ALBERTO FERREIRA DA SILVA

METODOLOGIA DE AUDITORIA DE MENSAGENS GOOSE EM REDES IEC61850 EM SUBESTAÇÕES DE ENERGIA ELÉTRICA

São Paulo
2021

ALBERTO FERREIRA DA SILVA

METODOLOGIA DE AUDITORIA DE MENSAGENS GOOSE EM REDES IEC61850 EM SUBESTAÇÕES DE ENERGIA ELÉTRICA

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências.

Área de Concentração: Sistemas de Potência
Orientador: Profª. Dra. Milana Lima dos Santos

São Paulo
2021

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

Silva, Alberto
Metodologia de Auditoria de Mensagens GOOSE em Redes IEC61850 em Subestações de Energia Elétrica / A. Silva -- versão corr. -- São Paulo, 2021.
122 p.
Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.
1.GOOSE 2.IEC61850 3.Substation 4.Auditor 5.SCD Files I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II.t.

Nome: Silva, Alberto Ferreira da

Título: Metodologia de auditoria de mensagens GOOSE em redes IEC-61850 em subestação de energia elétrica.

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Ciências

Aprovado em:

Banca Examinadora

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

Prof. Dr. _____

Instituição: _____

Julgamento: _____

AGRADECIMENTOS

Agradeço a Deus, que me deu a vida.

Agradeço à minha mãe Carmen Lúcia Ferreira da Silva e à Carmesita Antonia da Silva, minha tia, por me proporcionarem toda a educação e honestidade como pessoa.

Agradeço à Professora Doutora Milana Lima dos Santos, por me orientar com excelência.

Agradeço ao M.e Clovis Paulino, que me despertou a vontade contínua do aprendizado.

RESUMO

Silva, A. F. da **Metodologia de auditoria de mensagens GOOSE em redes IEC 61850 em subestações de energia elétrica**. 2021. 122 f. Dissertação (Mestrado em Engenharia Elétrica) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2021.

Com a digitalização das subestações de energia e a introdução de novos dispositivos de proteção e comando compatíveis com a norma IEC 61850 *The International Electrotechincal Commision* (IEC) 61850, o cabeamento elétrico de informações foi gradativamente substituído por trocas de mensagens GOOSE (Generic Object Oriented Substation Events). Contudo, não é prevista uma forma de verificar se a transmissão e a recepção das mensagens GOOSE pelos IED estão de acordo com as definidas no projeto inicial, conforme os *datasets* (conjuntos de dados) dos arquivos de configuração escritos em linguagem SCL (*Substation Configuration Language*). Dessa forma, no momento de uma anomalia, é preciso utilizar uma grande quantidade de horas das equipes técnicas para identificar a solução do problema. Este trabalho de pesquisa propõe uma metodologia e apresenta o desenvolvimento de uma ferramenta computacional que adquire as mensagens do protocolo GOOSE, definido na norma IEC 61850, e verifica a sua conformidade com a configuração documentada pelos arquivos SCD e com as definidas pelo usuário através de arquivo de configuração privado, auxiliando as equipes de operação e manutenção através da emissão de relatórios detalhados e claros e alertas em caso de falhas detectadas.

Palavras-chave: IEC 61850, Subestações. GOOSE. Arquivos SCD. Arquivos ICD. Arquivos CID. Auditor.

ABSTRACT

Silva, A. F. da **Methodology for auditing GOOSE messages on IEC 61850 networks in electronic power substation**. 2021. 122 f. Dissertação (Mestrado em Engenharia Elétrica) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2021.

With the digitalization of power substations, and the introduction of new protection and control devices compatible with the IEC 61850 standard *The International Electrotechincal Commision* (IEC) 61850, the electrical information cabling was gradually replaced by exchanges of GOOSE messages (Generic Object Oriented Substation Events). However, there is no way to verify that the transmission and reception of GOOSE messages by the IEDs are in accordance with those defined in the initial project, according to the datasets (data sets) of the configuration files written in SCL (Substation Configuration Language). Thus, in the event of an anomaly, it is necessary to use a large amount of hours from the technical teams to identify the solution to the problem. This research work proposes a methodology and presents the development of a computational tool that acquires the messages of the GOOSE protocol, defined in the IEC 61850 standard, and verifies their compliance with the configuration documented by the SCD files and those defined by the user through a file private configuration, assisting the operation and maintenance teams by issuing detailed and clear reports and alerts in case of detected failures.

Keywords: IEC 61850, Substations. GOOSE. SCD files. ICD files. CID files. Auditor.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AAG	Aplicativo Auditor GOOSE
ASN.1	Abstract Syntax Notation One
ATG	Arquivo de Tradução GOOSE
CID	Configuration IED Description
COS	Centro de Operação do Sistema
DNP3	Distributed Network Protocol
EPRI	Electric Power Research Institute
GOOSE	Generic Object Oriented Substation Events
GSSE	Generic Substation Status Event
ICD	IED Capability Description
IDE	Integrated Development Envrionment
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IEEE	Electrical and Electronics Engineers
IID	Instantiated IED Description
ISO	International Standards Organization
IP	Internet Protocol
LAN	Local Area Network
LN	Logical Node
MAC	Media Access Control
MMS	Manufacturing Message Specification
NTP	Network Time Protocol
MMXU	Logical Node for Measurement
OSI	Open Systems Interconnection
SCADA	Supervisory Control and Data Acquisition
QoS	Quality of Service
SCD	Substation Configuration Description
SCL	Substation Configuration Language
SEL	Schweitzer Engineering Laboratories

SE	Subestação de Energia Elétrica
SEP	Sistema Elétrico de Potência
SSC	Sistema de Supervisão e Controle
SSD	System Specification Description
SV	Sampled Values
TAC	Teste de Aceitação de Campo
TAF	Teste de Aceitação de Fábrica
TC	Transformador de Corrente
TCP	Transmission Control Protocol
TP	Transformador de Potência
UCA	Utility Communications Architecture
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
VLAN	Virtual LAN
XCBR	Logical Node for Circuit Break
XML	eXtensible Modeling Language

SUMÁRIO

Introdução

Nos últimos anos ocorreram mudanças nas tecnologias empregadas no setor elétrico, proporcionando redução de tamanho de equipamentos e trazendo novas funcionalidades. Como resultado dessas mudanças, foram mantidos ou até mesmo melhorados os níveis de confiabilidade dos equipamentos com aumento significativo da disponibilidade dos mesmos (RUSH, 2010).

Para adequar e padronizar os novos conceitos aplicados nos projetos de automação, operação e a manutenção das subestações de energia elétrica, a *International Electrotechnical Commission* (IEC), criou a norma 61850 – *Communication networks and systems for power utility automation*. Sua ampla utilização pelos principais fabricantes de *Intelligent Electronic Devices* (IED), fez surgir novas possibilidades de atuação integrada dos componentes de proteção, comando e supervisão através da troca de mensagens *Generic Object Oriented Substation Events* (GOOSE).

A norma IEC 61850 tem sido cada vez mais utilizada pelas empresas do setor de energia elétrica na concepção de novas subestações e no processo de atualização de SEs mais antigas. A possibilidade da troca de mensagens GOOSE no barramento horizontal para funcionalidades de proteção e controle reduz a necessidade da utilização de cabos elétricos para interligação dos IEDs com o objetivo de compartilhar informações (KIMURA, ABBOUD, *et al.*, 2008).

Com a utilização do GOOSE, a dependência de cabos elétricos para sensoriamento de sinais analógicos e digitais entre os IEDs instalados nas subestações é reduzido, pois as informações podem ser facilmente transportadas por mensagens de comunicação. Porém, a utilização do GOOSE em funções críticas como proteção e comando exige cuidado especial na verificação do seu correto funcionamento.

Para comunicação por GOOSE, um arquivo *Substation Configuration Description* (SCD) é configurado. Devido a particularidades da configuração, muitas delas com o objetivo de tornar a transmissão de mensagens ágil e não prejudicar o desempenho do IED, não é prevista uma forma de verificar se o dispositivo está enviando corretamente as informações previstas no projeto inicial, conforme os *datasets* (conjunto de dados) dos arquivos de configurações.

Contudo, a anterior interligação elétrica possibilitava analisar os pontos fisicamente utilizando equipamentos simples de teste de continuidade ou medição caso houvesse necessidade. Para o modelo IEC 61850, o diagnóstico de possíveis falhas envolvendo as mensagens GOOSE requer mão de obra especializada e a utilização de diversos recursos de *softwares* para conclusão.

Para que os equipamentos inteligentes possam atuar corretamente em subestações configuradas de acordo com a norma IEC 61850, se faz necessário o uso de mão de obra especializada, documentos padronizados e ferramentas que possibilitem aos usuários analisarem amigavelmente as mensagens GOOSE trafegadas. Como não é previsto na norma IEC 61850 uma forma de garantir o correto funcionamento dos IEDs para publicação GOOSE, algumas ferramentas disponibilizadas no mercado são de uso específico para alguns fabricantes, não sendo possível a utilização em ambientes heterogêneos. A metodologia aqui proposta vem ao encontro de atender uma parte desse novo cenário, pois visa propor uma ferramenta única que possibilita documentar, auditar e identificar possíveis falhas nas mensagens GOOSE publicadas no barramento horizontal independente do fabricante.

1.1. Estrutura do trabalho

A estrutura deste trabalho é composta por sete capítulos incluindo este primeiro introdutório. Também neste primeiro capítulo, algumas pesquisas utilizadas como referências são comentadas, indicando o atual momento do estado da arte para o tema.

O segundo capítulo aborda os detalhes da inovação tecnológica dos sistemas de proteção e automação das subestações ao longo dos anos. Neste capítulo, também são comentados os componentes que formam um sistema elétrico de potência.

No terceiro capítulo são apresentados os conceitos da norma IEC 61850, sendo o foco específico nos capítulos 6, 7 e 8 que, contém as principais informações da metodologia elaborada neste trabalho.

No quarto capítulo é apresentada a proposta para metodologia de auditoria das mensagens GOOSE. Apresentam-se também o arquivo auxiliar elaborado como referência da configuração definida no projeto inicial e o aplicativo auditor desenvolvido para o automatismo do processo.

No quinto capítulo a arquitetura física e lógica utilizada durante a pesquisa é apresentada. Também neste capítulo são detalhadas as configurações parametrizadas nos dispositivos.

O sexto capítulo apresenta os ensaios realizados e os resultados obtidos com o aplicativo desenvolvido, produto deste trabalho, que são a auditoria das mensagens GOOSE e a apresentação dos atributos de acordo com a tradução definida pelo usuário.

Uma conclusão é apresentada no sétimo e último capítulo, mostrando que é possível auditar as mensagens GOOSE em rede IEC 61850 e diagnosticar divergência com base no projeto inicial.

1.2. Estado da Arte

A norma IEC 61850 é um tema bastante abordado em pesquisas científicas da área de engenharia elétrica e automação. Ao pesquisar sobre o tema, percebe-se que os benefícios da utilização da norma IEC 68150 podem ser observados em diversos campos da sua aplicação, desde a infraestrutura de implantação quanto na velocidade de atuação dos disparos e monitoramento em tempo real do sistema. Acrescenta-se também o surgimento de um novo perfil profissional para atuar nos ambientes onde a norma é inserida.

Na pesquisa realizada por Vicente (VICENTE e SENGER, 2011), é proposta a troca de informações entre agentes de subestações compartilhadas de transmissão e distribuição de energia elétrica utilizando de recursos da norma IEC 61850. O trabalho visa a substituição de sinais elétricos por sinais lógicos para comunicação dos equipamentos compartilhados.

No trabalho de Netto (NETTO, 2012), foi pesquisado um parâmetro de dimensionamento para monitoramento do desempenho de mensagens GOOSE do padrão IEC 61850. O autor destaca as mudanças que ocorreram nas subestações com a implementação da norma IEC61850. Através desse processo de digitalização, os sinais que antes eram enviados através de contatos elétricos físicos, passaram a ser enviados por sinais lógicos; consequentemente, a confiabilidade do sistema de proteção passou a depender da rede de comunicação de dados. Diante disso o autor propõe o uso de um serviço de priorização de pacotes *Quality of Service* (QoS) e a

ocupação da banda dos IEDs para determinar um parâmetro que garanta o desempenho das mensagens GOOSE em uma rede IEC 61850.

No trabalho de Araujo (ARAUJO, CAMPELLO e GUALTIERI, 2011), foi pesquisado, uma aplicação da norma IEC 61850-8-1 nas redes de proteção do sistema elétrico. A autora aborda uma aplicação de auditoria para ser responsável por monitorar a rede de proteção e garantir a consistência da rede. Com a aplicação, uma melhor compreensão do que ocorre na rede, para permitir a identificação de problemas, como, por exemplo, relé não atualizado, é discutida. Além disso, a autora destaca que a auditoria de mensagens GOOSE é de fundamental importância para a análise dos telegramas e compreensão dos estados de cada relé e da subestação. Segundo a autora, ela facilita a combinação de informações para fornecer uma análise muito mais completa da rede.

No trabalho de Souza, (SOUZA e MARCOS, 2012), foi apresentado uma proposta de um sistema de monitoramento e validações de comunicação nas redes de subestações de energia. O autor defende a ideia de que, em redes que seguem os padrões da norma IEC 61850, seja possível prever mecanismos que realizem o controle dos telegramas GOOSE, garantindo dessa forma a integridade e periodicidade dentro dos parâmetros definidos em norma. O objetivo do trabalho foi desenvolver um *software* que pudesse ser executado em computadores, utilizando qualquer sistema operacional, para realizar auditoria das mensagens GOOSE e controle da largura de banda de rede.

Entre os trabalhos pesquisados, os trabalhos de Araujo (ARAUJO, CAMPELLO e GUALTIERI, 2011) e Souza (SOUZA e MARCOS, 2012), são os mais semelhantes à metodologia apresentada nessa pesquisa. A diferença é que a metodologia aqui apresentada visa a criação de um arquivo auxiliar utilizado para documentar as mensagens GOOSE publicadas e assinadas pelos IEDs. Também é proposto, nesse arquivo a inserção da tradução dos atributos contidos nas mensagens GOOSE para facilitar interpretação dos dados pelo usuário. Outra diferença entre os trabalhos pesquisados está na linguagem de programação utilizada para o desenvolvimento do *software* auditor, para os trabalhos anteriores foi utilizado a linguagem de programação JAVA e para o trabalho aqui proposto foi utilizado a linguagem *Microsoft dotNET C#.*

SISTEMA ELÉTRICO DE POTÊNCIA

1.3. Subestação de Energia

As SEs (Subestações de Energia Elétrica) são instalações que abrigam e conectam diversos equipamentos para garantir a transmissão, distribuição, proteção e controle de energia elétrica. É função de uma SE o direcionamento e o controle do fluxo energético, a elevação ou diminuição dos níveis de tensão, e a entrega de energia para os consumidores residenciais e industriais.

As SEs são pontos críticos do SEP (Sistema Elétrico de Potência). Ações e comandos executados devem ser coordenados por meio de programas, funções lógicas e filosofias de operação, em sintonia com informações disponibilizadas por sistemas de medição e proteção. Para isto, são utilizados dispositivos de manobras, como disjuntores e chaves, para medição, transformadores de corrente (TC) e de potencial (TP) são utilizados. (DUARTE, 2012)

Os sistemas de potência, devido aos seus requisitos de alta confiabilidade, alta disponibilidade e rapidez nas respostas a ocorrências, incorporam de forma bastante cautelosa o progresso tecnológico verificado, por exemplo, nas áreas de telecomunicação e computação (VICENTE e SENGER, 2011). A norma IEC 61850, desde sua elaboração inicial em 2002, tem sido referência no processo de modernização da concepção dos projetos e na forma de operação das subestações de energia elétrica, apesar de ainda enfrentar resistência por parte dos setores mais conservadores.

1.4. Dispositivos Eletrônicos Inteligentes de Proteção

Inicialmente, equipamentos com tecnologia eletromecânica eram utilizados para operação e proteção dos sistemas de geração, transmissão e distribuição de energia elétrica de todos os portes. Como esses equipamentos disponibilizavam poucas informações (como sinalizações em quadros de lâmpadas e bandeirolas), e não existia um meio de comunicação que permitisse o acesso remoto, a integração entre instalações era bastante reduzida.

Os relés estáticos começaram a substituir os relés eletromecânicos a partir da década de 60. Componentes eletrônicos foram inseridos na construção desses

equipamentos e todas as funcionalidades dos relés eletromecânicos puderam ser adicionadas aos dispositivos estáticos (JARDINI, 1997).

Estes dispositivos ainda estão em uso, porém foram descontinuados pelos fabricantes e estão sendo substituídos pelo setor elétrico após os primeiros relés digitais surgirem com a chegada dos microprocessadores. Os relés estáticos utilizam em seus algoritmos as premissas dinâmicas dos relés eletromecânicos integrados à evolução da eletrônica digital (VICENTE e SENGER, 2011).

O avanço tecnológico permitiu que os IEDs pudessem executar diversas funções, como por exemplo, aquisição de dados na rede, múltiplas funções de proteção, automação, controle, comunicação e até gerar arquivos de oscilografias (DUARTE, 2012).

Atualmente os IEDs, como também são conhecidos os relés digitais, estão consolidados no setor elétrico, sendo instalados tanto em projetos de novas subestações de energia elétrica como em projetos de substituição onde antigos relés eletromecânicos e estáticos são substituídos por dispositivos modernos (VICENTE e SENGER, 2011).

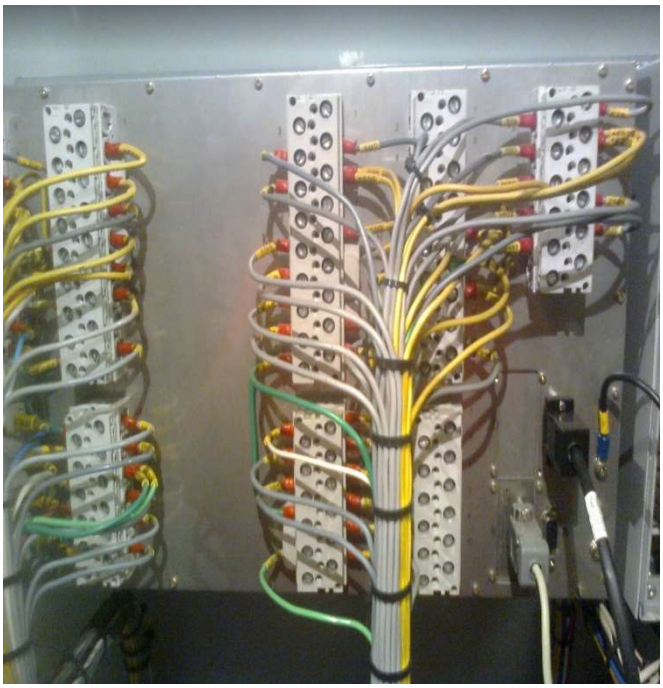
Normalmente a troca de informações entre os IEDs em projetos de subestações de energia elétrica anteriores à norma IEC 61850 é realizado através de cabos que interligam os contatos de entradas e saídas dos equipamentos instalados no ambiente e, dependendo do número de dispositivos e variáveis supervisionadas, a instalação dos cabos pode representar uma parcela considerável do custo do projeto (RUSH, 2010).

O rompimento de cabos elétricos de comunicação e o desgaste natural dos contatos dos equipamentos são exemplos comuns de problemas em uma subestação de energia pré-IEC-61850. Apesar disso, realizar análise dos problemas neste ambiente é menos complexo se comparado à subestações seguindo as premissas da IEC 61850, pois a utilização de equipamentos específicos que possibilitam identificar: continuidade, valores de corrente e tensão, podem ser suficientes para que um diagnóstico primário do problema seja realizado

A Figura 1 mostra a quantidade de cabos utilizados para troca de informações entre IEDs quando não realizada por mensagens GOOSE.

.

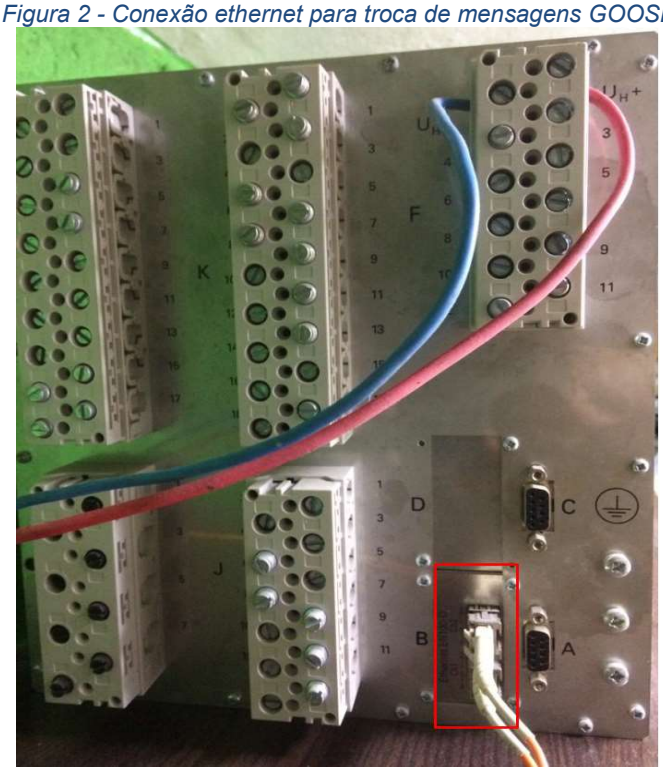
Figura 1 - Quantidade de cabos conectados no relé



Fonte: O autor

A Figura 2 contém a imagem de um IED integrado com IEC-61850. Observando a imagem é possível afirmar que para troca de informações entre os IEDs, apenas a conexão do dispositivo na rede é necessária. A porta de comunicação B está conectada na rede através de um cordão de fibra óptica.

Figura 2 - Conexão ethernet para troca de mensagens GOOSE



Fonte: O autor

1.5. Automação de Sistemas Elétricos

> "*O desenvolvimento da automação elétrica nas subestações está diretamente relacionado ao desenvolvimento dos relés digitais. Muitas aplicações deixaram de trabalhar isoladas e foram incorporadas aos demais sistemas das subestações, através de rede de comunicação ou através de funções agregadas aos relés de proteção*". (VICENTE e SENGER, 2011)

Os sistemas de automação de sistemas elétricos são conjuntos formados por *hardware* e *software* que possibilitam a visão completa dos equipamentos de energia e dos demais sistemas da operação, controlando sistemas elétricos cuja abrangência geográfica atende a pequenas instalações ou grande áreas. Uma das funções básicas dos sistemas é o monitoramento de todas as operações e a sinalização imediata caso ocorra qualquer evento divergente do esperado para a rede supervisionada. A utilização de computadores para o controle de redes elétricas vem sendo aplicado desde o início dos anos 1970. (RUSH, 2010)

1.6. Sistemas de Supervisão e Controle

SSCs (Sistema de Supervisão e Controle) são sistemas independentes e integrados, são responsáveis por controlar as SEs e/ou processos, através de interfaces com os dispositivos de campo. Os SSCs são responsáveis por promover a interface entre os sistemas de controle, os usuários e outros sistemas, bem como realizar operações como registro de dados e exibição de alarmes. No início, as primeiras aplicações dos sistemas SCADA (*Supervisory Control and Data Acquisition*) eram instaladas nos centros de comando das companhias, centralizando assim, toda a operação e controle das subestações. Atualmente, com a redução dos custos e maior capacidade de processamento de informação pelos equipamentos e *softwares,* viabilizou-se a implantação desses sistemas também na forma distribuída, proporcionando o controle e automação local. (RUSH, 2010)

A possibilidade de operar remotamente SEs a partir do COS (Centro de Operação do Sistema) tem proporcionado para as empresas a alternativa de não alocar recursos humanos, como por exemplo, operadores, dentro de subestações de energia, que podem funcionar de forma teleassistida, sendo supervisionada e comandadas remotamente. A SE supervisionada remotamente e sem operadores locais recebe o nome de Subestação Desassistida. Conforme a Resolução Normativa

n° 864, (MME e ANEEL, 2019), as SEs desassistidas também estão sujeitas a penalidades de PVI (Parcela Variável por Indisponibilidade) se houver impossibilidade de utilização de seus equipamentos para manobra ou operação enquanto ela permanecer energizada, assim, os meios de comunicações utilizados para operar essas SEs podem ser classificados como críticos.

1.7. Protocolos de Comunicação

O modelo de referência OSI (*Open Systems Interconnection*) se baseia em uma proposta desenvolvida pela ISO (*International Standards Organization*) como um primeiro passo em direção à padronização internacional dos protocolos usados nas várias camadas. Protocolo é um conjunto de regras que controla o formato e o significado dos pacotes ou mensagens trocadas pelas entidades pares contidos em uma camada.

O modelo OSI é constituído por sete camadas e a relação entre elas é apresentada na Figura 3.

A primeira camada, denominada camada física, é o núcleo das redes de computadores. Esta camada é responsável por definir os meios de transmissão de dados e sua largura de banda. Os meios podem ser definidos como: guiados; par trançados, coaxial e a fibra óptica, e não guiados; rádio, as micro-ondas, os raios infravermelhos, os raios laser através do ar e os satélites.

A segunda camada, camada de enlace de dados, tem a função de receber os sinais originados da camada física, organizá-los e entregá-los à camada de rede. A apresentação dos dados da camada de enlace pode utilizar vários níveis de confiabilidade, sendo estes níveis desde o serviço sem confirmação de resposta até com a confirmação. Esta camada possui funções específicas para detectar erros, que podem ser utilizados para anular ou corrigir o quadro danificado. Uma das funções da camada de enlace é o controle de fluxo, utilizado para que um emissor com maior capacidade de transmissão não oprima um emissor com capacidade inferior.

A terceira camada do modelo OSI é chamada de camada de rede. A camada recebe os pacotes ordenados da camada anterior, camada de enlace, e sua principal função é rotear os pacotes da origem até o destino. Existem diversos protocolos de roteamento utilizados para essa função, alguns desses protocolos tem a função de identificar de forma dinâmica o caminho mais curto para a rede destino.

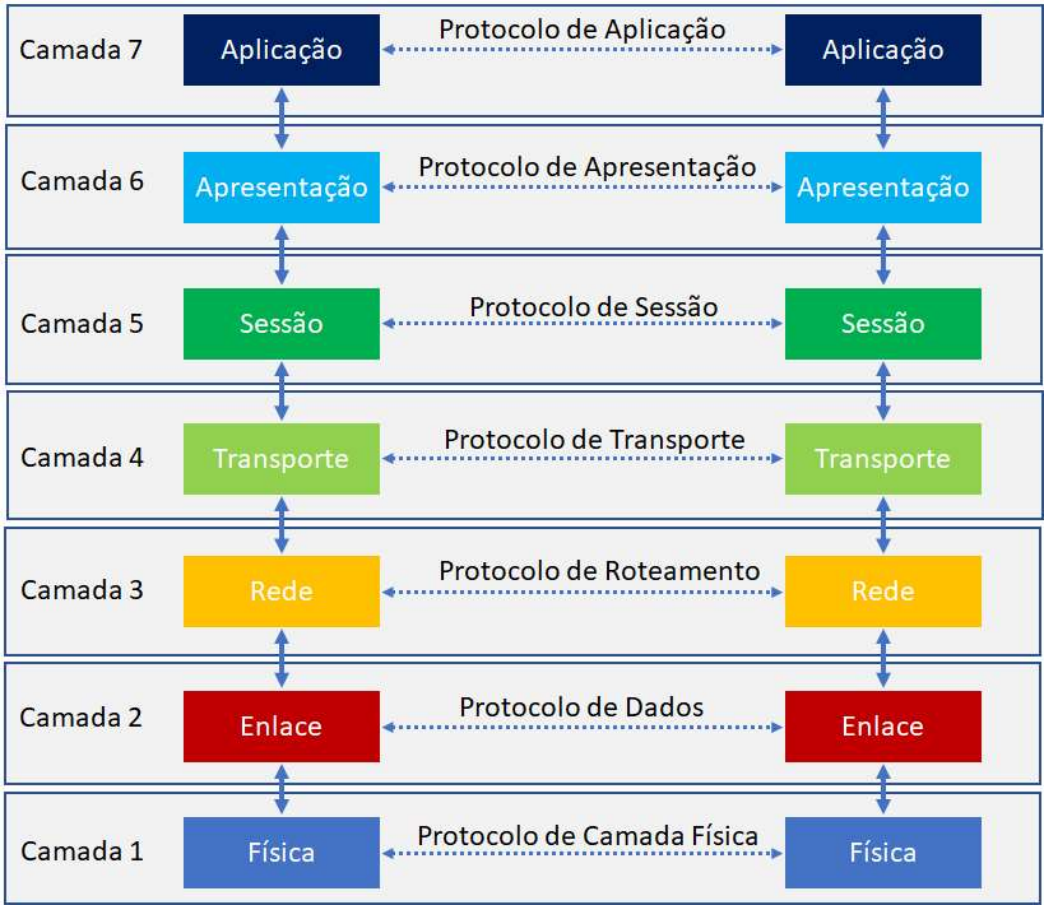
A quarta camada do modelo OSI é chamada de camada de transporte. A função mais importante desta camada é garantir a confiabilidade do tráfego de dados entre o emissor e o receptor. Os protocolos desta camada são capazes de realizar o gerenciamento da conexão em redes não confiáveis, utilizando recursos de *handshakes* para esta função. Os protocolos mais importantes para esta camada são: *User Datagram Protocol* (UDP); o destinatário não necessita confirmar o recebimento de mensagens para o emissor para a próxima mensagem seja enviada, e o *Transmission Control Protocol* (TCP); o emissor requer confirmação do destinatário para o recebimento da mensagem, só então a próxima mensagem é enviada.

A camada de sessão é a quinta camada do modelo OSI. A função desta camada é permitir que dispositivos conectados às redes estabeleçam sessões de comunicação entre si através das portas de serviços disponíveis. Esta camada direciona os dados para os serviços específicos da camada de apresentação.

A sexta camada do modelo OSI é a camada de apresentação. Diferente das camadas abaixo, que são responsáveis pela movimentação dos dados, esta camada tem a função de interpretar o conteúdo do bloco de dado e disponibilizá-lo à camada de aplicação para que possa ser transformado em informação.

A sétima e última camada do modelo OSI é a camada de aplicação; esta camada contém os protocolos necessários para formatar os dados recebidos e apresentá-los em forma de informação através das aplicações e programas específicos de cada protocolo.

Figura 3 - Modelo de referência OSI/ISO



Fonte: Adaptado de (TANENBAUM, 1997)

A Figura 3 ilustra a relação das 7 camadas do modelo OSI. A camada de rede n de uma máquina se comunica com a camada n da outra máquina. Portanto, protocolo, segundo (TANENBAUM, 1997), é um conjunto de regras que controla o formato e o significado dos quadros, pacotes ou mensagens trocadas pelos pares de entidades contidos em uma camada. Basicamente são definidos por um conjunto de: (VICENTE e SENGER, 2011)

a) formato de mensagens;
b) serviços;
c) procedimentos;
d) endereçamento;
e) convenção de nomes.

Com os novos recursos dos equipamentos instalados nas subestações, resultado do avanço tecnológico constante, novos protocolos de comunicação foram inseridos nos IEDs.

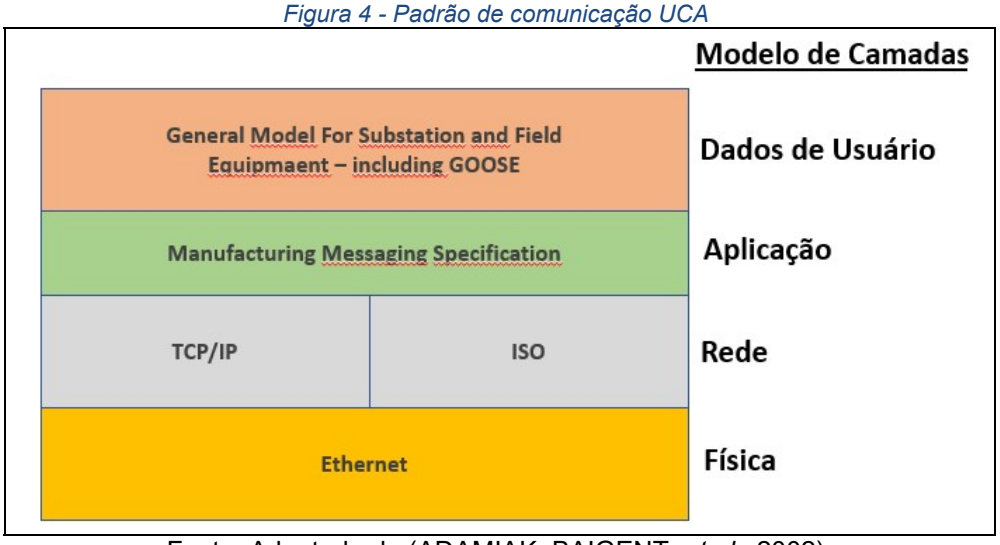
Antes do desenvolvimento dos protocolos abertos, como, por exemplo, o *Distributed Network Protocol* 3 (DNP3), os sistemas existentes nas subestações trabalhavam de forma isolada, pois cada fabricante desenvolvia protocolos específicos de comunicação para seus equipamentos e sistemas, dificultando a integração entre os sistemas de fabricantes diferentes.

Algumas iniciativas para padronizar a comunicação entre os equipamentos e os sistemas no setor elétrico começaram a ser idealizadas no início da década de 90. A Utility *Communications Architecture* (UCA) surgiu como um *framework* para o conjunto de requisitos de comunicação para serviços públicos cujo objetivo foi consolidar as comunicações entres os departamentos de planejamento, supervisão, medição, proteção e controle (ADAMIAK, BAIGENT, *et al.*, 2002). O processo começou com a criação da definição de requisitos de comunicação para as várias funções dentro de uma subestação. Os requisitos funcionais especificados foram seguidos por uma implementação e avaliação inicial que definiu o padrão de uma estrutura de comunicação para a subestação. A Figura 4 apresenta a estrutura de comunicação UCA. O objetivo na definição do padrão da subestação era implementar uma rede ponto a ponto de alta velocidade, conectável, usando o maior número de protocolos possível. Além disso, o objetivo da interoperabilidade de dados do usuário exigia a definição de nomes padronizados para objetos de dados comumente usados.

O padrão desenvolvido utiliza *Ethernet* para os processos físico e camadas do *Data Link.* Para camada de rede, embora o objetivo original foi permanecer dentro dos padrões da ISO, a IEC selecionou o TCP / IP como o protocolo de rede obrigatório para comunicações interna e externa da subestação e as camadas de rede OSI como opcional (ADAMIAK, BAIGENT, *et al.*, 2002). Para a camada de aplicação ou serviços, o *Manufactory Message Specification* (MMS) foi escolhido; sua capacidade de manipular objetos lógicos o diferencia de todos os outros padrões existentes. Atualmente a UCA encontra-se na versão 2.0 (UCA2.0).

Em 1997 foi criado um comitê formado pelo *Electric Power Research Institute* (EPRI), *Electrical and Electronics Engineers* (IEEE) e IEC *Technical Committee* 57 para criar um padrão internacional resultando na atual norma IEC-61850 (VICENTE e SENGER, 2011).

Figura 4 - Padrão de comunicação UCA



Fonte: Adaptado de (ADAMIAK, BAIGENT, *et al.*, 2002)

A Norma IEC 61850

1.8. Origem da IEC 61850

A norma IEC 61850 foi desenvolvida com o objetivo de melhorar a interoperabilidade entre os equipamentos instalados em subestações de energia (CLAVEL, SAVARY, *et al.*, 2014). A evolução contínua da norma tem ocorrido através da colaboração conjunta de fabricantes, instituições de pesquisas e órgãos normativos internacionais (VICENTE e SENGER, 2011),

A norma IEC 61850 teve sua primeira versão publicada em 2003, introduzindo princípios de interoperabilidade, quando dois dispositivos ou mais, independentes do fabricante, trocam informações e usam-nas para sua própria funcionalidade (DUARTE, 2012).

A estrutura da IEC 61850 é completamente diferente se comparada a modelos clássicos de protocolo de comunicação, como DNP3, por exemplo, e muda totalmente a maneira de organizar um projeto elétrico de subestação. Comparando ao DNP3, que é apenas um protocolo de comunicação, IEC 61850 é um padrão completo que define, entre outros:

a) modelagem de dados: como descrever os dispositivos em uma subestação elétrica;
b) protocolos de comunicação: como é realizado a troca de informação entre os dispositivos e sistemas.

Nos projetos de subestação de energia elétrica, seguindo os conceitos da norma IEC 61850, a estrutura de comunicação de dados se torna tão importante quanto o projeto elétrico. Consequentemente, para a validação de uma configuração, o Teste de Aceitação de Fábrica (TAF) e o Teste de Aceitação de Campo (TAC) devem ser reconsiderados.

Na Tabela 1 estão apresentadas as dez partes que constituem a norma IEC 61850 em sua primeira versão. A pesquisa contida neste trabalho foi realizada especificamente nas partes 61850-6, 61850-7 e 61850-8.

Tabela 1 – Partes da Norma IEC 61850 versão 1.

Parte	Descrição
1	Introdução e visão global

Parte	Descrição
2	Glossário
3	Requisitos gerais
4	Gerenciamento do sistema e projeto
5	Requisitos de comunicação para funções e modelos de dispositivos
6	Configuração da linguagem de descrição para comunicação em subestações com IEDs
7	Estruturas de comunicação básicas para subestações e alimentadores
7.1	Princípios e modelos
7.2	Interface do Serviço de Comunicação Abstrata (ACSI)
7.3	Classes de dados Comuns (CDC)
7.4	Classes de dados e nós lógicos compatíveis
8	Serviço de mapeamento de comunicação específico
8.1	Mapeamento para MMS (ISSO/IEC 9506 – Partes 1 e 2) e ISSO/IEC 8802-3
9.1	Mapeamento para serviço de comunicação específico (SCSM) – Valores amostrados sobre enlace serial unidirecional ponto a ponto
9.2	Mapeamento para serviço de comunicação específico (SCSM) – Valores amostrados sobre ISSO/IEC 8802-3
10	Testes de conformidade

Fonte: O autor

1.9. Modelagem de Objetos de Dados

A modelagem da comunicação orientada a objeto é a base da arquitetura IEC 61850. Diferente do utilizado em outros protocolos, a identificação dos dispositivos do sistema elétrico e as funções de proteção são feitas por um dicionário de nomes e utiliza uma estrutura hierárquica de objetos;

a) Dispositivo físico (PD, do inglês *physical device*) – destinado a equipamento que executa uma ou mais funções com seu endereço de rede. Exemplo de dispositivos físicos: disjuntores, chaves seccionadoras, relés de proteção, transformador de corrente, transformador de tensão etc.

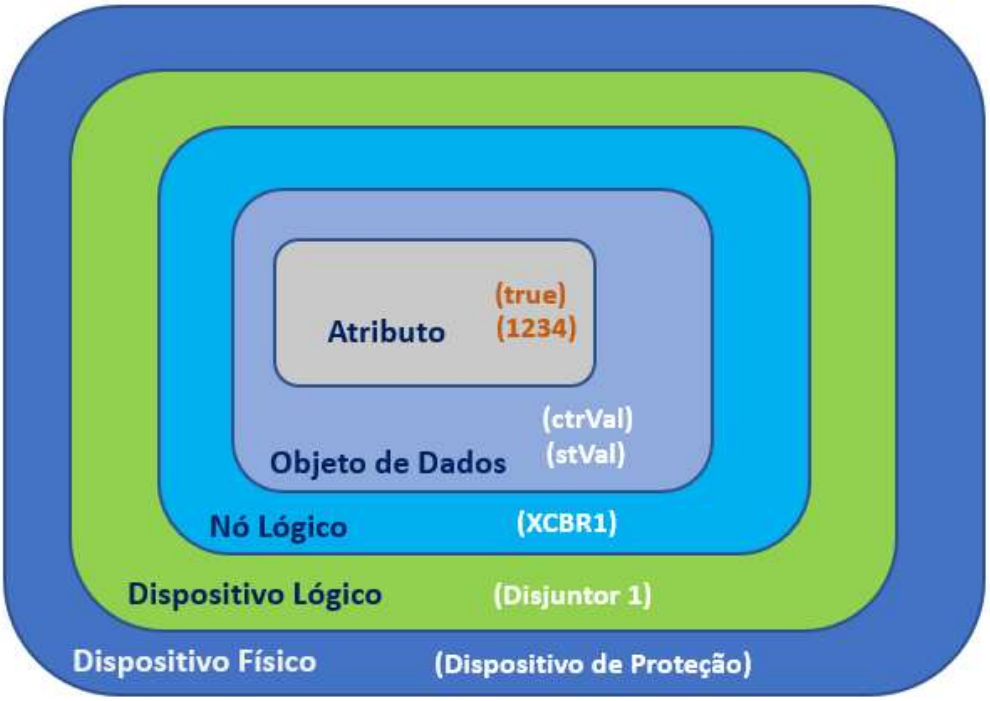
b) Dispositivo lógico (LD, de *logical device*) –dentro de um dispositivo físico, identifica um agrupamento de dados e serviços associados a alguma função do sistema de potência. Por exemplo: controle, medição, etc.
c) Nós lógicos (LN, *logical nodes*) – dentro do dispositivo lógico representam os objetos do sistema elétrico ou funções de proteção. Por exemplo: PDIF – proteção diferencial e XCBR – disjuntor;
d) Objetos de dados (DO, *data objects*) – dentro do nó lógico, carregam informações sobre o tipo de ponto, podendo ser um ponto genérico ou outro, por exemplo, a posição ou estado de um disjuntor.
e) Atributos de dados (DA, *data attibutes*) - cada DO possui uma série de atributos associados a ele, os DA representam o valor do ponto ou sua qualidade, por exemplo, valores de posição do disjuntor e sua qualidade, valores de corrente ou tensão. (COMACCIO, SILVA e COSTA, 2017)

Por exemplo, para o LN$XCBR, o *Logical Node* for *Circuit Breack* (XCBR) representa o disjuntor com os dados POS (posição), para a medição LN$MMUX, o *Logical Node for Measurement* (MMXU) representa a potência, tensões, correntes e impedâncias em um sistema trifásico e oferece centenas de valores: (IEC61850-7-4, 2010)

a) valores de medida;
b) valores de configuração;
c) descrição, e;
d) valores de substituição.

A Figura 5 apresenta a estrutura de dados para um disjuntor identificado pelo *Logical Node* LN$CXBR.

Figura 5 - Estrutura de dados definida pela norma



Fonte: Adaptado de (CLAVEL, SAVARY, *et al.*, 2014)

1.10. Serviços de Comunicação

A Figura 6 descreve uma arquitetura de rede IEC 61850. O sistema SCADA está conectado aos IEDs através do barramento de estação. A comunicação vertical entre o SCADA e os IEDs é realizada através do protocolo MMS, serviço baseado na camada 3 de comunicação TCP/IP com menores requisitos de tempo. O protocolo MMS permite realizar as seguintes ações nos dispositivos: envio de comandos, leitura dos dados, gravação de parâmetros e disponibilização de eventos por ações não solicitadas. (IEC-61850-8-1, 2011)

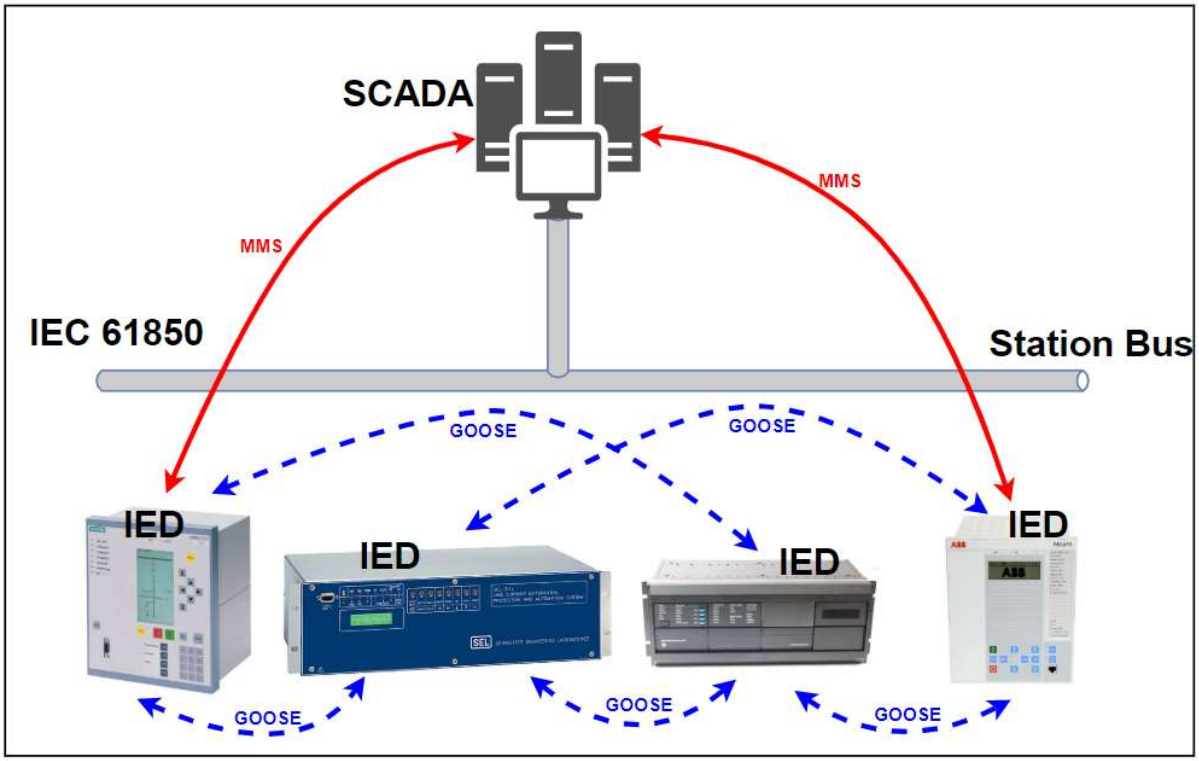
Os datagramas GOOSE informam estados e medições, e são mensagens publicadas e assinadas livremente pelos demais dispositivos.

A Figura 7 apresenta a relação dos protocolos utilizados pela norma IEC 61850 e o modelo OSI. AS mensagens GOOSE são transmitidas diretamente na camada *de Data Link*, por meio de datagramas e são usadas para transmitir informações críticas de proteção.

A Figura 7 mostra também outro tipo de mensagem chamado de *Sampled Value* (SV). Os valores amostrados de sinais analógicos de medições, por exemplo, são digitalizados e transmitidos via rede de comunicação na camada 2 do modelo OSI. Funções críticas de tempo, como por exemplo, valores de correntes e tensões para

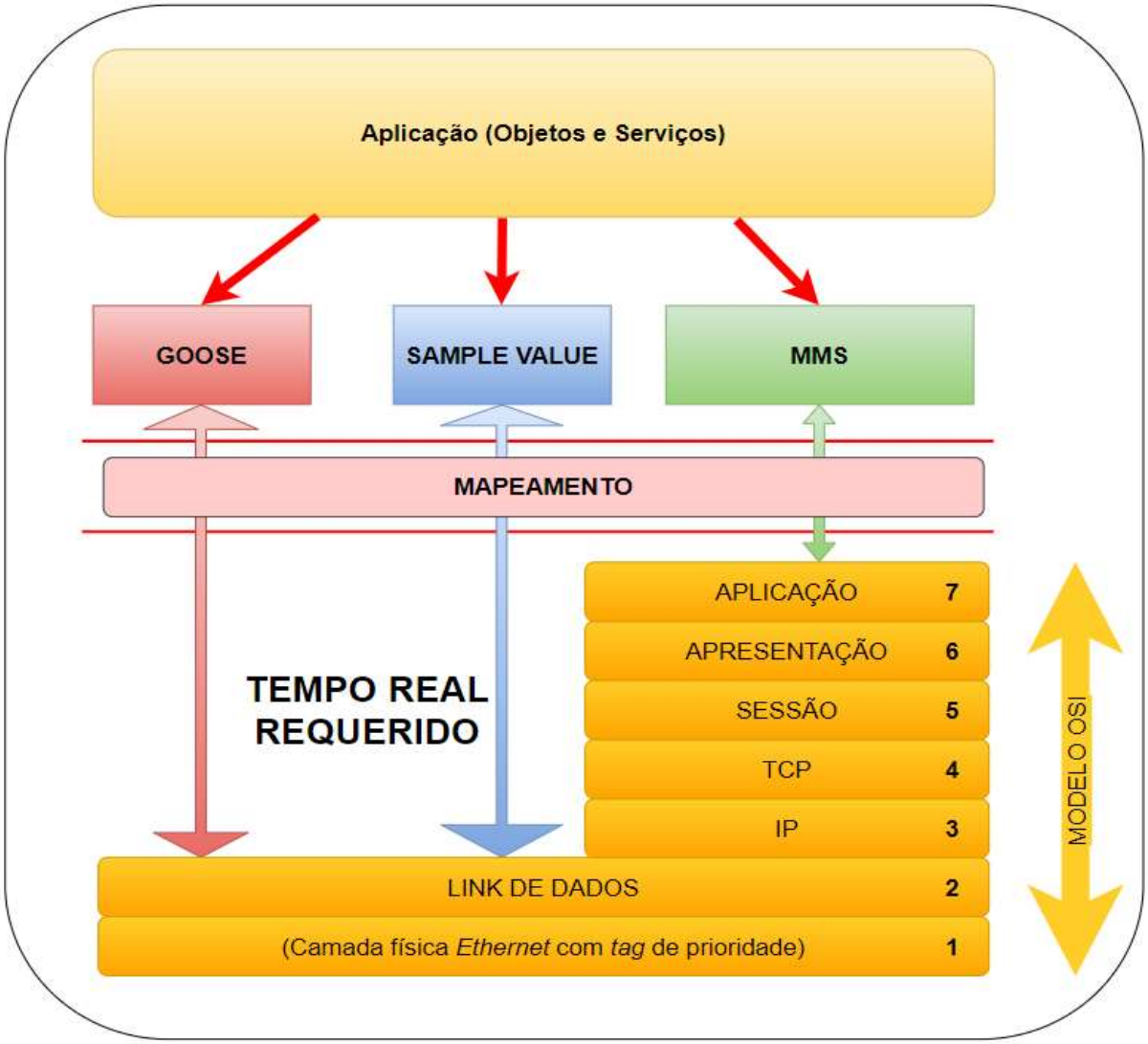
relés de proteção, são transmitidos através das mensagens SV. O protocolo MMS, como pode ser observado na Figura 7, utiliza as sete camadas do modelo de referência OSI para ser transmitido.

Figura 6 - Comunicação em uma arquitetura IEC-61850



Fonte: O autor

Figura 7 - Serviços de comunicação da norma IEC-61850 no modelo OSI



Fonte: O autor

1.11. Mensagens em tempo real GOOSE

O pacote de GOOSE é detalhado na parte 8-1 da norma IEC 61850. Esta parte da norma aborda a estrutura básica de informação e comunicação (IEC-61850-8-1, 2011).

Para fornecer serviços avançados para subestações de energia elétrica, a norma IEC 61850 permite que mensagens críticas entre equipamentos sejam trocadas em tempo real através do barramento de processos e de estação. São alguns exemplos de mensagens críticas:

- Intertravamento;
- transferência de carga;
- transferência automática de fonte.

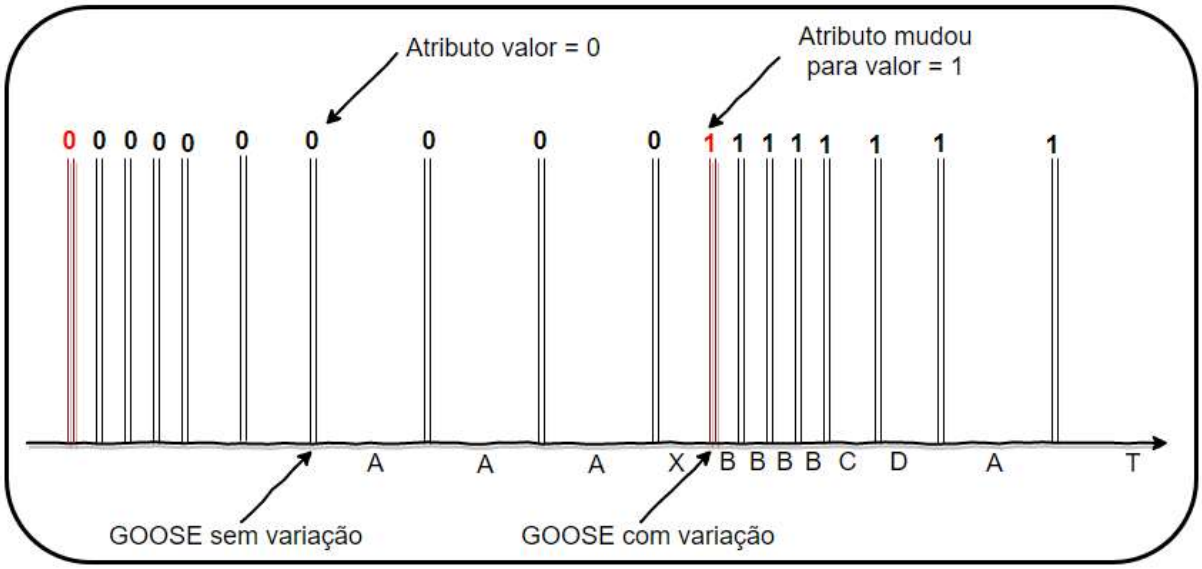
Como apresentado na Figura 7 as mensagens GOOSE são transmitidas diretamente na camada 2 do modelo OSI. As mensagens são enviadas por uma transmissão em *multicast* e possuem alta prioridade na rede. Com a utilização das mensagens a necessidade de uso de cabos elétricos é bastante reduzida. (KIMURA, ABBOUD, *et al.*, 2008)

As mensagens GOOSE são publicadas periodicamente como forma de verificação do correto funcionamento do equipamento publicador e também garantindo que todos os IEDs, incluindo os recém-inseridos na rede, tenham informações atualizadas. As mensagens GOOSE usam uma sequência específica de envio como pode ser observada na Figura 8.

Onde: (VICENTE e SENGER, 2011)

a) A: retransmissão do GOOSE em condições estáveis;
b) X: retransmissão do GOOSE interrompida por um evento;
c) B: retransmissão do evento em períodos curtos;
d) C e D: retransmissão do evento em períodos mais longos até atingir a estabilidade.

Figura 8 - Frequência da transmissão do GOOSE

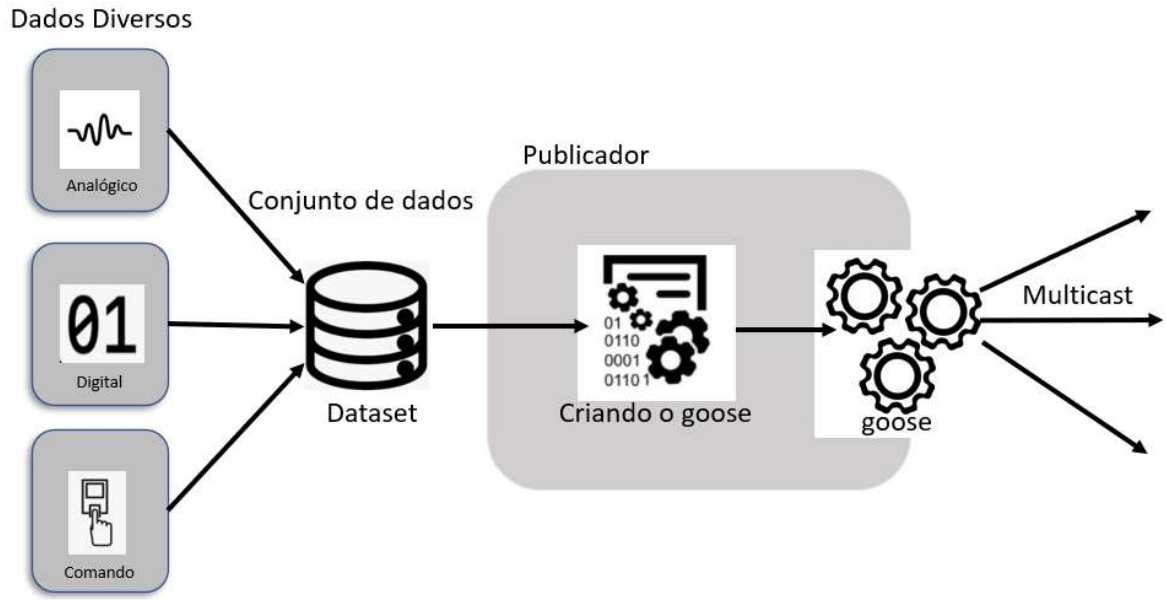


Fonte: Adaptado de (IEC-61850-8-1, 2011)

Os *datasets* (conjuntos de dados) usados na comunicação GOOSE podem ser compostos por dados analógicos, binários ou valores inteiros. A Figura 9 ilustra o processo de publicação de uma mensagem GOOSE. Os vários dados são agrupados

no *dataset*, posteriormente estruturado no IED publicador e por fim transmitido em *multicast* na rede. (VICENTE e SENGER, 2011)

Figura 9 - Mensagem GOOSE publicada em data sets



Fonte: Adaptado de (VICENTE e SENGER, 2011)

As mensagens GOOSE são transmitidas através dos frames *Ethernet* IEEE 802.3. A Tabela 2 mostra uma estrutura do quadro padrão *Ethernet* 802.3 definido pelo IEEE. (TANENBAUM, 1997)

Cada quadro começa com um preâmbulo de 7 *bytes*, cada um contendo o padrão de bit 10101010. Em seguida, vem um *byte* de início de quadro, contendo a sequência de bit 10101011 para sinalizar o início do quadro propriamente dito.

Os quadros seguintes contêm dois endereços de MAC de 2 a 6 *bytes* cada, uma para o destino e um para a origem. O campo comprimento informa quantos *bytes* existem no campo de dados, de um mínimo de 0 a um máximo de 1500. Apesar de válido, um campo de dados de 0 *bytes* causa problemas. O padrão *Ethernet* exige que os quadros válidos tenham pelo menos 64 *bytes*, então, se a parte de dados de um quadro for menor que 64 *bytes*, o campo *padding* será usado para preencher o quadro até o tamanho mínimo.

O campo final do padrão *Ethernet* IEEE 802.3 é o *checksum*. Este campo verifica a integridade do quadro.

Tabela 2 - Estrutura padrão IEEE 802.3, para transmissão de GOOSE

Bytes	Identificação	Sequência binária padrão
7	Preâmbulo	AA
1	Delimitador	AB
2 a 6	MAC GOOSE de destino	Depende da configuração
2 a 6	MAC GOOSE de origem	Depende do fabricante
2	Comprimento	
0 a 1500	Dados	
0 a 46	padding	
4	checksum	

Fonte: O autor

Alguns modelos de *switches* gerenciados possibilitam a configuração de VLANs (*Virtual Local Area Network),* redes virtualizadas configuradas em uma mesma estrutura física, porém separadas de forma lógica.

Cada frame GOOSE publicado contém informação de VLAN ID, prioridade definida pela norma 802.1 Q, e um endereço MAC (Media Access Control) *Multicast*, que permite que uma mesma mensagem GOOSE possa ser assinada por mais de um dispositivo na rede. O MAC *Multicast* é um endereço hexadecimal composto de seis bytes com as seguintes recomendações pela norma:

a) os três primeiros bytes são definidos pela IEEE como 01-0C-CD;

b) o quarto byte deve ser 01 para GOOSE, 02 para *Generic Substation Status Event* (GSSE) e 04 para SV;

c) os dois últimos bytes devem estar na faixa 00-00, 01-FF.

A estrutura da mensagem GOOSE é padronizada pelo *Abstract Syntax Notation One* (ASN.1). Um exemplo da sequência definida pela ASN.1 para as mensagens GOOSE é ilustrada na Figura 10. Observa-se que a estrutura do GOOSE definida pela ASN.1 é constituída por doze variáveis. (IEC-61850-8-1, 2011)

Figura 10 - ANS.1 referência GOOSE

```
IECGoosePdu :: = SEQUENCE {
goocbRef              [0]         IMPLICIT VISIBLE-STRING,
timeAllowedtoLive     [1]         IMPLICIT INTEGER,
datSet                [2]         IMPLICIT VISIBLE-STRING,
goID                  [3]         IMPLICIT VISIBLE-STRING OPTIONAL,
T                     [4]         IMPLICIT UtcTime,
stNum                 [5]         IMPLICIT INTEGER,
sqNum                 [6]         IMPLICIT INTEGER,
simulation            [7]         IMPLICIT BOOLEAN DEFAULT FALSE,
confRev               [8]         IMPLICIT INTEGER,
ndsCom                [9]         IMPLICIT BOOLEAN DEFAULT FALSE,
numDatSetEntries      [10]        IMPLICIT INTEGER,
allData               [11]        IMPLICIT SEQUENCE OF data,
}
```

Fonte: Adaptado de (IEC-61850-8-1, 2011)

Na Tabela 3, é apresentada a estrutura do formato da mensagem GOOSE definida pela ASN.1 e transmitida no padrão *Ethernet* IEEE *802.3*. Alguns valores de sequência hexadecimal padronizado são apresentados. Esses valorem indicam o início de blocos específicos (IEC-61850-8-1, 2011).

Tabela 3 - Estrutura do pacote GOOSE

Bloco	Bytes	Identificação	Sequência hexadecimal padrão
6	2	APPID	
8	1	GOOSE PDU	61
9	1	Tamanho do PDU	
10	1	Quadro de Controle de Referência	80
12	1	Quadro de Referência do tempo de vida do pacote GOOSE	81
14	X	Dado do tempo de vida do pacote GOOSE	
15	1	Quadro de Referência da Identificação do *dataset*	82
17	X	Quadro com a informação de identificação do *dataset*	
18	1	Quadro de Referência do GOOSE ID	83

Bloco	Bytes	Identificação	Sequência hexadecimal padrão
20	X	Quadro com a informação de GOOSE ID	
21	1	Quadro de Referência da Estampa de Tempo do GOOSE	84
23	X	Dado da Estampa de Tempo do PDU no formato epoch.	
24	1	Quadro de referência do número de estado do GOOSE	85
26	X	Dado do número de estado do GOOSE	
27	1	Quadro de referência do número sequencial do GOOSE	86
29	X	Dado do número sequencial do GOOSE	
30	1	Quadro de referência do estado do IED	87
32	1	Quadro com a informação do estado do IED	
33	1	Quadro de referência da identificação da revisão do GOOSE	88
35	X	Dado de identificação da revisão do GOOSE	
36	1	Quadro de referência da informação da necessidade de revisão do GOOSE	89
38	1	Dado com a informação da necessidade de revisão do GOOSE	
39	1	Quadro de referência dos dados de atributos do GOOSE	8A
41	XX	Quadro com a informação da quantidade de atributos contido no pacote GOOSE	
42	1	Quadro de referência do início dos atributos	AB
43	XX	Quadro com o datagrama GOOSE transmitido	
44	0 a 46	padding	
45	4	checksum	

Fonte: O autor

1.12. Arquivos de descrição da configuração

A norma IEC-61850-6 especifica uma linguagem de descrição para configuração de IEDs, denominada *Substation Configuration Description Language* (SCL). O SCL tem como um dos principais objetivos a padronização da nomenclatura utilizada através de um modelo único de descrição de dados. A configuração do SCL é baseada em *eXtensible Modeling Language* (XML), linguagem utilizada no auxílio da troca de dados de configuração do banco de dados entre ferramentas diferentes. Um arquivo SCL possui em sua configuração dados gerais da subestação, dos equipamentos de manobra, das funcionalidades utilizadas dos IEDs e serviços de comunicação de dados. (NETTO, 2012)

Na IEC-61850-6 são definidos cinco diferentes tipos de arquivos SCL utilizados na fase de projetos para compor a arquitetura da subestação (VICENTE e SENGER, 2011). A Tabela 4 apresenta um resumo dos arquivos SCL.

Tabela 4 - Arquivos SCL

Arquivo	Descrição
System Specification Description (SSD)	Descreve o diagrama e a funcionalidade da automação da subestação associado aos nós lógicos
SCD	Descreve a configuração completa da subestação incluindo a rede de comunicação e a informação sobre o fluxo de dados de comunicação
IED Capability Description (ICD)	Descreve as capacidades e pré-configurações dos IEDs
Configuration IED Description (CID)	Descrição da configuração de um IED específico, ou seja, dos dados que serão fornecidos pelos nós lógicos de cada IED
Instantiated IED Description (IID)	Descrição da configuração e recursos específicos de um IED no projeto. É usado como formato de troca de dados do configurador de IED para o configurador do sistema.

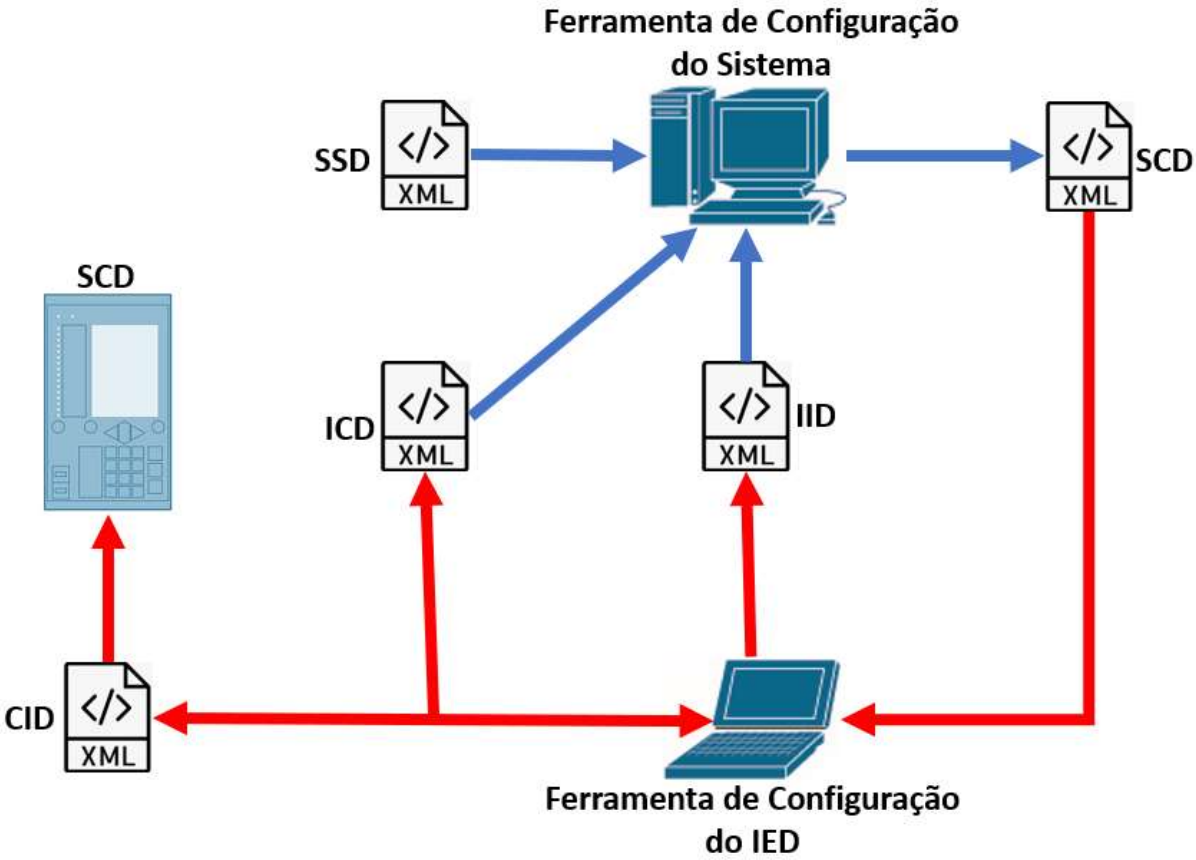
Fonte: O autor

A Figura 11 apresenta a relação entre os diversos arquivos mencionados. No arquivo ICD são detalhados as características e funcionalidades de um IED. No

arquivo SSD estão descritos o diagrama unifilar da SE e os LNs necessários. No arquivo SCD estão contidas informações sobre todos os IEDs, dados da configuração das comunicações, e uma descrição da subestação. O arquivo CID, que pode existir em várias instâncias em uma arquitetura de subestação, descreve um único IED criado dentro do projeto, e inclui informações de *Header, Communication, IED* e *DataTypeTemplates* específico do equipamento. (VICENTE e SENGER, 2011)

As configurações contidas no arquivo CID são transferidas para o equipamento através do software de configuração do fabricante, e as informações do arquivo IID são importadas pelo configurador do sistema.

Figura 11 - Linguagem de configuração da subestação



Fonte: Adaptado de (NETTO, 2012)

1.13. Análise do arquivo SCD

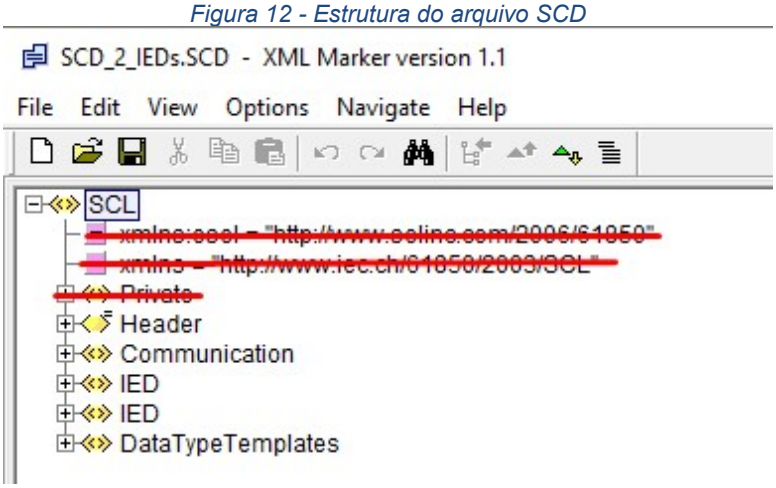
Como comentado no item anterior, o arquivo SCD contém dados sobre todos os IEDs configurados em uma subestação IEC-61850 e pode ser utilizado como referência base do projeto elétrico, físico, e logico de uma subestação IEC-61850.

A estrutura do arquivo é composta por cinco classes principais:

a) SCL:
b) Header:
c) Communication:
d) IED:
e) DataTypeTemplates:

Na Figura 12 uma estrutura de um arquivo SCD contendo dois IEDs é apresentada. O *software* XML *Maker v1.1* foi utilizado para abrir o arquivo. Na estrutura do SCD é possível identificar a classe principal SCL, *Header*, *Communication,* uma classe para o IED1, uma classe para o IED2 e o *DataTypeTemplates.*

Figura 12 - Estrutura do arquivo SCD



Fonte: O autor

Para as mensagens GOOSE trafegadas vale destacar as seguintes estruturas no arquivo SCD: Figura 13 a Fonte: O autor

Figura 15.

a) SCL\IED*AccessPoint**Server*\LN0$CFG\GSEControl: o nome do IED, endereço de MAC, *dataset* publicado e o nome do pacote GOOSE podem ser identificados neste caminho.
b) SCL\IED*AccessPoint**Server*\LN0$CFG\DataSet: listagem dos atributos contidos no *dataset*.
c) SCL\IED*AccessPoint**Server*\LN0$CFG\Inputs; identificação dos atributos externos que devem ser considerados pelo IED.

Figura 13 – Estrutura do LN GSEControl

```
<GSEControl desc="Envio de sinalização dos status de leds dos
botões 1 a 4 do IED2" name="GooseIED2" datSet="DSet01"
confRev="1" appID="IED2">
                    <Private type="SEL_GOOSETXAddress">
                      <esel:Address>
                        <esel:P type="MAC-Address">01-0C-CD-01-
      00-03</esel:P>
                        <esel:P type="APPID">0003</esel:P>
                        <esel:P type="VLAN-ID">001</esel:P>
                        <esel:P type="VLAN-PRIORITY">4</esel:P>
                      </esel:Address>
                    </Private>
                    <Private type="SEL_GOOSETXMinTime">
                                  <esel:MinTime>4</esel:MinTime>
                    </Private>
                    <Private type="SEL_GOOSETXMaxTime">
                      <esel:MaxTime>1000</esel:MaxTime>
                    </Private>
      </GSEControl>
```

Fonte: O autor

Figura 14 - Estrutura da classe DataSet do arquivo SCD

```
<DataSet desc="ATRIBUTOS SEL451-5" name="DSet01">
      <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
      doName="Ind01" daName="stVal" fc="ST" />
      <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
      doName="Ind02" daName="stVal" fc="ST" />
      <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
      doName="Ind03" daName="stVal" fc="ST" />
       <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
      doName="Ind04" daName="stVal" fc="ST" />
</DataSet>
```

Fonte: O autor

Figura 15 - Estrutura da classe Inputs do arquivo SCD

```
<Inputs>
              <ExtRef iedName="IED1" ldInst="ANN" prefix="PBLED"
lnClass="GGIO" lnInst="8" doName="Ind01" daName="stVal"
intAddr="VB001|q=0x1807" serviceType="GOOSE" srcLDInst="CFG"
srcPrefix="" srcLNClass="LLN0" srcCBName="GooseIED1" />
              <ExtRef iedName="IED1" ldInst="ANN" prefix="PBLED"
lnClass="GGIO" lnInst="8" doName="Ind02" daName="stVal"
intAddr="VB002|q=0x1807" serviceType="GOOSE" srcLDInst="CFG"
srcPrefix="" srcLNClass="LLN0" srcCBName="GooseIED1" />
              <ExtRef iedName="IED1" ldInst="ANN" prefix="PBLED"
lnClass="GGIO" lnInst="8" doName="Ind03" daName="stVal"
intAddr="VB003|q=0x1807" serviceType="GOOSE" srcLDInst="CFG"
srcPrefix="" srcLNClass="LLN0" srcCBName="GooseIED1" />
              <ExtRef iedName="IED1" ldInst="ANN" prefix="PBLED"
lnClass="GGIO" lnInst="8" doName="Ind04" daName="stVal"
intAddr="VB004|q=0x1807" serviceType="GOOSE" srcLDInst="CFG"
srcPrefix="" srcLNClass="LLN0" srcCBName="GooseIED1" />
</Inputs>
```

Fonte: O autor

Metodologia de auditoria

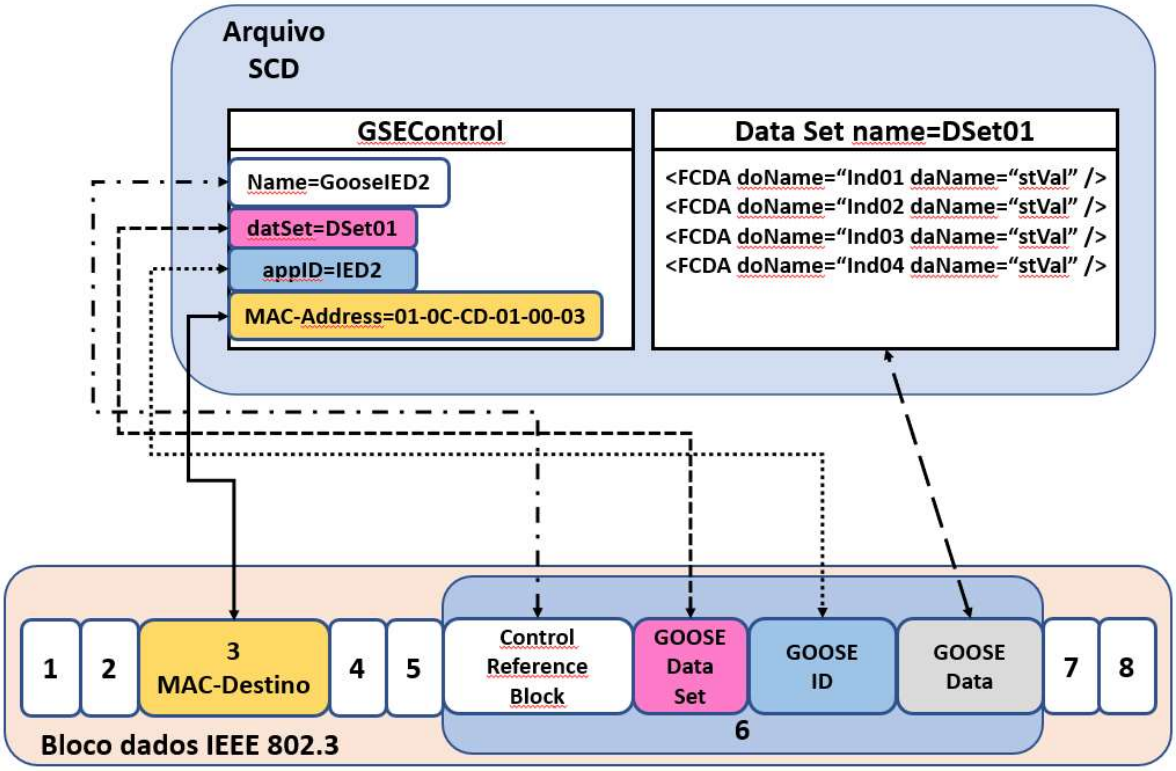
1.14. Processo de Auditoria das Mensagens GOOSE

Conforme explicado no item 1.13, as configurações dos datasets são definidos nos IEDs e a publicação do seu conteúdo é realizado através das mensagens GOOSE. As mensagens publicadas e assinadas pelos dispositivos podem ser identificadas dentro do arquivo SCD. Essas informações são compiladas nos sistemas internos dos IEDs e seu conteúdo é transmitido nos blocos GOOSE que são transportados pelo protocolo padrão Ethernet IEEE 802.3, ambos já discutidos no item 1.11.

As informações equivalentes contidas no arquivo SCD e publicadas na mensagem GOOSE foram identificadas e relacionadas.

A Figura 16 ilustra o encapsulamento das informações GOOSE no bloco de dados do quadro Ethernet 802.3. Um arquivo SCD e as classes GSEControl e DataSet com os atributos mais importantes para auditoria são apresentados. Analisando a relação entre o arquivo SCD e o protocolo Ethernet IEEE 802.3, é possível afirmar que a informação de MAC-Address contida na classe GSEControl é inserida no terceiro quadro do protocolo IEEE802.3, as demais informações estão inseridas no sexto quadro do mesmo protocolo.

Figura 16 - Relacionamento SCD e IEEE802.3

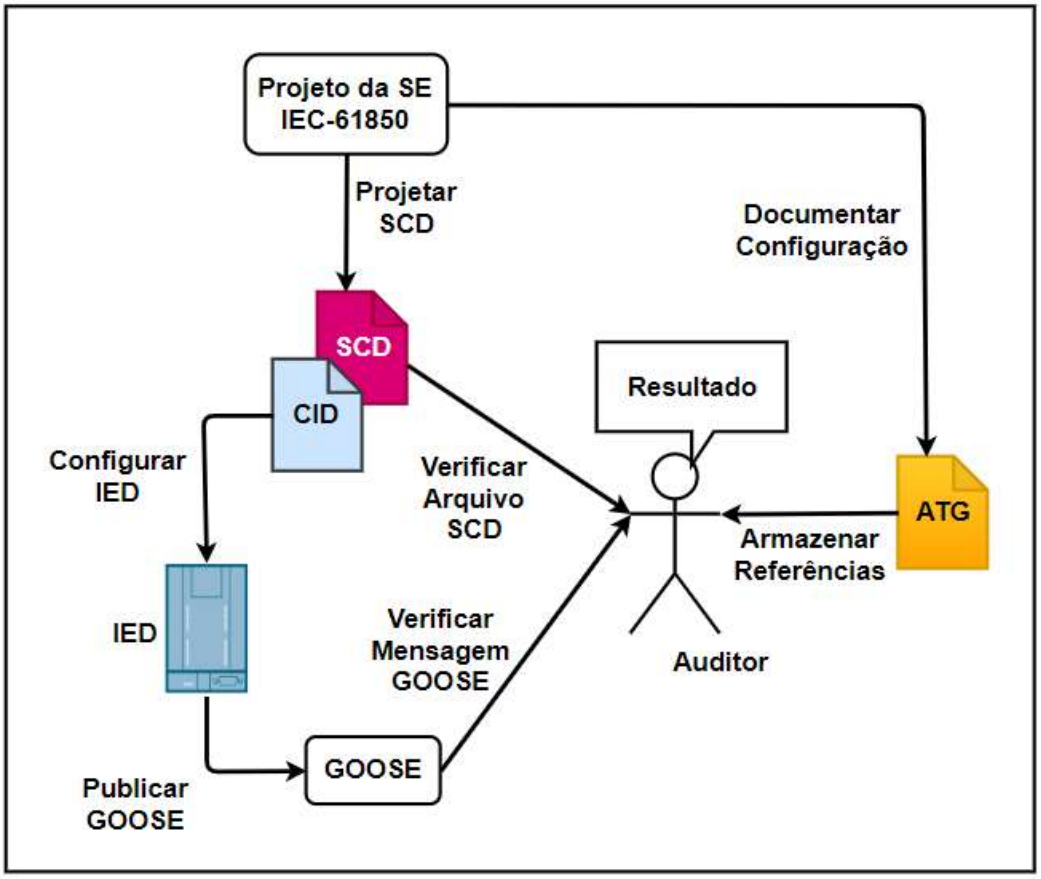


Fonte: O autor

O diagrama da metodologia desenvolvida é apresentado na Figura 17. Um projeto de SE IEC 61850 é elaborado e, como resultado, os arquivos SCD e CID são gerados. Durante o projeto de configuração dos dispositivos, um arquivo ATG é preenchido e disponibilizado como referência para o processo de auditoria. O arquivo ATG faz parte da proposta deste trabalho, e será detalhado ao longo deste capítulo.

Com o projeto da subestação finalizado, as configurações do ICD são descarregadas no IED e então os GOOSE são publicados. As mensagens enviadas pelos dispositivos são capturadas pelo *software* auditor e o conteúdo interno é analisado com base nas configurações dos arquivos SCD e ATG detalhados nos itens 3.6 e 1.16. Se não for identificada divergência na mensagem em relação aos arquivos, os atributos são interpretados e seus valores apresentados para o usuário.

Figura 17 - Metodologia de auditoria e tradução GOOSE



Fonte: O autor

1.15. Aplicativo Auditor GOOSE (AAG)

Foi desenvolvido um *software* auditor, denominado Aplicativo Auditor GOOSE (AAG), capaz de capturar e interpretar as mensagens GOOSE. O AAG foi desenvolvido para ambiente Windows e utiliza a linguagem de programação dotNET C#. Este aplicativo foi compilado na *Integrated Development Environment* (IDE) Visual Studio 2017 da Microsoft.

Para captura das mensagens GOOSE através da plataforma computacional foi instalado o *software* Winpcap. O programa é desenvolvido para sistemas operacionais Windows para capturar pacotes de dados trafegados em redes de computadores. O Winpcap é uma ferramenta gratuita que pode capturar e enviar pacotes de dados na rede de computadores através das interfaces de rede das plataformas computacionais e ao mesmo tempo realizar filtro e armazenar os pacotes capturados. O Winpcap funciona na camada de *drivers* e pode acessar as camadas mais baixa de rede com alta eficiência. (LU, SUN e LI, 2010)

O AAG é constituído por diversas classes e bibliotecas que foram desenvolvidas para garantir o automatismo da metodologia proposta por esta pesquisa. As principais classes e bibliotecas desenvolvidas são apresentadas nos subcapítulos a seguir:

a) *Program:* classe principal de inicialização do *software* AAG. O ANEXO A apresenta um trecho do código fonte da classe *Program*;

b) ThreadValidaRecepçãoGoose classe desenvolvida para gerenciar o tempo de recepção das mensagens GOOSE. Possui um contador em milissegundos que monitora o tempo de recepção da próxima mensagem GOOSE com o mesmo *name*, AppID e *dataset*; se, após o tempo configurado não for identificada a recepção de uma nova mensagem o AAG apresenta mensagem de falha para recepção GOOSE.

c) Classe Epoch: Epoch time é um valor inteiro referente ao tempo em milissegundos armazenado desde (1/1/1970 00:00:00.000). (SYMANTEC, 2010) A classe Epoch foi desenvolvida para tratar valores referentes a estampa de tempo no formato epoch. A Figura 18 ilustra a conversão da data (07/12/2019 02:15:00) para a estampa de tempo epoch em segundos (1575695700) e a conversão da data para estampa de tempo em milissegundos (1575695700000).

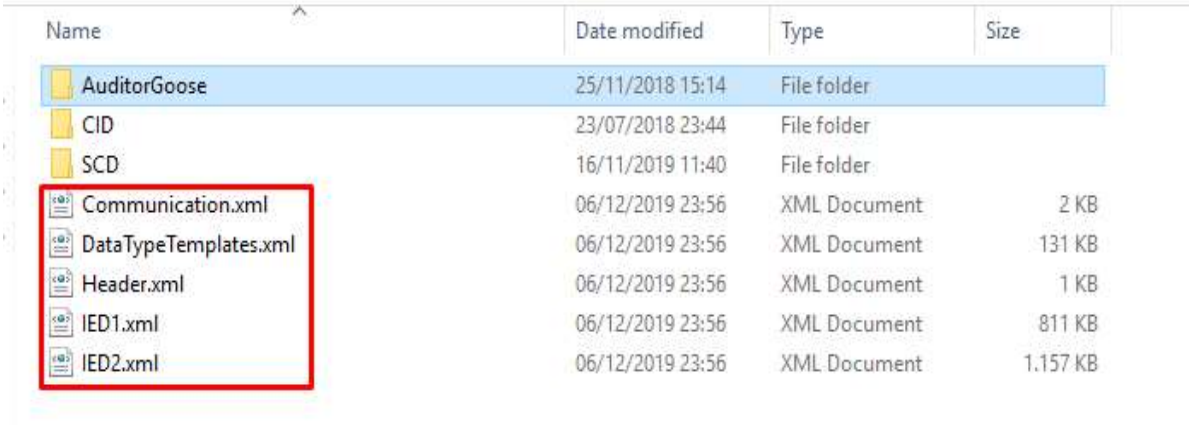
Figura 18 - Conversão epoch

Fonte: O autor

d) Biblioteca SCD: A biblioteca SCD foi desenvolvida para interpretar a estrutura de classes dos arquivos ATG e SCD. Este conjunto de classes também permite identificar as diversas variáveis utilizadas no processo de

auditoria. Outra função da biblioteca SCD é criar arquivos específicos para as classes <Header>, <Communication>, <IED> e <DataTypeTemplates> encontradas no arquivo SCD. A Figura 19 apresenta os arquivos que foram gerados a partir da interpretação do arquivo SCD.

Figura 19 - Classes SCD desmembradas



Fonte: O autor

e) Biblioteca GOOSE: A Biblioteca GOOSE é um conjunto de classes desenvolvidas para interpretar de forma individualizada os blocos da mensagem GOOSE.

A Tabela 5 contém as classes desenvolvidas e suas funcionalidades.

Tabela 5 - Classes desenvolvidas

Classe	Descrição	Anexo
AppID	identifica no quadro GOOSE a informação de APPID	Anexo B
GooseLength	identifica o tamanho do pacote GOOSE recebido pelo AAG	Anexo C
PDU	identifica a sinalização do início da mensagem GOOSE através da informação hexadecimal 61 e o tamanho da mensagem	Anexo D
ControlRefBlock	identifica o quadro *Control Reference Block da* mensagem GOOSE através da informação hexadecimal 80 e extrai a informação contida no quadro	Anexo E
Time2LiveAllowed	identifica a sinalização hexadecimal 81 e extrai a informação do tempo contido na mensagem.	Anexo F

Classe	Descrição	Anexo
DataSet	identifica a sinalização hexadecimal 82 e extrai a informação com o endereço de *DATA SET* contido na mensagem	Anexo G
GooseID	identifica a sinalização hexadecimal 83 e extrai a informação do GOOSE ID	Anexo H
TimeStamp	identifica a sinalização hexadecimal 84 e extrai a informação de *TIME STAMP*	Anexo I
StateNumber	identifica a sinalização hexadecimal 85 e extrai a informação do STATE NUMBER	Anexo J
SequenceNumber	identifica a sinalização hexadecimal 86 e extrai a informação do *SEQUENCE NUMBER DATA*	Anexo K
IEDState	identifica a sinalização hexadecimal 87 e extrai a informação do *IED STATE DATA*	Anexo L
ConfigRevision	identifica a sinalização hexadecimal 88 e extrai a informação do *ConfRev DATA*	Anexo M
NeedRevision	identifica a sinalização hexadecimal 89 e extrai a informação do *NdsCom DATA*	Anexo N
DataEntries	identifica a sinalização hexadecimal 8A e extrai a informação do *Data Entries*	Anexo O
GooseData	identifica divergência na quantidade de atributos a tradução não é realizada	

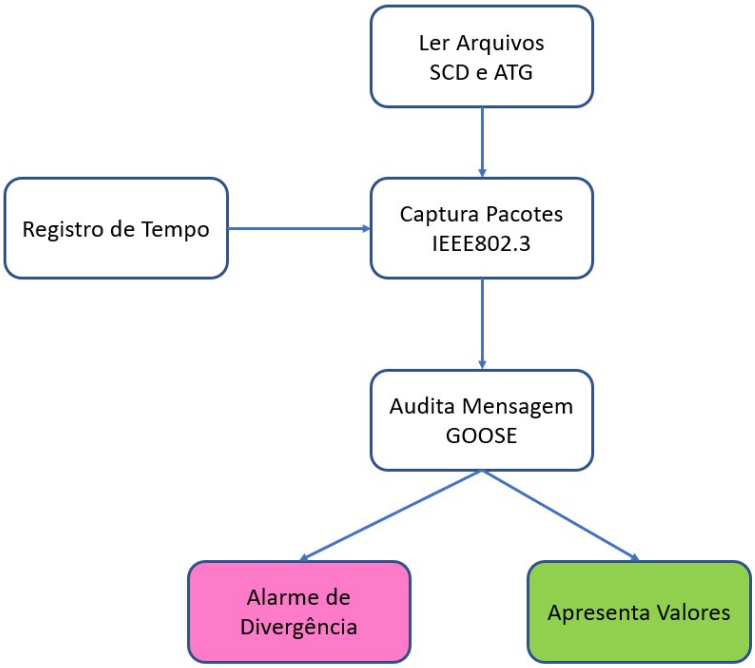
O *Software* Auditor GOOSE desenvolvido neste trabalho pode realizar as seguintes funcionalidades:

- Leitura dos arquivos SCD e ATG;
- Captura dos pacotes de dados inserindo uma identificação de estampa de tempo do horário recebido;
- Verificação dos endereços de MAC dos equipamentos conectados na rede;
- Auditoria do protocolo GOOSE baseado nos arquivos SCD e ATG;
- Tradução dos dados trafegados, apresentando para o usuário o valor do atributo e a respectiva unidade de medida conforme cadastro do ATG;

- Geração de alarme quando detectada divergência entre o dado trafegado e o informado nos arquivos SCD e ATG;

A Figura 20 apresenta a metodologia, explicada no item 1.14, utilizada pelo AAG para realizar a auditoria das mensagens GOOSE capturadas.

Figura 20 - Diagrama de Sequência do Software Auditor GOOSE



Fonte: O autor

Com o auxílio do *software acSELerator Architect* o arquivo SCD_2_IEDs.SCD contendo as configurações IEC61850 do projeto pode ser salvo no diretório c:/Nucleo/IEC61850/SCD da plataforma computacional. O arquivo com as informações dos IEDs utilizados no projeto pode ser analisado como apresentado na Figura 21.

Figura 21 - Estrutura XML do arquivo SCD do projeto

```
<?xml version="1.0" encoding="utf-8"?>
<SCL                xmlns:esel="http://www.selinc.com/2006/61850"
xmlns="http://www.iec.ch/61850/2003/SCL">
  <Header      id="SCD_2_IEDs"      version="26"      revision="1.0"
toolID="acSELerator Architect 2.2.32.0" nameStructure="IEDName" />
  <Communication>
    <SubNetwork name="W01">
      <ConnectedAP iedName="IED2" apName="S1">
        <Address>
          <P type="IP">192.168.10.202</P>
          <P type="IP-SUBNET">255.255.255.0</P>
          <P type="IP-GATEWAY">192.168.10.1</P>
          <P type="OSI-TSEL">0001</P>
          <P type="OSI-PSEL">00000001</P>
```

```
          <P type="OSI-SSEL">0001</P>
        </Address>
        <GSE ldInst="CFG" cbName="GooseIED2">
          <Address>
            <P type="MAC-Address">01-0C-CD-01-00-03</P>
            <P type="APPID">0003</P>
            <P type="VLAN-PRIORITY">4</P>
            <P type="VLAN-ID">001</P>
          </Address>
          <MinTime unit="s" multiplier="m">4</MinTime>
          <MaxTime unit="s" multiplier="m">1000</MaxTime>
        </GSE>
      </ConnectedAP>
      <ConnectedAP iedName="IED1" apName="S1">
        <Address>
          <P type="IP">192.168.10.201</P>
          <P type="IP-SUBNET">255.255.255.0</P>
          <P type="IP-GATEWAY">192.168.10.10</P>
          <P type="OSI-TSEL">0001</P>
          <P type="OSI-PSEL">00000001</P>
          <P type="OSI-SSEL">0001</P>
        </Address>
        <GSE ldInst="CFG" cbName="GooseIED1">
          <Address>
            <P type="MAC-Address">01-0C-CD-01-00-04</P>
            <P type="APPID">0004</P>
            <P type="VLAN-PRIORITY">4</P>
            <P type="VLAN-ID">001</P>
          </Address>
          <MinTime unit="s" multiplier="m">4</MinTime>
          <MaxTime unit="s" multiplier="m">1000</MaxTime>
        </GSE>

            <GSEControl desc="Envio de sinalização dos status de
leds dos botões 1 a 4 do IED1" name="GooseIED1" datSet="DSet01"
confRev="1" appID="IED1">
              <Private type="SEL_GOOSETXAddress">
                <esel:Address>
                  <esel:P        type="MAC-Address">01-0C-CD-01-00-
04</esel:P>
                  <esel:P type="APPID">0004</esel:P>
                  <esel:P type="VLAN-ID">001</esel:P>
                  <esel:P type="VLAN-PRIORITY">4</esel:P>
                </esel:Address>
              </Private>
              <Private type="SEL_GOOSETXMinTime">
                <esel:MinTime>4</esel:MinTime>
              </Private>
              <Private type="SEL_GOOSETXMaxTime">
                <esel:MaxTime>1000</esel:MaxTime>
              </Private>
            </GSEControl>

            <DataSet desc="ATRIBUTOS SEL487E-3" name="DSet01">
              <FCDA  ldInst="ANN"  prefix="PBLED"  lnClass="GGIO"
lnInst="8" doName="Ind01" daName="stVal" fc="ST" />
```

```
              <FCDA   ldInst="ANN"   prefix="PBLED"   lnClass="GGIO"
lnInst="8" doName="Ind02" daName="stVal" fc="ST" />
              <FCDA   ldInst="ANN"   prefix="PBLED"   lnClass="GGIO"
lnInst="8" doName="Ind03" daName="stVal" fc="ST" />
              <FCDA   ldInst="ANN"   prefix="PBLED"   lnClass="GGIO"
lnInst="8" doName="Ind04" daName="stVal" fc="ST" />
            </DataSet>


            <GSEControl desc="Envio de sinalização dos status de
leds dos botões 1 a 4 do IED2" name="GooseIED2" datSet="DSet01"
confRev="1" appID="IED2">
              <Private type="SEL_GOOSETXAddress">
                <esel:Address>
                  <esel:P        type="MAC-Address">01-0C-CD-01-00-
03</esel:P>
                  <esel:P type="APPID">0003</esel:P>
                  <esel:P type="VLAN-ID">001</esel:P>
                  <esel:P type="VLAN-PRIORITY">4</esel:P>
                </esel:Address>
              </Private>
              <Private type="SEL_GOOSETXMinTime">
                <esel:MinTime>4</esel:MinTime>
              </Private>
              <Private type="SEL_GOOSETXMaxTime">
                <esel:MaxTime>1000</esel:MaxTime>
              </Private>
            </GSEControl>

            <DataSet desc="ATRIBUTOS SEL451-5" name="DSet01">
              <FCDA   ldInst="ANN"   prefix="PBLED"   lnClass="GGIO"
lnInst="8" doName="Ind01" daName="stVal" fc="ST" />
              <FCDA   ldInst="ANN"   prefix="PBLED"   lnClass="GGIO"
lnInst="8" doName="Ind02" daName="stVal" fc="ST" />
              <FCDA   ldInst="ANN"   prefix="PBLED"   lnClass="GGIO"
lnInst="8" doName="Ind03" daName="stVal" fc="ST" />
              <FCDA   ldInst="ANN"   prefix="PBLED"   lnClass="GGIO"
lnInst="8" doName="Ind04" daName="stVal" fc="ST" />
            </DataSet>
```

Fonte: O autor

1.16. Arquivo de Tradução GOOSE (ATG)

A metodologia de auditoria das mensagens GOOSE proposta neste trabalho tem como objetivo identificar e informar de forma automática da ocorrência das seguintes falhas:

- Divergência na quantidade de atributos publicados no *dataset* e informados no projeto inicial;
- Ausência de mensagens GOOSE indicadas no arquivo SCD e ATG;

- Tráfego de mensagens GOOSE na rede local não informados no projeto inicial;
- Falha de sincronismo de tempo dos IEDs publicadores de mensagens GOOSE.

O ATG foi elaborado e desenvolvido neste trabalho e tem como objetivo garantir a integridade das informações auditáveis do arquivo SCD definidas no projeto inicial. Durante o desenvolvimento do *software* auditor foi identificado a possibilidade de integrar uma funcionalidade que permitisse a tradução dos atributos e acrescentar as unidades de medidas desses atributos contidos nas mensagens GOOSE visualizadas pelos usuários.

A partir desta motivação foram inseridos no ATG um campo para descrição do atributo de acordo com a tradução do usuário e um campo para descrição da unidade de medida. Outra funcionalidade atribuída ao ATG e desenvolvida durante o projeto foi a possibilidade da inserção de condições da apresentação dos resultados baseado nos valores contidos nos atributos, assim, os usuários podem visualizar o valor contido no atributo como informação sem a necessidade de utilizar softwares para interpretação.

O ATG possui então as seguintes funcionalidades:

- Documentar as informações GOOSE do arquivo SCD declarados no projeto inicial;
- Auxiliar o time técnico na interpretação dos dados GOOSE trafegados na rede;
- Permite ao usuário inserir condições que serão apresentadas como informação de acordo com o valor contido no atributo capturado.

O ATG é escrito utilizando a linguagem XML e possui em sua estrutura interna uma classe para cada dispositivo inserido no processo de auditoria. A estrutura da classe IED do arquivo ATG contém as seguintes subclasses:

- <IED *name*=" nome_do_ied"> : Identificação para o início dos dados de um IED. O campo *name* deve ser único para
- <*DATASET name*="identificação_*dataset*">: Identificação do *dataset* publicado pelo IED

- <ITEM id="posição_*dataset*">: Identificação do atributo, o id é a posição do atributo dentro do *dataset.* Para esse item são inseridas as seguintes informações:
    a) <LN>: Endereço lógico da informação conforme a norma IEC-61850;
    b) <*TRANSLATION*>: Tradução da informação para o usuário;
    c) <*UNIT*>: Campo utilizado para indicar o tipo de informação do dado
    d) <*CONVERT*>: Classe contendo as possíveis condições interpretadas pelo usuário.
- <*SUBSCRIBER*>: Identificação da classe de registro de GOOSE assinado
- <*PUBLISHER name*=”nome_do_ied”>: Identificação do IED que publicará as mensagens que devem ser assinadas.
- <*DATASET*_P *name*=”nome_do_*dataset”>:* Identificação do *dataset* que contém os atributos que deverão ser assinados pelo IED
- <*ATTRIBUTE*>: Posição do atributo no *dataset* para assinatura

A estrutura do arquivo ATG é apresentada na Figura 22.

Figura 22 - Estrutura do arquivo ATG

```
<?xml version="1.0" encoding="utf-8"?>
<GDT xmlns:esel=" " xmlns=" ">
     <IED name="IED2">
          <DATASET name="DSet01">
               <VAR id="1">
                    <ITEM id="1">
     <LN>IED2/CFG/LLNO/GooseIED2.ANN.PBLEDGGIO8.Ind01.stVal</LN>
                         <TRANSLATION>Push-button                1
IED2</TRANSLATION>
                         <UNIT>Status</UNIT>
                         <CONVERT>
                                   <OPTION>00=PB1     is     not
pushed</OPTION>
                                   <OPTION>01=PB1              is
pushed</OPTION>
                         </CONVERT>
                    </ITEM>
                    <ITEM id="2">

     <LN>IED2/CFG/LLNO/GooseIED1.ANN.PBLEDGGIO8.Ind02.stVal</LN>
                              <TRANSLATION>     Push-button     2
IED2</TRANSLATION>
                              <UNIT>Status</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB2   is   not
pushed</OPTION>
                                        <OPTION>01=PB2        is
pushed</OPTION>
                              </CONVERT>
```

```
                              </ITEM>
                              <ITEM id="3">

     <LN>IED2/CFG/LLNO/GooseIED2.ANN.PBLEDGGIO8.Ind03.stVal</LN>
                                   <TRANSLATION>     Push-button     3
IED2</TRANSLATION>
                                   <UNIT>Status</UNIT>
                                   <CONVERT>
                                             <OPTION>00=PB3   is   not
pushed</OPTION>
                                             <OPTION>01=PB3        is
pushed</OPTION>
                                   </CONVERT>
                              </ITEM>
                              <ITEM id="4">

     <LN>IED2/CFG/LLNO/GooseIED2.ANN.PBLEDGGIO8.Ind04.stVal</LN>
                                   <TRANSLATION>     Push-button     4
IED2</TRANSLATION>
                                   <UNIT>Status</UNIT>
                                   <CONVERT>
                                             <OPTION>00=PB4   is   not
pushed</OPTION>
                                             <OPTION>01=PB4        is
pushed</OPTION>
                                   </CONVERT>
                              </ITEM>
                         </VAR>
                    </DATASET>
                    <SUBSCRIBER>
                         <PUBLISHER name="IED1">
                              <DATASET_P name="DSet01">
                                   <ATTRIBUTE>1</ATTRIBUTE>
                                   <ATTRIBUTE>2</ATTRIBUTE>
                                   <ATTRIBUTE>3</ATTRIBUTE>
                                   <ATTRIBUTE>4</ATTRIBUTE>
                              </DATASET_P>
                         </PUBLISHER>
                    </SUBSCRIBER>
               </IED>
     </GDT>
```

Fonte: O autor

O arquivo ATG pode ser configurado com informações relacionadas ao arquivo SCD definido no projeto inicial. A Figura 23 apresenta o arquivo ATG configurado.

Figura 23 - Arquivo ATG configurado no projeto

```
<?xml version="1.0" encoding="utf-8"?>
<GDT xmlns:esel=" " xmlns=" ">
     <IED name="IED2">
          <DATASET name="DSet01">
               <VAR id="1">
                    <ITEM id="1">
```

```
     <LN>IED2/CFG/LLNO/GooseIED2.ANN.PBLEDGGIO8.Ind01.stVal</LN>
                              <TRANSLATION>  IED2  push-button  1  status
</TRANSLATION>
                              <UNIT>Valor</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB1      is      not
pushed</OPTION>
                                        <OPTION>01=PB1               is
pushed</OPTION>
                              </CONVERT>
                         </ITEM>
                         <ITEM id="2">

     <LN>IED2/CFG/LLNO/GooseIED2.ANN.PBLEDGGIO8.Ind02.stVal</LN>
                              <TRANSLATION>  IED2  push-button  2  status
</TRANSLATION>
                              <UNIT>Valor</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB2      is      not
pushed</OPTION>
                                        <OPTION>01=PB2               is
pushed</OPTION>
                              </CONVERT>
                         </ITEM>
                         <ITEM id="3">

     <LN>IED2/CFG/LLNO/GooseIED2.ANN.PBLEDGGIO8.Ind03.stVal</LN>
                              <TRANSLATION>  IED2  push-button  3  status
</TRANSLATION>
                              <UNIT>Valor</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB3      is      not
pushed</OPTION>
                                        <OPTION>01=PB3               is
pushed</OPTION>
                              </CONVERT>
                         </ITEM>
                         <ITEM id="4">

     <LN>IED2/CFG/LLNO/GooseIED2.ANN.PBLEDGGIO8.Ind04.stVal</LN>
                              <TRANSLATION>  IED2  push-button  4  status
</TRANSLATION>
                              <UNIT>Valor</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB4      is      not
pushed</OPTION>
                                        <OPTION>01=PB4               is
pushed</OPTION>
                              </CONVERT>
                         </ITEM>
                    </VAR>
               </DATASET>
               <SUBSCRIBER>
                    <PUBLISHER name="IED1">
                         <DATASET_P name="DSet01">
```

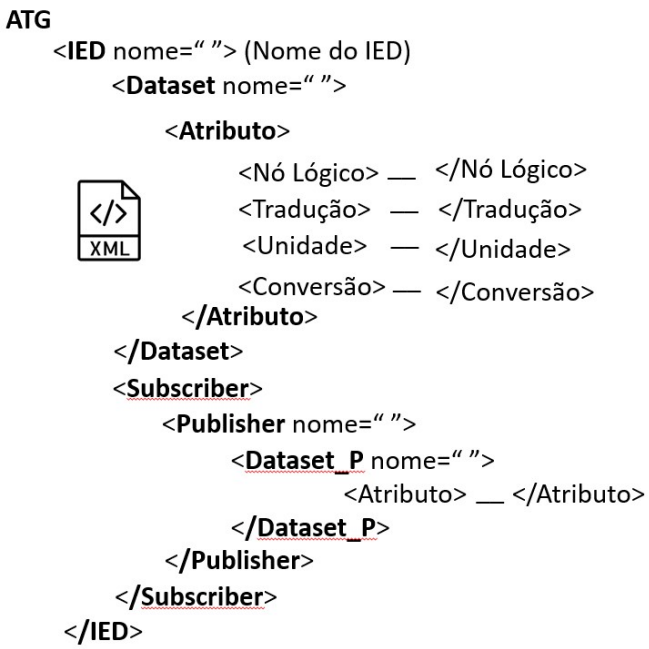
```
                              <ATTRIBUTE>1</ATTRIBUTE>
                              <ATTRIBUTE>2</ATTRIBUTE>
                              <ATTRIBUTE>3</ATTRIBUTE>
                              <ATTRIBUTE>4</ATTRIBUTE>
                         </DATASET_P>
                    </PUBLISHER>
               </SUBSCRIBER>
          </IED>
          <IED name="IED1">
               <DATASET name="DSet01">
                    <VAR id="1">
                         <ITEM id="1">

          <LN>IED1/CFG/LLNO/GooseIED1.ANN.PBLEDGGIO8.Ind01.stVal</LN>
                              <TRANSLATION>  IED1  push-button  1  status
</TRANSLATION>
                              <UNIT>Valor</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB1      is      not
pushed</OPTION>
                                        <OPTION>01=PB1               is
pushed</OPTION>
                              </CONVERT>
                         </ITEM>
                         <ITEM id="2">

          <LN>IED1/CFG/LLNO/GooseIED1.ANN.PBLEDGGIO8.Ind02.stVal</LN>
                              <TRANSLATION>  IED1  push-button  1  status
</TRANSLATION>
                              <UNIT>Valor</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB2      is      not
pushed</OPTION>
                                        <OPTION>01=PB2               is
pushed</OPTION>
                              </CONVERT>
                         </ITEM>
                         <ITEM id="3">

          <LN>IED1/CFG/LLNO/GooseIED1.ANN.PBLEDGGIO8.Ind03.stVal</LN>
                              <TRANSLATION>  IED1  push-button  1  status
</TRANSLATION>
                              <UNIT>Valor</UNIT>
                              <CONVERT>
                                        <OPTION>00=PB3      is      not
pushed</OPTION>
                                        <OPTION>01=PB3               is
pushed</OPTION>
                              </CONVERT>
                         </ITEM>
                         <ITEM id="4">

          <LN>IED1/CFG/LLNO/GooseIED1.ANN.PBLEDGGIO8.Ind04.stVal</LN>
                              <TRANSLATION>  IED1  push-button  1  status
</TRANSLATION>
                              <UNIT> </UNIT>
```

```
                              <CONVERT>
                                        <OPTION>00=PB4       is       not
pushed</OPTION>
                                        <OPTION>01=PB4                 is
pushed</OPTION>
                              </CONVERT>
                        </ITEM>
                  </VAR>
            </DATASET>
            <SUBSCRIBER>
                  <PUBLISHER name="IED2">
                        <DATASET_P name="DSet1">
                              <ATTRIBUTE>1</ATTRIBUTE>
                              <ATTRIBUTE>2</ATTRIBUTE>
                              <ATTRIBUTE>3</ATTRIBUTE>
                              <ATTRIBUTE>4</ATTRIBUTE>
                        </DATASET_P>
                  </PUBLISHER>
            </SUBSCRIBER>
      </IED>
</GDT>
```

Fonte: O autor

A Figura 24 apresenta a estrutura padrão desenvolvida para o arquivo ATG.

Figura 24 - Esquema Macro do Arquivo ATG



Fonte: O autor

A Tabela 6 contém as principais variáveis configuradas no arquivo ATG para os IEDs em testes com base na interpretação dos dados para o usuário, uma arquitetura de teste é apresentada no capítulo 0. O campo IED contém a identificação do IED para o *software* auditor, o campo *data set*; a identificação do *dataset* publicado

pelo dispositivo; nó lógico contém o endereço de nó lógico do atributo publicado pelo equipamento, o campo tradução contém a tradução do nó lógico na visão do usuário, o campo condição contém as possíveis condições de resultado que será apresentada com base no valor do dado contido no atributo capturado pelo *software* auditor.

Tabela 6 - Tabela de tradução dos atributos publicados

IED	data set	nó lógico	tradução	condição
IED1	Dset1	IED1.ANN.PBLEDGGIO8.Ind01.stVal	IED1 push-button 1 status	00 = PB1 is not pushed 01 = PB1 is pushed
		IED1.ANN.PBLEDGGIO8.Ind02.stVal	IED1 push-button 2 status	00 = PB1 is not pushed 01 = PB1 is pushed
		IED1.ANN.PBLEDGGIO8.Ind03.stVal	IED1 push-button 3 status	00 = PB1 is not pushed 01 = PB1 is pushed
		IED1.ANN.PBLEDGGIO8.Ind04.stVal	IED1 push-button 4 status	00 = PB1 is not pushed 01 = PB1 is pushed
IED2	Dset1	IED2.ANN.PBLEDGGIO8.Ind01.stVal	IED2 push-button 1 status	00 = PB1 is not pushed 01 = PB1 is pushed
		IED2.ANN.PBLEDGGIO8.Ind02.stVal	IED2 push-button 2 status	00 = PB1 is not pushed 01 = PB1 is pushed
		IED2.ANN.PBLEDGGIO8.Ind03.stVal	IED2 push-button 3 status	00 = PB1 is not pushed 01 = PB1 is pushed
		IED2.ANN.PBLEDGGIO8.Ind04.stVal	IED2 push-button 4 status	00 = PB1 is not pushed 01 = PB1 is pushed

Fonte: O autor

Com os arquivos SCD e ATG configurados o *software* auditor pode ser executado. A primeira ação do AAG é identificar no arquivo SCD os parâmetros: *Message name,* IED ID, *dataset* e a quantidade de atributos do *dataset* especificado pelo usuário. A **Erro! Autoreferência de indicador não válida.** apresenta o resultado da identificação das variáveis realizada pelo *software*.

Figura 25 - Identificação de IED e Dataset no SCD

```
c:\\Nucleo\\IEC61850\\SCD\\SCD_2_IEDs.SCD
        <GSE ldInst="CFG" cbName="GooseIED2">
        <GSE ldInst="CFG" cbName="GooseIED1">
IED NAME = IED2
DSet01 - <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind01" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind02" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind03" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind04" daName="stVal" fc="ST" />
IED NAME = IED1
DSet01 - <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind01" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind02" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind03" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind04" daName="stVal" fc="ST" />
```

Fonte: O autor

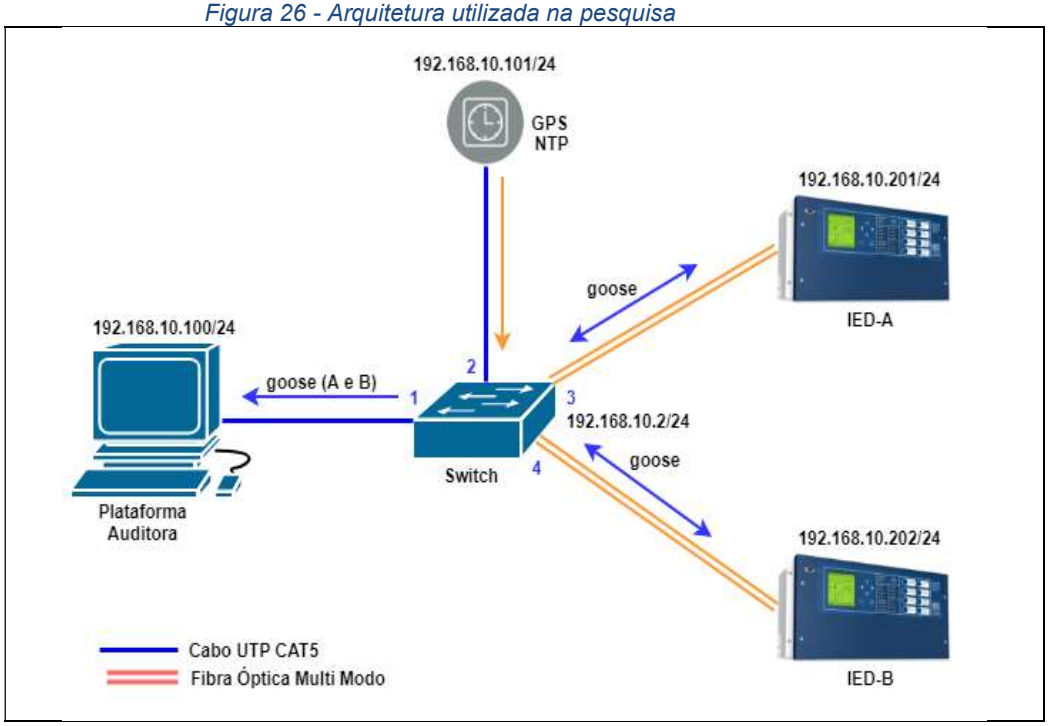
Após o AAG identificar as informações do arquivo SCD, o arquivo ATG é carregado e as informações são comparadas. Não sendo identificadas divergências nos arquivos SCD e ATG, o AAG inicializa a captura dos pacotes trafegados na interface de rede da plataforma computacional, inserindo o valor de *timestamp* da recepção em todas as mensagens GOOSE identificadas. A próxima ação do AAG é analisar os dados contidos nas mensagens capturas, como mencionado no item 1.15.

Arquitetura de ensaio

1.17. Ambiente de ensaio

As Figura 26 e Figura 27 apresentam a arquitetura e os equipamentos utilizados no ambiente de laboratório para realização dos ensaios. Na arquitetura preparada, um *switch* adequado à norma IEC-61850, gerenciável, foi utilizado para simulação de uma LAN contendo os equipamentos utilizados no ensaio. Os IEDs utilizados nos testes foram configurados de tal forma que a troca de mensagens GOOSE pôde ser realizada. Uma plataforma computacional com o *software* auditor instalado foi conectada ao *switch*, possibilitando assim, capturar e analisar as mensagens GOOSE publicadas pelos IEDs. Para garantir a mesma estampa de tempo dos dispositivos utilizados no ambiente de teste um *Network Time Protocol* (NTP) Server, com antena externa, foi configurado e conectado ao switch do ambiente.

Figura 26 - Arquitetura utilizada na pesquisa



Fonte: O autor

Figura 27 - Arquitetura de proposta para avaliação

Fonte: O autor

O laboratório foi montado em um ambiente privado e foi composto pelos seguintes equipamentos:

- um IED da *Schweitzer Engineering Laboratories* - SEL, modelo 451-X . *Part number* 04515615XC2X4H224, *firmware* SEL-451-5-R314-V0-Z019012-D20130618;
- um IED da SEL, modelo 487-X, *Part number* 0487E3X611XXC2, *firmware* SEL-487E-3R310-V0-Z104101-D2140515;
- um *switch* Siemens, modelo RX-1501, *Part number* RX1501-L2-MNT-HIP-L2SE-CG01-6TX01-4FX11-4FX11-4FX11-4FX11;
- um roteador Cisco, modelo CGR-1120, com a função *NTP Server* habilitada.
- dois cabos ópticos com conectores *Local Connector* - LC para conectar os IEDS ao switch;
- dois cabos *unshielded twisted pair* (UTP) para conectar a plataforma computacional e o *NTP Server* ao *switch*;
- uma plataforma computacional com *Windows* 10 *Professional* 64*bits*.
- uma interface de rede para computador da marca Realtek modelo PCIe GBE Family *controller*;

Os seguintes programas foram instalados na plataforma computacional:

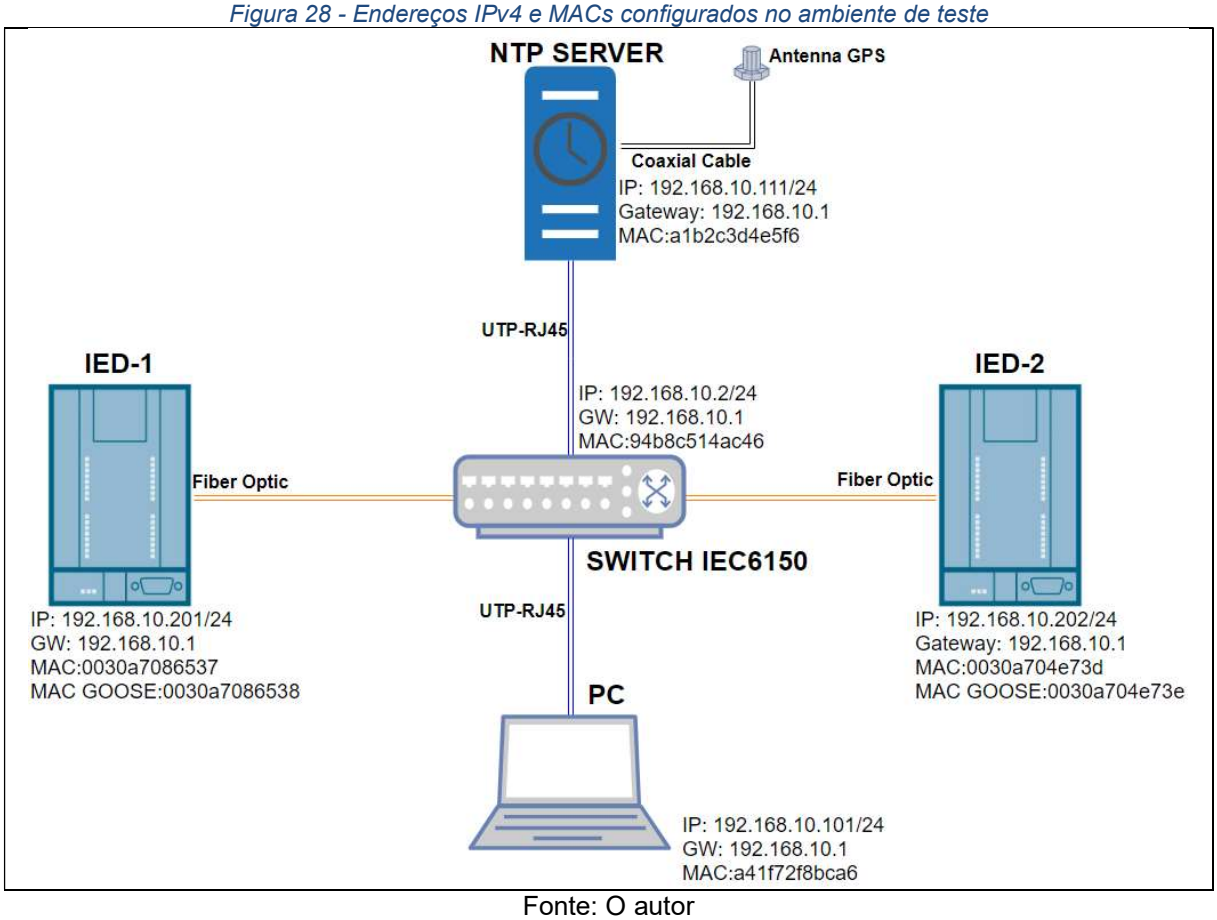
- **SEL-5032 acSELerator Architect versão 2.2.32.0**: programa da SEL, para configurar a arquitetura dos IEDs em subestações com IEC61850.

Possibilita a criação e o mapeamento das mensagens GOOSE, utiliza *reports* pré-definidos, cria e edita bases de dados e lê arquivos SCD, ICD e CID;

- **SEL-5030 acSELerator Quickset versão 6.7.6.1**: programa da SEL para configurar lógicas e os parâmetros de proteção dos IEDs;
- **Wireshark *Network Protocol Analyzer* versão 3.0.1(v3.0.1-0-gea351cd8)**: analisador gratuito de protocolos de rede Ethernet mantido por voluntários especialistas em rede ao redor do mundo.
- **Google Chrome versão 78.0.39.04.87 64bits**: *web browser* da empresa Google utilizado para configuração do *switch*.
- **Putty versão 0.70**: Programa utilizado para acessar e configurar o NTP

Para a arquitetura simplificada implantada no laboratório, cada dispositivo foi configurado com um endereço IPv4 para acesso à configuração de parâmetros. Além dos endereços IPv4 e da máscara de rede, um endereço de gateway também foi configurado. A configuração detalhada é apresentada na Figura 28.

Figura 28 - Endereços IPv4 e MACs configurados no ambiente de teste



Fonte: O autor

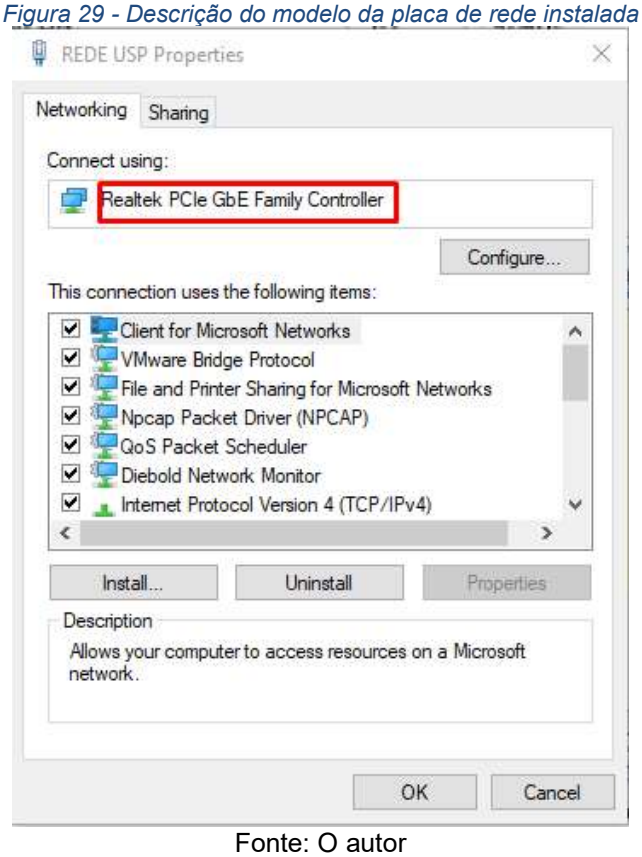
1.18. Configuração do ambiente de ensaio

Para a arquitetura simplificada elaborada em laboratório, foram inseridos endereços IPs diferentes para os devices de forma a possibilitar o acesso para configuração. Além dos IPs, e das sub redes, foi configurado o endereço do servidor de tempo para garantir a validação de detecção de falha de sincronismo através da estampa de tempo da mensagem GOOSE.

1.18.1. Configuração da interface de rede do PC

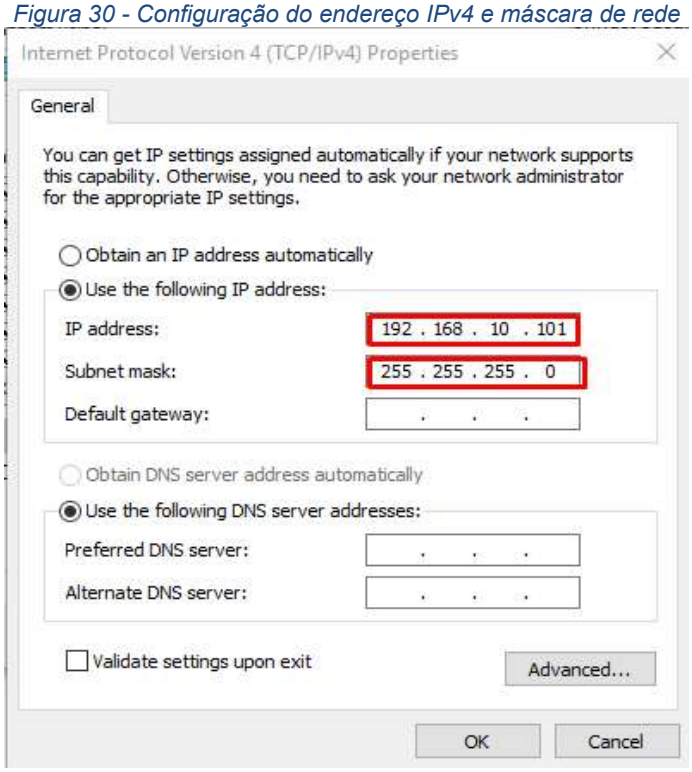
Uma forma de integrar a plataforma computacional a rede de comunicação do ambiente de teste, foi utilizar uma placa de rede padrão *ethernet* do modelo *Realtek* PCIe GbE *Family Controller* configurado com um endereço IPv4 fixo, no mesmo intervalo permitido aos IEDs. A Figura 29 apresenta a descrição do modelo da placa de rede que foi instalada para captura de dados e a Figura 30 o endereço IPv4 configurado na plataforma computacional. Após a configuração da interface de rede do computador foi possível acessar os dispositivos para as configurações iniciais através dos aplicativos utilizando endereço IPv4.

Figura 29 - Descrição do modelo da placa de rede instalada



Fonte: O autor

Figura 30 - Configuração do endereço IPv4 e máscara de rede



Fonte: O autor

1.18.2. Configuração do switch

Foi utilizado, no ambiente de teste, um switch gerenciável do fabricante Siemens. O equipamento modelo RX-1501 é homologado para norma IEC 61850 e da *North American Electric Reliability Corporation* (NERC) e *Critical Infrastructure Protection* (CIP).

O switch RX-1501 possibilita a configuração de VLANs nas interfaces de comunicação. Para os ensaios realizados foi necessário apenas uma VLAN.

Para acesso ao gerenciamento do switch o endereço IP 192.168.10.2 foi configurado. Após a configuração de gerência, foi adicionado o endereço IP do NTP *Server* para sincronismo do relógio interno.

1.18.3. Configuração do NTP Server

Com o objetivo de garantir o sincronismo de tempo dos dispositivos instalados na rede um NTP Server foi configurado. A Figura 31 apresenta o resultado dos comandos *show plataform gps time e show ntp status.* O primeiro comando apresenta

o resultado do horário e data recebidos pelo sinal do GPS do equipamento, já o segundo apresenta o *status* da função de servidor de tempo do equipamento.

Figura 31 - Resultado da configuração do NTP Server

```
LabRouter#show platform gps time
21:4:38.436 UTC Sat May 23 2020

LabRouter#show ntp status
Clock is synchronized, stratum 1, reference is .LOCL.
nominal freq is 500.0000 Hz, actual freq is 500.0000 Hz, precision
is 2**14
ntp uptime is 29677200 (1/100 of seconds), resolution is 2000
reference time is E2740E64.C946278D (21:04:36.786 UTC Sat May 23
2020)
clock offset is 0.0000 msec, root delay is 0.00 msec
root dispersion is 0.35 msec, peer dispersion is 0.29 msec
loopfilter state is 'CTRL' (Normal Controlled Loop), drift is
0.000000000 s/s
system poll interval is 16, last update was 1 sec ago.
```

Fonte: O autor

1.18.4. Configuração dos IEDs

Os IED 451-5-R314 e 487E-3-R310 da SEL possuem mais de uma porta de comunicação *ethernet.* Estes dispositivos apresentam uma característica no funcionamento das suas portas de comunicação, a qual permite que elas possam trabalhar de quatro formas diferentes, utilizando apenas um endereço IP:

a) ***fixed:*** apenas uma das duas portas fica ativa no IED, mantendo a outra desativada;

b) ***failover:*** trabalha com as duas portas, sendo uma definida como primária. Somente haverá comutação para a porta secundária se houver falha na primária e depois de decorrido o tempo de ajuste;

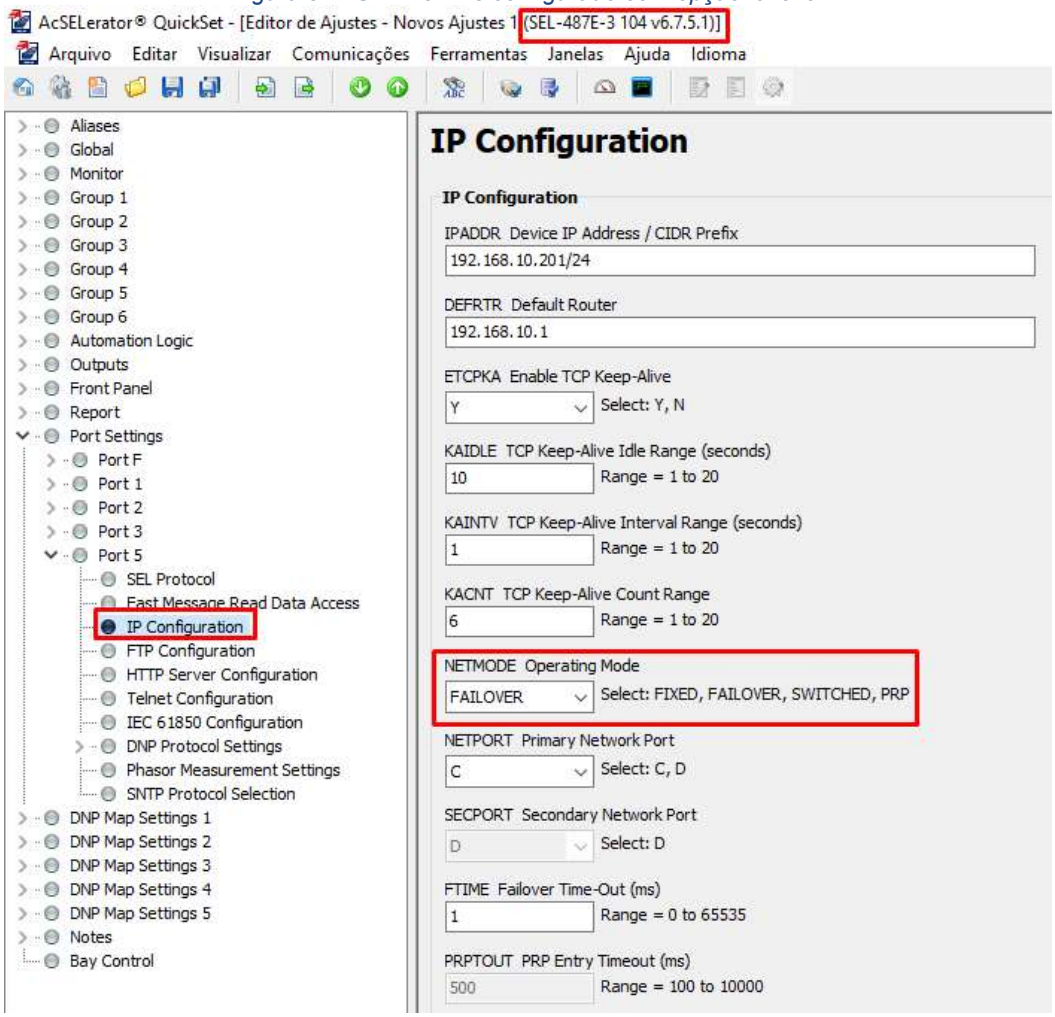
c) ***switched:*** trabalha com as duas portas ativas mantendo uma configuração de *switch* para implementação com rede em anel;

d) *Parallel Redundancy Protocol (**PRP**):* é um padrão de protocolo de rede *ethernet* que fornece *failover* contínuo contra falhas de qualquer componente de rede. A informação é publicada de forma duplicada já que o dispositivo transmite o mesmo dado através das duas portas de rede. Nesta configuração o dispositivo aceita o primeiro pacote recebido e descarta o segundo (CISCO, 2018).

Para os testes apresentados neste capítulo os IEDs foram configurados com a opção *failover*, configuração default dos IEDs da SEL. Para os testes aqui

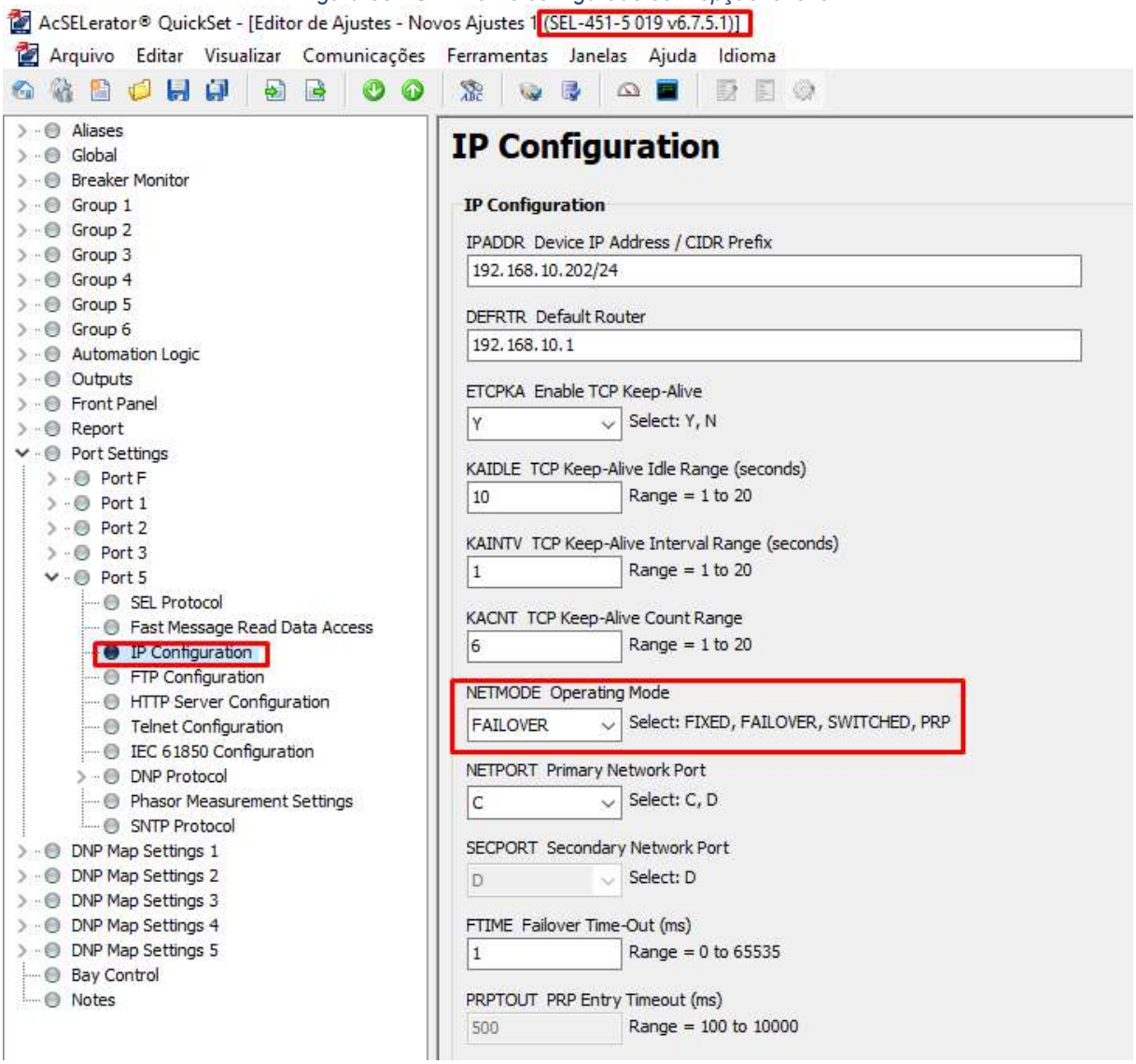
apresentados apenas uma das portas de comunicação *ethernet* de cada IEDs foi conectada ao switch do laboratório. As configurações podem ser observadas nas Figura 32 e Figura 33.

Figura 32 - SEL-487E-3 configurado com opção failover



Fonte: O autor

Figura 33 - SEL-451-5 configurado com opção failover



Fonte: O autor

Os IEDs da SEL utilizados durante os testes apresentaram uma característica similar na publicação dos pacotes. Como já mencionado neste capítulo os dispositivos possuem duas portas de comunicação *Ethernet*. Cada uma das portas possui um endereço físico de MAC, que pode ser identificados através da execução do comando *mac* no terminal de interação com o IED. As Figura 34 e Figura 35 apresentam o resultado do comando *mac* executado no IED SEL-487E-3 e SEL-451-5 respectivamente.

Para o IED SEL-487E-3 o endereço de MAC da porta 5-1 é 00:30:A7:08:65:37 e para a porta 5-2 o endereço de MAC é 00:30:A7:08:65:38, para o IED SEL-451-5 o endereço de MAC da porta 5-1 é 00:30:A7:08:65:37 e para a porta 5-2 o endereço de MAC é 00:30:A7:08:65:38. :

Figura 34 - Resultado do comando mac para IED SEL-487E-3

```
=>mac

Port 5-1 MAC Address:  00-30-A7-08-65-37
Port 5-2 MAC Address:  00-30-A7-08-65-38
```

Fonte: O autor

Figura 35 - Resultado do comando mac para o IED SEL-451-5

```
=>mac

Port 5-1 MAC Address:  00-30-A7-04-E7-3D
Port 5-2 MAC Address:  00-30-A7-04-E7-3E
```

Fonte: O autor

1.19. Definição das funções lógicas

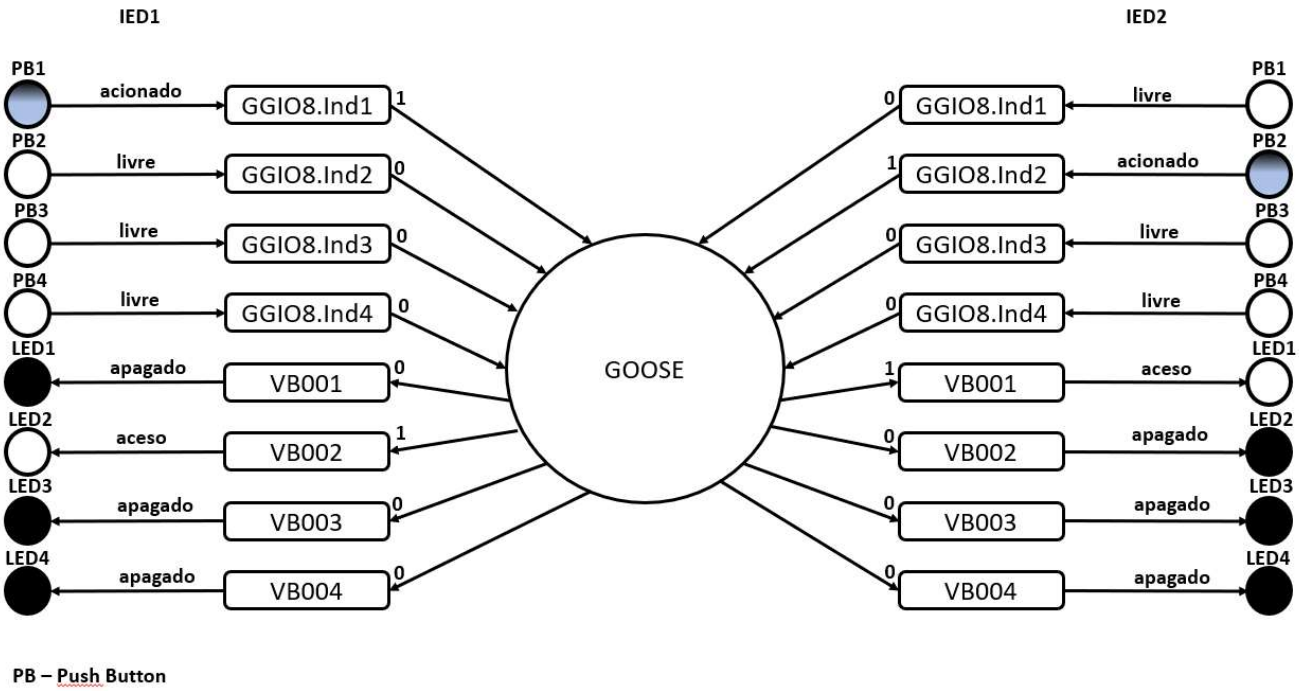
Com o objetivo de realizar a troca de mensagens GOOSE entres IEDs e capturá-las através da interface de rede *Ethernet* do PC, foram utilizados dois relés de proteção e, neste caso, as informações de *status* de *push-button* foram definidas para publicação.

Nas aplicações reais, as mensagens de *trips e* falhas de disjuntor, possuem uma maior importância se comparadas ao *status* do *push-button*, porém, do ponto de vista da construção e envio dos pacotes têm comportamentos semelhantes.

Primeiro um projeto lógico foi elaborado. Na lógica definida, um *led* do IED assinante acenderá quando um botão no relé publicador for pressionado. O *led* aceso tem o mesmo valor de sequência do *push-button* acionado.

O diagrama de blocos elaborado com as lógicas desenvolvidas é apresentado na Figura 36.

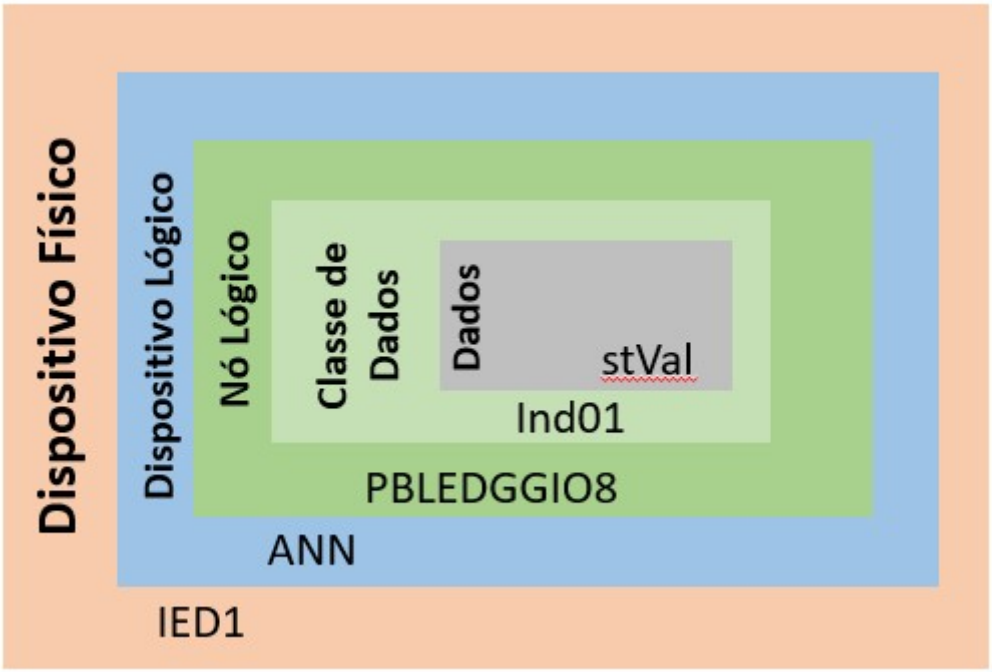
Figura 36 - Diagrama de bloco lógico



Fonte: O autor

Detalhando, pode-se citar o caminho para encontrar a variável binária utilizada. Acessando-se o dispositivo físico (IED1) através do seu endereço de rede, temos acesso em princípio ao dispositivo lógico. Este, uma vez acessado, disponibiliza a localização de vários tipos de dispositivos lógicos, que podem ser proteções, comandos, retornos digitais, configurações do próprio IED e o que nos interessa nesse exemplo, que são os *tags* digitais de status, localizadas no diretório “ANN“. Em seu interior estão os nós lógicos. O nó a ser utilizado será o PBLEDGGIO08, padronizado pela SEL para indicar os *status* dos *push-button*. Explorando o nó é possível visualizar cada uma das classes que o compoem. Na Figura 37, é possível perceber de forma mais clara como essa arquitetura é ilustrada.

Figura 37 - Estrutura de dados IEC61850



Fonte: O autor

Após identificar os atributos na estrutura, os botões frontais dos IEDs foram mapeados para os endereços lógicos. A Tabela 7 apresenta os botões frontais utilizados e os respectivos endereços dos atributos.

Tabela 7 - Botão e Atributos Lógicos

Botão	Atributo Lógico
1	IED1.ANN.PBLEDGGIO8.Ind01.stVal
2	IED1.ANN.PBLEDGGIO8.Ind02.stVal
3	IED1.ANN.PBLEDGGIO8.Ind03.stVal
4	IED1.ANN.PBLEDGGIO8.Ind04.stVal

Fonte: O autor

Em seguida, foram mapeados os atributos recebidos para as variáveis virtuais de entrada, então as os *leds* frontais puderam ser configurados. Para os IEDs da SEL, as variáveis virtuais de entrada são identificadas pelas iniciais VB seguidas de uma sequência numérica. A Tabela 8 apresenta o mapeamento realizado para sinalizar através dos *leds* frontais o *status* do *push-button.*

Tabela 8 - Relação Atributos e Leds

Atributo Recebido	Entrada Virtual	LED Frontal
IED2.ANN.PBLEDGGIO8.Ind01.stVal	VB001	1
IED2.ANN.PBLEDGGIO8.Ind02.stVal	VB002	2
IED2.ANN.PBLEDGGIO8.Ind03.stVal	VB003	3
IED2.ANN.PBLEDGGIO8.Ind04.stVal	VB004	4

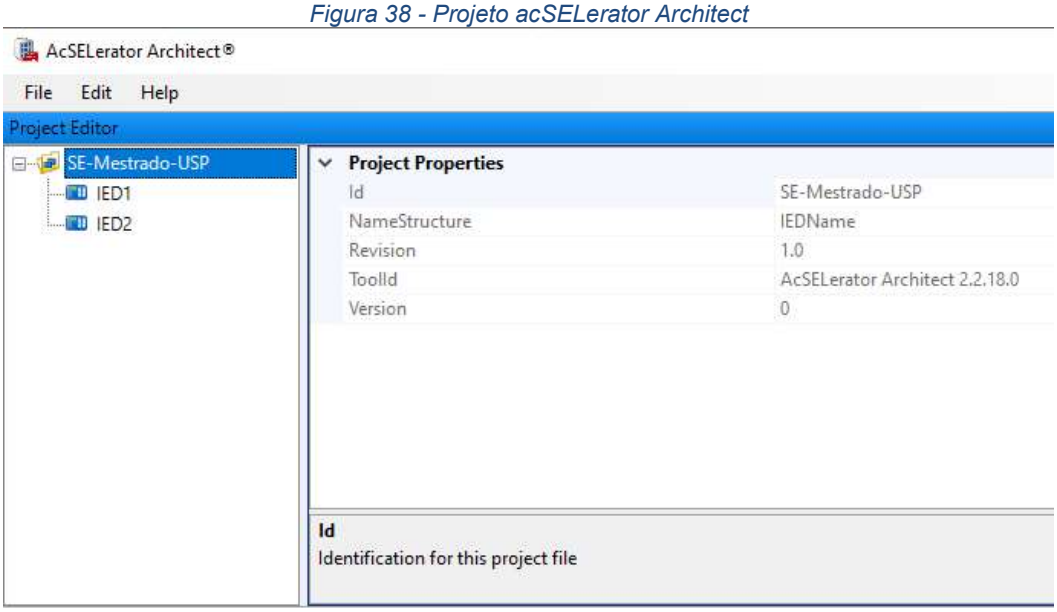
Fonte: O autor

1.20. Configuração da publicação GOOSE

Para um IED publicar mensagens GOOSE, é necessária uma sequência de configuração a ser seguida.

Primeiro foi criado um projeto no *software* acSELerator Architect, identificado como SE-USP-Mestrado. Posteriormente, os dispositivos utilizados foram importados para estrutura criada. A Figura 38 apresenta a estrutura criada e os dispositivos inseridos.

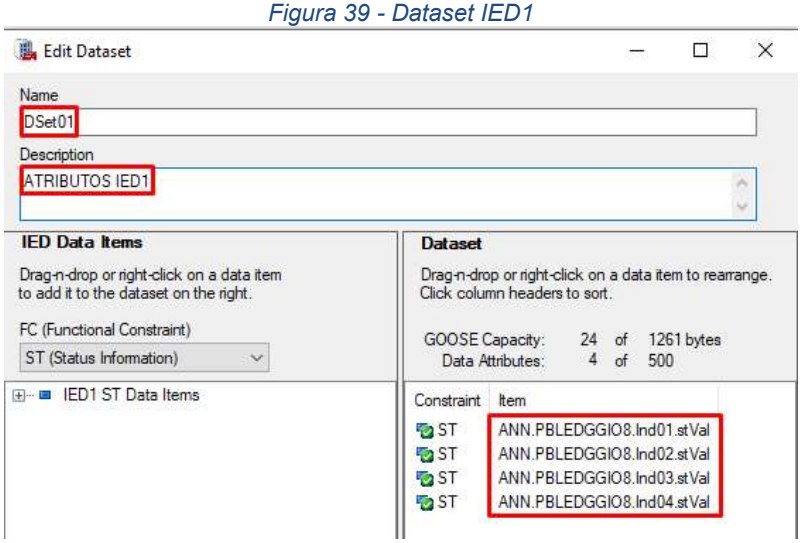
Figura 38 - Projeto acSELerator Architect



Fonte: O autor

Com o IED1 selecionado os atributos referentes ao st*atus* de *push-button do* dispositivo foram inseridos no *dataset* com o name DSet01. A Figura 39 apresenta o *dataset* criado e os atributos definidos.

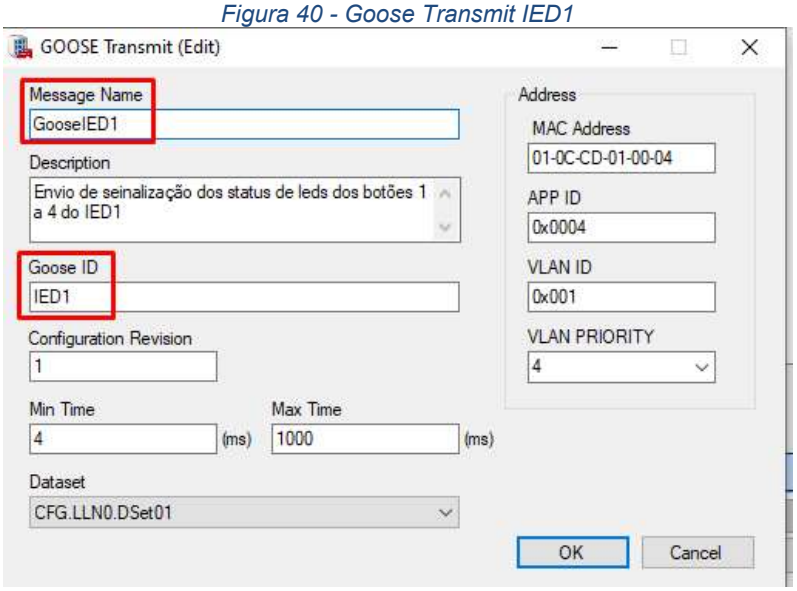
Figura 39 - Dataset IED1



Fonte: O autor

Após o *dataset* ser estruturado, a mensagem GOOSE é configurada. Acessando o guia de menu GOOSE *Transmit* do *software* acSELerator Architect são preenchidos os campos que identificam as mensagens criadas. A Figura 40 apresenta os parâmetros de identificação configurados para o IED1.

Figura 40 - Goose Transmit IED1



Fonte: O autor

A descrição dos campos contidos na Figura 40 é apresentada a seguir:

a) ***message name:*** identificador único contendo o nome da mensagem publicada;

b) ***description:*** campo destinado para descrição de comentários;

c) **GOOSE ID:** identificador do dispositivo publicador GOOSE. Este campo é obrigatório;

d) ***configuration revision:*** conforme IEC 61850-7-3, este campo identifica a instancia de configuração do dispositivo lógico.

e) ***min*** e ***max. time(ms)****:* valor numérico de quatro a sessenta mil que representa um período de tempo em milissegundos. Este período é o intervalo entre as mensagens GOOSE após o decaimento exponencial e onde não há mudanças no GOOSE.

f) ***dataset:*** agrupamento de nós lógicos;

g) ***MAC address:*** um único endereço MAC para o qual múltiplos dispositivos podem se inscrever para a entrega simultânea de um fluxo de dados comum;

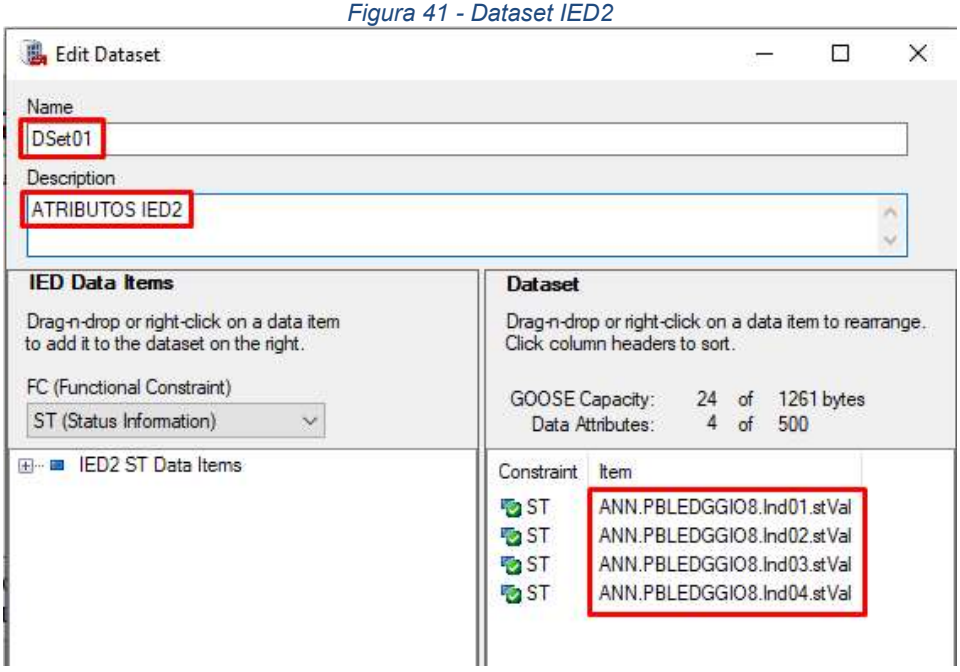
h) **APP ID:** identificador de aplicação. O usuário define este identificador na definição de uma mensagem GOOSE de saída conforme IEC 61850-8-1;

i) **VLAN ID:** identificador numérico de doze *bits* que identifica uma rede logicamente independente;

j) **VLAN *priority:*** valor numérico de zero a sete em que sete é a maior prioridade. Pacotes *Ethernet* com *tags* de prioridade são priorizados pelos *switches* e outros dispositivos de rede de tal forma que os pacotes de maior prioridade são processados antes dos pacotes de menor prioridade.

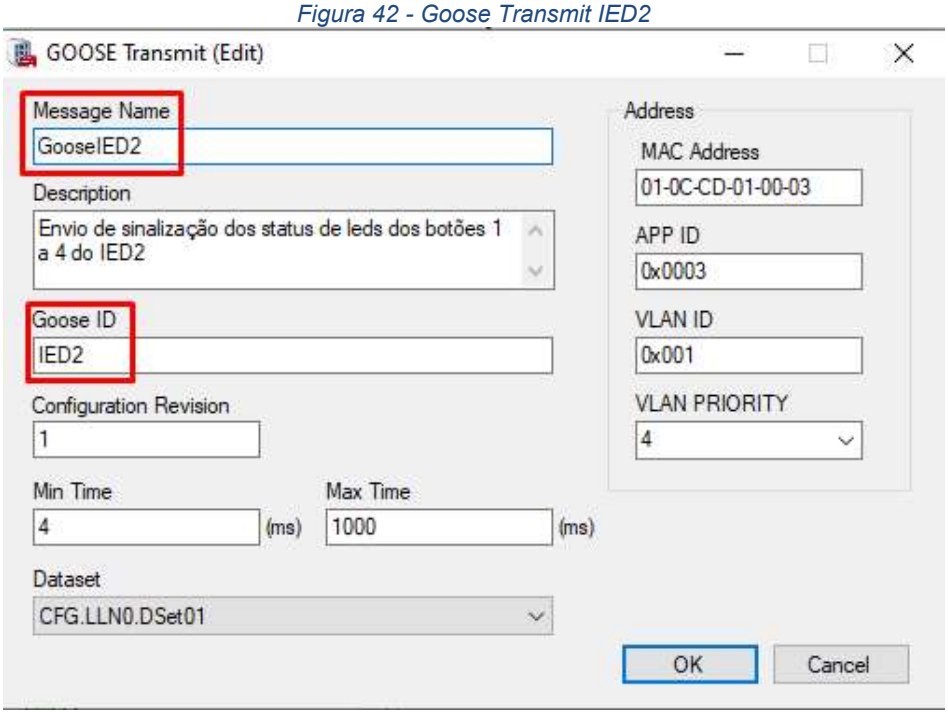
A mesma sequência utilizada para configurar a mensagem GOOSE no IED1 repetiu-se para o IED2. As Figura 41 e Figura 42 evidenciam a configuração.

Figura 41 - Dataset IED2



Fonte: O autor

Figura 42 - Goose Transmit IED2



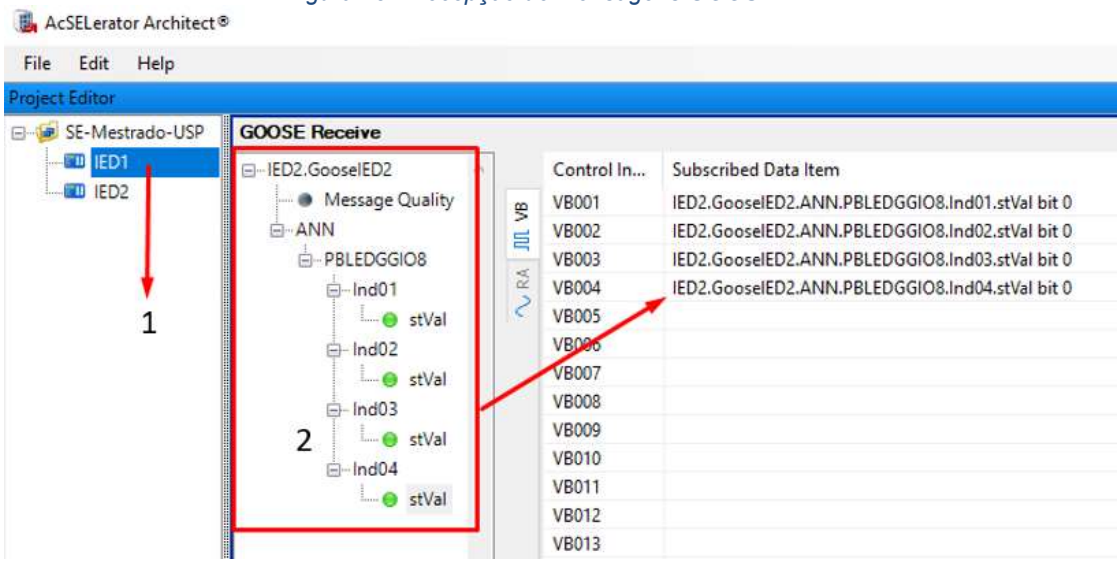
Fonte: O autor

1.21. Configuração da assinatura das mensagens GOOSE

Após a configuração da transmissão do GOOSE em ambos os dispositivos, a recepção da mensagem pode ser definida. Os atributos recebidos foram direcionados para as variáveis virtuais de entrada indicada no item 1.19 deste mesmo capítulo.

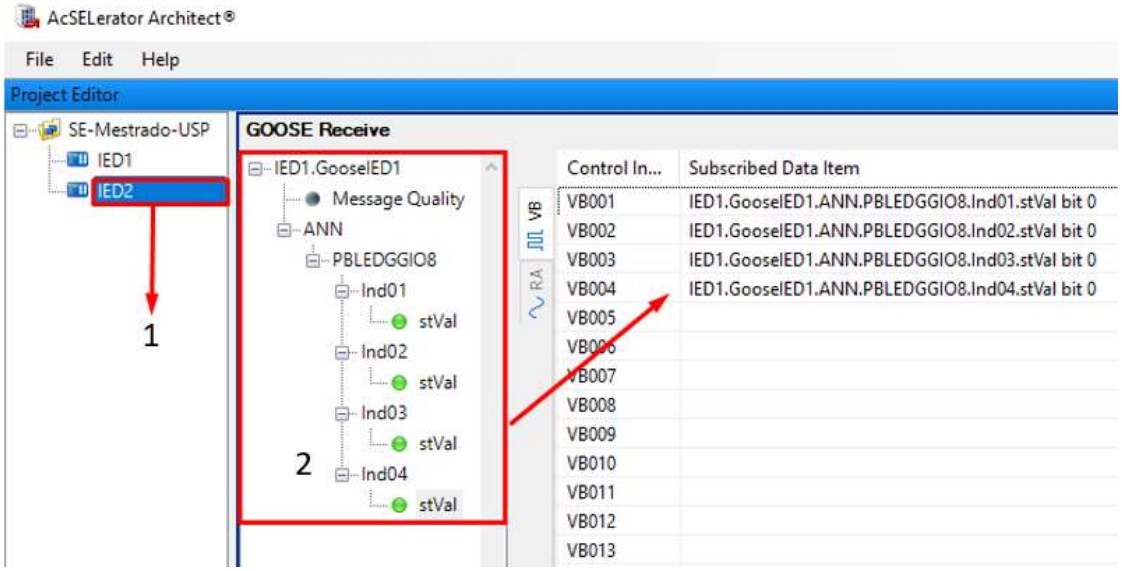
As Figura 43 e Figura 44 evidenciam as configurações realizadas nos IEDs para recepção das mensagens GOOSE.

Figura 43 - Recepção de mensagens GOOSE IED1



Fonte: O autor

Figura 44 - Recepção de mensagens GOOS IED2



Fonte: O autor

Após as configurações dos parâmetros, os arquivos CID foram descarregados nos IEDs, então o GOOSE passou a ser publicado na rede.

Ensaios e resultados

Este capítulo tem por objetivo demonstrar que, conforme metodologia desenvolvida e, analisando as mensagens GOOSE capturadas pelo AAG, é possível identificar divergências da configuração atual dos IEDs com a definida no projeto inicial.

Além de identificar divergências de configuração, os resultados obtidos evidenciam que o software desenvolvido é capaz de indicar falhas de comunicação e de sincronismo através da interpretação da estampa de tempo das mensagens GOOSE trafegadas.

1.22. Identificação de arquivos e interface de rede

Como teste inicial, foi realizada a validação da interpretação dos arquivos SCD e ATG pelo AAG, somado a estes a captura do pacote GOOSE.

Primeiro o AAG localizou o arquivo SCD e internamente identificou os parâmetros: *Message name,* IED ID, *dataset* e a quantidade de atributos do *dataset* especificado pelo usuário. A Figura 45 apresenta o resultado da identificação das variáveis realizada pelo *software*.

Figura 45 - Identificação de IED e Dataset

```
c:\\Nucleo\\IEC61850\\SCD\\SCD_2_IEDs.SCD
        <GSE ldInst="CFG" cbName="GooseIED2">
        <GSE ldInst="CFG" cbName="GooseIED1">
IED NAME = IED2
DSet01 - <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind01" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind02" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind03" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind04" daName="stVal" fc="ST" />
IED NAME = IED1
DSet01 - <FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind01" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind02" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind03" daName="stVal" fc="ST" />
DSet01 -<FCDA ldInst="ANN" prefix="PBLED" lnClass="GGIO" lnInst="8"
doName="Ind04" daName="stVal" fc="ST" />
```

Fonte: O autor

Após a interpretação e identificação das variáveis no arquivo SCD, o arquivo ATG foi localizado e seu conteúdo, que são os dados preenchido pelo usuário, foram

carregados para a memória do *software* auditor. O resultado da interpretação do arquivo ATG é mostrado na Figura 46.

Figura 46 - Identificação dos atributos no arquivo ATG

```
IED2;DSet01;1;1;Traducao Variavel 1 SEL451;Status;00=PB1 is not pushed,01=PB1
is pushed
IED2;DSet01;1;2;Traducao Variavel 2 SEL451;Status;00=PB2 is not pushed,01=PB2
is pushed
IED2;DSet01;1;3;Traducao Variavel 3 SEL451;Status;00=PB3 is no pushed,01=PB3
is pushed
IED2;DSet01;1;4;Traducao Variavel 4 SEL451;Status;00=PB4 is not pushed,01=PB4
is pushed
IED1;DSet01;1;1;Traducao Variavel 1 SEL487;Status;00= PB1 is not pushed,01=PB1
is pushed
IED1;DSet01;1;2;Traducao Variavel 2 SEL487;Status;00= PB2 is not pushed,01=PB
2 is pushed
IED1;DSet01;1;3;Traducao Variavel 3 SEL487;Status;00= PB3 is no pushed,01=PB3
is pushed
IED1;DSet01;1;4;Traducao Variavel 4 SEL487; Status;00= PB4 is not pushed,01=PB4
is pushed
```

Fonte: O autor

A terceira etapa do processo de inicialização da aplicação é identificar as interfaces de rede instaladas na plataforma computacional. O AAG listou nove interfaces existentes no PC, então a *interface* **3 *"Network adapter 'Realtek PCIe Gbe Family Controller' on local host"***, interface que está conectada na rede do ambiente de teste, foi selecionada para captura dos pacotes GOOSE trafegados. A Figura 47 apresenta o resultado da identificação das *interfaces* de redes instaladas na plataforma computacional e detectadas pelo soft*ware* auditor.

Figura 47 - Lista de interfaces de rede

```
1 - Network adapter 'Microsoft' on local host
2 - Network adapter 'Microsoft' on local host
3 - Network adapter 'Realtek PCIe GbE Family Controller' on local
host
4 - Network adapter 'Microsoft' on local host
5 - Network adapter 'Microsoft' on local host
6 - Network adapter 'VMware Virtual Ethernet Adapter' on local host
7 - Network adapter 'VMware Virtual Ethernet Adapter' on local host
8 - Network adapter 'Microsoft' on local host
9 - Network adapter 'MS NDIS 6.0 LoopBack Driver' on local host

System.String[]
    Count:    1
Enter the interface number (1-9):
3
```

Fonte: O autor

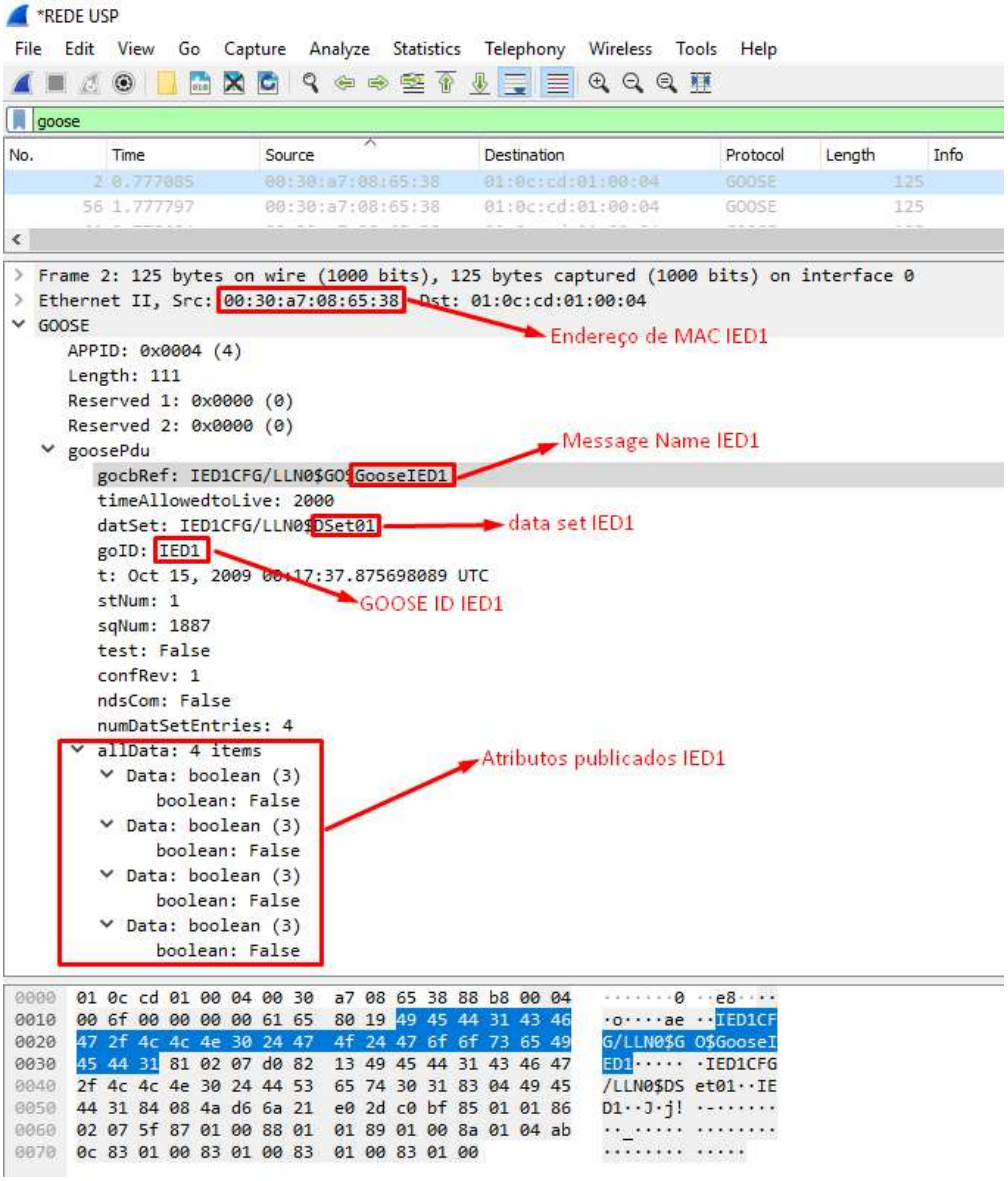
1.23. Validação da Captura do Pacote pelo AAG

Como foi mencionado no item 1.21, depois de descarregadas as configurações nos dispositivos, o GOOSE passou a ser publicado na rede.

Para o teste de validação da publicação das mensagens GOOSE dos IEDs e captura dos pacotes pela plataforma computacional, os push-buttons dos dispositivos não foram pressionados, assim o resultado esperado deve ser todos os atributos com valor zero “0”.

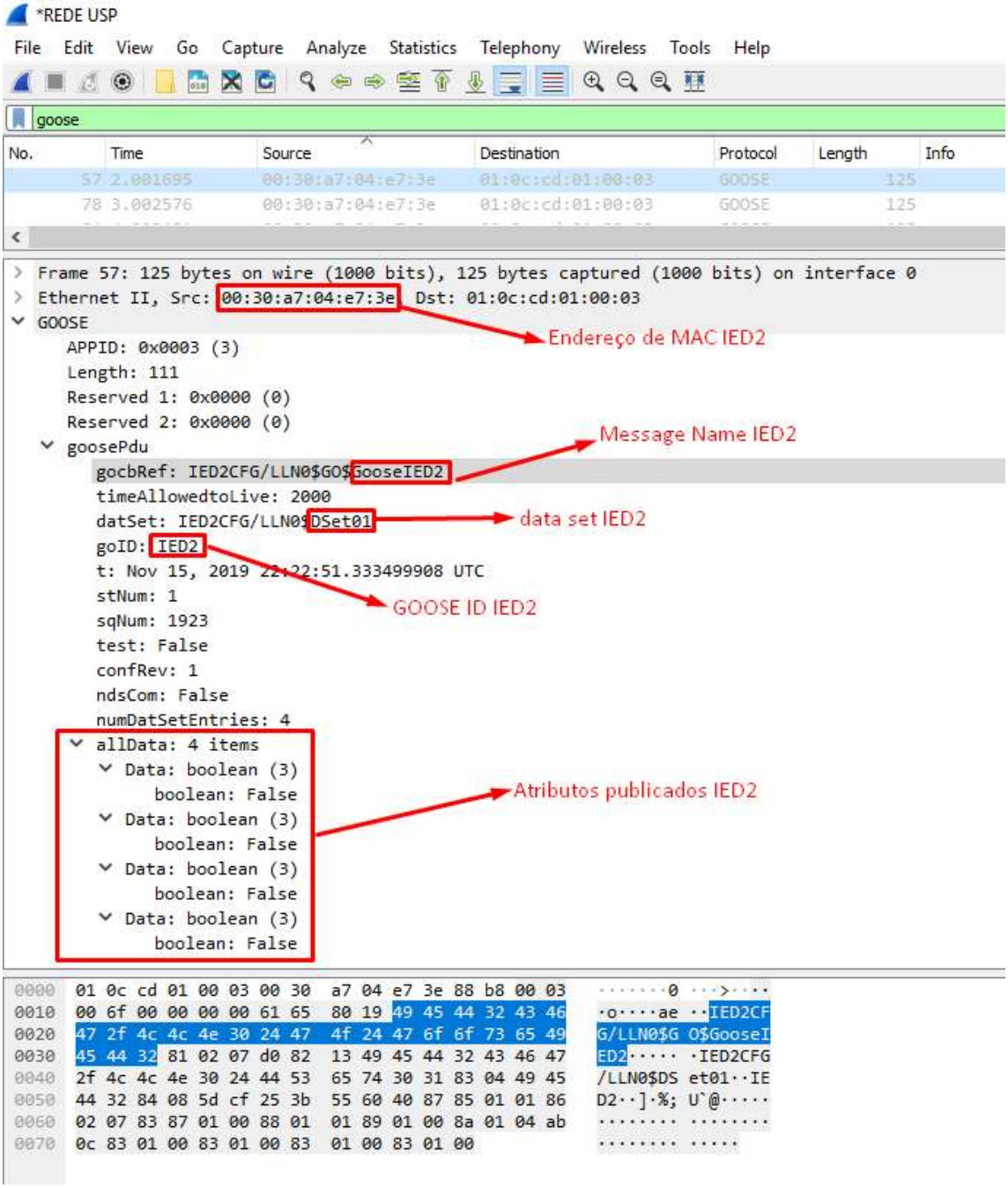
Para evidenciar o tráfego de mensagens GOOSE utilizou-se o programa *Wireshark.* Com auxílio do *software* foi possível verificar a publicação do GOOSE do IED1 (endereço 00:30:a7:08:65:38) e IED2 (endereço 00:30:a7:04:e7:3e), os pacotes capturados contém as informações de *Message Name, data set, GOOSE ID* e os atributos configurados com o estado *false* das saídas PBLEDGGIO8 quando os botões frontais dos IEDs não estão pressionados, conforme Figura 48 e Figura 49.

Figura 48 - GOOSE do IED1, botões não pressionados



Fonte: O autor

Figura 49 - GOOSE do IED2, botões não pressionados



Fonte: O autor

Com o auxílio do AAG, foi possível também, verificar o envio do GOOSE do IED1 (endereço 00:30:a7:08:65:38) e IED2 (endereço 00:30:a7:04:e7:3e). Foram identificadas as informações de *Message Name, data set, GOOSE ID* e os atributos configurados com o estado *false* das saídas PBLEDGGIO8 quando os botões frontais dos IEDs não estão pressionados. As Figura 50 e Figura 51 evidenciam o resultado da captura dos pacotes através do AAG.

Figura 50 - GOOSE do IED1 capturado pelo AAG

```
2019;11;16;04;52;44;952- 010ccd0100040030a708653888b80004006e0000000
06164801949454431434647 2f4c4c4e3024474f24476f6f73654945443181020 7d0821
3494544314346472f4c4c4e3024445365743031830449454431840 84ad6c7688bac40
bf85015786015a870100880101890 1008a0104ab0c830100830100830100830100

MAC ORIGEM = 0030a7086538
MAC DESTINO = 010ccd010004

PROTOCOLO = 88b8 GOOSE
APPID = 0004
GOOSE PACKET LENGTH = 006e GOOSE PDU LENGTH DEC = 110 byte
GOOSE RESERVED 1 = 0000
GOOSE RESERVED 2 = 0000
GOOSE PDU = 61
GOOSE LENGTH = 64
GOOSE LENGTH = 64 GOOSE PDU LENGTH DEC = 100 byte
CONTROL REFERENCE BLOCK = 80
CONTROL REFERENCE BLOCK DATA = IED1CFG/LLN0$GO$GooseIED1
TIME ALLOWED TO LIVE = 81
TIME ALLOWED TO LIVE LENGTH = 02 TIME ALLOWED TO LIVE LENGTH DEC = 2
 byte
TIME ALLOWED TO LIVE DATA = 07d0
DATA SET = 82
DATA SET LENGTH = 13 DATA SET LENGTH DEC = 19 byte
DATA SET DATA = IED1CFG/LLN0$DSet01
GOOSE ID = 83
GOOSE ID LENGTH = 04 DATA SET LENGTH DEC = 4 byte
GOOSE ID DATA = 49454431
GOOSE ID DATA = IED1
GOOSE TIME STAMP = 84
GOOSE TIME STAMP LENGTH = 08 GOOSE TIME STAMP LENGTH DEC = 8 byte
GOOSE TIME STAMP DATA = 4ad6c7688bac40bf
GOOSE TIME STAMP DATA = 1255589736
GOOSE TIME STAMP MILLISECONDS = 0,5455971
GOOSE STATE NUMBER = 85
GOOSE STATE NUMBER LENGTH = 01 GOOSE STATE NUMBER LENGTH DEC = 1 byte
GOOSE STATE NUMBER DATA = 57
GOOSE SEQUENCE NUMBER = 86
GOOSE SEQUENCE NUMBER LENGTH = 01 GOOSE SEQUENCE NUMBER LENGTH DEC = 1
 byte
GOOSE SEQUENCE NUMBER DATA = 5a
GOOSE IED STATE = 87
GOOSE IED STATE LENGTH = 01 GOOSE IED STATE LENGTH DEC = 1 byte
GOOSE IED STATE DATA = 00
GOOSE CONFIG REVISION = 88
GOOSE CONFIG REVISION LENGTH = 01 GOOSE CONFIG REVISION LENGTH DEC = 1
 byte
GOOSE CONFIG REVISION DATA = 01
GOOSE NEED REVISION = 89
GOOSE NEED REVISION LENGTH = 01 GOOSE NEED REVISION LENGTH DEC = 1 byte
GOOSE NEED REVISION DATA = 00
GOOSE DATA ENTRIES = 8a
GOOSE DATA ENTRIES LENGTH = 01 GOOSE DATA ENTRIES LENGTH DEC = 1 byte
GOOSE DATA ENTRIES DATA = 04
GOOSE DATA = ab
```

```
GOOSE DATA LENGTH = 0c GOOSE DATA LENGTH DEC = 12 byte
TAMANHO DO PARAMETRO= 24
VARIAVEL = 1
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
ITEM TAMANHO = 01 ITEM TAMANHO DEC = 1 byte
IED1;DSet01;1;00
VARIAVEL = 2
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
ITEM TAMANHO = 01 ITEM TAMANHO DEC = 1 byte
IED1;DSet01;2;00
VARIAVEL = 3
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
ITEM TAMANHO = 01 ITEM TAMANHO DEC = 1 byte
IED1;DSet01;3;00
VARIAVEL = 4
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
ITEM TAMANHO = 01 ITEM TAMANHO DEC = 1 byte
IED1;DSet01;4;00
```

Fonte: O autor

Figura 51 - GOOSE do IED2 capturado pelo AAG

```
2019;11;16;05;11;03;347                                                    -
010ccd0100030030a704e73e88b80003006f00000000616580194945443243464 7
2f4c4c4e3024474f24476f6f736549454432810207d0821349454432434647 2f4c
4c4e3024445365743031830449454432840 85dcf7fc3b339c08785010b860209f3
8701008801018901008a0104ab0c830100830100830100830100
MAC ORIGEM = 0030a704e73e
MAC DESTINO = 010ccd010003
PROTOCOLO = 88b8 GOOSE
APPID = 0003
GOOSE PACKET LENGTH = 006f GOOSE PDU LENGTH DEC = 111 byte
GOOSE RESERVED 1 = 0000
GOOSE RESERVED 2 = 0000
GOOSE PDU = 61
GOOSE LENGTH = 65
CONTROL REFERENCE BLOCK = 80
CONTROL REFERENCE BLOCK DATA = IED2CFG/LLN0$GO$GooseIED2
TIME ALLOWED TO LIVE = 81
TIME ALLOWED TO LIVE DATA = 07d0
DATA SET = 82
DATA SET LENGTH = 13 DATA SET LENGTH DEC = 19 byte
DATA SET DATA = IED2CFG/LLN0$DSet01
```

```
GOOSE ID = 83
GOOSE ID DATA = 49454432
GOOSE ID DATA = IED2
GOOSE TIME STAMP = 84
GOOSE TIME STAMP DATA = 5dcf7fc3b339c087
GOOSE TIME STAMP DATA = 1573879747
GOOSE TIME STAMP MILLISECONDS = 0,7000999
GOOSE STATE NUMBER = 85
GOOSE STATE NUMBER DATA = 0b
GOOSE SEQUENCE NUMBER = 86
GOOSE SEQUENCE NUMBER DATA = 09f3
GOOSE SEQUENCE NUMBER DATA = 2547
GOOSE IED STATE = 87
GOOSE IED STATE DATA = 00
GOOSE CONFIG REVISION = 88
GOOSE CONFIG REVISION DATA = 01
GOOSE NEED REVISION = 89
GOOSE NEED REVISION DATA = 00
GOOSE DATA ENTRIES = 8a
GOOSE DATA ENTRIES DATA = 04
GOOSE DATA = ab
VARIAVEL = 1
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
IED2;DSet01;1;00
VARIAVEL = 2
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
IED2;DSet01;2;00
VARIAVEL = 3
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
IED2;DSet01;3;00
VARIAVEL = 4
ITEM = 83
TAMANHO ITEM = 01
TIPO DE DADO = 83
TAMANHO ITEM = 01
IED2;DSet01;4;00
```

Fonte: O autor

Comprovou-se que as mensagens GOOSE publicadas pelos IEDs na camada de enlace de dados do modelo OSI, podem ser capturadas e identificadas por suas particularidades de Endereço de MAC, *Message Name, DataSet, GOOSE ID* e quantidade de atributos.

1.24. Validação da Quantidade de Atributos

As Figura 52 e Figura 53 evidenciam a análise do AAG para quantidade de dados publicados pelos IEDs. A informação (***Não encontrado divergência na quantidade de atributos para o IED)*** apresentada pelo *software* indica que a mensagem capturada contém o número exato de atributos definidos nos arquivos SCD e ATG para o IED publicador da mensagem.

Figura 52 - Não divergência na quantidade de atributos para o IED1

```
IED IED1 existe na tabela de IEDs.
Não foram detectadas divergências na quantidade de atributos para IED1
e DSet01
Não foram detectadas divergências na quantidade de atributos ente ATG
e Mensagem Goose
```

Fonte: O autor

Figura 53 - Não divergência na quantidade de atributos para o IED2

```
IED IED2 existe na tabela de IEDs.
Não foram detectadas divergências na quantidade de atributos para IED2
e DSet01
Não foram detectadas divergências na quantidade de atributos ente ATG
e Mensagem Goose
```

Fonte: O autor

1.25. Tradução dos Atributos GOOSE Capturados

Para evidenciar a funcionalidade do AAG para interpretação dos atributos recebidos, capturas com o *push-button* não pressionado e pressionado foram realizadas para o IED1.

Para o teste de interpretação no momento que o *botão* não estava pressionado um pacote GOOSE foi capturado. O software auditor interpretou os atributos contidos na mensagem e traduziu a informação conforme indicada no arquivo ATG. O resultado da tradução dos atributos é apresentado na Figura 54.

Figura 54 - Tradução dos atributos pelo software auditor

```
IED IED1 ---> Traducao Push-Button 1 = PB1 IED1 is not pushed Status
IED IED1 ---> Traducao Push-Button 2 = PB2 IED1 is not pushed Status
IED IED1 ---> Traducao Push-Button 3 = PB3 IED1 is not pushed Status
IED IED1 ---> Traducao Push-Button 4 = PB4 IED1 is not pushed Status
```

Fonte: O autor

Em uma outra simulação, no momento que os botões 2 e 4 do IED1 estavam pressionados um pacote GOOSE foi capturado. O software auditor interpretou os

atributos e indicou que os botões estavam pressionados. O resultado da tradução dos atributos é apresentado na Figura 55.

Figura 55 - Tradução dos atributos 2 e 4 pressionados

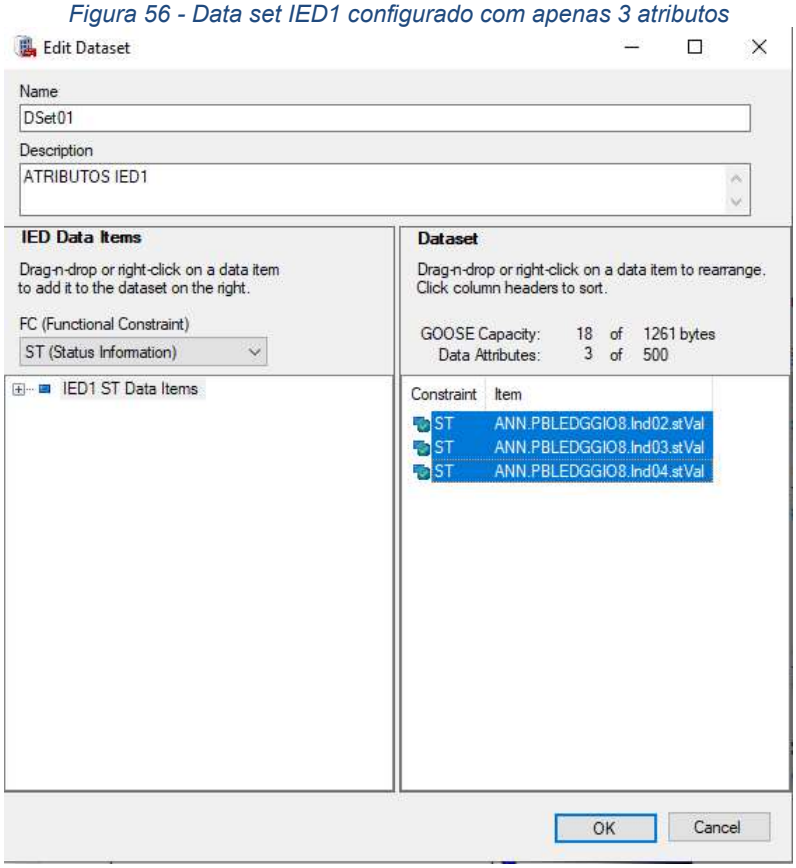
```
IED IED1 ---> Traducao Push-Button 1 = PB1 IED1 is not pushed Status
IED IED1 ---> Traducao Push-Button 2 = PB2 IED1 is pushed Status
IED IED1 ---> Traducao Push-Button 3 = PB3 IED1 is not pushed Status
IED IED1 ---> Traducao Push-Button 4 = PB4 IED1 is pushed Status
```

1.26. Identificação de Alteração da Configuração GOOSE

O objetivo deste ensaio é evidenciar, através do *software* auditor, a identificação da divergência da quantidade de atributos publicado pelo IED e o indicado no arquivo ATG.

Com o auxílio do *software acSELerator Architect* foi excluído a publicação do atributo PBLEDGGIO8.Ind01.stVal referente ao *status* do *push-button* 1 do IED1, então a nova parametrização foi descarregada no dispositivo. A Figura 56 apresenta a tela do aplicativo com apenas três atributos configurados.

Figura 56 - Data set IED1 configurado com apenas 3 atributos



Fonte: O autor

Após a parametrização do IED1, a mensagem GOOSE configurada com um atributo a menos foi publicada e capturada pela *interface* de rede do PC. Após análise do AAG foi identificada a divergência da quantidade de atributos. Como resultado, o *software* auditor apresentou um alerta com a informação da discordância encontrada. Evidências da detecção da diferença da quantidade de atributos é apresentada na Figura 57.

Figura 57 - Mensagem de divergência encontrada para o IED1

```
IED IED1 existe na tabela de IEDs.
Detectado divergencia no numero de variaveis para IED1 e DSet01
Variavel 3 não detectada no arquivo SCD.
Detectado divergencia na quantidade de atributos ente ATG e mensagem
Goose capturada
Quantidade de variaveis encontradas no ATG para o IED = 4
Quantidade de variaveis encontradas no DataSet Goose  = 3
```

Fonte: O autor

1.27. Detecção de falha no sincronismo

O *software* auditor é capaz de identificar falha de sincronismo do IED através da comparação da estampa de tempo da mensagem GOOSE recepcionada e a *tag* de tempo inserida no momento da captura do pacote. Esta função também pode ser utilizada para estimar e supervisionar a latência da rede.

O cálculo utilizado para identificar a diferença do sincronismo é apresentado abaixo:

Sincronismo = timestamp recebido – timestamp GOOSE

Para simular a detecção da falha de sincronismo, o horário do IED1 foi atrasado trinta e cinco minutos em relação a plataforma computacional. Após a comparação dos tempos e identificado o atraso, o AAG apresentou alarme indicando possível problema de sincronismo do IED1.

A Figura 58 apresenta o resultado da identificação da falha de sincronismo do IED1 pelo *software* auditor. O valor apresentado, < 2100 segundos, é o resultado do valor do timestamp da recepção do GOOSE subtraído do valor do timestamp da mensagem GOOSE.

Figura 58 - Identificação da falha de sincronismo

```
Tempo recepção GOOSE = 1595554334
Tempo mensagem GOOSE = 1595552234
IED IED1 falha de sincronismo < 2100 segundos
```

Fonte: O autor

1.28. Detecção da Ausência de Mensagens GOOSE

O objetivo deste ensaio é apresentar a funcionalidade do AAG para identificar falha da recepção de mensagens GOOSE, indicando a possibilidade de problemas no relé ou na infra de rede de comunicação.

Para simular ausência das mensagens GOOSE publicadas pelo IED2, foi desconectada a fibra óptica da porta do dispositivo, interrompendo a comunicação do mesmo na rede.

Como indicado no item 1.15, o software auditor foi parametrizado para identificar ausência de GOOSE após vinte segundos do não recebimento de mensagens de qualquer IED. O valor do tempo pode ser ajustado para valores menores.

A Figura 59 apresenta o resultado da identificação da ausência de mensagem GOOSE para o IED2 identificado pelo *software* de auditoria. A classe que identifica a ausência da mensagem GOOSE é apresentada no ANEXO Q.

Figura 59 - Identificação da ausência de mensagens GOOSE

```
Tempo sem receber GOOSE = 26 segundos
IED IED2 falha no envio goose devido ao tempo > 20 segundos
```

Fonte: O autor

Conclusão

O trabalho abordou e discutiu os diversos aspectos referentes a troca de mensagens GOOSE entre equipamentos eletrônicos inteligentes instalados em subestações de energia elétrica. Como a norma que deu origem a este tipo de comunicação é recente, as formas para identificar problemas na configuração dos arquivos de projetos da subestação e o correto funcionamento dos equipamentos com base nas mensagens trafegadas ainda não foram definidas.

Nesse cenário, o trabalho propôs uma solução que resultou no desenvolvimento de um aplicativo, baseado na norma IEC-61850, para auxiliar na identificação de possíveis falhas de configuração e comunicação entre equipamentos elétricos inteligentes instalados em subestações de energia elétrica. Essa solução baseia-se na implantação de uma plataforma computacional que captura e traduz as mensagens GOOSE trafegadas no barramento de processo em seguida compara os dados capturados com a configuração do arquivo SCD prevista no projeto inicial.

Através dos testes de laboratório realizados com uma plataforma computacional contendo o aplicativo desenvolvido, foram verificadas e testadas a identificação de divergências entre as mensagens capturadas e o arquivo de configuração dos equipamentos. Com os resultados apresentados, foram evidenciadas as funções do software desenvolvido para auditoria e a tradução da informação capturada para os agentes envolvidos.

A análise de defeitos em instalações baseadas na norma IEC-61850 é algo complexo e que deve ser efetuada por profissionais com conhecimento específicos. A modernização de equipamentos realizados em períodos diferentes, em uma mesma subestação, pode acarretar modificações nos projetos físicos e lógicos que muitas vezes não são previstos, pois podem ser realizadas por empresas diferentes e em momentos distintos. A fase de projeto, portanto, deve ser muito bem trabalhada e discutida, sendo proposto a utilização de um padrão de documentação dos projetos e das mensagens GOOSE para consultas futuras.

Muito tem se falado em digitalização de subestação de energia, porém há de se enfatizar que toda inovação seja ela física ou digital requer conhecimento técnico e ferramentas adequadas para operação e análise de falhas no ambiente. Neste cenário o objetivo de um aplicativo de auditoria de mensagens GOOSE é o de identificar divergência da configuração dos arquivos SCD em relação as mensagens

GOOSE trafegadas e de minimizar o tempo de análise de ocorrência com a tradução das mensagens capturadas.

O aprimoramento nas questões de monitorar as mensagens GOOSE para identificar problemas em subestações de energia, minimizar o tempo de indisponibilidade de operação e risco de acidentes, são questões que devem ser melhor exploradas através de uma futura tese de doutorado, bem como em programas de Pesquisa e Desenvolvimento (P&D).

A adoção dos padrões IEC-61850 é um processo que exige um plano estratégico, quebra do tradicionalismo e adoção de tecnologias por parte das companhias que pretendem implementá-los. Os benefícios indicados pela utilização da norma, são garantidos quando a norma é aplicada na sua totalidade.

Fica evidente que o aprendizado contínuo dos envolvidos e pesquisas em ambientes modernos e equipados são importantes para consolidar as soluções pesquisadas. Apenas o conhecimento superficial dos conceitos da norma não é o suficiente para garantir o total domínio de sua aplicação, testes exaustivos em ambientes práticos e reais devem ser realizados.

Como evidenciado neste trabalho, a viabilidade da metodologia proposta, em seus aspectos de documentar projetos lógicos e traduzir mensagens GOOSE capturadas no barramento de processos, pode ser utilizada para otimizar o processo de operação e análise de ocorrências em subestação de energia IEC-61850. A aplicação dessa solução em ambientes controlados como testes de aceitação de fabricada e testes de aceitação de campo, torna-se uma excelente oportunidade para avaliar o seu desempenho e refletir sobre as práticas aplicadas.

Referências Bibliográficas

ADAMIAK, M. et al. Design of a protection relay incorporating UCA2/MMS communcations, 24 abr. 2002.

ARAUJO, A. R. D.; CAMPELLO, S.; GUALTIERI, S. **Aplicação da norma IEC61850-8-1 nas redes de proteção do sistema elétrico**. Recife. 2011.

CISCO. Parallel Redundancy Protocol (PRP) for IE 4000, IE 4010 and IE 5000 Switches, 26 jun. 2018. Disponivel em: <https://www.cisco.com/c/en/us/td/docs/switches/lan/industrial/software/configuration/guide/b_prp_ie4k_5k.pdf>. Acesso em: 10 nov. 2019.

CLAVEL, F. et al. Integration of a new Standard: A network simulatior of IEC 61850 architectures for electrical substations. **Industry Applications Magazine**, v. 21, n. 1, p. 41-48, 2014.

COMACCIO, A. F.; SILVA, A. F.; COSTA, D. T. **Norma IEC 61850: testes de velocidade das mensagens Goose**. [S.l.]. 2017.

COMACCIO, A.; SILVA, A.; COSTA, D. **Norma IEC 61850 - testes de velocidade das mensagens goose**. [S.l.]. 2017.

DUARTE, A. B. **Fundamentos da série de normas IEC 61850 e sua aplicação nas subestações**. Curitiba. 2012.

IEC-61850-1, I. E. C. **Communication Networks and Systems for Power Utility Automation**. Geneva. 2013.

IEC-61850-7-2. **Communication networks and system for power utility automation**. [S.l.]. 2010.

IEC61850-7-4. **Basic communication structure - Compatible logical node classes and data object classes**. [S.l.]. 2010.

IEC-61850-8-1. **Communication anetowrk and system for power utility automation**. [S.l.]. 2011.

JARDINI, A. J. **Sistemas Elétrico de Potência: Automação**. São Paulo. 1997.

KIMURA, S. et al. Aplicação do IEC61850 no Mundo Real: Projeto de Modernização de 30 Subestações Elétricas, Dublin, Irlanda, 2008.

LABORATORIES, S. E. **SEL-451 Relay Protection, Atuomation, and Control System**. Pullman. 2012.

LU, X.; SUN, W.; LI, H. Design and Research /based on WinPcap Protocol Analysus System, 2010.

MME, M. D. M. E. E.; ANEEL, A. N. D. E. E. **Resolução Normativa No 864**. Brasília. 2019.

NETTO, U. C. **Determinação de um parâmetro para monitoramento do desempenho de mensagens GOOSE do padrão IEC 61850 utilizadas em subestações de energia elétrica**. São Carlos: http://teses.usp.br/teses/disponiveis/18/18154/tde-16102012-083711/pt-br.php, 2012.

RUSH, P. **Proteção e Automação de Redes**. São Paulo: Edgard Blucher Ltda, 2010. Disponivel em: <https://industriaautomatica.wordpress.com/2015/09/24/evolucao-dos-reles-de-protecao/>. Acesso em: 26 Abril 2019.

SIDHU, T. S.; KANABAR, M. G.; PARICH, P. P. Implementation Issues with IEC 61850 Based Substation Automation System, Bombay, December 2008.

SOUZA, D.; MARCOS, A. **Proposta de um sistema de monitoramento e validações de comunicação de subestações elétrica**. Campo Limpo Pta-SP. 2012.

STEINHAUSER, F. New Challenges with Substations utilizing Communication Networks, Bologna, 2003.

SYMANTEC. Symantec Connect, 23 Jul 2010. Disponivel em: <https://www.symantec.com/connect/articles/what-epoch-time-and-how-convert-human-understandable-format>. Acesso em: 06 dez. 2019.

TANENBAUM, A. S. **Redes de Computadores**. Rio de Janeio: Campus, 1997.

TANENBAUM, A.; WETHERALL, D. **Redes de Computadores**. São Paulo: Pearson, 2011.

VICENTE, D. T. D.; SENGER, E. C. **Aplicação dos padrões a aorma IEC61850 a subestações compartilhadas de transmissão/distribuição de energia elétrica**. São Paulo. 2011.

ANEXO A

Figura 60 - Classe Program

```
static void Main(string[] args)
        {

            Arquivos c = new Arquivos();
            Calendario a = new Calendario();
            Log l = new Log();
            SCD.SCD s = new SCD.SCD();
            Rede rede = new Rede();
            //Teste 20180628
            {

                //Teste Escrita
                /*c.AbrirArquivo("c:\\dados\\macoratti6.txt",
true,"ESCREVER");
                c.EscreveLinha("TESTE7");
                c.FecharArquivo();*/

                //Teste Leitura
                /*c.AbrirArquivo("c:\\dados\\macoratti6.txt", true,
"LER");
                String linha2 = c.LerLinha();
                while (linha2 != null)
                {
                    //Console.WriteLine(linha2);
                    linha2 = c.LerLinha();
                    if(linha2 != null)
                    if (linha2.IndexOf("TESTEOK") != -1)
                        Console.WriteLine(linha2.Trim());

                }
                c.FecharArquivo();*/

                //Teste Renomear
                /*if(!c.RenomearArquivo("c:\\dados\\Carol2.txt",
"c:\\dados\\Carol.txt"))
                {
                    Console.WriteLine("Arquivo não renomeado!");
                }
                else
                {
                    Console.WriteLine("Arquivo    renomeado    com
sucesso!");
                }
                */


                //Teste Copiar
                /*if(!c.CopiarArquivo("c:\\dados\\Carol2.txt",
"c:\\dados\\Carol.txt"))
                {
                    Console.WriteLine("Arquivo não copiado!");
                }
```

```
                else
                {
                    Console.WriteLine("Arquivo     copiado     com
sucesso!");
                }*/

                //Teste Criar
                /*if (!c.CriarArquivo("c:\\dados\\Carol10.txt"))
                {
                    Console.WriteLine("Arquivo não criado!");
                }
                else
                {
                    Console.WriteLine("Arquivo      criado      com
sucesso!");
                }*/

                //Teste Excluir
                /*if
(!c.ExcluirArquivo("c:\\dados\\macoratti6.txt"))
                {
                    Console.WriteLine("Arquivo não excluido!");
                }
                else
                {
                    Console.WriteLine("Arquivo     excluído     com
sucesso!");
                }*/

                //Teste dia
                //Console.WriteLine(a.DDMMYYYYHHMMSS());


                //Teste log
                //l.EscreveLog("c:\\dados\\arquivo.log",    "INFO",
"Escrevendo log de teste!");

                //Teste Leitura SCD;
                //s.AbrirSCDXML("c:\\dados\\FILE.SCD");

            }
            //Teste Leitura SCD;
            s.ExecutaSCD();

            //Teste Leitura GDT;
            SCD.GDT.translateDataAttribute();

            //Teste Leitura GDT;
            //SCD.GDT.translateDataAttribute();

            //Teste Inicializa Thread Valida Goose
            threadValidaRecepcaoGoose.gooseThreadValidaRecepcao();

            /**
            for (int z = 0; z < SCD.DataSet.dataSetArray.Count; z++)
            {
```

```
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine(SCD.DataSet.dataSetArray[z]);
                Console.ResetColor();
                Thread.Sleep(500);
            }
            **/
            //Teste Listar Interfaces de Rede
            //Console.WriteLine(rede.ListaInterfaces());

            //Teste Leitura GMAC
            Gmac.gmacExecutar();

            String numeroInterface = rede.ListaInterfaces();

            Console.WriteLine(numeroInterface);
            //Cria um array devices para separar os dados das
interfaces
            ArrayList devices = new ArrayList();
            //Separa os dados recebido do ListaInterfaces com ";"
            devices.Add(numeroInterface.Split(';'));
            Console.WriteLine(devices[0]);
            //Verifica o tamanho do Array devices, este devera ser
maior que 0 "zero"
            Console.WriteLine("            Count:            {0}",
devices.Count.ToString());

            rede.AbrirInterface();


            Console.WriteLine("Pressione ENTER para sair!");
            Console.ReadKey();
        }
```

Fonte: O autor

ANEXO B

Figura 61 - Classe AppID

```
        public static String goose_APPID(String parameter)
        {

            /**
             *
             *  Separa  o  gooseAPPID  do  frame  recebido  e
reescreve a variavel sem a informação gooseAPPID. (2 bytes)
             *
             * */
            var gooseAPPID = parameter.Substring(0, 4);
            Console.WriteLine("APPID = " + gooseAPPID);
            parameter                                          =
(parameter.Substring(gooseAPPID.Length, (parameter.Length -
(int) gooseAPPID.Length)));
            //Console.WriteLine("FRAME = "+parameter);
            parameter = gooseAPPID + ";" + parameter;

            return parameter;


             }
```

Fonte: O autor

ANEXO C

Figura 62 - Classe GooseLength

```
        public static String goose_GooseLength(String
parameter)
        {

            var gooseGooseLength = parameter.Substring(0,
4);
            //Console.WriteLine("GOOSE PACKET LENGTH = " +
gooseGooseLength);
            int             decValueGooseLength             =
int.Parse(gooseGooseLength,
System.Globalization.NumberStyles.HexNumber);
            Console.WriteLine("GOOSE PACKET LENGTH = " +
gooseGooseLength + " GOOSE PDU LENGTH DEC = " +
decValueGooseLength + " byte");
            parameter                                       =
(parameter.Substring(gooseGooseLength.Length,
(parameter.Length - (int)gooseGooseLength.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            parameter = gooseGooseLength + ";" + parameter;

            return parameter;
        }
```

Fonte: O autor

ANEXO D

Figura 63 - Classe PDU

```
* Identifica o tamanho do pacote GOOSE. Campo com valor
em HEX. (1 byte)
            * O frame é reescrito sem esta informação.
            * O valor em HEX é convertido para Decimal.
            *
            * */
            var goosePDU = parameter.Substring(0, 2);
            Console.WriteLine("GOOSE   PDU   =   "   +
Convert.ToDecimal(goosePDU));
            parameter                                   =
(parameter.Substring(goosePDU.Length,
(parameter.Length - (int)goosePDU.Length)));
            //Console.WriteLine("GOOSE PDU LENGTH DEC =
"+ decValuePDULength);

            var gooseLength = "";
            int decValuePDULength;
            int x = 0;
            int y = 0;
            String aux = parameter;
            while (!(aux.Substring(0, 2).IndexOf("80")
!= -1))
            {
                //Console.WriteLine(gooseLength);
                gooseLength  =  parameter.Substring(0,
x);
                //decValuePDULength                     =
int.Parse(gooseLength,
System.Globalization.NumberStyles.HexNumber);
                //Console.WriteLine("GOOSE LENGTH = " +
gooseLength);//+  "  GOOSE  PDU  LENGTH  DEC  =  "  +
decValuePDULength + " byte");
                aux = (parameter.Substring(x,2));
                x = x + 1;
            }
            Console.WriteLine("GOOSE  LENGTH  =  "  +
gooseLength);
            if(gooseLength.Length == 2)
            {
                decValuePDULength                     =
int.Parse(gooseLength,
System.Globalization.NumberStyles.HexNumber);
                Console.WriteLine("GOOSE LENGTH = " +
gooseLength  +  "  GOOSE  PDU  LENGTH  DEC  =  "  +
decValuePDULength + " byte");
            }
            if (gooseLength.Length == 4)
            {
                decValuePDULength                     =
int.Parse(gooseLength.Substring(2,2),
System.Globalization.NumberStyles.HexNumber);
```

```
                Console.WriteLine("GOOSE LENGTH = " +
gooseLength + " GOOSE PDU LENGTH DEC = " +
decValuePDULength + " byte");
            }
            if (gooseLength.Length == 6)
            {
                decValuePDULength                    =
int.Parse(gooseLength.Substring(2,                 4),
System.Globalization.NumberStyles.HexNumber);
                Console.WriteLine("GOOSE LENGTH = " +
gooseLength + " GOOSE PDU LENGTH DEC = " +
decValuePDULength + " byte");
            }
            parameter                                =
(parameter.Substring(gooseLength.Length,
(parameter.Length - (int)gooseLength.Length)));

            //int          decValuePDULength          =
int.Parse(gooseLength,
System.Globalization.NumberStyles.HexNumber);
            //Console.WriteLine("GOOSE LENGTH = " +
gooseLength + " GOOSE PDU LENGTH DEC = " +
decValuePDULength + " byte");
            //Console.WriteLine("GOOSE PDU LENGTH DEC =
"+ decValuePDULength);
            //parameter                              =
(parameter.Substring(gooseLength.Length,
(parameter.Length - (int)gooseLength.Length)));

            parameter = goosePDU + ";" + gooseLength +
";" + parameter;

            return parameter;
              }
```

Fonte: Pessoal

ANEXO E

Figura 64 - Classe ControlRefBlock

```
            * Identifica o campo Control Reference Block, retirando a
informação do inicio do pacote goose.
            * Campo sempre com valor 80HEX. (1 byte)
            **/
            var gooseControlRefBlock = parameter.Substring(0, 2);
            Console.WriteLine("CONTROL REFERENCE BLOCK = " +
gooseControlRefBlock);
            parameter = (parameter.Substring(gooseControlRefBlock.Length,
(parameter.Length - (int) gooseControlRefBlock.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             * Identifica o tamanho do pacote Control Reference Block.
Campo com valor em HEX. (1 byte)
             * O frame é reescrito sem esta informação.
             * O valor em HEX é convertido para Decimal.
             *
             * */
            var gooseControlRefBlockLength = parameter.Substring(0, 2);
            int decValuegooseControlRefBlockLength =
int.Parse(gooseControlRefBlockLength,
System.Globalization.NumberStyles.HexNumber);
            Console.WriteLine("CONTROL REFERENCE BLOCK LENGTH = " +
gooseControlRefBlockLength + " CONTROL REFERENCE BLOCK DEC = " +
decValuegooseControlRefBlockLength + " byte");
            //Console.WriteLine("GOOSE PDU LENGTH DEC = "+
decValuePDULength);
            parameter =
(parameter.Substring(gooseControlRefBlockLength.Length, (parameter.Length
- (int)gooseControlRefBlockLength.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Extrai do frame os dados referente a variavel CONTROL
REFERENCE BLOCK.
             * O tamanho da variavel é baseado no item
gooseControlRefBlockLength.
             *
             * */
            var gooseDataControlRefBlock = parameter.Substring(0,
(decValuegooseControlRefBlockLength * 2));
            Console.WriteLine("CONTROL REFERENCE BLOCK DATA = " +
gooseDataControlRefBlock);
            Console.WriteLine("CONTROL REFERENCE BLOCK DATA = " +
converter.HexString2Ascii(gooseDataControlRefBlock));
            parameter =
(parameter.Substring(gooseDataControlRefBlock.Length, (parameter.Length -
(int)gooseDataControlRefBlock.Length)));

            parameter = gooseControlRefBlock + ";" +
gooseControlRefBlockLength+";"+gooseDataControlRefBlock+";"+ parameter;

                return parameter;
```

Fonte: O autor

ANEXO F

Figura 65 - Classe Time2LiveAllowed

```
/**
             * Identifica o campo Time Allowed to Live, retirando a
informação do inicio do pacote goose.
             * Campo sempre com valor 81HEX. (1 byte)
             * */
            var gooseTimeAllowedtoLive = parameter.Substring(0, 2);
            Console.WriteLine("TIME   ALLOWED   TO   LIVE   =   "   +
gooseTimeAllowedtoLive);
            parameter                                                =
(parameter.Substring(gooseTimeAllowedtoLive.Length,
(parameter.Length - (int)gooseTimeAllowedtoLive.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             * Identifica o tamanho do pacote Time Allowed to Live.
Campo com valor em HEX. (1 byte)
             * O frame é reescrito sem esta informação.
             * O valor em HEX é convertido para Decimal.
             * */
            var            gooseTimeAllowedtoLiveLength              =
parameter.Substring(0, 2);
            int        decValuegooseTimeAllowedtoLiveLength          =
int.Parse(gooseTimeAllowedtoLiveLength,
System.Globalization.NumberStyles.HexNumber);
            Console.WriteLine("TIME  ALLOWED  TO  LIVE  LENGTH  =  "  +
gooseTimeAllowedtoLiveLength + " TIME ALLOWED TO LIVE LENGTH DEC = "
+ decValuegooseTimeAllowedtoLiveLength + " byte");
            //Console.WriteLine("GOOSE   PDU   LENGTH   DEC   =   "+
decValuePDULength);
            parameter                                                =
(parameter.Substring(gooseTimeAllowedtoLiveLength.Length,
(parameter.Length - (int)gooseTimeAllowedtoLiveLength.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             * Extrai do frame os dados referente a variavel Time
Allowed to Live.
             *  O  tamanho  da  variavel  é  baseado  no  item
gooseTimeAllowedtoLiveLength.
             *
             * */
            var gooseDataTimeAllowedtoLive = parameter.Substring(0,
(decValuegooseTimeAllowedtoLiveLength * 2));
            Console.WriteLine("TIME  ALLOWED  TO  LIVE  DATA  =  "  +
gooseDataTimeAllowedtoLive);
            parameter                                                =
(parameter.Substring(gooseDataTimeAllowedtoLive.Length,
(parameter.Length - (int)gooseDataTimeAllowedtoLive.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            parameter          =          gooseTimeAllowedtoLive+";"+
gooseTimeAllowedtoLiveLength+";"+
gooseDataTimeAllowedtoLive+";"+parameter;

            return parameter;
```

Fonte: O autor

ANEXO G

Figura 66 - Classe DataSet

```
            /**
             *
             * Identifica o campo Data Set, retirando a informação
do inicio do pacote goose.
             * Campo sempre com valor 82HEX. (1 byte)
             * */
            var gooseDataSet = parameter.Substring(0, 2);
            Console.WriteLine("DATA SET = " + gooseDataSet);
            parameter  =  (parameter.Substring(gooseDataSet.Length,
(parameter.Length - (int)gooseDataSet.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Identifica o tamanho do pacote Data Set. Campo com
valor em HEX. (1 byte)
             * O frame é reescrito sem esta informação.
             * O valor em HEX é convertido para Decimal.
             *
             * */
            var gooseDataSetLength = parameter.Substring(0, 2);
            int             decValuegooseDataSetLength             =
int.Parse(gooseDataSetLength,
System.Globalization.NumberStyles.HexNumber);
            Console.WriteLine("DATA    SET    LENGTH    =    "    +
gooseDataSetLength   +   "   DATA   SET   LENGTH   DEC   =   "   +
decValuegooseDataSetLength + " byte");
            //Console.WriteLine("GOOSE   PDU   LENGTH   DEC   =   "+
decValuePDULength);
            parameter                                              =
(parameter.Substring(gooseDataSetLength.Length, (parameter.Length -
(int)gooseDataSetLength.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Extrai do frame os dados referente a variavel Data
Set.
             *  O  tamanho  da  variavel  é  baseado  no  item
gooseDataSetLegth.
             *
             * */
            var    gooseDataDataSet    =    parameter.Substring(0,
(decValuegooseDataSetLength * 2));
            Console.WriteLine("DATA        SET        DATA        =
"+converter.HexString2Ascii(gooseDataDataSet));
            parameter                                              =
(parameter.Substring(gooseDataDataSet.Length,  (parameter.Length  -
(int)gooseDataDataSet.Length)));
            //Console.WriteLine("FRAME = " + parameter);

            parameter  =  gooseDataSet+";"+  gooseDataSetLength+";"+
gooseDataDataSet+";"+parameter;
            return parameter;
```

Fonte: O autor

ANEXO H

Figura 67 - Classe GooseID

```
/**
             *
             * Identifica o campo GOOSE ID, retirando a informação
do inicio do pacote goose.
             * Campo sempre com valor 83HEX. (1 byte)
             *
             * */
            var gooseID = parameter.Substring(0, 2);
            Console.WriteLine("GOOSE ID = " + gooseID);
            parameter    =    (parameter.Substring(gooseID.Length,
(parameter.Length - (int)gooseID.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Identifica o tamanho do pacote Goose ID. Campo com
valor em HEX. (1 byte)
             * O frame é reescrito sem esta informação.
             * O valor em HEX é convertido para Decimal.
             *
             * */
            var gooseIDLength = parameter.Substring(0, 2);
            int  decValuegooseIDLength  =  int.Parse(gooseIDLength,
System.Globalization.NumberStyles.HexNumber);
            Console.WriteLine("GOOSE ID LENGTH = " + gooseIDLength
+ " DATA SET LENGTH DEC = " + decValuegooseIDLength + " byte");
            //Console.WriteLine("GOOSE   PDU   LENGTH   DEC   =   "+
decValuePDULength);
            parameter = (parameter.Substring(gooseIDLength.Length,
(parameter.Length - (int)gooseIDLength.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Extrai do frame os dados referente a variavel Goose
ID.
             *  O  tamanho  da  variavel  é  baseado  no  item
gooseDataSetLegth.
             *
             * */
            var     gooseIdData     =     parameter.Substring(0,
(decValuegooseIDLength * 2));
            Console.WriteLine("GOOSE ID DATA = " + gooseIdData);
            Console.WriteLine("GOOSE       ID       DATA       =
"+converter.HexString2Ascii(gooseIdData));
            parameter  =  (parameter.Substring(gooseIdData.Length,
(parameter.Length - (int)gooseIdData.Length)));
            //Console.WriteLine("FRAME = " + parameter);

            parameter = gooseID + ";" + gooseIDLength + ";" +
gooseIdData + ";" + parameter;

            return parameter;
```

Fonte: O autor

ANEXO I

Figura 68 - Classe TimeStamp

```
/**
             *
             * Identifica o campo Time Stamp, retirando a informação
do inicio do pacote goose.
             * Campo sempre com valor 84HEX. (1 byte)
             *
             * */
            var gooseTimeStamp = parameter.Substring(0, 2);
            Console.WriteLine("GOOSE    TIME    STAMP    =    "    +
gooseTimeStamp);
            parameter                                              =
(parameter.Substring(gooseTimeStamp.Length,   (parameter.Length   -
(int)gooseTimeStamp.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Identifica o tamanho do pacote Time Stamp. Campo com
valor em HEX. (1 byte)
             * O frame é reescrito sem esta informação.
             * O valor em HEX é convertido para Decimal.
             *
             * */
            var gooseTimeStampLength = parameter.Substring(0, 2);
            int            decValuegooseTimeStampLength            =
int.Parse(gooseTimeStampLength,
System.Globalization.NumberStyles.HexNumber);
            Console.WriteLine("GOOSE  TIME  STAMP  LENGTH  =  "  +
gooseTimeStampLength  +  "  GOOSE  TIME  STAMP  LENGTH  DEC  =  "  +
decValuegooseTimeStampLength + " byte");
            //Console.WriteLine("GOOSE   PDU   LENGTH   DEC   =   "+
decValuePDULength);
            parameter                                              =
(parameter.Substring(gooseTimeStampLength.Length,
(parameter.Length - (int)gooseTimeStampLength.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Extrai do frame os dados referente a variavel Time
Stamp.
             *  O  tamanho  da  variavel  é  baseado  no  item
gooseTimeStampLegth.
             *
             * */
            var    gooseTimeStampData    =    parameter.Substring(0,
(decValuegooseTimeStampLength * 2));
            Console.WriteLine("GOOSE  TIME  STAMP  DATA  =  "  +
gooseTimeStampData);
            Console.WriteLine("GOOSE    TIME    STAMP    DATA    =
"+h2epoch.HexString2Epoch(gooseTimeStampData.Substring(0, 8)));
            Console.WriteLine("GOOSE TIME STAMP MILLISECONDS = " +
h2epoch.getMilliseconds(gooseTimeStampData.Substring(8, 8)));
```

```
            parameter                                                    =
(parameter.Substring(gooseTimeStampData.Length,   (parameter.Length
- (int)gooseTimeStampData.Length)));
            //Console.WriteLine("FRAME = " + parameter);

            parameter             =             gooseTimeStamp+";"+
gooseTimeStampLength+";"+gooseTimeStampData + ";" + parameter;


            return parameter;
```

Fonte: O autor

ANEXO J

Figura 69 - Classe StateNumber

```
/**
                *
Identifica o campo State Number, retirando a informação do
inicio do pacote goose.
Campo sempre com valor 85HEX. (1 byte)
                *
*/
                var gooseStateNumber = parameter.Substring(0,
                2);
                //Console.WriteLine("GOOSE STATE NUMBER = " +
                gooseStateNumber);
                parameter                                    =
                (parameter.Substring(gooseStateNumber.Length
                ,             (parameter.Length              -
                (int)gooseStateNumber.Length)));
                //Console.WriteLine("FRAME = " + parameter);
                /**
                *
Identifica o tamanho do pacote state Number. Campo com valor
em HEX. (1 byte)
O frame é reescrito sem esta informação.
O valor em HEX é convertido para Decimal.
                *
*/
                var          gooseStateNumberLength          =
                parameter.Substring(0, 2);
                int      decValuegooseStateNumberLength      =
                int.Parse(gooseStateNumberLength,
                System.Globalization.NumberStyles.HexNumber)
                ;
                //Console.WriteLine("GOOSE    STATE    NUMBER
                LENGTH = " + gooseStateNumberLength + " GOOSE
                STATE    NUMBER    LENGTH    DEC    =    "    +
                decValuegooseStateNumberLength + " byte");
                //Console.WriteLine("GOOSE PDU LENGTH DEC =
                "+ decValuePDULength);
                parameter                                    =
                (parameter.Substring(gooseStateNumberLength.
                Length,          (parameter.Length           -
                (int)gooseStateNumberLength.Length)));
                //Console.WriteLine("FRAME = " + parameter);
                /**
                *
Extrai do frame os dados referente a variavel State Number.
O tamanho da variavel é baseado no item gooseTimeStampLegth.
                *
*/
                var           gooseStateNumberData           =
                parameter.Substring(0,
                (decValuegooseStateNumberLength * 2));
                //Console.WriteLine("GOOSE STATE NUMBER DATA
                = " + gooseStateNumberData);
```

```
                    parameter                                          =
                    (parameter.Substring(gooseStateNumberData.Le
                    ngth,           (parameter.Length           -
                    (int)gooseStateNumberData.Length)));
                    //Console.WriteLine("FRAME = " + parameter);

                    parameter   =   gooseStateNumber   +   ";"   +
                    gooseStateNumberLength      +      ";"      +
                    gooseStateNumberData + ";" + parameter;

                return parameter;
```

Fonte: O autor

ANEXO K

Figura 70 - Classe SequenceNumber

```
            /**
             *
             *  Identifica  o  campo  Sequence  Number,  retirando  a
informação do inicio do pacote goose.
             * Campo sempre com valor 86HEX. (1 byte)
             *
             * */
            var gooseSequenceNumber = parameter.Substring(0, 2);
            //Console.WriteLine("GOOSE  SEQUENCE  NUMBER  =  "  +
gooseSequenceNumber);
            parameter                                                =
(parameter.Substring(gooseSequenceNumber.Length,  (parameter.Length
- (int)gooseSequenceNumber.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Identifica o tamanho do pacote Sequence Number. Campo
com valor em HEX. (1 byte)
             * O frame é reescrito sem esta informação.
             * O valor em HEX é convertido para Decimal.
             *
             * */
            var  gooseSequenceNumberLength  =  parameter.Substring(0,
2);
            int         decValuegooseSequenceNumberLength         =
int.Parse(gooseSequenceNumberLength,
System.Globalization.NumberStyles.HexNumber);
            //Console.WriteLine("GOOSE SEQUENCE NUMBER LENGTH = " +
gooseSequenceNumberLength + " GOOSE SEQUENCE NUMBER LENGTH DEC = "
+ decValuegooseSequenceNumberLength + " byte");
            //Console.WriteLine("GOOSE  PDU  LENGTH  DEC  =  "+
decValuePDULength);
            parameter                                                =
(parameter.Substring(gooseSequenceNumberLength.Length,
(parameter.Length - (int)gooseSequenceNumberLength.Length)));
            //Console.WriteLine("FRAME = " + parameter);
            /**
             *
             * Extrai do frame os dados referente a variavel Sequence
Number.
             *  O  tamanho  da  variavel  é  baseado  no  item
gooseTimeStampLegth.
             *
             * */
            var  gooseSequenceNumberData  =  parameter.Substring(0,
(decValuegooseSequenceNumberLength * 2));
            //Console.WriteLine("GOOSE SEQUENCE NUMBER DATA = " +
gooseSequenceNumberData);
            //Console.WriteLine("GOOSE  SEQUENCE  NUMBER  DATA  =
"+h2i.HexString2Int(gooseSequenceNumberData));
            parameter                                                =
(parameter.Substring(gooseSequenceNumberData.Length,
(parameter.Length - (int)gooseSequenceNumberData.Length)));
```

```
            //Console.WriteLine("FRAME = " + parameter);

            parameter    =    gooseSequenceNumber    +    ";"    +
gooseSequenceNumberLength + ";" + gooseSequenceNumberData + ";" +
parameter;

        •               return parameter;
```

Fonte: O autor

ANEXO L

Figura 71 - Classe IEDState

```
                /**
                *
Identifica o campo IED State, retirando a informação do inicio do
pacote goose.
Campo sempre com valor 87HEX. (1 byte)
                *
   */
                var gooseIEDState = parameter.Substring(0, 2);
                //Console.WriteLine("GOOSE   IED   STATE   =   "   +
                gooseIEDState);
                parameter                                          =
                (parameter.Substring(gooseIEDState.Length,
                (parameter.Length - (int)gooseIEDState.Length)));
                //Console.WriteLine("FRAME = " + parameter);
                /**
                *
Identifica o tamanho do pacote IED State. Campo com valor em HEX.
(1 byte)
O frame é reescrito sem esta informação.
O valor em HEX é convertido para Decimal.
                *
   */
                var gooseIEDStateLength = parameter.Substring(0, 2);
                int          decValuegooseIEDStateLength           =
                int.Parse(gooseIEDStateLength,
                System.Globalization.NumberStyles.HexNumber);
                //Console.WriteLine("GOOSE IED STATE LENGTH = " +
                gooseIEDStateLength + " GOOSE IED STATE LENGTH DEC
                = " + decValuegooseIEDStateLength + " byte");
                //Console.WriteLine("GOOSE  PDU  LENGTH  DEC  =  "+
                decValuePDULength);
                parameter                                          =
                (parameter.Substring(gooseIEDStateLength.Length,
                (parameter.Length                                  -
                (int)gooseIEDStateLength.Length)));
                //Console.WriteLine("FRAME = " + parameter);
                /**
                *
Extrai do frame os dados referente a variavel IED State.
O tamanho da variavel é baseado no item gooseTimeStampLegth.
                *
   */
                var   gooseIEDStateData   =   parameter.Substring(0,
                (decValuegooseIEDStateLength * 2));
                //Console.WriteLine("GOOSE  IED  STATE  DATA  =  "  +
                gooseIEDStateData);
                parameter                                          =
                (parameter.Substring(gooseIEDStateData.Length,
                (parameter.Length                                  -
                (int)gooseIEDStateData.Length)));
                //Console.WriteLine("FRAME = " + parameter);
```

```
                parameter = gooseIEDState + ";" +
                gooseIEDStateLength + ";" + gooseIEDStateData + ";"
                + parameter;

                 return parameter;
```

Fonte: O autor

ANEXO M

Figura 72 - Classe ConfigRevision

```
                    /**
                    *
Identifica o campo  Config Revision, retirando a informação do
inicio do pacote goose.
Campo sempre com valor 88HEX. (1 byte)
                    *
*/
                    var gooseConfigRevision = parameter.Substring(0,
                    2);
                    //Console.WriteLine("GOOSE CONFIG REVISION = " +
                    gooseConfigRevision);
                    parameter                                       =
                    (parameter.Substring(gooseConfigRevision.Length
                    ,               (parameter.Length               -
                    (int)gooseConfigRevision.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
                    /**
                    *
Identifica o tamanho do pacote  Config Revision. Campo com valor
em HEX. (1 byte)
O frame é reescrito sem esta informação.
O valor em HEX é convertido para Decimal.
                    *
*/
                    var          gooseConfigRevisionLength          =
                    parameter.Substring(0, 2);
                    int      decValuegooseConfigRevisionLength      =
                    int.Parse(gooseConfigRevisionLength,
                    System.Globalization.NumberStyles.HexNumber);
                    //Console.WriteLine("GOOSE    CONFIG    REVISION
                    LENGTH = " + gooseConfigRevisionLength + " GOOSE
                    CONFIG    REVISION    LENGTH    DEC    =    "    +
                    decValuegooseConfigRevisionLength + " byte");
                    //Console.WriteLine("GOOSE  PDU  LENGTH  DEC  =  "+
                    decValuePDULength);
                    parameter                                       =
                    (parameter.Substring(gooseConfigRevisionLength.
                    Length,             (parameter.Length           -
                    (int)gooseConfigRevisionLength.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
                    /**
                    *
Extrai do frame os dados referente a variavel  Config Revision.
O tamanho da variavel é baseado no item gooseTimeStampLegth.
                    *
*/
                    var           gooseConfigRevisionData           =
                    parameter.Substring(0,
                    (decValuegooseConfigRevisionLength * 2));
                    //Console.WriteLine("GOOSE  CONFIG  REVISION  DATA
                    = " + gooseConfigRevisionData);
                    parameter                                       =
                    (parameter.Substring(gooseConfigRevisionData.Le
```

```
                    ngth,              (parameter.Length              -
                    (int)gooseConfigRevisionData.Length)));
                    //Console.WriteLine("FRAME = " + parameter);

                    parameter   =   gooseConfigRevision   +   ";"   +
                    gooseConfigRevisionLength      +      ";"      +
                    gooseConfigRevisionData + ";" + parameter;

                return parameter;
```

Fonte: O autor

ANEXO N

Figura 73 - Classe NeedRevision

```
                    /**
                     *
                     * Identifica o campo  Need Revision, retirando a
informação do inicio do pacote goose.
                     * Campo sempre com valor 89HEX. (1 byte)
                     *
                     * */
                    var gooseNeedRevision = parameter.Substring(0, 2);
                    //Console.WriteLine("GOOSE  NEED  REVISION  =  "  +
gooseNeedRevision);
                    parameter                                          =
(parameter.Substring(gooseNeedRevision.Length,  (parameter.Length  -
(int)gooseNeedRevision.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
                    /**
                     *
                     * Identifica o tamanho do pacote  Need Revision.
Campo com valor em HEX. (1 byte)
                     * O frame é reescrito sem esta informação.
                     * O valor em HEX é convertido para Decimal.
                     *
                     * */
                    var           gooseNeedRevisionLength              =
parameter.Substring(0, 2);
                    int       decValuegooseNeedRevisionLength          =
int.Parse(gooseNeedRevisionLength,
System.Globalization.NumberStyles.HexNumber);
                    //Console.WriteLine("GOOSE NEED REVISION LENGTH =
" + gooseNeedRevisionLength + " GOOSE NEED REVISION LENGTH DEC = "
+ decValuegooseNeedRevisionLength + " byte");
                    //Console.WriteLine("GOOSE  PDU  LENGTH  DEC  =  "+
decValuePDULength);
                    parameter                                          =
(parameter.Substring(gooseNeedRevisionLength.Length,
(parameter.Length - (int)gooseNeedRevisionLength.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
                    /**
                     *
                     * Extrai do frame os dados referente a variavel
Need Revision.
                     *  O  tamanho  da  variavel  é  baseado  no  item
gooseTimeStampLegth.
                     *
                     * */
                    var gooseNeedRevisionData = parameter.Substring(0,
(decValuegooseNeedRevisionLength * 2));
                    //Console.WriteLine("GOOSE NEED REVISION DATA = "
+ gooseNeedRevisionData);
                    parameter                                          =
(parameter.Substring(gooseNeedRevisionData.Length,
(parameter.Length - (int)gooseNeedRevisionData.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
```

```
                    parameter = gooseNeedRevision + ";" +
gooseNeedRevisionLength + ";" + gooseNeedRevisionData + ";" +
parameter;

                    return parameter;
```

Fonte: O autor

ANEXO O

Figura 74 - Classe DataEntries

```
                    /**
                     *
                     * Identifica o campo  Data Entries, retirando
a informação do inicio do pacote goose.
                     * Campo sempre com valor 8aHEX. (1 byte)
                     *
                     * */
                    var gooseDataEntries = parameter.Substring(0,
2);
                    //Console.WriteLine("GOOSE DATA ENTRIES = " +
gooseDataEntries);
                    parameter                                    =
(parameter.Substring(gooseDataEntries.Length,
(parameter.Length - (int)gooseDataEntries.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
                    /**
                     *
                     *  Identifica  o  tamanho  do  pacote   Data
Entries. Campo com valor em HEX. (1 byte)
                     * O frame é reescrito sem esta informação.
                     * O valor em HEX é convertido para Decimal.
                     *
                     * */
                    var          gooseDataEntriesLength          =
parameter.Substring(0, 2);
                    int      decValuegooseDataEntriesLength      =
int.Parse(gooseDataEntriesLength,
System.Globalization.NumberStyles.HexNumber);
                    //Console.WriteLine("GOOSE DATA ENTRIES LENGTH
= " + gooseDataEntriesLength + " GOOSE DATA ENTRIES LENGTH DEC
= " + decValuegooseDataEntriesLength + " byte");
                    //Console.WriteLine("GOOSE PDU LENGTH DEC = "+
decValuePDULength);
                    parameter                                    =
(parameter.Substring(gooseDataEntriesLength.Length,
(parameter.Length - (int)gooseDataEntriesLength.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
                    /**
                     *
                     *  Extrai  do  frame  os  dados  referente  a
variavel  Data Entries.
                     * O tamanho da variavel é baseado no item
gooseTimeStampLegth.
                     *
                     * */
                    var          gooseDataEntriesData            =
parameter.Substring(0, (decValuegooseDataEntriesLength * 2));
                    //Console.WriteLine("GOOSE DATA ENTRIES DATA =
" + gooseDataEntriesData);
                    parameter                                    =
(parameter.Substring(gooseDataEntriesData.Length,
(parameter.Length - (int)gooseDataEntriesData.Length)));
                    //Console.WriteLine("FRAME = " + parameter);
```

```
                parameter = gooseDataEntries + ";" +
gooseDataEntriesLength + ";" + gooseDataEntriesData + ";" +
parameter;

    •             return parameter;
```

Fonte: O autor

ANEXO P

A estampa de tempo de uma mensagem GOOSE é localizada no bloco 23 do quadro *Ethernet* IEEE 802.3*,* já apresentado na Tabela 3. A informação de estampa de tempo localizada no bloco 23 é constituída por um conjunto de 8 *bytes* no formato *hexadecimal,* sendo os quatros primeiros *bytes* utilizados para conversão do total de segundos e os quatros últimos utilizados para conversão dos milissegundos. (IEC-61850-7-2, 2010)

Para conversão dos segundos, o valor hexadecimal constituído por quatro *bytes* é convertido para um valor binário formado por trinta e dois *bits*, sendo o primeiro *bit "0"* menos significativo e o último *bit "31"* o mais significativo.

A *Tabela 9* apresenta o cálculo utilizado para conversão do valor hexadecimal 4ADA86F4 para segundos. Nota-se que o valor convertido para binário é 01001010110110101000011011110100 e o valor binário convertido para decimal é 1255835380 total de segundos desde 01/01/1970 00:00:00.000.

Tabela 9 - Conversão de segundos epoch

bit	2^*bit*	Valor do *bit*	(2^bit x Valor do bit)
31	2147483648	0	0
30	1073741824	1	1073741824
29	536870912	0	0
28	268435456	0	0
27	134217728	1	134217728
26	67108864	0	0
25	33554432	1	33554432
24	16777216	0	0
23	8388608	1	8388608
22	4194304	1	4194304
21	2097152	0	0
20	1048576	1	1048576
19	524288	1	524288
18	262144	0	0
17	131072	1	131072

16	65536	0	0
15	32768	1	32768
14	16384	0	0
13	8192	0	0
12	4096	0	0
11	2048	0	0
10	1024	1	1024
9	512	1	512
8	256	0	0
7	128	1	128
6	64	1	64
5	32	1	32
4	16	1	16
3	8	0	0
2	4	1	4
1	2	0	0
0	1	0	0
Total de segundos = ∑(2^bit x Valor do bit)			1255835380

Fonte: O autor

Para conversão dos milissegundos, o valor hexadecimal constituído pelos quatro últimos *bytes* é convertido para um valor binário formado por trinta e dois *bits*, sendo o primeiro *bit "1"* mais significativo e o último *bit "32"* o menos significativo. Para conversão dos milissegundos são utilizados os vinte e quatros primeiros *bits,* os oitos *bits* restantes são utilizados para qualidade.

A Tabela 10 apresenta o cálculo utilizado para conversão do valor hexadecimal 7D1B40 para milissegundos. Nota-se que o valor convertido para binário é 011111010001101101000000 e o valor binário convertido para decimal é 0,488697052 total de milissegundos do valor *epoch.*

Tabela 10 - Conversão epoch milissegundos

bit	1 / (2^*bit)*	Valor do *bit*	(1 / (2^bit)) x (Valor do *bit)*
1	0,5	0	0
2	0,25	1	0,25
3	0,125	1	0,125
4	0,0625	1	0,0625
5	0,03125	1	0,03125
6	0,015625	1	0,015625
7	0,0078125	0	0
8	0,00390625	1	0,00390625
9	0,001953125	0	0
10	0,000976563	0	0
11	0,000488281	0	0
12	0,000244141	1	0,000244141
13	0,00012207	1	0,00012207
14	6,10352E-05	0	0
15	3,05176E-05	1	3,05176E-05
16	1,52588E-05	1	1,52588E-05
17	7,62939E-06	0	0
18	3,8147E-06	1	3,8147E-06
19	1,90735E-06	0	0
20	9,53674E-07	0	0
21	4,76837E-07	0	0
22	2,38419E-07	0	0
23	1,19209E-07	0	0
24	5,96046E-08	0	0
Total de milissegundos = ((1 / (2^*bit))* x Valor do bit)			0,488697052

Fonte: O autor

ANEXO Q

Figura 75 - Código parametrizado para detectar falha GOOSE

```
                    if((epoch_now_var                                -
time_stamp_goose_recebeido) > 20)
                    {
                        Console.WriteLine("Tempo sem receber
GOOSE = " + (epoch_now_var - time_stamp_goose_recebeido));
                        Console.WriteLine("IED
"+validar_pacote[1]+" falha no envio goose devido ao tempo >
20 segundos");
                        Thread.Sleep(2000);
                    }
```

Fonte: O autor