part of the physical layer) destination address, source address, length or type of data unit, data padding and cyclic redundancy check (CRC).

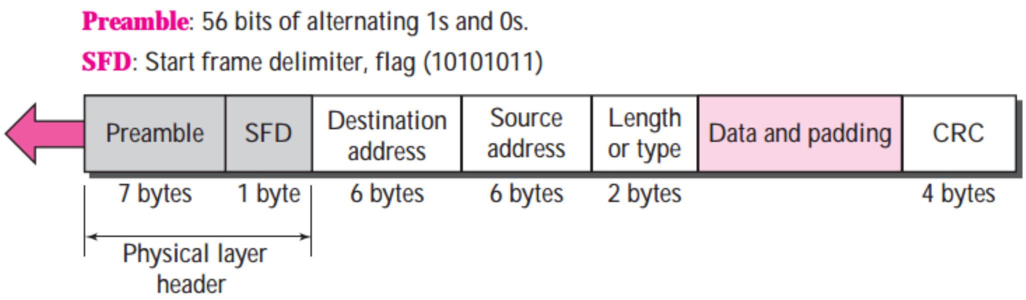


Figure 8. The Ethernet frame (Forouzan 2010: 48).

Based on the basic segments of a packet as in **Figure 2**, the header of an Ethernet frame is made up of the destination and source addresses, each of which are 6 bytes long, as well as 2 bytes for the type or length of the data unit. The trailer portion of the frame holds 4 bytes CRC while the payload has a minimum length of 46 bytes and a maximum length of 1500 bytes.

2.1.4. Internet Protocol

The Internet Protocol (IP) is a network layer (OSI Model) and an internet layer (TCP/IP protocol suite) transmission mechanism which provides end–to–end communication in a network. The IP packet which is the unit of communication in the network layer known as *datagram* is as shown in **Figure 9**. These datagrams can move along various routes in the network thereby arriving at the destination out of order or repeated. A record of the datagram route is not kept by the IP and it does not reorder datagrams at the endpoint. **Figure 10** also shows the header format of the datagram. It is 20–60 bytes in length and contains information for routing and delivery. The significant difference between the frame and the datagram is that the frame contains physical addresses in the header while the datagram header has IP addresses. The maximum length of an IP datagram is 65535 bytes.

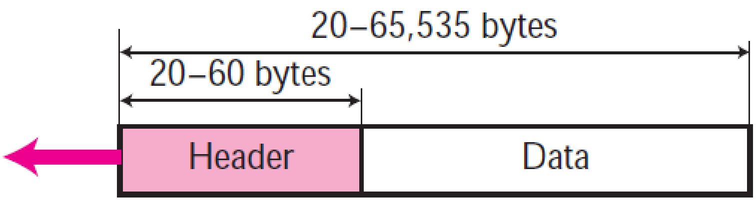


Figure 9. IP datagram (Forouzan 2010: 188).

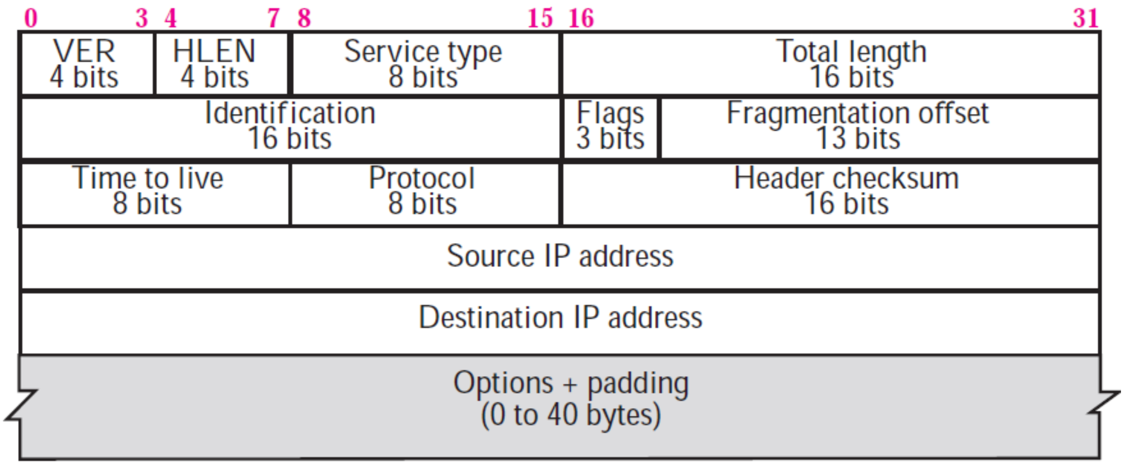


Figure 10. IP datagram Header format (Forouzan 2010: 188) (VER – IP version, HLEN – header length).

2.1.5. User Datagram Protocol

Located between the network and application layer of the TCP/IP protocol suite the User Datagram Protocol (UDP) is a connectionless transport layer protocol which uses port numbers to achieve process–to–process communication. The UDP does not give any acknowledgement for packets received. Also, no flow control system is in place. Packets are known as *user datagrams* and have an 8 byte header as in **Figure 11**. (Forouzan 2010: 415–416) Little overhead is added to the user datagram when compared to that of the IP therefore, it is termed lightweight protocol. With the UDP, data throughput is most important and can be used for time service, VOIP, videoconferencing and so on. TimeSync is mapped over UDP in IEC 61850.

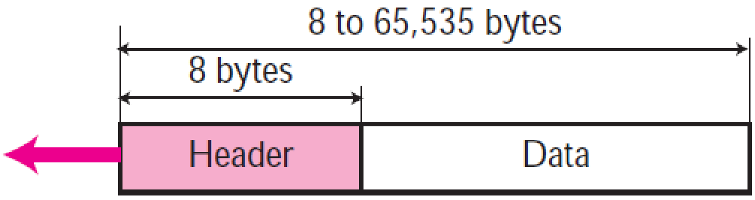


Figure 11. UDP User datagram (Forouzan 2010: 416).

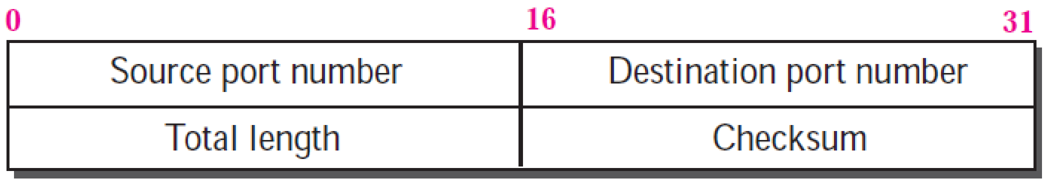


Figure 12. UDP User datagram header format (Forouzan 2010: 416)

2.1.6. Transmission Control Protocol

Transmission Control Protocol (TCP) is a connection–oriented transport layer protocol which offers a means of reliable transfer of messages from the source to destination port. It gives an acknowledgement for every packet sent, detection and retransmission for lost packets and reassembling of messages by using the sequence number in the header. A packet in TCP is known as *segment* and is shown in **Figure 13**. The header of a segment as shown in **Figure 14** is at least 20 bytes and contains the source and destination addresses, acknowledgement number, and sequence number. The TCP is considered a heavyweight protocol because it has router and link properties for its reliability functions. (Budka et al. 2014: 81) In this protocol, data integrity is very important. It is used for services such as HTTP, file transfer and Telnet.

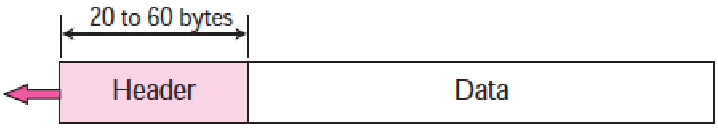


Figure 13. A segment (Forouzan 2010: 439).

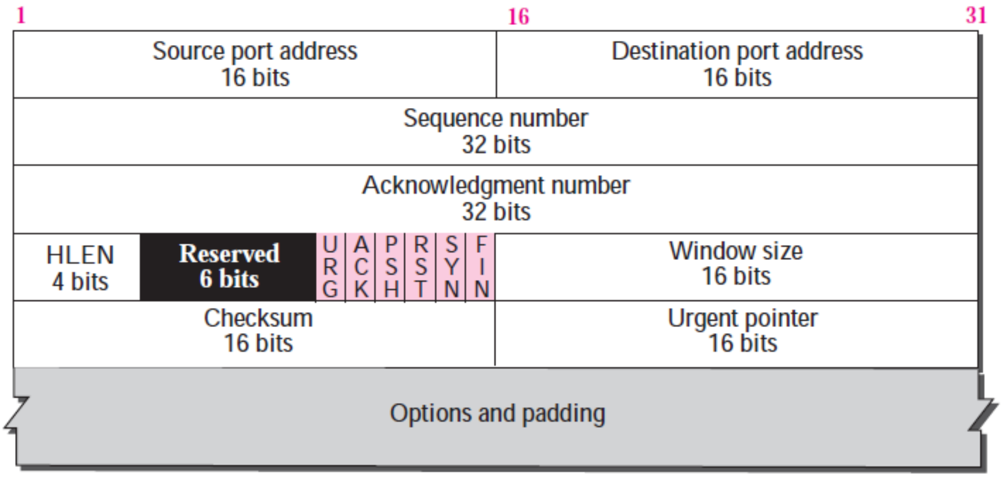


Figure 14. The header of a segment (Forouzan 2010: 439).

2.1.7. Manufacturing Message Specification

Manufacturing Message Specification (MMS) is a protocol in the application layer developed for the exchange of data in real time and supervision of control information between computer applications and networked devices. Developed for industrial process optimization, the MMS specifies a communication mechanism in a Client/Server form that uses an object – oriented modelling approach. The modelling approach includes *object classes* such as event condition, named variable; *instances* from the classes and *methods* such as write, store and stop. The Client may be a control center, an operating system or a monitoring system while the Server symbolizes real devices or a system. The server encompasses the objects which are accessible by the client and also execute services (NettedAutomation 2002). It models real time data such as pressure measurement and defines how the object is presented to and accessed by a client.

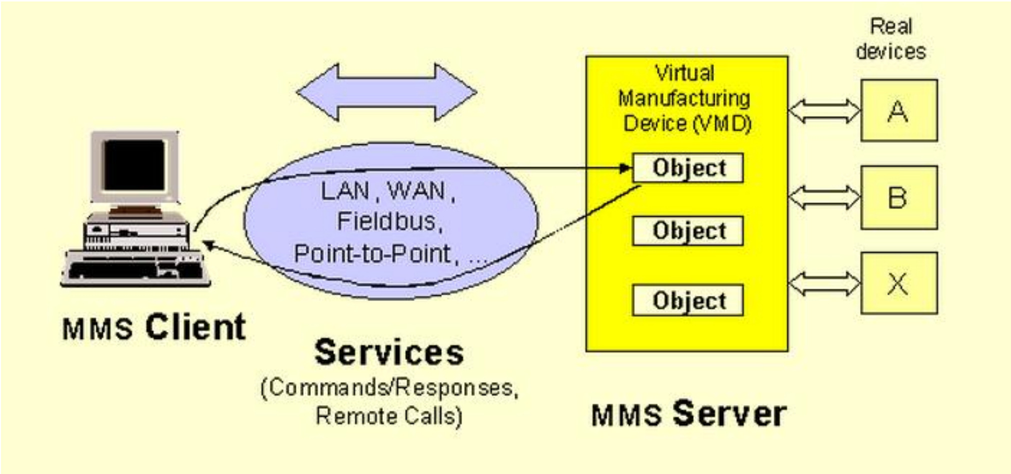


Figure 15. MMS Client/Server Model showing Virtual Manufacturing Device (NettedAutomation 2002).

From the Client/Server model shown in **Figure 15** the Virtual Manufacturing Device (VMD) which is a basic component of the MMS is seen. It explains how servers behave when seen from the view point of an MMS client application. It specifies the objects (variables in the servers), the services (used to access or manipulate the objects) and the server responses when service requests are received. The VMD shows how data is transferred between MMS clients and server. (NettedAutomation 2002) The object model is generic and can be adapted for various devices, industries and applications. The MMS is advantageous in that it makes it possible for network layer applications to exchange data while they remain independent of the type of task performed, connectivity and developer application. This makes it a vital component of IEC 61850 as virtualization is a major part of the standard.

2.2. Substation Automation System

The Substation is an important part of the grid as it enables the electricity from the generating station to be collected and distributed by connecting various links of the network and by monitoring the energy as it travels to the consumer. Today, the substations are increasingly intelligent, equipped with monitoring tools and control systems that embrace the use of evolving technology such as multi-task operation systems and relational databases (Liang & Campbell 2008: 1–12) thereby making the management of the large amount of data in the power grid possible.

The phrase “Substation Automation” usually denotes the utilization of microprocessor-based Intelligent Electronic Devices instead of conventional Voltage Transformers (VTs), Current Transformers (CTs), Remote Terminal Units (RTUs), bay controllers and relays. Also, it refers to the evolution and usage of Distribution Management System (DMS) applications based on the improved control and monitoring functions delivered by IEDs (Budka et al. 2014: 95). Technology advancement brought about the replacement of fuses by electromechanical relays which in turn are being replaced by micro-processor based Intelligent Electric Devices (IEDs). The devices in substation are monitored, protected and controlled by substation automation system (SAS) which is a system that gathers information from power equipment and acts on it. The automation functions in the substation are made possible because of the recent developments in communication technology and electronics (Roostaee, Hooshmand & Ataei 2011: 393).

2.2.1. Substation Automation System Architecture

The IEC 61850 standard (communication networks and systems in substations) has been identified as the key standard for substation automation and protection for the smart grid. The SAS architecture is made of three functional levels based on the IEC 61850 communication protocol; these are the process level, bay level and the station level. IEC 61850–7-1 2003: 14). **Figure 16** shows a typical SAS architecture.

Gathering of data from the switchyard devices and switching operations occur at the process level. Primary equipments such as merging units, VTs, CTs, and remote input/output actuators belong to this level. The Bay level is the one between the process level and the station level; it consists of IEDs for protection and control. The station level is mainly for supervision of the equipment’s in the substation. Engineering workstations, Human Machine Interface (HMI), gateways that link the control center of the substation to Wide Area Network (WAN) as well as computers/software for protection and monitoring functions are found at this level. All functions of the power system that need data from two or more bays are implemented at the station level.

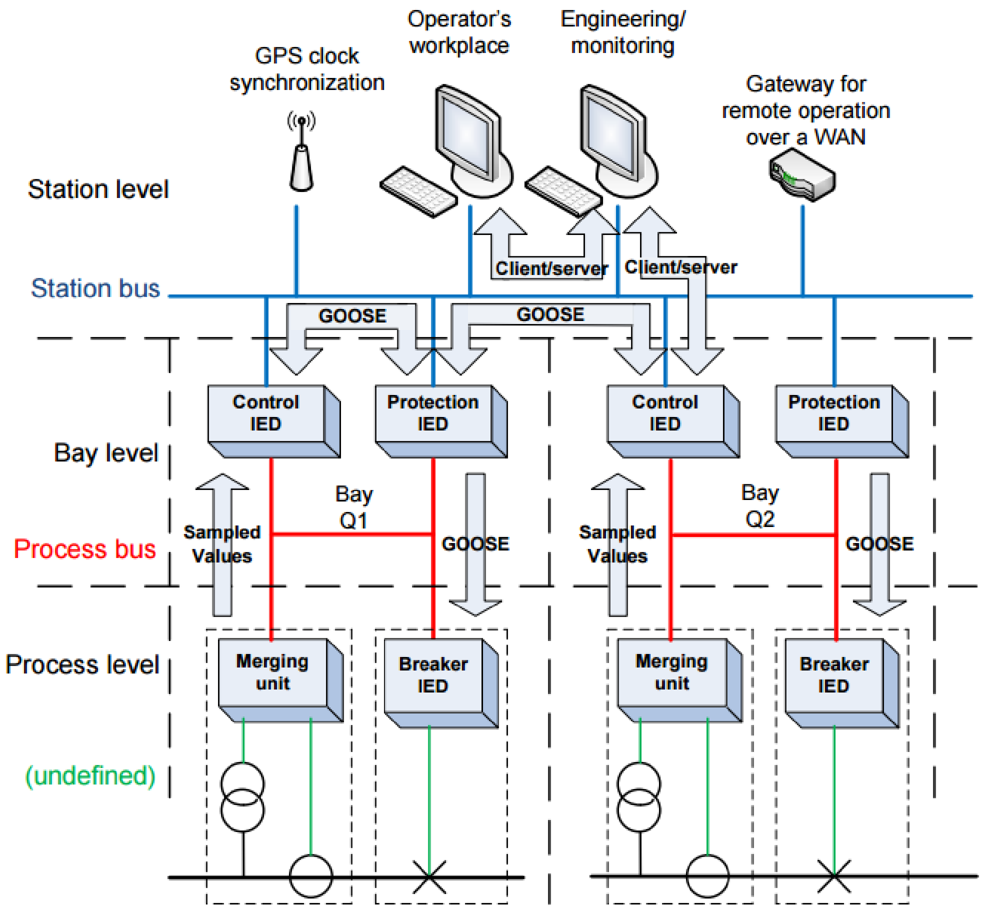


Figure 16. Substation architecture based on IEC 61850 standard (Christian Söderbacka 2013).

The process bus and station bus are the communication networks by which information exchange within these levels take place. Information exchange between process level and bay level is through the process bus while communication between the bay level and the station level is via the station bus (Golshani et al. 2014: 2). The focus of this thesis is on the process bus.

3. COMMUNICATION AND AUTOMATION STANDARD– IEC 61850

The International Electrotechnical Commission (IEC) standard 61850 on "Communication Networks and Systems in Substations" is an Ethernet-based communication network standard for automation and protection within the substation. The design objectives of IEC 61850 are to solve data management issues through the use of modern communication methods, gain high level of application interoperability by the use of object models that are standardized and also make the substation engineering process simpler by using a common configuration language. The standard which was drafted by substation automation specialists from 22 countries enables interoperability between devices from different vendors in the electric power substation as it defines the *communication protocol*, *configuration language* and the *data format* within a substation (IEC 61850–1 2003: 6,11; Kanabar & Sidhu 2011: 725–727). The standard describes high-speed peer-to-peer and or client/server communication between IEDs and devices in the substation. It also stipulates other requirements of the system such as information security, and message performance in the substation automation system network.

3.1. IEC 61850 Parts

IEC 61850 Standard document is divided into 10 main parts as shown in **Table 1** and it explains the different features of substation communication networks. Parts 1 to 4 give general details of the standard and also the requirements for communication in a substation. Part 5 provides the details of the parameters needed for physical implementation while Subscription Communication Language (SCL) based on Extensible Markup Language (XML) that presents a formal view of the relationship between the switchyard and the SAS is presented in Part 6. Furthermore, Part 7, which has four sub–parts, defines the logical concepts. Part 8 describes mapping of abstract services to the protocols. Information on how the mapping of sampled measurement value onto an Ethernet data frame is given in Part 9 and part 10 explains how conformance testing should be carried out (Liang et al. 2008: 1–4).

Table 1. Parts of IEC 61850 standard document.

Part	Title
1	Introduction and overview
2	Glossary of terms
3	General Requirements
4	System and Project Management
5	Communication Requirements for Functions and Device models
6	Configuration Description Language for Communication in Electrical Substations Related to IEDs
7	Basic Communication Structure for Substation and Feeder Equipment
7–1	Principles and models
7–2	Abstract Communication Service Interface (ACSI)
7–3	Common Data Classes (CDC)
7–4	Compatible logical node classes and data classes
8	Specific Communication Service Mapping (SCSM)
8–1	Mappings to MMS (ISO/IEC 9506 – Part 1 and Part 2) and to ISO/IEC 8802 – 3
9	Specific Communication Service Mapping (SCSM)
9–1	Sampled Values over Serial Unidirectional Multidrop Point – to – point Link
9–2	Sampled Values over ISO/IEC 8802 – 3
10	Conformance Testing

3.2. IEC 61850 Application and Communication views

The architectural concept that IEC 61850 assumes is the abstracting technique. This makes it possible for data objects and services to be defined without any underlying protocol and ensures that the system will is well-suited for advancements in communication technology (Golshani et al. 2014: 3). Abstract services are designed to separate object models and applications from system specifics and only those aspects needed to define the required actions on the receiving side of a service request are described. The standard is designed to be able to function in domains other than substation automation; therefore, emphasis was placed on the semantics of the data by obtaining the communication specifics. This permits the mapping of data objects and services to protocols that can meet the data and service needs of the system.

Figure 17 depicts the application and communication views of IEC 61850. The application view is made up of organized and standardized data models; logical nodes, data objects, and attributes that stipulate the information essential to perform an application and exchange of data between IEDs which makes interoperability possible while the communication view presents object–oriented communication that organizes the data by functions to facilitate distributed applications. (Janssen Marco: 2010).

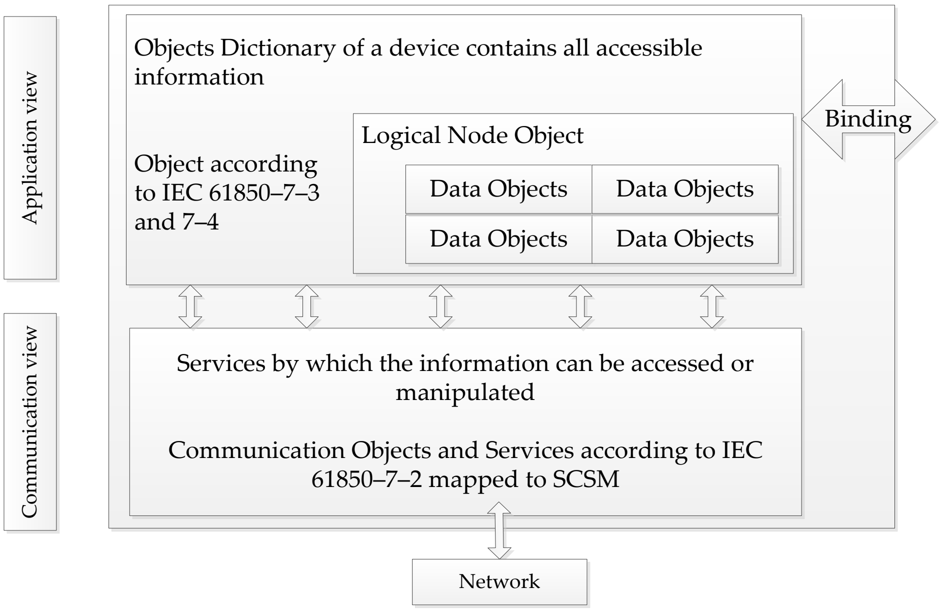


Figure 17. Application and communication views of IEC 61850.

As seen in **Figure 18,** applications in the substations which are accomplished by data objects and services have an abstract interface. This interface provides the means for mapping various communication services/profiles to the specific communication stack that can meet the requirements of the services based on the standard specification. The syntax, encoding of messages that transmit the service parameters and the method of how these are sent over a communication network are defined in a specific communication service mapping (SCSM) (IEC 61850–7-1 2003: 22). This is further explained in section 3.2.2.

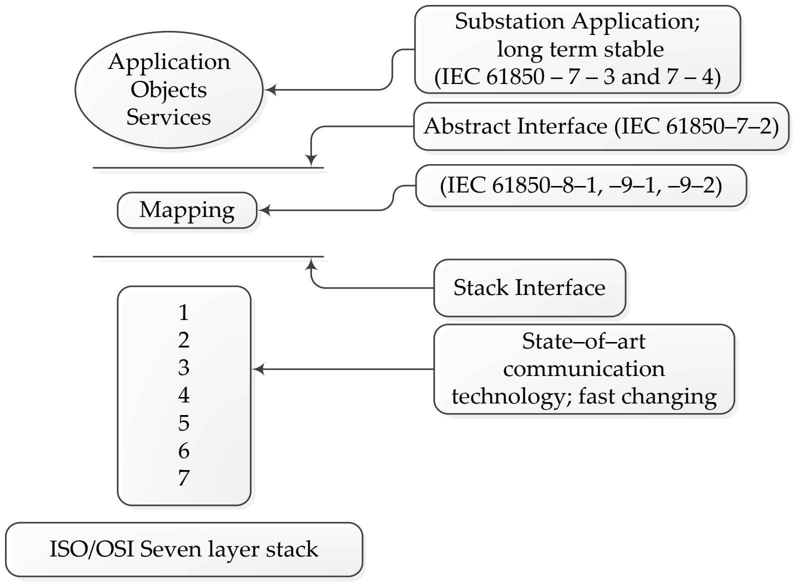


Figure 18. IEC 61850 view from application to communication (Janssen Marco: 2010)

Virtualized Model

Virtualization means that there can be an abstract representation of every real device. It gives a view of the features of a physical device that are of interest for the exchange of information between devices. This is depicted in **Figure 19.** IEC 61850 is designed with interoperability of various functions that are performed by different physical devices in mind; hence, the use of standardized data objects in the exchange of data within the standard. These standardized objects are defined such that they give only the details of the aspect of the physical device that are needed to achieve interoperability.

Virtualization is enabled as a result of the mapping of the standard over MMS which has the VMD as a functional component as explained in section 2.1.7.

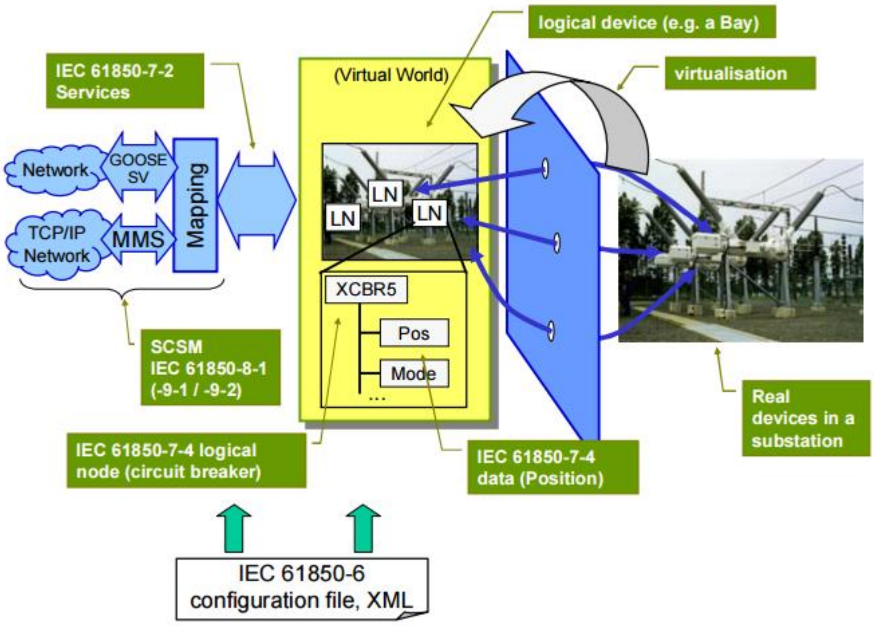


Figure 19. Substation and virtual model (Schwarz Karlheinz: 2004).

3.2.1. IEC 61850 Application view and data model

From the application point of view, the IEC 61850 standard decomposes application functions into *logical nodes* which are key elements used for information exchange. A group of logical nodes and additional services form a *logical device*. This grouping is done by associating common features of logical nodes (IEC 61850–7-1 2003: 15–16). The logical device usually does not depict a physical device. Instead, it represents an aspect of various physical devices or different logical nodes from various physical devices.

Logical nodes are made of data objects such as mode, position, and health, which have dedicated data attributes that holds actual values as shown in **Figure 20**. The format of the data is described by common data class (CDC); for example double point control (DPC) and integer status value (INS) which defines the type of data attributes that are available for a data object (Triangle MicroWorks. Inc. 2013).

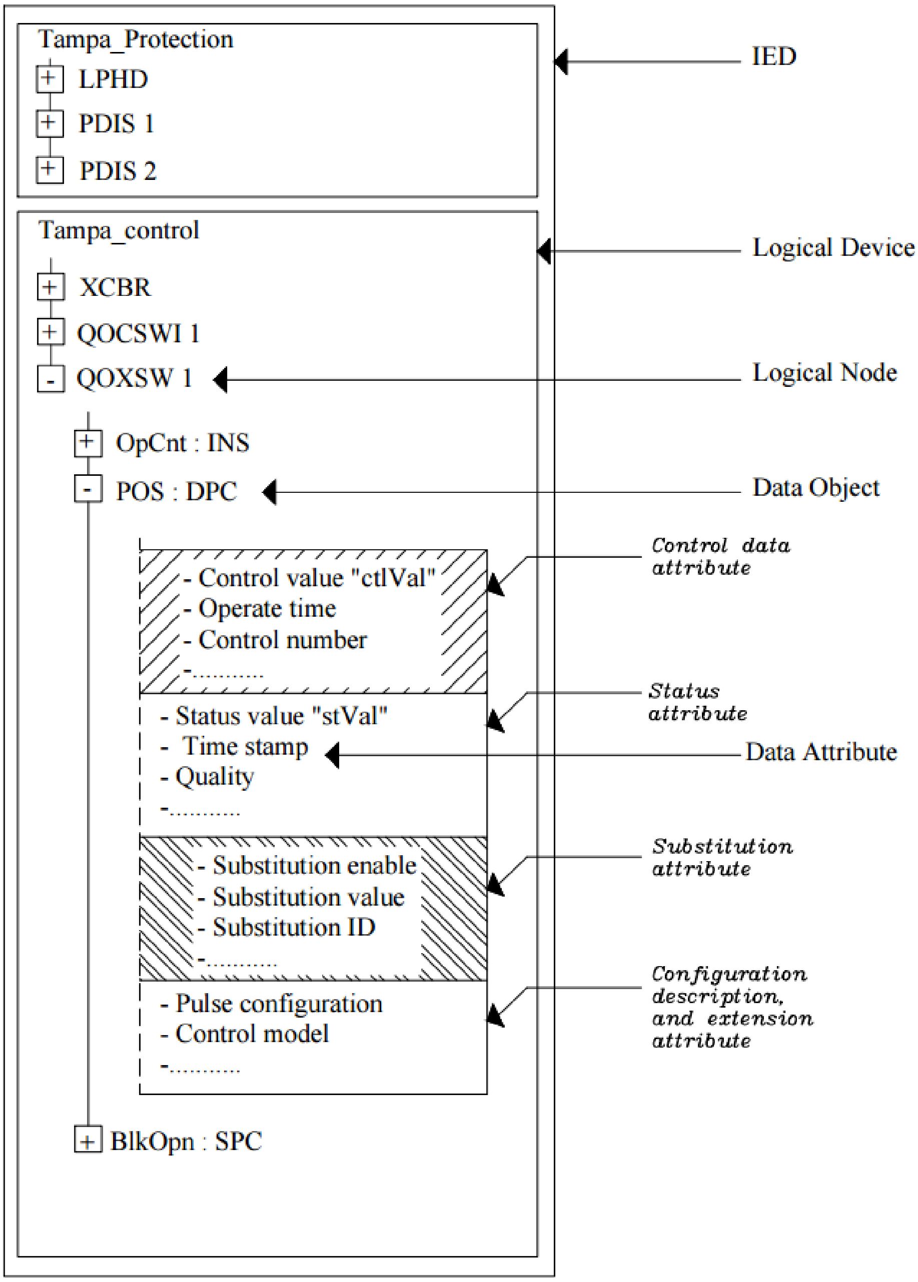


Figure 20. Elements in the data model defined by IEC 61850.

Figure 20 shows the construct of the data model based on IEC 61850. It shows the IED which contains the logical devices, logical nodes, data objects, and attributes. Each of the data objects such as the position of a circuit switch XSWI contains numerous data attributes. Position in the logical node belongs to the category called controls and all the data attributes can be categorized as: status, control, substitution, as well as configuration, description and extension which are classified together. Other examples of logical nodes are Distance Protection (PDIS), Circuit Breaker (XCBR), Trip Conditioning (PTRC), Measurement Unit (MMXU), and Switch Controller (CWSI).

Furthermore, the data and data attribute which have a clear definition in the substation automation system context states information needed to perform an application and for information exchange among IEDs. These information exchanged achieved by services such as operate, log report, and substitute based on the guidelines and requested performance stipulated in IEC 61850–5. For instance, the operate service manipulates data attribute specific to the control of a circuit breaker (i.e. open or close the circuit breaker) and report service notify other devices of the change in circuit breaker position. (IEC 61850–7-1 2003: 21–22). These services are realized by the means of Specific Communication and Service Mapping using TCP/IP, MMS, Ethernet, and other communication stack are discussed in section 3.2.2.

3.2.2. IEC 61850 Communication and Information exchange model

Interactions within the SAS are divided into 3 main types: setting/gathering of data, reporting/monitoring of data, and logging of substation events. To achieve the type of interactions mentioned, IEC 61850 defines five communication profiles/services (Liang et al. 2008: 2–5) as shown in **Figure 21**; the profiles are: Sampled Value (SV), Time Synchronization, Generic Object Oriented Substation Event (GOOSE), Generic Substation Status Event (GSSE), and the Abstract Communication Service Interface (ACSI). The abstract model is mapped to a specific communication protocol stack suitable to meet the requirements of data and services (Liang et al. 2008: 1–4; Golshani et al. 2014: 2–4).

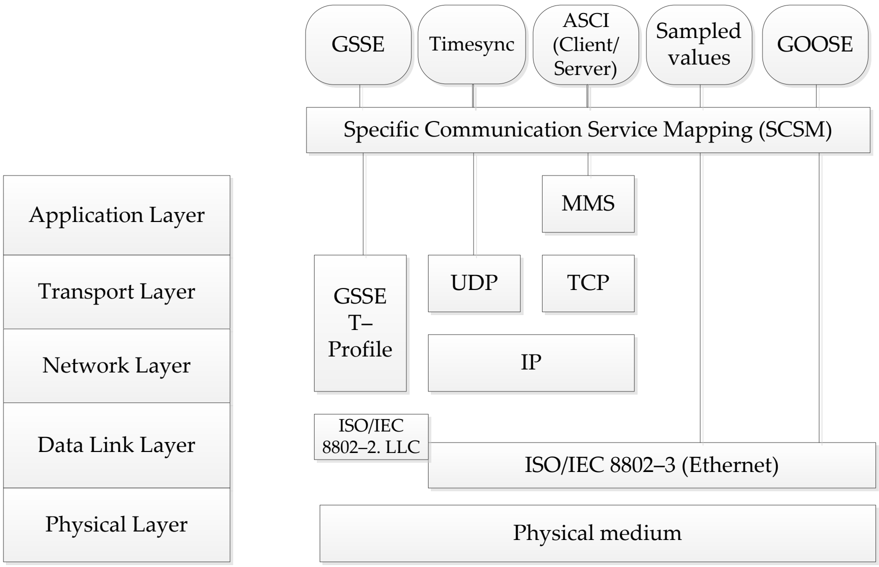


Figure 21. The communication profiles defined in IEC 61850

Abstract Communication Service Interface (ACSI) is the key interface in the IEC 61850 standard and outlines a system of client/server connection–based communication with services such as get data values or control. **Figure 22** shows the concept where there is a need for connection to be established before information exchange can be achieved. Typical applications are in the control of switch gear, file transfer and logging of sequence of events, and events reporting. It is defined in IEC 61850–7–2.

Also, it outlines the publisher/subscriber connectionless communication with generic substation event services for time critical purposes such as fast and reliable transfer of data among IEDs. **Figure 23** illustrates the process where the publisher sends out information and the subscriber can access it without prior connection. Applications are in circuit breaker tripping and sampled value transmission. (IEC 61850–7-1 2003: 49–51). ACSI explains the semantics of data exchange between applications and servers and is a central part of the logical connection between logical nodes.

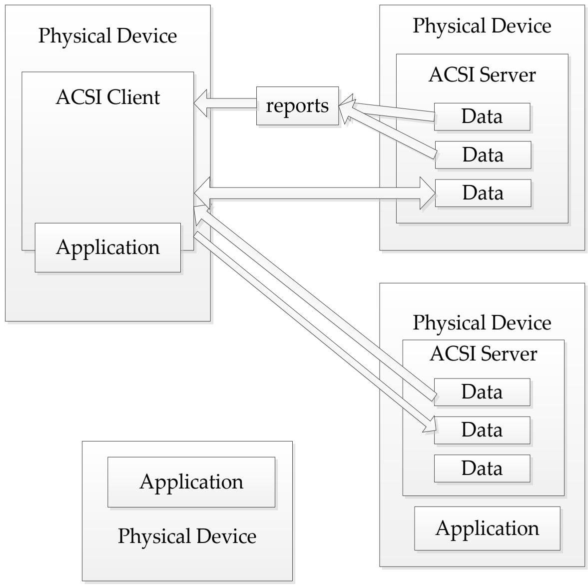


Figure 22. Client/Server communication (Janssen Marco: 2010).

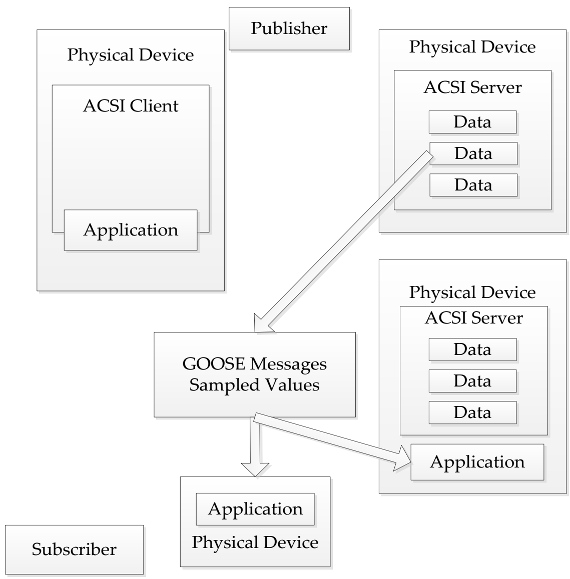


Figure 23. Publisher–subscriber communication. (Janssen Marco: 2010)

TimeSync is required to synchronize mission–critical tasks in the substation and generally in the electric grid. Precision in timing is very important so that there can be accuracy in the clocking systems of devices for data acquisition and control functions. It is particularly essential for SV timestamp because current and voltage measurements require precise timing in the MU (Ingram et al. 2013: 1445–1447).

Specific Communication Service Mappings (SCSM) enables the exchange of event and control information in the physical world. It is the expression used to describe the mapping of ACSI to a communication stack and it defines how models and services such as report controls, log controls, logical devices, logical nodes, and data are realized using a particular communication stack.

While the mapping and the application layer in question specify the syntax for data exchange over the communication network, the SCSM is independent of the communication stack or application protocols; it has been designed to achieve interoperability amongst devices by providing a platform for all devices and applications to access the required communication services. The SCSM maps objects, parameters, and abstract communication services to the specific application layer which may require one or more communication stack based on the technology of the communication network (IEC 61850–7-1 2003: 65–66; Triangle MicroWorks.Inc 2013).

GSSE and **GOOSE** as defined in IEC 61850–8–1 are time–critical and provide a system of fast information exchange within a substation. GSSE specifically transmit information about a change in status of logical nodes which enables the monitoring of data objects/attributes in the SAS. GOOSE is used for data exchange involving protection functions where high throughput multicast peer-to-peer communication is required.

Sampled Value which is also time–critical offers an effective means of transmitting high throughput streams of sampled data on the process bus. IEC 61850–9–2 describes an SCSM for SV exchange. SV and GOOSE are based on IEEE Standard 802.3/IEC 8802.3 Ethernet with Virtual Local Area Network (VLAN) tagging based on IEEE 802.1Q for prioritization (Ingram et al. 2013: 1446).

The maximum communication delay based on IEC 61850 for SV and GOOSE messages (time-critical messages) has been fixed within the range of 3ms to 4ms without consideration for the load on the network. To achieve this, the SV and GOOSE messages has been mapped directly onto the Ethernet link layer thereby eliminating all intermediary layers on the Transmission Control Protocol/Internet Protocol (TCP/IP) stack which may cause delay (Kanabar et al. 2011: 726).

3.2.3. IEC 61850 Substation Configuration Language

Introduced in part 6 of the standard, the primary purpose of the XML–based Substation Configuration Language (SCL) is to enable exchange of information between IEDs configured by different configuration tools from various manufacturers to the station computer i.e. interoperability.

The SCL consists of detailed information about the logical nodes, device models, communication structure, the application components, and how they relate with the power system. The task of the SCL is achieved by using four categories of SCL common files which are: Substation Configuration Description (SCD), Configured IED Description (CID), IED Capability Description (ICD), and System Specification Description (SSD) files.

SCD file builds a single source of all components of the substation system operation. It describes the single line diagram, configuration network, IED configuration, and substation functionality in general. The file removes all design interpretation and allows each application to retrieve its specific data subset. ICD file holds the data from a type of IED; it specifies the capabilities and sometimes the preconfigured data structure of an IED (IEC TC57 2006).

Furthermore, SSD describes power system functions such as single line diagram and the substation automation capabilities. The CID file is the configuration of a specific device or IED in the substation. It describes an IED with all device–specific configuration parameters and data applicable to it.

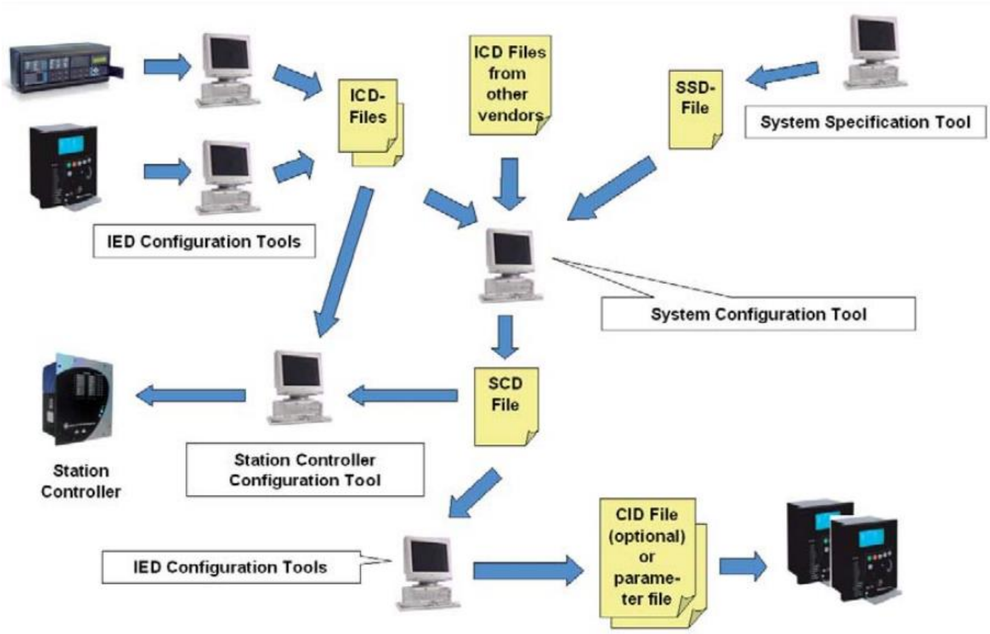


Figure 24. Conceptual substation engineering process using SCL (Goraj M. & Herrmann J: 2007).

The substation single line diagram, data, function allocations, services, defaults, configuration parameters, and communications are all defined in the SCL. As shown in **Figure 24**, the system configuration tool edits the ICD and SSD files to create the SCD file which gives the complete substation configuration parameters. The IED configuration tool extracts the CID file. The file contains device–specific data which is then downloaded to all IEDs. The configuration process is the development and setting of interface between various IEDs or IEDs and Human Machine Interface (HMI) in the substation. (Kim Y.K, Han J.K, Lee Y.J, An Y.H & Song I.J 2011: 272–273).

3.2. IEC 61850-9-2 Process Bus and benefits

The IEC 61850-9-2 Ethernet–based communication network is known as the Process Bus. This standard is the transmission protocol used to transmit measured SV from MU to IED by using a multicast address (Liu, Gao, Xiang, Wei, Wei & Zhou 2011: 83–87).

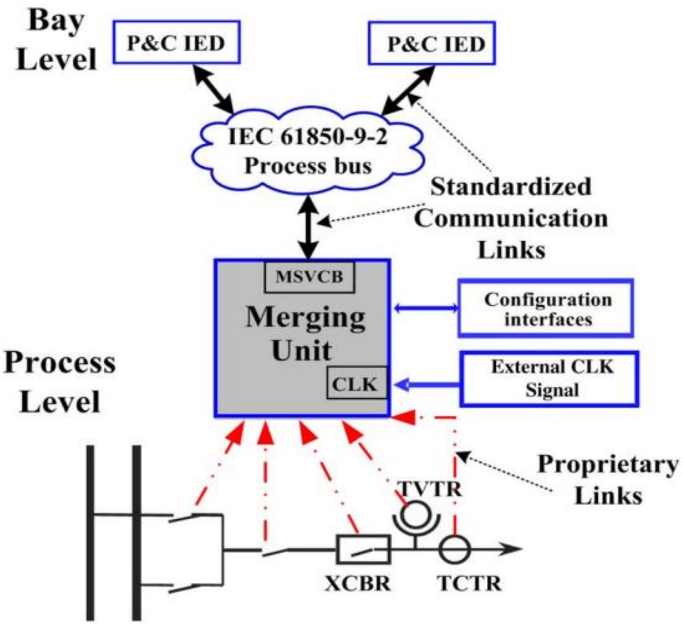


Figure 25. IEC 61850-9-2 Process Bus concept (Kanabar & Sidhu 2011: 726).

The merging unit as shown in **Figure 25** is a major element of the Process Bus. It is a network-enabled device which collects data such as phase voltages, currents, and status information from instrument transformers and transducers in the switchyard. It achieves Analog–to–Digital Conversion (ADC) and Digital Signal Processing (DSP) on the analog signals from process or bay level equipment's which are then combined into a standard sampled value packet format to make the data IEC 61850 compliant and synchronized using time stamp (Honeth et al. 2013: 1–2) . The sample value packets from the MU are sent to the protection and control IEDs over the IEC 61850 Process Bus network. With the use of IEDs in SAS, the Process Bus enables the SAS to achieve distributed control and protection abilities through the sharing of digital information in the communication network. Also, the addition of new technologies and secondary schemes in a substation becomes easier and possibly done without power outage in addition to the cost and time savings that is achieved as there are fewer cables and less wiring done.

3.4. Sampled Value Communications

The transmission protocol of SVs is described in IEC 61850-9-2 and IEC 61850-9-2 LE (Implementation Guidelines for Digital Interface to Instrument Transformers). SV is time critical and that is the reason for mapping it directly to the Data Link layer of OSI–7 model thereby having a higher transmission rate due to the reduced protocol overhead. IEC 61850–9–2 LE was developed to minimize the difficulty encountered when implementing the Process Bus. Its implementation involves defining the physical interfaces used, sampling rate, and requirements for time synchronization. (Ingram et al. 2013: 1445–1454).

3.4.1. Sampled Value packet

A standardized Ethernet frame is used in the transmission of SVs. This SV frame as shown in **Figure 26** holds the Application Protocol Data Unit (APDU). The APDU contains SV Application Subscriber Data Unit (ASDU) which holds the SV data. The byte structure of SV APDU and SV ASDU is based on Abstract Syntax Notation One (ASN.1) so as to unify types of data and value representation of the application layer (Baranov et al. 2013: 478–481; Liu et al. 2011: 83–87). The details of SV data structure as defined in APDU of the standardized IEC 61850 frames is shown in **Figure 26** and **Figure 27.** The transmission syntax follows the Basic Encoding Rule (BER) and transmission is in a format described by the triplet Type Length and Value (TLV) each of which is a series of bytes as shown in **Figure 27** (IEC 61850–9-2 2003: 24–26; Baranov et al. 2013: 478–481; Konka, Arthur, Garcia & Atkinson 2011: 43–48).

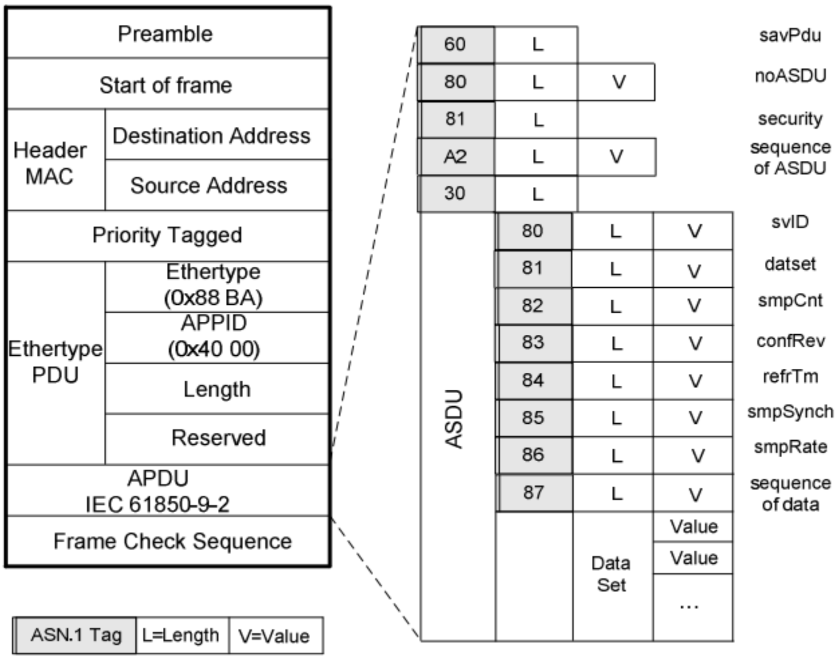


Figure 26. Frame format of IEC 61850-9-2 in ISO/IEC 8802-3 (Liu et al. 2011: 85).

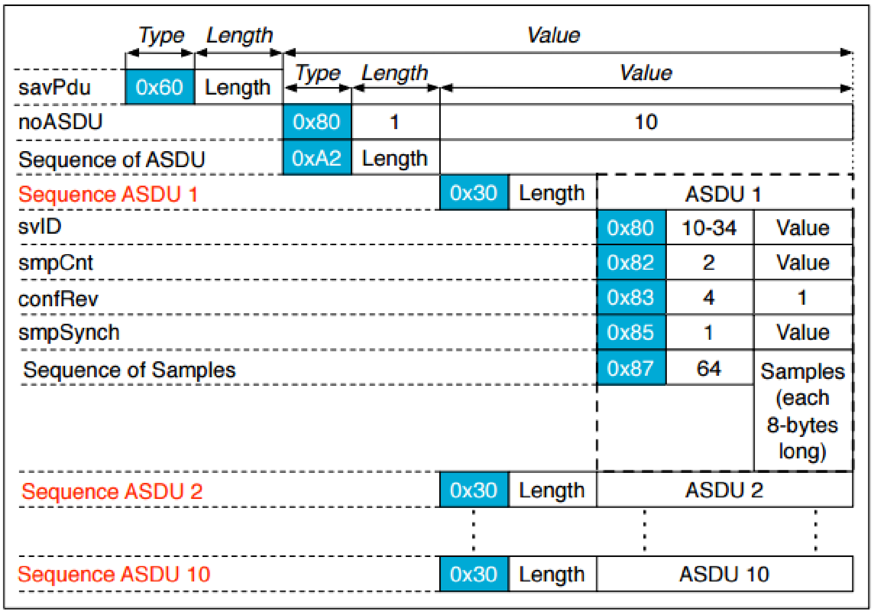


Figure 27. Structure of modelled SV APDU/ASDU (ASN.1/BER TLV triplets) (Konka et al. 2011: 45).

The SV APDU has a specific header affixed to it before sending it to the lower layers of the protocol stack. **Figure 28** shows the encapsulation of SV APDU as it goes down the stack. The header is made up of 4 fields; Application identifier (*APPID), Length, Resrv1*, and *Resrv2* is 8 bytes long (Konka et al. 2011: 45). *Resrv1* and *Resrv2* are indicated as Reserved in the diagram **Figure 26.**

Application identifier chooses the Ethernet frames which hold SV APDU and also differentiate between SV and GOOSE/GSSE protocols. The identifier is 2 bytes long, has the two most significant bits (MSBs) set to 01 in the case of SV and the remaining bits represents the identity in the substation communication network. Thus, APPID values are from 0x4000 to 0x7FFF. Also, the *length* shows the length of the packet and it is 2 bytes long. It is the sum of the SV APDU length and the SV header length written as m+8 in **Figure 28** where m is the length of APDU bytes and 8 is the length of the standard header.

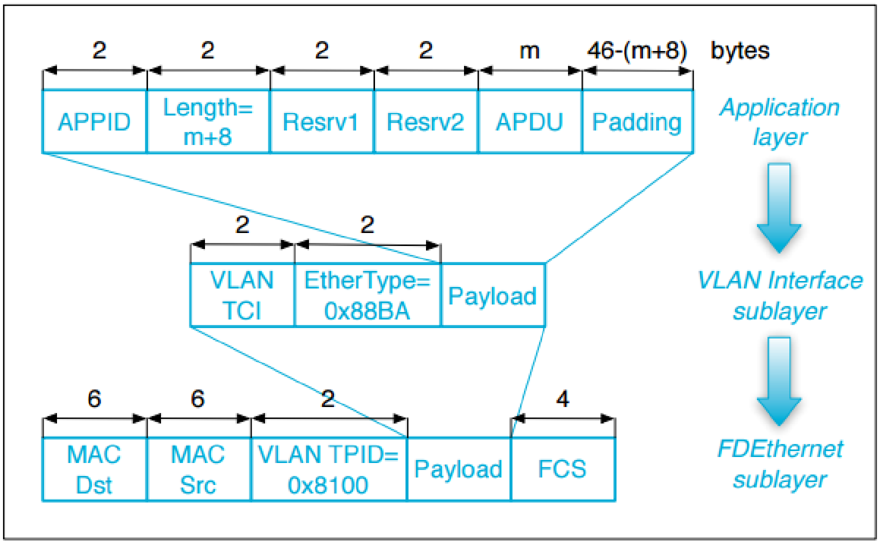


Figure 28. Encapsulation of SV APDU as it goes down the protocol stack (Konka et al. 2011: 46)

Additionally, *Resrv1* represent Reserved 1. The 2–byte long field is a block of bits for different purposes. The MSB set to 1 means the SV packet is from a test device, the next 3 bits are for future standardized application and the last 12 bits are for security purposes. *Resrv2* represent Reserved 2. It is used for more security parameters in the SV packet and is also 2 bytes long (Konka et al. 2011: 45–47).

Figure 27 shows the structure and composition of the SV APDU/ASDU fields. The starting field is *SavPdu*; it has a field length of 4 bytes. The field label is held in the first byte as 0x60, the second byte is 0x82 and the remaining bytes contain the value of the APDU length. Following the SavPdu is the ASDU number designated as *noASDU*. The field is 3 bytes in length with the first byte storing the field label as 0x80, the second holds the value length 0x01 and the third is equal to 0x08 (Baranov et al. 2013: 478–481). The *Sequence of ASDU* field is the starting part of all ASDUs. It is 4 bytes in length; the field label comes first as 0xA2, the second byte is 0x82 and the following 2 bytes contains ASDUs length value.

Considering *Sequence ASDU 1* as seen in **Figure 27**, it is the identifier of the start of ASDU 1 with a field length of 2 bytes. The field label is the first byte and equal to 0x30, the second byte is the length of ASDU 1. *SvID* field represents a name of the SV packet. It length ranges from 21 to 69 bytes with the first byte as the field label - 0x80, second byte holds the length value of the SvID and bytes 10 to 34 are the value.

Furthermore, with a field length of 4 bytes, the *SmpCnt* field which represents sample number has a field label of 0x82 as the first byte, a value length equal to 0x02 is held in the second byte and the last two bytes hold the sample number. *ConfRev* field indicates the configuration number. It is a 6 byte field where the first represents the field label marked as 0x83, the second holds the value length 0x04 and the following 4 bytes holds the value of the configuration number. *SmpSynch* field is the field that identifies the synchronization. With a field length of 3 bytes, the field label, 0x85 is the first byte. The second byte is the value length which is 0x01 while the third byte is the synchronization value. For SVs not synchronized, the value is 0x00, 0x01 for local synchronized SV and 0x02 for global synchronized SVs (Baranov et al. 2013: 478–481).

The final field in each ASDU is the *Sequence of Data* field. It is an order of measured Sampled Values with a field length of 66 bytes. With the first byte as the field label, 0x87; the second is the value length which is 0x40 and the 40 bytes that follow holds the sequence of measured sampled values. Data as regards the SVs of currents and voltages in 3 phases and neutral of the power system are in the Sequence of data field each SV is denoted by 8-byte hexadecimal code (Baranov et al. 2013: 478–481).

The frame composition example for a Sampled Value current is shown in **Figu**re **29.** In the example, the 4 byte *InnxTCTR1.Amp.instMag.i* field represents the amplitude of an SV current. This current can be for any of the phases and x in the expression *InnxTCTR1.Amp.instMag.i* can be A, B, or C phase.

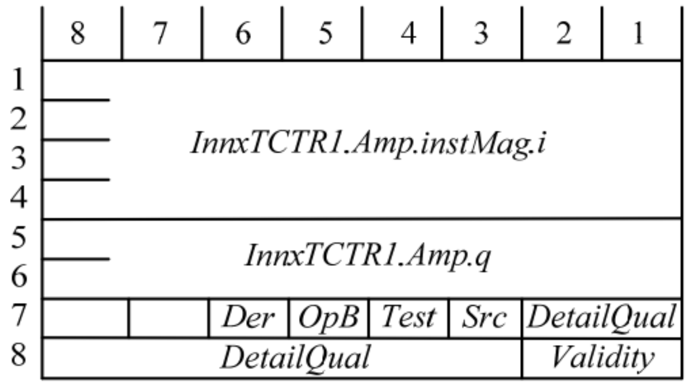


Figure 29. An example for frame format for one current sampled Value (Baranov et al. 2013: 480).

The 2 byte field named *InnxTCTR1.Amp.q* in the figure represents additional information about the SV current, *x* in the name also represent the A, B, or C phase and the default value is 0x0000. The *Der* field with a length of 1 bit shows if the derived current SV is calculated (0b1) or measured (0b0). *OpB* field also with a field length of 1 bit is used to describe the information blocking mode; setting the field to 1 implies that further value update has been blocked. The 1 bit *Test* field shows the functioning mode– testing (0b1) or working (0b0). It is used to categorize the value of current so that the value to be used for fiscal accounting can be known.

Furthermore, the 1 bit *Src* field is the source identifier; it provides information regarding the origin of the value which can be calculated value (0b1) or one that is obtained from the process (0b0). The *DetailQual* field is also 1 bit in size and is used to identify the quality of the data. The *Validity* field is a 2 byte field which can be “good”, “questionable” or “invalid” is also used to classify the quality of data (Baranov et al. 2013: 480).

4. DATA EXTRACTION AND FORMATTING

Wireshark, a network capture tool is used to capture data packets from the network and saves it as a Pcap file. The measurement data; four current and voltages is extracted from the capture file and organized in the format needed by Power System Computer Aided Design (PSCAD) simulation tool. Details of the data input format and requirements of PSCAD are in **Appendix 2**. This can also be imported to the MATLAB environment. The subsequent sections gives the details of Visual Studio 2010 configurations and programming used to achieve this using the C++ programming language.

The PSCAD is a Graphical User Interface (GUI) simulation tool developed based on the Electromagnetic Transient simulation program (EMTDC) for electric power system. It is developed by Manitoba High Voltage Direct Current (HVDC) Research Centre and has a comprehensive library which makes it a useful tool in modelling various electric power networks. (Manitoba HVDC Research Centre 2010).

4.1. Compile PCAP files

In order to read a Pcap file in Visual studio, it must be configured and pacp.h must be a header file in the program. A new project was created in C++, and new item added by selecting the 'add new item' from the drop down menu displayed by a right click on the project. C++ File (.cpp) was selected and the standard main function was added to the file.

WinPcap version 4.1.3 was downloaded from the WinPcap developer website at http://www.winpcap.org/install/default.htm and saved in the solution directory of the project after which the properties of the project were configured. The resources needed for the program are in the WinPcap file that has been saved in the solution directory. To use it in the C++ program, additional configurations were made. These included; (CACE Technologies 2005)

1. Add supplementary include directories

From the properties of the project in Visual studio, the configuration properties followed by the C/C++ entry of the project were selected. As shown **Figure 30**, The relative path "**..\WpdPack\Include**" was added to the Additional Include Directories to accommodate directories that are peculiar to the WinPcap library.

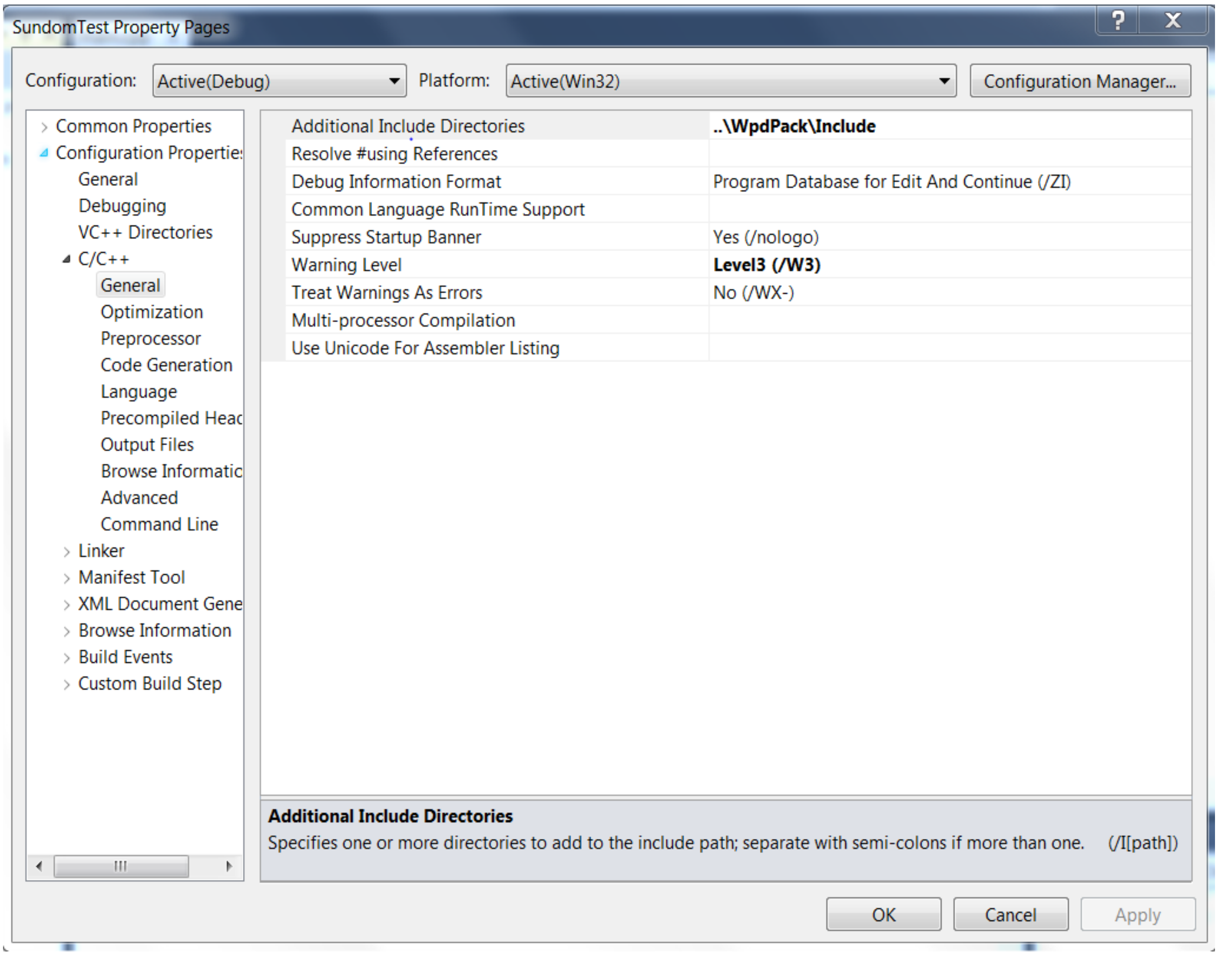


Figure 30. Add supplementary include directories.

2. Add Preprocessor Definitions

Also, from the C/C++ dropdown in the configuration properties of the project, the preprocessor was selected as shown in **Figure 31** and **_MBCS;WIN32;WPCAP;HAVE_REMOTE;%(PreprocessorDefinitions);** was added to the Preprocessor Definitions. This defines a preprocessing symbol for the source file.

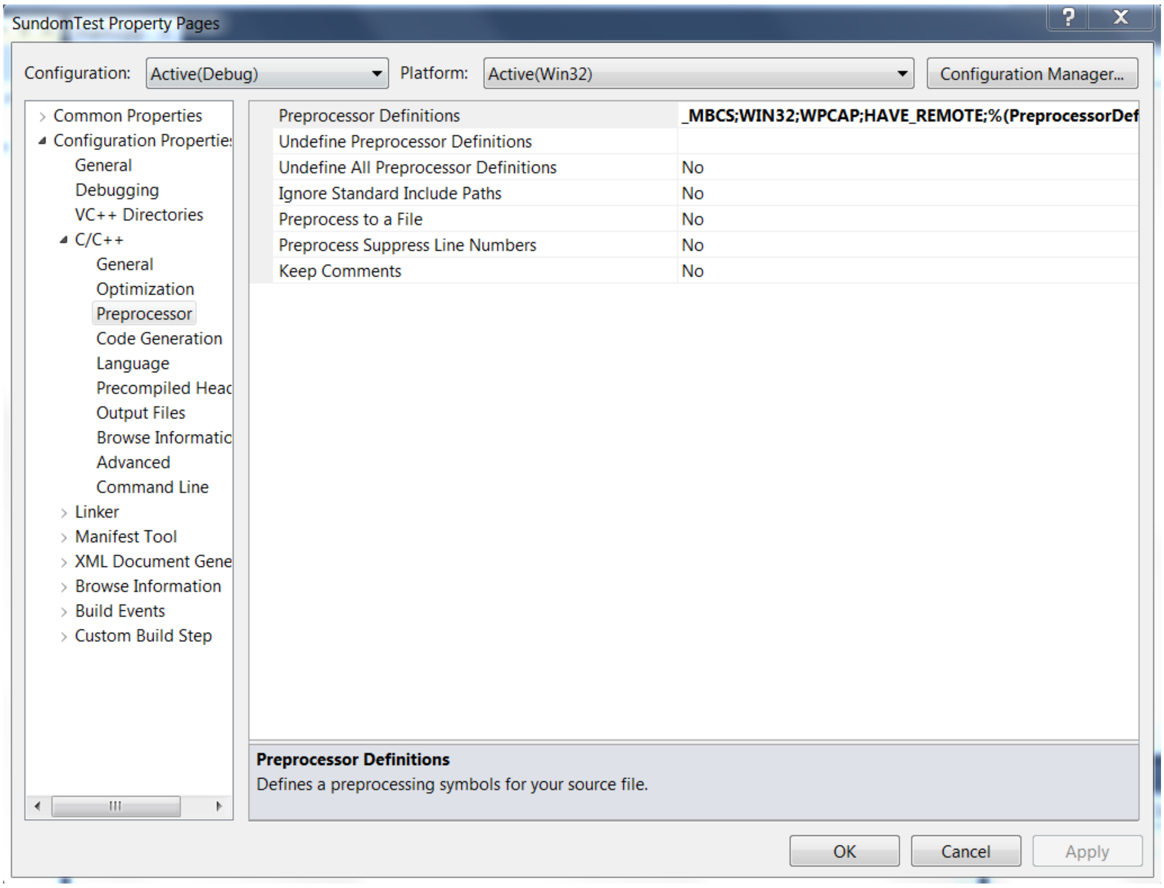


Figure 31. Add Preprocessor Definitions

3. Add additional Library directories

From the configuration properties, "Linker" and then "General" were selected. "**..WpdPack\Lib**" was added to have access to additional libraries and allows the user to override the environmental library path (/LIBPATH:folder). The configuration is as shown in **Figure 32.**

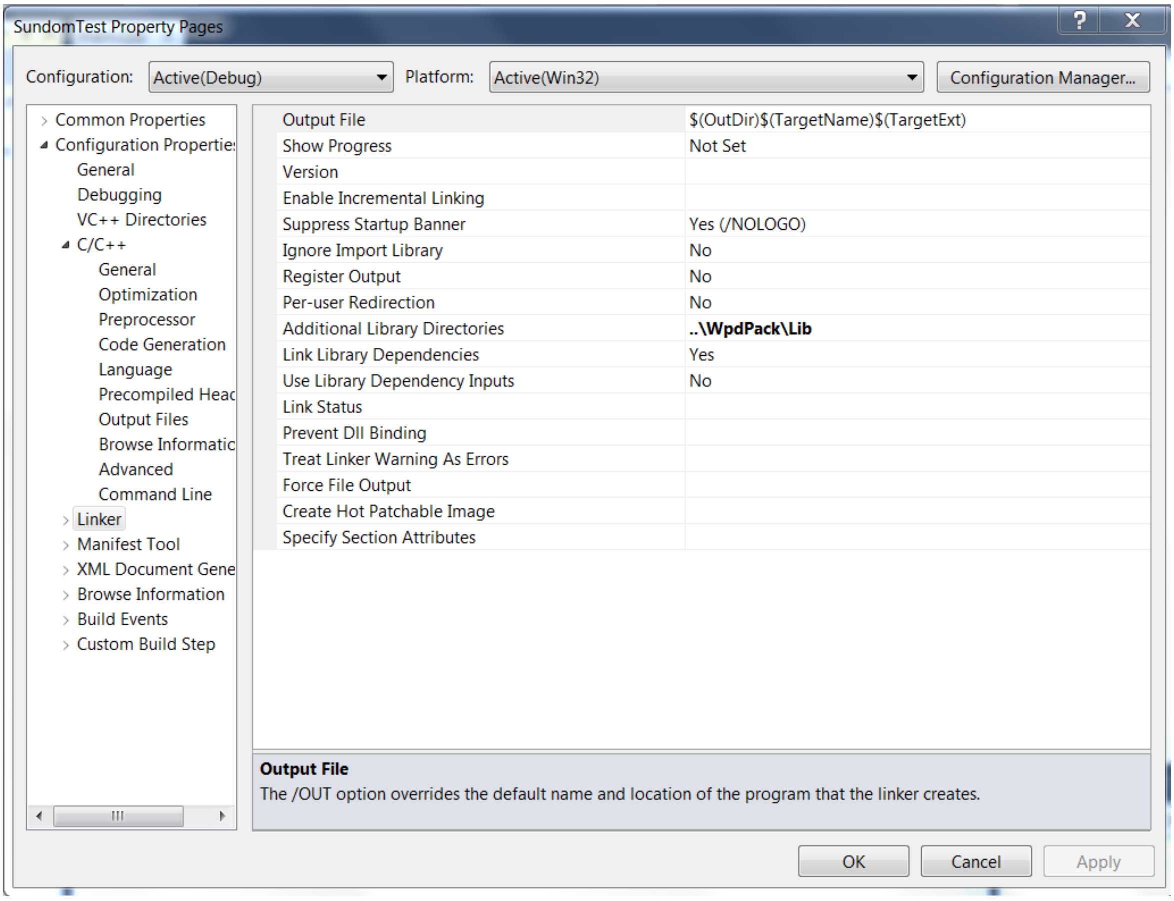


Figure 32. Add additional Library directories.

4. Add additional dependencies

As shown in **Figure 33**, %(AdditionalDependencies);wpcap.lib;Packet.Lib; which specifies the additional items to add to the link were also added by selecting "Input" under the "Linker" in the configuration properties.

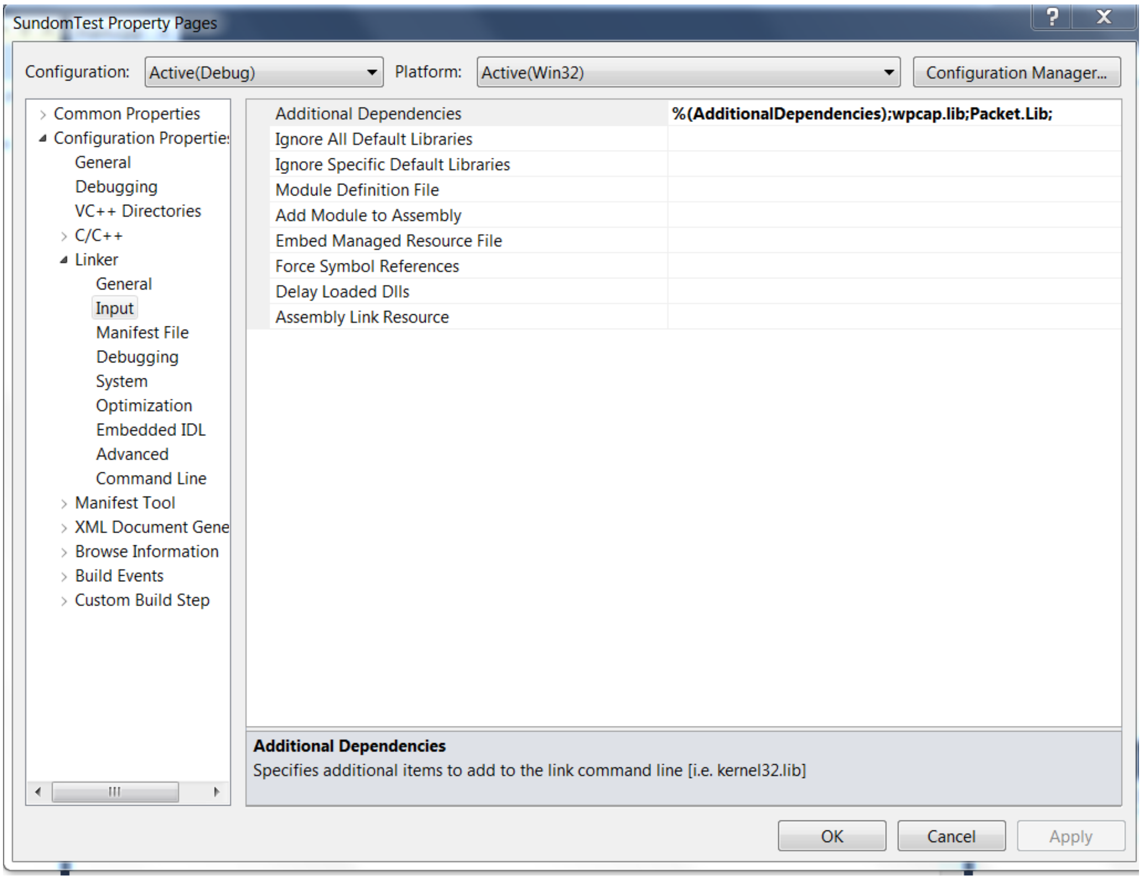


Figure 33. Add additional dependencies.

After the four configurations above, the pcap.h was included in the main.cpp file and compiled.

4.2. Data Sources

At the time of writing this thesis, 12 sources of data were identified from the frames captured by Wireshark in the Protection, Automation and Communication (PAC) laboratory at the University of Vaasa. The MAC addresses of these devices on the Sundom Smart Grid network is as shown in **Table 2**.

Table 2. MAC addresses of data sources.

S/N	Source MAC addresses
1	00:21:c1:25:de:d4
2	00:21:c1:25:de:b4
3	00:21:c1:25:de:ba
4	00:21:c1:25:de:bc
5	00:21:c1:25:de:c6
6	00:21:c1:25:de:a8
7	00:21:c1:25:de:ae
8	00:21:c1:25:de:b2
9	00:21:c1:25:de:b8
10	00:21:c1:25:de:ac
11	00:21:c1:25:de:b6
12	00:21:c1:25:de:aa

These sources send out data simultaneously and Wireshark capture data from all sources at the same. **Figure 34** shows Wireshark interface showing capture from various sources. Therefore, it is important to separate the captured packet from each source so that the data from an IED can be mapped to the representation on the PSCAD model.

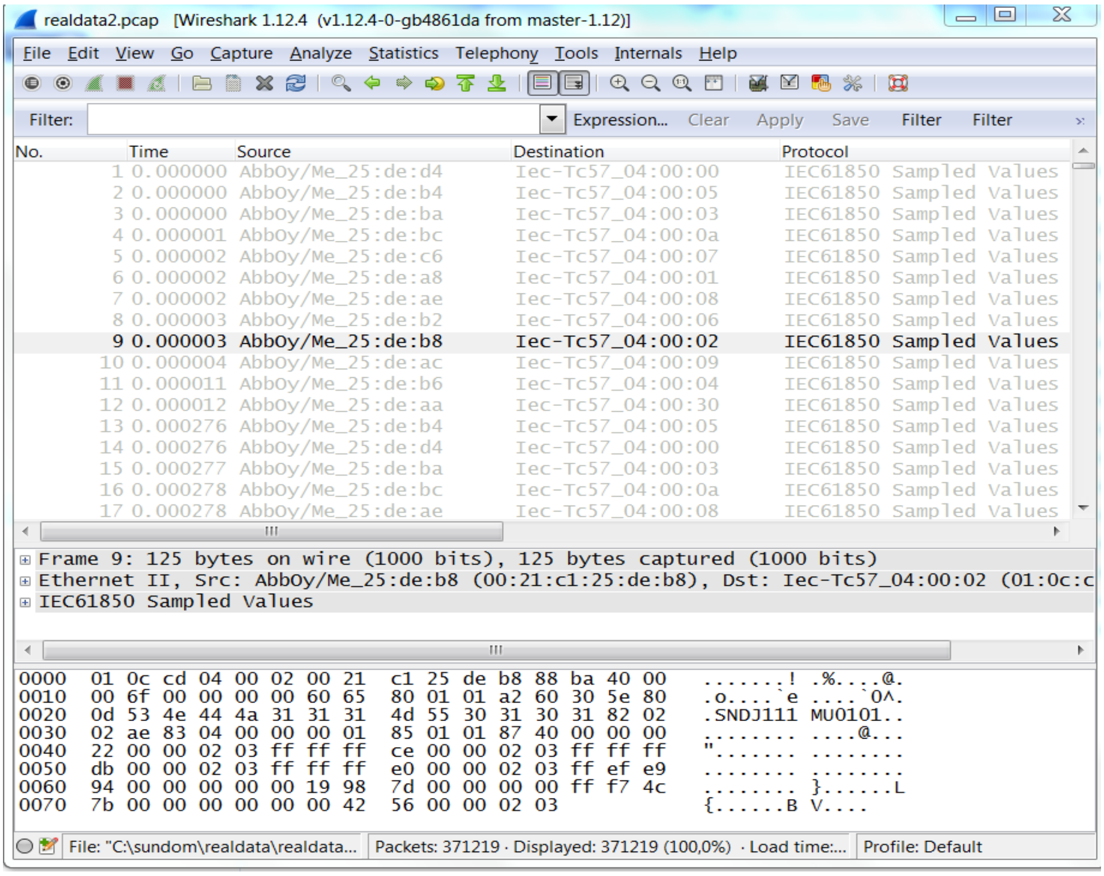


Figure 34. Outputs from Wireshark capture showing various sources.

The command eth.addr = = the MAC address i.e. eth.addr = = 00:21:c1:25:de:b8 (for source number 9 in Table 3) is applied to the capture via the filter panel on Wireshark interface and the "Export specified packets" is selected from the file menu of the capture. With this, the data from the selected source can be saved in a new file. **Figure 35** shows the output from this procedure.

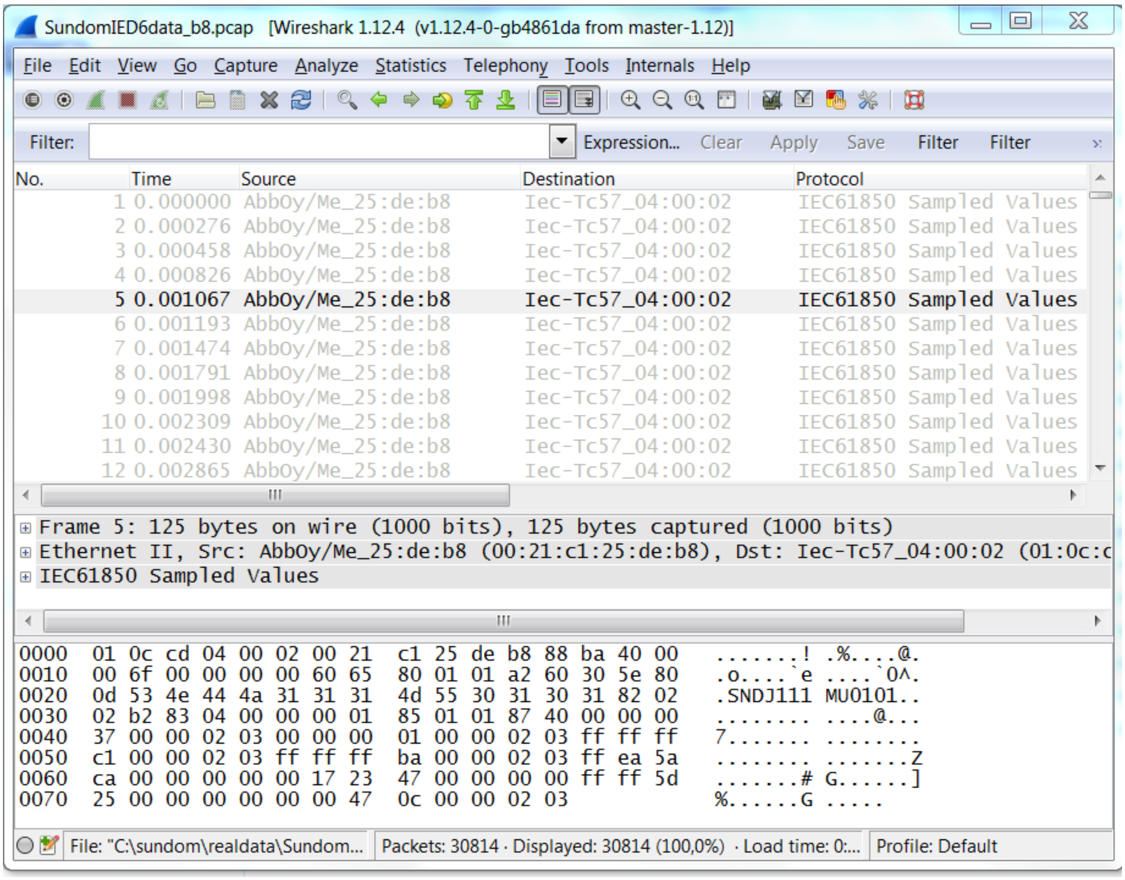


Figure 35. The output from source with MAC address 00:21:c1:25:de:b8.

The saved file can be accessed by the computer program by indicating the directory of the saved file in the program.

4.3. Computer Program to convert data to suitable format for PSCAD

The software is developed to open PCAP files, read through the file and build a list of the values in the fields needed by the PSCAD. The sequence of operation is shown in **Figure 36**.

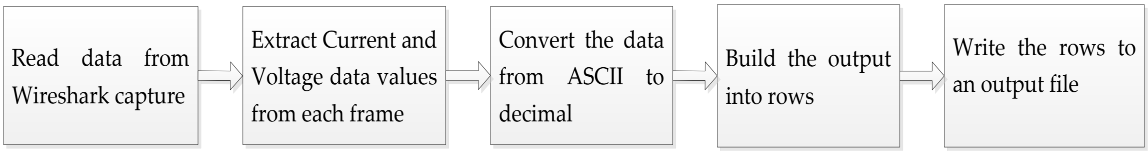


Figure 36. Sequence of operation of the computer program.

WinPcap API documentation provides details on how to read data from the Pcap file in the Windows environment. In the second line of the computer program below is the directory in which the Pcap file accessed is stored on the computer.

```
int readFileCont(){

        string file = "C:\\sundom\\realdata\\SundomIED6data_b8.pcap"; // get the
file its location on the computer

        char errbuff[PCAP_ERRBUF_SIZE];// size to use when allocating the buffer
that contains the libpcap errors
        pcap_t * pcap;
        if((pcap = pcap_open_offline(file.c_str(), errbuff)) == NULL) // open the
file and store the result in pointer to pcap_t file.c_str()
                return -1;

        int returnValue;
        struct pcap_pkthdr *header; // create a header object (ts-time stamp,
caplen-length of portion present, len-length of this packet (offwire)


        // loop through packets and print to screen
        u_int i = 0; // i is packetCount
        while (int returnValue = pcap_next_ex(pcap, &header, &data) >= 0) // read
a packet from an interface or offline capture
        {

                printf("packet # %i\n", ++i);//show packet number
                printf("packet size: %d bytes\n", header->len); // show the size in
bytes of the packet
                printf("Output Time: %d:%d seconds\n",header->ts.tv_sec, header-
>ts.tv_usec); // output Epoch time
```

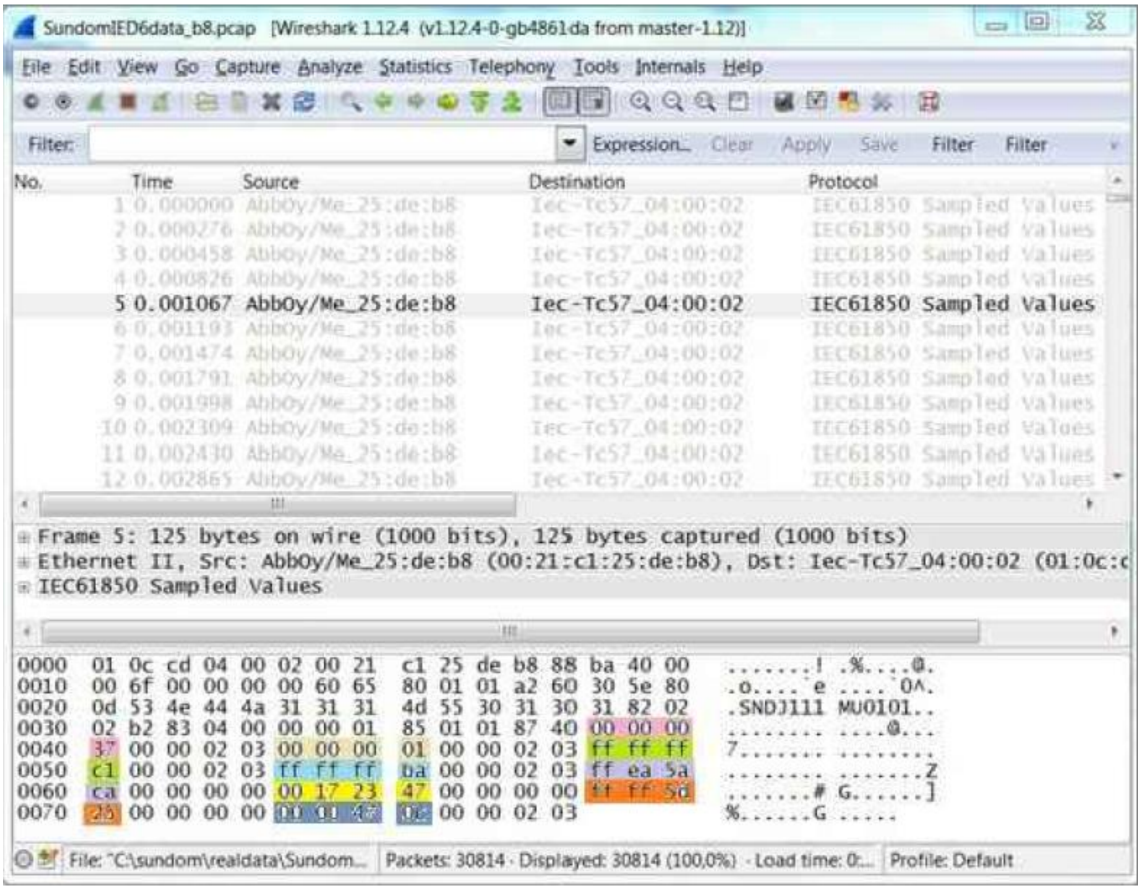
The program loops through every frame and extract the bytes that represent the information needed. The computer program can be used to extract other information from the packets captured by selecting the byte’s number. The bytes representing the three phases of currents and voltages as well as the neutral points were identified during the study of IEC 61850 Standard**.** As an example, the representation of the current in phase A of the first frame in the Pcap file is in bytes 61 – 64 and is selected in the lines of code below.

```
/* Current in phase A */
	int a,b;
	for( b = 61,a=0; b<65; b++,a++) // Selection if bytes 61 to 64 which is
the representation of current in phase 1 based on IEC 61850-9-2
```

All bytes selected from frame 5 in **Figure 35**, their data values and interpretation is as shown in **Table 3**. Also, **Figure 37** shows the data points from the Wireshark interface. More information about the frame structure of a sampled value packet is available in section 3.4.1 of this thesis.

Table 3. Bytes representing currents and voltages in frame 5 of **Figure 35**.

Byte number	Data value	Interpretation
61 – 64	00 00 00 37	Current in Phase A
69 – 72	00 00 00 01	Current in Phase B
77 – 80	ff ff ff c1	Current in Phase C
85 – 88	ff ff ff ba	Neutral current
93 – 96	ff ea 5a ca	Voltage in Phase A
101 – 104	00 17 23 47	Voltage in Phase B
109 – 112	ff ff 5d 25	Voltage in Phase C
117 – 120	00 00 47 0c	Neutral voltage



Legend

Color	Combination	Description
	00 00 00 37	Current in phase A
	00 00 00 01	Current in phase B
	ff ff ff c1	Current in phase C
	ff ff ff ba	Neutral Current
	ff ea 5a ca	Voltage in phase A
	00 17 23 47	Voltage in phase B
	ff ff 5d 25	Voltage in phase C
	00 00 47 0c	Neutral Voltage

Figure 37. Data points identified on Wireshark capture.

The output from the selection of bytes is in ASCII format, thus, it was converted to decimal value by the block of code below.

```
long int hex2dec(char * in)
{
 int n=0,N;
 unsigned char temp;
 unsigned long int out=0;
 n=sizeof(in)-1;

N=n;
 while(n>-1)
 {
        temp= in[n] & 0xFF;
        out= out | (temp << ((N-n)*8));
     n--;
 }
```

The converted value is the arranged in a row and the row is written to an output file with the function “writeRowToFile” below.

```
void writeRowToFile(string data, long sampleCount){

       fstream fileStream;
       fileStream.open("output.txt", ios::app); //opening the file for writing.
       fileStream <<sampleCount << "             " << currentA << "          " <<
currentB << "        " << currentC << "          " << currentN << "          " <<
voltageA << "        " << voltageB << "          " << voltageC << "          " <<
voltageN << endl; //writing to the file
       fileStream.close();
}
```

The complete computer program is in appendix 4 of this thesis.

4.4. Result and analysis

4.4.1. Extracted data and sample measurement plots

An output of the computer program is shown in **Figure 38** while the plot of the current and voltage values are in **Figure 39** and **Figure 40** respectively. The plot gives a sinusoid signals as expected for current and voltage. The plots also show some phase changes.

ied5 - Notepad

File Edit Format View Help

SmpCnt	IA	IB	IC	IN	VA	VB	VC	VN
0	-34547	57753	-23225	-58	-1049202	1673617	-571815	17248
1	-39025	56958	-18053	-20	-1145151	1641834	-443334	17658
2	-44324	56934	-12674	10	-1243658	1605478	-307290	18211
3	-48621	57543	-8896	40	-1337762	1566585	-172500	18556
4	-50992	55585	-4621	17	-1413756	1512520	-42796	18488
5	-53473	52946	345	5	-1477550	1445441	88072	18302
6	-55925	50134	5618	-20	-1541608	1376139	220459	18734
7	-59073	47245	11721	42	-1600585	1302837	352005	18579
8	-61742	44757	16949	31	-1648431	1221265	481217	18083
9	-61874	39924	21958	41	-1674408	1121051	606395	17798
10	-61339	33921	27438	81	-1677147	1001015	727170	16937
11	-61627	28828	32663	71	-1672000	876989	844072	16253
12	-61660	24122	37397	45	-1666913	758327	955136	15139
13	-62122	19518	42487	39	-1656760	642779	1057435	14484
14	-62104	16002	46217	28	-1633360	525097	1149853	14140
15	-60545	11254	49296	48	-1593254	392370	1239798	12845
16	-59579	5941	53599	40	-1546922	247685	1334409	11557
17	-58891	1325	57609	63	-1499809	106690	1424703	10790
18	-58480	-3757	62279	111	-1444545	-27137	1499698	9798
19	-57708	-8009	65713	98	-1381939	-153187	1559382	8533
20	-54439	-11873	66301	37	-1308588	-275736	1603993	7117
21	-49647	-16276	65891	78	-1224643	-402108	1642140	5606
22	-45061	-20164	65031	78	-1129342	-533580	1674081	4032
23	-40603	-25018	65333	92	-1020859	-664362	1692736	2884

Figure 38. Real data values written to output file.

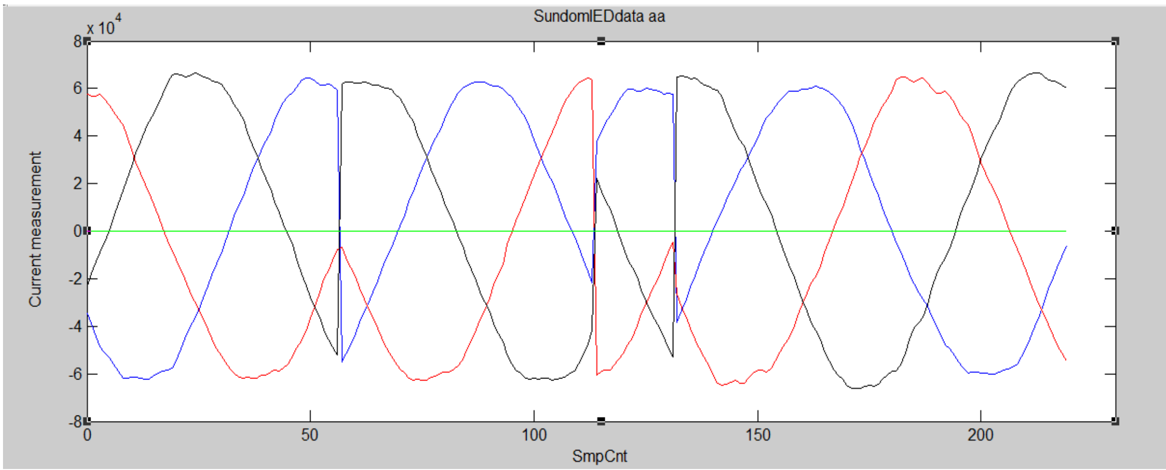


Figure 39. Plot of real current measurement.

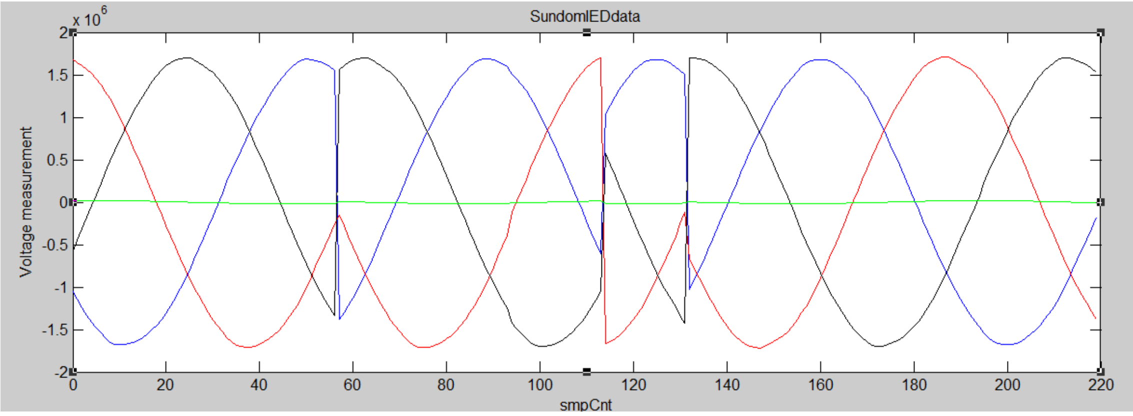


Figure 40. Plot of real voltage measurement.

4.4.2. Software Validation

As a control mechanism, sampled value data was generated from the Omicron CMC 850 device. The plot of the data values from the device is as shown in **Figure 41** and data from the Pcap file generated by the device was extracted using the computer program written.

The output of the program which follows the requirement of the PSCAD is in **Figure 42** (See Appendix 2 for requirements of the PSCAD). The extracted currents and voltages were plotted with MATLAB as shown in **Figure 43** and **Figure 44** respectively.

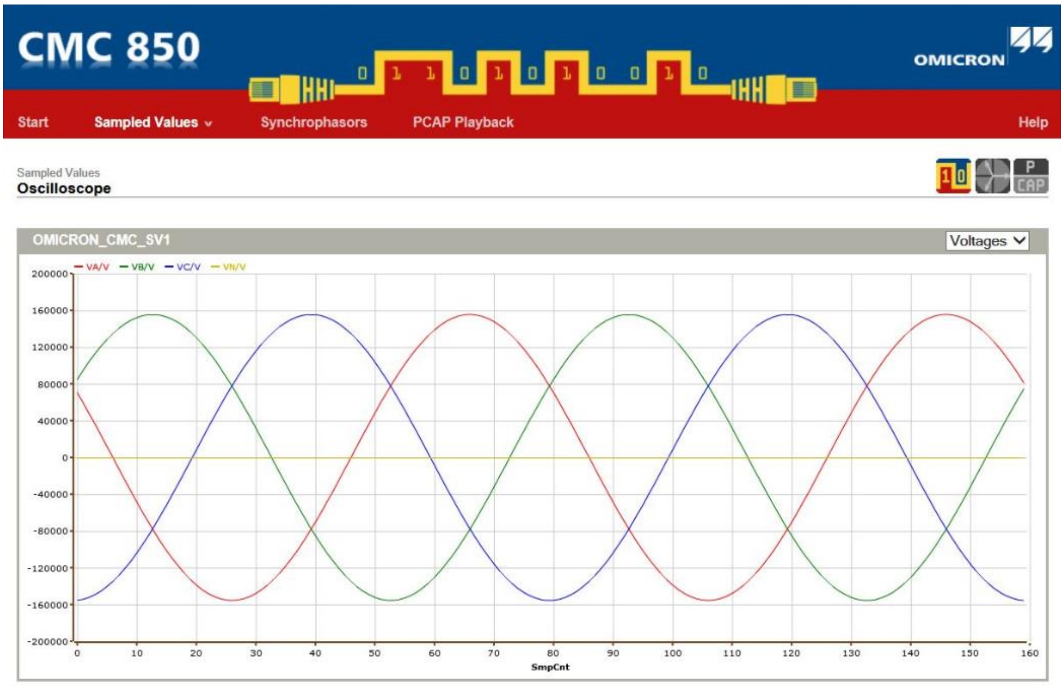


Figure 41. Waveform of Sampled Value generated from Omicron CMC 850

Output data - Notepad

SmpCnt	IA	IB	IC	IN	VA	VB	VC	VN
0	-4160689	7031281	-2870610	-18	-9153525	15468889	-6315365	-1
1	-4596491	6951314	-2354829	-6	-10112252	15292830	-5180601	-23
2	-5003777	6828204	-1824440	-13	-11008324	15022110	-4013776	10
3	-5380415	6663269	-1282867	-13	-11836868	14659138	-2822300	-30
4	-5723653	6456983	-733342	-12	-12592064	14205427	-1613360	3
5	-6031847	6211158	-179318	-7	-13270018	13664494	-394500	-24
6	-6302596	5926777	375806	-13	-13865750	13038964	826780	-6
7	-6534755	5606100	928649	-6	-14376419	12333382	2043024	-13
8	-6726352	5250626	1475703	-23	-14798018	11551426	3246568	-24
9	-6876759	4863009	2013753	3	-15128828	10698587	4430233	-8
10	-6984482	4445193	2539266	-23	-15365884	9779466	5586405	-13
11	-7049435	4000171	3049268	4	-15508723	8800344	6708361	-18
11	-7070626	3530306	3540297	-23	-15555406	7766706	7788698	-2
12	-7048534	3038834	4009704	4	-15506732	6685407	8821312	-13
13	-6982669	2528487	4454159	-23	-15361900	5562705	9799188	-7
14	-6874072	2002667	4871409	4	-15122908	4405855	10717052	-1
15	-6722775	1464412	5258340	-23	-14790140	3221727	11568394	-19
16	-6530342	917190	5613161	9	-14366716	2017817	12348904	5
17	-6297347	364283	5933046	-18	-13854195	801422	13052755	-18
18	-6025821	-190874	6216704	9	-13256760	-419908	13676683	15
19	-5716861	-744822	6461655	-28	-12577115	-1638617	14215709	-23
20	-5372921	-1294234	6667164	9	-11820389	-2847301	14667695	5
21	-4995598	-1835585	6831166	-17	-10990343	-4038300	15028630	-13
22	-4587719	-2365732	6953461	10	-10092964	-5204580	15297542	-2
23	-4151334	-2881151	7032468	-17	-9132952	-6338553	15471487	-18
24	-3689556	-3378972	7068537	9	-8117000	-7433711	15550715	4
25	-3204850	-3855770	7060601	-19	-7050683	-8482726	15533379	-30
26	-2700539	-4309019	7009567	9	-5941166	-9479795	15420988	27
27	-2179455	-4735450	6914883	-22	-4794803	-10418021	15212795	-29
28	-1645030	-5132961	6778000	9	-3619054	-11292474	14911527	-1
29	-1100393	-5498523	6598893	-23	-2420863	-12096789	14517628	-24
30	-549013	-5830513	6379530	4	-1207822	-12827072	14034899	5

Figure 42. Output obtained by extracting and formatting data generated by the Omicron CMC 850 device.

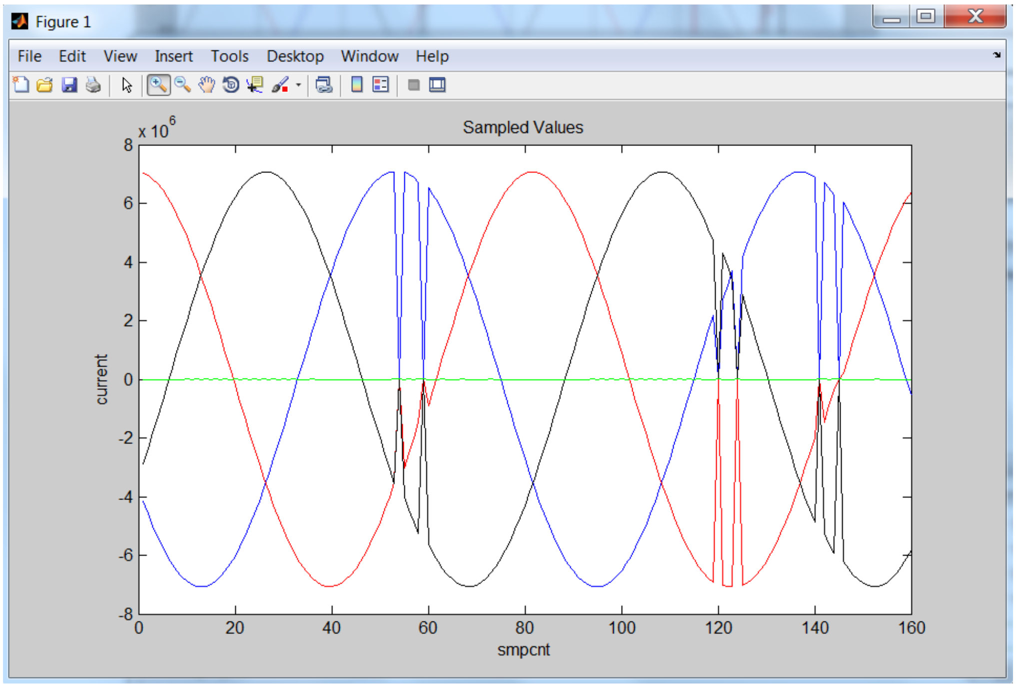


Figure 43. Current waveform of the output data from Omicron 850.

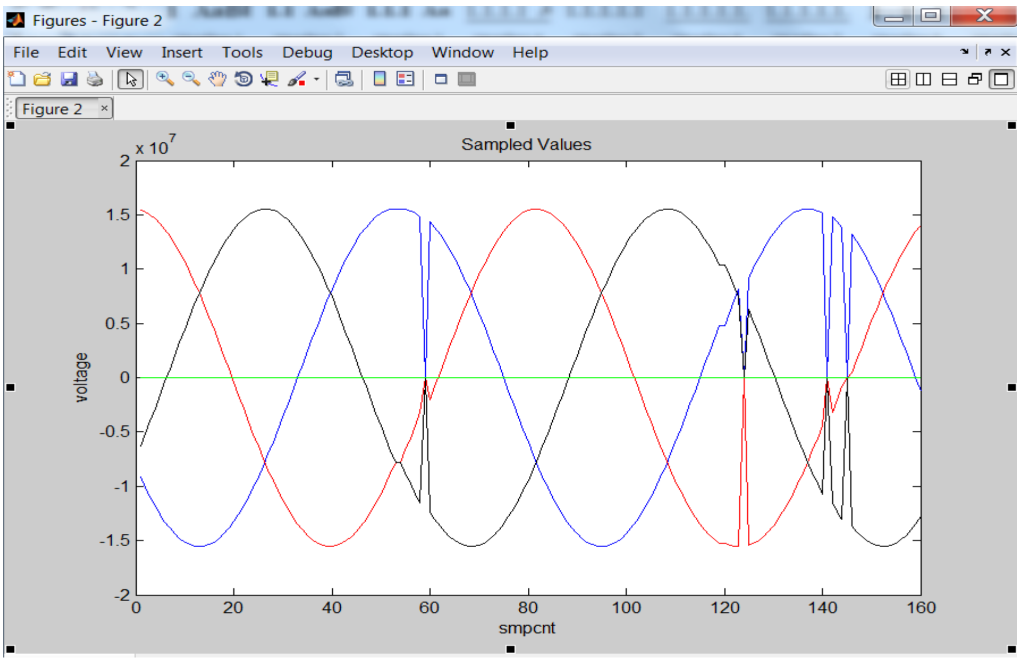


Figure 44. Voltage waveform of the output data from Omicron 850.

Comparing the voltage waveform from Omicron CMC 850 device in **Figure 41** and the plot of the output voltage in **Figure 44**, the output shows the same pattern and values. The MATLAB plot shows some error point which are frames that do not follow the frame format defined in IEC 61850 9–2.

5. CONCLUSION AND FUTURE WORK

The work done in this Thesis is in two parts; First is the theoretical part which include the explanation of fundamental communication solutions and protocols, the basic concept of IEC 61850 and also the packet format for Sampled Value communication as defined in IEC 61850–9–2 standard document. The second part is the development of an application based on WinPcap API to extract data from the network capture (PCAP file) which is based on IEC 61850 9–2 standard format. Four currents and four voltages are extracted and formatted according to the requirements of the PSCAD.

The thesis is part of the Sundom Smart Grid project and it integrates the data from the real electric power network in Sundom village, to the Protection, Automation and Communication laboratory and Smart Grid test environment at the University of Vaasa. The computer program is an efficient technique to extract data from a PCAP file and organize in any format needed for follow–on tools. The output data can be ported to MATLAB for further simulations or analysis.

The output obtained from the software was confirmed to be correct by comparing the values with that on the Wireshark capture. This is confirmed from the plots of the data extracted using MATLAB when compared to that generated from Omicron device. The MATLAB plot shows some error point which are frames that do not follow the frame format defined in IEC 61850 9–2.

As part of future work, it is important to develop an algorithm to estimate the value of erroneous, lost or delayed Sampled Value packet. Also, in cases where the sample count is not sequential, there will also be a need to develop an algorithm to reorder it based on the sample count to get the correct stream of data.

REFERENCES

Ali, Ikbal., Thomas, Mini. & Gupta, Sunil. (2013). *Integration of PSCAD based Power System & IEC 61850 IEDs to Test Fully Digital Protection Schemes*. IEEE Innovative Smart Grid Technologies – Asia (ISGT Asia). Bangalore. November 10 – 13 2013. Pages 1–6. ISBN 978–1–4799–1346–6

Baranov, Pavel F., Muravyov, Sergey V., Sulaymanov, Almaz O. & Khudonogova, Lyudmila I. (2013). *Software for Emulating the Sampled Values Transmission in Accordance with IEC 61850*. In: 2nd International Symposium on Computer, Communication, Control and Automation (3CA 2013), pages 478–481. Atlantis Press

Budka, Kenneth C., Deshpande, Jayant G. & Thottan, Marina (2014). *Communication Networks for Smart Grids – Making Smart Grid Real*. London: Springer. Pages: 369. ISBN: 978 – 1 – 4471 – 6301 – 5

CACE Technologies (2005). *Using WinPcap in your Programs* (2005–2007) [Online].[cited: 14 July 2015].Available from World Wide Web: <URL: https://www.winpcap.org/docs/docs_40_2/html/group_wpcapsamps.html

Carvallo, Andres & Cooper, John (2011). *The Advanced Smart Grid edge power driving sustainability*. Norwood: ARTECH House. 237 p. ISBN 13: 978–1–60807–127–2.

Christian Söderbacka (2013). *The GOOSE Protocol.* [Online] Master's Thesis, University of Vaasa, Finland [cited 2 August 2015]. Page 15. Available from World Wide Web: URL: http://www.tritonia.fi/fi/e-opinnaytteet/tiivistelma/5225/The+GOOSE+Protocol>.

Eero Lukin (2014). *Smart Grid Pilot Project in Vaasa*. [Online]. [cited 12 July 2015]. Available from World Wide Web: <URL: http://www.tekes.fi/en/whats-goingon/news/smart-grid-pilot-project-in-vaasa/>.

European Union Smart Cities (2015). *Smart Grid pilot project*. [Online].[cited 11 July 2015]. Available from World Wide Web: <URL: https://eusmartcities.eu/place/ vaasa>.

Forouzan, Behrouz A. (2010). *TCP/IP Protocol Suite*. New York: McGraw – Hill. Pages: 979. ISBN: 978–0–07–016678.

Fowler, Charles A. & Hammell II, Robert J. (2014). *Converting PCAPs into Weka minable data*. Proceedings of the 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/ Distributed Computing (SNPD). Las Vegas. June 30 – July 2 2014. Page1–6.

Golshani, Mohammad., Taylor, Gareth A., & Pisica Ioana. (2014). *Simulation of Power System Substation Communications Architecture Based on IEC 61850 Standard*. Proceedings of the IEEE 49th International Universities Power Engineering Conference (UPEC). Cluj – Napoca. September 2 – 5 2014. Pages 1–6. ISBN: 978–1–4799–6556–4

Goraj Maciej & Herrmann Juergen (2007). Experience in IEC 61850 and possible improvements of SCL Language. Praxis Profiline [Online] [cited 21 August 2015], Available from Internet: <URL: http://www.sisconet.com/downloads/GE_E_2007.pdf

Honeth, Nicholas., Khurram, Zeeshan Ali,. Zhao, Pencheng & Nordström, Lars (2013). *Development of the IEC 61850 – 9 – 2 Software Merging Unit IED Test and Training Platform*. IEEE PowerTech (POWERTECH) Conference. Grenoble. June 16 – 20 2013. Pages 1–6.

IEC TC57 (2006). *Substation Configuration Language (summary)*. [Online][cited 21 August 2015], Available from Internet: <URL: http://cimug.ucaiug.org/Harmonization%20Documents/EPRI%20Harmonization%20Project%20Notes%20and%20Minutes/IEC%20TC57%20Substation%20Configuration%20Language%20Summary.pdf

IEC 61850 (2003). *Technical report on communication networks and systems in substation* – International Electrotechnical Commission (IEC). First edition.

Ingram, David M. E., Schaub, Pascal., Taylor, Richard R. & Campbell, Duncan A., (2013). *Performance Analysis of IEC 61850 Sampled Value Process Bus Networks*. IEEE Transactions on Industrial Informatics. August 16 2013. Vol 9, Issue 3. pages 1445–1454. ISSN 1551–3203.

Janssen, Marco (2010). *IEC 61850*. In: Curso Smartgrids Cocier–Cno–Cac [Online]. Bogota, D.C.:[24 August 2015]. Available from World Wide Web: URL:http://www.energiamayorista.com.co/memorias_SG/taller/IEC61850_1.pdf

Kanabar, Mitalkumar G., Sidhu, Tarlochan S. (2011). *Performance of IEC 61850–9–2 Process Bus and Corrective Measure for Digital Relaying*. IEEE Transactions on Power delivery. April 2011. Pages 725–735. ISSN: 0885–8977.

Kim, Y.K, Han, J.K, Lee, Y.J, An, Y.H & Song, I.J. (2011). *Development of IEC 61850 Based Substation Engineering Tools with IEC 61850 Schema Library*. Scientific Research [Online] 2:3 [cited 20 August 2015]. Available from internet: <URL:http://www.scirp.org/journal/sgre/

Konka, Jakub W., Arthur, Colin M., Garcia, Francisco J., Atkinson Robert C. (2011). *Traffic Generation of IEC 61850 Sampled Values*. Proceedings of IEEE First International Workshop on Smart Grid Modeling and Simulation. Brussels. October 17 2011. Pages 43 – 48. Print ISBN: 978–1–4673–0194–7.

Leon – Garcia, Alberto & Widjaja, Indra (2000). *Communication Networks – Fundamentals Concepts and Key Architectures.* Singapore: McGraw – Hill. Pages 867. ISBN 0–07–116840–0

Liang, Yingyi. Campbell, Roy H. (2008). *Understanding and Simulating the IEC 61850 Standard*. Research and Tech Reports – Computer Science, University of Illinois at Urbana – Champaign [online] :12 [cited 5 May 2015]. Available from Internet: <URL: https://www.ideals.illinois.edu/handle/2142/11457>.

Liu, Yiqing., Gao, Houlei., Xiang, Mingjiang., Wei, Xin., Wei, Peng., Zhou, Chunsheng (2011). *Performance Testing of Complete Digital Relays Based on ATP – EMTP and IEC 61850 – 9 – 2*. Proceedings of IEEE 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT). Weihai. July 6 – 9 2011. Pages 83–87. ISBN: 978–1–4577–0364–5.

Manitoba HVDC Research Centre (2010). *User's guide on PSCAD – Power Systems Computer Aided Design.* [Online] [Cited: 2 March 2015]. Available from World Wide Web: <URL: https://hvdc.ca/uploads/ck/files/reference_material/PSCAD_User_Guide_v4_3_1.pdf>.

NettedAutomation (2002). The MMS Client/Server model [online] [cited 10 Aug. 2015]. Available from the internet: <URL: http://www.nettedautomation.com/standardization/iso/tc184/sc5/wg2/mms_intro/intro3.html>.

Peter, Crossley., Li, Yang., An, Wen., Ray, Chatfield., Miles Redfern & Xin Sun (2011). *Design and performance evaluation for a protection system utilizing IEC 61850 – 9 – 2 process bus*. Proceedings of IEEE International Conference on Advanced Power System Automation and Protection (APAP). Beijing. October 16 – 20 2011. Vol 1. Pages 534 – 538.

Roostaee , S., Hooshmand, R. & Ataei, Mohammad (2011). *Substation Automation System Using IEC 61850*. Proceedings of IEEE 5th International Power Engineering and Optimization Conference (PEOCO2011). Selangor. June 6 – 7 2011. Pages 393 – 397. ISBN: 978–1–4577–0355–3

Schwarz Karlheinz (2004). *What is a Logical Node*. [Online].[cited 17 August 2015]. Available from World Wide Web: <URL: http://www.nettedautomation.com/download/What-is-a-Logical-Node_2004-08-12.pdf>.

Triangle MicroWorks.Inc (2013). *IEC 61850 Training Videos*. [online] [cited 5 March 2015]. Available from World Wide Web: <URL: www.trianglemicroworks.com/video/videoplaylists/iec-61850-training >.

United States Department of Energy, Office of Electric Transmission and Distribution (2003). *"GRID 2030" A National vision for electricity's second 100 years.* [online]. Available from World Wide Web: <URL: http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/Electric_Vision_Document.pdf >.

Zhang Hao, Chen Wen, Zhang Yuan & Yang Jing (2012). *Make use of WinPcap and ARP cheats to carry out a network wiretap monitoring*. Proceedings of IEEE International Conference on Computer Science and Electronics Engineering (ICCSEE). Hangzhou. March 23 – 25 2012. Vol.2. Pages 56 – 58. ISBN 978-1-4673-0689-8

APPENDIX 1. United States Energy In Independence and Security Act (EISA) of 2007 (Enrolled Bill [Final as passed Both House and Senate] - ENR)

TITLE – SMART GRIDS

SEC.1301. Statement of policy on Modernization of Electricity Grid

It is the policy of the United States to support the modernization of the Nation’s electricity transmission and distribution system to maintain a reliable and secure electricity infrastructure that can meet future demand growth and to achieve each of the following, which together characterize a Smart Grid:

(1) Increased use of digital information and controls technology to improve reliability, security, and efficiency of the electric grid.
(2) Dynamic optimization of grid operations and resources, with full cyber – security.
(3) Deployment and Integration of distributed resources and generation, including renewable resources.
(4) Development and incorporation of demand response, demand – side resources, and energy – efficiency resources.
(5) Deployment of “smart” technologies (real – time, automated, interactive technologies that optimize the physical operation of appliances and consumer devices) for metering, communications concerning grid operations and status, and distribution automation.
(6) Integration of “smart” appliances and consumer devices.
(7) Deployment and integration of advanced electricity storage and peak – shaving technologies, including plug – in electric and hybrid electric vehicles, and thermal – storage air conditioning.
(8) Provision of consumers of timely information and control options.

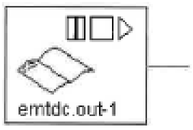
(9) Development of standards for communication and interoperability of appliances and equipment connected to the electric grid, including the infrastructure serving the grid.

(10) Identification and lowering of unreasonable or unnecessary barriers to adoption of Smart Grid technologies, practices and services.

APPENDIX 2. File read and Data file format in PSCAD

File Read

Description
Input Parameters



Description

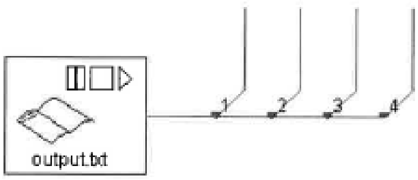
This component can be used to read pre-formatted, columnar text data from a file, and then input this data directly into a PSCAD simulation. The data file may include up to 11 columns of data, each column containing information representing an individual scalar control signal.

Before the data file can be used within a simulation project, the data sampling rate (or time step) must be somehow related to the simulation environment. This is accomplished in one of two ways:

1. **Pre-Defined Time Step**: Indicate that the first column of data is the simulation time in seconds (which leaves a maximum of 10 columns remaining for data).
2. **Data Sampling Rate**: Indicate a sampling frequency (no time column).

If the time step in the time column of the data file (or the sampling frequency for that matter) is not an integer multiple of the PSCAD simulation time step, then the component will linearly interpolate the data to a value corresponding to the PSCAD simulation time step.

The output of this component is provided to the PSCAD simulation as an output data vector, where each element of the output array represents the corresponding single column of the input data file. If you intend to inject the data from a particular column (or columns) in the file, then you must tap the output array using the Data Signal Array Tap component, as shown below for a 4 column data file.



NOTE: If the input file contains only one column of data (not including time), then the output signal will be a scalar (i.e. dimension 1). Therefore, a Data Signal Array Tap is not required in this case.

More: Data File Format

Input Parameters

Configuration

Data File Format

End of File

The File Read component allows you to read up to 11 data signals (columns in the data file) simultaneously. The data file itself must conform to certain structured format.

An example data file is given below for a 4 signal data file (first column is time):

```
! The first line must be either blank OR a comment as shown here.
  0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00
  1.0000000E-03   2.828947      0.2618097      0.1476223
  2.0000001E-03   4.408800       1.089826      0.3367114
  3.0000000E-03   4.545383       2.462704      0.5100455
  4.0000002E-03   3.334809       4.172538      0.6182301
  5.0000004E-03   1.110011       5.889144      0.6278525
  6.0000005E-03  -1.639397       7.239620      0.5262269
  7.0000007E-03  -4.360832       7.888721      0.3224114
  8.0000004E-03  -6.532493       7.607064      4.4689782E-02
  9.0000005E-03  -7.744767       6.317216     -0.2648376
  1.0000001E-02  -7.759348       4.111763     -0.5575674
  1.1000001E-02  -6.539557       1.242012     -0.7864339
  1.2000001E-02  -4.249408      -1.919620     -0.9134614
  1.3000001E-02  -1.223144      -4.936765     -0.9157405
  1.4000001E-02   2.089199      -7.376456     -0.7890044
  1.5000002E-02   5.191667      -8.876119     -0.5483486
  1.6000001E-02   7.617365      -9.200063     -0.2260490
  1.7000001E-02   8.997789      -8.277058      0.1331513
  1.8000001E-02   9.116805      -6.213536      0.4785683
  1.9000001E-02   7.941910      -3.280582      0.7609522
  2.0000001E-02   5.628680      0.1233175      0.9396424
```

NOTE: The first data point in the file must correspond to time t = 0.0 (TIMEZERO) of the simulation.

End of File

The File Read component provides three options for what to do once the end of the data file is reached:

- **Output the last read values**: The component will continue to output the data given in the last line of the file, once the end of the file is reached
- **Rewind and replay again**: The component will 'rewind' the file and replay again starting from the first line. It assumes that the time interval between the last and the first lines is the same as that between the one before the last and the last lines.
- **Extrapolate**: The component will continue to extrapolate the data based on the last 2 sets of data, once the end of the file is reached

APPENDIX 3. Tools used to capture and analyze Sampled Value packet

1. WinPcap

WinPcap, the Windows version of libpcap in UNIX – system is a powerful industry accepted standard for link–layer network analysis. (Riverbed Technology 2013) with extensible architecture that affords applications the opportunity to capture and transmit data packets without the Windows' TCP/IP stack, WinPcap makes the development of applications with real network traffic simulation achievable (Wang & Zhang 2013: 94–95). It is an open source library built on Win32 platform used in acquisition and analysis for packets and has features such as packet filtering at the kernel–level and support for remote capture. The extensible architecture makes it possible for the operating system to implement low–level network access while the library helps to connect to the low – level network layers. (Riverbed Technology 2013)

WinPcap is made up of three parts as can be seen from **Figure 1**. These are:

1. Network Interface Card (NIC) driver
2. Kernel – level Net group Packet Filter (NPF)
3. User – level Application Programming Interface (API)

These parts work together to achieve network packet capture, packet injection, packet dump as well as network analysis.
As can be observed from **Figure 1** packets from the network is received by the NIC and moved to NIC buffer. NIC also produces an interrupt which activates the interrupt service routine (ISR) of the NIC driver. Furthermore, the ISR programs a function to inform the packet capture drivers of the upper layer about the reception of a packet and also to process hardware requests (Fang & Zeng 2013: 555-556).
The NPF at the Kernel – level which interacts directly to driver of the NIC is responsible for packet capture, monitoring and transmitting packets. It has a programmable filter system which drops packets that are not required based to the rules specified by the user. All other packets are stored in the kernel buffer so that any

application that needs it can access it through the user defined buffer in the user level of the WinPcap architecture (Fang et al. 2013: 555).

The User level consists of the Packet.dll module, Wpcap.dll module and also the user code. Packet.dll is a dynamic link library which provides an interface to connect to the low level services of WinPcap in a way that it provides a system – independent Application Programming Interface (API). This is particularly important because various versions of windows operating system (OS) offer separate interfaces between the kernel and the user levels of WinPcap. Thus, the running Packet.dll on different versions of Windows operating system is possible without recompiling (Jiang, Yang & Li 2014: 1–4). It is responsible for installing, starting and stopping the NPF device driver, transmission and reception of packets to and from the driver and obtaining the list of available network adapters. (CACE Technologies 2005)

Wpcap.dll is also a dynamic link library which offers a higher level interface to the developers; it is not dependent on the OS. It is designed based on libPCAP and provides functions which work independent of the adapter type and the OS. It has a driver that extends the OS and provides a set of high level libraries.

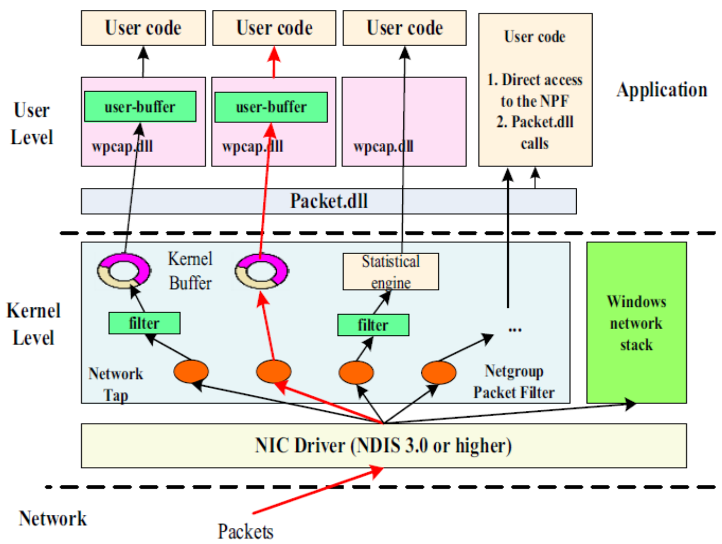


Figure 1. WinPcap Architecture (Fang et al. 2013: 556)

WinPcap features make it a versatile tool for packet capture and filtering in numerous networks tools such as network monitors, protocol analyzers, sniffers, network testers and also network intrusion systems. These tools include Wireshark, Snort and Nmap. (Yang, Wang & Ding 2014: 223)

2. Wireshark

Wireshark is an open source tool for protocol analysis and network monitoring; it captures packets and analyses the frames transmitted over a network. Wireshark offers multi – platform support and is considered the best network protocol analyzer that works well on both Linux and Windows platform. It is a based on Libpcap (Luo, Dong & Jia 2010: 291–294) and is capable of analyzing about 1000 protocols and 141000 fields (Sahin, Özcelik, Balta & Iskefiyeli 2013: 88–91). The Wireshark is designed to open and save the captured packet data, show comprehensive protocol information of packets as well as export and import data to and from other programs. It offers a filtering system based on set conditions, also searches for packets based on predetermined requirement and generates statistics from the packet capture (Pruthvi, Bhuvaneswari & Sudheendran 2013: 1–8).

Furthermore, it can capture from various types of network hardware, can stop the capture process on various predefined interrupts such as capture time and number and can concurrently display packets that have been unraveled while Wireshark continues with the capturing process (Pruthvi et al. 2013: 1–8). **Figure 2** shows some features of the Wireshark and its representation on the Wireshark interface this include: command menus, display filter specification, list of captured packets, details of captured packet header that are selected and the packet content in hexadecimal and ASCII format.

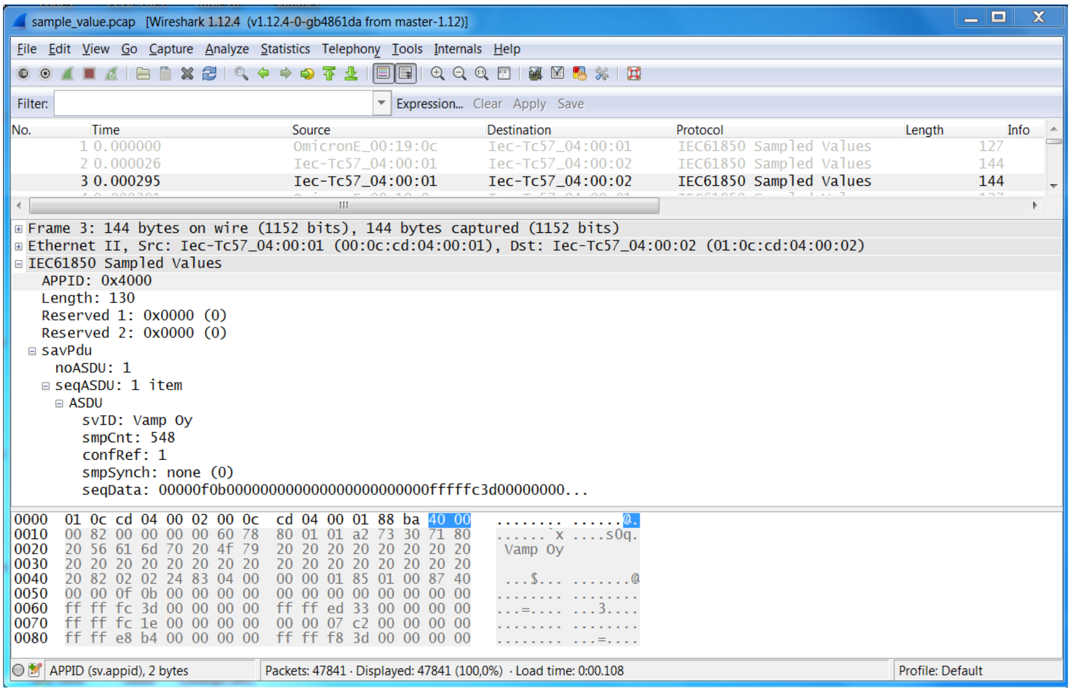


Figure 2. Screenshot of Wireshark interface

Wireshark Architecture

Six functional modules are included in the Wireshark architecture as shown in **Figure 3**. These are the data capture interface, graphical user interface (GUI), capture, packet analysis engine (Epan), GTK and Core.

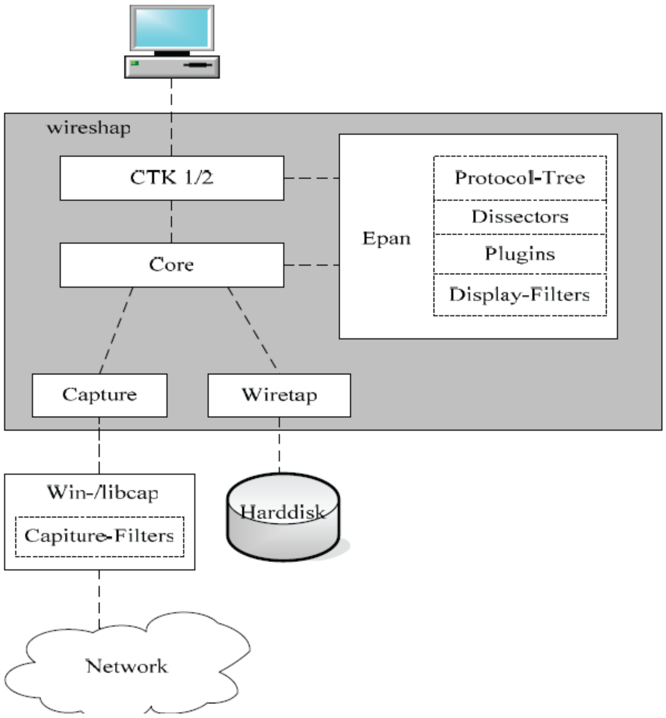


Figure 3. Function module of Wireshark (Luo, et al. 2010: 291–294).

The development of the capture module which includes the packet filtering engine is based on Libpcap/WinPcap and does not depend on of the system platform. GTK module is a toolkit for developing graphical user interfaces that can work on variety of platforms (The GTK+ Project: 2015). The core module is the unit that provides the necessary connections between other functional entities of the system while the Epan module which comprises of plugins, dissectors, display filters and protocol tree as seen in **Figure 3** completes the protocol analysis (Luo, et al. 2010: 291–294). The output of the process is seen on the graphical user interface (GUI).

The Wireshark is a useful tool to network administrators, network security engineers, software engineers, researchers and a host of others. It has found application in troubleshooting network issues, investigating security challenges in a network, debugging protocol implementations and analyzing network traffic (Pruthvi et al. 2013: 1–8).

REFERENCES

CACE Technologies (2005). *WinPcap documentation: Packet.dll – Packet Driver API (WinPcap internals)* (2005–2007) [Online]. [cited: 14 July 2015]. Available from World Wide Web: <URL: https://www.winpcap.org/docs/docs_40_2/html/group__packetapi.html>.

Feng Luo, Ligang Dong & Fenggen Jia (2010). *Method and Implementation of Building ForCES protocol Dissector Based on Wireshark*. Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME). Chengdu. April 16 – 18 2010. Pages 291–294. E-ISBN 978-1-4244-5265-1.

The GTK+ Team (2015). *The GTK+ Project* 2007 – 2015. [Online]. [cited 24 May 2015]. Available from World Wide Web: <URL:http://www.gtk.org/>.

The GTK+ Project Team (2015). What is GTK+, and how can I use it. 2007 – 2015 [online]. [Cited 28 May 2015]. Available from World Wide Web: < URL: http://www.gtk.org/

Lili Jiang, Xiaohui Yang & Tao Li (2014). *The analysis and design for a network protocol analysis system based on WinPcap*. Proceedings of IET Communications Security Conference (CSC 2014). Beijing. May 22 – 24 2014. Pages 1–4.

Pruthvi P, Bhuvaneswari H.B. & Sudheendran L (2013) *Analysis of utility communication protocol IEC 61850 for substation automation systems.* IET national conference on challenges in research & technology in the coming decades (CRT 2013). Ujire. September 27 – 28 2013. Pages 1 – 8. E-ISBN 978-1-84919-868-4

Riverbed Technology (2013). *Introduction to WinPcap*. [Online] [cited 16 March 2015]. Available from World Wide Web: <URL: http://www.winpcap.org/>.

Sahin V.H, Özcelik S.I, Balta M & Iskefiyeli M (2013). *Topology discovery of Profinet Networks using Wireshark*. IEEE International conference on Electronics, Computer and Computation (ICECCO). Ankara. November 9 2013.Pages 88-91.

Zhenqi Wang & Dankai Zhang (2013). *Research on WinPcap capture IPv6 packet method*. Proceeding of IEEE International conference on Computer Sciences and Applications (CSA). Wuhan. December 14 – 15 2013. Pages 94–97.

Dongxiang Fang & Peifeng Zeng (2103). *An architecture of virtual NIC driver based on WinPcap and a method to test it*. Proceeding of 15th IEEE International Conference on Communication Technology (ICCT). Guilin. November 17–19 2013. Pages 555–559.

Mingji Yang, Yizhe Wang & Hui Ding (2014). *Design of WinPcap based ARP spoofing defense system.* IEEE Computer society fourth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC). Harbin. September 18 – 20 2014. Pages 221–225.

APPENDIX 4. The computer program

```
/*Computer program to access, extract and format data
from a pcap file and write into an output file in the
format required by PSCAD. This is part of the Sundom
Smart Grid project.*/


#include <string>
#include <iostream>
#include <pcap.h>
#include <fstream>
#include <ios>
#include <sstream>
#include <math.h>

using namespace std;

char buff[17];
char current1_buff[4];
#define LINE_LEN 16


const u_char *data; // creating a character array
u_char *readData;


long long currentA; //Output for current on phase A
long long currentB; //Output for current on phase B
long long currentC; //Output for current on phase C
long long currentN; //Output on the neutral current line
long long voltageA; //Output for voltage on phase A
long long voltageB; //Output for voltage on phase B
long long voltageC; //Output for voltage on phase C
long long voltageN; //Output on the neutral voltage line

struct pcap_pkthdr *header;
string row;

int  readFileCont();
string buildRow();
void writeRowToFile(string, long x);
int pageCommentCounter = 0;


//function definition
long int hex2dec(char * in);
unsigned int power(int base, int power);

int main(int argc, char *argv[])
{

	if(pageCommentCounter==0)
	{
```

```
		row += "Smpcnt	    iA		  iB		iC
	iN		vA		vB		vC		vN\n";
	}
	else
		row = " ";
	pageCommentCounter++;

	readFileCont();
	system("PAUSE");
}


int readFileCont(){

	string file = "C:\\sundom\\realdata\\SundomIED5data_b2.pcap"; // 
getting the file
	char errbuff[PCAP_ERRBUF_SIZE];				// size to use 
when allocating the buffer that contains the libpcap errors
	pcap_t * pcap;
	if((pcap = pcap_open_offline(file.c_str(), errbuff)) == NULL)	// 
open the file and store the result in pointer to pcap_t file.c_str()
		return -1;

	int returnValue;
	struct pcap_pkthdr *header; // create a header object (ts-time stamp, 
caplen-length of portion present, len-length of this packet (offwire)


	// loop through packets and print to screen
	u_int i = 0; // i is packetCount
	while (int returnValue = pcap_next_ex(pcap, &header, &data) >= 0) // 
read a packet from an interface or offline capture
	{

		printf("packet # %i\n", ++i);	//show packet number
		printf("packet size: %d bytes\n", header->len); //show the size 
in bytes of the packet
		printf("Output Time: %d:%d seconds\n",header->ts.tv_sec, header-
>ts.tv_usec); //Output Epoch time

		if (header->len != header->caplen) // show warning if length 
captured is different
			printf("warning! Capture size different from packet size: 
%ld bytes\n", header->len);


		static unsigned long smpCnt = -1;
			smpCnt++;
			printf("smpCnt is: %lu\n",smpCnt);


		string row = buildRow();
		//giving the created row to the writeRowToFile function

		writeRowToFile(row, smpCnt);
```

```
                //row = " ";
        }
}

//This function takes a string and writes it to a file  (the output file)
//added the new line character to the row built, so when this function can
write it at once.

void writeRowToFile(string data, long sampleCount){

        fstream fileStream;
        fileStream.open("output.txt", ios::app); //opening the file for
writing, with appending not over-writing
        fileStream <<sampleCount << "          " << currentA << "          " <<
currentB << "       " << currentC << "        " << currentN << "          " <<
voltageA << "       " << voltageB << "        " << voltageC << "          " <<
voltageN << endl; //writing to the file
        fileStream.close();
}

//this function builds one row to be written on the file
//it appends the Current, Voltages and tab to the row string and at the end
of the row it appends new line character
//e.g. 12    3      &      \n

string buildRow(){

        int current;
        string tab = "\t";


        /* Current in phase A */
        int a,b;
        for( b = 61,a=0; b<65; b++,a++) // Selection if bytes 61 to 64 which
is the representation of current in phase A based on IEC 61850-9-2
        {
                current1_buff[a]= data[b];
                printf("%02X", data[b]);
        }
        current1_buff[a]= '\0';
        printf("\n");
        //Here is where the conversion of hexadecimal to decimal

        currentA = hex2dec(current1_buff);
        printf(" Decimal Number is %u  \n", currentA);
        system("PAUSE");
        printf("\n");
        row += tab;



        /* Current in phase B */
        int c,d;
        for( d = 69,c=0; d<73; d++,c++) // Selection if bytes 69 to 72 which
is the representation of current in phase B based on IEC 61850-9-2
        {
```

```
        current1_buff[c]= data[d];
        printf("%02X", data[d]);
    }
    current1_buff[c]= '\0';
    printf("\n");

    //Here, conversion of hexadecimal representation from the capture to
decimal is done
    currentB = hex2dec(current1_buff);
    printf(" Decimal Number is %u  \n", currentB);
    system("PAUSE");
    printf("\n");
    row += tab;



    /* Current in phase C */
    int e,f;
    for( e = 77,f=0; e<81; e++,f++)
    {
        current1_buff[f]= data[e];
        printf("%02X", data[e]);
    }
    current1_buff[f]= '\0';
    printf("\n");

    currentC = hex2dec(current1_buff);
    printf(" Decimal Number is %u  \n", currentC);
    system("PAUSE");
    printf("\n");
    row += tab;



    /* Neutral Current */
    int g,h;
    for( g = 85,h=0; g<89; g++,h++)
    {
        current1_buff[h]= data[g];
        printf("%02X", data[g]);
    }
    current1_buff[h]= '\0';
    printf("\n");

    currentN = hex2dec(current1_buff);
    printf(" Decimal Number is %u  \n", currentN);
    system("PAUSE");
    printf("\n");
    row += tab;
    //return row;



    /* Voltage in phase A */
    int i,j;
    for( i = 93,j=0; i<97; i++,j++)
```

```
{
        current1_buff[j]= data[i];
        printf("%02X", data[i]);
}
current1_buff[j]= '\0';
printf("\n");

voltageA = hex2dec(current1_buff);
printf(" Decimal Number is %u  \n", voltageA);
system("PAUSE");
printf("\n");
row += tab;



/* Voltage in phase B */
int k,l;
for( k = 101,l=0; k<105; k++,l++)
{
        current1_buff[l]= data[k];
        printf("%02X", data[k]);
}
current1_buff[l]= '\0';
printf("\n");

voltageB = hex2dec(current1_buff);
printf(" Decimal Number is %u  \n", voltageB);
system("PAUSE");
printf("\n");
row += tab;



/* Voltage in phase C */

int m,n;
for( m = 109,n=0; m<113; m++,n++)
{
        current1_buff[n]= data[m];
        printf("%02X", data[m]);
}
current1_buff[n]= '\0';
printf("\n");

voltageC = hex2dec(current1_buff);
printf(" Decimal Number is %u  \n", voltageC);
system("PAUSE");
printf("\n");
row += tab;



/* Neutral Voltage */
int o,p;
for( o = 117,p=0; o<121; o++,p++)
{
```

```
                current1_buff[p]= data[o];
                printf("%02X", data[o]);
        }
        current1_buff[p]= '\0';
        printf("\n");

        voltageN = hex2dec(current1_buff);
        printf(" Decimal Number is %u  \n", voltageN);
        system("PAUSE");
        printf("\n");
        row += tab;
        return row;
}

long int hex2dec(char * in)
{
 int n=0,N;
 unsigned char temp;
 unsigned long int out=0;
 n=sizeof(in)-1;

 N=n;
 while(n>-1)
 {
        temp= in[n] & 0xFF;
        out= out | (temp << ((N-n)*8));
      n--;
 }
 //system("PAUSE");
 return out;
}
```