INT J COMPUT COMMUN, ISSN 1841-9836
8(5):708-721, October, 2013.

A Detailed Analysis of the GOOSE Message Structure in an IEC 61850 Standard-Based Substation Automation System

C. Kriger, S. Behardien, J. Retonda-Modiya

Carl Kriger, Shaheen Behardien,
John-Charly Retonda-Modiya
Centre for Substation Automation and Energy Management Systems
Cape Peninsula University of Technology
South Africa, P.O Box 1906, Bellville, Cape Town, 7535
krigerc@cput.ac.za, behardiens@cput.ac.za, retonda7@yahoo.fr

Abstract: In order to implement an IEC 61850 communication system, there needs to be a complete understanding of the methods, tools and technologies associated with the communication network, protocol and messaging underpinning the services. The IEC 61850 standard allows for communication between devices within a substation where a peer-to-peer model for Generic Substation Events (GSE) services is used for fast and reliable communication between Intelligent Electronic Devices (IEDs). One of the messages associated with the GSE services is the Generic Object Oriented Substation Event (GOOSE) message. A detailed analysis of the structure for the GOOSE message is required for fault diagnosis, or when developing hardware that is compliant with the IEC 61850 standard. This is one of the stated objectives of the Centre for Substation Automation and Energy Management Systems (CSAEMS) in the training of prospective specialists and engineers. A case study is presented where the structure of the GOOSE message as described in IEC 61850-8-1 is confirmed using firstly simulation, then experimentation with actual IEDs. In the first instance the message structure is confirmed by simulation of the GOOSE message and capturing it using network protocol analyzer software, after which analysis of the packet frame is performed. Data encoding of the GOOSE Protocol Data Unit (PDU) is analyzed with emphasis on the Abstract Syntax Notation (ASN. 1) Basic Encoding Rules (BER). The second part of the case study is conducted through experimentation with IEDs which are used to generate a GOOSE message and network protocol analyzer software is used to analyze the structure. Both the simulation and practical experimentation with actual devices confirm the GOOSE message structure as specified in part 8-1 of the IEC 61850 standard.
Keywords: IEC 61850, substation automation, Generic Object Oriented Substation Event (GOOSE), Intelligent Electronic Device (IED).

1 Introduction

The International communication standard for devices within a substation environment known as the International Electrotechnical Commission (IEC) IEC 61850 standard has contributed immensely to the way communication and information exchange are implemented within an electrical substation. This fairly new communication standard aims to ensure, amongst other things interoperability among devices from different vendors. For time-critical events such as the protection of electrical equipment, messages known as Generic Object-Oriented Substation Event (GOOSE) messages are exchanged between devices by means of a local Ethernet network. The paper presents a detailed examination and analysis of the captured GOOSE message structure generated firstly by means of simulation of the GOOSE message using software, then using actual protection devices.

The next section contains a brief examination of the GOOSE message followed by the simulation and practical case study setup. The GOOSE message frame is analyzed according to the specification in IEC 61850-8-1 of the standard. The Application Protocol Data Unit and its relevance to the ASN. 1 Basic Encoding Rules is discussed and finally the conclusion and future prospects for this work are presented.

2 GOOSE Message Background

The devices in substations evolved from the older electromechanical relays into the current Intelligent Electronic Devices (IEDs) utilizing embedded microcontroller capabilities with communication between devices. The Ethernet technology was a natural part of this evolution. [1] With the advent of virtual local area networks (VLANs), an increase in data communication speeds, and flow control of switched systems, the Ethernet has become a reliable technology for this type of real-time application. [2]

The IEC 61850 standard allows for two groups of communication services between entities within the Substation Automation System (SAS), (IEC 61850-7-1) as shown in Fig. 1. [3] One group utilizes a client-server model, accommodating services such as Reporting and Remote Switching. The second group utilizes a peer-to-peer model for Generic Substation Event (GSE) services associated with time-critical activities such as fast and reliable communication between Intelligent Electronic Devices (IEDs) used for Protection purposes. In the IEC 61850-8-1 part of the standard, one of the messages associated with the GSE services are the Generic Object Oriented Substation Event (GOOSE) messages that allow for the broadcast of multicast messages across the Local Area Network (LAN). [4]

The Abstract Communications Service Interface (ACSI) shown in Fig. 1 and covered in greater detail in IEC 61850-7-2, defines common utility services for substation devices and shows the two groups of communication services for the client-server model and peer-to-peer model. The GSE model services provides a fast and reliable system-wide distribution of input and output data values; is based on a publisher/subscriber mechanism and, supports the distribution of the same generic substation event information to more than one physical device through the use of multicast/broadcast services (if so required and so engineered through the publish/subscribe mechanism).

Amongst others, two control classes and the structure of two messages are defined in IEC 61850: Generic Object Oriented Substation Event (GOOSE) supports the exchange of a wide range of possible common data organized by a DATA-SET; and Generic Substation State Event (GSSE) provides the capability to convey state change information (bit pairs). [5]

The scope of communication in IEC 61850 and the Logical interfaces are referenced in page 13 of the IEC61850-1 and give an indication that logical interface 8 is used for direct data exchange between bays (Peer-to-Peer), and GOOSE messages may also be passed between the bay and Station bus (level) as shown in Fig. 2. [6]

The Protocol Mapping Profile for the ACSI services in IEC 61850 is shown in Fig. 3. [4] The GOOSE message is associated with three layers of the Open Systems Interconnection (OSI) model, namely the Physical layer, Data-link layer and the Application layer. [4], [5] Page 114 of part IEC 61850-8-1 displays the structure of the GOOSE message and this is the starting point for the investigation and analysis of the GOOSE message structure. [4] To gain insight into the structure of GOOSE messages, a case study is presented in this paper in which a GOOSE message is simulated by means of the OMICRON IEDScout software, captured via network protocol analyzer software - Wireshark, and analyzed relative to the structure as defined in part 8-1

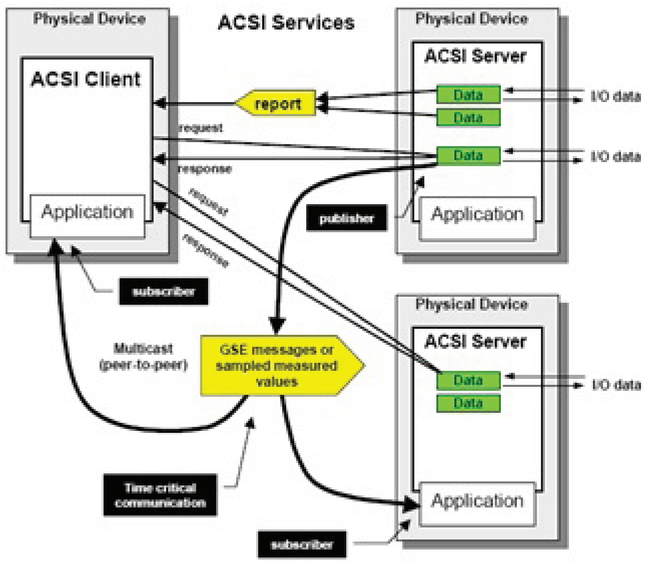


Figure 1: Abstract Communications Service Interface (ACSI) Services IEC 61850-7-1 pg 50 [3]

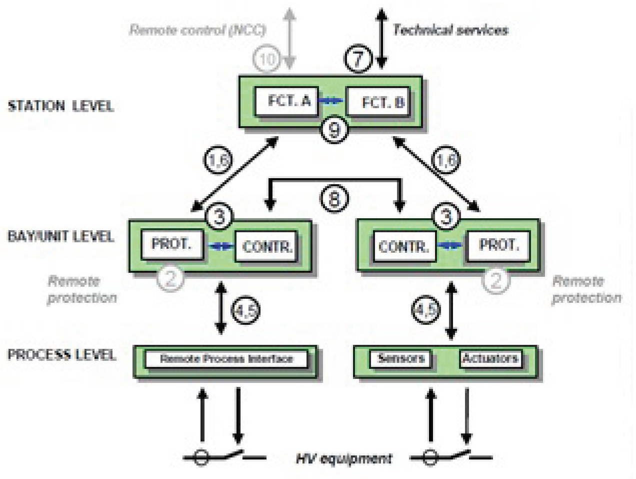


Figure 2: Interface Model of a Substation Automation System (pg 13 of [6])

pages 114-116 of the IEC61850 standard. [4] Further confirmation of the structure of a GOOSE message and the encoding of its data, is obtained through experimentation with actual IEDs from different vendors thus confirming interoperability as well.

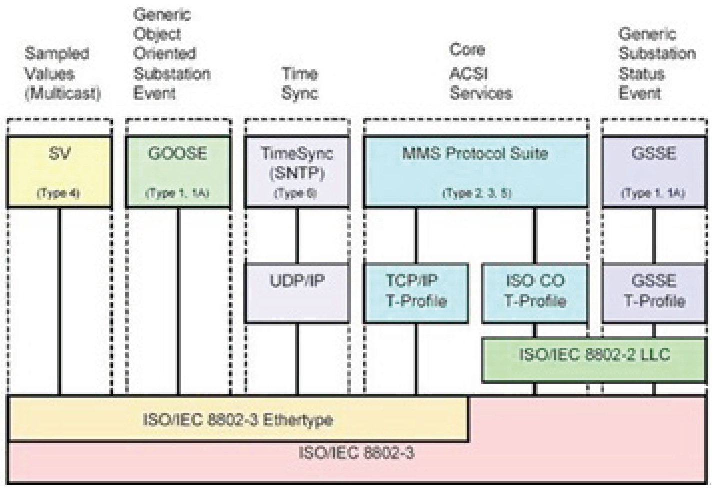


Figure 3: Communication profiles in IEC 61850 (pg 19 of [4])

3 Simulation of the GOOSE message and practical implementation

- Simulation study:

The OMICRON IEDScout software is an invaluable tool for educational purposes as it allows the simulation of the IEC61850 functionality such as GOOSE messaging (via OMICRON hardware platforms), and also has capabilities for monitoring real-time response on the network. The experimental setup for the GOOSE message simulation and validation is illustrated in Fig. 4 where IEDScout is used to generate a GOOSE message within the Personal Computer (PC) through the Network Interface Card (NIC) and Wireshark software running on a notebook is used to capture the GOOSE message packets. Both computers are connected to a network switch.

- Implementation study:

Fig. 5 shows the practical setup for confirmation of the GOOSE message structure by experimentation rather than simulation. IEDs from two different vendors are connected in a local area network via a network switch and GOOSE messages are exchanged between them. The test injection set injects 3-phase current into the Vendor no.1 IED. GOOSE messages are published from the Vendor no. 1 device and subscribed to by the Vendor no. 2 device. The measured 3-phase value currents are displayed on the Vendor no. 2 device front panel. To check for interoperability from both directions, the Vendor 2 device is configured as the publisher and the Vendor no.1 IED as the subscriber. A pushbutton on the Vendor 2 IED is pressed and a sequence of light emitting diodes (LEDs) on the Vendor no. 1 IED is lit. The data contained in the GOOSE messages being exchanged are binary and measurement data. The Wireshark software is used to confirm the structure of the captured message. [7]

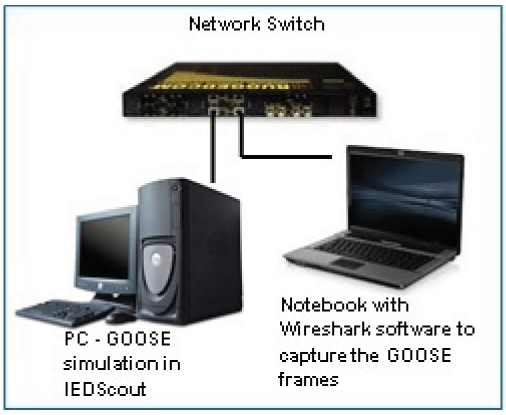

Network Switch
PC - GOOSE simulation in IEDScout
Notebook with Wireshark software to capture the GOOSE frames

Figure 4: Case study - PC with IEDScout and PC with Wireshark Network Analyzer Software

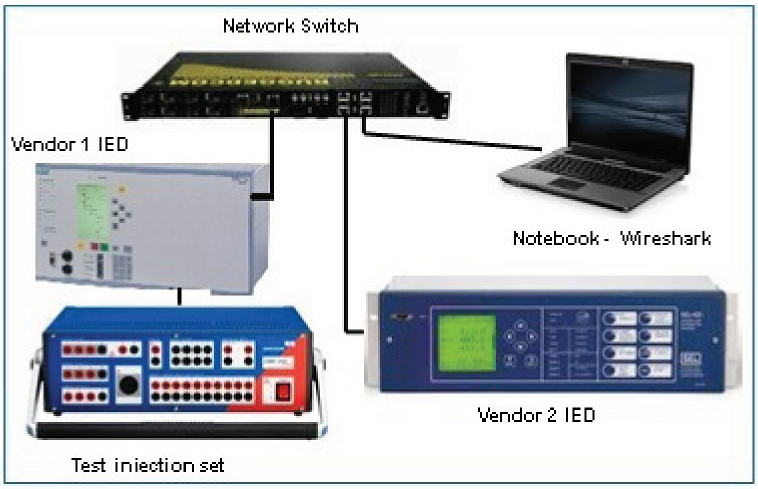
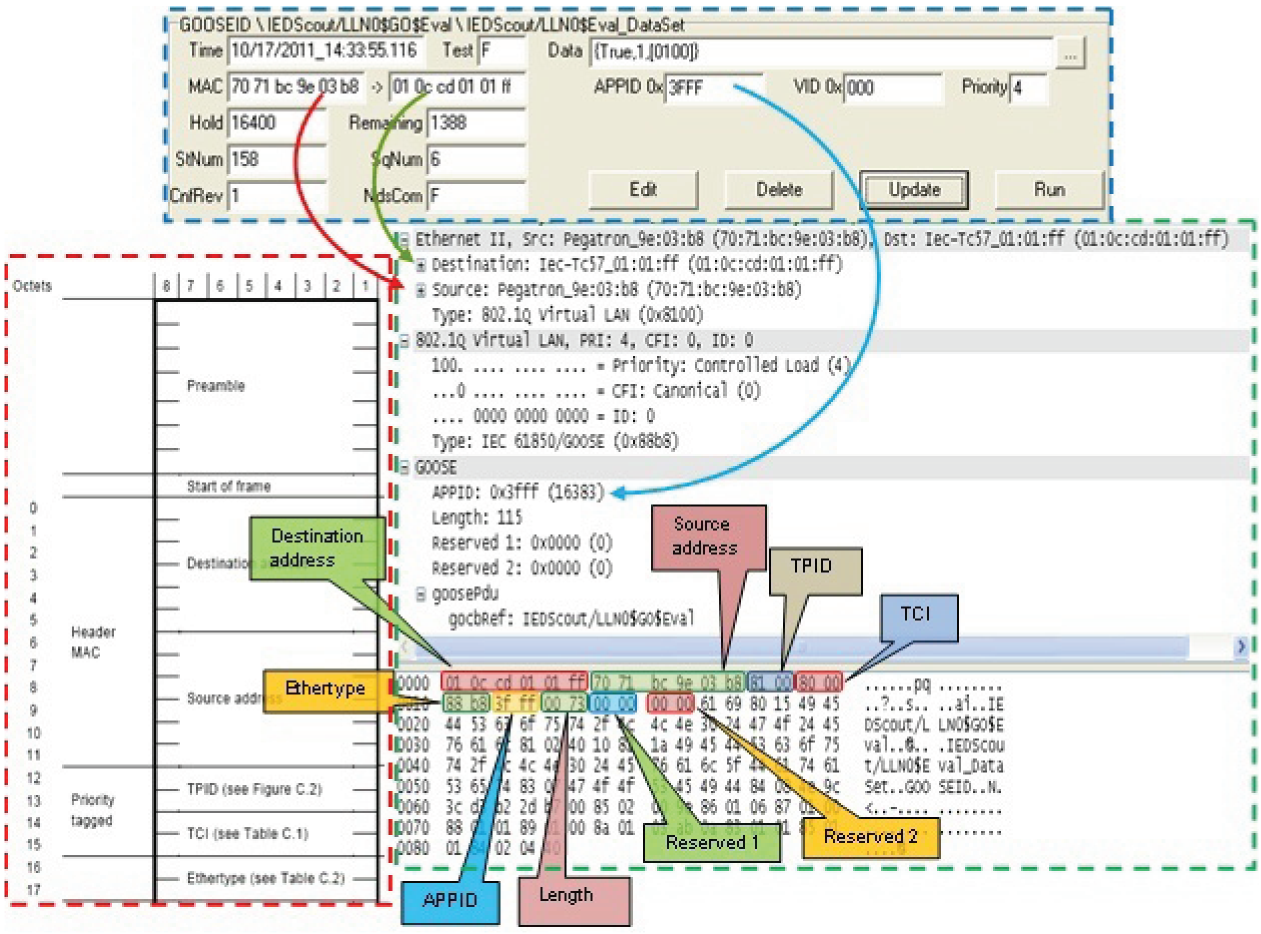


Figure 5: Practical experimental setup with IEDs from different vendors

4 Confirmation of the GOOSE Message Structure

Only the results of the GOOSE messages captured from the simulation case study in Fig. 4 are used in the next section for the message structure confirmation. A portion of the GOOSE message presented in the standard is shown in Fig. 6 on the LEFT hand side (red box). At the top of Fig. 6 (blue box) are the user-defined parameters for the GOOSE message generated by IEDScout software. To the bottom right of Fig. 6 (green box) is the Wireshark capture showing the hexadecimal values in the pane.



GOOSEID \IEDScout/LLN0GOEval \IEDScout/LLN0$Eval_DataSet

Time	10/17/2011_14:33:55.116	Test	F	Data	{True,1,[0100]}
MAC	70 71 bc 9e 03 b8 -> 01 0c cd 01 01 ff	APPID 0x	3FFF	VID 0x	000
Priority	4				
Hold	16400	Remaining	1388		
StNum	158	SqNum	6		
CnfRev	1	NdsCom	F		

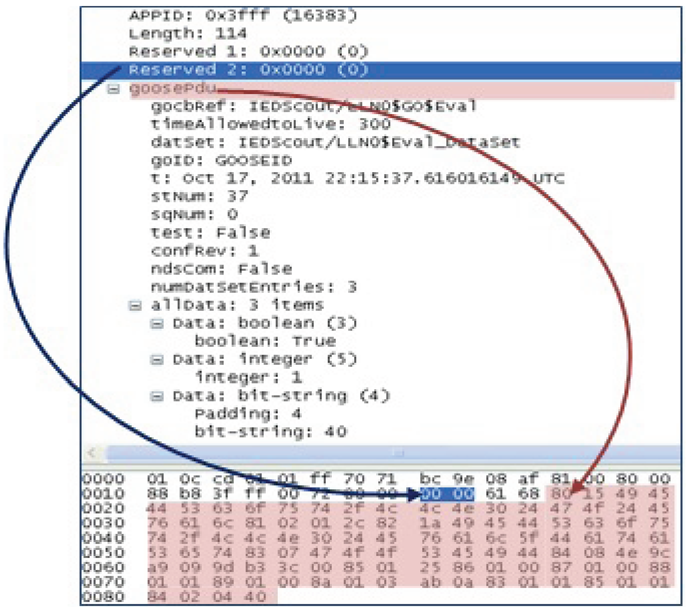
```
⊟ Ethernet II, Src: Pegatron_9e:03:b8 (70:71:bc:9e:03:b8), Dst: Iec-Tc57_01:01:ff (01:0c:cd:01:01:ff)
  ⊞ Destination: Iec-Tc57_01:01:ff (01:0c:cd:01:01:ff)
  ⊞ Source: Pegatron_9e:03:b8 (70:71:bc:9e:03:b8)
    Type: 802.1Q Virtual LAN (0x8100)
⊟ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 0
    100. .... .... .... = Priority: Controlled Load (4)
    ...0 .... .... .... = CFI: Canonical (0)
    .... 0000 0000 0000 = ID: 0
    Type: IEC 61850/GOOSE (0x88b8)
⊟ GOOSE
    APPID: 0x3fff (16383)
    Length: 115
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: IEDScout/LLN0$GO$Eval

0000  01 0c cd 01 01 ff 70 71  bc 9e 03 b8 81 00 80 00   ......pq ........
[illegible]  88 b8 3f ff 00 73 00 00  00 00 61 69 80 15 49 45   ..?..s.. ..ai..IE
0020  44 53 6[illegible] 6f 75 74 2f [illegible]  4c 4e 3[illegible] 24 47 4f 24 45   DScout/L LN0$GO$E
0030  76 61 6[illegible] 81 02 40 10 8[illegible]  1a 49 45 4[illegible] [illegible] 63 6f 75   val..@.. .IEDScou
0040  74 2f [illegible] 4c 4[illegible] 30 24 45  [illegible] 61 6c 5f 4[illegible] [illegible] 74 61   t/LLN0$E val_Data
0050  53 65 [illegible] 83 0[illegible] 47 4f 4f  [illegible] 45 49 44 84 0[illegible] [illegible] 9c   Set..GOO SEID..N.
0060  3c d[illegible] b2 2d [illegible] 00 85 02  [illegible] 86 01 06 87 [illegible] [illegible]   <..-.... ........
0070  88 [illegible] 01 89 [illegible] 00 8a 01  [illegible]                       ........
0080  01 [illegible] 02 04 [illegible]
```

Figure 6: The FIXED section of the GOOSE message structure frame on the bottom left as defined in IEC 61850-8-1 pg 114 of [4]. The simulated GOOSE from IEDScout is at the top and the Wireshark window is at the bottom right

The GOOSE message structure consists of a portion that is fixed in terms of length, and fixed in terms of what content is specified to be there (entered via dialogue box) (Fig. 6). The next section of the message consists of a portion that is variable in terms of both length and content chosen to be communicated, e.g. the user defines which data elements and data attributes are to be transmitted (Fig. 10). Firstly the fixed portion of the message structure is examined and briefly discussed, starting with the Preamble and ending at the Ethertype. The Preamble and Start of frame are performed at the hardware level. The Destination address is a multicast address consisting of 6 bytes. The Source address is also 6 bytes long. As per IEEE 802.1Q, priority tagging is used to separate time critical and high priority bus traffic for protection-relevant applications. The 802.1Q Virtual Local Area Network (VLAN) is 4 bytes in length and consists of the TPID (Tag protocol identifier), TCI (Tag Control Information) and Ethertype. The TPID is the Ethertype assigned for 802.1Q Ethernet encoded frames and is given by 0x8100. The TCI consists of the CFI (Canonical Frame Indicator) and optional VID (VLAN Identifier). The TCI and Ethertype (0x88b8 for GOOSE) consists of 2 bytes each (IEC 61850-8-1 pg 115 . [4]). The application identifier (APPID) is 2 bytes in length and is used to select GOOSE messages from the frame and to distinguish the application association.

The discussion above indicates that all fixed components of the GOOSE message structure have been confirmed and accounted for. However, there were fields in the hexadecimal pane of the subsequent section in the message that warranted further investigation. As can be seen from Fig. 7, that when going from highlighted section Reserved 2 to goosePdu (GOOSE Data Protocol Unit), two bytes (with values 61 and 68) have been skipped. The result of this investigation pointed to another standard also referenced in IEC 61850-8-1, known as the Abstract Syntax Notation ONE, Basic Encoding Rules (ASN.1/BER) standard for Data networks and open system communications. [4] The next section considers aspects relating the ASN.1/BER standard to IEC 61850-8-1. The order in which elements occur in the GOOSE Protocol Data Unit (goosePdu) is always TAG, LENGTH followed by the DATA according to the ASN.1, as shown in Fig. 8. This will be explained in more detail later in this paper.



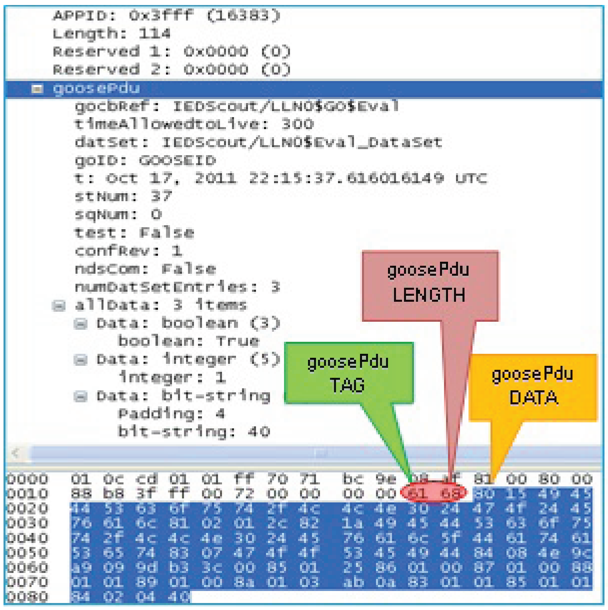
```
    APPID: 0x3fff (16383)
    Length: 114
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: IEDScout/LLN0$GO$Eval
      timeAllowedtoLive: 300
      datSet: IEDScout/LLN0$Eval_DataSet
      goID: GOOSEID
      t: Oct 17, 2011 22:15:37.616016149 UTC
      stNum: 37
      sqNum: 0
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 3
    ⊟ allData: 3 items
      ⊟ Data: boolean (3)
          boolean: True
      ⊟ Data: integer (5)
          integer: 1
      ⊟ Data: bit-string (4)
          Padding: 4
          bit-string: 40

0000   01 0c cd 01 01 ff 70 71   bc 9e 08 af 81 00 80 00
0010   88 b8 3f ff 00 72 00 00   00 00 61 68 80 15 49 45
0020   44 53 63 6f 75 74 2f 4c   4c 4e 30 24 47 4f 24 45
0030   76 61 6c 81 02 01 2c 82   1a 49 45 44 53 63 6f 75
0040   74 2f 4c 4c 4e 30 24 45   76 61 6c 5f 44 61 74 61
0050   53 65 74 83 07 47 4f 4f   53 45 49 44 84 08 4e 9c
0060   a9 09 9d b3 3c 00 85 01   25 86 01 00 87 01 00 88
0070   01 01 89 01 00 8a 01 03   ab 0a 83 01 01 85 01 01
0080   84 02 04 40
```

Figure 7: Indication of values skipped when transisitioning from "Reserved 2" to "goosePdu"

The variable portion of the message structure starts with the GOOSE Pdu Length (Fig. 8) up until the end of the message frame. The following 4 bytes are for the Reserved 1 and Reserved 2 fields. Following this is the Application Protocol Data Unit (APDU) in Fig. 9.

The APDU Length is 2 bytes long and indicates the size of the APDU with 8 octets added. The goosePdu in this case is 104 bytes in length and the GOOSE Control block reference (gocbRef) consists of 21 bytes. The timeAllowedtoLive is 2 bytes long and the data set of Logical node 0 is 26 bytes long. The GOOSE ID (goID) consists of 7 bytes, while the time (t) consists of 8 bytes. The status number (stNum), sequence number (sqNum), test bit, con-



```
APPID: 0x3fff (16383)
Length: 114
Reserved 1: 0x0000 (0)
Reserved 2: 0x0000 (0)
⊟ goosePdu
    gocbRef: IEDScout/LLN0$GO$Eval
    timeAllowedtoLive: 300
    datSet: IEDScout/LLN0$Eval_DataSet
    goID: GOOSEID
    t: Oct 17, 2011 22:15:37.616016149 UTC
    stNum: 37
    sqNum: 0
    test: False
    confRev: 1
    ndsCom: False
    numDatSetEntries: 3
  ⊟ allData: 3 items
    ⊟ Data: boolean (3)
        boolean: True
    ⊟ Data: integer (5)
        integer: 1
    ⊟ Data: bit-string
        Padding: 4
        bit-string: 40

0000   01 0c cd 01 01 ff 70 71   bc 9e 08 af 81 00 80 00
0010   88 b8 3f ff 00 72 00 00   00 00 61 68 80 15 49 45
0020   44 53 63 6f 75 74 2f 4c   4c 4e 30 24 47 4f 24 45
0030   76 61 6c 81 02 01 2c 82   1a 49 45 44 53 63 6f 75
0040   74 2f 4c 4c 4e 30 24 45   76 61 6c 5f 44 61 74 61
0050   53 65 74 83 07 47 4f 4f   53 45 49 44 84 08 4e 9c
0060   a9 09 9d b3 3c 00 85 01   25 86 01 00 87 01 00 88
0070   01 01 89 01 00 8a 01 03   ab 0a 83 01 01 85 01 01
0080   84 02 04 40
```

Figure 8: Illustrating the Length and Tag bytes and the contents of the GOOSE Protocol Data Unit

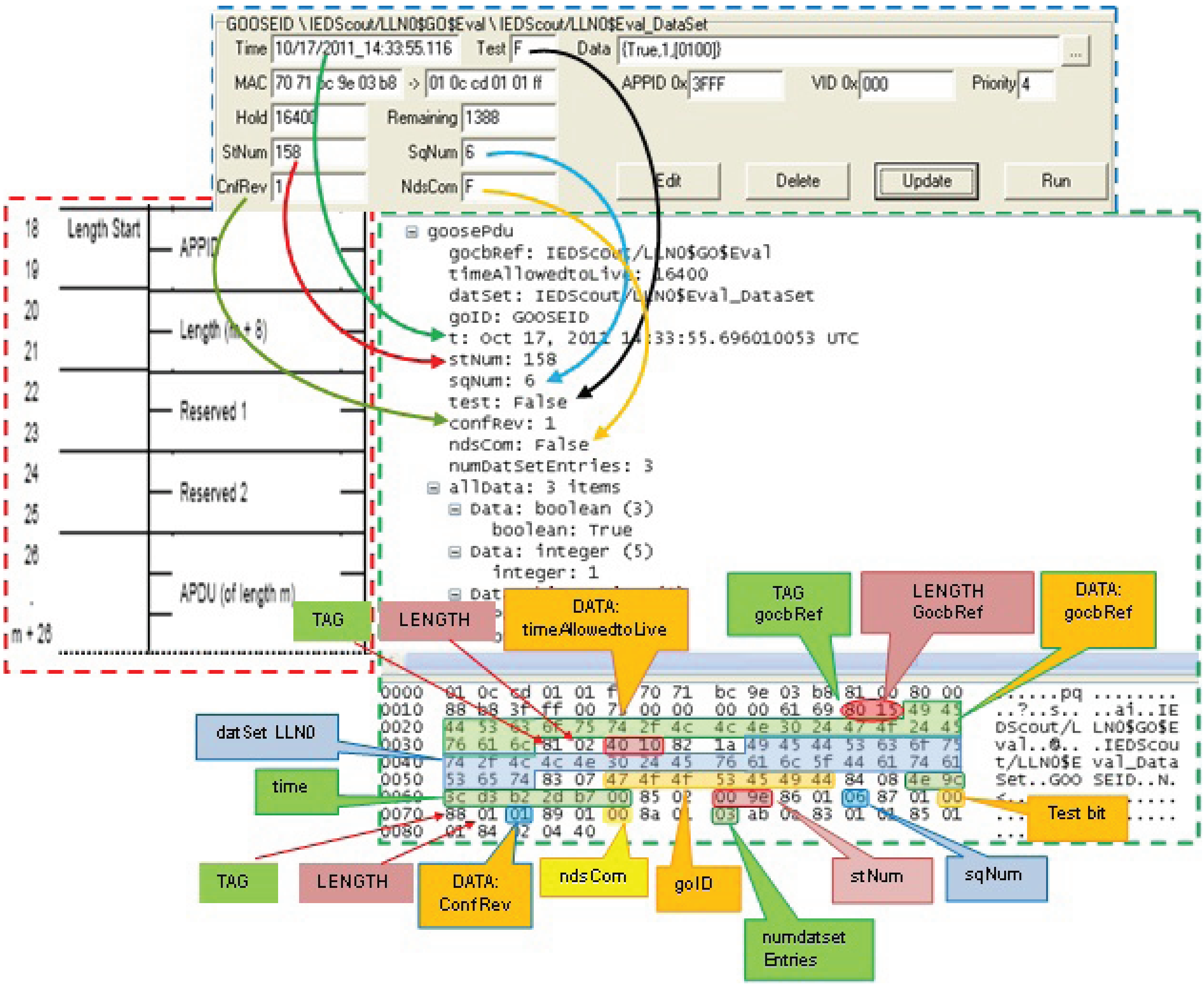
figuration revision (confRev), needs commissioning (ndsCom), and number of data set entries (numDatSetEntries) all are 1 byte in length. All these fields together with their functions are explained in detail in IEC 61850-8-1 page 114-116. [4] What is important to note is that before each of the preceding bytes, we first have the TAG, then the LENGTH followed by the actual DATA.

The last portion of the GOOSE message is the user-defined data content shown in Fig. 10. The user-defined data attributes in this particular case consists of three different items, namely a Boolean value, an integer and a data bit-string with padding. The Boolean and integer data items are 1 byte in length while the last data entry is 2 bytes long. The data section of the GOOSE message structure can be referenced in IEC 61850-7-2 page 116.] [8]

5 Application Protocol Data Unit (APDU) and ASN1 Basic Encoding Rules

The data within the GOOSE message are contained within the GOOSE Protocol Application Unit (PDU) and which is encoded in accordance with the Abstract Syntax Notation ONE (ASN. 1) standard for Data networks and open system communications (IEC 61850-8-1 pg 111). [4] The ASN.1 is an international standard used to define protocols of communication by means of encoding rules, IEC 61850-8-1, page111. The GOOSE protocol is defined using the ASN. 1/BER encoding rule (ASN. 1 Encoding Rule X.690-0207).

The encoding of GOOSE is not rigorously based on the original ASN. 1/BER but on an adaptation of ASN, 1/BER for Manufacturing Message Specification (MMS). It is important to note that this is not the original ASN.1 but a modified version as the original version does not cater for signed integers. [9], [10], [11]



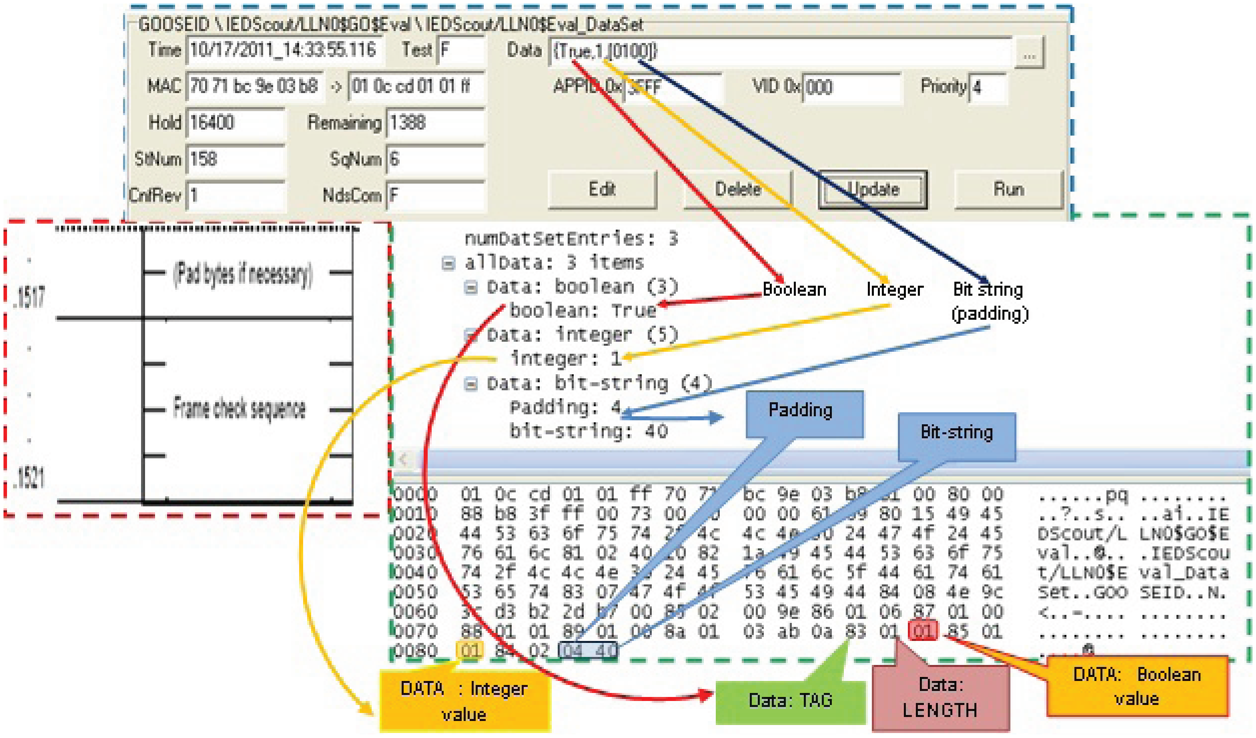
GOOSEID \ IEDScout/LLN0$GO$Eval \ IEDScout/LLN0$Eval_DataSet

Time	10/17/2011_14:33:55.116	Test	F	Data	{True,1,[0100]}
MAC	70 71 bc 9e 03 b8 -> 01 0c cd 01 01 ff	APPID 0x	3FFF	VID 0x	000
Priority	4				
Hold	16400	Remaining	1388		
StNum	158	SqNum	6		
CnfRev	1	NdsCom	F		

```
⊟ goosePdu
    gocbRef: IEDScout/LLN0$GO$Eval
    timeAllowedtoLive: 16400
    datSet: IEDScout/LLN0$Eval_DataSet
    goID: GOOSEID
    t: Oct 17, 2011 14:33:55.696010053 UTC
    stNum: 158
    sqNum: 6
    test: False
    confRev: 1
    ndsCom: False
    numDatSetEntries: 3
  ⊟ allData: 3 items
    ⊟ Data: boolean (3)
        boolean: True
    ⊟ Data: integer (5)
        integer: 1
    ⊟ Dat
```

```
0000  01 0c cd 01 01 ff 70 71  bc 9e 03 b8 81 00 80 00   ......pq ........
0010  88 b8 3f ff 00 7  00 00  00 00 61 69 80 15 49 45   ..?..s.. ..ai..IE
0020  44 53 63 6f 75 74 2f 4c  4c 4e 30 24 47 4f 24 45   DScout/L LN0$GO$E
0030  76 61 6c 81 02 40 10 82  1a 49 45 44 53 63 6f 75   val..@.. .IEDScou
0040  74 2f 4c 4c 4e 30 24 45  76 61 6c 5f 44 61 74 61   t/LLN0$E val_Data
0050  53 65 74 83 07 47 4f 4f  53 45 49 44 84 08 4e 9c   Set..GOO SEID..N.
0060  3c d3 b2 2d b7 00 85 02  00 9e 86 01 06 87 01 00   <....... ........
0070  88 01 01 89 01 00 8a 01  03 ab 0  83 01 0  85 01   ........ ........
0080  01 84 02 04 40                                     ...
```

Figure 9: The VARIABLE section of the GOOSE message including the TAG, LENGTH, DATA order indicated by ASN. 1



GOOSEID \IEDScout/LLN0GOEval \IEDScout/LLN0$Eval_DataSet

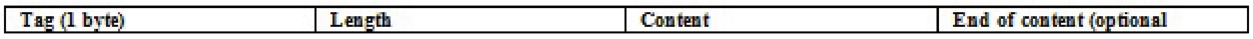
Time	10/17/2011_14:33:55.116	Test	F	Data	{True,1,[0100]}
MAC	70 71 bc 9e 03 b8 -> 01 0c cd 01 01 ff	APPID 0x	3FFF	VID 0x	000
Hold	16400	Remaining	1388	Priority	4
StNum	158	SqNum	6		
CnfRev	1	NdsCom	F		

```
numDatSetEntries: 3
⊟ allData: 3 items
  ⊟ Data: boolean (3)
      boolean: True
  ⊟ Data: integer (5)
      integer: 1
  ⊟ Data: bit-string (4)
      Padding: 4
      bit-string: 40
```

```
0000  01 0c cd 01 01 ff 70 71  bc 9e 03 b8 81 00 80 00   ......pq ........
0010  88 b8 3f ff 00 73 00 [illegible]  00 00 61 69 80 15 49 45   ..?..s.. ..ai..IE
0020  44 53 63 6f 75 74 2f 4c  4c 4e 30 24 47 4f 24 45   DScout/L LN0$GO$E
0030  76 61 6c 81 02 40 [illegible] 82  1a 49 45 44 53 63 6f 75   val..@.. .IEDScou
0040  74 2f 4c 4c 4e 30 24 45  76 61 6c 5f 44 61 74 61   t/LLN0$E val_Data
0050  53 65 74 83 07 47 4f 4f  53 45 49 44 84 08 4e 9c   Set..GOO SEID..N.
0060  3c d3 b2 2d b7 00 85 02  00 9e 86 01 06 87 01 00   <..-.... ........
0070  88 01 01 89 01 00 8a 01  03 ab 0a 83 01 01 85 01   ........ ........
0080  01 84 02 04 40                                     .....@
```

Figure 10: The user-defined dataset of the GOOSE message strucure

The purpose of ASN.1 is to provide encoding and decoding specifications for protocol syntax that is to be sent over a network. The intent of this standard (ISO/IEC 8824 and 8825) is to have a neutral representation of fields as they are exchanged over a communication media. ASN. 1 encoded values always have the format of TAG, LENGTH, followed by VALUE. [10] Fig. 11. The order in which each element appears within the GOOSE APDU (Application Protocol Data Unit) is also considered in ASN. 1. [11]

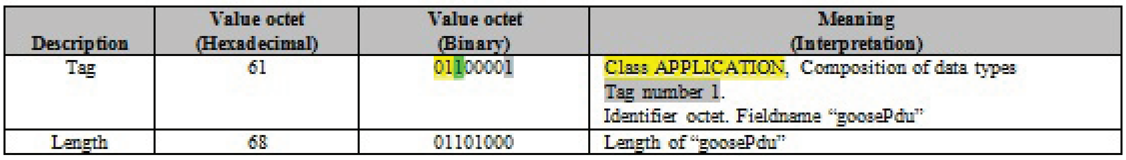


Tag (1 byte)	Length	Content	End of content (optional

Figure 11: An example of ASN. 1 encoding showing the order of elements within the GOOSE ADPU

The ASN.1 TAG describes the kind of information represented by the frame. The LENGTH is how many bytes (OCTETS) follow the DATA part of the frame. The LENGTH has a simple and extended form. If the length is less than 128 bytes, a single octet is used with its MSB set to 0. If the length is greater than 128, the MSB is set to 1, and the remaining 7 bits expressing the number of bytes that will contain the following parameter length. [11] The VALUE is the actual data content being specified by the frame; the value in the field must be interpreted according to the type of field. For example, if the field type is string, the octets must be decoded as characters, but if specified as an integer, the numeric value will be calculated based on the binary content.

The goosePdu always starts with a Tag (Identifier octet) containing the Class/Type of tag; the Primitive or Constructed Flag and the Tag number.



Description	Value octet (Hexadecimal)	Value octet (Binary)	Meaning (Interpretation)
Tag	61	01100001	Class APPLICATION, Composition of data types Tag number 1. Identifier octet. Fieldname "goosePdu"
Length	68	01101000	Length of "goosePdu"

Figure 12: Start of goosePdu tag header

With respect to Fig. 7, 8 and the table in Figure 12, it can clearly be seen that the bytes which were skipped refer to a class application type in this case goosePdu, and the associated length of the goosePdu. Similar common header tags appear throughout the PDU section of the frame. Some of these are presented in the tables displayed in Figures 13 and 14 including the tag headers, lengths and values for each byte within the goosePdu for the values in the frame represented in Figures 6 through 10.

The results of the simulation, practical implementation and the analysis thereof, confirm that the structure and data content for the GOOSE message is the same for both the experimentation and the simulation. The conclusion reached is that the structure of the GOOSE frame with respect to the IEC 61850-8-1 standard specified in page 114 and the content (user-defined data) of the GOOSE message with respect to IEC 61850-7-2 page 116 is confirmed.

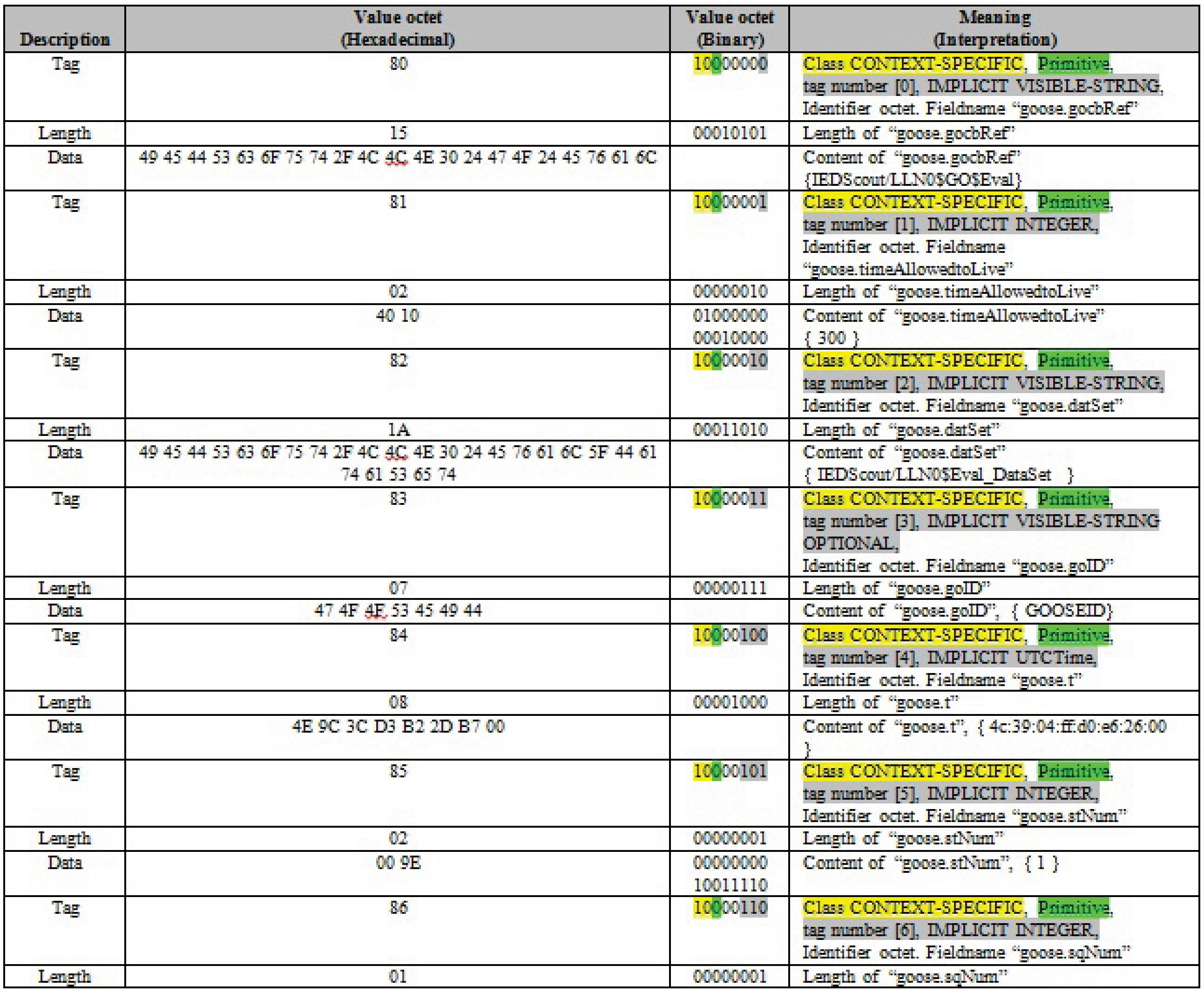
6 Conclusions and Future Works

The IEC 61850 standard consists of many different parts and some are extremely difficult to understand and interpret even among domain experts [12]. Students also have to grapple with the challenges presented by the IEC 61850 standard and the approach adopted in this work was to develop a detailed understanding of the message structure communicated between the various devices within the substation network for instances requiring fault diagnosis to determine the cause of maloperation or interoperability issues [13].

The paper has discussed a detailed investigation and confirmation of the GOOSE message structure and data content through a process of simulation and experimentation. This was confirmed by referencing to the relevant parts of the IEC 61850 standard and also the modified ASN.1/BER standard. The process of confirmation of message structure and content has enabled a foundation to be established from which further educational activities such as embedded systems development and diagnostic activities might be pursued.

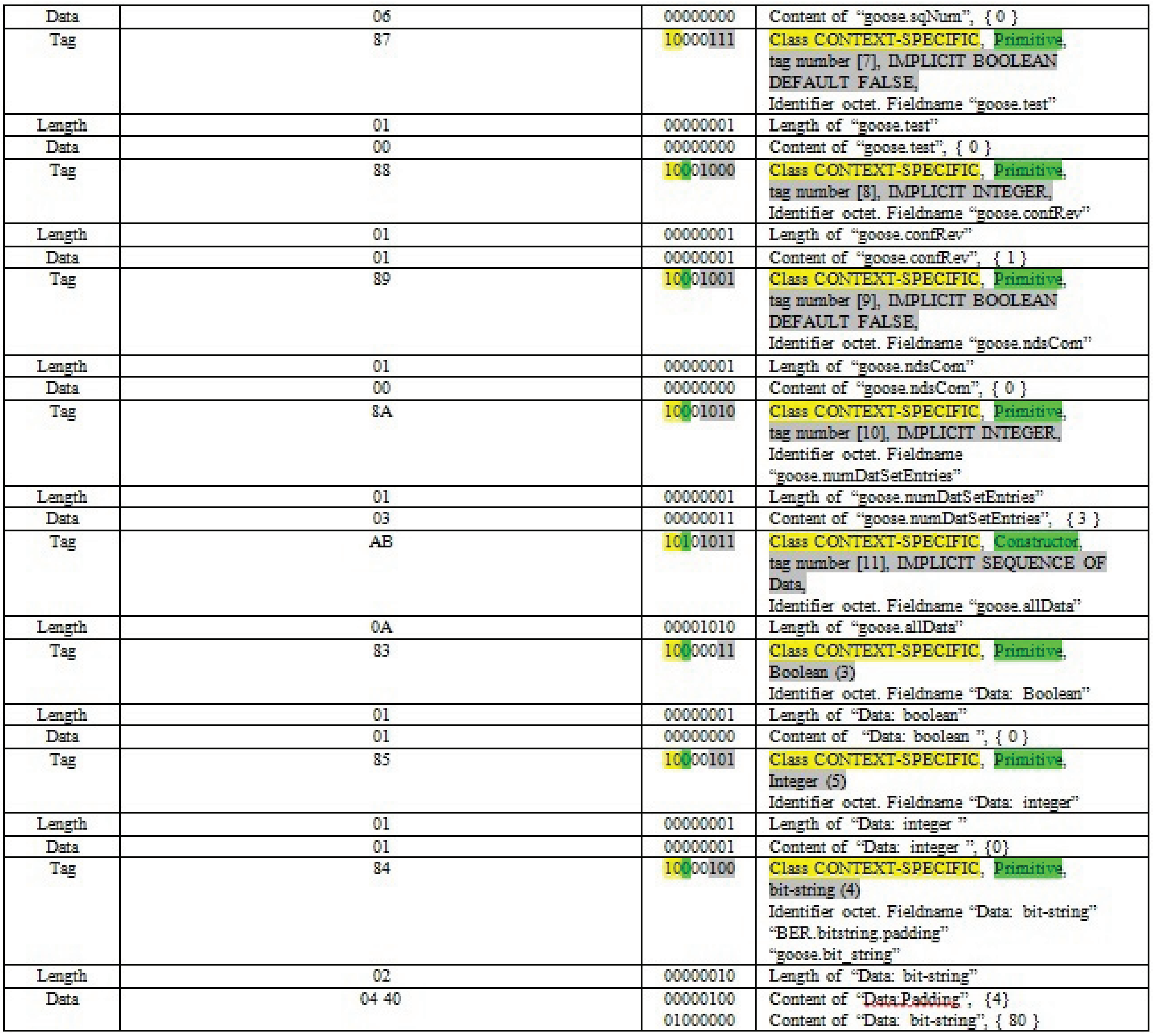
Acknowledgements

The research work described in the paper is partly funded by the National Research Foundation grant ICD2006061900021 and by the Eskom TESP program under the grants Investigation into Standard Based Substation Automation and Energy Management Systems, Methods for Power System State Estimation and GOOSE and Sampled Value Message structure investigation for IEC 61850 Standard-based implementation.



Description	Value octet (Hexadecimal)	Value octet (Binary)	Meaning (Interpretation)
Tag	80	10000000	Class CONTEXT-SPECIFIC, Primitive, tag number [0], IMPLICIT VISIBLE-STRING, Identifier octet. Fieldname “goose.gocbRef”
Length	15	00010101	Length of “goose.gocbRef”
Data	49 45 44 53 63 6F 75 74 2F 4C 4C 4E 30 24 47 4F 24 45 76 61 6C		Content of “goose.gocbRef” {IEDScout/LLN0$GO$Eval}
Tag	81	10000001	Class CONTEXT-SPECIFIC, Primitive, tag number [1], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.timeAllowedtoLive”
Length	02	00000010	Length of “goose.timeAllowedtoLive”
Data	40 10	01000000 00010000	Content of “goose.timeAllowedtoLive” { 300 }
Tag	82	10000010	Class CONTEXT-SPECIFIC, Primitive, tag number [2], IMPLICIT VISIBLE-STRING, Identifier octet. Fieldname “goose.datSet”
Length	1A	00011010	Length of “goose.datSet”
Data	49 45 44 53 63 6F 75 74 2F 4C 4C 4E 30 24 45 76 61 6C 5F 44 61 74 61 53 65 74		Content of “goose.datSet” { IEDScout/LLN0$Eval_DataSet }
Tag	83	10000011	Class CONTEXT-SPECIFIC, Primitive, tag number [3], IMPLICIT VISIBLE-STRING OPTIONAL, Identifier octet. Fieldname “goose.goID”
Length	07	00000111	Length of “goose.goID”
Data	47 4F 4F 53 45 49 44		Content of “goose.goID”, { GOOSEID}
Tag	84	10000100	Class CONTEXT-SPECIFIC, Primitive, tag number [4], IMPLICIT UTCTime, Identifier octet. Fieldname “goose.t”
Length	08	00001000	Length of “goose.t”
Data	4E 9C 3C D3 B2 2D B7 00		Content of “goose.t”, { 4c:39:04:ff:d0:e6:26:00 }
Tag	85	10000101	Class CONTEXT-SPECIFIC, Primitive, tag number [5], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.stNum”
Length	02	00000001	Length of “goose.stNum”
Data	00 9E	00000000 10011110	Content of “goose.stNum”, { 1 }
Tag	86	10000110	Class CONTEXT-SPECIFIC, Primitive, tag number [6], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.sqNum”
Length	01	00000001	Length of “goose.sqNum”

Figure 13: Examination of each tag within the goosePdu for the given case study



Data	06	00000000	Content of “goose.sqNum”, { 0 }
Tag	87	10000111	Class CONTEXT-SPECIFIC, Primitive, tag number [7], IMPLICIT BOOLEAN DEFAULT FALSE, Identifier octet. Fieldname “goose.test”
Length	01	00000001	Length of “goose.test”
Data	00	00000000	Content of “goose.test”, { 0 }
Tag	88	10001000	Class CONTEXT-SPECIFIC, Primitive, tag number [8], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.confRev”
Length	01	00000001	Length of “goose.confRev”
Data	01	00000001	Content of “goose.confRev”, { 1 }
Tag	89	10001001	Class CONTEXT-SPECIFIC, Primitive, tag number [9], IMPLICIT BOOLEAN DEFAULT FALSE, Identifier octet. Fieldname “goose.ndsCom”
Length	01	00000001	Length of “goose.ndsCom”
Data	00	00000000	Content of “goose.ndsCom”, { 0 }
Tag	8A	10001010	Class CONTEXT-SPECIFIC, Primitive, tag number [10], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.numDatSetEntries”
Length	01	00000001	Length of “goose.numDatSetEntries”
Data	03	00000011	Content of “goose.numDatSetEntries”, { 3 }
Tag	AB	10101011	Class CONTEXT-SPECIFIC, Constructor, tag number [11], IMPLICIT SEQUENCE OF Data, Identifier octet. Fieldname “goose.allData”
Length	0A	00001010	Length of “goose.allData”
Tag	83	10000011	Class CONTEXT-SPECIFIC, Primitive, Boolean (3) Identifier octet. Fieldname “Data: Boolean”
Length	01	00000001	Length of “Data: boolean”
Data	01	00000000	Content of “Data: boolean ”, { 0 }
Tag	85	10000101	Class CONTEXT-SPECIFIC, Primitive, Integer (5) Identifier octet. Fieldname “Data: integer”
Length	01	00000001	Length of “Data: integer ”
Data	01	00000001	Content of “Data: integer ”, {0}
Tag	84	10000100	Class CONTEXT-SPECIFIC, Primitive, bit-string (4) Identifier octet. Fieldname “Data: bit-string” “BER.bitstring.padding” “goose.bit_string”
Length	02	00000010	Length of “Data: bit-string”

Data	04 40	00000100
		01000000
			Content of “Data:Padding”, {4}
			Content of “Data: bit-string”, { 80 }

Figure 14: Examination of each tag within the goosePdu for the given case study (continued)

Bibliography

[1] Dolezilek D.; IEC 61850: What you need to know about functionality and practical implementation, Power Systems Conference: *Advanced Metering, Protection, Control, Communication, and Distributed Resources*, PS 06 117, 2006.

[2] De Oliveira J.C., Varella W.A., Marques A.E., Forster G.; Real Time Application using multicast Ethernet in Power Substation Automation according to IEC61850, *PAC World Journal*, 2007.

[3] IEC 61850-7-1.; Communication networks and systems in substations - Basic communication structure for substation and feeder equipment Principles and models, *International Electrotechnical Commission (IEC)*, 2003.

[4] IEC 61850-8-1.; Communication networks and systems in substations - Specific Communication Service Mapping (SCSM) Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3, *International Electrotechnical Commission (IEC)*, 2003.

[5] Apostolov A.; Fundamentals of IEC 61850, Seminar presented at the *Cape Peninsula University of Technology*, Cape Town, South Africa, 2009.

[6] IEC 61850-1.; Communication networks and systems in substations Introduction and overview, *International Electrotechnical Commission (IEC)*, 2003.

[7] Makhetha M., Kriger C.; Data acquisition and data distribution in an IEC 61850 standards-based substation environment. Unpublished thesis, Cape Peninsula University of Technology, Electrical Engineering Department, 2011.

[8] IEC 61850-7-2.; Communication networks and systems in substations - Basic communication structure for substation and feeder equipment Abstract Communication Service Interface (ACSI), *International Electrotechnical Commission (IEC)*, 2003.

[9] Retonda J., Behardien S.; Simulation of an IEC 61850 Based GOOSE Message, and Analysis of its Structure, *OMICRON Users Conference*, 2010.

[10] Falk, H., Burns, M.; MMS and ASN.1 Encodings Simple Examples and Explanations on How to Crack an MMS PDU, *Systems Integration Specialists Company, Inc.(SISCO), USA*, 1996.

[11] Cesi Ricerca, Valutazione delle tempistiche associate ai messaggi di tipo GOOSE nellambito del protocollo IEC-61850, *TTD Tecnologie T and D*, 2006.

[12] Liang, Y., Campbell, R.H.; Understanding and Simulating the IEC 61850 Standard. *Department of Computer Science University of Illonois. Urbana, USA*, 2008.

[13] Tzoneva,R. Apostolov, A. Behardien, S. Kriger, C. Boesak, D. Gumede, C. ; IEC 61850 Standard Based Substation Automation Challenges To Universities. *PAC World Congress, Dublin Ireland*, 2010.