UNIVERSIDADE DE SÃO PAULO–USP
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Lázaro Eduardo da Silva

**Sistema Embarcado Inteligente para
Detecção de Intrusão em Subestações
de Energia Elétrica Utilizando o
Protocolo *OpenFlow***

Lázaro Eduardo da Silva

**Sistema Embarcado Inteligente para
Detecção de Intrusão em Subestações
de Energia Elétrica Utilizando o
Protocolo *OpenFlow***

Tese de doutorado apresentada ao Programa de Engenharia Elétrica da Escola de Engenharia de São Carlos como parte dos requisitos para a obtenção do título de Doutor em Ciências.

Área de concentração: Sistemas Elétricos de Potência

Orientador: Prof. Tit. Denis Vinícius Coury

São Carlos

2016

Trata-se da versão corrigida da tese. A versão original se encontra disponível na EESC/USP que aloja o Programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

da Silva, Lázaro Eduardo
D586s Sistema Embarcado Inteligente para Detecção de
Intrusão em Subestações de Energia Elétrica Utilizando
o Protocolo OpenFlow / Lázaro Eduardo da Silva;
orientador Denis Vinicius Coury. São Carlos, 2016.

Tese (Doutorado) - Programa de Pós-Graduação em
Engenharia Elétrica e Área de Concentração em Sistemas
Elétricos de Potência -- Escola de Engenharia de São
Carlos da Universidade de São Paulo, 2016.

1. IEC-61850. 2. IEC-62351. 3. Redes de Comunicação
de Dados. 4. Segurança Cibernética. 5. Sistema de
Detecção de Intrusão. 6. Sistema Elétrico. 7. Sistema
Embarcado. 8. Sistema Inteligente. I. Título.

FOLHA DE JULGAMENTO

Candidato: Bacharel **LÁZARO EDUARDO DA SILVA**.

Título da tese: "Sistema embarcado inteligente para detecção de intrusão em subestações de energia elétrica utilizando o protocolo OpenFlow".

Data da defesa: 05/10/2016.

Comissão Julgadora:

Resultado:

Prof. Titular **Denis Vinicius Coury**
(Orientador)
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Associado **Mario Oleskovicz**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Dr. **Renato Machado Monaro**
(Escola Politécnica/EP-USP)

APROVADO

Prof. Dr. **César Augusto Cavalheiro Marcondes**
(Universidade Federal de São Carlos/UFSCar)

APROVADO

Prof. Dr. **Francisco José Mônaco**
(Instituto de Ciências Matemáticas e de Computação/ICMC-USP)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Associado **Luis Fernando Costa Alberto**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Luis Fernando Costa Alberto**

*Dedico esta tese aos meus pais, Lázaro Silvério e Maria de Fátima,
à minha esposa Joyce, às minhas irmãs Leila, Letícia e Lidianne.
Aos meus cunhados, sobrinhos e afilhados, Luana, Gabriel, Miguel e Marcela,
por fazerem minha vida mais feliz. Vocês também são responsáveis por esta conquista.*

Agradecimentos

A Deus por iluminar sempre meus passos, trazendo realizações para minha vida e colocar pessoas tão especiais, no decorrer desta caminhada.

À minha família, que tanto me apoia, oferecendo ajuda incondicional, acreditando no meu esforço e estando sempre presente nas batalhas e conquistas da minha vida. Em especial ao meu pai a quem Deus levou para mais perto de Si. Tenho certeza de que ele, no céu, está muito feliz com esta conquista.

Ao Professor Tit. Denis Vinicius Coury, da Escola de Engenharia de São Carlos (EESC/USP), por acreditar no meu trabalho e me acolher no programa de Doutorado, por todas as facilidades oferecidas, pela liberdade de trabalho e opinião, pelos desafios propostos, pela orientação amigável e competente, sendo muito mais que um orientador, um exemplo de vida acadêmica.

Ao Professor Dr. Mário Oleskovicz (EESC/USP), pela postura, dedicação e objetividade com que trata os assuntos, promovendo o entendimento, e pelas criteriosa leitura e sugestões oportunas, na fase de qualificação e defesa.

Ao Professor Dr. José Carlos de Melo Vieira Júnior (EESC/USP), pelo convívio no laboratório e em congressos.

Ao Professor Dr. César Augusto Cavalheiros Marcondes (DECOM/UFSCar), pelo cordial atendimento, pelas orientações e contribuições dadas durante a fase de qualificação e defesa deste trabalho.

Ao Professor Dr. Francisco José Mônaco (ICMC/USP), pelas contribuições dadas durante a defesa deste trabalho.

Ao Professor Dr. Renato Machado Monaro (Poli/USP), pelas contribuições dadas durante o desenvolvimento do trabalho, pela criteriosa leitura e sugestões realizadas na defesa deste trabalho.

Ao amigo Juliano Coêlho Miranda e a toda sua família, por compartilharem as viagens, madrugadas de trabalho e por toda ajuda, neste tempo de convivência.

Ao Carlos Magno, por emprestar-me a placa ZYBO.

Ao Vinicius pelas orações.

Aos amigos André, Rui, Rodrigo, Marsolla, Leandro, Jeovane, Luiz, Gustavo, Rodolfo, Jáder, Athila, Monaro, Gerson, Eduardo, Carlos, Luciano, Thais, Viviane, Guacira, Aline e a todos os que fizeram parte do LSEE, pelo compartilhamento de experiências e harmonioso convívio.

Aos profissionais da secretaria da pós-graduação (EESC/USP), pelo atendimento cordial, competente e respeitoso.

Aos colegas das disciplinas do programa de doutorado da EESC/USP e ICMC/USP pelos contatos realizados e pelo compartilhamento de experiências.

Ao Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) e a todos os profissionais que participaram direta e indiretamente desta conquista, em especial aos professores e amigos da unidade Varginha.

A Professora Helena Maria Gramiscelli Magalhães, por realizar as correções de português do texto.

A todas as pessoas que estiveram e estão na minha vida nesses anos de muita dedicação, por fazerem parte deste momento tão importante.

Parafraseando Prof. Gilberto Teixeira
“Que Deus abençoe a educação.
Estudar é impregnar de sentido a vida.”

Resumo

da Silva, Lázaro Eduardo **Sistema Embarcado Inteligente para Detecção de Intrusão em Subestações de Energia Elétrica Utilizando o Protocolo *Open-Flow***. 167 p. Tese de doutorado – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2016.

O protocolo *International Electrotechnical Commission* (IEC)-61850 tornou possível integrar os equipamentos das subestações de energia elétrica, através de uma rede de comunicação de dados *Ethernet* de alta velocidade. A utilização deste protocolo tem como objetivo principal a interligação dos *Intelligent Electronic Devices* (IEDs) para a automação dos processos no sistema elétrico. As contribuições deste protocolo para a integração do controle e supervisão do sistema elétrico são diversas, porém, o fato de utilizar uma rede de comunicação de dados *Ethernet* integrada expõe o sistema elétrico à ataques cibernéticos. A norma IEC-62351 estabelece uma série de recomendações para prover segurança à rede de comunicação do sistema elétrico, dentre elas, o gerenciamento da rede de comunicação, a análise dos campos da mensagem *Generic Object Oriented Substation Event* (GOOSE) e a utilização de sistemas de detecção de intrusão. O presente trabalho descreve o desenvolvimento de um *Intrusion Detection System* (IDS) que atende os requisitos de segurança propostos pelo protocolo IEC-62351, para a identificação de ataques à comunicação realizada por mensagens GOOSE do protocolo IEC-61850, e entre equipamentos do sistema elétrico. Para o desenvolvimento desta aplicação, foram identificados os campos que compõem as mensagens GOOSE, de forma a reconhecer os valores esperados em diferentes situações de operação do sistema elétrico. Determinaram-se padrões de comportamento a serem utilizados para discernir mensagens falsas na rede de comunicação. Instalou-se e configurou-se um sistema operacional de tempo real embarcado na plataforma de desenvolvimento *Zynq Board* (ZYBO), juntamente com o controlador *Open-Mul*, para gerenciar a rede de comunicação da subestação, através do protocolo *OpenFlow*, realizando otimizações para o tráfego *multicast*. Foi desenvolvido um sistema de detecção

e bloqueio de mensagens GOOSE falsas que utiliza o protocolo *OpenFlow* para controle da rede de comunicação do Sistema Elétrico. Desenvolveu-se ainda um sistema inteligente, utilizando-se uma Rede Neural Artificial (RNA) *Nonlinear Autoregressive Model with Exogenous Input* (NARX), para predição do tráfego realizado por mensagens GOOSE e detecção de ataques *Distributed Deny of Service* (DDOS). Os resultados obtidos demonstraram que o protocolo *OpenFlow* pode ser uma ferramenta interessante para controle da rede, porém, os fabricantes necessitam amadurecer sua implementação nos *switches*, para que sejam utilizados em produção nas redes de comunicação das subestações. O sistema de predição do tráfego gerado por mensagens GOOSE apresentou benefícios interessantes para a segurança da rede de comunicação, demonstrando potencial para compor um sistema de detecção de ataques DDOS realizado por mensagens GOOSE, na rede de comunicação das subestações de energia elétrica.

Palavras-chave: IEC-61850. IEC-62351. Redes de Comunicação de Dados. Segurança Cibernética. Sistema de Detecção de Intrusão. Sistema Elétrico. Sistema Embarcado. Sistema Inteligente.

Abstract

da Silva, Lázaro Eduardo **Embedded Intelligent System for Intrusion Detection in Electric Power Substations Using the OpenFlow Protocol.** 167 p. Ph.D. Thesis – São Carlos School of Engineering, University of São Paulo, 2016.

The IEC-61850 made it possible to integrate equipments of electric power system substations to a high-speed Ethernet data communication network. Its main goal is the interconnection of IEDs for the automation of processes in an electrical system. The contributions of this protocol for the integration of the control and supervision of the electrical system are diverse, although an Ethernet network exposes the electrical system for cyber attacks. The IEC-62351 states a series of recommendations to provide security to the communication network of the electrical system, such as the communication network management, the analysis of GOOSE messages and the use of intrusion detection systems. This study describes the development of an IDS that meets the security requirements proposed by the IEC-62351 standard to identify attacks on communication between GOOSE messages exchanged by electrical equipment using IEC-61850. For the development of this application, fields of the GOOSE messages were identified, in order to recognize the expected values in different power system operating conditions. Behaviour patterns were determined to detect false messages on the communication network. A real-time embedded operating system on ZYBO was installed and configured, as well as the OpenMul controller to manage the communication network of the substation through the OpenFlow protocol, performing optimizations for multicast traffic. A detection system and block tamper GOOSE messages, using the OpenFlow protocol for control of the electrical system communication network, were developed. In addition, an intelligent system using an Artificial Neural Network (ANN) Nonlinear Autoregressive Model with Exogenous Input (NARX) for predicting of the GOOSE messages traffic and the detection of Distributed Deny of Service attack (DDOS) were also developed. The results obtained show that the OpenFlow protocol may be a valuable tool for network control, however,

manufacturers should maturely carry on with its implementation in the switches, so that it be used in substation communication networks. The traffic prediction system of the GOOSE messages presented interesting benefits for the security of the communication network, demonstrating potential to built a DDOS attack detection system performed by GOOSE messages on the communication network of electric power substations.

Keywords: IEC-61850. IEC-62351. Data Communication Network. Cyber Security. Intrusion Detection System. Power System. Embedded System. Intelligent System.

Lista de ilustrações

Figura 1	Modelo ilustrativo de um sistema elétrico de potência.	31
Figura 2	Níveis hierárquicos de um SAS.	43
Figura 3	Exemplo de relação entre funções, nós lógicos e nós físicos.	44
Figura 4	Função de proteção com 3 nós lógicos.	44
Figura 5	Correspondência das camadas do modelo <i>Open Systems Interconnection</i> (OSI) na mensagem GOOSE e seus serviços.	46
Figura 6	Estrutura de cabeçalho da <i>tagging</i> de prioridade <i>Virtual Local Area Network</i> (VLAN).	47
Figura 7	Estrutura de cabeçalho do <i>Ethernet Protocol Data Unit</i> (PDU).	48
Figura 8	Mensagem GOOSE capturada no Wireshark.	50
Figura 9	Tempo de transmissão de eventos.	52
Figura 10	Requisitos de segurança, ameaças, e possíveis ataques	61
Figura 11	Categorias, ataques típicos e medidas de segurança	62
Figura 12	Cenário de atuação do invasor	70
Figura 13	Equipamentos do Laboratório de Sistemas de Energia Elétrica (LSEE).	71
Figura 14	Arranjo laboratorial criado para realização dos <i>pentests</i>	72
Figura 15	Funcionamento da lógica de mudanças de estado em mensagens GOOSE.	72
Figura 16	<i>Hardware</i> ZYBO	75
Figura 17	Sistema de arquivos utilizado no MicroSD para instalação do Sistema Operacional (SO)	76
Figura 18	Comparação entre <i>kernel</i> preemptivo e não preemptivo	78
Figura 19	Configuração do <i>kernel</i> para preempção.	79
Figura 20	Estrutura de funcionamento de um <i>switch</i> tradicional e um <i>switch OpenFlow</i>	80
Figura 21	<i>Switch Openflow</i> e sua comunicação com o controlador	81
Figura 22	Tabela de fluxos do <i>OpenFlow</i> versão 1.0	82

Figura 23	Fluxo de funcionamento do protocolo <i>OpenFlow</i> versão 1.0 no <i>switch</i> .	83
Figura 24	Especificação <i>Abstract Syntax Notation</i> (ASN).1 <i>Basic Encoding Rules</i> (BER) de mensagens GOOSE.	88
Figura 25	Abordagem <i>Type Length Value</i> (TLV) de parâmetros e grupos.	89
Figura 26	<i>Buffer</i> circular GOOSE.	90
Figura 27	Fluxo de verificação da mensagem GOOSE.	91
Figura 28	Novo arranjo laboratorial criado para realização dos <i>pentests</i>	96
Figura 29	Arranjo laboratorial emulado para realização dos <i>pentests</i>	98
Figura 30	NetFPGA	100
Figura 31	<i>Hardware</i> NetFPGA	101
Figura 32	<i>Pipeline Network Field-Programmable Gate Array</i> (NetFPGA)	102
Figura 33	Diagrama de blocos dos componentes da aplicação <i>packet generator</i> . . .	103
Figura 34	Captura de mensagens GOOSE enviadas pelo RTDS.	105
Figura 35	Comunicação NetFPGA computador organizada para realização dos testes.	106
Figura 36	Histograma do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando <i>OpenFlow</i>	108
Figura 37	Tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando <i>OpenFlow</i>	109
Figura 38	Histograma do tempo de tráfego para verificar a segurança das mensagens GOOSE com ajustes no <i>switch OpenFlow</i>	111
Figura 39	Tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando configurações ajustadas do <i>OpenFlow</i>	112
Figura 40	Histograma do tempo de tráfego para mensagens GOOSE autorizadas na tabela de fluxos do <i>switch OpenFlow</i>	114
Figura 41	Ilustração da comunicação NetFPGA plataforma ZYBO organizada para realização dos testes.	115
Figura 42	Histograma do tempo de execução da verificação de segurança das mensagens GOOSE sem a Interferência do <i>Switch</i>	116
Figura 43	Histograma do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando OpenvSwitch.	118
Figura 44	Composição do <i>Perceptron</i> multicamadas (PMC).	127
Figura 45	Medição do tráfego gerado por mensagens GOOSE	128
Figura 46	Topologia de uma rede neural NARX	130
Figura 47	Janelamento realizado para a seleção dos dados de treinamento	131
Figura 48	Passos de treinamento e teste para definição da topologia da RNA . . .	133
Figura 49	Ajuste do Erro Quadrático Médio (EQM) durante o treinamento. . . .	134
Figura 50	Valores desejado e obtido por meio da predição com a RNA NARX. . .	134
Figura 51	Passos de execução da predição.	135

Figura 52	Erro quadrático médio em relação aos passos de predição.	136
Figura 53	Erro percentual médio absoluto em relação aos passos de predição. . .	137
Figura 54	Predição dos valores de tráfego com a RNA NARX.	137
Figura 55	Erro relativo percentual para amostras com eventos no sistema elétrico.	139
Figura 56	Estrutura do ataque DDOS.	141
Figura 57	Erro relativo percentual das amostras com ataques no sistema elétrico.	141
Figura 58	Comportamento do erro relativo percentual para 56 segundos em função dos ataques utilizando mensagens GOOSE.	142
Figura 59	Ilustração das diferenças entre o erro relativo percentual para 56 segundos em função dos ataques utilizando mensagens GOOSE.	143
Figura 60	Ilustração do erro relativo percentual com variações no tráfego de ataque utilizando mensagens GOOSE.	144
Figura 61	Estrutura de execução dos <i>softwares</i> para Detecção de Ataques DDOS.	145

Lista de tabelas

Tabela 1	Tipos de mensagens do padrão IEC-61850.	45
Tabela 2	Suite de aplicativos do OpenMul.	85
Tabela 3	Relação dos valores esperados nos campos da mensagem GOOSE com a situação de funcionamento do Sistema Elétrico (SE).	92
Tabela 4	Testes e Classificação do Controlador <i>OpenFlow</i> para os IEDs.	97
Tabela 5	Testes e Classificação do Controlador <i>OpenFlow</i> para a Rede Emulada.	99
Tabela 6	Parâmetros <i>OpenFlow</i> para o teste.	106
Tabela 7	Resultado estatístico do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando <i>OpenFlow</i>	109
Tabela 8	Resultado estatístico do tempo de tráfego para verificar a segurança das mensagens GOOSE com ajustes no <i>switch OpenFlow</i>	111
Tabela 9	Resultado estatístico do tempo de tráfego para mensagens GOOSE autorizadas na tabela de fluxos do <i>switch OpenFlow</i>	114
Tabela 10	Resultado estatístico do tempo de execução da verificação de segurança das mensagens GOOSE sem a interferência do <i>switch</i>	115
Tabela 11	Resultado estatístico do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando <i>OpenvSwitch</i>	118
Tabela 12	Resultados do teste para definição da topologia da RNA.	133
Tabela 13	Resultado do erro para os passos de predição.	136

Lista de algoritmos

1	Verifica situação normal de operação no SE.	92
2	Verifica situação de evento no SE.	93
3	Verifica os campos que não devem modificar.	94
4	Verifica situação de início ou parada na comunicação.	94
5	Lógica final para análise de segurança nas mensagens GOOSE.	95
6	Coleta do tráfego, preenchimento do banco de dados e criação dos arquivos texto.	146
7	Calcula os passos de predição.	148
8	Calcula o erro relativo percentual.	150
9	Classifica o funcionamento do SE de acordo com os valores do erro relativo percentual.	151

Lista de siglas

APDU *Application Protocol Data Unit*

APPID *Application Identifier*

API *Application Interface*

ARM *Advanced RISC Machine*

ASN *Abstract Syntax Notation*

BER *Basic Encoding Rules*

CPU *Central Processing Unit*

CSV *Comma-Separated Values*

DATA-SET *Conjunto de Dados*

DDR *Double Data Rate*

DOS *Deny of Service*

DDOS *Distributed Deny of Service*

DRAM *Dynamic Random Access Memory*

ELF *Executable Linkable Format*

EQM *Erro Quadrático Médio*

FANN *Fast Artificial Neural Network*

FPGA *Field-Programmable Gate Array*

FSBL *First Stage Boot Loader*

GoCB *GOOSE Control Block*

GOOSE *Generic Object Oriented Substation Event*

GSE *Generic Substation Event*

GSSE *Generic Substation State Events*

ISO *International Organization for Standardization*

IDS *Intrusion Detection System*

ID *Identifier*

IEC *International Electrotechnical Commission*

IED *Intelligent Electronic Device*

IEEE *Institute of Electrical and Electronics Engineers*

IHM *Interface Homem Máquina*

IP *Internet Protocol*

LAN *Local Area Network*

LN *Logical Node*

LSEE *Laboratório de Sistemas de Energia Elétrica*

MAC *Media Access Control*

MAPE *Erro Percentual Médio Absoluto*

MHz *Megahertz*

MPLS *Multiprotocol Label Switching*

MU *Merging Unit*

NARX *Nonlinear Autoregressive Model with Exogenous Input*

ndsCom *Needs Commissioning*

NetFPGA *Network Field-Programmable Gate Array*

NIC *Network Interface Card*

OSI *Open Systems Interconnection*

ONF *Open Networking Foundation*

OVS *Open Virtual Switch*

OXM *OpenFlow Extensible Match*

PCAP *Packet Capture*

PCI *Peripheral Component Interconnect*

PDU *Protocol Data Unit*

PMC *Perceptron multicamadas*

PTP *Precision Time Protocol*

QoS *Qualidade de Serviço*

RAM *Random Access Memory*

RNA *Rede Neural Artificial*

RPROP *Resilient-Propagation*

RTDS *Real Time Digital Simulator*

SAS *Sistema de Automação de Subestação*

SCADA *Supervisory Control and Data Acquisition*

SDN *Software-Defined Networking*

SDRAM *Synchronous Dynamic Random Access Memory*

SE *Sistema Elétrico*

SSL *Secure Socket Layer*

SNMP *Simple Network Management Protocol*

SO *Sistema Operacional*

SQL *Structured Query Language*

SRAM *Static Random Access Memory*

STP *Spanning Tree Protocol*

SV *Sampled Values*

TCI *Tagging Control Information*

TCP *Transmission Control Protocol*

TLV *Type Length Value*

TPID *Tagging Protocol Identifier*

TI *Tecnologia da Informação*

UDP *User Datagram Protocol*

UML *Unified Modeling Language*

USB *Universal Serial Bus*

UTP *Unshielded Twisted Pair*

VLAN *Virtual Local Area Network*

ZYBO *Zynq Board*

Sumário

1	Introdução	31
1.1	Objetivos	34
1.2	Contribuições desta Pesquisa	35
1.3	Organização do Texto	35
2	O Protocolo IEC-61850	39
2.1	Revisão Bibliográfica	39
2.2	Histórico	42
2.3	Funções e Nós Lógicos	42
2.4	Caracterização das Mensagens GOOSE	45
2.5	Formação das Mensagens GOOSE	47
2.6	Identificação da Unidade de Dados das Mensagens GOOSE	49
2.7	Comportamento das Mensagens GOOSE	51
3	Segurança para a Rede de Comunicação das Subestações	53
3.1	Revisão Bibliográfica	53
3.2	Ameaças a Rede de Comunicação das Subestações	56
3.2.1	Ameaças Deliberadas	57
3.2.2	Requisitos de Segurança	59
3.2.3	Tipos de Ataque e Formas de Segurança	60
3.3	Recomendações do Padrão IEC-62351	63
3.4	Tipos de Ataque Pertinentes às Mensagens GOOSE	63
3.5	Prevenção e Detecção de Intrusão	65
4	Sistema de Detecção e Bloqueio de Ataques <i>man-in-the-middle</i>: Abordagem Utilizada, Tempos Envolvidos e Lógica Implementada	67
4.1	Revisão Bibliográfica	67
4.2	Análise de Segurança das Mensagens GOOSE	69

4.2.1	Ataque à IEDs pelo Envio de Mensagens Salvas	73
4.2.2	Ataque à IEDs por Alterações no DATA-SET	73
4.3	Plataforma de Desenvolvimento do Sistema Embarcado	74
4.3.1	Sistema Operacional Linux Embarcado	76
4.4	Redes Definidas por <i>Software Protocolo OpenFlow</i>	80
4.5	<i>Switch OpenFlow</i>	82
4.5.1	Controlador	84
4.6	Detecção de Mensagem Falsa por Análise dos Valores Esperados nos Campos da Mensagem GOOSE	87
4.6.1	Situação Normal de Operação	92
4.6.2	Situação de Evento no SE	93
4.6.3	Situação de Verificação para Ambas as Situações	93
4.6.4	Situação de Início ou Parada da Comunicação	94
4.6.5	Análise da Segurança	95
4.6.6	Validação da Lógica Implementada	95
4.7	Ferramenta de Medição	99
4.7.1	Composição da <i>NetFPGA</i>	100
4.7.2	Funcionamento da <i>NetFPGA</i>	102
4.7.3	Aplicação <i>Packet Generator</i>	103
4.7.4	Criação do Arquivo de Testes	104
4.8	Avaliação do Tempo de Processamento da Detecção e Bloqueio de Ataques <i>Man-in-the-middle</i> Utilizando o Protocolo <i>OpenFlow</i>	104
4.8.1	Ambiente Configurado	105
4.8.2	Medição do Tempo de Tráfego da Aplicação de Segurança Utilizando as Configurações Recomendadas pelo Fabricante do <i>Switch OpenFlow</i>	106
4.8.3	Medição do Tempo de Tráfego da Aplicação de Segurança com Ajustes nas Configurações do <i>Switch OpenFlow</i>	110
4.8.4	Medição do Tempo de Tráfego para Mensagens GOOSE Autorizadas na Tabela de Fluxos do <i>Switch OpenFlow</i>	113
4.8.5	Medição do Tempo de Execução da Aplicação de Segurança sem a Interferência do <i>Switch Openflow</i>	114
4.8.6	Medição do Tempo de Tráfego da Aplicação de Segurança Controlando um <i>Open Virtual Switch</i>	117
4.9	Evolução do Protocolo <i>OpenFlow</i>	119

5 Predição do Tráfego de Rede para Detecção de Ataques DDOS 123

5.1	Revisão Bibliográfica	123
5.2	Redes Neurais Artificiais	125
5.2.1	Redes Perceptron Multicamadas	126

5.3	Definição do Conjunto de Treinamento do Sistema Inteligente	128
5.4	Funcionamento da RNA NARX	129
5.5	Descoberta da Melhor Topologia	130
5.5.1	Validação Cruzada	131
5.6	Predição dos Passos da RNA	135
5.7	Análise do Erro para Situação de Evento no Sistema Elétrico	138
5.8	Análise do Erro para Situação de Ataque DDOS no Sistema Elétrico . . .	140
5.9	Desenvolvimento do Sistema de Detecção de Ataques DDOS no Sistema Elétrico	145
5.9.1	Sistema de Leitura, Preenchimento do Banco de Dados e Criação dos Arquivos da RNA	145
5.9.2	<i>Software</i> para o Cálculo da Predição	148
5.9.3	<i>Software</i> para o Cálculo do Erro Relativo Percentual	149
6	Conclusão	153
	Referências	157

Introdução

O consumo de energia elétrica em residências, comércios, serviços e indústrias, necessita da instalação, manutenção e do controle de um aparato tecnológico que viabilize sua geração, transmissão e distribuição. O contínuo crescimento da demanda de energia elétrica requer um aumento dos recursos despendidos, o que torna este sistema cada vez mais complexo. O conjunto de infraestruturas que o compõem é comumente chamado de Sistema Elétrico (SE) que pode ser ilustrado, conforme Figura 1, onde:

Figura 1 – Modelo ilustrativo de um sistema elétrico de potência.

Fonte: Gaudí (2009).

1. A geração de energia elétrica é realizada por usinas que podem ser hidráulicas, eólicas, térmicas ou termonucleares. Estas usinas utilizam do movimento rotatório de turbinas, resultado da passagem de água, ar ou vapor, acopladas a geradores elétricos para produzirem eletricidade.
2. Uma vez gerada a energia, transformadores elevadores de tensão são utilizados para a transmissão em linhas de alta tensão, é possível transmiti-la a grandes distâncias com pequenas perdas. As linhas de transmissão podem ser terrestres ou aéreas e têm

por objetivo conduzir a energia gerada nas usinas, até as subestações de distribuição, percorrendo vários quilômetros.

3. As subestações de distribuição possuem localização próxima aos centros consumidores. Nela, acontecerá a redução da tensão, de forma que sua distribuição possa ocorrer dentro dos centros urbanos. Neste percurso, linhas de distribuição, geograficamente menores, irão conduzir a energia para abastecer cargas industriais, comerciais, residenciais e de serviços.

Toda esta estrutura está disposta por quilômetros de distância, ficando, por isso, sujeita às intempéries. Para detectar e evitar possíveis falhas, o SE conta com um sistema de proteção, cuja principal tarefa é isolá-las, para que o SE não entre em colapso (COURY; OLESKOVICZ; GIOVANINI, 2007).

Com a evolução da tecnologia digital, os equipamentos que compõem o SE se tornaram digitais, possibilitando a construção de dispositivos multi-funções denominados *Intelligent Electronic Devices* (IEDs) que, entre suas funções, possuem a capacidade de interligação por redes de comunicação.

A utilização de tecnologia digital aliada à capacidade de comunicação viabilizam o desenvolvimento dos Sistemas de Automação de Subestação (SAS) que são utilizados nas concessionárias de energia elétrica para controle, automatização e supervisão dos equipamentos e processos.

Nesse cenário, a comunicação realizada entre IEDs vem passando, nas últimas décadas, por um período de padronização realizada pela norma *International Electrotechnical Commission* (IEC)-61850. O desenvolvimento de aplicações utilizando este protocolo tem evoluído constantemente, impulsionado pelo desejo de automatização de tarefas do SE. O sistema de proteção das subestações tem se tornado cada vez mais robusto, pois, suas decisões são resultado da análise de vários IEDs interligados. Contudo, a utilização de tecnologias de comunicação no SE abre espaço para outro perigo: os ataques cibernéticos (TEN; HONG; LIU, 2011; DZUNG et al., 2005).

A norma IEC-62351 é uma tentativa de ditar padrões de segurança para o ambiente das subestações. Porém, observa-se, na literatura pertinente, que o setor se revela incipiente no que tange à pesquisa. Ataques cibernéticos à infraestrutura de comunicação da subestação têm consequências mais graves do que aqueles feitos às redes empresariais. Neste contexto, a precisão das ferramentas de segurança da informação para o setor elétrico devem ser mais apuradas, chegando a um nível de retidão não aplicado nas redes corporativas.

O presente trabalho propõe-se a investigar formas de detecção de intruso nas redes de comunicação das subestações, no que tange ao tráfego realizado por mensagens *Generic Object Oriented Substation Event* (GOOSE), na *Local Area Network* (LAN). Este tráfego

é restrito à *bays* de transmissão e devem atender aos requisitos de tempo estipulado para a entrega das mensagens, que deve ficar abaixo de 4 milissegundos (CLAVEL et al., 2015).

O desenvolvimento de um *Intrusion Detection System* (IDS) depende de ferramentas de *hardware* e *software* que deverão viabilizar a execução de tarefas para a análise do tráfego de rede, execução de rotinas de verificação, classificação e tomada de decisão do informe sobre uma intrusão. O IDS aqui proposto, tem como especificidade realizar tais tarefas na rede de comunicação da subestação de energia elétrica, para que se detecte ataques pertinentes às mensagens GOOSE do protocolo IEC-61850.

O tráfego gerado pela comunicação entre os equipamentos via mensagens GOOSE é singular. Ferramentas de detecção de intruso utilizadas no tráfego *Transmission Control Protocol* (TCP)/*Internet Protocol* (IP) não estão preparadas para analisar tal tráfego (PREMARATNE et al., 2008). Evidencia-se, portanto, a necessidade de desenvolvimento de uma ferramenta personalizada para este tipo de tráfego.

Conforme enfatizado anteriormente, as redes de comunicação de dados são cruciais para a evolução das redes de energia elétrica, dada a evolução proposta pelas *smart grids*, alavancada pela padronização realizada com o padrão IEC-61850. Segundo McKeown et al. (2008), a comunidade científica tem trabalhado para desenvolver redes de comunicação de dados programáveis, chamadas *Software-Defined Networking* (SDN). Estas redes programáveis controlam *switches* e roteadores que podem processar pacotes para múltiplas redes, simultaneamente. O objetivo destas redes é impulsionar as pesquisas nas redes de comunicação, possibilitando a experimentação de novos protocolos em configurações suficientemente realistas para ganharem a confiança necessária para sua implantação. O *OpenFlow* é a primeira interface de comunicação padrão para SDNs. Ele possibilita a separação da lógica de controle dos equipamentos que compõem a rede do encaminhamento de pacotes por eles realizados. Quando esta lógica de controle é migrada para um ambiente programável, os chamados sistemas operacionais de rede, muito se expandem quanto às possibilidades de implementação de ferramentas que controlam o tráfego da rede, bem como aplicações para controle e segurança da rede de comunicação de dados.

A implantação das SDNs nas subestações de energia elétrica requer cautela, e este protocolo necessita de maturidade para ser utilizado em ambientes com mensagens de tempo crítico. O desenvolvimento do IDS atrelado às SDNs possibilita a identificação e o bloqueio de mensagens falsas na rede de comunicação das subestações. Cabe investigar, se este protocolo está suficientemente apurado para sua utilização neste ambiente que requer alta confiabilidade e disponibilidade.

A análise da formação e comunicação das mensagens GOOSE descreve comportamentos que podem ser observados para a implementação do IDS. Porém, Wu e Banzhaf (2010) apontam que ferramentas tradicionais de detecção de intruso têm falhado na tarefa de proteger as redes de informação, diante da sofisticação dos ataques.

Neste contexto, a presente pesquisa apresenta o desenvolvimento de um IDS embarcado inteligente aplicado à rede de comunicação das subestações de energia elétrica, capaz de analisar as mensagens GOOSE do padrão IEC-61850, para prover segurança à comunicação prioritária entre IEDs do SE. O sistema implementado em um controlador *OpenFlow* analisa as mensagens GOOSE que trafegam pela rede, autorizando-as ou bloqueando-as. O sistema ainda prevê a variação no tráfego de rede das mensagens GOOSE, de forma a detectar ataques do tipo *Distributed Deny of Service* (DDOS), através de uma Rede Neural Artificial (RNA) *Nonlinear Autoregressive Model with Exogenous Input* (NARX).

1.1 Objetivos

Geral: Desenvolver um *Intrusion Detection System* em conformidade com os requisitos de segurança definidos pelo padrão IEC-62351 na identificação de ataques à comunicação realizada por mensagens GOOSE do padrão IEC-61850, entre equipamentos do sistema elétrico.

Para atingir o objetivo geral foram trabalhados os seguintes objetivos específicos:

- ❑ caracterização do estado da arte do protocolo IEC-61850;
- ❑ caracterização do estado da arte de segurança aplicada às subestações;
- ❑ caracterização do estado da arte de sistemas inteligentes aplicados à segurança da informação;
- ❑ caracterização da formação da mensagem GOOSE, bem como seu comportamento;
- ❑ identificação das assinaturas e situações que podem ser utilizadas para caracterização de mensagens legítimas em detrimento aos ataques à segurança;
- ❑ desenvolvimento de sistemas que analisem a mensagem GOOSE e identifique situações indesejadas;
- ❑ análise da plataforma de tempo real que melhor se adapte para realizar a detecção da intrusão;
- ❑ medição do atraso incluído na comunicação realizada pela plataforma de desenvolvimento, para processar as mensagens GOOSE;
- ❑ identificação dos atributos do tráfego e dos equipamentos, importantes para a detecção do intruso;
- ❑ análise de técnicas inteligentes capazes de trabalhar com os atributos disponíveis;
- ❑ implementação de técnicas inteligentes para identificar as tentativas de intrusão, a formação ou a realização de um ataque;

- ❑ embarcar o sistema desenvolvido na plataforma de tempo real escolhida;
- ❑ medir o atraso incluído pelo sistema para identificar um ataque;
- ❑ realizar testes de intrusão pertinentes às mensagens GOOSE.
- ❑ propor soluções de alerta e bloqueio de ataques ao identificá-los;

1.2 Contribuições desta Pesquisa

Com o desenvolvimento do presente trabalho obtiveram-se as seguintes contribuições:

- ❑ identificar vulnerabilidades nos equipamentos e na rede de comunicação das subestações;
- ❑ propor medidas de segurança e ferramentas de proteção para minimizar as vulnerabilidades encontradas;
- ❑ dissecar a mensagem GOOSE e produzir material bibliográfico para viabilizar análises de segurança neste pacote de rede;
- ❑ desenvolver sistemas para análise do conteúdo do pacote GOOSE de forma a realizar controles pertinentes à segurança da comunicação;
- ❑ desenvolver um sistema que identifique e previna ataques do tipo *main-in-the-middle*;
- ❑ desenvolver um sistema inteligente que identifique e previna ataques do tipo DDOS;
- ❑ explorar as funcionalidades da plataforma *Zynq Board* (ZYBO);
- ❑ desenvolver sistemas utilizando plataformas de tempo real;
- ❑ desenvolver um IDS inteligente utilizando a plataforma ZYBO.

1.3 Organização do Texto

Vale apontar que optou-se por fazer uma revisão bibliográfica em cada um dos capítulos desenvolvidos nesta tese, para melhor embasamento das questões abordadas.

O Capítulo 1, a Introdução, expõe o contexto em que o trabalho se insere, seus objetivos, as lacunas encontradas nos sistemas de segurança atuais, as ferramentas utilizadas para o desenvolvimento desta pesquisa, as contribuições do trabalho para a sociedade do conhecimento e a estrutura organizacional do texto.

O Capítulo 2 aborda o protocolo IEC-61850, o qual padroniza a comunicação de dados realizada entre equipamentos do sistemas elétrico. O capítulo se organizou de forma a

apresentar os conceitos necessários para a compreensão da comunicação realizada pelos equipamentos do SE, utilizando mensagens GOOSE do padrão IEC-61850.

Neste capítulo, é realizada, inicialmente, uma revisão bibliográfica dos trabalhos pertinentes à compreensão do padrão do ponto de vista de implementação. Posteriormente, é ressaltada, ainda na revisão bibliográfica a necessidade de avaliação de desempenho da rede de comunicação da subestação. Em seguida são apresentadas implementações de proteção, controle e supervisão utilizando este protocolo. Finalizando a revisão bibliográfica, discutem-se trabalhos que destacam a importância do IEC-61850 para as *Smart Grids*.

O tópico seguinte relata o histórico de criação do padrão, seguido de duas subestações que explanam sobre sua modelagem e seu funcionamento, integrando os equipamentos da subestação de energia elétrica. A seguir, é descrita a atuação das mensagens GOOSE no protocolo IEC-61850, seguida de um relato sobre a formação desta mensagem, o significado e a funcionalidade de cada campo que a compõem, e uma explicação sobre a propagação desta mensagem na rede de comunicação.

O Capítulo 3 trata de segurança para a rede de comunicação das subestações. O início da revisão bibliográfica deste capítulo é formado por artigos que destacam a importância de se proteger a rede de comunicação das subestações, seguida por contínuas, aplicações voltadas para à segurança da infraestrutura de comunicação. Comparações com ferramentas de segurança da informação utilizadas no ambiente empresarial são realizadas. Porém, resalta-se, no capítulo, a especificidade das ameaças à IEDs, e justificando a necessidade de desenvolvimento de novas ferramentas de proteção à informação voltadas para o ambiente da subestação. Em destaque, são apresentadas aplicações de detecção de intrusão.

Ainda, a revisão bibliográfica inclui a análise de *performance* de ferramentas de segurança, ressaltando as mensagens de tempo crítico utilizadas no protocolo IEC-61850.

Findada a revisão bibliográfica, o assunto seguinte está relacionado às ameaças pertinentes à rede de comunicação das subestações segundo o padrão IEC-62351. O tópico subdivide-se em ameaças deliberadas, requisitos de segurança, tipos de ataque e formas de proteção. Estes aspectos têm como função apresentar conceitos que embasarão a construção do IDS. Posteriormente, são apresentadas recomendações de segurança realizadas pela norma IEC-62351. De acordo com as características da rede das subestações, apresentadas no Capítulo 2 e as medidas de segurança descritas pelo padrão IEC-62351, foram elencados alguns ataques pertinentes às mensagens GOOSE. Por fim, são descritos alguns tipos de IDS, sua filosofia de funcionamento e a tarefa que desempenha na rede de comunicação.

O Capítulo 4 descreve o sistema de detecção e bloqueio de ataques *man-in-the-middle*. O capítulo é iniciado com uma revisão bibliográfica que aborda o uso de mensagens de tempo crítico nas redes de comunicação de dados das subestações, a necessidade de

desenvolvimento de *hardware* específico para o processamento dessas mensagens, para que se obtenha baixa latência no seu tráfego. Posteriormente, os artigos apresentados descrevem a implementação de *hardwares* específicos nas subestações de energia elétrica. Seguindo, são descritos trabalhos sobre o protocolo *OpenFlow*, as vantagens de se usar este protocolo e sua aplicação nas redes de comunicação das subestações, com destaque para seus benefícios aplicados às *Smart Grids*.

Após a revisão bibliográfica, são apresentadas as vulnerabilidades das redes de comunicação das subestações com a realização de *pentests*.

Neste contexto, observa-se a necessidade de sistemas de detecção de intrusão nas redes das subestações. A plataforma de desenvolvimento de sistemas embarcados ZYBO é apresentada, demonstrando a relevância de seus recursos para o desenvolvimento deste trabalho.

Uma descrição do protocolo *OpenFlow* também é realizada, na qual se expõem suas características, funcionalidades e seus controladores. Nesta parte, destaca-se o controlador *OpenMul*, utilizado para o desenvolvimento deste trabalho. Em seguida, é relatada a lógica de detecção de intruso, didaticamente dividida, é feita sua validação, através dos *pentests* realizados, findando com análises do tempo de tráfego, para verificar a segurança nas mensagens GOOSE em diferentes cenários.

O Capítulo 5 descreve técnicas de inteligência artificial utilizadas na literatura pesquisada para a detecção de intruso. O capítulo inicia-se, também, com uma revisão bibliográfica que descreve ataques DDOS na rede de comunicação do Sistema Elétrico, bem como trabalhos que utilizam a RNA NARX, para previsão de séries temporais. O tópico seguinte contempla a descrição das principais características das RNAs, para a provisão de séries temporais. Finalmente, são apresentados os procedimentos para a definição da topologia da RNA implementada. Procede-se, a seguir, à descrição do sistema inteligente, com base no tráfego gerado pela mensagem GOOSE na rede de comunicação, além de diversas análises do erro entre as amostras, para que se diferencie o erro de um evento no sistema elétrico, do erro causado por um ataque DDOS.

No Capítulo 6, são apresentadas as conclusões deste trabalho baseado na revisão bibliográfica, nos testes e nas implementações realizadas, nos resultados obtidos, bem como sugerem-se trabalhos para futuras investigações.

O Protocolo IEC-61850

O emprego do protocolo *International Electrotechnical Commission* (IEC)-61850, nas redes de comunicação das subestações de energia elétrica, tornou possível integrar os *Intelligent Electronic Devices* (IEDs), utilizando redes *Ethernet*. O anseio por aplicações que automatizam os processos, possibilitando controle, supervisão e monitoramento remoto dos equipamentos, tem impulsionado a realização de pesquisas sobre este protocolo. As mensagens *Generic Object Oriented Substation Event* (GOOSE) compõem o modelo *Generic Substation Event* (GSE) que provê o serviço de distribuição rápida e confiável dos dados de entrada e saída. Elas são classificadas como mensagens rápidas ou de abertura (*trip*). A detecção de ataques a este tipo de mensagem é o alvo da presente pesquisa. Para a compreensão do funcionamento desta comunicação em nível de implementação, são descritos, no presente capítulo, os conceitos fundamentais para caracterizar uma comunicação distribuída em IEC-61850, utilizando mensagens GOOSE, bem como, toda a composição destas mensagens.

2.1 Revisão Bibliográfica

A padronização de uma linguagem de comunicação entre IEDs é a chave para o avanço da conectividade e interoperabilidade nos Sistema de Automação de Subestação (SAS) (OZANSOY; ZAYEGH; KALAM, 2009b). O padrão IEC-61850 possibilitou integrar IEDs, através de uma rede *Ethernet* de alta velocidade (OZANSOY; ZAYEGH; KALAM, 2009a) (OZANSOY; ZAYEGH; KALAM, 2007). Vislumbrar a norma IEC-61850, sob o ponto de vista de implementação, conforme descrito pelos autores supracitados, constitui uma respeitável contribuição para a presente pesquisa. Tais autores caracterizam um conjunto dos modelos de dados orientados à objeto utilizados para descrever os processos implementados e controlados pela subestação. São apresentados modelos de serviço utilizados nas interações entre os dispositivos da subestação que demonstram a transferência de dados entre IEDs, e propõem a padronização dos dados do IED, usando objetos de dados referidos como nós lógicos. Tal fato possibilita o compartilhamento de informações e

comandos, em uma rede, independente da implementação.

A apreciação destes artigos, juntamente com a norma IEC61850 (2013), oferece uma visão abrangente do padrão, além de abordar o funcionamento do protocolo, demonstrando como se realiza a definição dos dados dos IEDs e da comunicação entre os dispositivos das subestações. Kostic, Preiss e Frei (2005), acrescentam, ainda, que a representação do modelo em *Unified Modeling Language* (UML) torna a complexidade inerente ao padrão de dados viável para a compreensão de humanos e máquinas. Outro aspecto importante é que o modelo permite estabelecer e manter a consistência entre os dados da norma e os modelos de comunicação. O estudo destes artigos e normas contribuem para a compreensão do protocolo, possibilitando a implementação de sistemas que interagem com mensagens do padrão IEC-61850.

Sevov, Zhao e Voloh (2013), Ransom e Chelmecki (2014), Dixon et al. (2014) descrevem a implementação de esquemas de proteção que se comunicam, através de mensagens GOOSE do padrão IEC-61850. Entre as vantagens de tal utilização, podem-se citar: a redução da fiação; a transmissão e recepção de valores digitais e analógicos, através da rede Ethernet; a utilização de funções lógicas programáveis, proporcionando flexibilidade na implementação de esquemas de proteção e controle. Medições de latência da mensagem GOOSE em arranjos de proteção foram realizadas, em trechos nos quais os autores narram que o seu desempenho é similar às fiações de cobre.

Sidhu e Yin (2007) narram em seu estudo sobre o desempenho de rede, que o sistema de comunicação de subestações IEC-61850 baseado em *Ethernet* tem mostrado desempenho suficiente para atender mensagens de tempo crítico. No entanto, IEDs e comutadores com a capacidade de fornecer codificação de prioridade devem ser utilizados para as redes com tráfego pesado. Em sistemas de automação de subestações reais, é importante avaliar o atraso das mensagens dentro da rede, para que o desempenho especificado na norma IEC-61850 possa ser conseguido pela escolha da topologia adequada.

Zadeh, Sidhu e Klimek (2011) discutem dois desafios principais a serem enfrentados, para se realizar a função de proteção de um ambiente de barramento, através da rede de comunicação *Ethernet*: a perda de pacotes e os atrasos no algoritmo de retransmissão. No entanto, foi observado, através de testes que consideram o impacto do tráfego da rede, que aquela perda de pacotes e aqueles atrasos não acarretam a quebra da confiabilidade e da segurança do IED. O atraso máximo observado, em decorrência da comunicação, foi de 1,3 ms.

O padrão IEC-61850 compõe, também, as *Smart Grids* (NGO; YANG, 2016), redes de energia elétrica que integram de forma inteligente as ações de produtores e consumidores a ela conectados, desempenhando importante função na agregação dos dados para supervisão, controle e proteção. Sučić et al. (2011) relatam que um pré-requisito para se concretizar a visão de *Smart Grids* é a integração de dispositivos com interfaces de comunicação padronizadas. Utilizando a capacidade de modelagem do padrão IEC-61850,

foi moldada uma usina virtual genérica. Este modelo demonstra que o IEC-61850 fornece dados para o controle de recursos energéticos distribuídos. Yoo et al. (2011) completam que, à medida que as redes de energia são integradas formando as *Smart Grids*, o protocolo de comunicação desempenha um papel fundamental nas operações bem sucedidas. A implantação de interoperabilidade nas redes inteligentes é um obstáculo que precisa ser superado. O protocolo IEC-61850 é o padrão para se obter a interoperabilidade em automação na concessionária de energia. Colet-Subirachs et al. (2012) descrevem a criação de um *gateway* para monitoramento de uma rede de energia elétrica que pode ser implantado para controle de recursos energéticos distribuídos.

Giustina et al. (2015) afirmam que as redes de distribuição da energia elétrica são uma infraestrutura crítica da sociedade moderna. Sua continuidade é um tema chave que necessita de automatização para melhorar a qualidade do serviço. No referido artigo, foi realizada uma caracterização experimental com o uso do IEC-61850 para automação de *Smart Grids*.

A acelerada evolução do Sistema Elétrico (SE) tem estimulado o desenvolvimento de abordagens para o tratamento dos problemas causados pelo aumento da demanda de energia. A proteção, supervisão e o controle da rede de energia elétrica têm migrado para uma Intranet de utilidade comum (HOPKINSON et al., 2009). Esta transição permite, por exemplo, gestão da demanda de energia elétrica e uma aplicação desejada pelas *Smart Grids* (PALENSKY; DIETRICH, 2011). Para a integração das redes de energia elétrica, a comunicação desempenha papel chave na operação confiável e estável do SE.

A contribuição dada pelos trabalhos citados são cruciais para se entender a importância do protocolo IEC-61850, bem como para se analisarem as mudanças causadas no SE devidas à sua adoção. O desejo por aplicações que automatizam as tarefas realizadas nas subestações de energia, onde vários equipamentos se tornaram inteligentes, realizando diversas tarefas, sem a intervenção humana, impulsionaram o uso deste protocolo. Mais recentemente, com o surgimento das *Smart Grids* consideradas redes de energia de nova geração (FANG et al., 2012), compostas por sistemas que utilizam recursos de tecnologia da informação com elevado grau de automação, cresce a necessidade de protocolos de comunicação que irão interligar os equipamentos inteligentes da rede elétrica. Estas redes de comunicação que interligam os equipamentos viabilizaram a execução de operações de forma distribuída, nas quais a tomada de decisão é cada vez mais robusta, pois é resultado da análise dos dados de situações que diversos equipamentos estão a transcorrer.

O presente capítulo faz um breve histórico do protocolo IEC-61850, descrevendo seus principais recursos, com enfoque nas mensagens GOOSE utilizadas durante o desenvolvimento deste trabalho.

2.2 Histórico

Em 1994, um grupo de trabalho de “Interfaces de Controle e Proteção de Subestações”, do Comitê Técnico do IEC, propôs a padronização da comunicação em Sistema de Automação de Subestação (SAS). O objetivo da normalização era desenvolver um padrão de comunicação que atendesse os requisitos funcionais e de desempenho da subestação, apoiado na evolução tecnológica, além de especificar requisitos para alcançar a interoperabilidade e a intercambialidade entre os IEDs de diferentes fornecedores (IEC61850, 2003a).

Em abril de 2003, foram publicadas as normas que estabeleciam um padrão de comunicação entre IEDs denominado IEC-61850. Este padrão propunha a comunicação entre os IEDs, através de uma rede de comunicação de dados que ocorria por fiação elétrica. A estratégia utilizada para a elaboração da norma foi a mesclagem dos pontos fortes de três métodos: decomposição funcional, fluxo de dados e modelagem de informações, a saber:

- ❑ A decomposição funcional é usada para explicitar a relação lógica entre os componentes de uma função distribuída. Elas são apresentadas em termos de *Logical Nodes* (LNs) que descrevem as funções, subfunções e interfaces funcionais.
- ❑ O fluxo de dados é usado para demonstrar as interfaces de comunicação que devem apoiar o intercâmbio de informações entre componentes distribuídos e de requisitos de desempenho funcional.
- ❑ A modelagem de informações é utilizada para definir a sintaxe e a semântica abstrata da troca de informações. Elas são representadas em termos de classes de objetos de dados e tipos, atributos, métodos de objetos abstratos (serviços), e seus relacionamentos (IEC61850, 2003a).

2.3 Funções e Nós Lógicos

A alocação de funções e níveis de controle para os IEDs não é fixa. Ela depende de requisitos de disponibilidade, performance, restrições de custo, estado da arte da tecnologia e filosofia dos equipamentos. Todavia, o padrão deve permitir a livre alocação de funções para os IEDs, mantendo a interoperabilidade entre as funções a serem executadas em uma subestação, porém, residentes em equipamentos de diferentes fornecedores. As funções podem ser divididas em partes a serem realizadas em diferentes IEDs que se comunicam uns com os outros (função distribuída). Portanto, o comportamento da comunicação dos LNs deve suportar a interoperabilidade solicitada pelos IEDs. As funções de um SAS são controle e supervisão, bem como, a proteção e o monitoramento dos equipamentos do SE (IEC61850, 2003a). Suas funções podem ser divididas em três níveis hierárquicos, conforme ilustra a Figura 2, a seguir.

Figura 2 – Níveis hierárquicos de um SAS.

Fonte: IEC61850 (2003a).

No nível de processo, disjuntores e chaves seccionadoras, entre outros, recebem comandos e enviam a situação dos seus estados para dispositivos do SE. Os transformadores de corrente e potencial fazem leituras das grandezas do SE, convertendo-as em valores plausíveis para uso em níveis acima. No nível de *bay*, estão as unidades de controle e proteção responsáveis por todo o aparato lógico da proteção e do controle do SE. Cabe observar, que este processo muitas vezes é realizado de maneira conjunta entre os dispositivos que estarão trocando informações, através do meio de comunicação. No nível de estação, estão os *softwares* de controle e configuração, banco de dados, e os sistemas supervisórios, interligados por meios de comunicação no nível de *bay*. As unidades deste nível são importantes para a aquisição e o processamento de dados do SE. Para atingir as metas de normalização, as funções do SAS foram identificadas e divididas em LNs. Os LNs podem estar em diferentes níveis ou dispositivos. A Figura 3, ilustra algumas relações entre as funções lógicas, os nós lógicos, e nós físicos (dispositivos).

Uma função pode ser classificada como distribuída, quando for executada por dois ou mais LNs, que estão em dispositivos físicos diferentes. As funções consistem no mínimo de três LNs: o núcleo da funcionalidade, o LN de processo e o da Interface Homem Máquina (IHM) (IEC61850, 2003b).

A Figura 4, exemplifica uma função de proteção com três LNs.

O nó lógico IHM é a interface do operador, o nó lógico P.. é a proteção, e o nó lógico XCBR é o disjuntor que receberá a mensagem de abertura (*trip*). Estes nós lógicos residem em três dispositivos físicos distintos: computador, relé e interface de processo remoto,

Nós Lógicos	[----- Funções -----]			Dispositivos físico
	Comutação de Disjuntor Sincronizado	Proteção de distância	Proteção de sobrecorrente	
IHM	X	X	X	Computador
Comutação de sincronia	X			Dispositivo de comutação sincronizado
Proteção de Distância		X		Proteção de distância com unidade de sobrecorrente integrada
O/C Proteção			X	
Disjuntor	X	X	X	Unidade de controle do bay
Bay CT		X	X	Unidade de controle do bay
Bay VT	X	X		Transformadores de corrente e tensão
BB VT	X			
				Transformador de tensão no barramento

Figura 3 – Exemplo de relação entre funções, nós lógicos e nós físicos.

Fonte: IEC61850 (2003b).

Figura 4 – Função de proteção com 3 nós lógicos.

Fonte: IEC61850 (2003b).

respectivamente. As *Logical Connection (LC)* são conexões lógicas e as (Serial) *Interfaces (IFs)* constituem as interfaces seriais de comunicação entre os dispositivos físicos, onde

IF1 informa que, nesta interface, existe tráfego de dados de proteção entre dispositivos dos níveis de *bay* e de estação. IF4 informa que, nesta interface de comunicação, trafegam valores instantâneos dos transformadores de tensão e corrente entre os níveis de processo e de *bay*. IF5 informa que, nesta interface de comunicação, trafegam dados de controle entre os níveis de processo e de *bay*. O significado de cada interface de comunicação do SAS pode ser observado em IEC61850 (2003b);

2.4 Caracterização das Mensagens GOOSE

O padrão IEC-61850 define perfis de implementação de serviços, utilizando as camadas do modelo *Open Systems Interconnection* (OSI). O modelo de referência OSI (*International Organization for Standardization* (ISO)/IEC 7498-1) é o padrão de normatização em redes de comunicação de dados. O propósito deste modelo é facilitar a comunicação entre sistemas distintos, sem a necessidade de realizar mudanças na lógica do *hardware* e *software* de cada um deles (FOROUZAN, 2008). Ele é formado por sete camadas, e cada uma delas define uma parte do processo de transferência de informações através da rede. Este modelo detalha requisitos funcionais de cada camada para obtenção de um sistema de comunicação robusto. Apesar desta definição, o modelo não especifica os protocolos a serem utilizados para atingir a funcionalidade, nem restringe a solução para um conjunto único de protocolos.

Dentre os perfis de implementação dos serviços executados pelas camadas do modelo OSI desenvolvidos no padrão IEC-61850, estão os serviços providos pelas mensagens GOOSE, pertencentes à classe GSE. O modelo GSE prevê a possibilidade de uma distribuição rápida e confiável dos dados de entrada e saída. Este modelo é baseado no conceito de descentralização autônoma, fornecendo um método eficiente de entrega simultânea das informações para mais de um dispositivo físico, através de serviços *multicast* e *broadcast* (IEC61850, 2004). Esta mensagem GOOSE possibilita a troca de dados organizados por um Conjunto de Dados (DATA-SET) e se classifica como mensagem rápida, de abertura (*trip*), ou de prioridade na rede, como se pode observar na Tabela 1, a seguir.

Tabela 1 – Tipos de mensagens do padrão IEC-61850.

Tipo	Classe de Desempenho	Serviço
1	Mensagem rápida	GOOSE/GSE
1A	<i>Trip</i>	GOOSE/GSE
2	Mensagem de média velocidade	Cliente/Servidor
3	Mensagens lentas	Cliente/Servidor
4	Dados em rajada (<i>Raw Data</i>)	Transmissão de valores amostrados
5	Funções de transferência de arquivos	Cliente/Servidor
6	Mensagens de sincronismo de tempo	<i>Time Sync</i>

Fonte: IEC61850 (2004).

GOOSE é a mensagem rápida mais importante na subestação, portanto possui requisitos mais exigentes em relação às outras mensagens rápidas. O mesmo desempenho pode ser solicitado para intertravamento, *intertrips* e lógica discreta entre funções de proteção. Para *bays* de distribuição, o tempo total de transmissão deve ser de, no máximo, 10 ms. Para *bays* de transmissão, este tempo deve ser menor que 4 ms.

Mensagens do tipo 1A são mapeadas por pacotes *Ethernet* simplificados, para otimizar a decodificação na recepção das mensagens. O padrão IEC-61850 baseou-se na comunicação realizada pelas redes *Ethernet*, a fim de definir os serviços de suas mensagens. A Figura 5, a seguir, relaciona as camadas definidas pelo modelo OSI com a formação da mensagem GOOSE do padrão IEC-61850, e descreve os serviços providos pelas camadas da mensagem GOOSE.

Figura 5 – Correspondência das camadas do modelo OSI na mensagem GOOSE e seus serviços.

Fonte: IEC61850 (2004).

A camada física da mensagem GOOSE aceita os padrões: 10Base-T que correspondem ao padrão de redes com transmissão digital a velocidades de 10 Megabits por segundo, (Mbps) utilizando cabeamento par trançado; 100Base-T que são redes com transmissão digital a velocidades de 100Mbps, utilizando cabeamento par trançado; e 100Base-FX que constituem redes com transmissão digital a velocidades de 100Mbps que utilizam o cabeamento de fibra óptica.

A camada de enlace de dados utiliza a mesma disciplina de acesso ao meio do *Ethernet*, o *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD). Ela também inclui uma *tagging* de acordo com o padrão do *Institute of Electrical and Electronics Engineers*(IEEE) 802.1Q, que é usado para separar mensagens de tempo crítico e alta prioridade das aplicações de baixa prioridade. Esta *tagging* viabiliza a configuração da *Virtual Local Area Network* (VLANs) que possibilita a identificação do tráfego dos IEDs, permitindo a criação de grupos de trabalhos virtuais, isolando a rede dos equipamentos. As mensagens GOOSE utilizam a faixa de endereços *Media Access Control* (MAC)¹ de 01:0C:CD:01:00:00 a 01:0C:CD:01:01:FF, para se propagarem *multicasts* na rede. Visando

¹Endereço físico associado à interface de comunicação (FOROUZAN, 2008)

atender a requisitos de velocidade e tempo de transmissão, os serviços das camadas de rede, transporte e sessão, não foram utilizados neste tipo de mensagem. Desta forma, estas camadas não compõem a formação do pacote GOOSE.

A camada de apresentação desempenha o papel de tradução do formato dos dados utilizado pela camada de aplicação, em um formato comum a ser utilizado na transmissão, e vice versa. Neste caso, ela utiliza a notação *Abstract Syntax Notation* (ASN).¹ definida pela ISO/IEC 8824-1 e ISO/IEC 8825-1, para definir a sintaxe de informação dos dados. Esta notação elenca um número de tipos de dados simples e especifica uma notação para referenciar esses tipos e para especificar valores para os tipos (IEC61850, 2004).

A camada de aplicação possui um conjunto de dados DATA-SETs utilizado para o tráfego de informações de interesse dos IEDs, como, por exemplo, a mudança do estado de um equipamento e o instante de sua alteração. Desta forma, as mensagens GOOSE são caracterizadas na rede como sendo o tipo de mensagem mais utilizada do padrão IEC-61850.

2.5 Formação das Mensagens GOOSE

A mensagem GOOSE possui algumas especificidades na sua formação para prover os serviços necessários ao tráfego de dados do padrão IEC-61850. Além de utilizar os serviços das camadas física e de enlace, tal como supracitado, cabe analisar a formação da *tagging* de prioridade VLAN. A Figura 6 ilustra a formação do cabeçalho da VLAN.

Octets		8	7	6	5	4	3	2	1
12	TPID	0X8100							
13									
14	TCI	User priority			CFI		VID		
15		VID							

Figura 6 – Estrutura de cabeçalho da *tagging* de prioridade VLAN.

Fonte: IEC61850 (2004).

O cabeçalho é formado pelos campos *Tagging Protocol Identifier* (TPID) e *Tagging Control Information* (TCI). O campo TPID indica o tipo do *Ethernet* atribuído a quadros 802.1Q. Este campo ocupa 2 *bytes* do cabeçalho e o seu valor deve ser 0x8100. O campo TCI ocupa os outros 2 *bytes* do cabeçalho, e é subdividido em:

- *User priority*: utilizado para indicar a prioridade dos pacotes. Este campo pode assumir valores entre 1 e 7 o qual, em comunicação de dados que façam uso do padrão IEC-61850, é utilizado para separar valores amostrados e mensagens GOOSE de pacotes de baixa prioridade. Pacotes de alta prioridade devem ter valores entre 4 e 7, e de baixa prioridade devem ter prioridade de 1 a 3. O valor 1 de prioridade

é o padrão para pacotes que não pertencem a nenhuma VLAN, e o valor 0 deve ser evitado, pois pode causar atraso ao tráfego normal. O valor padrão de prioridade para mensagens GOOSE é 4.

- ❑ *Canonical Format Identifier (CFI)*: é utilizado para compatibilidade entre redes *Ethernet* e *Token Ring*. Este campo recebe o valor zero para o pacote trafegar nos *switches Ethernet*.

O *VLAN Identifier (VID)* tem 12 *bits* que podem assumir valores entre 1 e 4095. Se o VID não for utilizado, seu valor deve ser definido como zero. As mensagens GOOSE, geralmente, estão alocadas em um VID diferente do tráfego da rede *Local Area Network (LAN)* da subestação de energia elétrica. Assim, o tráfego GOOSE fica separado do tráfego restante da rede local.

Findada a *tagging* da VLAN, inicia-se a caracterização do pacote GOOSE camada de apresentação. A Figura 7 ilustra a estrutura de cabeçalho do *Ethernet Protocol Data Unit (PDU)*.

Octets		8	7	6	5	4	3	2	1
16	Ether- type PDU	Ethertype							
17									
18		APPID							
19									
20		Length							
21									
22		Reserved 1							
23									
24		Reserved 2							
25									
26	...	APDU							
m+26									

Figura 7 – Estrutura de cabeçalho do *Ethernet PDU*.

Fonte: IEC61850 (2004).

O *Ethernet PDU* é baseado no ISO/IEC 8802-3 subcamada MAC e registrada pelo *Institute of Electrical and Electronics Engineers (IEEE)*. O primeiro campo *Ethertype* é registrado com o valor 0x88B8, que identifica a mensagem GOOSE.

O segundo campo intitulado *Application Identifier (APPID)* é um atributo que permite padronizar um sistema de identificação exclusivo para o aplicativo que está emitindo a mensagem GOOSE (IEC61850, 2003c). O valor do APPID é uma combinação do seu tipo, definido como os dois *bits* mais significativos do *Ethertype* e o seu identificador atual. A faixa de valor reservada para GOOSE é entre 0x0000 e 0x3FFF. Caso não seja configurada,

seu valor padrão será 0x0000. A fonte geradora do GOOSE deve ter um único APPID. A administração do GSE deve ter o mesmo APPID emitido pelo GOOSE onde as solicitações de gestão GSE estão sendo emitidas. O APPID de resposta às solicitações de gestão deve ser o mesmo que o do pedido (IEC61850, 2004).

Length é o número de octetos incluindo o cabeçalho *Ethernet* PDU, começando do APPID, até o comprimento da *Application Protocol Data Unit* (APDU). Seu valor deve ser de $m + 8$, no qual m é o comprimento do APDU, e m é inferior a 1492. Quadros com campo *length* inválido ou inconsistente devem ser descartados (IEC61850, 2004).

Os campos *Reserved1* e *Reserved2* são destinados ao uso futuro de aplicações da norma e devem assumir o valor 0 como padrão.

O APDU é definido de acordo com a gramática especificada para GOOSE e administração do GSE definida em ASN.1 (IEC61850, 2004).

2.6 Identificação da Unidade de Dados das Mensagens GOOSE

Para o estudo prático da unidade de dados que forma as mensagens GOOSE, foi instalado, na rede LAN do Laboratório de Sistemas de Energia Elétrica (LSEE), um IED configurado para publicar mensagens GOOSE. Nesta mesma rede, foi interligado um computador capaz de executar o *software* analisador de protocolos *Wireshark* (COMBS, 1998). Este *software* coloca a placa de rede do computador em modo promíscuo, o que causa a captura de todas as mensagens que passam por ela, até mesmo aquelas que não tiverem como destino o computador instalado. A Figura 8 expõe a tela do *Wireshark* onde é possível observar os cabeçalhos da VLAN, do *Ethertype* PDU e do APDU GOOSE.

O APDU da mensagem GOOSE é nomeado *goosePdu*. Seu primeiro campo é definido como *GOOSE Control Block* (GoCB)Ref que deve ser o caminho único para uma instância GoCB dentro do LLN0. A referência ao objeto GoCBRef deve ser formada pelo nome do dispositivo lógico, nó lógico 0, e nome do bloco de controle GOOSE, exemplificado por *LDName/LLN0.GoCBName*.

O *timeAllowedtoLive* representa o tempo de vida da última mensagem enviada. Sua unidade está em milissegundos, tempo que deve ser menor ou igual a 60 segundos. Cada mensagem transmitida carrega este parâmetro que informa ao receptor o tempo máximo que ele deve esperar até a próxima retransmissão. Se uma nova mensagem não for recebida dentro daquele intervalo de tempo, o receptor assume que a associação feita com o emissor foi perdida (IEC61850, 2004). Os intervalos específicos utilizados pelo emissor da mensagem GOOSE são questões locais definidas no equipamento. O parâmetro *timeAllowedtoLive* informa aos receptores quanto tempo ele deve esperar.

O valor do *DatSet* deve seguir a mesma estrutura para definição do nome realizada no GoCB especificado pelo GoCBRef.

Figura 8 – Mensagem GOOSE capturada no Wireshark.

O *GoID*, assim como o *DatSet* é associado ao GoCB especificado pelo GoCBRef.

O campo *T* é um valor que estampa o tempo da ocorrência de uma mudança de estado.

O *stNum* é um valor inteiro que pode variar entre 1 e 4.294.967.295. Este campo conta as mudanças de estado no envio de mensagens GOOSE.

O *sqNum* é um valor inteiro com o mesmo tamanho do *stNum*. No *sqNum*, o valor zero é reservado para indicar a primeira transmissão de uma mudança em *stNum*. Este campo conta as replicações das mensagens GOOSE. Quando ocorre uma mudança de estado, ele reinicia sua contagem, iniciando pelo valor 0 (IEC61850, 2003c). Chegando ao último valor de 4.294.967.295 assumido por este campo, ele retorna a 1.

O campo *test* determina se a informação é gerada por uma operação normal do SE, ou por um teste (IEC61850, 2003c).

O parâmetro *confRev* contém o número de revisões da configuração do GoCB, no momento em que a mensagem GOOSE foi enviada. Pode assumir valores inteiros entre 0 e 4.294.967.295.

O *Needs Commissioning* (ndsCom) deve ter o valor *true*, se o atributo *DatSet* tem o valor *NULL*. Ele deve ser utilizado para indicar que o GoCB requer configuração adicional (IEC61850, 2003c).

O campo *numDataSetEntries* especifica o número de membros do *DatSet*. O parâmetro *allData* possui a lista dos dados transmitidos pela mensagem GOOSE. As informações contidas neste campo são padronizadas como uma sequência de dados definida pela ISO-IEC-9506-2 IEC61850 (2004), que podem ser o *bit-string*, um parâmetro que especifica a qualidade do dado transmitido. Ele tem 2 *bytes* de tamanho e pode assumir valores entre 0 e 15. Seus campos são formados por *paddings* que completam os *bits* não definidos e o *bit-string* propriamente dito. Maiores informações sobre os valores que o *bit-string* pode assumir são encontradas em IEC61850 (2004). Além do valor de qualidade, a mensagem GOOSE pode assumir valores *boolean* ou *floating-point* com tamanho 1 e 5 *bytes*, respectivamente. Com estes dados, o pacote GOOSE está totalmente formado.

2.7 Comportamento das Mensagens GOOSE

O serviço GOOSE modelado conforme descrição realizada por IEC61850 (2003c) prevê a possibilidade de um sistema de distribuição rápido e confiável dos dados de entrada e saída. O modelo de eventos da subestação é baseado no conceito de descentralização autônoma, fornecendo um método que permite a entrega simultânea da mesma informação do evento para mais de um dispositivo físico, através da utilização de serviços *multicast* e *broadcast*. Para o modelo genérico de eventos da subestação, os valores são transmitidos do ponto de vista do dispositivo lógico de comunicação. O modelo genérico de evento da subestação aplica-se à troca de valores de *DataAttribute*. Duas classes de controle e a estrutura das duas mensagens são definidas:

- GOOSE permite a troca de dados comuns organizados por um DATA-SET.
- *Generic Substation State Events* (GSSE) oferece a capacidade de transmitir informações de mudança de estado.

A troca de informações é baseada em um mecanismo de publicador (*publisher*)/receptor (*subscriber*). O publicador escreve os valores no lado emissor. O receptor faz a leitura dos valores. O sistema de comunicação é responsável por atualizar os *buffers* locais dos publicadores. Se o valor de um ou vários *DataAttributes* de uma função no DATA-SET muda, o *buffer* de transmissão do publicador é atualizado e os valores são transmitidos em uma mensagem GOOSE. As mensagens GOOSE contêm informações que informam

ao receptor que houve alteração no DATA-SET e o tempo da última mudança o qual possibilita ao dispositivo receptor o ajuste dos tempos locais de determinado evento. Para conferir confiabilidade adicional à entrega das mensagens com mudanças no DATA-SET, são realizadas retransmissões dos dados em curtos intervalos de tempo conforme ilustra a Figura 9, a seguir.

Figura 9 – Tempo de transmissão de eventos.

Fonte: IEC61850 (2004).

Cada mensagem carrega o parâmetro *timeAllowedToLive* que informa ao receptor o tempo máximo de espera, até a próxima retransmissão. Na ocorrência de um evento, este tempo é diminuído e várias mensagens são enviadas em intervalos de tempo menores os quais vão aumentando, gradativamente, conforme Figura 9. Este tempo se estabiliza em condições estáveis de retransmissão, até que um novo evento de alteração no DATA-SET ocorra.

Segurança para a Rede de Comunicação das Subestações

A utilização de redes *Ethernet* para comunicação nas subestações de energia elétrica empregando o padrão *International Electrotechnical Commission* (IEC)-61850 para a comunicação entre *Intelligent Electronic Devices* (IEDs) traz uma série de benefícios para a automação das subestações, porém, expõe a rede de comunicação à ataques cibernéticos. As ameaças aos IEDs de uma rede IEC-61850 diferem, significativamente, das enfrentadas por uma rede empresarial (PREMARATNE et al., 2008). Assim, um Sistema de Detecção de Intrusão (*Intrusion Detection System* (IDS)) desenvolvido para uma rede empresarial seria ineficaz contra a maioria dos ataques específicos ao protocolo IEC-61850 e aos IEDs.

O presente capítulo descreve as ameaças à infraestrutura de rede de comunicação de dados das subestações de energia elétrica. Baseado na norma IEC-62351, criada para caracterizar a necessidade de segurança da informação no ambiente das subestações, são apresentadas recomendações de segurança do padrão. Este referencial teórico oferece embasamento para a determinação dos tipos de ataques pertinentes às mensagens *Generic Object Oriented Substation Event* (GOOSE), bem como, as formas de atuação de um IDS.

3.1 Revisão Bibliográfica

Redes de comunicação industriais estão cada vez mais baseadas em protocolos abertos e plataformas utilizadas também nas redes corporativas de Tecnologia da Informação (TI) e ambiente Internet, conforme artigo de Dzung et al. (2005). Essa reutilização facilita o desenvolvimento e a implantação de sistemas altamente conectados, mas, também, torna o sistema vulnerável a ataques cibernéticos. O artigo fornece uma visão geral dos problemas de segurança de TI em sistemas de automação industrial, que são baseados em sistemas abertos de comunicação tal como o IEC-61850. Ten, Hong e Liu (2011) corroboram, afirmando que os Sistema Elétricos (SEs) estão cada vez mais dependentes dos equipamentos digitais e das redes de comunicação de dados, o que torna a segurança cibernética um

agravante para a continuidade no fornecimento de energia elétrica. Ainda, no referido artigo, um mecanismo de inferência é proposto para antever a detecção de ataques cibernéticos (*cyber*) nas subestações. O método de detecção proposto considera anomalias temporais. Eventos potenciais de intrusão são classificados com base no impacto ao SE. O resultado desta pesquisa é uma ferramenta de detecção de intrusão cibernética que identifica ataques capazes de causar danos significativos à rede elétrica. Premaratne et al. (2010b) também apresentam o desenvolvimento de um IDS. Seu trabalho ressalta que um IDS deve ser adaptado para conter as ameaças em uma subestação de energia elétrica que utiliza o IEC-61850. Seu sistema foi desenvolvido, analisando os dados gerados em ataques simulados à IEDs. O sistema é testado, simulando novos ataques e em atividade por um utilizador autêntico. Um novo método para avaliar o risco temporal de uma intrusão em subestação de energia elétrica baseado na análise estatística dos ataques conhecidos é proposto nesse mesmo artigo.

Premaratne et al. (2009) detalham o desenvolvimento de um IDS para ser utilizado em IEDs que utilizam o padrão IEC-61850 de subestações de energia elétrica. Ataques simulados são realizados aos IEDs e todo o tráfego, do acesso genuíno e dos ataques maléficos, é capturado. A classificação é feita, utilizando árvores de decisão que demonstraram eficiência na identificação do tráfego sob ataque.

Liu et al. (2010) enfatizam que a avaliação da segurança é uma função chave que deve ser realizada antes da implantação de segurança. As dificuldades de avaliação da segurança incluem a análise do processo de controle de energia, bem como o grau de segurança de cada etapa de controle. Seu trabalho descreve uma metodologia de avaliação de segurança para a rede de comunicação de sistemas de controle de energia.

Avaliação de riscos de segurança da informação de sistemas de controle de energia IEC-61850 é um problema não resolvido (LIU; ZHANG; WU, 2011a). Uma das razões é a falta de metodologia para a análise de ativos, importante processo de avaliação de risco. Como as características de sistemas de controle de energia baseados no padrão IEC-61850 são diferentes dos sistemas gerais de TI, os autores introduzem uma metodologia de análise específica.

A interrupção das operações de energia elétrica pode ser catastrófica para a segurança e a economia. Devido à complexidade dos ativos dispersos e às interdependências entre os equipamentos, comunicação e infra-estruturas de energia, a necessidade de atender à segurança, em conformidade com a qualidade nas operações, é uma questão desafiadora.

Liu, Zhang e Wu (2011b) aplicam uma metodologia para a avaliação da segurança em sistemas de automação de subestações baseada na norma IEC-61850 apresentada por Liu, Zhang e Wu (2011a).

Os procedimentos gerais de identificação e avaliação de ativos são representados passo a passo. A partir dos resultados da aplicação, a metodologia pode satisfazer às exigências de avaliação de riscos.

A segurança cibernética é também relevante para as *Smart Grids*. Lu et al. (2010) fazem uma classificação e avaliação das ameaças à segurança nas redes de comunicação das *Smart Grids*. Os possíveis ataques a estas redes foram categorizados em três tipos: disponibilidade da rede, a integridade dos dados e a privacidade de informações. O impacto do ataque *Deny of Service* (DOS) é avaliado quantitativamente. Os experimentos apresentados no trabalho, demonstram que o desempenho da rede diminui, drasticamente, quando a intensidade do ataque DOS se aproxima do máximo.

No trabalho de Ten, Manimaran e Liu (2010) eles propõem o controle, a supervisão e aquisição de dados da estrutura de segurança com quatro componentes principais: monitoração em tempo real, detecção de anomalias, análise de impacto, e estratégias de mitigação. Para avaliar o nível de vulnerabilidade do sistema é proposta uma metodologia baseada em árvores de ataque. A segurança da informação para concessionárias de energia elétrica deve ter um tratamento adequado (ERICSSON, 2009). Neste artigo, a segurança da informação é analisada sob três aspectos principais: estruturas de segurança, avaliação de riscos e segurança tecnológica. Para a utilização de tecnologias de segurança, a orientação é a implantação de soluções diferentes, utilizando controles diferentes, com base em um diagrama lógico.

Hurd, Smith e Leischner (2008) apresentam um tutorial prático sobre as medidas de segurança que podem ser implementadas em redes de comunicação de sistemas de potência. O tutorial analisa, ainda, os protocolos de segurança para o desenvolvimento de aplicações em subestações para alcançar a segurança cibernética, utilizando protocolos comprovados, controlados e adaptados para o uso nas subestações. Ericsson (2010) relata que o desenvolvimento das capacidades de comunicação, passando de sistemas de controle de energia, a partir de ilhas de automação, para ambientes computacionais totalmente integrados, abriram novas possibilidades e vulnerabilidades.

Analisar a *performance* da ferramenta de segurança é também relevante, pois, requisitos de tempo restrito são definidos na norma IEC-61850 para mensagens de missão crítica, GOOSE por exemplo. Yang, Kim e Jang (2012) aplicaram algoritmos de segurança como *Message-Digest algorithm 5* (MD5), *Secure Hash Algorithm* (SHA)-1 e RSA para medir o tempo total de processamento dos algoritmos de segurança e o tempo de transmissão entre os equipamentos, para encontrar um mecanismo de segurança que atendesse aos requisitos de tempo do serviço GOOSE.

Em seu artigo, Coates et al. (2009) fazem uma análise semelhante, realizando cálculos de latência, porém, aplicados ao sistema *Supervisory Control and Data Acquisition* (SCADA). A latência é utilizada para estimar os limites de aplicabilidade dentro da subestação de energia e entre locais geograficamente distantes, como a concessionária, os centros de controle e as implementações regionais. O artigo investiga políticas de rede e sobre mecanismos para melhorar a segurança em redes, usando os protocolos de transporte *Transmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP).

Sobre a segurança de sistemas SCADA, Ten, Liu e Manimaran (2008) propõem um quadro de avaliação de vulnerabilidades para analisar, sistematicamente, as brechas de segurança dos sistemas *Supervisory Control and Data Acquisition* (SCADA) em três níveis: sistema, cenários e pontos de acesso. O impacto de uma possível invasão eletrônica é avaliado por seu potencial de perda de carga no sistema de energia.

A supervisão e o controle dos equipamentos não estão limitados ao ambiente da subestação. Wu, Bin e Zhou (2009) mostram a necessidade de integração da segurança realizada por comunicação de longa distancia nas subestações. O artigo apresenta a criação de uma camada de defesa nas subestações para receber informações dinâmicas da rede elétrica. Os dados coletados de várias subestações são integrados e utilizados como condições de restrição de segurança para o SE.

Premaratne et al. (2010a) propõem um esquema para a auditoria de segurança de uma rede IEC-61850 baseado em métricas de segurança para IEDs. A análise detalhada de uma subestação automatizada IEC-61850 é realizada com foco nos possíveis objetivos do atacante. Para realizar a auditoria de segurança, é necessário o uso de ferramentas analisadoras de rede, gerando tráfego que pode impactar a performance da rede. O trabalho demonstra que a detecção de intrusão, em nível de rede e equipamentos, é uma medida de segurança viável para redes IEC-61850.

A pesquisa sobre segurança no SE tem se intensificado nos últimos anos. Os artigos apresentados e discutidos possuem propostas similares que sustentaram o desenvolvimento do presente trabalho, tanto no que se refere ao IDS, quanto à análise temporal das mensagens para detecção do intruso.

Vale dizer que os artigos analisados neste capítulo, no que tange à metodologia e aos resultados obtidos pelos autores, foram cruciais para o desenvolvimento deste trabalho.

3.2 Ameaças a Rede de Comunicação das Subestações

Proteção, segurança e confiabilidade sempre foram temas importantes para o projeto e a operação de sistemas do setor de energia elétrica. Segurança da informação tem se tornado importante para esta indústria, visto que ela se baseia, cada vez mais, em infraestrutura de comunicação.

O mercado desregulamentado impôs novas ameaças como o conhecimento dos bens de um concorrente e a operação de seu sistema podem ser benéficos, sendo a aquisição de tais informações possível (IEC62351, 2007a). Além disso, as ações involuntárias, como negligência e catástrofes naturais, podem ser tão prejudiciais quanto as ações deliberadas.

Os protocolos de comunicação constituem uma das partes mais críticas da operação do sistema de energia e responsáveis pela recuperação de informações dos equipamentos de campo e envio de comandos para controle. Apesar desta função crucial, raramente estes

protocolos incorporam medidas de segurança contra erros involuntários, mau funcionamento dos equipamentos do SE, falha dos equipamentos de comunicação, ou sabotagem deliberada. Devido à especificidade destes protocolos, a segurança por obscuridade tem sido a principal abordagem (IEC62351, 2007a). Afinal, somente os operadores são autorizados a controlar os disjuntores dos centros de controle altamente protegidos.

O interesse em conhecer a quantidade de *megawatts* em uma linha de transmissão, ou em aprender a ler os *bits* e *bytes* idiossincráticos dos diversos protocolos de comunicação, ou mesmo o interesse em interromper sistemas de energia tem crescido. Segurança por obscuridade não é mais uma concepção válida. O mercado de energia elétrica tem pressionado as concessionárias a mostrarem vantagens competitivas. Não só o mercado e a complexidade da operação do SE têm aumentado ao longo dos anos, tornando as falhas nos equipamentos e os erros operacionais mais prováveis, sendo seu impacto, escopo e seus custos também maiores (IEC62351, 2007a). Os protocolos denominados obscuros vêm sendo substituídos por protocolos padronizados, bem documentados, mas susceptíveis a *hackers* e espiões industriais.

Como a indústria de energia elétrica depende cada vez mais de informações para controlar o sistema de alimentação, a infraestrutura de comunicação precisa ser gerenciada. A gestão do SE tornou-se dependente da infraestrutura de comunicação com automação contínua que substitui operações manuais, informações mais precisas e oportunas, além de deter o controle sobre a vida útil dos equipamentos. Portanto, a confiabilidade do SE é cada vez mais afetada por problemas que a infraestrutura de informação pode sofrer.

O setor elétrico está acostumado a se preocupar com falhas de equipamentos e descuidos relacionados à segurança. O que tem mudado é o interesse em se protegerem as informações, aspecto cada vez mais importante da operação do sistema de energia segura, confiável e eficiente.

A avaliação do risco de segurança é vital para determinar o que precisa ser protegido contra as ameaças e qual é o grau de segurança. O ideal seria determinar a relação custo benefício, porém, se a segurança adicional for realizada contra possíveis ataques deliberados, esse monitoramento pode ser utilizado para melhorar a segurança, minimizar o descuido e aumentar a eficiência da manutenção dos equipamentos.

3.2.1 Ameaças Deliberadas

Ameaças deliberadas podem causar danos às instalações e aos equipamentos. Os incentivos a estas ameaças vêm aumentando à medida que o resultado de ataques bem sucedidos oferecem benefícios econômicos, políticos e sociais aos atacantes. O monitoramento das instalações e dos equipamentos previne algumas destas ameaças e ameniza o impacto dos ataques bem sucedidos, por meio de notificações, em tempo real, e auditoria forense (IEC62351, 2007a). Os ataques mais comuns podem ser caracterizados como:

- ❑ Funcionários descontentes, que possuem o conhecimento necessário para causar dano ao SE, são considerados uma ameaça mais danosa que um não empregado, em especial, na indústria de energia na qual os sistemas e equipamentos são exclusivos.
- ❑ Espionagem industrial em sistemas de energia que tem se tornado uma ameaça pela competição entre as concessionárias, com o incentivo crescente ao acesso não autorizado à informação. A danificação dos equipamentos para fins nefastos, expõe a incompetência ou a falta de fiabilidade dos concorrentes.
- ❑ Vandalismo também é uma ameaça corrente nas instalações de energia elétrica. O monitoramento do acesso em tempo real pode ajudar a diminuir o vandalismo.
- ❑ *Hackers*¹ são pessoas que pretendem violar a segurança da informação. Em sua maioria, utilizam a Internet como porta de entrada nos sistemas de comunicação. Os computadores utilizados para a realização de operações no sistema de energia são mantidos isolados da Internet. No entanto, o *Hacker* pode chegar ao sistema de energia, através de um acesso remoto, utilizando informações do *laptop* de manutenção, ou com o acesso ao IED feito pelo engenheiro para a coleta de dados (IEC62351, 2007a).
- ❑ Como os *hackers* os vírus² e *worms*³ atacam através da Internet. No entanto, alguns vírus e *worms* podem ser incorporados a *softwares*, arquivos ou se propagarem através da rede. Neste íterim, sua entrada ao sistema de energia é viabilizada por equipamentos ditos seguros, possibilitando a realização de ataques do tipo *man-in-the-middle*⁴, *keylogger*⁵ e *trojan horses*⁶.
- ❑ Roubo de informações confidenciais também é considerado uma ameaça e, como os demais ataques, requer o monitoramento do acesso às instalações com emissão de alarmes sobre o estado de funcionamento dos equipamentos, e que pode alertar sobre a existência de um ataque.

---

¹Indivíduos que exploram a vulnerabilidade dos sistemas computacionais, com o objetivo de comprometer ou afetar sua segurança (STALLINGS, 2008).

²Programa malicioso de computador que se propaga, tornando-se parte de outros programas e arquivos (CERT.BR; CGI.BR; NIC.BR, 2012).

³Programas capazes de se propagar automaticamente pelas redes, enviando cópias de si para outros computadores (CERT.BR; CGI.BR; NIC.BR, 2012)

⁴Ataque através do qual o intruso se interpõe na comunicação entre as partes, recebendo e encaminhando as mensagens entre o emissor e o receptor (BRAZIL; ALBUQUERQUE, 2006).

⁵Captura e armazena as teclas digitadas pelo usuário no teclado do computador (CERT.BR; CGI.BR; NIC.BR, 2012).

⁶Cavalo de troia é um programa que, além de executar as funções para os quais foi projetado, executa outras, porém, maliciosas sem o conhecimento do usuário (CERT.BR; CGI.BR; NIC.BR, 2012).

3.2.2 Requisitos de Segurança

Para discorrer sobre a segurança da informação, é necessária a apreciação de alguns requisitos como: confidencialidade, integridade, disponibilidade, autenticidade e não repúdio ou responsabilidade.

- ❑ **Confidencialidade:** prevenir o acesso não autorizado à informação. Quando informações são lidas ou copiadas por alguém não autorizado a fazê-lo, o resultado é conhecido como perda de confidencialidade. Para algumas informações, confidencialidade é um atributo indispensável. Como exemplo, citam-se dados de autenticação dos usuários, pesquisas, especificações de um novo produto e estratégias de investimento da concessionária.
- ❑ **Manutenção da integridade da informação:** prevenir a modificação não autorizada da informação. Quando disponíveis em uma rede de comunicação de dados, as informações podem ser corrompidas. Quando uma informação é modificada de forma inesperada, o resultado é a perda da integridade. Isso significa que mudanças não autorizadas são realizadas nas informações, por erro humano ou de forma intencional. A integridade é particularmente importante para as informações de segurança crítica usadas para atividades como troca de mensagens de alta prioridade, na rede dos equipamentos do SE.
- ❑ **Segurança quanto à disponibilidade** significa prevenir o DOS e assegurar o acesso autorizado. A informação também pode ser apagada ou tornar-se inacessível, resultando na falta de disponibilidade. Isto significa que equipamentos que necessitam da informação disponibilizada, em determinado momento, e estão autorizados a acessar a referida informação, não conseguirão acessá-la.
- ❑ **Autenticidade** é um requisito para que o acesso seja legítimo. Para isso, há que se ter um tipo de segurança que visa à proteção contra ataques de roubo de identidade, conhecidos como *Phishing* e caracterizado por tentativas de obter informações confidenciais. No *Phishing*, o transgressor se faz passar por uma pessoa, empresa, ou um equipamento confiável.
- ❑ **Não repúdio ou responsabilidade** é o ato de impedir a negação da autoria de uma ação que ocorreu, ou a autoria do pedido de uma ação que não ocorreu (IEC62351, 2007a). Este tipo de requisito demanda uma segurança que tem como finalidade identificar o atacante, através de técnicas de auditoria realizadas nos agentes da comunicação.

Devem-se, também, observar três conceitos relacionados ao uso da informação, autenticação, autorização e irretratabilidade/irrevogabilidade.

- ❑ Autenticação: recomendado para tornar a informação disponível aos que precisam e são autorizados a acessá-la. A autenticação é utilizada para provar que o usuário, empresa ou equipamento são, realmente, quem demonstram ser. Esse processo envolve algo que o usuário conheça, como uma senha, que ele tenha, como um cartão, ou algo sobre o usuário que prove sua identificação, como o uso de recursos biométricos ou identificações únicas de equipamentos gravadas em *hardware*.
- ❑ Autorização: é o ato de determinar se um usuário particular, sistema computacional ou equipamento têm o direito de executar certa atividade, como, por exemplo, a leitura de um arquivo ou a execução de um programa.

Autenticação e autorização são processos empregados em conjunto. Os agentes da comunicação precisam ser autenticados, antes da execução de uma atividade que estão autorizados a realizar. A segurança é considerada eficaz, quando o resultado de uma autenticação não pode ser negado posteriormente, ou seja, se o agente não puder negar que executou uma determinada atividade.

- ❑ Irretratabilidade/irrevogabilidade: um ataque bem sucedido ocorre, quando um *hacker* invade o sistema alvo, obtém os dados de seu interesse e se desconecta, sem ser identificado. Neste caso, o ataque é classificado como irretratável e esse torna possível graças à negligência na auditoria do sistema invadido, ou por se revogarem os dados da sua identificação, apagando, assim, os rastros deixados no acesso ao sistema. Portanto, além da segurança para o acesso, é necessária a segurança nos dados de auditoria do sistema, para que não ocorra a irretratabilidade da operação realizada e se obtenha a irrevogabilidade dos dados de auditoria.

3.2.3 Tipos de Ataque e Formas de Segurança

As ameaças podem ser realizadas através de diversos tipos de ataques, conforme ilustra a Figura 10. O mesmo ataque pode causar diferentes ameaças à segurança. Cada tipo de ataque precisa ser combatido. Uma cadeia de ataques envolvendo diferentes ativos, e possivelmente ocorrendo ao longo do tempo, pode gerar novas ameaças.

Muitas vezes, informações aparentemente simples podem contribuir para o comprometimento de um sistema computacional. Informações que os invasores consideram úteis incluem quais *hardwares*, *softwares*, configurações, tipo de conexão com a rede, informações pessoais dos usuários, procedimentos de acesso e autorização que estão sendo utilizados. Informações relacionadas à segurança podem permitir que indivíduos não autorizados ganhem o acesso a importantes arquivos e programas e comprometam a segurança do sistema. Exemplos de informações importantes que podem contribuir para o comprometimento da segurança em sistemas de comunicação são: senhas, informações de autenticação, chaves de criptografia e versões de sistemas.

Figura 10 – Requisitos de segurança, ameaças, e possíveis ataques

Fonte: IEC62351 (2007a)

Segundo IEC62351 (2007a), segurança pode ser classificada em quatro categorias (Figura 11). Usualmente, estas categorias precisam ter formas de alcançar a segurança, fim a fim. Garantir a segurança de uma categoria somente não é o suficiente. Estas medidas de segurança devem ser cuidadosamente integradas para que problemas inesperados sejam controlados.

Segurança engloba um conjunto complexo e multi-facetado de questões. Não existem mecanismos padronizados ou claramente definidos para compor o espaço do problema de segurança. Portanto, a análise e implantação de medidas de segurança adequadas são frequentemente percebidas como tarefas bastante complexas (IEC62351, 2007a).

Todavia, a segurança pode ser decomposta em regiões menores de análise e gerenciamento. Como exemplo, podem-se citar três abordagens diferentes, dependendo dos problemas envolvidos, a saber:

- ❑ Segurança física da região onde os ativos críticos são alojados e para os quais o acesso é controlado.
- ❑ Segurança de perímetro, ou seja, da fronteira lógica em torno de uma rede de comunicação.
- ❑ Domínio de segurança, no qual regiões, departamentos, ou protocolos, que possuem requisitos de segurança distintos, são analisados de forma independente.

Figura 11 – Categorias, ataques típicos e medidas de segurança

Fonte: IEC62351 (2007a)

A operação do sistema elétrico pode representar muitos desafios de segurança diferentes das demais indústrias. Por exemplo, a maioria das formas de segurança foram desenvolvidas para combater *hackers* na Internet. O ambiente da Internet é diferente do das operações do sistema de energia. Pouco se compreende sobre o impacto das medidas de segurança nos requisitos de comunicação de operações do SE. Os serviços de segurança e tecnologias foram desenvolvidos para indústrias que possuem poucos requisitos de confiabilidade e desempenho. A rede *Local Area Network* (LAN) da subestação deve atender requisitos rigorosos para realizar as operações do sistema elétrico, por exemplo:

- ❑ Impedir que um operador autorizado acesse os controles das subestações de energia tem consequências mais graves que se um usuário não conseguir acessar sua conta bancária. A ameaça de ataques DOS são mais críticas nas instalações do SE.
- ❑ Muitos canais de comunicação utilizados no SE são de banda limitada e, muitas vezes, o equipamento final é limitado em memória e poder de processamento, o que inviabiliza a utilização de medidas de segurança como a criptografia.
- ❑ Muitos sistemas são conectados por canais multiponto diferentemente da indústria, o que impede o uso das ferramentas de segurança presentes na indústria.
- ❑ A comunicação sem fio no SE é dificultada por ser um ambiente ruidoso e exigir respostas rápidas e confiáveis. Medidas de segurança estão disponíveis para os sistemas sem fio, porém, elas aumentam a sobrecarga na comunicação (IEC62351, 2007a).

3.3 Recomendações do Padrão IEC-62351

Padrões de segurança têm sido desenvolvidos para diferentes perfis de protocolos de comunicação. O protocolo IEC-62351 possui diferentes objetivos de segurança tais como (IEC62351, 2007a):

- a) autenticação para minimizar a ameaça de ataques *man-in-the-middle*;
- b) autenticação para minimizar alguns tipos de desvios de controle;
- c) autenticação para minimizar as falhas causadas por descuido ou por ações de funcionários descontentes;
- d) autenticação de entidades por meio de assinaturas digitais para:
  - assegurar o acesso à informação apenas a autorizados;
  - controlar o acesso à comunicação;
- e) confidencialidade das chaves de autenticação, através do uso de criptografia;
- f) confidencialidade das mensagens por criptografia para comunicações que utilizam recursos para lidar com carga adicional;
- g) integridade por detecção de sabotagem;
- h) prevenção de *playback*⁷ e *spoofing*⁸;
- i) dispositivos seguros e não seguros devem co-existir na mesma rede, mesmo que isso possa abrir algumas questões de segurança (*backdoors*⁹);
- j) monitoramento da infra-estrutura de comunicação, que pode fornecer:
  - detecção de intrusão;
  - monitoramento da carga nos recursos;
  - disponibilidade de componentes no sistema de informação.
- k) um conjunto de políticas de gerenciamento de identidade;
- l) desejo de utilizar metodologias tradicionais de TI.

3.4 Tipos de Ataque Pertinentes às Mensagens GO- OSE

A norma IEC-62351 especifica mensagens, procedimentos e algoritmos para a segurança da operação dos protocolos derivados do padrão IEC-61850. Para aplicações que

---

⁷Ataque através do qual o invasor captura algum dado de autenticação e tenta utilizá-lo para uma nova autenticação no sistema (BRAZIL; ALBUQUERQUE, 2006).

⁸Técnica que consiste em falsificar os campos das mensagens, de forma a aparentar que elas foram enviadas por outra pessoa (CERT.BR; CGI.BR; NIC.BR, 2012).

⁹Programa que permite o retorno do invasor a um microcomputador comprometido, por meio da inclusão de serviços criados ou modificados para este fim (CERT.BR; CGI.BR; NIC.BR, 2012).

utilizam mensagens GOOSE, para o qual são exigidos tempos de resposta dentro dos 4 ms, configurações de *multicast* e baixo uso da Unidade Central de Processamento (*Central Processing Unit* (CPU)), o uso de criptografia não é recomendado (IEC62351, 2007b). Para oportunizar confidencialidade na troca de mensagens, é realizada a seleção de caminhos de comunicação, através de *Virtual Local Area Network* (VLANs). A fim de proteger o tráfego de mensagens GOOSE nas subestações de energia elétrica, as ferramentas de segurança devem implementar as seguintes medidas (IEC62351, 2007b):

- a) o *subscriber* deve acompanhar o tempo corrente das mensagens GOOSE. A mensagem cujo *timestamp* exceder a 2 minutos não deve ser processada. A variação do período é configurável e deve suportar oscilações máximas de 10 segundos para mais ou para menos;
- b) o *subscriber* deve filtrar as transições, avaliando as mudanças em *stNum*;
- c) o *subscriber* deve registrar e acompanhar o *stNum* recebido pelo *publisher*. Se um valor menor para *stNum* é recebido, e este não chegou ao valor máximo ou o *timeallowedtoLive* não chegou ao limite, a mensagem deve ser descartada;
- d) se o tempo limite da mensagem for atingido, o *stNum* deve ser reiniciado;
- e) se o *stNum* chegar ao valor máximo, ele deve ser reiniciado;
- f) após a inicialização, o *stNum* deve ser zero.

Observando os requisitos de segurança para a replicação das mensagens GOOSE supracitados, podem-se caracterizar alguns ataques pertinentes às mensagens GOOSE, a saber:

- a) *man-in-the-middle*: tipo de ataque através do qual um equipamento espião se torna o intermediário entre duas ou mais comunicações recebendo as mensagens do emissor e encaminhando-as para o receptor (STALLINGS, 2008). Em relação às mensagens GOOSE, o espião pode criar mensagens falsas para a realização de comandos, ou mesmo criar mensagens malformadas com o objetivo de causar erros nos equipamentos.
- b) *VLAN hopping*: método de ataque utilizado para que o vilão tenha acesso aos pacotes de outra VLAN. Na rede da subestação, a comunicação GOOSE estará isolada por uma VLAN. Para que o *hacker* proceda a ataques à esta rede, ele deverá transpor esta barreira.
- c) DOS: classe de ataque que busca estourar a capacidade de *buffers* e memórias, através de mensagens com erros na sua formação ou ocasionados por ataques, para interromper a operação do sistema (IEC62351, 2010).
- d) *Distributed Deny of Service* (DDOS): um ataque que tem como objetivo esgotar os recursos ou causar significativo impacto no desempenho de equipamentos, devido

a um inesperado elevado número de mensagens enviadas, deliberadamente, para que impeçam ou atrasem a chegada de mensagens legítimas.

3.5 Prevenção e Detecção de Intrusão

Técnicas tradicionais de prevenção de intrusão como: *firewall*, controle de acesso ou criptografia, têm falhado na proteção de sistemas e redes. Tal fato se deve à elevada sofisticação dos ataques e *malwares*¹⁰, ou seja, códigos maléficos. Como resultado, os IDSs tem se tornado um componente de segurança indispensável para detectar as ameaças, antes que elas causem danos generalizados (WU; Banzhaf, 2010).

No desenvolvimento de um IDS é necessário considerar muitas questões como: coleta de dados, pré-processamento dos dados, pós processamento de dados, reconhecimento da intrusão, relatórios e respostas ao sistema monitorado. Dentre eles, o principal é a descoberta do intruso. Dados de auditoria são comparados com modelos de detecções que descrevem os padrões ou de intrusão ou de comportamento correto, de modo que ambos possam identificar, com ou sem êxito, as tentativas de intrusão.

Um IDS monitora, dinamicamente, os eventos de um sistema e decide se eles são sintomas de um ataque, ou constituem o seu uso legítimo. Em geral, os IDSs são classificados em duas categorias, de acordo com os métodos de detecção que eles empregam (Chebrolu; Abraham; Thomas, 2005), são elas: detecção de uso indevido e detecção de anomalias.

Detecção de uso indevido identifica intrusões pela observação dos dados, com descrições pré-definidas de comportamento do intruso. Por este motivo, a aproximação é largamente adotada na maioria dos sistemas comerciais. Todavia, as intrusões são polimórficas e se desenvolvem continuamente. A detecção de uso indevido pode falhar, quando estiver enfrentando uma intrusão desconhecida. Uma forma de contornar este problema é atualizar, regularmente, a base de conhecimento de intrusões.

Detecção de anomalias é independente à detecção de uso indevido. Ela levanta a hipótese de que o comportamento anormal é raro e diferente do comportamento normal. Por isso, este modelo de construção para comportamentos normais, detecta anomalias nos dados observados por perceber desvios desses modelos.

Existem dois tipos de detecção de anomalias a estática e a dinâmica: A estática assume que o comportamento do sistema monitorado nunca muda. O segundo tipo, a detecção dinâmica de anomalia, extrai padrões de comportamento dos usuários finais, ou do histórico de uso das redes e computadores. Algumas vezes, estes padrões são chamados de perfis.

---

¹⁰São programas desenvolvidos para se executarem ações e atividades danosas no computador (CERT.BR; CGI.BR; NIC.BR, 2012).

Claramente, detecção de anomalias tem a habilidade de detectar novos tipos de intrusão, e trabalha com dados normais, apenas para a definição dos perfis. Todavia, a grande dificuldade é descobrir os limites entre o comportamento normal e anormal, devido à falta de exemplos de comportamentos anormais. Outra dificuldade é a constante mudança no comportamento normal, especialmente, nos sistemas dinâmicos.

Sistema de Detecção e Bloqueio de Ataques *man-in-the-middle*: Abordagem Utilizada, Tempos Envolvidos e Lógica Implementada

Para o desenvolvimento de um modelo eficiente de *Intrusion Detection System* (IDS), diversas análises foram necessárias no que se refere à seleção de atributos relevantes à identificação do intruso, definição de estratégias para seu reconhecimento, escolha da plataforma de desenvolvimento, sincronização e medição do tempo de resposta das implementações realizadas. O presente capítulo descreve os recursos, ferramentas e *softwares* desenvolvidos para detecção e bloqueio de ataques do tipo *man-in-the-middle*, utilizando *Software-Defined Networking* (SDN) através do protocolo *OpenFlow*.

4.1 Revisão Bibliográfica

Dmitrenko (2011) relata que a implementação do *International Electrotechnical Commission* (IEC)-61850 como simples protocolo de comunicação é irrelevante para o setor de energia. Este padrão foi concebido para oferecer condições de comunicação, integração e medições em tempo real. Em seu artigo, o autor descreve a implementação de um sistema de medição com base na norma IEC-61850, demonstrando as reorganizações necessárias nos *Intelligent Electronic Devices* (IEDs) para a automação da subestação, o controle e monitoramento.

O padrão IEC-61850 implementa mensagens de tempo crítico necessárias para a tomada de decisões dos IEDs. Para trafegar essas mensagens é necessária uma rede de comunicação confiável, rápida e determinística, para sustentar a comunicação entre os IEDs, *Merging Unit* (MU) e IEDs, monitoramento em tempo real, proteção e controle das funções das subestações.

O trabalho de Thomas e Ali (2010) propõe uma arquitetura lógica baseada no *Ethernet* para a rede de comunicação da subestação. Foi medido o tempo de tráfego fim a fim de mensagens de tempo crítico, demonstrando estar dentro do limite permitido pela norma IEC-61850. Neste ínterim, a arquitetura demonstrada é recomendada para a utilização em aplicações de proteção e controle para manter a confiabilidade, disponibilidade e continuidade no funcionamento da rede elétrica.

Grandes fabricantes estão desenvolvendo produtos para o barramento de processo das subestações, em conformidade com a norma IEC-61850. Para o sucesso da implementação do IEC-61850 no barramento de processo, é importante analisar o desempenho de mensagens de tempo crítico, tais como *Generic Object Oriented Substation Event* (GOOSE) e *Sampled Values* (SV), para a proteção e o controle da subestação (KANABAR; SIDHU, 2011). A pesquisa mostrou, ainda, técnicas de estimativa de valor como atitude corretiva para os pacotes perdidos. Kanabar, Sidhu e Zadeh (2011) apresentam a implementação de um *hardware* que comunique IEC-61850 baseado na rede de comunicação do barramento de processo, para investigar a medida corretiva proposta para o atraso e a perda de mensagens no barramento de processo.

Os trabalhos apresentados apontam as restrições de latência no desenvolvimento de sistemas para o Sistema Elétrico (SE), bem como, analisam a capacidade da rede *Ethernet* para troca de mensagens de alta prioridade. Neste contexto, Almas e Vanfretti (2016) descrevem um *Real Time Hardware-in-the-loop* para medição da *performance* de diferentes algoritmos de proteção do SE, utilizando o canal de comunicação, dentre eles, os algoritmos que utilizam mensagens GOOSE do padrão IEC-61850. Neste ínterim, foi observado que ferramentas específicas são necessárias para um rápido acesso aos pacotes de rede para o desenvolvimento de aplicações que utilizam este protocolo. A necessidade de flexibilização e processamento no *hardware* torna as *Field-Programmable Gate Arrays* (FPGAs) e processadores *Advanced RISC Machine* (ARM) *on-chip* os melhores candidatos à implementação de dispositivos de rede em geral.

Moreira et al. (2016) investigaram o uso do protocolo *Precision Time Protocol* (PTP) para o sincronismo de tempo entre os equipamentos da rede de comunicação das subestações. Para tal, eles exploram as funcionalidades do PTP em uma Xilinx Zynq para alcançar nanosegundos de sincronização, assim como Astarloa et al. (2015). Astarloa et al. (2013) apresentam a interligação entre uma rede anel *Ethernet* entre IEDs, que oferece tolerância a falhas no canal de comunicação, com uma rede estrela da subestação, através de um *gateway* implementado em uma *Xilinx Zynq*, que integra os protocolos com baixa latência. A plataforma *Zynq da Xilinx*, possui recursos interessantes para o desenvolvimento de dispositivos de rede. Sua arquitetura é composta por um processador ARM acoplado a uma FPGA Artix-7 e a uma grande quantidade de interfaces de comunicação. Sua estrutura permite a instalação e configuração de um Sistema Operacional (SO) de tempo real. Aliada a tecnologias de rede, esta plataforma é uma ferramenta importante

para implementação de sistemas de rede com baixa latência e alta performance.

O protocolo *Openflow* fornece a capacidade de programação e flexibilidade necessárias para possibilitar a inovação em uma nova geração de arquiteturas de comunicação para as *Smart Grids* (SYDNEY et al., 2013). Neste artigo, um *switch Openflow* é comparado com um *Multiprotocol Label Switching* (MPLS) e demonstrou performance similar para controle das *Smart Grids*.

Molina et al. (2015b) avaliam o desempenho do *OpenFlow* em balanceamento de carga para múltiplos caminhos, simultaneamente. Para verificar a redução de latência, foram realizados testes com as mensagens do padrão IEC-61850. Molina et al. (2015a) apresentam uma arquitetura baseada em *OpenFlow* que estabelece diferentes tipos de fluxos baseados nos arquivos de configuração do IEC-61850, de acordo com o estado da rede. Com base em fluxos, são aplicadas técnicas de automação para executar uma gestão que permite recursos de filtragem de tráfego, Qualidade de Serviço (QoS), balanceamento de carga ou técnicas de segurança.

Lopes et al. (2015) propõem uma arquitetura completa para gerência e controle autônomos da rede de comunicação IEC-61850 das subestações. A solução proposta usa *OpenFlow*, a fim de atingir flexibilidade e granularidade no tratamento dos fluxos de dados. A implementação calcula, proativamente, a árvore *multicast* para encaminhar mensagens GOOSE e SV, e reconfigura os fluxos, em caso de falhas na rede de comunicação.

Pfeiffenberger e Du (2014) avaliaram as vantagens e os riscos do uso das SDNs no sistema elétrico. Os autores mencionam que as SDNs possuem potencial para oferecer significantes avanços para as redes de comunicação, mas não recomendam o seu uso em ambiente produtivo nas *Smart Grids*, por entenderem que existem questões em aberto, em relação à especificação e às implementações mostradas nas medições. Para o sistema aqui descrito, foram utilizadas ferramentas para o desenvolvimento de *software* embarcado, sistema operacional de tempo real *open source*, protocolos específicos para o acesso à rede de comunicação e para controle do *switch* concentrador da rede de comunicação.

O primeiro passo para o desenvolvimento desse sistema consistiu na identificação de vulnerabilidades na rede de comunicação das subestações, observando o protocolo IEC-62351 e realizando testes de invasão de forma a avaliar as ações de um *hacker* que obtenha acesso à rede da subestação.

4.2 Análise de Segurança das Mensagens GOOSE

Testes de invasão, ou *pentesting*, correspondem a simulação de ataques reais para avaliar o risco associado a potenciais violações de segurança. Em um *pentest*, propõe-se a descobrir não só vulnerabilidades que poderiam ser usadas por atacantes, mas, também, explorá-las sempre que possível, para avaliar o que os invasores conseguirão com uma exploração bem sucedida (WEIDMAN, 2014). Explorando as vulnerabilidades das mensagens

GOOSE, Hoyos, Dehus e Brown (2012) realizaram ataques de falsificação de mensagens, para demonstrar que os IEDs estão susceptíveis a tais ações. Para proteger-se de ataques com mensagens falsas, o autor sugere algumas técnicas clássicas de segurança de redes *Ethernet* na camada 2, tais como: definir uma *Virtual Local Area Network* (VLAN) *Identifier* (ID) dedicada para todas as *trunk ports*¹; desativar portas não utilizadas ou colocá-las em uma VLAN não utilizada; usar VLAN diferente do padrão VLAN 1; definir todas as portas como não *trunk*; criar uma lista de acesso ou prefixo com base nas credenciais do dispositivo; evitar o uso de *Ethernet* compartilhada com Wireless LAN (WLAN)s ou *hubs*.

O fato é que, para todas as configurações apontadas, existe, pelo menos, um tipo de ataque para tentar ludibriá-las, tais como: *VLAN hopping*, *Media Access Control* (MAC) *spoofing*, *Address Resolution Protocol* (ARP) *spoofing*, *buffer overflow*, *loops* na rede e ataques ao *Spanning Tree Protocol* (STP). Um *hacker* pode obter acesso a uma subestação, através de um *vector* ataque, que consiste no caminho ou meio pelo qual o atacante ganha acesso a uma rede, a fim de alcançar o seu objetivo final (HOYOS; DEHUS; BROWN, 2012). A Figura 12 ilustra os equipamentos, a estrutura de rede e a comunicação realizada por mensagens GOOSE, em uma rede IEC-61850, cenário de um possível ataque do invasor.

Figura 12 – Cenário de atuação do invasor

Um invasor pode tentar o acesso à rede de comunicação da subestação através da rede de longa distância, geralmente, ferramentas de segurança como um *firewall* são instaladas logo após o roteador, para realizar o bloqueio deste tipo de acesso. O invasor pode ainda conseguir acesso à rede através das interfaces do usuário, tais quais vetores ataques dos tipos: *backdoor*, *trojan horses*, vírus ou *worms*. Caso um *hacker* consiga acesso à rede de comunicação de dados da subestação, o intruso pode efetuar ataques *man-in-the-*

¹ *Trunk ports* são portas do *switch* que aceitam tráfego com e sem *tag* da VLAN.

middle de falsificação de mensagens, possibilitando a abertura de disjuntores, execução de lógicas digitais indevidas e acionamentos indesejados. Para este tipo de ataque, IDSs são configurados na rede de comunicação para detectar e prevenir tais ações.

O protocolo IEC-62351 define alguns controles que o equipamento receptor deve realizar, para não processar mensagens adulteradas. Por exemplo: descartar mensagens GOOSE cujo campo *timestamp* exceda 2 minutos, avaliar a variação dos campos *stNum* e *sqNum* para não utilizar mensagens cujo valor deste campo esteja fora do esperado, entre outras que estão descritas na Seção 3.4 desta tese.

O Laboratório de Sistemas de Energia Elétrica (LSEE) possui IEDs com capacidade de comunicação, utilizando o protocolo IEC-6180 (Figura 13).

Figura 13 – Equipamentos do LSEE.

Para verificar se estes equipamentos realizam o controle e a validação dos campos da mensagem GOOSE, conforme recomendações do padrão IEC-62351, foi realizado um *pentest*, utilizando IEDs conectados a uma rede de comunicação de dados. Neste contexto, foi configurada uma lógica de geração de mensagens GOOSE, conforme proposto por Miranda (2009). Foram, configurados três IEDs, um como *publisher*, e dois como *subscribers*, sendo os IEDs *subscribers* de fabricantes distintos. Para realizar os testes, foi incluído na mesma rede de comunicação, um microcomputador utilizando SO usuário, portando *softwares* analisadores de rede e compiladores para linguagens de alto nível, de forma a viabilizar a implementação de algoritmos para realização dos testes (Figura 14).

Figura 14 – Arranjo laboratorial criado para realização dos *pentests*.

A lógica para geração de mensagens GOOSE foi configurada para que, durante 3 segundos, o IED *publisher* publicasse mensagens com valor falso, no Conjunto de Dados (DATA-SET). Transcorrido os 3 segundos, o IED simula a ocorrência de um evento, através do envio de mensagens GOOSE, com a mudança de um estado *booleano* no DATA-SET, de falso para verdadeiro. Este evento se estende por 1 segundo sinalizado por um *Led* na Interface Homem Máquina (IHM) dos IEDs *subscribers*.

Posteriormente, o DATA-SET é novamente alterado, retornando o valor *booleano* para falso. Este processo se repete 494 vezes, durante 32 minutos e 56 segundos, como demonstra a Figura 15, a seguir.

Figura 15 – Funcionamento da lógica de mudanças de estado em mensagens GOOSE.

Transcorrido este tempo, a sequência de mudanças nos estados *booleanos* é interrompida e o IED continua a enviar mensagens de condições estáveis de funcionamento, indefinidamente. Cabe salientar que, a cada mudança no DATA-SET de falso para verdadeiro, ou de verdadeiro para falso, as mensagens GOOSE são enviadas em intervalos de tempo menores, conforme descrito na Seção 2.7 desta tese, além de alterar os valores dos campos *stNum*, *sqNum*, *T* e *timeAllowedtoLive*, de acordo com a especificação listada na Seção 2.6 desta tese.

A configuração realizada possibilitou a elaboração de testes importantes para situar o alcance do IDS.

4.2.1 Ataque à IEDs pelo Envio de Mensagens Salvas

O primeiro teste pertinente à possíveis ataques que um invasor com acesso à rede de comunicação da subestação possa realizar, consistiu em capturar as mensagens da rede e, posteriormente, reenviá-las. As mensagens GOOSE do teste supracitado foram capturadas, utilizando o *software Wireshark* e salvas em um arquivo *Packet Capture* (PCAP). Após a execução do teste, com as transições mencionadas, utilizou-se o *software* TCPReplay, onde as mensagens do arquivo PCAP foram replicadas na rede de comunicação.

Sobre este teste, é importante observar que os IEDs *subscribers* recebem mensagens do IED *publisher* com o DATA-SET carregando o valor falso, na cadência de situação estável do SE. Ao enviar as mensagens GOOSE do arquivo PCAP, as quais simulam mudanças no DATA-SET em situações de evento no SE, os IEDs *subscribers* recebem mensagens como se tivessem dois *publishers* com as mesmas configurações de *hardware*, endereço MAC e de atributos do IED. Porém, uma mensagem está em situação normal de operação, sincronizada e na cadência correta dos campos e a outra corresponde a um momento anterior de execução do sistema, no qual os campos de tempo estão inconsistentes, apresentando marcações antigas para os eventos informados. O comportamento observado nos IEDs *subscribers* foi idêntico ao vislumbrado, quando as transições de estado *booleano* foram enviadas pelo IED *publisher*, ou seja, os comandos enviados por mensagens antigas foram executados nos IEDs.

Observa-se portanto, uma vulnerabilidade nos IEDs *subscribers* que, se submetidos ao ataque realizado, poderiam executar *TRIPs* e acionar lógicas digitais configuradas. O *pentest* realizado demonstrou que a rede de comunicação da subestação necessita de ferramentas de proteção contra ataques de falsificação de mensagens.

4.2.2 Ataque à IEDs por Alterações no DATA-SET

Outro teste relacionado à ataques que um invasor com acesso à rede de comunicação da subestação pode realizar, consiste em alterar o DATA-SET de uma mensagem que trafega na rede. Para a realização deste teste, foi implementado, em linguagem C, uma aplicação para capturar a mensagem, verificar se a mensagem era GOOSE, e alterar um valor do DATA-SET de falso para verdadeiro. A implementação realizada foi executada, após a execução do teste entre os IEDs. A mensagem presente na rede é enviada pelo *publisher* com o DATA-SET contendo o valor falso, na cadência de situação estável do SE. Esta mensagem é capturada pelo programa em execução que a replica com o DATA-SET com valor verdadeiro. Neste caso, os IEDs *subscribers* não sinalizaram na IHM a realização de um evento. Supõe-se portanto, que os IEDs *subscribers* fazem a verificação dos campos do pacote GOOSE, para utilizar a mensagem.

Neste íterim, alterou-se o *software* para mudar também o campo *sqNum* da mensagem com o DATA-SET adulterado. Como seria uma situação de evento, o *sqNum* foi alterado

para zero. Observando o teste, percebeu-se que os IEDs também não sinalizaram, através do *Led* na IHM, a utilização da mensagem. Após este teste, o campo *stNum* também foi alterado, e recebeu o valor atual mais um. Com estas três modificações na mensagem, os IEDs *subscribers* passaram a utilizar a mensagem GOOSE adulterada, sinalizando no *Led* da IHM, conforme realizado, quando o envio do evento foi feito pelo IED *publisher*.

Com este teste, observou-se que os IEDs fazem o controle para uso da mensagem de execução de um evento, pelos campos *sqNum* e *stNum*, porém, esta análise não é suficiente para verificar se as mensagens são falsas.

Existem outros campos pertinentes às transições e cadências na mensagem GOOSE que podem contribuir para a detecção de um invasor. O *pentest* realizado identificou mais uma vulnerabilidade nos IEDs, fazendo com que eles utilizassem mensagens GOOSE adulteradas. O acesso de um *hacker* à rede de comunicação de dados da subestação, pode ter consequências catastróficas. O desenvolvimento de códigos maliciosos que realizem os *pentests* feitos é uma situação possível de acontecer, a julgar pelos casos dos vírus Stuxnet, Duqu e Flame que são *malwares* desenvolvidos para atacar sistemas de controle industriais específicos (KUSHNER, 2013). Para mitigar tais ataques, ferramentas de segurança particulares para este tipo de mensagem devem ser implementadas.

O IDS desenvolvido necessita de uma plataforma para execução que seja robusta o suficiente, para realizar os processamentos necessários em baixas latências, possua interfaces de comunicação que viabilizem a troca de mensagens, através da rede de comunicação *Ethernet*, e seja flexível para possibilitar a evolução da pesquisa.

4.3 Plataforma de Desenvolvimento do Sistema Embarcado

A plataforma escolhida para o desenvolvimento de um sistema capaz de detectar e bloquear ataques de falsificação de mensagens, como os identificados no *pentest*, foi uma *Zynq Board* (ZYBO), que possui capacidade de processamento e recursos computacionais que viabilizaram a implementação do IDS para mensagens GOOSE. A placa ZYBO é usada para o desenvolvimento de *software* embarcado e circuitos digitais, composta por uma FPGA Xilinx Z-7010, que se baseia na Artix-7, integrada a um processador ARM Cortex-A9, por um conjunto multimídia e conectividade de periféricos (Figura 16).

Dentre os recursos disponíveis nesta plataforma estão (Digilent (2014)):

- ❑ processador dual-core ARM Cortex-A9 de 650Mhz;
- ❑ memória de 512 *Mega Bytes (MB)* x32 *Double Data Rate* (DDR)3 com 1050 *Megabits* por segundo (*Mbps*) de largura de banda e com 8 canais *Direct Memory Access* (*DMA*);

Figura 16 – Hardware ZYBO

Fonte: Digilent (2014)

- ❑ periféricos de alta largura de banda: Ethernet 1 *Gigabits* por segundo (*Gbps*); *Universal Serial Bus* (USB) 2.0; *Secure Digital Input Output* (SDIO);
- ❑ periféricos de baixa largura de banda: *Serial Peripheral Interface* (SPI); *Universal Asynchronous Receiver/Transmitter* (UART); *Controller Area Network* (CAN); *Inter-Integrated Circuit* (I2C);
- ❑ a FPGA Z-7010 possui 4.400 *slices* lógicas, cada um com quatro *Look Up Tables* (LUTs) de 6 entradas e 8 flip-flops, 240 KB de memória *Random Access Memory* (RAM) rápida, 80 fatias *Digital Signal Processor* (DSP), com velocidade de clock interno de 450 *Megahertz* (MHz) e conversor analógico digital;
- ❑ porta VGA de 16 bits por pixel;
- ❑ slot MicroSD;
- ❑ codec de áudio com fone, microfone e linha;
- ❑ *General Purpose Input/Output* (GPIO) com 6 botões, 4 comutadores e 5 LEDs.

Seus recursos viabilizam o desenvolvimento de *softwares* para diversas aplicações, possibilitando, inclusive, a utilização de um SO embarcado.

4.3.1 Sistema Operacional Linux Embarcado

Sistema Operacional é um *software* que faz a interface entre o usuário e o microcomputador. Sua principal função é controlar o funcionamento do computador, gerenciando a utilização e o compartilhamento dos recursos como processador, memória e dispositivos de entrada e saída. Ele simplifica e agiliza o trabalho do programador, uma vez que as aplicações irão interagir com o SO e não com o *hardware*. O uso de um sistema operacional embarcado possui uma série de vantagens para o desenvolvimento de sistemas, uma vez que oportuniza a implementação de programas, utilizando linguagens de alto nível, aumenta a portabilidade do sistema, possibilitando movê-lo para outros dispositivos ou arquiteturas, e viabiliza o uso das ferramentas disponíveis nos SOs. Alternativamente, ainda se pode configurar um sistema operacional de tempo real que oferece previsibilidade na execução das tarefas programadas.

O Linux é um SO gratuito, com código fonte aberto e desenvolvido de maneira cooperativa. Para se formar um SO completo e utilizável, o *kernel*² Linux é empacotado junto com interpretadores de *scripts*, *daemons* e outros *softwares* dos quais se originam as distribuições Linux (NEMETH et al., 2007). Para a instalação e configuração do Linux, de forma que o mesmo seja executado na plataforma ZYBO, foi utilizado um cartão de memória MicroSD que foi dividido em duas partições. A primeira com 512 Mbytes e sistema de arquivos FAT32 para ser utilizado como partição de *boot*. A segunda com o tamanho restante do MicroSD e sistema de arquivos ext4 para ser a partição de *root* do sistema de arquivos montado pelo Linux (Figura 17).

Figura 17 – Sistema de arquivos utilizado no MicroSD para instalação do SO

Fonte: Crockett et al. (2014)

A partição de *boot* é composta por quatro arquivos utilizados na inicialização da placa:

²*Kernel* é o núcleo do sistema operacional. Ele controla todas as tarefas realizadas pelo computador (NEMETH et al., 2007).

1. BOOT.bin que contém três arquivos: o *bitstream* da FPGA, o *First Stage Boot Loader* (FSBL) e o *u-boot.elf*;
2. *devicetree.dtb*;
3. *uEnv.txt*;
4. *uImage*.

Para integrar as partições, inicialmente, criou-se o *bitstream* que contém a programação da FPGA. O projeto utilizado para a criação deste arquivo foi o *Zybo Base System Design*, encontrado em Digilent (2016). Neste projeto foi sintetizado, implementado e gerado o *bitstream*, para compor o arquivo BOOT.bin. A inclusão deste arquivo na inicialização da placa, possibilita a implementação de sistemas capazes de trocar informações entre o SO e a FPGA.

Ainda com o projeto *Zybo Base System*, os dados do *hardware* descrito neste projeto foram utilizados para gerar um arquivo FSBL. Este arquivo é um programa que tem a descrição do dispositivo que irá executar o *boot*. Ele inicializa os *hardwares* críticos e passa a execução para o segundo estágio de *boot* na memória RAM. O segundo estágio de *boot* é definido no arquivo *u-boot.elf*. U-Boot é um *bootloader* Linux universal *open source* que a Xilinx customizou para suas placas, sendo utilizado pela família de processadores Zynq-7000 (CROCKETT et al., 2014). Este *bootloader* foi modificado para carregar o arquivo *uEnv.txt* antes de iniciar o *boot*, possibilitando realizar configurações antes de carregar o SO.

O arquivo *bitstream* é utilizado para programar a FPGA e o arquivo *Executable Linkable Format* (ELF) para programar o processador ARM. Apesar de não se ter utilizado os recursos da FPGA neste projeto, o *bitstream* foi incluído no arquivo de *boot*, para permitir esta comunicação em trabalhos futuros. Os arquivos *bitstream*, FSBL e *u-boot.elf* formam o arquivo de inicialização BOOT.bin.

A informação do *hardware* que o Linux será inicializado está contida no arquivo *device-tree.dtb*. Para criar este arquivo, as configurações da placa como: modelo do processador, memória, placa de rede e demais periféricos são definidas em um arquivo *.dts* e compiladas para serem carregadas no *u-boot*. Por fim, o arquivo *uImage* contém o *kernel* do Linux compactado, o qual é descompactado, quando for carregado na memória secundária.

A organização de engenharia Linaro, trabalha no desenvolvimento de *software* livre e de código fonte aberto para diversas ferramentas e utilitários, com o objetivo de fornecer bases comuns para projetos de tecnologia. Esta organização é composta por um grupo de empresas que trabalham juntamente com a comunidade *open source*. Dentre os projetos em desenvolvimento, pode-se citar o aperfeiçoamento do *kernel* do Linux para dispositivos com processadores ARM (COMPANY, 2016).

Para compor a partição de *root* do sistema de arquivos, foi utilizado o *rootfs*³ da versão *developer* do Linux Ubuntu Linaro. Essa versão possui um console mínimo com ferramentas de desenvolvimento integradas por padrão, depuradores e instrumentos necessários para o desenvolvimento do *kernel*, bibliotecas e de sistemas. A Xilinx mantém uma versão do *kernel* do SO Linux voltada para os dispositivos da família Zynq chamada Zynq-Linux. O código fonte deste projeto é aberto e distribuído gratuitamente. Este *kernel* inclui um *Board Support Packages (BSP)* e *drivers* de dispositivos específicos para placas de desenvolvimento de sistemas como a ZYBO (CROCKETT et al., 2014).

Para dar início à configuração do *kernel* implementado neste projeto, foi utilizada uma versão pré-compilada para a execução do primeiro *boot* disponível em Xilinx (2016b). Realizado o acesso ao SO Linux, executando na placa ZYBO, foram realizadas as configurações de rede, atualização da lista de pacotes, sincronização do relógio e instalação das ferramentas utilizadas na configuração do *kernel*. Com o SO preparado, foi realizado o *download* do *kernel* linux-xlnx versão 4.4.0. (XILINX, 2016a).

A comunidade Linux mantém o desenvolvimento de um *patch* do *kernel* PREEMPT_RT, para possibilitar a implementação de sistemas que executem tarefas de tempo real. Esse *patch* modifica o código fonte do *kernel* Linux, viabilizando sua compilação para o desenvolvimento de sistemas de baixa latência e preempção de tarefas do SO (TS'O; HART; KACUR, 2016) (Figura 18).

Figura 18 – Comparação entre *kernel* preemptivo e não preemptivo

Um *kernel* preemptivo permite a interrupção de tarefas de menor prioridade, para a execução de tarefas de maior prioridade, diminuindo a latência de execução destas (LI; YAO, 2003).

Para permitir a implementação de sistemas com baixa latência e previsibilidade temporal, foi aplicado no *kernel* linux-xlnx o *patch* PREEMPT_RT (TS'O; HART; KACUR, 2016). Com o *patch* aplicado, foi possível customizar o *kernel* para possibilitar o desenvolvimento de sistemas de tempo real (Figura 19), bem como a seleção dos recursos necessários para o desenvolvimento do IDS.

³Raiz do sistema de arquivos Linux com pacotes diferenciados para cada versão (YAGHMOUR, 2003).

Figura 19 – Configuração do *kernel* para preempção.

Realizadas as devidas configurações, o *kernel* foi compilado, gerando um novo arquivo uImage. Os módulos e *firmwares* compilados, neste processo, foram instalados na partição *rootfs*. O arquivo do kernel compactado, foi copiado para a partição de *boot* de forma que, ao reiniciar a placa, o novo *kernel* foi carregado. Um sistema de tempo real não implica, necessariamente, que ele possua uma alta performance, mas que garante o tempo de resposta de uma determinada aplicação. Para o desenvolvimento da aplicação descrita no presente trabalho, esta garantia é essencial, sendo que o tempo de tráfego das mensagens GOOSE deve ser de no máximo 4 ms. Definida a plataforma de hospedagem do sistema e instalado o SO de tempo real, procedeu-se à determinação da estrutura de comunicação. As vulnerabilidades detectadas no *pentest* realizado demonstraram que existe a necessidade de identificação e bloqueio de mensagens GOOSE falsas, originadas de vírus, *worms* ou construídas e enviadas por *hackers*.

Em uma rede de comunicação empresarial, *firewalls* e *proxys* protegeriam a entrada na rede e sistemas de detecção de intrusão iriam identificar o intruso na rede interna, conforme se pode ver na Figura 12. Essas ferramentas também devem ser utilizadas na rede da subestação, porém, caso não sejam desenvolvidos instrumentos específicos para este tipo de tráfego, as ferramentas empresariais, fatalmente, irão falhar na detecção deste tipo de ataque.

Outro agravante para a identificação e o bloqueio do intruso são os atrasos proporcionados pelo processamento, identificação e bloqueio da mensagem forjada. Portanto, utilizando o SO de tempo real descrito, realizou-se o desenvolvimento do sistema de detecção e bloqueio de mensagens GOOSE, utilizando SDN por meio do protocolo *OpenFlow*.

4.4 Redes Definidas por *Software* Protocolo *OpenFlow*

As redes de comunicação de dados são cruciais para a evolução das redes de energia elétrica, dada a evolução proposta pelas *smart grids* alavancada pela padronização realizada com o padrão IEC-61850. Em uma infraestrutura de rede tradicional, as aplicações são implementadas pelos fabricantes e embarcadas nos equipamentos. Fica a cargo dos administradores de rede a realização de configurações limitadas aos recursos pré-programados nos equipamentos. Pesquisas sobre redes de comunicação de dados geram pouco impacto nos produtos finais. A grande quantidade de protocolos, equipamentos instalados e a dificuldade de experimentar novas ideias em configurações suficientemente realistas faz acreditar que a infraestrutura de rede está "ossificada". Segundo McKeown et al. (2008), a comunidade científica tem trabalhado para desenvolver redes de comunicação de dados programáveis, chamadas SDN. Estas redes programáveis controlam *switches* e roteadores que podem processar pacotes para múltiplas redes, simultaneamente. Seu objetivo é impulsionar as pesquisas nas redes de comunicação, possibilitando a experimentação de novos protocolos, em configurações suficientemente realistas para ganhar a confiança necessária para sua implantação.

O protocolo *OpenFlow* é a primeira interface de comunicação padrão para SDNs. Ele possibilita uma nova forma de funcionamento dos equipamentos de rede (Figura 20).

Figura 20 – Estrutura de funcionamento de um *switch* tradicional e um *switch OpenFlow*

Fonte: Lopes (2013).

Em 2011, as empresas Deutsche Telekom, Facebook, Google, Microsoft, Verizon, e Yahoo! lançaram a *Open Networking Foundation* (ONF), organização que tem como principal objetivo repensar a rede de forma colaborativa, para acelerar a entrega e comercialização de soluções SDN abertas (PITT et al., 2016). Dentre os principais projetos desta organização está a contínua evolução do protocolo *OpenFlow*, o qual permite dissociar o controle da função de transmissão de dados da rede, na qual o controlador é diretamente programável. O resultado é uma arquitetura dinâmica, controlável e adaptável que oferece aos administradores programação, automação e controle, sem precedentes.

O *OpenFlow* define as regras de comunicação entre o controle, implementado em servidores e o encaminhamento de pacotes realizado por *switches*. Com isso, os pacotes de

rede continuarão a ser encaminhados na taxa da linha, obedecendo aos encaminhamentos configurados na tabela de encaminhamento do *switch* e a lógica de decisão passa a ser implementada em um servidor utilizando o controlador *openflow* que se comunica com o *switch*, através de um canal de comunicação seguro (Figura 21).

Figura 21 – *Switch Openflow* e sua comunicação com o controlador

Fonte: Foundation (2009).

Para prevenir ataques ao protocolo, recomenda-se que a conexão entre controlador e *switch* seja realizada, através de um canal de comunicação seguro, utilizando criptografia *Secure Socket Layer* (SSL), embora seja possível, também, conectar-se por via do *Transmission Control Protocol* (TCP). O protocolo *OpenFlow* sustenta três tipos de mensagens (FOUNDATION, 2009):

1. *controller-to-switch* - são mensagens iniciadas pelo controlador usadas para gerenciamento e inspeção do estado do *switch*. Elas podem ser empregadas para atribuir ou solicitar configurações do *switch*, coletar estatísticas da tabela de fluxos, portas e de fluxos individuais, adicionar, modificar ou remover fluxos e enviar pacotes por portas específicas do *switch*;
2. *asynchronous* - são mensagens iniciadas pelo *switch* usadas para avisar ao controlador sobre a chegada de um pacote, a ocorrência de eventos, as mudanças no estado do *switch*, ou a ocorrência de erros. As quatro principais mensagens são: *Packet-in*, *Flow-removed*, *Port-status* e *Error*;
3. *symmetric* - são mensagens que podem ser iniciadas por ambos, o controlador ou o *switch*, sem qualquer solicitação entre eles. As mensagens mais comuns são: *Hello* e *Echo request and reply*.

A comunicação entre controlador e *switch OpenFlow* pode ser configurada como reativa ou pró-ativa. Uma configuração reativa configura os fluxos sob demanda, o que gera tráfego de controle na rede, porém, simplifica o processo de configuração dos equipamentos. Uma configuração pró-ativa consiste em realizar o registro dos fluxos no *switch*, no início da comunicação, para que o equipamento possua as ações pré-configuradas na sua tabela de fluxo, melhorando o desempenho geral do sistema (FERNANDEZ, 2013).

4.5 Switch OpenFlow

O padrão *OpenFlow* explora um conjunto de funções executadas em *switches* e roteadores fornecendo um protocolo aberto para programar sua tabela de fluxos⁴ permitindo controlar fluxos, escolher as rotas que os pacotes seguem, bem como o tratamento que recebem. Desta forma, pesquisadores podem experimentar novos protocolos de roteamento, modelos de segurança, esquemas de endereçamento e alternativas ao *Internet Protocol* (IP) (MCKEOWN et al., 2008).

Um *switch OpenFlow* é composto por três partes:

1. a tabela de fluxo, com uma ação associada a cada entrada de fluxo;
2. um canal seguro, que liga o *switch* ao controlador remoto;
3. o protocolo *OpenFlow* que fornece um padrão aberto, através do qual podem ser definidas entradas, na tabela de fluxo.

Os fluxos da tabela dos *switches* são limitados pela capacidade de implementação de cada tabela. Um fluxo pode ser definido por uma conexão TCP, por pacotes de um determinado endereço IP ou MAC, por pacotes com a mesma VLAN, por pacotes da mesma porta do *switch*. A versão 1.0 do protocolo *OpenFlow*, implementada pelo *switch* utilizado na presente pesquisa, possui doze campos no cabeçalho do pacote para identificação dos fluxos (Figura 22).

Figura 22 – Tabela de fluxos do *OpenFlow* versão 1.0

Fonte: Foundation (2009).

Cada fluxo de entrada tem uma ação associada que pode ser (MCKEOWN et al., 2008):

- a) encaminhar pacotes deste fluxo para uma determinada porta;
- b) encapsular e encaminhar pacotes deste fluxo para um controlador. Normalmente utilizado para o primeiro pacote de um novo fluxo, de modo que o controlador

⁴A maioria das *switches* e roteadores *Ethernet* possuem tabelas de encaminhamento construídas a partir de *Ternary Content Addressable Memories* (TCAMs) que são utilizadas para a instalação das tabelas de fluxo (JIANG, 2013).

decida se o novo fluxo deve ser acrescentado à tabela de fluxos. Em algumas pesquisas, pode ser utilizado para transmitir todos os pacotes para um controlador de processamento;

- c) descartar pacotes deste fluxo;
- d) encaminhar pacotes deste fluxo, através do *pipeline* de processamento normal do *switch*. Esta ação é habilitada em *switches* híbridos e pode ser utilizada para dividir o tráfego experimental do tráfego de produção.

Uma entrada na tabela de fluxos tem três campos (MCKEOWN et al., 2008):

1. cabeçalho do pacote que define o fluxo;
2. ação que define como os pacotes devem ser processados; e
3. estatísticas que mantêm o controle do número de pacotes e *bytes* para cada fluxo e o tempo decorrido, desde o último pacote deste fluxo, para ajudar na remoção de fluxos inativos.

Um fluxo pode ser especificado por todos os campos da tabela de fluxos dependendo do protocolo em uso. Cada campo do cabeçalho pode ter um valor polivalente para permitir a agregação dos fluxos ou mesmo, um fluxo pode ser definido por apenas um valor, por exemplo, a VLAN ID, sendo o mesmo aplicado a todo o tráfego em uma VLAN particular. O processamento das mensagens no *switch OpenFlow* versão 1.0 do protocolo, ocorre, conforme ilustra a Figura 23.

Figura 23 – Fluxo de funcionamento do protocolo *OpenFlow* versão 1.0 no *switch*

Fonte: Foundation (2009).

O primeiro procedimento realizado ao receber o pacote da rede é processar o protocolo *Spanning Tree*. O *OpenFlow* oferece, também, suporte opcional ao protocolo STP (IEEE,

2004). Caso o *switch* implemente esta funcionalidade, fica obrigatória a verificação deste protocolo, antes de iniciar a decodificação dos campos. O processo de decodificação do cabeçalho dos campos depende do tipo do pacote. Na versão 1.0 do *OpenFlow*, os valores dos doze campos da tabela de fluxo, de acordo com o cabeçalho da mensagem, podem ser extraídos do pacote para identificação na tabela de fluxos.

O passo seguinte corresponde à comparação da informação extraída do cabeçalho da mensagem, com as regras dos fluxos cadastrados. Caso a mensagem se enquadre em algum dos fluxos presentes na tabela de encaminhamento, as ações associadas ao referido fluxo são executadas. Caso contrário, a mensagem é enviada ao controlador, para que ele decida a ação a ser tomada com a mensagem.

4.5.1 Controlador

O controlador é uma plataforma para o desenvolvimento de aplicações que irá monitorar e tomar decisões quanto ao funcionamento do *switch*. Segundo (MCKEOWN et al., 2008) o controlador *OpenFlow* atua como um SO de rede, oferecendo uma interface de alto nível para o desenvolvimento de aplicações que utilizam o *hardware* de rede, em uma analogia ao SO dos computadores. Diversos controladores *OpenFlow* estão disponíveis para a implementação de aplicações, dentre os quais incluem-se:

- ❑ NOX: desenvolvido em linguagem C pela Nicira⁵ foi o primeiro controlador *OpenFlow*. Em 2008 a Nicira doou o NOX para a comunidade e, desde então, ele tem sido a base para muitos e variados projetos de pesquisa na exploração do SDN (MCCAULEY, 2014).
- ❑ POX: considerado sucessor do controlador NOX desenvolvido em Python. Na sua essência, é uma plataforma para o rápido desenvolvimento e prototipagem de *software* de controle de rede (MCCAULEY, 2014).
- ❑ Ryu: fluxo em japonês, é um *framework* SDN desenvolvido em Python que sustenta diversos protocolos de gerenciamento de dispositivos de rede, dentre eles o *OpenFlow* (RYU, 2014).
- ❑ Beacon: criado em 2010, é um controlador *OpenFlow* baseado em Java largamente utilizado por professores, pesquisadores e como base para o controlador Floodlight (ERICKSON, 2013).
- ❑ Floodlight: é um controlador *OpenFlow* desenvolvido em Java para trabalhar com um grande número de *switches*, roteadores, *switches* virtuais e *access points* com uma interface simples para construir e executar aplicações (FLOODLIGHT, 2016).

---

⁵Empresa fundada em 2007 por Martin Casado, Nick McKeown e Scott Shenker que criou versões do *OpenFlow* e *Open vSwitch*. Em agosto de 2012 foi comprada pela *VMware* (WILLIAMS, 2012).

- ❑ Maestro: é um controlador *OpenFlow* desenvolvido em Java que mantém um modelo de programação *single-threaded*, mas permite e gerencia o paralelismo, como serviço aos programadores de aplicações (CAI; COX; NG, 2010);
- ❑ OpenMUL: é uma plataforma de controle e gerenciamento de redes que faz a ponte entre os recursos virtualizados e a rede física. Implementado em linguagem C é projetado para alto desempenho e confiabilidade; indicado para implementação de redes de missão crítica (SAIKIA; MALIK, 2014).

O controlador utilizado na implementação realizada na presente pesquisa foi o *Open-Mul*. Ele fornece uma *Application Interface* (API) multi-nível e multi-linguagem para sustentar diferentes necessidades das aplicações. A linguagem C é utilizada para aplicações sensíveis ao desempenho e à latência. A linguagem Python pode ser utilizada para o desenvolvimento rápido de aplicações, enquanto a API RESTful⁶ pode ser utilizada para aplicações na Web. Suas principais ferramentas são apontadas na Tabela 2, a seguir (SAIKIA; MALIK, 2014):

Tabela 2 – Suite de aplicativos do OpenMul.

Fonte: Saikia e Malik (2014).

Ferramentas	Descrição
Protocolos implementados em baixo nível	<i>OpenFlow</i> 1.4, 1.3.x, 1.0, Ovsdb
Multi versão	Suporta diferentes versões simultaneamente
Taxa de download do fluxo	10,5 Milhões/segundo
FlexPlug	Oferece a flexibilidade de inserir ou remover qualquer aplicação
FirmFlow	Coerência de fluxos entre reinicializações de aplicação, controlador e <i>switch</i>
SSL	Suporte a TLS 1.2
Learning Module	Módulo L2 learning com IGMP <i>snooping</i>
<i>OpenFlow</i> multi modo	Pode operar com <i>switches</i> heterogêneos
Descoberta de Topologia	Descoberta automática de topologia
APIs	Conjunto rico de APIs, RESTful, Python e C
Estatísticas	Coleta de estatísticas do <i>OpenFlow</i>
Hot standby HA	O controlador disponível para atenuar quedas
Detecção de Loop	Detecção rápida de <i>loops</i> baseado no protocolo xSTP
Proteção de espaço de endereços	O controlador executa completamente independente de todas as demais aplicações
Suporte utilizando linha de comando	Utilizando o padrão industrial, possui uma interface de gerenciamento em linha de comando.

⁶*Representational State Transfer* é uma aplicação que fornece acesso a um serviço qualquer baseado na arquitetura da Web, utilizando as mensagens do protocolo HTTP.

A instalação do *OpenMul* no Linux embarcado foi realizada, utilizando os *scripts* da própria aplicação. Para possibilitar a utilização da biblioteca de tempo real e a criação de *posix threads*, as aplicações em linguagem C do *OpenMul* foram compiladas com as *flags -lrt e -pthread*. Instalado o controlador, baseou-se na aplicação *l2switch* para o desenvolvimento do sistema de detecção e bloqueio de mensagens GOOSE. A aplicação *l2switch* é executada como um módulo do kernel. Todos os avisos e mensagens de erro desta aplicação, são escritos no arquivo de *log* do SO localizado em */var/log/syslog*.

O *switch* utilizado na implementação realizada é um DM4100 da empresa DATACOM, modelo ETH20GT+4GC+L3 com 24 portas *Gigabit Ethernet* par trançado e 4 portas de fibra óptica SFP (*small form-factor pluggable transceptor*). Toda comutação de pacotes L2/L3 é executada sempre em *hardware* em velocidade *wire speed*, com matriz de comutação de até 152Gbit/s (DATACOM, 2015).

O *firmware OpenFlow* DataCom OF 1.0.8 foi instalado no *switch*. Este *firmware* implementa a versão 1.0 do protocolo *OpenFlow*. A solução consiste em um *hardware* especializado em tratamento e encaminhamento de pacotes em *wire-speed* baseado na série DM4100 de *switches* de alta capacidade da DataCom (DATACOM, 2014).

A implementação é híbrida, na qual podem-se agrupar as portas do *switch* em VLANs destinadas para cada tipo de acesso. As portas que estiverem com o *openflow* habilitado devem pertencer à VLAN destinada ao *OpenFlow*. Para que um pacote seja encaminhado através das portas *OpenFlow*, o *switch* deve estar conectado a um controlador que indica o caminho, através dos fluxos no *switch*. Para a conexão entre *switch* e controlador é necessária a realização de configurações no *switch* tais como (DATACOM, 2014):

- ❑ IP porta e tipo de conexão (TCP ou SSL) com o controlador *OpenFlow*;
- ❑ tipo dos contadores utilizados nos fluxos, por pacotes ou *bytes*;
- ❑ prioridade dos filtros do *OpenFlow*;
- ❑ *idle-timeout-mode*. Quando o *idle-timeout* do fluxo for menor que o configurado com *min-idle-timeout*, os fluxos podem ser ajustados ou rejeitados;
- ❑ *min-idle-timeout* configura o tempo mínimo para expiração por *idle timeout*;
- ❑ *max-pending-req* configura o tamanho da fila de requisições pendentes de modificação de fluxo (*Flow Mod*);
- ❑ modo proativo ou reativo refere-se à forma de operação do *OpenFlow*. No modo proativo, os pacotes que chegam ao *switch*, e não possuem um fluxo correspondente, são descartados. No modo reativo, os pacotes na mesma situação são enviados para o controlador em uma mensagem *Packet-In*;
- ❑ *native-vlan* corresponde à VLAN referente às portas *OpenFlow*. As portas configuradas como sendo *OpenFlow* farão, automaticamente, parte desta VLAN;

- ❑ *rate-limit* é a taxa máxima de Packet-Ins tratada no *switch* e enviada para o controlador;
- ❑ *strip-fcs* utilizada para configurar a presença ou a remoção do FCS (*frame check sequence*) do frame *ethernet* encapsulado nas mensagens de Packet-in;
- ❑ desabilitar o *spanning-tree*;
- ❑ configurar a fila de prioridades da CPU;
- ❑ limitar a banda de pacotes enviados a CPU.

O *switch* DATACOM possui uma porta de gerência que, geralmente, é utilizada para a realização de configurações no equipamento, podendo ser aproveitada para a conexão com o controlador. Neste trabalho, esta porta não foi utilizada para este fim, pois sua velocidade de conexão é 100 *Mbps*. Como a interface de rede da plataforma ZYBO possui capacidade de comunicação a 1 *Gbps*, aproveitou-se a oportunidade para criar um link entre controlador e *switch* nesta velocidade, sendo que o *switch* possui 24 portas *Ethernet* de 1 *Gbps*, possibilitando maior largura de banda nesta comunicação. A porta utilizada para conexão com o controlador não deve estar configurada como *OpenFlow*, ou seja, ela deve estar fora da VLAN utilizada para este fim. Para configurar a interface do *switch* para gerência e conexão com o controlador, foi criada uma VLAN específica, em que o IP de conexão foi configurado e o protocolo *OpenFlow* desabilitado. Para realizar a conexão entre *switch* e controlador, foi configurada uma rede lógica independente, utilizando conexão TCP na porta 6653. Esta porta, tornou-se padrão para o *OpenFlow* a partir da versão 1.4 do protocolo. Feitas as configurações, ao inicializar a aplicação *OpenMul*, a conexão entre *switch* e controlador é realizada, possibilitando a execução das lógicas de segurança implementadas.

4.6 Detecção de Mensagem Falsa por Análise dos Valores Esperados nos Campos da Mensagem GOOSE

Segundo IEC62351 (2007b), algumas contra-medidas aos ataques podem ser realizadas, observando, temporalmente, os valores esperados dos campos da mensagem GOOSE que informam a realização de configurações no equipamento e que caracterizam as transições dos estados booleanos e que modelam o comportamento de uma comunicação padrão entre equipamentos que se comunicam, utilizando mensagens GOOSE, conforme descrito na Seção 3.4.

Para a implementação de um modelo eficiente, capaz de detectar e bloquear mensagens falsas foi utilizado SDN. Aproveitando as funcionalidades de controle e autorização

do fluxo de dados no *switch OpenFlow*, foi projetado um *software*, para ser executado no controlador *OpenMul*, que identifica a mensagem GOOSE, realiza verificações de segurança e autoriza ou bloqueia o seu envio aos IEDs *subscriber*. Ao estabelecer uma conexão entre controlador e *switch*, o controlador, proativamente, cadastra na tabela de fluxos do *switch* uma entrada, especificando que o *switch* deve encaminhar todas as mensagens do tipo 0x88b8 GOOSE para o controlador. Neste caso, quando mensagens deste tipo chegarem no *switch*, elas serão encapsuladas em um pacote *OpenFlow Packet-in* e enviadas ao controlador. A aplicação *l2switch* do *OpenMul* foi modificada para identificar o tipo de mensagem GOOSE e executar as funções de verificação de segurança implementadas.

A identificação das mensagens falsas na rede de comunicação foi realizada, através dos campos que compõem o *goosePdu*. O *goosePdu* é definido pela *Abstract Syntax Notation* (ASN).1 *Basic Encoding Rules* (BER) como um padrão para a descrição de protocolos abertos. A ASN.1 fornece uma notação para definir a forma das mensagens, juntamente com regras de codificação que especificam os *bits* na comunicação, para qualquer mensagem que for definida utilizando uma sequência *Type Length Value* (TLV) (DUBUISSON, 2001). A Figura 24 apresenta um trecho da especificação ASN.1 BER para mensagens GOOSE (IEC61850 (2011)).

```

IEC61850 DEFINITIONS ::= BEGIN
IMPORTS Data FROM ISO-IEC-9506-2
IEC 61850-8-1 Specific Protocol ::= CHOICE {
    gseMngtPdu      [APPLICATION 0] IMPLICIT GSEMngtPdu,
    goosePdu        [APPLICATION 1] IMPLICIT IECGoosePdu,
    ... }

IECGoosePdu ::= SEQUENCE {
    gocbRef          [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,
    }

UtcTime ::= OCTETSTRING – format and size defined in 8.1.3.6.

END

```

Figura 24 – Especificação ASN.1 BER de mensagens GOOSE.

Fonte: IEC61850 (2011).

Baseado no conceito intitulado TLV, tipo, comprimento e valor, todas as partes da mensagem são codificadas com um campo identificador da informação; um campo tama-

nho que delimita o comprimento do valor, e, em seguida, o valor em uma codificação que determina a informação a ser enviada. Este padrão, prevê extensibilidade, utilizando uma abordagem generalizada descrita na ASN.1 BER, para cobrir grupos de grupos a qualquer profundidade, conforme ilustrado pela Figura 25.

Figura 25 – Abordagem TLV de parâmetros e grupos.

Fonte: Dubuisson (2001).

Para o desenvolvimento de uma aplicação que faz a leitura dos campos que compõem o *goosePdu*, foi utilizada a biblioteca de código fonte aberto ASN1C implementada por Walkin (2003). A biblioteca gera código em linguagem C, atendendo as especificações do código ASN.1. A partir do código C gerado pela biblioteca, foram implementadas funções no controlador *OpenMul*, para identificar as mensagens GOOSE, decodificar os campos desta mensagem, realizar verificações de segurança e classificá-las como verdadeira ou falsa. As mensagens verdadeiras são encaminhadas para os IEDs *subscribers*, as mensagens falsas são descartadas no controlador.

O controlador *OpenMul* possui uma aplicação chamada *l2switch*, que aprende os endereços MAC dos dispositivos que se comunicam, através do *switch* controlado, e preenche sua tabela de fluxos, relacionando o endereço MAC de destino com a porta de destino dos pacotes, para que os novos pacotes sejam encaminhados diretamente para o referido dispositivo. Esta aplicação foi modificada para identificar a mensagem GOOSE, através do tipo 0x88b8 e executar a função de verificação de segurança, enviando como parâmetros: o pacote de rede, sem o cabeçalho *Packet-in* do *OpenFlow*, o tamanho deste pacote e um identificador do *switch*.

O código C com as funções de verificação de segurança e o gerado pela biblioteca ASN1C foram incluídos na biblioteca do *OpenMul*, de forma que, pela inclusão do cabeçalho na aplicação *l2switch*, foi possível sua execução. Inicialmente, a função decodifica os campos fixos da mensagem GOOSE, desde o endereço MAC de destino, até o campo *Reserved 2*; posteriormente, utiliza-se a função *ber_decode* do código gerado pela biblioteca ASN1C, para decodificar o *goosePdu*. Com todos os campos identificados, iniciou-se a verificação de segurança.

Para a realização de uma análise temporal das mensagens GOOSE na rede, foi cri-

ado um *buffer* circular que armazena os dados da mensagem e uma estampa de tempo, para cada *switch* conectado ao controlador e para cada IED *publisher* conectado na rede Figura 26.

Figura 26 – *Buffer* circular GOOSE.

As mensagens GOOSE são capturadas e armazenadas em sequência no *buffer* circular, identificadas pelo campo GoCBRef e pelo *switch* a que ela pertence. Os valores dos campos da primeira mensagem são utilizados como referência para a análise temporal. De posse destes dados, foram desenvolvidas algumas funções para comparar os valores dos campos das mensagens GOOSE no *buffer*. A lógica implementada para verificação dos campos da mensagem GOOSE possui três partes principais:

- ❑ situação normal de operação;
- ❑ situação de evento no SE;
- ❑ situação de parada ou configuração.

Quando um IED está configurado para enviar mensagens GOOSE utilizando o protocolo IEC-61850, são enviadas, periodicamente, mensagens à rede de comunicação com o DATA-SET em estado inicial, informando que o SE está em estado normal de operação. A norma IEC-61850 e os testes realizados demonstraram que, na composição da mensagem GOOSE, existem campos que seguem uma cadência e outros que assumem valores que não devem ser alterados, que são uma situação típica deste estado de execução do SE.

Quando ocorre algum evento, juntamente com o dado que é enviado no DATA-SET, alguns campos são alterados, caracterizando a mudança realizada na mensagem, informando que decisões devem ser tomadas, diante da referida ocorrência. Neste caso, através

dos valores esperados nestes campos, a situação de evento é caracterizada. Para ambas as situações, outros campos não devem ser alterados, tendo em vista que são campos que caracterizam o emissor, a VLAN através da qual está ocorrendo a comunicação, a formação do DATA-SET e as configurações que a mensagem recebeu. Neste caso, uma nova verificação é realizada, para analisar se não existem mensagens na rede com valores diferentes do esperado.

Para que estes valores alterem, a comunicação deve ser interrompida, caracterizando outra situação pertinente de verificação que é analisada em paralelo às situações supracitadas. Cabe ressaltar que esta situação de início da comunicação, ou de parada para realização de configurações deve ocorrer e, logo em seguida, a mensagem posterior deve ser enquadrada em situação normal de operação ou de evento no SE.

A Figura 27 ilustra o fluxo de execução do algoritmo implementado.

Figura 27 – Fluxo de verificação da mensagem GOOSE.

Para a implementação da lógica ilustrada no fluxograma, 5 campos da mensagem GOOSE merecem destaque: *stNum*, *sqNum*, *timeAllowedtoLive*, *T* e o DATA-SET, campos cujos valores variam, de acordo com as situações citadas na Tabela 3.

Cada situação de comunicação descrita no SE é caracterizada pelos valores apresentados para os campos descritos na Tabela 3. As demais verificações são necessárias para identificar mensagens que não pertencem a esta comunicação ou tentativas de modificações das mensagens. Tais análises são especialmente importantes para a detecção do intruso. A lógica implementada tem como princípio fundamental identificar situações possíveis de acontecer. Caso os campos da mensagem GOOSE assumam valores em descompasso com o esperado, a mensagem é classificada como falsa, sendo descartada no controlador. As subseções seguintes detalham as diferentes etapas do fluxo de execução da Figura 27.

Tabela 3 – Relação dos valores esperados nos campos da mensagem GOOSE com a situação de funcionamento do SE.

Campos	normal	evento	parada ou configuração
<i>stNum</i>	sem alterações	anterior + 1	1
<i>sqNum</i>	anterior + 1	0	1
<i>timeAllowedtoLive</i>	tempo atual - tempo anterior $\leq$ <i>timeAllowedtoLive</i>	tempo atual - tempo anterior $\leq$ <i>timeAllowedtoLive</i> máximo	tempo atual - tempo anterior $>$ <i>timeAllowedtoLive</i>
<i>T</i>	sem alterações	= tempo atual	não verificar
DATA-SET	sem alterações	modificado	não verificar

4.6.1 Situação Normal de Operação

O Algoritmo 1 descreve a lógica implementada no método de verificação da situação normal de operação.

Algoritmo 1 Verifica situação normal de operação no SE.

Entrada: *GOOSEanterior*, *GOOSEatual*

Saída: Verdadeiro ou Falso

1. $tempo \leftarrow (atual.timeStamp - anterior.timeStamp)$
  2. $timeAllowedtoLive \leftarrow (tempo \leq atual.timeAllowedtoLive)$
  3. $t \leftarrow (anterior.t == atual.t)$
  4. $stNum \leftarrow (anterior.stNum == atual.stNum)$
  5. $sqNum \leftarrow ((anterior.sqNum + 1) == atual.sqNum)$
  6. **Se** ($timeAllowedtoLive$ **E** t **E** $stNum$ **E** $sqNum$ **E** Não MudouDataSet) **Então**
  7.   Verdadeiro
  8. **Se Não**
  9.   Falso
  10. **Fim Se**
-

A situação normal de operação é caracterizada pela ausência de mudanças no DATA-SET. As mensagens são replicadas na rede de comunicação, porém, com os mesmos valores anteriores. Alguns campos caracterizam esta situação, como o campo *stNum* que não deve ser alterado e o campo *sqNum* que deve possuir o valor anterior acrescido de um. O campo *T* que estampa o tempo de ocorrência da última transição não pode ser diferente do seu valor anterior. O tempo atual de chegada da mensagem GOOSE subtraído do tempo de chegada da última mensagem deve ser menor ou igual ao valor do campo *timeAllowedtoLive*. Os testes realizados no método que verifica se a mensagem GOOSE pertence a uma situação de comunicação em estado normal do SE devem ser todos verdadeiros, para que receba tal classificação.

4.6.2 Situação de Evento no SE

Assim como ocorreu na situação normal de operação do SE, em condições de evento, os testes realizados nos valores temporais dos campos caracterizam este cenário (Algoritmo 2).

Algoritmo 2 Verifica situação de evento no SE.

Entrada: *GOOSEanterior*, *GOOSEatual*

Saída: Verdadeiro ou Falso

1. $tempo \leftarrow (atual.timeStamp - anterior.timeStamp)$
  2. $timeAllowedtoLive \leftarrow (tempo \leq atual.timeAllowedtoLive)$
  3. $t \leftarrow (anterior.t == atual.t)$
  4. $stNum \leftarrow ((anterior.stNum + 1) == atual.stNum)$
  5. $sqNum \leftarrow (anterior.sqNum == 0)$
  6. **Se** ($timeAllowedtoLive$ **E** t **E** $stNum$ **E** $sqNum$ **E** *MudouDataSet*) **Então**
  7.   Verdadeiro
  8. **Se Não**
  9.   Falso
  10. **Fim Se**
-

A situação de evento no SE é caracterizada por mudança nos valores do DATA-SET. Quando isso acontece, o tempo de cadência das mensagens e o valor do campo *timeAllowedtoLive* são diminuídos. O campo *stNum* recebe o seu valor anterior acrescido de um. O campo *sqNum* passa a ter o valor zero. O campo *T* recebe o tempo atual de envio da mensagem.

4.6.3 Situação de Verificação para Ambas as Situações

Para qualquer das situações, os campos da VLAN não podem alterar, sem que a comunicação seja interrompida e todo o processo de replicação das mensagens GOOSE seja reiniciado. Tratando-se do trecho da mensagem chamado *Ethernet Protocol Data Unit* (PDU), o campo *Application Identifier* (APPID) não pode alterar. Sobre o cabeçalho da aplicação *Application Protocol Data Unit* (APDU) intitulado *goosePdu*, os campos *GOOSE Control Block* (GoCB)Ref, *DatSet*, *goID*, *test*, *confRev* e *Needs Commissioning* (ndsCom) também não podem ter alterações, sem a interrupção da replicação das mensagens. Neste teste, todos os campos mencionados, devem seguir a lógica exemplificada pelo algoritmo 3.

A cada mensagem classificada como estado normal de operação, ou situação de evento, os valores dos campos supracitados são comparados com seus respectivos valores na mensagem anterior. Desta forma, propõe-se garantir que eles não tenham sido alterados, sem a interrupção da comunicação, o que caracterizaria uma mensagem falsa na rede.

Outra análise pertinente se faz quanto ao tamanho da mensagem, que deve ser realizada para ambas as situações. A mensagem GOOSE possui um campo *Length* que

---

Algoritmo 3 Verifica os campos que não devem modificar.

Entrada: *GOOSEanterior*, *GOOSEatual*

Saída: Verdadeiro ou Falso

1. $vlan \leftarrow (atual.vlan == anterior.vlan)$
  2. $appId \leftarrow (atual.appId == anterior.appId)$
  3. $gocbRef \leftarrow (atual.gocbRef == anterior.gocbRef)$
  4. $datSet \leftarrow (atual.datSet == anterior.datSet)$
  5. $goId \leftarrow (atual.goId == anterior.goId)$
  6. $test \leftarrow (atual.test == anterior.test)$
  7. $confRev \leftarrow (atual.confRev == anterior.confRev)$
  8. $ndsCom \leftarrow (atual.ndsCom == anterior.ndsCom)$
  9. $length \leftarrow (atual.length == tamanho)$
  10. $numDatSetEntries \leftarrow (atual.numDatSetEntries == getSizeAllData(atual))$
  11. **Se** ($vlan$ **E** $appId$ **E** $gocbRef$ **E** $datSet$ **E** $goId$ **E** $test$ **E** $confRev$ **E** $ndsCom$
 $length$ **E** $numDatSet$) **Então**
  12. Verdadeiro
  13. **Se Não**
  14. Falso
  15. **Fim Se**
-

armazena o número de octetos, do *APPID* ao DATA-SET, no final do pacote. Caso este valor não esteja consistente, a referida mensagem está mal formada, indicando que o pacote não foi criado por um dispositivo autorizado a realizar a comunicação. O campo *numDatSetEntries* especifica o número de DATA-SETs que formam a mensagem. A comparação entre a quantidade de DATA-SETs e o valor do campo *numDatSetEntries* pode indicar uma má formação da mensagem GOOSE indicativo de mensagem adulterada.

4.6.4 Situação de Início ou Parada da Comunicação

Para a situação de início ou parada da comunicação, os campos da mensagem GOOSE devem atender à lógica descrita no algoritmo 4.

Algoritmo 4 Verifica situação de início ou parada na comunicação.

Entrada: *GOOSEanterior*, *GOOSEatual*

Saída: Verdadeiro ou Falso

1. $tempo \leftarrow (atual.timeStamp - anterior.timeStamp)$
  2. $timeAllowedtoLive \leftarrow (tempo > atual.timeAllowedtoLive)$
  3. $stNum \leftarrow (atual.stNum == 1)$
  4. $sqNum \leftarrow (atual.sqNum == 1)$
  5. $confRev \leftarrow (atual.confRev >= anterior.confRev)$
  6. **Se** ($timeallowedtoLive$ **E** $stNum$ **E** $sqNum$ **E** $confRev$) **Então**
  7. Verdadeiro
  8. **Se Não**
  9. Falso
  10. **Fim Se**
-

Dois dos campos da mensagem GOOSE que caracterizam uma situação de início ou interrupção da comunicação, para a realização de configurações ou manutenção nos equipamentos são o *stNum* e o *sqNum*. Estes campos iniciam com o valor 1, toda vez que a comunicação é interrompida. Outro campo que caracteriza o início ou a parada da comunicação é o *timeAllowedtoLive*. Este campo define o tempo de espera, até a próxima mensagem. Para este caso, a diferença entre o tempo atual e o tempo de chegada da última mensagem deve ser maior que o seu valor, indicando que o tempo de espera para a próxima replicação foi expirado, necessitando de um novo reinício. Para o caso de realização de configurações no equipamento, o campo *confRev* deve possuir um valor maior que o anterior, caso contrário, seu valor deve permanecer o mesmo.

Para que a mensagem seja classificada como situação de início ou parada da comunicação para configuração, todos os testes supracitados devem ter verdadeiro como resposta.

4.6.5 Análise da Segurança

Finalmente, o algoritmo 5 descreve a lógica final para a análise de segurança das mensagens GOOSE.

Algoritmo 5 Lógica final para análise de segurança nas mensagens GOOSE.

Entrada: *GOOSEanterior, GOOSEatual*

Saída: Verdadeiro ou Falso

1. **Se** (((*SituaçãoNormal*(*anterior*, *atual*) **Ou** *SituaçãoEvento*(*anterior*, *atual*))
    E *SituaçãoComum*(*anterior*, *atual*)) **Ou** *ParadaComunicação*(*anterior*, *atual*))
    Então
  2.   Verdadeiro
  3. **Se Não**
  4.   Falso
  5. **Fim Se**
-

Neste íterim, as mensagens GOOSE encaminhadas pelos *switches* da rede *OpenFlow* são analisadas individualmente, para cada IED *publisher*, comparando os valores de seus campos com os da sua antecedente, para a realização da segurança descrita. Cabe ressaltar que, para não perder a sequência temporal da mensagem verdadeira, a mensagem classificada como falsa não é inserida no *buffer* circular.

4.6.6 Validação da Lógica Implementada

Para validar o *software* desenvolvido, foram organizados novos ambientes de teste para analisar a eficiência da aplicação. Baseado nos *pentests* realizados na Seção 4.2, foram executados novos ataques aos equipamentos que possuem o protocolo IEC-61850 no LSEE. Os testes foram divididos em duas partes: o primeiro envolvendo IEDs reais em uma subestação pequena, o segundo utilizando uma rede emulada para uma grande subestação.

4.6.6.1 Rede de Comunicação Envolvendo IEDs em uma Subestação Pequena

O conjunto de testes utilizado na validação da aplicação foi o *pentest* apresentado na Seção 4.2. Para a realização deste teste, foram utilizados o *Real Time Digital Simulator* (RTDS)⁷ e três IEDs. Foi incluído na rede de comunicação, um microcomputador utilizando SO usuário, com *softwares* analisadores de rede e compiladores para linguagens de alto nível. Foi interligada, nesta mesma rede, a plataforma ZYBO com o controlador *OpenMul* executando a aplicação *l2switch* modificada para analisar a segurança dos pacotes (Figura 28).

Figura 28 – Novo arranjo laboratorial criado para realização dos *pentests*.

No primeiro cenário, um IED foi configurado como *publisher* e os outros dois como *subscribers* da comunicação. A lógica de geração de pulsos foi colocada em execução, onde foi possível acompanhar o andamento do teste, observando o *led* dos IEDs *subscribers*. O controlador foi configurado para gravar, no arquivo de *log* do SO, a classificação realizada para cada uma das mensagens. Para verificar a qualificação dada pelo controlador às mensagens, foi examinado o arquivo de *log*. Após o fim da execução da lógica de geração de pulsos, as mensagens capturadas no microcomputador foram replicadas na rede, utilizando o *software TCPReplay*, conforme teste realizado na Subseção 4.2.1.

Neste teste, o invasor simula ser o IED *publisher* da comunicação que informa as transições realizadas na lógica de geração de pulsos, porém, existe um IED *publisher* com as mesmas configurações, mas em uma cadência de envio diferente, que informa situação normal de operação.

Para realizar outro teste de validação da aplicação, desenvolveu-se um programa em linguagem C para capturar as mensagens, ao fim da lógica de geração de pulsos, alterar o DATA-SET para verdadeiro, incrementar o *stNum*, atribuir o valor 0 (zero) ao *sqNum* e replicar a mensagem na rede. Neste caso, o invasor simula ser um IED *publisher* que

⁷RTDS é um simulador digital de SE em tempo real, com passo de integração de 50μ (MAGUIRE; GIESBRECHT, 2005).

informa um evento, porém, existe outro IED *publisher* com as mesmas configurações, que informa situação normal de operação. Os três casos de teste foram repetidos, configurando o RTDS como *publisher* e os IEDs como *subscribers*. A Tabela 4 apresenta os testes e a resposta do controlador para os cenários configurados.

Tabela 4 – Testes e Classificação do Controlador *OpenFlow* para os IEDs.

Teste realizado	Resposta do controlador
Lógica de geração de pulsos entre os IEDs	Mensagens legítimas
Replicação das mensagens dos IEDs salvas pelo atacante	Mensagens falsas
Captura das mensagens geradas pelo IED, alteração dos campos stNum, sqNum, booleano e replicação das mensagens na rede	Mensagens falsas
Lógica de geração de pulsos no RTDS	Mensagens legítimas
Replicação das mensagens do RTDS salvas pelo atacante	Mensagens falsas
Captura das mensagens geradas pelo RTDS, alteração dos campos stNum, sqNum, booleano e replicação das mensagens na rede	Mensagens falsas

Em todos os cenários de teste, acompanhou-se o *led* na IHM dos IEDs e o arquivo de *log* do SO, na plataforma ZYBO. Durante a execução da lógica de geração de pulsos pelo IED, ou pelo RTDS, o *led* dos IEDs *subscribers* piscaram, conforme esperado. Nos ataques de replicação de mensagens e alteração dos campos, os *leds* não piscaram. Observando o arquivo de *log* do SO na plataforma ZYBO, pôde-se comprovar que estas mensagens foram bloqueadas no controlador.

4.6.6.2 Rede de Comunicação Emulada para uma Grande Subestação

De acordo com (IEC61850, 2003a), uma grande subestação de transmissão possui mais que 10 elementos, e uma grande subestação de distribuição possui 20 elementos ou mais. Considerando cada elemento como sendo um IED *subscriber*, um novo teste para validação da operação do controlador, em grandes subestações, foi realizado. Para simular o tráfego de mensagens GOOSE em uma grande subestação, foi instalado, em um computador com duas placas de rede, o emulador de redes Mininet⁸ e o *software switch* OpenvSwitch⁹. Executando-se o *script* do emulador, foi configurada uma rede com 30 computadores conectados a 3 *switches* (10 em cada *switch*) (Figura 29).

A plataforma ZYBO com o controlador em execução foi interligada ao computador com a Mininet, através da placa de rede. O computador responsável pela realização dos

⁸Mininet, é um sistema para protótipo rápido de grandes redes. Ela possibilita a implementação de SDNs sobre os recursos de um único computador (LANTZ; HELLER; MCKEOWN, 2010).

⁹OpenvSwitch é um *switch* virtual multicamadas projetado para permitir a automatização de rede, através de uma extensão programável, apoiando interfaces de gerenciamento padrão e protocolos (OPENVSWITCH, 2014).

Figura 29 – Arranjo laboratorial emulado para realização dos *pentests*.

ataques também foi conectado ao que hospeda a rede emulada, através da outra interface de rede.

Para possibilitar a comunicação entre os computadores e a ZYBO, foi utilizado o *OpenvSwitch*, *software* que permite configurar os *switchs* emulados, o que possibilitou a inclusão das interfaces de rede do computador ao primeiro *switch* emulado. Realizadas as interligações físicas e lógicas entre os dispositivos, os *switchs* *OpenFlow* emulados se conectaram ao controlador de rede *OpenMul*, ação que possibilitou analisar a segurança das mensagens GOOSE através da aplicação desenvolvida. Cada computador foi utilizado como um IED *publisher*. Usando a biblioteca *libiec61850* (ZILLGITH, 2016), foram geradas mensagens GOOSE em cada IED, seguindo a lógica de geração de pulsos descrita na Seção 4.2, durante 30 minutos. Transcorrido este tempo, os IEDs passam a enviar a mensagem GOOSE com o valor *booleano* do DATA-SET com valor falso, indefinidamente.

Utilizando o computador interligado à rede emulada, as mensagens do teste com a lógica de geração de pulsos foram capturadas. Após finalizar a sequência de eventos do referido teste, a captura foi interrompida e as mensagens capturadas foram salvas no computador e reenviadas na rede, utilizando o *software* *TCPReplay* conforme se procedeu na Subseção 4.2.1.

Para realizar o terceiro *pentest* foi utilizado o programa em linguagem C desenvolvido para capturar as mensagens ao fim da lógica de geração de pulsos, alterar o DATA-SET para verdadeiro, incrementar o *stNum*, atribuir o valor 0 (zero) ao *sqNum* e replicar a mensagem na rede, conforme descrito na Subseção 4.2.2. A Tabela 5 apresenta os testes e a resposta do controlador para os cenários configurados.

Cabe ressaltar que a implementação descrita não irá resolver, totalmente, os problemas de falsificação de mensagens GOOSE na rede de comunicação. Um intruso com conhecimento do protocolo, analisando os valores das mensagens GOOSE temporalmente na rede, pode gerar uma mensagem com valor correto para os campos. É importante destacar que, ao detectar uma mensagem falsa, o alerta inserido no *log* seja investigado, pois em algum momento, o intruso pode criar uma mensagem que esteja com os valores dos campos corretos para a execução da informação contida na mensagem adulterada.

Tabela 5 – Testes e Classificação do Controlador *OpenFlow* para a Rede Emulada.

Teste realizado	Resposta do controlador
Lógica de geração de pulsos na Mininet	Mensagens legítimas
Replicação das mensagens salvas pelo atacante na Mininet	Mensagens falsas
Captura das mensagens geradas na Mininet, alteração dos campos <i>stNum</i> , <i>sqNum</i> , booleano e replicação das mensagens na Mininet	Mensagens falsas

Com a lógica implementada, é importante investigar, seu tempo de execução. Tendo em vista que a mensagem GOOSE possui um tempo de tráfego de até 4 milissegundos, o sistema implementado deve realizar a análise da mensagem em um tempo menor que este.

4.7 Ferramenta de Medição

Para avaliar o tempo de processamento da detecção e do bloqueio de mensagens falsas na rede de comunicação das subestações, é importante ter uma ferramenta robusta e confiável que não inclua atrasos na medição, por interferências externas no teste realizado. De início, foi cogitada a possibilidade do *publisher* ser um relé digital e utilizar o *log* de eventos do relé *subscriber* para avaliar o tempo de tráfego das mensagens. Esta alternativa não foi adotada, porque, o processamento feito pelo relé, para estampar o tempo no *log* de eventos é desconhecido e não especificado pelo equipamento, dúvida indesejada nesta investigação.

Outra alternativa cogitada foi colocar um microcomputador que processasse uma sequência de pacotes GOOSE salvas em um arquivo PCAP, utilizando o *software* TCPReplay (TURNER, 2001), que permite transmitir um tráfego de rede salvo em um arquivo PCAP. Por se executar em nível de *software*, este programa deixa evidentes, na realização de algumas sequências de testes, uma variação de tempo, devida à carga da *Central Processing Unit* (CPU), operações de *Input/Output* (I/O), e variação nas interrupções de serviços do SO. O *jitter*¹⁰, no acesso à placa de rede, iria interferir nos experimentos para a medição do tempo de processamento da aplicação implementada. A alternativa encontrada foi a utilização de uma *Network Field-Programmable Gate Array* (NetFPGA) que possui uma aplicação chamada *packet generator* desenvolvida por Covington et al. (2009) e voltada para a realização de medições de rede. Esta implementação é disponibilizada juntamente com o conjunto de *softwares* que compõem a NetFPGA, plataforma de *hardware* reconfigurável otimizada para redes de alta velocidade e composta por recursos lógicos, memória e interfaces *Gigabit Ethernet* (GbE) necessárias à completa construção

¹⁰É a medida de variação do atraso entre os pacotes sucessivos de dados (FOROUZAN, 2008).

de pontes, roteadores e dispositivos de segurança. Seu caminho de dados é implementado em *hardware*, no qual a latência do processamento é medida em poucos ciclos de *clock*. O projeto NetFPGA surgiu como ferramenta didática para o ensino de *hardware* de rede e projeto de roteadores, tornando-se útil, também, para pesquisadores interessados em redes (STANFORD, 2006).

As redes de comunicação de dados evoluíram rapidamente, como decorrência da implementação de dispositivos de comutação e roteamento utilizando aceleradores de *hardware*. O estudo de comunicação de dados deve perpassar o processamento de pacotes em *hardware*, caso o pesquisador tenha interesse em construir sistemas de tempo real.

Grande parte deste estudo é realizada, utilizando simuladores e emuladores que são dependentes de atrasos do SO. O projeto NetFPGA surgiu da necessidade de treinamento de estudantes no projeto e desenvolvimento de *hardware* de rede capaz de processar um grande volume de pacotes de *links GbE* carregados (GIBB et al., 2008).

4.7.1 Composição da *NetFPGA*

A Figura 30 exibe a NetFPGA em que é possível observar no centro da plataforma uma FPGA *Xilinx Virtex2-Pro 50* que é utilizada para hospedar a lógica do usuário, sendo a FPGA principal. Este *chip* possui um *clock* de 125 MHz e 53.163 *LUTs*. A placa contém ainda um pequeno *chip* FPGA *Xilinx Spartan II* que implementa o controle lógico da interface *Peripheral Component Interconnect* (PCI) que conecta a placa à um computador.

Figura 30 – NetFPGA

Fonte: Stanford (2006)

Em torno da FPGA principal, estão 4 dispositivos de memória. Duas *Static Random Access Memory* (SRAM) que operam em sincronismo com a FPGA, e duas *DDR2 Synchronous Dynamic Random Access Memory* (SDRAM) operando de forma assíncrona. A capacidade da memória SRAM da placa é de 4.5 MB. O banco de memórias DDR2

SDRAM tem capacidade total de 64 MB. Usando extremos 200 MHz do *clock*, a memória fica com uma largura de banda de 12.800 Mbps (Figura 31).

Figura 31 – Hardware NetFPGA

Fonte: Gibb et al. (2008)

A NetFPGA é composta por 4 interfaces *Broadcom quad-port gigabit PHY* de 1 Gbps Ethernet que se comunica com o meio utilizando cabeamento par trançado categoria 5, 5e ou 6¹¹. A NetFPGA possui, ainda, duas interfaces *Serial Attachment (SATA)* que permite a interligação de várias placas NetFPGA, em um sistema de troca de tráfego, sem o uso do barramento PCI. Além do *hardware*, ferramentas de *design* são utilizadas para simular e implementar projetos na NetFPGA. Este conjunto de ferramentas permite que os projetos sejam simulados, antes de compilar, para a verificação da operação lógica correta. O código fonte para o desenvolvimento de aplicações é escrito em *Verilog*, e, posteriormente, sintetizado onde a lógica é mapeada, organizada e encaminhada por *scripts* que utilizam ferramentas da *Xilinx*, em segundo plano. Finalmente, o *bitfile* resultante é enviado à FPGA, usando um programa de linha de comando executado no computador. A depuração dos circuitos de *hardware* pode ser realizada, utilizando *softwares* analisadores lógicos, através da porta *Joint Test Action Group (JTAG)* (GIBB et al., 2008).

No site do fabricante, é disponibilizada uma instalação do SO Linux distribuição Fedora, que possui em seu *kernel* o *driver* para comunicação com a placa. O *driver* possibilita ler e escrever em registros mapeados de memória SRAM e *Dynamic Random*

¹¹Cabeamento par trançado categoria 5, 5e ou 6 que consiste em quatro pares de fios isolados e levemente trançados, agrupados em uma capa plástica, para proteger os fios e mantê-los juntos (TANENBAUM, 2011).

Access Memory (DRAM) da placa. No total, são 128 MB de memória do computador reservado para a placa NetFPGA. Os programas conseguem enviar e receber pacotes, utilizando a NetFPGA, através de interfaces de *socket* padrão. Utilizando *Direct Memory Access (DMA)*, é possível transferir quadros *Ethernet* completos entre o computador e a NetFPGA. Esta via de acesso é importante, porquanto permite trafegar dados entre o computador e a NetFPGA, de forma simultânea, sem interromper os processos de entrada e saída que a placa estiver executando.

4.7.2 Funcionamento da *NetFPGA*

A implementação do *hardware* é dividida em módulos. Os projetos de referência disponibilizados juntamente com a instalação do *software*, inclusive a placa de rede da NetFPGA, seguem esta sequência de módulos. Sua estrutura é um *pipeline*, em que cada fase é um módulo separado (Figura 32).

Figura 32 – *Pipeline* NetFPGA

Fonte: Gibb et al. (2008)

A primeira fase do *pipeline* é composta por filas de recepção Rx que recebem pacotes das portas *Ethernet* e PCI sobre *DMA*, proporcionando uma interface unificada de recepção dos pacotes para os demais módulos. Estas filas são conectadas em um invólucro chamado *User Data Path* que contém as etapas de processamento. No *User Data*

Path, o pacote atravessa o primeiro módulo *Input Arbiter* que decide qual fila de entrada será encaminhada para o módulo seguinte, isto é, o *Output Port Lookup* responsável por decidir a porta de saída do pacote. Após realizar esta decisão, o pacote é entregue ao módulo *Output Queues* que armazena o pacote, na fila de saída correspondente à porta de saída, até que a fila de transmissão Tx esteja pronta para transmiti-lo. As filas Tx são análogas às filas Rx, porém, estas são utilizadas para envio dos pacotes. As filas Tx enviam os pacotes não somente para as portas *Ethernet*, mas, também, para as filas *PCI DMA*. Para cada um destes módulos existe um conjunto de registros com informações sobre o estado e os sinais de controle de acesso ao módulo (STANFORD, 2006).

4.7.3 Aplicação *Packet Generator*

A aplicação *packet generator* envia os pacotes de um arquivo PCAP para a memória SRAM da NetFPGA. O *software* presente no computador carrega os parâmetros da aplicação no *hardware*. Posteriormente, os dados carregados são enviados por uma ou mais interfaces de rede da NetFPGA onde todo o tráfego é controlado pelo *hardware*. O *packet generator* não é afetado por preempções do *kernel* ou latência dos acessos ao barramento PCI. Além de gerar tráfego, a aplicação possibilita salvar os dados recebidos em arquivo PCAP onde as estampas de tempo são feitas em *hardware*, em qualquer um dos quatro *Network Interface Cards* (NICs) da NetFPGA, individualmente. A Figura 33 ilustra o *pipeline* da aplicação *packet generator* em NetFPGA.

Figura 33 – Diagrama de blocos dos componentes da aplicação *packet generator*.

Fonte: Covington et al. (2009).

Observa-se, na Figura 33, que pequenas modificações no *pipeline* padrão da NetFPGA foram inseridas e estão em destaque. O primeiro destaque é o *Time Stamp* antes das filas das interfaces. Ele permite que o pacote tenha o tempo estampado, assim que for

recebido pelo *hardware*. Esta estampa de tempo é utilizada, posteriormente, na criação do arquivo PCAP resultante da execução dos testes de rede.

Outro bloco em destaque é o *Packet Capture* que desempenha a função de compilação de estatísticas agregadas da geração e captura de pacotes, como o número de pacotes recebidos e o tempo total de captura. Caso o *software* não esteja em execução, o bloco *Packet Capture* tem a função de desabilitar a estampa de tempo onde a NetFPGA passa a responder como uma NIC de quatro portas. As 4 filas incluídas no bloco *Output Queues* são utilizadas para armazenar os dados do arquivo PCAP para transmissão, quando o *packet generator* está em execução. O bloco *Packet Generator Output Select (Pkt Gen Output Sel)* é utilizado para conectar as 12 filas do bloco *Output Queues* nas 8 filas das 4 interfaces de rede do *pipeline* principal. Cada fila da interface de rede possui dois blocos, *Rate Limiter* e *Delay*, conectados a ela. Estes blocos permitem a inserção de atrasos e limites de taxa de transmissão para cada uma das filas de saída, individualmente. A utilização desta ferramenta e a característica desta aplicação de estampar o tempo no *hardware* da placa de rede permitiram a execução de testes com estampas de tempo precisas.

4.7.4 Criação do Arquivo de Testes

O arquivo PCAP utilizado para a realização dos testes foi criado, a partir da captura de mensagens GOOSE geradas no RTDS. O RTDS é um simulador de SE digital capaz de funcionar em tempo real. É um equipamento modular composto por placas de processamento e de funcionalidades específicas como a GTNet. Com a placa GTNet e *firmware* específico para a devida funcionalidade, é possível simular o envio e recebimento de mensagens GOOSE, através de uma rede de comunicação de dados. A Figura 34 ilustra o envio de mensagens GOOSE realizado no RTDS, para compor um arquivo PCAP utilizado nos testes mencionados neste capítulo.

A simulação realizada no RTDS exemplifica um relé ligado a um SE em condições normais de operação. O arquivo contém 1.000 mensagens GOOSE que foram enviadas via placa GTNet do RTDS. As mensagens geradas no RTDS possuem um tamanho de 300 *bytes* cada, compostas por 36 itens no DATA-SET.

4.8 Avaliação do Tempo de Processamento da Detecção e Bloqueio de Ataques *Man-in-the-middle* Utilizando o Protocolo *OpenFlow*

Para analisar o tempo de processamento da ferramenta de detecção de intrusão desenvolvida, foram utilizadas uma NetFPGA, que tem como função executar a aplicação

Figura 34 – Captura de mensagens GOOSE enviadas pelo RTDS.

packet generator para injetar e medir o tráfego. O *switch OpenFlow* recebe as mensagens, encaminha-as para o controlador, recebe o retorno do controlador e encaminha a mensagem para a NetFPGA, em uma interface diferente.

4.8.1 Ambiente Configurado

O ambiente configurado é composto por uma NetFPGA que possui duas das suas portas interligadas ao *switch*, utilizando cabos de rede *Unshielded Twisted Pair* (UTP), categoria 5e *path cord* e outro *path cord* interligando a outras duas interfaces da NetFPGA, de tamanho igual à soma do cabo de ida e volta ao *switch*. O *switch* possui uma porta configurada para gerência ligada à plataforma ZYBO, também *patch cord*, onde o controlador *OpenMul* é executado, sob um SO de tempo real (Figura 35).

As mensagens contidas no arquivo PCAP são carregadas na memória SRAM da NetFPGA, utilizando a aplicação *packet generator*, posteriormente enviadas, simultaneamente, em duas interfaces, a que está interligada ao *switch OpenFlow* e a que está interligando as outras duas interfaces da NetFPGA. A mensagem que chegar ao *switch OpenFlow* será encaminhada em um pacote Packet-in para o controlador. O controlador realiza as verificações de segurança e bloqueia ou autoriza a mensagem. Quando autorizada, a mensagem retorna ao *switch* que a encaminha para a NetFPGA. Quando a mensagem retornar à NetFPGA, o *packet generator* irá estampar o tempo de chegada de ambas as mensagens e armazená-las em dois arquivos PCAP, um para cada interface de recepção. Desta forma, pela diferença de tempo entre o pacote que passou pelo processo de verificação da segurança e o pacote que passou apenas pelo cabo, foi desconsiderado o tempo do cabeamento entre a NetFPGA e o *switch*, analisando somente, o tempo gasto entre a chegada do pacote no *switch*, o envio e processamento no controlador, o retorno da mensagem, até sua saída no *switch*.

Figura 35 – Comunicação NetFPGA computador organizada para realização dos testes.

4.8.2 Medição do Tempo de Tráfego da Aplicação de Segurança Utilizando as Configurações Recomendadas pelo Fabricante do *Switch OpenFlow*

Para a realização do teste, configurou-se o *switch OpenFlow*, seguindo as recomendações do fabricante e analisando o referencial bibliográfico, para que se obtivesse menor atraso e melhor performance do *switch*. Seus parâmetros *Openflow* para este teste foram (Tabela 6):

Tabela 6 – Parâmetros *OpenFlow* para o teste.

Configuração	Valor
Modo	<i>proactive</i>
<i>Rate-limit</i>	10.000.000 kbit/s
Prioridade dos filtros <i>OpenFlow</i>	4
<i>Strip-fcs</i>	disabled
<i>Max Pending Requests</i>	1024
<i>Minimum Idle Timeout</i>	10
<i>Idle Timeout Mode</i>	reject
<i>CPU Protocol Priority</i>	enable
<i>CPU Protocol Priority OpenFlow</i>	7

Justificando os dados da Tabela 6 tem-se que:

- ❑ o modo proativo foi escolhido, pois, o fluxo estando previamente cadastrado no *switch*, diminui a latência na escolha da ação do protocolo. Neste caso, a aplicação *l2switch* foi modificada para que, no ato da conexão com os *switches*, seja cadastrado um fluxo informando que mensagens do tipo 0x88b8 devem ser encaminhadas para o controlador;
- ❑ o parâmetro *rate-limit* foi configurado para o maior valor que o mesmo pode receber. Como este parâmetro limita a taxa das mensagens Packet-in, optou-se por deixá-lo com este valor alto, para que ele não limite a troca de mensagens entre *switch* e controlador. Caso na realização dos testes se descubra algum gargalo, deseja-se que este não seja causado por um parâmetro do *switch*, mas, sim, pelos recursos de *hardware* ou de rede utilizados;
- ❑ a prioridade dos filtros *OpenFlow* foi configurada para o seu valor máximo, ou seja, o *switch* irá dar preferência para o processamento das filas do *OpenFlow*;
- ❑ o parâmetro *strip-fcs* foi desabilitado, com o intuito de remover mais este processamento do *switch*. Como o *Frame Check Sequence* não influencia o funcionamento da aplicação, sua presença, ou ausência, no *Packet-in* não será notada;
- ❑ o *Max Pending Requests* foi configurado para o seu valor máximo, por recomendação do fabricante. Este parâmetro terá grande importância em uma rede que possui constantes modificações na tabela de fluxos, o que não é o caso da presente aplicação;
- ❑ o parâmetro *Minimum Idle Timeout* foi configurado para 10 segundos, por ser o valor padrão mínimo para este parâmetro;
- ❑ o *Idle Timeout Mode* foi configurado para *reject* com o intuito de que não onerar o *switch* com mais este processamento;
- ❑ o parâmetro *CPU Protocol Priority* foi habilitado para direcionar os recursos do *switch* para o *OpenFlow* definindo alta prioridade; e
- ❑ por fim, o *CPU Protocol Priority OpenFlow* foi configurado para o seu maior valor.

Realizadas as configurações do *OpenFlow* no *switch*, deu-se início aos testes. O primeiro teste consistiu no envio das 1.000 mensagens salvas no arquivo PCAP, utilizando o ambiente configurado. Este teste foi repetido 10 vezes, com o intuito de verificar as variações e obter um número maior de casos. Ao todo, são 10.000 mensagens enviadas a cada 2 segundos, o que exemplifica 5 horas, 33 minutos e 20 segundos de tráfego de mensagens GOOSE sendo medido.

Os arquivos PCAP resultantes das duas comunicações realizadas, passando pelo controlador e passando somente pelo cabeamento, são salvos e, utilizando o *software wireshark*, as informações dos pacotes são exportadas em formato *Comma-Separated Va-*

lues (CSV). Com as informações em uma planilha eletrônica, o tempo dos pacotes são subtraídos e o maior valor, o menor, a média, a moda e o desvio padrão são calculados. Após o cálculo das 10 repetições, os dados são agrupados em uma única planilha onde foi construído um histograma. Para a construção do histograma, é necessária uma classificação dos dados por distribuição de frequências. Para a realização desta seriação, deve-se dividir a amplitude dos dados em intervalos de classe de tamanho igual, de forma a possibilitar uma informação visual dos valores que mais se repetem, bem como, a dispersão dos tempos e o tamanho das classes, para que se classifique o tempo gasto pela implementação de segurança realizada via *switch OpenFlow*. Hines et al. (2006) sugerem que a escolha do número de intervalos de uma classe deva ser menor ou igual à raiz quadrada do número de observações.

Atendendo a esta especificação e analisando o conforto visual do gráfico apresentado, foram definidas 50 classes de mesmo tamanho. Os valores foram categorizados e contados de forma a quantizar o número de valores que pertencem a cada uma das classes. Além do histograma, foi apresentada no gráfico a porcentagem da frequência acumulada dos dados (Figura 36).

Figura 36 – Histograma do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando *OpenFlow*.

Os demais valores calculados no presente teste estão listados na Tabela 7.

Observa-se no gráfico da Figura 36 e nos valores apresentados na Tabela 7, que o desvio padrão está muito alto. A faixa de tempo mais frequente está entre 2,26 e 0,45 segundos, e que o tempo médio gasto para executar as lógicas implementadas via *OpenFlow* está

Tabela 7 – Resultado estatístico do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando OpenFlow.

Cálculo	Valor
Maior	3,602 s
Menor	0,005 s
Média	1,357 s
Moda	0,026 s
Desvio Padrão	0,905 s
Média + Desvio Padrão	2,262 s
Média - Desvio Padrão	0,451 s

em 1,35 segundos. Diante da grande variação no tempo, gerou-se, também, o gráfico da quantidade de mensagens em função do tempo (Figura 37).

Figura 37 – Tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando OpenFlow.

Observa-se, neste gráfico, que existem linhas de tendência que demonstram atrasos ocorridos durante o processamento. Nota-se que estes padrões horizontais que se apresentam de forma paralela são característicos de latências de equipamentos que realizam determinados processamentos em janelas de execução variadas no tempo. Outro fator que corrobora com os comportamentos observados são as filas dos equipamentos. Neste contexto, cabe investigar em qual equipamento está o atraso e o que os está provocando. Com as investigações realizadas até o momento, conclui-se que para a situação medida, a implementação deste sistema, em uma rede da subestação, é totalmente inviável. A faixa do tempo de resposta está muito acima dos 4 milissegundos recomendados para o tráfego das mensagens GOOSE.

4.8.2.1 Teste de *Throughput*

Para analisar a capacidade de processamento da plataforma juntamente com o *software* desenvolvido, foi gerada, utilizando a biblioteca libiec61850, a maior mensagem GOOSE que a placa de rede da plataforma ZYBO é capaz de receber, 1500 *bytes*, limitada pela *Maximum Transmission Unit* (MTU) da mesma. Utilizando a mesma estrutura apresentada na Subseção 4.8.1, a mensagem foi carregada na NetFPGA, utilizando o *packet generator*, e enviada, variando o fluxo de pacotes. O *packet generator* executado na NetFPGA possui configurações para a definição da taxa de transmissão, atraso entre os pacotes e o número de iterações para o envio do pacote. Com o intuito de avaliar a maior quantidade de pacotes que o controlador *OpenMul*, executando as análises de segurança, é capaz de processar, o *packet generator* foi configurado para enviar a mensagem a uma taxa de 100 Mbps, atualmente, a maior velocidade utilizada nas redes das subestações, sem atrasos entre os pacotes, para diferentes quantidades de mensagens.

O *packet generator* envia uma rajada de mensagens para a plataforma e registra as que retornam, calculando a velocidade de transmissão, a quantidade de *bytes* trafegados e o tempo de transmissão. O aplicativo foi avaliado para diferentes quantidades de mensagens. Como critério de parada, adotou-se que o dispositivo deveria responder a todas as mensagens enviadas para até 10 testes seguidos. Caso ele conseguisse, a quantidade de mensagens seria aumentada, caso não, a quantidade seria diminuída. O melhor caso, sem perder pacotes e para a maior quantidade de mensagens foi:

- ❑ 127 mensagens;
- ❑ 191.008 *Bytes* trafegados;
- ❑ variação de 2,214 a 2,568 *Mbps* de taxa de transmissão;
- ❑ variação de 0,602 a 0,699 segundos;
- ❑ variação de 4,740 a 5,503 ms no tempo médio de processamento de cada mensagem.

Com isso, constata-se que, em situação extrema de tráfego, o *switch* apresentou um comportamento melhor, em relação à latência de cada mensagem, do que na cadência normal das mensagens GOOSE. Neste contexto, acredita-se que a latência do teste anterior pode diminuir.

4.8.3 Medição do Tempo de Tráfego da Aplicação de Segurança com Ajustes nas Configurações do *Switch OpenFlow*

Realizadas algumas medições, observou-se que é possível diminuir o tempo de tráfego das mensagens GOOSE, diminuindo o parâmetro *min-idle-timeout* do *switch*. Uma informação importante sobre este parâmetro é que, além de ajustar o tempo mínimo de

expiração dos fluxos recebidos, ele determina o tempo de atualização dos contadores, deixando-os visíveis para consulta do controlador a cada *min-idle-timeout* dividido por 2, segundos. O valor mínimo padrão para este parâmetro é de 10 segundos. Ele foi ajustado para 1 segundo, o valor mínimo que ele é capaz de assumir, porém, o fabricante alerta que cenários com escalabilidade de fluxos, valores abaixo de 10 segundos não são recomendados, devido à elevada carga de processamento (DATACOM, 2014). As demais configurações do *switch* foram mantidas, conforme a Tabela 6. O conjunto dos 10 testes com as mesmas 1.000 mensagens GOOSE foram repetidos. O histograma com a frequência dos valores medidos e a porcentagem acumulada podem ser observados na Figura 38.

Figura 38 – Histograma do tempo de tráfego para verificar a segurança das mensagens GOOSE com ajustes no *switch OpenFlow*.

Os demais valores calculados no presente teste estão listados na Tabela 8.

Tabela 8 – Resultado estatístico do tempo de tráfego para verificar a segurança das mensagens GOOSE com ajustes no *switch OpenFlow*.

Cálculo	Valor
Maior	0,451 s
Menor	0,005 s
Média	0,203 s
Moda	0,026 s
Desvio Padrão	0,122 s
Média + Desvio Padrão	0,326 s
Média - Desvio Padrão	0,080 s

Analisando os resultados deste teste, pode-se observar que a variação do tempo diminui

com relação ao primeiro teste, porém, ela ainda continua alta. Para verificar se os resultados estão mais homogêneos, gerou-se também o gráfico da quantidade de mensagens em função do tempo (Figura 39).

Figura 39 – Tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando configurações ajustadas do *OpenFlow*.

Observa-se, neste gráfico, uma pequena mudança no comportamento dos atrasos. As linhas horizontais paralelas observadas no gráfico anterior já não são claras neste. Existe um padrão de linhas transversais que pôde ser notado no gráfico anterior, que continua neste gráfico. Durante o acompanhamento da classificação das mensagens no controlador, constatou-se uma descontinuidade na cadência de chegada das mensagens. Observou-se que um grupo de mensagens eram classificadas em curtos espaços de tempo e a cadência entre elas vão aumentando, até que um novo grupo de mensagens seja entregue, novamente. Apesar de o comportamento observado indicar que o atraso está no *switch*, é importante investigar o tempo de resposta nos dois *hardwares*, afinal, ambos podem se tornar um gargalo nesta comunicação. Os resultados apresentados com este teste, demonstram que, mesmo ajustando as configurações do *switch* a implementação deste sistema em uma rede da subestação ainda é inviável. Os valores medidos ultrapassam os 4 milissegundos recomendados para o tráfego das mensagens GOOSE.

4.8.3.1 Teste de *Throughput*

Para analisar o *throughput* nesta nova configuração do *switch*, repetiu-se o processo realizado anteriormente. A mensagem de 1500 *bytes* foi enviada para passar pelo *switch*,

controlador e *switch*, novamente, variando o fluxo de pacotes. As mensagens foram enviadas a uma taxa de 100 Mbps, sem atrasos entre os pacotes, para diferentes quantidades de mensagens. O melhor caso, sem perder pacotes e para a maior quantidade de mensagens foi:

- ❑ 134 mensagens;
- ❑ 201.000 *Bytes* trafegados;
- ❑ variação de 0,915 a 2,559 *Mbps* de taxa de transmissão;
- ❑ variação de 0,636 a 1,779 segundos; e
- ❑ variação de 4,746 a 13,276 ms no tempo médio de processamento de cada mensagem.

Observa-se um aumento na quantidade de mensagens, porém, a variação na taxa e no tempo de transmissão ficaram maiores. Mesmo com o ajuste realizado, os resultados de ambos os testes não chegaram a patamares aceitáveis para o tráfego de mensagens GOOSE. Existe um atraso ocorrendo em algum equipamento, neste processo de comunicação, que precisa ser investigado.

4.8.4 Medição do Tempo de Tráfego para Mensagens GOOSE Autorizadas na Tabela de Fluxos do *Switch OpenFlow*

Para medir o tempo de tráfego, sem que a mensagem passe pelo controlador, o código fonte do controlador foi modificado para registrar, na tabela do *switch*, um fluxo indicando que a mensagem GOOSE que chegasse na porta que o *packet generator* está enviando as mensagens, seja encaminhada para a porta em que a NetFPGA está recebendo e registrando o tempo das mensagens. Ao estabelecer a conexão entre *switch* e controlador, o fluxo é registrado, proativamente, no *switch*. Com isso, foi possível observar o tempo de encaminhamento de mensagens do *switch Openflow*. O mesmo conjunto de medições realizado nos testes anteriores foi aplicado nesta nova estrutura e o histograma do tráfego foi gerado (Figura 40)

Os demais valores calculados no presente teste estão listados na Tabela 9.

Observa-se, pelos resultados do histograma e da tabela, que o *switch* possui latência adequada para envio das mensagens, caso essas estejam autorizadas na sua tabela de fluxos. O teste do *throughput* no *switch* também foi realizado, porém, observou-se que o parâmetro de 100 Mbps deixa 10% da banda do *switch* livres, portanto, não há a necessidade de medição do referido teste, sendo que, nesta condição de funcionamento, o *switch* não será o gargalo da comunicação, mas, sim, os demais equipamentos. Analisada esta condição, cabe ainda investigar se o atraso está no controlador, juntamente com o *software* desenvolvido, ou no processo de encaminhamento da mensagem para o controlador.

Figura 40 – Histograma do tempo de tráfego para mensagens GOOSE autorizadas na tabela de fluxos do *switch OpenFlow*.

Tabela 9 – Resultado estatístico do tempo de tráfego para mensagens GOOSE autorizadas na tabela de fluxos do *switch OpenFlow*.

Cálculo	Valor
Maior	6,810 μs
Menor	6,280 μs
Média	6,553 μs
Moda	6,384 μs
Desvio Padrão	0,136 μs
Média + Desvio Padrão	6,690 μs
Média - Desvio Padrão	6,417 μs

4.8.5 Medição do Tempo de Execução da Aplicação de Segurança sem a Interferência do *Switch Openflow*

Para verificar se o atraso está na aplicação que verifica a segurança, foi desenvolvido um *software* para receber a mensagem, verificar se é uma mensagem GOOSE, realizar a chamada das funções que classificam se a mensagem é verdadeira ou falsa, exatamente como é realizado no controlador *OpenFlow*, e encaminhá-la de volta pela interface de rede da plataforma ZYBO. O ambiente configurado é composto pela NetFPGA, interligando uma das suas portas à plataforma ZYBO, utilizando cabo de rede UTP, categoria 5e *path cord* e outro *path cord* interligando outras duas interfaces da NetFPGA de tamanho igual à soma do cabo de ida e volta à placa. A plataforma ZYBO, executa o SO Linux Preempt_rt, onde o *software* foi compilado com as bibliotecas *-lrt* e *-pthread*, de forma que

o mesmo execute em tempo real (Figura 41).

Figura 41 – Ilustração da comunicação NetFPGA plataforma ZYBO organizada para realização dos testes.

O conjunto dos 10 testes com as mesmas 1.000 mensagens GOOSE foram repetidos. Porém, desta vez, a aplicação *Packet Generator* executada na NetFPGA para medir o tráfego, irá enviar para o *software* e receber do *software* a mensagem processada no sistema de segurança. O histograma com a frequência dos valores medidos neste teste, podem ser observados na Figura 42.

Os demais valores calculados neste teste estão listados na Tabela 10.

Tabela 10 – Resultado estatístico do tempo de execução da verificação de segurança das mensagens GOOSE sem a interferência do *switch*.

Cálculo	Valor
Maior	567,45 μs
Menor	511 μs
Média	539,65 μs
Moda	524 μs
Desvio Padrão	0,169 μs
Média + Desvio Padrão	556,55 μs
Média - Desvio Padrão	522,74 μs

O resultado das medições do tempo de execução do *software* de verificação da segurança das mensagens GOOSE apresentou resultados interessantes, quanto ao tempo de execução da verificação de segurança. O tempo de processamento ficou em torno dos 540 μs o que atende à especificação de latência das mensagens GOOSE.

Figura 42 – Histograma do tempo de execução da verificação de segurança das mensagens GOOSE sem a Interferência do *Switch*.

4.8.5.1 Teste de *Throughput*

Para analisar o *throughput* do *software* desenvolvido, repetiu-se o processo realizado anteriormente para este novo ambiente configurado. A mensagem de 1500 *bytes* foi enviada para passar pelo *software*, analisar a segurança e retornar para a NetFPGA. As mensagens foram enviadas a uma taxa de 100 Mbps, sem atrasos entre os pacotes, para diferentes quantidades de mensagens. O melhor caso, sem perder pacotes e para a maior quantidade de mensagens foi:

- ❑ 267 mensagens;
- ❑ 400.500 *Bytes* trafegados;
- ❑ variação de 4,269 a 4,274 *Mbps* de taxa de transmissão;
- ❑ variação de 0,759 a 0,760 segundos; e
- ❑ variação de 2,842 a 2,846 ms no tempo médio de processamento de cada mensagem.

Observa-se, neste caso, que, mesmo submetido a alto tráfego, o *software* é capaz de processar uma grande quantidade de mensagens dentro do tempo de 4 milissegundos definidos pela IEC-61850. O encaminhamento da mensagem realizado no *switch*, sem passar pelo controlador e o processamento da aplicação de verificação da segurança, possuem

latências pertinentes necessárias para trafegar mensagens GOOSE. Segundo Saikia e Malik (2014), o tempo de resposta do controlador *OpenMul* é de 50 μs por fluxo. Quando estas três situações se juntam, o *switch* encaminha a mensagem para o controlador, o controlador executa as verificações de segurança e devolve a mensagem para o *switch*; o atraso resultante desta medição tem apresentado latências muito acima da soma desses três processos, se executados em separado. Neste íterim, propôs-se investigar o tempo de resposta desta mesma aplicação, controlando um *Open Virtual Switch* (OVS).

4.8.6 Medição do Tempo de Tráfego da Aplicação de Segurança Controlando um *Open Virtual Switch*

Segundo Pan e Wang (2015), *OpenvSwitch* é a implementação em *software* mais popular de um *switch OpenFlow* no SO Linux. Sua implementação inclui um módulo de *kernel* e um *daemon* em espaço de usuário, para ele se conectar com o controlador *OpenFlow*. O *OpenvSwitch* é comumente utilizado em servidores físicos, para conectar todas as máquinas virtuais em execução no servidor com a rede local. Além disso, ele pode ser executado em um SO Linux, para que as placas de rede funcionem como um *hardware switch* (CHIU; WANG, 2015). A partir da versão 3.3 do Linux, o *OpenvSwitch* é incluído como parte do *kernel* e seus utilitários de espaço do usuário estão disponíveis para instalação no repositório de arquivos das distribuições mais populares. Para a realização dos testes da aplicação de segurança, o *OpenvSwitch* foi instalado em um microcomputador composto por um processador Intel(R) Core(TM)2 Duo CPU, modelo E6550 de 2.33GHz, 2 Gb de memória RAM, 250 Gb de *Hard Disc (HD) SATA*, placa de rede *dual port Gigabit Ethernet* e SO Linux Ubuntu Server versão 16.04 LTS kernel 4.4.0. Para a execução dos testes, foi instalada a versão 2.5 do *OpenvSwitch* disponível no repositório de arquivos do Linux Ubuntu. Esta versão tem suporte para se conectar através da versão 1.3 do *Openflow*, utilizada para a execução dos testes aqui descritos. Com os pacotes necessários instalados, foram seguidos os seguintes passos para a configuração:

- ❑ inicializou-se o serviço *OpenvSwitch*;
- ❑ criou-se a *bridge*;
- ❑ foram incluídas as interfaces de rede na *bridge*;
- ❑ as interfaces de rede foram configuradas para modo promíscuo;
- ❑ o modo de falhas do *OpenvSwitch* foi configurado para *secure*, para que a comunicação somente aconteça, se o *switch* estiver conectado a um controlador *OpenFlow*;
- e
- ❑ foi indicado o endereço IP e porta do controlador *OpenMul*.

Realizada a conexão, o controlador irá instalar um fluxo indicando para o *OpenvSwitch* encaminhar para o controlador todas as mensagens do tipo 0x88b8 GOOSE. Neste ínterim, os 10 testes com as mesmas 1.000 mensagens GOOSE foram repetidos. O histograma com a frequência dos valores medidos pode ser observado na Figura 43.

Figura 43 – Histograma do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando OpenvSwitch.

Os demais valores calculados no presente teste estão listados na Tabela 11.

Tabela 11 – Resultado estatístico do tempo de tráfego para verificar a segurança das mensagens GOOSE utilizando *OpenvSwitch*.

Cálculo	Valor
Maior	2,183 ms
Menor	1,397 ms
Média	1,824 ms
Moda	1,762 ms
Desvio Padrão	0,181 ms
Média + Desvio Padrão	2,005 ms
Média - Desvio Padrão	1,643 ms

Os resultados apresentados nesta medição comprovam que é possível executar a aplicação de segurança, utilizando o protocolo *OpenFlow* e obter latências coerentes com a necessidade das mensagens GOOSE.

4.8.6.1 Teste de *Throughput*

Para analisar o *throughput* deste novo cenário, repetiu-se o processo realizado anteriormente, para proceder a este novo ambiente configurado. A mensagem de 1500 *bytes* foi enviada para passar pelo *OpenvSwitch*, encaminhar para o controlador na plataforma ZYBO que analisa a segurança das mensagens e encaminha-as de volta para o *OpenvSwitch* que envia a mensagem pela outra porta. As mensagens foram enviadas a uma taxa de 100 Mbps, sem atrasos entre os pacotes, para diferentes quantidades de mensagens. O melhor caso, sem perder pacotes e para a maior quantidade de mensagens foi:

- ❑ 410 mensagens;
- ❑ 616.640 *Bytes* trafegados;
- ❑ variação de 4,838 a 5,626 *Mbps* de taxa de transmissão;
- ❑ variação de 0,888 a 1,033 segundos; e
- ❑ variação de 2,165 a 2,519 ms no tempo médio de processamento de cada mensagem.

Observa-se, neste caso, que, mesmo submetido a alto tráfego, o *OpenvSwitch* conectado ao *OpenMul*, executando as verificações de segurança no controlador, processa as mensagens GOOSE em um tempo coerente com a norma IEC-61850.

4.9 Evolução do Protocolo *OpenFlow*

Liderado pela ONF, o protocolo *OpenFlow* tem evoluído, rapidamente, permitindo o desenvolvimento de aplicações relevantes para as redes de comunicação de dados. Estes sistemas irão contribuir não somente para as redes de computadores, mas, também, para as redes das subestações de energia elétrica, possibilitando o desenvolvimento de ferramentas que contribuem para o controle, gerência, supervisão e segurança das redes de comunicação. Da versão 1.0 implementada no *switch* DATACOM utilizado nos testes, para a versão 1.3 utilizada no *OpenvSwitch*, o protocolo evoluiu muito. Dentre as implementações realizadas nesta evolução, pode-se citar (FOUNDATION, 2012):

- ❑ utilização de múltiplas tabelas com um pipeline mais flexível, na versão 1.0 o pipeline *OpenFlow* poderia internamente ser mapeado para várias tabelas, mas elas sempre agiam logicamente como uma única tabela;
- ❑ nova abstração de grupos permitindo ao *OpenFlow* representar um conjunto de portas como uma única entidade para encaminhar pacotes;
- ❑ inclusão do mecanismo de *fast failover*;

- ❑ novo suporte a VLANs;
- ❑ inclusão do suporte a MPLS, portas virtuais;
- ❑ controle de falha na conexão com o controlador;
- ❑ inclusão de uma flexível e compacta estrutura TLV chamada *OpenFlow Extensible Match* (OXM) que aumenta a flexibilidade do protocolo;
- ❑ suporte ao IPv6;
- ❑ suporte mais flexível para as decisões aplicadas a pacotes não classificados pelos fluxos presentes no *switch*;
- ❑ medição do tráfego, estatísticas e capacidades por fluxo;
- ❑ suporte a um limitador de taxa de transferência por fluxo;
- ❑ filtro de eventos por conexão; e
- ❑ conexões auxiliares entre outras.

Com a inclusão destes recursos, o protocolo tem se tornado cada vez mais flexível e robusto. A atualização do protocolo é contínua, a última versão lançada é a 1.5, trazendo outras novidades, abrangendo, cada vez mais, dispositivos e recursos da rede de comunicação. Muitos equipamentos que implementam este protocolo necessitam de uma maior maturidade para uso, em ambiente com mensagens de tempo crítico. Pfeifferberger e Du (2014) mediram a latência de *switches OpenFlow* e identificaram latências em torno de 50 a 60 milissegundos que inviabilizam o uso de mensagens GOOSE.

Apesar das ressalvas levantadas, o uso do protocolo *OpenFlow*, nas redes de comunicação das subestações de energia elétrica, é promissor. Lopes et al. (2015) por exemplo, propõem uma série de configurações e implementações, explorando os recursos do protocolo. Neste trabalho, não foram abordadas todas as mensagens do protocolo IEC-61850, porém, direcionando os experimentos para que se obtivesse segurança na comunicação por mensagens GOOSE, foram utilizadas as seguintes configurações:

- ❑ priorização máxima para as mensagens GOOSE na rede de comunicação; e
- ❑ separação dos grupos *multicast* por VLANs, uma recomendação da norma IEC-61850, utilizando a *tag* VLAN ID e o tipo da mensagem para criação dos fluxos no *switch OpenFlow*.

A implementação realizada, neste capítulo, buscou identificar e bloquear ataques do tipo *man-in-the-middle*. Estes ataques são resultantes da presença de um intruso na

rede de comunicação da subestação. A partir deste acesso, o *hacker* consegue capturar e falsificar mensagens, conforme testes realizados.

O capítulo seguinte descreve outro tipo de ataque muito comum em redes de comunicação de dados intitulado *Distributed Deny of Service* (DDOS). Neste tipo de ataque, o invasor coordena equipamentos com acesso à rede dos IEDs, para inundá-la com pacotes que irão dificultar a entrega das mensagens legítimas da comunicação a ser realizada.

Predição do Tráfego de Rede para Detecção de Ataques DDOS

Técnicas de prevenção de intrusão tradicionais, tais como *firewalls*, controle de acesso ou criptografia, têm falhado na proteção de redes e sistemas, diante da sofisticação dos ataques (WU; BANTHAF, 2010). Como resultado, os *Intrusion Detection Systems* (IDSs) tornaram-se um componente indispensável da infraestrutura de segurança, para detectar ameaças, antes que elas causem danos generalizados. As técnicas inteligentes para detecção de intrusão, como ferramenta para aumentar a precisão das decisões na identificação de um intruso vêm sendo utilizadas. Por isso, este capítulo apresenta uma revisão bibliográfica com artigos sobre ataques *Distributed Deny of Service* (DDOS), na rede de comunicação do Sistema Elétrico, bem como discute trabalhos que utilizam a Rede Neural Artificial (RNA) *Nonlinear Autoregressive Model with Exogenous Input* (NARX), para a previsão de séries temporais, de forma a contextualizar a aplicação para a predição do tráfego gerado por mensagens *Generic Object Oriented Substation Event* (GOOSE), nas redes de comunicação do Sistema Elétrico (SE), apresentada como parâmetro de referência para detecção de ataques DDOS.

5.1 Revisão Bibliográfica

DDOS é uma significativa ameaça para infraestruturas críticas das redes de comunicação do sistema elétrico, tais como as *Smart Grids*. Seu objetivo é incapacitar um equipamento, infraestrutura de rede ou de serviços, ao inundar um alvo com tráfego extremo de dados sob a forma de pedidos de comunicação (HURST; SHONE; MONNET, 2015). DDOS tem potencial para causar falhas infraestruturais críticas, impactando não somente no alvo, mas, também, causando um efeito cascata nas subestações e *Smart Grids*, efeito ocorrido devido à ausência de algum recurso de comunicação. O trabalho de Hurst, Shone e Monnet (2015) apresenta uma avaliação sobre o impacto das ameaças cibernéticas em redes de infraestrutura crítica, com foco em ataques DDOS.

Segundo Levesque et al. (2012), um dos desafios significativos das *Smart Grids* é a construção de uma nova infraestrutura de comunicação para conectar a rede de distribuição de energia e os clientes, de maneira segura. Ataques DDOS podem degradar a disponibilidade da rede. Os autores propõem um algoritmo adaptativo para gerenciar a confiabilidade e latência da rede. Iminentes ataques na infraestrutura das *Smart Grids* podem interromper as operações e a disponibilidade das informações (GUO et al., 2015). O artigo discute ataques DDOS na rede de comunicação *Advanced Metering Infrastructure (AMI)* onde são analisadas as etapas deste tipo de ataque e potenciais tecnologias de defesa.

Wang e Lu (2013) discutem segurança cibernética com foco nos requisitos de segurança, vulnerabilidades da rede, contramedidas aos ataques e protocolos de segurança da comunicação, nas arquiteturas das *Smart Grids*.

DDOS é abordado sob duas vertentes: ataques à camada de rede com intenção de gerar uma exaustão nos recursos, e a camada física com o intuito de congestionar as redes de comunicação. Filtragem de pacotes e limitação do tráfego são técnicas de mitigação destes ataques utilizadas, atualmente. As ameaças de ataques DDOS são iminentes nas redes de comunicação do sistema elétrico. Cabe à comunidade científica, desenvolver técnicas mais eficientes para diminuir o impacto desses ataques às redes de comunicação. Neste contexto, a análise do tráfego utilizando ferramentas inteligentes, pode apresentar resultados interessantes na detecção desta vertente de ataques cibernéticos.

Haviluddin e Alfred (2015), em seu artigo, afirmam que a análise de séries temporais e o uso de ferramenta para aprendizagem do tráfego da rede são importantes para garantir a qualidade dos serviços fornecidos. Nesse trabalho apresentam uma modelagem, utilizando redes NARX para previsão dos conjuntos de dados do tráfego de rede.

Andrade et al. (2014) apresentam uma previsão da demanda de energia elétrica em curtíssimo prazo baseada em RNAs NARX. Procedem, também, a uma comparação com outras RNAs de predição, demonstrando a superioridade da rede recorrente NARX para esta classe de problemas. Por sua vez, Abdel-Galil, El-Saadany e Salama (2003) apresentam uma rede neural Adaline como um preditor de sinal adaptativo. Através de um treinamento online, esta RNA é utilizada como detector de distúrbios de qualidade de energia elétrica baseado no erro calculado no preditor.

Para Gu, Liu e Feng (2011), detectar ataques à rede de comunicação é fundamental para proteger a informação. As RNAs têm recebido especial atenção, por fornecerem um reconhecimento inteligente da intrusão. No entanto, a taxa de detecção de intrusão é, frequentemente, afetada pelos parâmetros da estrutura das RNAs. Uma concepção inadequada do modelo da RNA pode resultar em uma baixa precisão de detecção. Para superar este problema, os autores apresentam uma abordagem de detecção de intrusão diferenciada baseada em um *Perceptron* multicamadas (PMC).

Observou-se, na literatura pertinente, que o uso de técnicas inteligentes tem contri-

buído, consideravelmente, para a qualidade dos IDSs, melhorando a precisão na detecção do intruso. O tráfego de mensagens GOOSE de prioridade na subestação, de acordo com IEC62351 (2007b), deve estar isolado por uma *Virtual Local Area Network* (VLAN) dos demais pacotes da rede de comunicação da subestação de energia elétrica. *A priori*, uma rede de comunicação composta por um único tipo de mensagem gera uma assinatura no tráfego passível de determinação de um padrão. Neste contexto, ferramentas inteligentes podem ser ainda mais eficientes, quando aplicadas em uma rede que recebe especificado tipo de pacote. Cabe portanto, o desenvolvimento de um sistema de predição que determine este tráfego, criando uma referência para a determinação do tráfego autêntico no qual variações que se distanciem desta referência sejam utilizadas para a identificação de ataques do tipo DDOS.

5.2 Redes Neurais Artificiais

RNAs são modelos computacionais inspirados no cérebro humano. Sua estrutura é composta por unidades de processamento de informações, neurônios artificiais, interligados por um grande número de conexões, sinapses artificiais, onde o conhecimento fica armazenado em matrizes que representam os pesos sinápticos (HAYKIN, 2001).

Silva, Spatti e Flauzino (2010) apresentam diversas características das RNAs. Algumas delas vão de encontro com às capacidades desejadas para o IDS. Dentre elas, podem ser citadas:

- ❑ Adaptação por experiência: é a capacidade de se modificar, de acordo com a apresentação sucessiva de exemplos, possibilitando a aquisição do conhecimento por experimentação. O tráfego na rede de comunicação é dinâmico e polimórfico. A capacidade da RNA de se adaptar, possibilita o desenvolvimento de aplicações que podem detectar ataques desconhecidos, mesmo depois de implementada.
- ❑ Capacidade de aprendizado: através de um algoritmo de treinamento, a RNA é capaz de extrair o relacionamento existente entre as variáveis que compõem a aplicação. Através deste aprendizado, ela pode identificar situações legítimas de funcionamento da rede de comunicação, bem como, situações de ataque aos equipamentos que à compõem.
- ❑ Habilidade de generalização: é a capacidade de produzir respostas corretas, mesmo para situações que não fazem parte do treinamento. Esta competência é desejada no projeto do IDS, porquanto os ataques podem produzir diferentes comportamentos na rede de comunicação.

- ❑ Facilidade de prototipagem: a implementação de boa parte das RNAs realizam o processo de treinamento, anteriormente à sua execução. A fase de treinamento é a mais complexa, em termos de implementação e esforço computacional.

Geralmente, os dados do processo são coletados para a realização do treinamento *off-line*. Após treinado, o conhecimento fica armazenado nos pesos sinápticos que são utilizados para a classificação dos dados, em tempo de execução. O processo de classificação é realizado por operações matemáticas menos complexas. Esta capacidade permite a utilização das RNAs em *softwares* e *hardwares* com relativa facilidade.

5.2.1 Redes Perceptron Multicamadas

Quando se pesquisa sobre RNAs, observa-se uma quantidade numerosa de trabalhos que implementam o PMC. Tal fato se deve à sua arquitetura versátil em relação à aplicabilidade.

Segundo Silva, Spatti e Flauzino (2010), ele pode ser utilizado para:

- ❑ aproximação universal de funções;
- ❑ reconhecimento de padrões;
- ❑ identificação e controle de processos;
- ❑ previsão de séries temporais; e
- ❑ otimização de sistemas.

O PMC é uma RNA supervisionada composta por uma camada de entrada, uma ou mais camadas de neurônios escondidas, e uma camada neural de saída. O algoritmo de aprendizado utilizado no PMC é denominado *backpropagation*. Este algoritmo consiste na propagação dos valores de entrada para a frente, camada a camada, até que se obtenha os valores nos neurônios da camada de saída calculados com os pesos sinápticos. Os valores de saída são comparados com os valores desejados e, depois, é calculado o erro. Através deste erro, os pesos sinápticos são ajustados para trás, camada a camada, até a primeira camada de neurônios. Findado o ajuste dos pesos, o processo se reinicia, para que novos valores de saída sejam calculados, conforme ilustra a Figura 44.

Para o desenvolvimento do sistema inteligente descrito neste capítulo foram analisados e classificados dados do tráfego da rede em situação de execução normal, evento e ataque, compondo um conjunto de treinamento que foi utilizado em um PMC. Uma vez treinados, os dados da RNA foram coletados, embarcados na plataforma *Zynq Board* (ZYBO) e utilizados para a classificação do tráfego gerado por mensagens GOOSE.

O presente capítulo descreve um sistema inteligente de previsão do tráfego gerado por mensagens GOOSE, na rede de comunicação das subestações. Esta predição do tráfego é

Figura 44 – Composição do PMC.

Fonte: Silva, Spatti e Flauzino (2010).

utilizada como referência para a detecção de ataques do tipo DDOS que têm como objetivo deixar equipamentos, e até mesmo toda a rede de comunicação, inoperantes, através da geração de uma sobrecarga de pacotes criada pelo intruso. O ataque consiste em invadir máquinas com acesso à rede a ser atacada e coordená-las para que façam, conjuntamente, requisições ao dispositivo alvo. Como todo equipamento tem uma capacidade limitada para atender requisições, ele deixará de enviar e receber mensagens legítimas que deveriam trafegar pela rede, no momento do ataque. As máquinas “zumbis” irão imputar elevado tráfego na rede de comunicação que irá divergir do tráfego previsto pelo sistema inteligente, tornando esta previsão uma ferramenta interessante para a detecção de ataques DDOS.

Para o desenvolvimento do sistema proposto, cabe medir o tráfego legítimo gerado por mensagens GOOSE na rede de comunicação. Variações que divergirem das previsões realizadas são consideradas situações de formação ou realização de ataque.

Inicialmente, vale destacar que o sistema foi desenvolvido para ser executado juntamente com o protocolo *OpenFlow*. Utilizando a configuração sugerida na Seção 4.9, é possível obter o tráfego de cada IED *publisher*, através dos fluxos cadastrados no *switch* para dividir os grupos *multicast*. Neste ínterim, foi examinado o tráfego gerado pela mensagem GOOSE, individualmente.

5.3 Definição do Conjunto de Treinamento do Sistema Inteligente

Para criar as mensagens que geram o tráfego na rede, foi utilizada a biblioteca *libiec61850* que implementa, em linguagem C, uma série de funcionalidades do padrão *International Electrotechnical Commission* (IEC)-61850, dentre elas um *publisher* GOOSE (ZILLGITH, 2016). Utilizando a *libiec61850* foi criada e enviada para a rede de comunicação uma mensagem GOOSE muito pequena com os menores valores possíveis para os campos do *goosePdu*, e contendo apenas um campo *boolean* no Conjunto de Dados (DATA-SET), totalizando 73 bytes. As mensagens GOOSE variam seu tamanho como resultado do incremento nos campos *sqNum* e *stNum*.

As variações no tamanho dos demais campos dependem de uma parada na comunicação. As mensagens GOOSE foram geradas na rede, simulando o funcionamento de um *Intelligent Electronic Device* (IED). À medida que as mensagens são geradas, elas vão aumentando de tamanho, até o valor limite do campo *sqNum*, caso nenhum evento tenha que ser informado pelo IED *publisher*, o que denota a situação normal de operação descrita na Subseção 4.6.1. Observando tal fato, mediu-se toda a sua extensão, para apenas um campo *boolean* no DATA-SET. Posteriormente, é acrescentado mais um campo *boolean* no DATA-SET e o tráfego de uma nova simulação do funcionamento do IED é novamente medido. O processo de acrescentar um campo *boolean* e medir, novamente, o tráfego, é realizado até o tamanho limite do *frame ethernet* de 1522 bytes com a tag VLAN (THALER, 1998), abrangendo, assim, toda a extensão de tráfego resultante da comunicação por mensagens GOOSE, entre IEDs em situação normal de operação.

Para medir o tráfego nas placas de rede foi instalado neste computador o *software* cacti (URBAN, 2011), que tem como função gerar gráficos de utilização da rede e dos equipamentos de comunicação, utilizando protocolos de gerência de rede. O *software* utiliza o protocolo *Simple Network Management Protocol* (SNMP) (IEC62351, 2010), para coletar informações de tráfego da placa de rede (Figura 45).

Figura 45 – Medição do tráfego gerado por mensagens GOOSE

As mensagens geradas com a biblioteca foram capturadas e salvas em um arquivo

Packet Capture (PCAP). Utilizando o *software tcpstat*, que retorna estatísticas de tráfego na rede ou em arquivos PCAP, foram medidos os *bits* por segundo (*bps*) trafegados em um determinado intervalo de tempo (HERMAN; COLOGNE, 2016). Como as mensagens foram geradas a cada 1 segundo, este mesmo intervalo foi utilizado para obter as medições do tráfego.

Para utilizar os dados medidos como conjunto de treinamento para a RNA, criou-se subconjuntos de 70 valores para cada variação medida. Estes valores foram utilizados como conjunto de treinamento para o sistema de predição desenvolvido, utilizando uma RNA NARX.

5.4 Funcionamento da RNA NARX

A RNA NARX possui uma arquitetura recorrente com realimentação global. Ela incorpora um PMC, explorando sua capacidade de mapeamento não-linear. Este modelo de RNA possui uma única entrada que é aplicada a uma memória de linha de atraso derivada de q unidades. Ela tem uma única saída que é realimentada para a entrada, através de uma outra memória de linha de atraso derivada, também com q unidades. O conteúdo destas duas linhas de atraso com derivações são utilizadas para alimentar a camada de entrada do PMC. O valor presente da entrada do modelo é representado por $u(n)$, e o valor correspondente da saída do modelo é representado por $y(n+1)$, isto é, a saída adiantada, em relação à entrada por uma unidade de tempo. Assim, o vetor de sinal aplicado à camada de entrada do PMC consiste em uma janela de dados constituídas das componentes:

- ❑ Os valores presente e passado da entrada, ou seja $u(n), u(n-1), \dots, u(n-q+1)$, que representam entradas exógenas originárias de fora da rede.
- ❑ Os valores atrasados da saída, ou seja, $y(n), y(n-1), \dots, y(n-q+1)$, sobre os quais é feita a regressão da saída do modelo $y(n+1)$ (HAYKIN, 2001).

Com isso, a rede é referida como um modelo auto-regressivo não-linear com entradas exógenas NARX. O comportamento dinâmico do modelo NARX é descrito pela Equação 1:

$$y(n+1) = F(y(n), \dots, y(n-q+1), u(n), \dots, u(n-q+1)) \quad (1)$$

na qual, F é uma função não-linear de seus argumentos. Fazendo-se a aproximação dessa função, por meio de uma rede neural PMC, a topologia resultante pode ser observada na Figura 46.

Figura 46 – Topologia de uma rede neural NARX

5.5 Descoberta da Melhor Topologia

Caracterizada a RNA adotada, prosseguiu-se para a descoberta da melhor topologia para o PMC. Para utilizar as medições dos *bps* do tráfego da rede como conjunto de treinamento, os dados foram normalizados entre -1 e 1, utilizando a Equação 2

$$y = \frac{2x - x^{max} - x^{min}}{x^{max} - x^{min}} \quad (2)$$

na qual y é a saída do valor normalizado, x é o valor do tráfego medido na rede de comunicação, x^{max} e x^{min} são o maior e menor valor do conjunto de treinamento, respectivamente.

Foram criadas 26.901 topologias de treinamento, variando a janela dos valores de entrada da rede entre 10 e 70 e variando os neurônios das duas camadas escondidas entre 5 e 25. A rede possui uma única saída que é realimentada, obedecendo ao modelo da RNA NARX, para a obtenção de novas predições. A Figura 47 ilustra o janelamento realizado na seleção dos dados para os arquivos de treinamento.

A ferramenta utilizada para a implementação da RNA foi a *Fast Artificial Neural Network* (FANN), uma biblioteca livre de código fonte aberto que implementa redes neurais artificiais multicamadas em linguagem C. Presente no repositório de arquivos do Linux, ela possui um *framework* para treinamento e testes do conjunto de dados para modelagem de uma RNA (NISSENY, 2016).

Atendendo à formatação necessária para utilização na biblioteca FANN, foram criados 61 arquivos com conjuntos de treinamento, nos quais cada arquivo possui a janela de valores formatados para a camada de entrada da RNA. Estes arquivos são carregados dinamicamente na aplicação, para a realização do treinamento para o qual foram utili-

Figura 47 – Janelamento realizado para a seleção dos dados de treinamento

zando 61 computadores do laboratório de informática do Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG), Unidade de Varginha. Para realização do treinamento foram desenvolvidos *shell scripts* que executam a RNA implementada em linguagem C, variando os valores da camada de entrada, das camadas neurais escondidas 1 e 2.

O algoritmo de treinamento utilizado na FANN foi o *Resilient-Propagation* (RPROP) (RIEDMILLER; RPROP, 1994) e a função de ativação usada foi a sigmóide simétrica para todas as camadas neurais. Ao treinar cada topologia, é registrado o declínio do Erro Quadrático Médio (EQM), a cada 100 épocas, fornecido pela própria biblioteca FANN. As condições de parada do treinamento foram 1.000 épocas ou EQM, menor que ou igual a 10^{-5} .

A biblioteca FANN possui um formato próprio para armazenar em arquivo a estrutura e os pesos sinápticos da RNA. Este arquivo, possui todas as informações necessárias para a execução da RNA treinada. Neste caso, foram salvos, no formato definido pela FANN, os dados de todos os 26.901 treinamentos realizados.

5.5.1 Validação Cruzada

De posse dos dados de treinamento, foi realizada uma validação cruzada. Os dados de teste e verificação foram gerados com novas medições do tráfego, na rede para mensagens com DATA-SET possuindo valores do tipo *float-point*. Toda a extensão de tamanho das mensagens GOOSE foi novamente medida, e os valores foram re-amostrados em 70 valores para cada variação medida, conforme realizado no conjunto de treinamento.

Foram criados 61 arquivos utilizando os dados do conjunto de teste, variando a janela dos valores da camada de entrada de 10 a 70. As topologias foram avaliadas calculando-se o EQM dado pela Equação 3 (SILVA; SPATTI; FLAUZINO, 2010)

$$EQM = \frac{1}{N} \sum_{i=1}^N (d_i - y_i)^2 \quad (3)$$

e o Erro Percentual Médio Absoluto (MAPE) Equação 4 para o conjunto de teste citado

$$MAPE = \frac{1}{N} \sum_{i=1}^N \left| \frac{d_i - y_i}{d_i} \right| \times 100\% \quad (4)$$

para ambas as equações, N é o número de amostras, d é o valor desejado e y é o valor de saída estimado pela RNA.

O EQM é considerado uma medida importante para a escolha do modelo, já que os maiores desvios resultam em um peso grande para a média, enquanto os baixos valores resultam em um erro menor. Assim, o ideal é adotar como modelo aquele que minimize a média. Este erro pode ser utilizado para definir o modelo de previsão mais preciso. O MAPE é a média de todos os erros absolutos percentuais, fornecendo uma indicação do tamanho médio do erro, expresso como uma porcentagem do valor observado, o que permite uma avaliação mais intuitiva da topologia (ZULYADI; SEMBIRING, 2015).

Utilizando a função de testes da FANN, foi obtida a saída de cada amostra do conjunto de teste e com ela, foi calculado o erro quadrático médio EQM e o erro percentual médio absoluto MAPE para todas as topologias treinadas. As RNAs foram ranqueadas e as 10 melhores, ou seja, as que possuem menores erros, o treinamento foi repetido 5 vezes. A Figura 48 ilustra os passos de treinamento e teste realizados para a definição da topologia utilizada na predição do tráfego gerado por mensagens GOOSE.

Sinteticamente, os dados foram gerados e coletados em um computador, onde as mensagens foram geradas em *localhost* e o tráfego foi medido. A medição ocorreu de duas formas: utilizando o *software* cacti, através do protocolo SNMP, e capturando as mensagens na rede, salvando-as em arquivos PCAP e calculando o tráfego presente no arquivo, através do *software* tcpstat. Com os *bps* medidos, normalizaram-se os dados e criaram-se os arquivos de treinamento. Realizaram-se os treinamentos e as redes criadas foram testadas, utilizando novas leituras de tráfego da rede. Os resultados foram classificados e para os 10 melhores realizaram-se 5 novos treinamentos em cada um deles. As RNAs resultantes deste novo treinamento foram classificadas para possibilitar a escolha da melhor topologia.

A Tabela 12 apresenta os resultados do erro das 5 melhores topologias para os últimos testes realizados.

Observando os resultados apresentados na Tabela 12, cabe destacar que a topologia que obteve o menor valor do EQM foi com 31 medições na janela de entrada, 15 neurônios na cada escondida 1, e 7 neurônios na camada escondida 2. Diferentemente, a topologia que obteve menor MAPE foi com 60 medições na janela de entrada, 5 neurônios na camada

Figura 48 – Passos de treinamento e teste para definição da topologia da RNA

Tabela 12 – Resultados do teste para definição da topologia da RNA.

Tamanho da Janela	Camada Escondida 1	Camada Escondida 2	EQM	MAPE
31	15	7	$4,006 \times 10^{-5}$	1,046%
28	9	5	$4,019 \times 10^{-5}$	2,805%
26	18	12	$4,040 \times 10^{-5}$	1,509%
60	5	23	$4,178 \times 10^{-5}$	0,747%
19	5	23	$4,536 \times 10^{-5}$	1,444%

escondida 1, e 23 neurônios na camada escondida 2. Neste contexto, estas duas topologias são fortes candidatas à utilização na aplicação de predição desenvolvida.

A topologia com 31 neurônios na camada de entrada, 15 na camada escondida 1, e 7 na camada escondida 2 foi a escolhida por:

- trabalhar com menos neurônios, necessitando de menor poder computacional para sua execução;
- trabalhar com uma janela de entrada menor, o que denota uma maior eficiência na predição, ou seja, necessita-se de menos medições para a previsão dos passos

posteriores da rede.

A Figura 49 ilustra o ajuste ocorrido no EQM durante o treinamento da topologia utilizada.

Figura 49 – Ajuste do EQM durante o treinamento.

Escolhida a topologia, a curva do valor de saída da rede e do valor desejado pode ser observada na Figura 50.

Figura 50 – Valores desejado e obtido por meio da predição com a RNA NARX.

5.6 Predição dos Passos da RNA

A implementação descrita até o momento, calcula somente um passo de predição. Para determinar uma maior quantidade de passos à frente, o valor calculado para a previsão de um passo, foi apresentado como entrada para a RNA NARX, para que ela estime o passo seguinte. A predição calculada no passo dois foi inserida na entrada, para o cálculo do terceiro passo e, assim, sucessivamente (Figura 51).

Figura 51 – Passos de execução da predição.

No processo de estimação dos dados realizado pela RNA NARX os valores antigos são descartados e as novas predições vão compondo a entrada da rede, até que todos os passos de previsão sejam calculados. Cabe ressaltar que a cada novo valor de tráfego medido, o processo se reinicia, ou seja, a cada segundo, utilizam-se as medições presentes no banco de dados do tráfego, para dar início a um novo processo de predição.

Para cada passo estimado, calcularam-se o EQM e o MAPE da predição realizada, até o limite de 5% de erro (Tabela 13).

Com erro de até 5% foram realizados 62 passos de predição. Como as leituras foram

Tabela 13 – Resultado do erro para os passos de predição.

Passo de Predição	EQM	MAPE
1	$4,006 \times 10^{-5}$	1,046%
2	$4,109 \times 10^{-5}$	1,095%
3	$4,196 \times 10^{-5}$	1,141%
10	$4,796 \times 10^{-5}$	1,453%
20	$6,395 \times 10^{-5}$	2,061%
30	$8,940 \times 10^{-5}$	2,816%
40	$11,071 \times 10^{-5}$	3,514%
50	$13,355 \times 10^{-5}$	4,210%
60	$15,876 \times 10^{-5}$	4,918%
61	$16,130 \times 10^{-5}$	4,988%
62	$16,382 \times 10^{-5}$	5,058%

realizadas a cada 1 segundo, seriam 1 minuto e 2 segundos de predição do tráfego das mensagens GOOSE, porém, mais passos podem ser calculados, sabendo-se que haverá um aumento proporcional no erro.

Para uma visualização do aumento no EQM, a Figura 52 apresenta graficamente sua evolução em relação aos passos de previsão realizados.

Figura 52 – Erro quadrático médio em relação aos passos de predição.

Percentualmente, a evolução do erro nos passos de predição medido através do MAPE é apresentado na Figura 53.

Os resultados demonstram o aumento do erro à medida que os passos de predição são realizados. Este comportamento é esperado, porque, a cada passo de previsão, o erro do passo anterior é acrescido na estimação seguinte.

A Figura 54 ilustra uma predição realizada com a RNA implementada.

Figura 53 – Erro percentual médio absoluto em relação aos passos de predição.

Figura 54 – Predição dos valores de tráfego com a RNA NARX.

Na Figura 54 foram apresentados 31 passos de cálculo da saída da RNA, comparando com o valor desejado medido, de acordo com o tráfego de rede. Posteriormente, são apresentados 62 passos de predição realizados pela RNA NARX treinada.

A previsão do tráfego em si, contribui para aumentar a segurança na rede de comunicação que utiliza mensagens GOOSE. A exibição, em um sistema supervisor, da predição e do erro envolvido em seu cálculo, são norteadores para identificar tráfego espúrio, ou mesmo a formação ou realização de um ataque DDOS. Valores que se distanciarem da previsão, ou um grande aumento no erro da RNA podem ser indicadores de ataques DDOS.

5.7 Análise do Erro para Situação de Evento no Sistema Elétrico

A RNA NARX apresentada é capaz de prever o tráfego de uma situação normal de operação do SE, em que o IED *publisher* envia mensagens em uma determinada cadência, para manter a comunicação com seus *subscribers*. Caso algum evento tenha que ser informado, os IEDs irão diminuir o tempo de emissão entre as mensagens, informando o evento ocorrido, o que gera um maior tráfego durante este tempo, até que se estabilize o período de envio das mensagens novamente, conforme explicado na Seção 2.7. A RNA descrita, não foi treinada para este tipo de situação, o que irá gerar um alto erro na predição, durante este evento. O valor deste erro torna-se uma ferramenta interessante para a identificação de eventos que ocorrem na rede. Cabe, ainda, investigar a diferença entre o erro de predição causado por um evento no sistema elétrico, situação plausível de acontecer, e o erro de predição causado por um ataque DDOS.

Para a análise do comportamento da RNA para esta nova situação de funcionamento do SE, foi gerado um novo conjunto de testes com 200 mensagens. Cada mensagem possui o tamanho de 300 *bytes* que geram um tráfego de 2.288 *bps*. Visando a abordar situações distintas de comportamento do erro, foram realizados eventos nos tempos 6, 43, 56, 93, 106, 118, 156, 158 e 161 segundos.

A implementação do evento realizada na biblioteca *libiec61850* se comporta enviando uma mensagem, no instante da ocorrência do evento e, 4 milissegundos depois, 8 milissegundos depois e, assim, sucessivamente, dobrando o tempo entre as mensagens, até retornar à cadência de estado normal de operação de 1 segundo. Caso o evento ocorra de forma sincronizada com a leitura do tráfego da rede, o tráfego de 9 mensagens no tempo de 1 segundo é medido. Essas 9 mensagens geram um tráfego de 20.592 *bps*, ou seja, 9 vezes o tráfego de 2.288 *bps* gerado por uma mensagem de 300 bytes. Caso não esteja sincronizado, o tráfego gerado por essas 9 mensagens estará dividido em duas amostras.

Para analisar a influência do evento no tráfego, foi calculado, neste teste, o Erro Relativo Percentual (ER%) de cada amostra, através da Equação 5.

$$ER\% = \frac{|y - d|}{d} \times 100 \quad (5)$$

na qual y é o valor de saída estimado pela RNA e d é o valor desejado. A Figura 55 ilustra o erro relativo percentual para as 200 amostras com os eventos nos tempos descritos.

O gráfico apresentado possui 4 regiões bem distintas separadas por medições nas quais o erro está em torno de 1%. Essas regiões onde o erro é baixo, correspondem às amostras de situação normal de operação que não foram influenciadas pelo tráfego gerado pelos eventos. As demais regiões apresentam picos e variações bruscas do erro pela intervenção do tráfego dos eventos. A primeira região que se diferencia da situação normal, inicia no

Figura 55 – Erro relativo percentual para amostras com eventos no sistema elétrico.

tempo 6, correspondente ao erro gerado pelo evento inserido nesta amostra. O aumento repentino no tráfego gerou um erro de 89%, valor que diminui no segundo seguinte para 34% e fica variando entre 13 e 41%, durante 31 amostras. No segundo 38, o erro retorna para o patamar de 1% de situação normal de operação. A variação do erro durante 31 amostras, coincide com a janela de dados utilizada na camada de entrada da RNA, ou seja, mesmo depois que o tráfego retornar à situação normal de operação, a predição continuará sob influência do tráfego gerado pelo evento, por um período correspondente ao tamanho da janela de entrada dos dados na RNA.

A segunda região, em que novos picos de erro ocorrem, inicia no tempo 43 correspondente ao evento realizado nesta amostra. O comportamento do erro nos passos seguintes ao evento é muito parecido com o observado no evento anterior, porém, sem transcorrerem as 31 amostras para que o erro se estabilize, foi inserido um novo evento em 56 segundos. Os dois eventos se misturam e a variação do erro passa para um patamar maior neste caso. No tempo 88, são ultrapassados os 31 segundos da janela dos valores de entrada da RNA, após a realização do segundo evento desta região em análise, e o erro se estabiliza.

A terceira região de novos picos de erro inicia no segundo 93, resultado do evento ocorrido nesta amostra, e perdura por um tempo maior que a região descrita anteriormente. Neste caso, o evento ocorrido na amostra 106 acontece, sem que tenha transcorrido o tempo para que o erro do evento anterior, e também o evento inserido no tempo 118, se estabilizem. Portanto, são três eventos que se misturam, apresentando oscilações no erro bem variadas. No segundo 150, o erro retorna a patamares de situação normal de operação.

A quarta região observada apresenta alta porcentagem de erros, a partir do segundo 156, sendo esta região mais densa, com a maioria dos erros em patamares maiores que as regiões anteriores. Este comportamento é resultado de três eventos que ocorrem em tempos muito próximos, 156, 158 e 161 segundos. O alto tráfego medido em um curto

espaço de tempo gerou esta assinatura para o erro. Mesmo com esta grande variação, o erro retornou para patamares de situação normal de operação na amostra 193, quando a janela de 31 amostras, após o evento no segundo 161, é ultrapassada.

Os resultados deste teste apontam que uma janela de valores menores para a camada de entrada, contribui para diminuir a propagação do erro, nos passos seguintes de predição, o que reforça a decisão de escolher a topologia da RNA com menor janela de entrada. Conclui-se, portanto, que a variação observada na Figura 55 caracteriza a assinatura dos eventos ocorridos na rede de comunicação.

5.8 Análise do Erro para Situação de Ataque DDOS no Sistema Elétrico

Por fim, passou-se para a investigação do tráfego em situação de ataque DDOS. Inicialmente, é importante relatar as situações da rede, para que este ataque chegue aos IEDs. Um ataque desta natureza é realizado, explorando as formas de comunicação que os dispositivos que compõem esta rede possuem. Em uma rede na qual os equipamentos se comunicam, através do protocolo IEC-61850. As mensagens *multicast* com as quais os IEDs podem trabalhar são: GOOSE e *Sampled Values* (SV). As demais mensagens utilizarão o protocolo TCP/IP como suporte para comunicação. Para a comunicação através do protocolo TCP/IP, ferramentas consolidadas de detecção de intrusão devem ser utilizadas, seguindo as recomendações da norma IEC-62351-7 (IEC62351, 2010).

A comunicação através de mensagens SV deve estar isolada por VLANs, conforme foi realizado com as mensagens GOOSE. Utilizando o protocolo *OpenFlow*, é possível obter o tráfego de cada VLAN, independentemente. Neste caso, o tráfego de mensagens SV não interfere no tráfego de mensagens GOOSE e ambos não afetam o tráfego TCP/IP.

Para realizar um ataque DDOS a um IED que se comunica utilizando mensagens GOOSE na estrutura mencionada, o *hacker* deve inundar a rede de mensagens GOOSE com a respectiva VLAN. Utilizando a aplicação desenvolvida no Capítulo 4, este ataque, dificilmente, irá atingir o IED, porém, a inundação de mensagens afetará o *switch* concentrador da rede, causando o mesmo efeito de negação do serviço de entrega de mensagens. Neste contexto, o ataque DDOS realizado, para verificar a eficiência do método de detecção aqui descrito, foi aplicado, utilizando mensagens GOOSE. A Figura 56 ilustra o funcionamento deste ataque na infraestrutura utilizada para implementação da RNA.

Para analisar o efeito de um ataque desta natureza, foi medido o tráfego gerado por mensagens GOOSE, aumentando, uma a uma, para averiguar o comportamento do erro relativo percentual das amostras. Iniciando o processo, o software *Wireshark* é colocado em execução, para que sejam capturadas as mensagens GOOSE enviadas na interface *loopback* do computador. Utilizando a biblioteca *libiec61850*, foram geradas mensagens GOOSE em situação normal de execução, ou seja, uma mensagem sendo enviada a cada se-

Figura 56 – Estrutura do ataque DDOS.

gundo. Transcorridos 19 segundos de execução, uma nova mensagem GOOSE, na mesma cadência, é enviada. Ao atingir 80 segundos, o processo é finalizado e as mensagens capturadas no *Wireshark* são salvas em um arquivo PCAP. O processo se repete, aumentando, gradativamente, a quantidade até 30 mensagens. Para analisar o comportamento para grandes quantidades de mensagens, repetiu-se o processo para 62, 110 e 901 mensagens.

As mensagens geradas para a realização deste teste possuem 300 bytes, o que gera um tráfego de 2.288 *bps*. Ao aumentar o número de mensagens, o tráfego aumenta, proporcionalmente.

De posse dos arquivos PCAP, mediu-se o tráfego, utilizando o *software tcpstat* e criaram-se os arquivos de teste para a RNA desenvolvida, com os dados da aferição realizada.

Para cada conjunto de teste, foi calculado o erro relativo percentual das amostras. A Figura 57 ilustra os valores do erro para os ataques realizados.

Figura 57 – Erro relativo percentual das amostras com ataques no sistema elétrico.

Foram apresentados 80 valores no gráfico onde, na amostra 18, ocorre o ataque. Neste

ponto, o erro tem um pico e vai diminuindo até que se estabiliza em 2% na amostra 50. Este comportamento é observado para os ataques com 2, 3, 4 e 5 mensagens. O fato do erro se estabilizar na amostra 50 se justifica pelo mesmo motivo apresentado nas situações de evento. A janela de 31 valores de entrada da RNA foi ultrapassada e o tráfego medido está estável. Apesar de ser um tráfego de mais de uma mensagem, seu valor está dentro do domínio de treinamento realizado.

A partir do ataque com 6 mensagens, o valor medido do tráfego fica acima do limite superior utilizado no treinamento que foi de de 12.064 *bps*, gerando um grande aumento na porcentagem de erro, quando este atinge uma estabilidade. A partir deste ponto, observa-se, também, que, ao aumentar a quantidade de mensagens, o erro tende a se estabilizar, cada vez mais rapidamente. Assim sendo, o ponto de estabilidade passa a estar antes dos 50 segundos, ou seja, antes de ultrapassar a janela de 31 amostras da camada de entrada. À medida que os valores do tráfego vão aumentando, o erro se estabiliza em valores maiores, porém, a diferença entre os pontos de estabilidade diminui.

Constata-se, portanto, que a principal diferença entre um evento e uma situação de ataque está no ponto de estabilização da rede. Pode-se inferir que, a partir do momento em que se detectou um alto erro e este erro passou a se estabilizar com valores maiores que 10%, esta situação pode ser caracterizada como um ataque DDOS.

Para uma análise mais detalhada do comportamento da estabilidade do erro, foi produzido um novo gráfico com os valores do erro para o ponto 56 no qual todos os erros estão estabilizados (Figura 58). Para não criar descontinuidades no gráfico, os valores entre 30 e 62, entre 62 e 110 e entre 110 e 901 foram calculados, utilizando uma interpolação logarítmica, através da função `interp1` do MATLAB.

Figura 58 – Comportamento do erro relativo percentual para 56 segundos em função dos ataques utilizando mensagens GOOSE.

Observa-se no gráfico que, após ultrapassar o limiar superior do treinamento realizado na RNA, houve um aumento gradativo no erro. No início, o aumento é acelerado, porém,

à medida que o erro se aproxima de 100%, o seu crescimento é menor, tendendo a se estabilizar.

Outra relação importante de se analisar, quanto ao erro nas situações de ataque, são as diferenças entre a amostra atual e a anterior, quando o erro estiver estabilizado, para que se tenha a real caracterização do seu crescimento. Neste contexto, calculou-se a diferença entre os erros para o mesmo ponto 56, utilizado no gráfico anterior, criando a Figura 59.

Figura 59 – Ilustração das diferenças entre o erro relativo percentual para 56 segundos em função dos ataques utilizando mensagens GOOSE.

Observa-se no gráfico que, até a amostra 5, a diferença entre os erros é ínfima, pois a RNA possui a capacidade de prever seu tráfego com boa margem de precisão. Da amostra 5 para a 6, ocorre o maior salto, demonstrando que o tráfego da mensagem excedeu o limite treinado na RNA. Para os ataques com 6 mensagens ou mais, a diferença entre os erros diminui, até chegar a valores muito baixos. Apresentaram-se os 130 primeiros valores calculados, referentes à diferença entre os erros, porém, observou-se uma estabilização na diferença, a partir deste ponto.

A principal diferença entre a situação de evento e um ataque está na estabilização do erro. A situação de evento no sistema é transitória. As situações medidas nos ataques proveem caracterizações do tráfego que possuem um pico de aumento, mas são contínuas. Para analisar o comportamento do erro para situações não contínuas, realizou-se uma última medição em que os ataques são variados. A Figura 60 apresenta o gráfico dos erros para este novo cenário.

O teste foi coordenado para começar com modificações espaçadas na quantidade de mensagens utilizadas no ataque. Durante o seu decorrer, a alteração na quantidade de mensagens fica mais curta.

O gráfico inicia, apresentando o erro em situação normal de operação e no ponto 8 é apresentado para a RNA o tráfego de um ataque com 6 mensagens. O comportamento do erro é semelhante ao apresentado na Figura 57. O ataque com 6 mensagens perdura, até

Figura 60 – Ilustração do erro relativo percentual com variações no tráfego de ataque utilizando mensagens GOOSE.

a amostra 48, na qual o tráfego passa a ser de um ataque com 20 mensagens. Observa-se, neste ponto, o aumento do erro e sua estabilização. Posteriormente, na amostra 68, o ataque passa a ser realizado com 10 mensagens. Seu comportamento é semelhante ao anterior, porém, há um percentual de erro menor, porque são menos mensagens. No ponto 108, ocorre uma nova alteração no tráfego para 15 mensagens; no ponto 113, outra mudança para 30 mensagens. No ponto 118, ocorre a última alteração para 25 mensagens.

Analisando o comportamento exibido no gráfico, observa-se que a estabilização do erro, durante a alteração no tráfego, é quase instantânea. Alguns patamares de erro retratados no gráfico, possuem descontinuações para baixo, ou para cima. Este comportamento é resultado de pequenos assincronismos entre o intervalo da medição e a quantidade de mensagens medidas no referido intervalo, variando em uma mensagem para cima ou para baixo.

Conclui-se, portanto, que o alto valor do erro relativo percentual é uma condição de alerta para o operador que supervisiona a rede. Caso este erro se estabilize acima de 10%, pode-se considerar que esta rede está sob influência de um tráfego espúrio que pode comprometer seu funcionamento. Neste caso, a duração da anormalidade é a principal medida para a detecção do ataque DDOS.

Analisando o gráfico da Figura 60 e equiparando-o ao apresentado na Figura 57, observa-se que o tráfego de ataque possui uma assinatura característica no erro relativo percentual da RNA. Esta propriedade permite analisar a gravidade do ataque baseado na influência do tráfego medido na rede de comunicação. Quanto mais próximo de 100% o erro estiver, mais a rede de comunicação está sendo degradada. O limiar para detecção do ataque, observando o erro relativo percentual, é de 10%. Caso o erro calculado a partir do tráfego de rede se estabilize acima desta fronteira, considera-se que a rede de comunicação está sob ataque DDOS.

5.9 Desenvolvimento do Sistema de Detecção de Ataques DDOS no Sistema Elétrico

O sistema de Detecção de Ataques DDOS foi dividido em três partes. A primeira consiste em um *script* em *python* que faz a leitura do tráfego a cada segundo, salva os valores em um banco de dados SQLite¹ e cria os arquivos para a execução da RNA. A segunda parte executa a RNA NARX treinada para o cálculo dos 62 passos de predição e a terceira parte realiza o cálculo do erro relativo percentual dos 100 últimos valores selecionados no banco de dados. Nas partes dois e três do *software*, os resultados são salvos em arquivos *Comma-Separated Values* (CSV). A partir dos valores do erro relativo percentual foi implementado ainda, um *software* para classificar a situação de funcionamento do SE em normal, evento ou ataque DDOS. A Figura 61 ilustra a execução dos *softwares*.

Figura 61 – Estrutura de execução dos *softwares* para Detecção de Ataques DDOS.

5.9.1 Sistema de Leitura, Preenchimento do Banco de Dados e Criação dos Arquivos da RNA

Esta parte do sistema tem como finalidade comunicar com o controlador, para coletar os dados do tráfego de rede de um determinado fluxo. O *script* utiliza a *Application Interface* (API) REST do *OpenMul*, para coletar os valores do tráfego de rede e preencher um banco de dados *SQLite*. Ao completar 31 medidas, o *script* passa a salvar, em um

¹O SQLite é uma biblioteca que implementa um banco de dados *Structured Query Language* (SQL) transacional de domínio público (SQLITE, 2014).

arquivo texto, os valores normalizados para a execução com a biblioteca FANN da RNA treinada, para predição do tráfego. Ao completar 131 leituras, o *script* passa a salvar em outro arquivo texto, também formatado para a execução na biblioteca FANN, os valores normalizados para que seja calculado o erro relativo percentual. A lógica de execução do *software* é descrita no Algoritmo 6.

Algoritmo 6 Coleta do tráfego, preenchimento do banco de dados e criação dos arquivos texto.

Entrada: Tráfego de Rede

Saída: Arquivos Texto

1. Conecta ao banco de dados.
2. Cria a tabela *fluxo*, caso ela não exista.
3. $ListaSwitch \leftarrow SwitchesConectados$
4. **Para** *switch* **IN** *ListaSwitch* **Faça**
5. $ListaFluxo \leftarrow FluxosdoSwitch$
6.   **Para** *fluxo* **IN** *ListaFluxo* **Faça**
7.   **Se** 88b8 **IN** *fluxo* **Então**
8.   Solicita as informações específicas do fluxo.
9.   Seleciona a informação *stat*.
10.   **Se** $tamanho(stat) > 0$ **Então**
11.   Insere *IDdoSwitch*, *IDdoFluxo* e informações estatísticas do fluxo, no banco de dados.
12. $registros \leftarrow$ Seleciona *bps* quando *IDdoSwitch* e *IDdoFluxo* atuais forem iguais ao do banco de dados, ordenado decrescente, com limite de 131 registros.
13.   **Se** $Número de Registros \geq 31$ **Então**
14.   Cria o arquivo texto para o cálculo da predição.
15.   **Para** *linhaPred* **IN** *registros* **Faça**
16.   Normaliza *linhaPred*.
17.   Salva *linhaPred* no arquivo texto.
18.   **Fim Para**
19.   Fechar p arquivo texto para o cálculo da predição.
20.   **Fim Se**
21.   **Se** $Número de Registros == 131$ **Então**
22.   Cria o arquivo texto para o cálculo do erro relativo percentual.
23.   **Para** *linhaErro* **IN** *registros* **Faça**
24.   Normaliza *linhaErro*.
25.   Salva *linhaErro* no arquivo texto.
26.   **Fim Para**
27.   Fecha o arquivo texto para o cálculo do erro relativo percentual.
28.   **Fim Se**
29.   **Fim Se**
30.   **Fim Se**
31.   **Fim Para**
32. **Fim Para**

A implementação foi realizada em *python*, a qual se inicia com a conexão com o banco de dados *SQLite*. Ao executar o comando de conexão, o arquivo do banco de dados é criado, caso ele não exista. O comando SQL para criar a tabela *fluxo* é executado, caso esta tabela exista. Posteriormente, são executados os comandos *Hypertext Transfer Protocol (HTTP)*, para requisitar ao controlador a lista dos *switches* retornados em formato *JavaScript Object Notation (JSON)*. Esta lista é percorrida no primeiro *loop*, linha 4, para que sejam criados os arquivos para execução, na RNA, de todos os *switches*. Utilizando a informação do campo *dpid*, é realizada uma nova requisição ao controlador, para obter a lista de fluxos do *switch*. Esta lista é percorrida no segundo *loop*, linha 6.

Para encontrar os fluxos correspondentes à mensagem GOOSE, foi incluído o teste da linha 7. Na linha 8, são solicitadas as informações específicas dos fluxos referentes à mensagem GOOSE, para que sejam selecionados os dados estatísticos do tráfego, campo *stat*, que são: pacotes por segundo (*pps*), quantidade de *bytes* (*byte_count*), *bits* por segundo (*bps*), quantidade de pacotes (*pkt_count*) e tempo de vida (*alive*). As informações estatísticas do tráfego separado por fluxo foram inseridas na versão 1.3 do *OpenFlow*. Como o controlador pode trabalhar com *switches* de versões anteriores, foi inserida a condição da linha 10 do algoritmo.

Na linha 11, o comando de inserção dos registros no banco de dados é realizado e são inseridos os valores *Identifier* (ID) do *switch*, ID do Fluxo e os campos com informações estatísticas do fluxo: *pps*, *byte_count*, *bps*, *pkt_count* e *alive*. Além desses campos, o banco de dados possui um campo ID, auto incremento e chave primária utilizado para a indexação da tabela, sendo importante para a seleção e ordenação dos registros. O comando seguinte, linha 12, faz a seleção do campo *bps* quando: o ID do *switch* e o ID do fluxo atuais, forem iguais aos valores do banco de dados. A ordenação decrescente dos valores foi feita, para que fossem selecionados os registros mais novos. O limite de 131 registros foi incluído, porque, esta é a quantidade máxima de valores utilizadas no arquivo de testes da RNA.

Com o resultado da consulta na variável *registro*, a quantidade de registros é verificada para que, ao chegar em 31, os dados fossem normalizados utilizando a Equação 2 e salvos no arquivo texto para a realização da predição. Ao chegar em 131, os dados foram normalizados e salvos no arquivo para o cálculo do erro relativo percentual. Cabe ressaltar que os registros do banco de dados foram formatados para comporem o arquivo texto, tanto para a predição, quanto para o cálculo do erro, tendo ambos que serem invertidos, devido ao retorno da ordenação dos registros realizados no comando SQL.

O *software* em *python* desenvolvido foi salvo na pasta do *OpenMul*, na qual foi criado um *shell script* que cria a pasta onde os arquivos são salvos, e executa o *software* a cada segundo. Foi inserida uma chamada a este *script* no comando de início do *OpenMul*, para que ele executasse, durante todo o tempo que o controlador estiver em execução. Foi inserida no comando de parada do *OpenMul* a finalização da execução do *script* que

modifica o nome da pasta onde os arquivos são salvos, incluindo a data e a hora de término da execução, assim, ficando registrado um conjunto de arquivos para cada execução do controlador. Como resultado da execução deste *script*, obtém-se o arquivo do banco de dados *SQLite*, com as informações estatísticas dos fluxos correspondentes às mensagens GOOSE e dois arquivos de teste da RNA, para cada *switch* e cada fluxo selecionado.

5.9.2 Software para o Cálculo da Predição

Esta parte do sistema é realizada logo após a execução do *software* em *python*; o mesmo *shell script*, utilizado para iniciar o código em *python*, é utilizado para dar início à execução deste. Este *software* foi implementado em linguagem C, e compilado incluindo a biblioteca FANN. O *software* cria a RNA, a partir do arquivo salvo na fase de treinamento explicada na Seção 5.5, carrega o arquivo de testes, criado na Subseção 5.9.1 deste capítulo, calcula os passos de predição e salva o resultado em um arquivo CSV. O Algoritmo 7 descreve a execução do *software* implementado.

Algoritmo 7 Calcula os passos de predição.

Entrada: Arquivo de teste para predição do tráfego

Saída: Arquivos CSV

1. *dadosRNA* = CaminhodoArquivodaRNATreinada;
  2. *rna* = *fann_create_from_file*(*dadosRNA*);
  3. *dadosTeste* = CaminhodoArquivodeTestesdaRNA;
  4. *dados* = *fann_read_train_from_file*(*dadosTeste*);
  5. Cria o arquivo CSV para salvar os dados da predição.
  6. **Para** ($i = 0; i < 62; i++$) **Faça**
  7. $\text{calculado} = \text{fann\_test}(\text{rna}, \text{dados} \rightarrow \text{input}[0], \text{dados} \rightarrow \text{output}[0]);$
  8. $\text{saida} = \text{cálculo da saída } \text{calculado}[0] \text{ normalizada, para os bps da entrada original.}$
  9.   Salva os valores de saída no arquivo CSV.
  10.   **Para** ($j = 1; j \leq 30; j++$) **Faça**
  11. $\text{dados} \rightarrow \text{input}[0][(j - 1)] = \text{dados} \rightarrow \text{input}[0][j];$
  12.   **Fim Para**
  13. $\text{dados} \rightarrow \text{input}[0][30] = \text{calculado}[0];$
  14. **Fim Para**
  15. Fecha o arquivo CSV
  16. *fann_destroy_train*(*dados*);
  17. *fann_destroy*(*rna*);
-

A implementação foi realizada em linguagem C, a qual se inicia com a definição do caminho para o arquivo da RNA treinada, sendo armazenado na variável *dadosRNA*, linha 1. Na linha 2, é executada a função *fann_create_from_file* que cria a RNA, a partir do arquivo salvo na fase de treinamento. Na linha 3, é definido o caminho para o arquivo de testes criado na Subseção 5.9.1. Na linha 4, é executada a função *fann_read_train_from_file*, destinada a carregar o arquivo de testes e utilizada para

carregar tanto o arquivo de treinamento, quanto o de testes. Na linha 5, são executados os comandos para criar o arquivo CSV, caso ele não exista, e abri-lo para escrita. O *loop* na linha 6 é executado, para indicar os 62 passos de predição, ou seja, de 0 à 61.

Na linha 7, a função *fann_test* é executada, para calcular o valor de saída da rede; ela recebe como parâmetros: a RNA, o vetor de entradas e o vetor de saídas e, como só existe uma entrada e uma saída no conjunto de testes, foi passada a posição zero do vetor. O retorno desta função é um vetor com os valores de saída calculados. O cálculo realizado na linha 8 retorna o valor calculado normalizado para os *bps* da entrada original, conforme Equação 6

$$x = \frac{x^{max}(y + 1) - x^{min}(y - 1)}{2} \quad (6)$$

na qual x é a saída dos *bps* da entrada original, y é o valor da saída normalizada e x^{max} e x^{min} são o maior e menor valores do conjunto de treinamento, respectivamente.

O valor de *saida* é salvo em um arquivo, no formato CSV. O *loop* realizado na linha 10 procede à mudança dos valores de posição, para que seja acrescentado o valor calculado no conjunto de entrada. Na linha 11, o valor do vetor na posição 1 é copiado para a posição 2, o valor da posição 2 é copiado para a posição 1 e, assim, por diante, até o da posição 30 ser copiado para a posição 29, finalizando o *loop*. Na linha 13, o valor calculado é inserido na posição 30 do vetor, para que uma nova predição ocorra. A Seção 5.6 deste capítulo, explica a realização dos passos de predição.

Finalizado o cálculo dos 62 valores de previsão, o *software* fecha o arquivo CSV, os dados de teste e da RNA, nas linhas 15, 16 e 17, respectivamente.

O resultado da execução deste algoritmo é um arquivo CSV, que pode ser utilizado para criar o gráfico dos 62 passos de predição do tráfego das mensagens GOOSE.

5.9.3 *Software* para o Cálculo do Erro Relativo Percentual

Esta parte do sistema é realizada, logo após a execução do *software* em *python*, paralelamente ao cálculo da predição. Este *software* foi implementado em linguagem C, e compilado incluindo a biblioteca FANN. O *software* cria a RNA, a partir do arquivo salvo na fase de treinamento explicada na Seção 5.5, carrega o arquivo de testes, criado na Subseção 5.9.1, calcula o erro relativo percentual e salva o resultado em um arquivo CSV. Apesar de selecionar 131 registros para criar o arquivo de teste, os 31 primeiros valores compõem a janela utilizada para o cálculo do valor 1; essa janela se desloca para frente, um a um para o cálculo dos demais, portanto, o resultado são 100 valores calculados. O Algoritmo 8 descreve a execução do *software* implementado.

A implementação foi realizada em linguagem C, a qual se inicia com a definição do caminho para o arquivo da RNA treinada, sendo armazenado na variável *dadosRNA*, linha

---

Algoritmo 8 Calcula o erro relativo percentual.

Entrada: Arquivo de teste para predição do tráfego

Saída: Arquivos CSV

```

1. dadosRNA = CaminhodoArquivodaRNATreinada;
2. rna = fann_create_from_file(dadosRNA);
3. dadosTeste = CaminhodoArquivodeTestesdaRNA;
4. dados = fann_read_train_from_file(dadosTeste);
5. Cria o arquivo CSV para salvar os dados do erro relativo percentual.
6. Para (i = 0; i < fann_length_train_data(dados); i++) Faça
7.     calculo = fann_test(rna, dados->input[i], dados->output[i]);
8.     saida = cálculo da saída calculo[0] normalizada, para os bps da entrada original.
9.     desejado = cálculo do valor de dados->output[i][0] normalizado, para os bps da entrada original.
10.    erro = fann_abs(saida - desejado)/desejado × 100;
11.    Salva os valores do erro no arquivo CSV.
12. Fim Para
13. Fecha o arquivo CSV
14. fann_destroy_train(dados);
15. fann_destroy(rna);

```

---

1. Na linha 2, é executada a função *fann_create_from_file* que cria a RNA, a partir do arquivo salvo na fase de treinamento. Na linha 3, é definido o caminho para o arquivo de testes criado na Subseção 5.9.1. Na linha 4, é executada a *fann_read_train_from_file*, função destinada a carregar o arquivo de testes, e utilizada para carregar tanto o arquivo de treinamento, quanto o de testes. Na linha 5, são executados os comandos para criar o arquivo CSV, caso ele não exista, e abri-lo para escrita. O *loop*, na linha 6, é executado para percorrer os 100 valores de teste. A função *fann_length_train_data* retorna o tamanho do conjunto de testes carregado com a função *fann_read_train_from_file*.

Na linha 7, a função *fann_test* é executada para calcular o valor de saída da rede; ela recebe como parâmetros: a RNA, o vetor de entradas e o de saídas e estes vetores serão percorridos, um a um, para os 100 valores de teste de acordo com a posição indicada pela variável *i*. O retorno desta função é um vetor com os valores de saída calculados. O cálculo realizado na linha 8, retorna o valor calculado normalizado, para os *bps* da entrada original, conforme a Equação 6. O cálculo realizado na linha 9 retorna o valor de saída do conjunto de testes, normalizado, para os *bps* da original, utilizando a mesma equação. Na linha 10, é calculado o erro relativo percentual, de acordo com a Equação 5.

Os valores do erro calculado para cada teste são salvos em uma arquivo no formato CSV. Finalizado o cálculo dos 100 erros, o *software* fecha o arquivo CSV, os dados de teste e da RNA, nas linhas 13, 14 e 15, respectivamente. O resultado da execução deste algoritmo é um arquivo CSV, que pode ser utilizado para criar o gráfico do erro relativo percentual dos 100 últimos valores do tráfego de mensagens GOOSE.

Para classificar o funcionamento do SE baseado nos valores do erro inseridos no arquivo

CSV, foi implementado o Algoritmo 9.

Algoritmo 9 Classifica o funcionamento do SE de acordo com os valores do erro relativo percentual.

Entrada: Arquivo CSV com o erro relativo percentual

Saída: Classificação do funcionamento do SE

1. $erro \leftarrow$ dados do arquivo CSV com os valores do erro relativo percentual.
  2. **Para** ($i = 0; i < 100; i++$) **Faça**
  3.   **Se** $erro[i] \leq 10$ **Então**
  4.   **Escreve** Situação normal de operação
  5.   **Se Não**
  6.   **Se** $erro[i] \neq erro[(i - 1)]$ **Então**
  7.   **Escreve** Situação de evento
  8.   **Se Não**
  9.   **Escreve** Situação de ataque DDOS
  10.   **Fim Se**
  11.   **Fim Se**
  12. **Fim Para**
  13. Fecha o arquivo CSV
-

O algoritmo inicia abrindo o arquivo CSV para leitura e carregando os valores do erro relativo percentual no vetor erro, linha 1. O *loop* na linha 2 é realizado para percorrer os 100 valores do erro, de 0 a 99. Na linha 3 é verificado se o valor do erro é menor que 10, caso seja, é escrito situação normal de operação, linha 4, caso não seja, na linha 6 é verificado se o erro atual é diferente do erro anterior, se for, escreve-se situação de evento, linha 7, se não for, escreve-se situação de ataque DDOS. Após passar por todos os valores do erro relativo percentual carregados, o arquivo CSV é fechado na linha 13.

O *software* descrito é executado logo após a execução dos demais *scripts*, sendo iniciado pelo *shell script* inserido na inicialização do *OpenMul*. Ele finaliza o processo de detecção de ataques DDOS no SE compondo a plataforma ZYBO. Com isso, obtém-se duas aplicações integradas ao controlador *OpenFlow* capaz de detectar ataques *man-in-the-middle* e DDOS.

Conclusão

As contribuições devidas pela adoção do protocolo IEC-61850 nas subestações de energia elétrica são inegáveis. A capacidade de automatização de processos, controle e supervisão remota e comunicação entre IEDs, para tomada de decisão distribuída, dão suporte para a expansão do setor. O fato de utilizar um sistema aberto de comunicação integrada oportuniza a interoperabilidade, mas expõe o SE aos ataques cibernéticos. As ameaças às redes de comunicação das subestações diferem, substancialmente, daquelas das redes corporativas. A especificidade do protocolo de comunicação torna necessário o desenvolvimento de aplicações específicas de segurança para este tipo de tráfego.

O *pentest* realizado nos IEDs do Laboratório de Sistemas de Energia Elétrica (LSEE) demonstraram que estes equipamentos não estão preparados para verificar se a mensagem é antiga ou foi adulterada. Os campos da mensagem GOOSE possuem um comportamento esperado que se apresentou relevante para a detecção de mensagens falsas. A evolução das redes de comunicação de dados alavancadas, pela necessidade de inovação em uma nova geração de arquiteturas de comunicação para *Smart Grids*, perpassa o uso do protocolo *OpenFlow* que oferece programação e flexibilidade para o controle e gerenciamento da rede. A configuração da rede para encaminhamento de mensagens *multicast*, bem como, o desenvolvimento do sistema de detecção e bloqueio de mensagens GOOSE adulteradas, são apenas duas das contribuições que se podem obter com a adoção deste protocolo nas redes de comunicação das subestações.

O teste de tempo realizado demonstrou que a implementação do protocolo no *switch* necessita de maturidade, uma vez que atrasos na entrega dos pacote inviabilizariam a utilização do IDS da forma como foi concebido. O teste em que se utilizou o *software switch OpenvSwitch* atestou que é viável a implementação do IDS com o processamento no controlador, apresentando latência, conforme as determinações do protocolo IEC-61850.

A implementação do protocolo IEC-61850, utilizando mensagens GOOSE, necessita de uma rede de comunicação de dados de alta velocidade. Para a implementação de um sistema que irá incluir tempo de processamento à comunicação realizada, é necessário que se utilizem ferramentas de processamento em tempo real. A plataforma de desenvolvimento

ZYBO, juntamente com o Sistema Operacional (SO) de tempo real configurado, habilitam o desenvolvimento de aplicações no controlador *OpenMul*. O sistema de detecção e bloqueio de mensagens GOOSE adulteradas apresentou-se como metodologia eficiente para a detecção e o bloqueio de mensagens GOOSE falsas na rede de comunicação de dados IEC-61850. Os testes realizados demonstraram que tal metodologia possui tempo de resposta e acuracidade para ser utilizado em grandes subestações.

A predição do tráfego embarcado na plataforma ZYBO é uma forma eficiente para análise de anomalias na rede de comunicação. A RNA NARX implementada permitiu 62 passos de previsão com erro relativo percentual de até 5%. Este sistema contribui para a detecção de tráfegos espúrios na rede de comunicação, uma vez que irá apontar variações fora do esperado, por uma comunicação através de mensagens GOOSE. O sistema contribui, também, para a detecção de eventos no sistema elétrico. A análise do erro demonstrou que as variações bruscas e o seu ponto de estabilidade são determinantes para diferenciar o tráfego de eventos no sistema elétrico do tráfego de ataques DDOS. Os erros em torno de 80%, sem estabilidade do erro relativo percentual, podem ser classificados como eventos no sistema elétrico. Erros acima de 10% estabilizados, podem ser considerados um ataque DDOS. As análises dos respectivos tráfegos, determinaram assinaturas do erro relativo percentual que se tornam um parâmetro eficiente para detecção destes ataques.

Os *softwares* desenvolvidos, integrados ao controlador *OpenFlow OpenMul*, demonstraram ser ferramentas robustas para comporem a rede de comunicação do Sistema Elétrico. Com a implantação destes, obtém-se um aumento na segurança e confiabilidade da rede da subestação através de sistemas que detectam ataques *man-in-the-middle* e DDOS.

Nesse sentido, acredita-se que os resultados obtidos na presente pesquisa encorajam sua continuidade e vislumbram um futuro próximo de utilização do protocolo *OpenFlow*, juntamente com as ferramentas de segurança para a rede de comunicação das subestações. Neste ínterim, podem-se elencar algumas sugestões de continuidade da pesquisa:

- ❑ desenvolver uma interface *Web* para apresentar os gráficos da previsão do tráfego da rede, do erro relativo percentual e que emita alertas identificando as situações de funcionamento do Sistema Elétrico baseado no erro relativo percentual;
- ❑ modificar o processo de cálculo da predição, do erro relativo percentual e da classificação do estado de funcionamento do SE para otimizar o tempo de execução, entregando os valores calculados diretamente na memória do controlador, sem acesso à arquivos;
- ❑ realizar um experimento fatorial completo, para analisar a influência dos parâmetros de configuração do *OpenFlow* no *switch* DM4100 DATACOM, nos atrasos observados na medição realizada na Subseção 4.8.2;

- ❑ realizar a conexão entre o controlador e o *switch*, utilizando criptografia *Secure Socket Layer* (SSL), e medir os tempos da aplicação para detecção de ataques *man-in-the-middle*;
- ❑ analisar uma possível perda de pacotes e criar condições para prever as mesmas evitando uma classificação incorreta do algoritmo;
- ❑ expandir o sistema de detecção de ataques *man-in-the-middle* para as mensagens SV;
- ❑ expandir o sistema de predição e análise do erro para as mensagens SV;
- ❑ implementar a classificação da RNA na *Field-Programmable Gate Array* (FPGA) da ZYBO;
- ❑ implementar a análise de segurança da mensagem GOOSE na FPGA;
- ❑ instalar e configurar soluções de gerenciamento de *log* para integrar a ferramenta de segurança desenvolvida, com sistemas supervisórios; e
- ❑ explorar o recurso *OpenFlow Extensible Match* (OXM) do *OpenFlow*, visando a migrar a lógica de detecção de ataques *man-in-the-middle* para o *switch OpenFlow*.

Referências

- ABDEL-GALIL, T.; EL-SAADANY, E.; SALAMA, M. Power quality event detection using adaline. **Electric Power Systems Research**, 2003. v. 64, n. 2, p. 137 – 144, 2003. ISSN 0378-7796. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0378779602001736>>.
- ALMAS, M. S.; VANFRETTI, L. Rt-hil implementation of the hybrid synchrophasor and goose-based passive islanding schemes. **IEEE Transactions on Power Delivery**, 2016. v. 31, n. 3, p. 1299–1309, June 2016. ISSN 0885-8977.
- ANDRADE, L. C. M. de; OLESKOVICZ, M.; SANTOS, A. Q.; COURRY, D. V.; FERNANDES, R. A. S. Very short-term load forecasting based on narx recurrent neural networks. In: **2014 IEEE PES General Meeting | Conference Exposition**. [S.l.: s.n.], 2014. p. 1–5. ISSN 1932-5517.
- ASTARLOA, A.; LÁZARO, J.; BIDARTE, U.; ZULOAGA, A.; IDIRIN, M. System-on-chip implementation of reliable ethernet networks nodes. In: **Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE**. [S.l.: s.n.], 2013. p. 2329–2334. ISSN 1553-572X.
- ASTARLOA, A.; MOREIRA, N.; BIDARTE, U.; URBINA, M.; MODRONO, D. Fpga based nodes for sub-microsecond synchronization of cyber-physical production systems on high availability ring networks. In: **2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig)**. [S.l.: s.n.], 2015. p. 1–6.
- BRAZIL, W. G.; ALBUQUERQUE, C. V. N. de. Protegendo redes ad hoc com certificados digitais: Uma proposta de arquitetura. **Brazilian Symposium on Information and Computer System Security (SBSeg)**, 2006. 2006. Disponível em: <<http://www.lbd-dcc.ufmg.br/colecoes/sbseg/2006/001.pdf>>.
- CAI, Z.; COX, A. L.; NG, T. S. E. Maestro: a system for scalable openflow control. In: . [S.l.: s.n.], 2010.
- CERT.BR; CGI.BR; NIC.BR. **Cartilha de Segurança para Internet**. 2th. ed. 2012. ISBN 978-85-60062-54-6.
- CHEBROLU, S.; ABRAHAM, A.; THOMAS, J. P. Feature deduction and ensemble design of intrusion detection systems. **Computers & Security**, 2005. v. 24, n. 4, p. 295 – 307, 2005. ISSN 0167-4048. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S016740480400238X>>.

- CHIU, H. W.; WANG, S. Y. Boosting the openflow control-plane message exchange performance of openvswitch. In: **2015 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2015. p. 5284–5289. ISSN 1550-3607.
- CLAVEL, F.; SAVARY, E.; ANGAYS, P.; VIEUX-MELCHIOR, A. Integration of a new standard: A network simulator of iec 61850 architectures for electrical substations. **IEEE Industry Applications Magazine**, 2015. v. 21, n. 1, p. 41–48, Jan 2015. ISSN 1077-2618.
- COATES, G.; HOPKINSON, K.; GRAHAM, S.; KURKOWSKI, S. Collaborative, trust-based security mechanisms for a regional utility intranet. In: **Power Energy Society General Meeting, 2009. PES '09. IEEE**. [S.l.: s.n.], 2009. p. 1–1. ISSN 1944-9925.
- COLET-SUBIRACHS, A.; RUIZ-ALVAREZ, A.; GOMIS-BELLMUNT, O.; ALVAREZ-CUEVAS-FIGUEROLA, F.; SUDRIA-ANDREU, A. Centralized and distributed active and reactive power control of a utility connected microgrid using iec61850. **Systems Journal, IEEE**, 2012. v. 6, n. 1, p. 58–67, 2012. ISSN 1932-8184.
- COMBS, G. **Wireshark**. 1998. <http://www.wireshark.org/>. Online. Acessado 30/08/2013.
- COMPANY, L. **Linaro**. 2016. <http://www.linaro.org/about/>. Online. Acessado 30/05/2016.
- COURY, D.; OLESKOVICZ, M.; GIOVANINI, R. **Proteção digital de sistemas elétricos de potência: dos relés eletromecânicos aos microprocessados inteligentes**. [S.l.]: SEL/EESC/USP, 2007. ISBN 9788585205782.
- COVINGTON, G. A.; GIBB, G.; LOCKWOOD, J. W.; MCKEOWN, N. A packet generator on the netfpga platform. In: **Proceedings of the 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines**. Washington, DC, USA: IEEE Computer Society, 2009. (FCCM '09), p. 235–238. ISBN 978-0-7695-3716-0. Disponível em: <<http://dx.doi.org/10.1109/FCCM.2009.29>>.
- CROCKETT, L. H.; ELLIOT, R. A.; ENDERWITZ, M. A.; STEWART, R. W. **The Zynq Book**. 2014. <http://www.zynqbook.com/>. Online. Acessado 30/05/2016.
- DATACOM. **Guia de Configuração Inicial OpenFlow**. 7. ed. <http://www.datacom.ind.br/>, 10 2014. Documento Confidencial, exclusivo para membros das universidades participantes dos testes ANSP Openflow com Datacom e que assinaram termo de compromisso NDA.
- _____. **Manual do Produto DM4100**. 5. ed. <http://www.datacom.ind.br/>, 7 2015. Linha DM4100.
- DIGILENT. **ZYBO Reference Manual**. 2014. http://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPZYBO/documentation/ZYBO_RM_B_V6.pdf. Online. Acessado 02/05/2016.
- _____. **Zybo Resource Center**. 2016. <https://reference.digilentinc.com/zybo:zybo>. Online. Acessado 23/06/2016.

DIXON, F.; YUNAS, M. T.; WEDELICH, V.; HOWARD, J.; BROWN, H. E.; SAUER, S. N.; XU, Y. M.; MARKELLO, T.; SHEIKH, W. Mitigating arc flashes using iec 61850: Examining a case at a chemical and refining facility. **IEEE Industry Applications Magazine**, 2014. v. 20, n. 1, p. 64–69, Jan 2014. ISSN 1077-2618.

DMITRENKO, I. Iec 61850 and measurements. **Elektronika Ir Elektrotechnika**, 2011. n. 6, p. 106–110, 2011. ISSN 1392-1215. Times Cited: 0. Disponível em: <<Go to ISI>:-//WOS:000292491000024>.

DUBUISSON, O. **ASN.1: Communication Between Heterogeneous Systems**. [S.l.]: Morgan Kaufmann, 2001. ISBN 9780126333619.

DZUNG, D.; NAEDELE, M.; HOFF, T. von; CREVATIN, M. Security for industrial communication systems. **Proceedings of the IEEE**, 2005. v. 93, n. 6, p. 1152–1177, 2005. ISSN 0018-9219.

ERICKSON, D. The beacon openflow controller. In: **Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking**. New York, NY, USA: ACM, 2013. (HotSDN '13), p. 13–18. ISBN 978-1-4503-2178-5. Disponível em: <<http://doi.acm.org/10.1145/2491185.2491189>>.

ERICSSON, G. Information security for electric power utilities (epus) - cigre developments on frameworks, risk assessment, and technology. **Power Delivery, IEEE Transactions on**, 2009. v. 24, n. 3, p. 1174–1181, 2009. ISSN 0885-8977.

_____. Cyber security and power system communication - essential parts of a smart grid infrastructure. **Power Delivery, IEEE Transactions on**, 2010. v. 25, n. 3, p. 1501–1507, 2010. ISSN 0885-8977.

FANG, X.; MISRA, S.; XUE, G.; YANG, D. Smart grid - the new and improved power grid: A survey. **Communications Surveys Tutorials, IEEE**, 2012. v. 14, n. 4, p. 944–980, Fourth 2012. ISSN 1553-877X.

FERNANDEZ, M. P. Comparing openflow controller paradigms scalability: Reactive and proactive. In: **Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on**. [S.l.: s.n.], 2013. p. 1009–1016. ISSN 1550-445X.

FLOODLIGHT, P. **Open Source Software for Building Software-Defined Networks**. 2016. <http://www.projectfloodlight.org/floodlight/>. Online. Acessado 07/06/2016.

FOROUZAN, B. **Comunicação de Dados e Redes de Computadores**. [S.l.]: McGraw Hill Brasil, 2008. ISBN 9788563308474.

FOUNDATION, O. N. **OpenFlow Switch Specification Version 1.0.0 (Wire Protocol 0x01)**. December 2009. <http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf>. Online. Acessado 01/06/2016.

_____. **OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04)**. June 2012. <http://archive.openflow.org/documents/openflow-spec-v1.3.0.pdf>. Online. Acessado 01/06/2016.

- GAUDÍ, M. **Maquete Didática Eletrosul**. 2009. <http://www.gaudimaquetes.com.br/>. Online. Acessado 06/04/2012.
- GIBB, G.; LOCKWOOD, J.; NAOUS, J.; HARTKE, P.; MCKEOWN, N. Netfpga an open platform for teaching how to build gigabit-rate network switches and routers. **Education, IEEE Transactions on**, 2008. v. 51, n. 3, p. 364–369, 2008. ISSN 0018-9359.
- GIUSTINA, D. D.; DEDÈ, A.; INVERNIZZI, G.; VALLE, D. P.; FRANZONI, F.; PEGOIANI, A.; CREMASCHINI, L. Smart grid automation based on iec 61850: An experimental characterization. **IEEE Transactions on Instrumentation and Measurement**, 2015. v. 64, n. 8, p. 2055–2063, Aug 2015. ISSN 0018-9456.
- GU, Y.; LIU, Y.; FENG, H. New method for intrusion detection based on bpnn and improved ga optimization. In: LIU, C.; CHANG, J.; YANG, A. (Ed.). **Information Computing and Applications**. Springer Berlin Heidelberg, 2011, (Communications in Computer and Information Science, v. 244). p. 434–440. ISBN 978-3-642-27451-0. Disponível em: <http://dx.doi.org/10.1007/978-3-642-27452-7_59>.
- GUO, Y.; TEN, C. W.; HU, S.; WEAVER, W. W. Modeling distributed denial of service attack in advanced metering infrastructure. In: **Innovative Smart Grid Technologies Conference (ISGT), 2015 IEEE Power Energy Society**. [S.l.: s.n.], 2015. p. 1–5.
- HAVILUDDIN; ALFRED, R. Performance of modeling time series using nonlinear autoregressive with exogenous input (narx) in the network traffic forecasting. In: **2015 International Conference on Science in Information Technology (ICSITech)**. [S.l.: s.n.], 2015. p. 164–168.
- HAYKIN, S. **Redes Neurais princípios e prática - 2ed**. [S.l.]: BOOKMAN COMPANHIA ED, 2001. ISBN 9788573077186.
- HERMAN, P.; COLOGNE, G. **tcpstat report network interface statistics**. 2016. <http://manpages.ubuntu.com/manpages/xenial/man1/tcpstat.1.html>. Online. Acessado 06/04/2016.
- HINES, W.; MONTGOMERY, D.; GOLDSMAN, D.; BORROR, C. **Probabilidade e estatística na engenharia**. 4th. ed. [S.l.]: Livros Técnicos e Científicos, 2006. ISBN 9788521614746.
- HOPKINSON, K.; ROBERTS, G.; WANG, X.; THORP, J. Quality-of-service considerations in utility communication networks. **Power Delivery, IEEE Transactions on**, 2009. v. 24, n. 3, p. 1465–1474, 2009. ISSN 0885-8977.
- HOYOS, J.; DEHUS, M.; BROWN, T. Exploiting the goose protocol: A practical attack on cyber-infrastructure. In: **Globecom Workshops (GC Wkshps), 2012 IEEE**. [S.l.: s.n.], 2012. p. 1508–1513.
- HURD, S.; SMITH, R.; LEISCHNER, G. Tutorial: Security in electric utility control systems. In: **Protective Relay Engineers, 2008 61st Annual Conference for**. [S.l.: s.n.], 2008. p. 304–309.
- HURST, W.; SHONE, N.; MONNET, Q. Predicting the effects of ddos attacks on a network of critical infrastructures. In: **Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing**

(CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. [S.l.: s.n.], 2015. p. 1697–1702.

IEC61850, I. E. C. Standard, **IEC-61850 Communication networks and systems in substations Part 1: Introduction and overview**. 2003. IEC 61850-1:2003(E).

_____. Standard, **IEC-61850 Communication networks and systems in substations Part 5: Communication requirements for functions and device models**. 2003. IEC 61850-5:2003(E).

_____. Standard, **IEC-61850 Communication networks and systems in substations Part 7-2: Basic communication structure for substation and feeder equipment - Abstract communication service interface (ACSI)**. 2003. IEC 61850-7-2:2003(E).

_____. Standard, **IEC-61850 Communication networks and systems in substations Part 8-1: Specific Communication Service Mapping (SCSM) - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3**. 2004. IEC 61850-8-1:2004(E).

_____. Standard, **IEC-61850 Communication networks and systems for power utility automation Part 8-1: Specific Communication Service Mapping (SCSM) - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3**. 2011. IEC 61850-8-1:2011-06.

_____. Standard, **IEC-61850 Communication networks and systems for power utility automation Part 1: Introduction and overview**. 2013. IEC/TR 61850-1:2013-3.

IEC62351, I. E. C. Standard, **IEC Technical Specification 62351-1 Power systems management and associated information exchange Data and communications security Part 1: Communication network and system security - Introduction to security issues**. 2007. IEC/TS 62351-1:2007(E).

_____. Standard, **IEC Technical Specification 62351-6 Power systems management and associated information exchange Data and communications security Part 6: Security for IEC 61850**. 2007. IEC/TS 62351-6:2007(E).

_____. Standard, **IEC Technical Specification 62351-7 Power systems management and associated information exchange Data and communications security Part 7: Network and system management (NSM) data object models**. 2010. IEC/TS 62351-7:2010(E).

IEEE, C. S. Standard, **IEEE Standard for Local and metropolitan area networks Media Access Control (MAC) Bridges**. 2004.

JIANG, W. Scalable ternary content addressable memory implementation using fpgas. In: **Architectures for Networking and Communications Systems (ANCS), 2013 ACM/IEEE Symposium on**. [S.l.: s.n.], 2013. p. 71–82.

KANABAR, M.; SIDHU, T. Performance of iec 61850-9-2 process bus and corrective measure for digital relaying. **Power Delivery, IEEE Transactions on**, 2011. v. 26, n. 2, p. 725–735, 2011. ISSN 0885-8977.

KANABAR, M.; SIDHU, T.; ZADEH, M. R. D. Laboratory investigation of iec 61850-9-2-based busbar and distance relaying with corrective measure for sampled value loss/delay. **Power Delivery, IEEE Transactions on**, 2011. v. 26, n. 4, p. 2587–2595, 2011. ISSN 0885-8977.

KOSTIC, T.; PREISS, O.; FREI, C. Understanding and using the iec 61850: a case for meta-modelling. **Comput. Stand. Interfaces**, 2005. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 27, n. 6, p. 679–695, jun. 2005. ISSN 0920-5489. Disponível em: <<http://dx.doi.org/10.1016/j.csi.2004.09.008>>.

KUSHNER, D. The real story of stuxnet. **Spectrum, IEEE**, 2013. v. 50, n. 3, p. 48–53, March 2013. ISSN 0018-9235.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. In: **Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks**. New York, NY, USA: ACM, 2010. (Hotnets-IX), p. 19:1–19:6. ISBN 978-1-4503-0409-2. Disponível em: <<http://doi.acm.org/10.1145/1868447.1868466>>.

LEVESQUE, M.; MAIER, M.; DESAI, Y.; JOOS, G. Adaptive admission control for a smart grid fiwi communications network facing power blackouts during a ddos attack. In: **Green Technologies Conference, 2012 IEEE**. [S.l.: s.n.], 2012. p. 1–3. ISSN 2166-546X.

LI, Q.; YAO, C. **Real-Time Concepts for Embedded Systems**. [S.l.]: Taylor & Francis, 2003. (CMP books). ISBN 9781578201242.

LIU, N.; ZHANG, J.; WU, X. Asset analysis of risk assessment for iec 61850-based power control systems - part i: Methodology. **Power Delivery, IEEE Transactions on**, 2011. v. 26, n. 2, p. 869–875, 2011. ISSN 0885-8977.

_____. Asset analysis of risk assessment for iec 61850-based power control systems - part ii: Application in substation. **Power Delivery, IEEE Transactions on**, 2011. v. 26, n. 2, p. 876–881, 2011. ISSN 0885-8977.

LIU, N.; ZHANG, J.; ZHANG, H.; LIU, W. Security assessment for communication networks of power control systems using attack graph and mcdm. **Power Delivery, IEEE Transactions on**, 2010. v. 25, n. 3, p. 1492–1500, 2010. ISSN 0885-8977.

LOPES, Y. **SMARTFlow SisteMa Autoconfigurável para Redes de Telecomunicações IEC 61850 com arcabouço OpenFlow**. Dissertação (Mestrado) — Universidade Federal Fluminense, 2013.

LOPES, Y.; FERNANDES, N. C.; BASTOS, C. A. M.; MUCHALUAT-SAADE, D. C. Smartflow: A solution for autonomic management and control of communication networks for smart grids. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2015. (SAC '15), p. 2212–2217. ISBN 978-1-4503-3196-8. Disponível em: <<http://doi.acm.org/10.1145/2695664.2695733>>.

LU, Z.; LU, X.; WANG, W.; WANG, C. Review and evaluation of security threats on the communication networks in the smart grid. In: **MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010**. [S.l.: s.n.], 2010. p. 1830–1835. ISSN 2155-7578.

MAGUIRE, T.; GIESBRECHT, J. Small Time-step ($< 2\mu\text{Sec}$) VSC Model for the Real Time Digital Simulator. 2005. p. 6, 2005.

MCCAULEY, M. **NOX and POX**. 2014. <http://www.noxrepo.org/>. Online. Acessado 30/05/2016.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. Openflow: Enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, 2008. ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1355734.1355746>.

MIRANDA, J. C. **IEC-61850: Interoperabilidade e Intercambialidade entre Equipamentos de Supervisão, Controle e Proteção Através das Redes de Comunicação de Dados**. Dissertação (Mestrado) — Escola de Engenharia de São Carlos da Universidade de São Paulo, 2009.

MOLINA, E.; JACOB, E.; MATIAS, J.; MOREIRA, N.; ASTARLOA, A. Using software defined networking to manage and control {IEC} 61850-based systems. **Computers & Electrical Engineering**, 2015. v. 43, p. 142 – 154, 2015. ISSN 0045-7906. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0045790614002626>.

MOLINA, E.; MATIAS, J.; ASTARLOA, A.; JACOB, E. Managing path diversity in layer 2 critical networks by using openflow. In: **Network and Service Management (CNSM), 2015 11th International Conference on**. [S.l.: s.n.], 2015. p. 394–397.

MOREIRA, N.; LÁZARO, J.; BIDARTE, U.; JIMENEZ, J.; ASTARLOA, A. On the utilization of system-on-chip platforms to achieve nanosecond synchronization accuracies in substation automation systems. **IEEE Transactions on Smart Grid**, 2016. PP, n. 99, p. 1–11, 2016. ISSN 1949-3053.

NEMETH, E.; SNYDER, G.; HEIN, T.; BOGGS, A.; CROSBY, M.; MCCLAIN, N. **Manual Completo do Linux**. 2th. ed. [S.l.]: Pearson Prentice Hall, 2007. ISBN 9788576051121.

NGO, H. D.; YANG, H. S. Latency and traffic reduction for process-level network in smart substation based on high-availability seamless redundancy. **IEEE Transactions on Industrial Electronics**, 2016. v. 63, n. 4, p. 2181–2189, April 2016. ISSN 0278-0046.

NISSSEN, S. **FANN Fast Artificial Neural Network Library**. 2016. <http://leenissen.dk/fann/wp/>. Online. Acessado 06/04/2016.

OPENVSWITCH. **OVS Open vSwitch**. 2014. <http://openvswitch.org/>. Online. Acessado 06/04/2016.

OZANSOY, C. R.; ZAYEGH, A.; KALAM, A. The real-time publisher/subscriber communication model for distributed substation systems. **Ieee Transactions on Power Delivery**, 2007. v. 22, n. 3, p. 1411–1423, 2007.

_____. The application-view model of the international standard iec 61850. **Ieee Transactions on Power Delivery**, 2009. v. 24, n. 3, p. 1132–1139, 2009.

_____. Object modeling of data and datasets in the international standard iec 61850. **Ieee Transactions on Power Delivery**, 2009. v. 24, n. 3, p. 1140–1147, 2009.

PALENSKY, P.; DIETRICH, D. Demand side management: Demand response, intelligent energy systems, and smart loads. **Industrial Informatics, IEEE Transactions on**, 2011. v. 7, n. 3, p. 381–388, 2011. ISSN 1551-3203.

PAN, H. Y.; WANG, S. Y. Optimizing the sdn control-plane performance of the openvswitch software switch. In: **2015 IEEE Symposium on Computers and Communication (ISCC)**. [S.l.: s.n.], 2015. p. 403–408.

PFEIFFENBERGER, T.; DU, J. L. Evaluation of software-defined networking for power systems. In: **Intelligent Energy and Power Systems (IEPS), 2014 IEEE International Conference on**. [S.l.: s.n.], 2014. p. 181–185.

PITT, D.; BAUER, R.; DAS, S.; KHARGHARIA, B.; BLAIR, C.; SASO, D.; HUNG, W.-C. **Open Networking Foundation**. 2016. <https://www.opennetworking.org/>. Online. Acessado 30/05/2016.

PREMARATNE, U.; LING, C.; SAMARABANDU, J.; SIDHU, T. Possibilistic decision trees for intrusion detection in iec61850 automated substations. In: **Industrial and Information Systems (ICIIS), 2009 International Conference on**. [S.l.: s.n.], 2009. p. 204–209.

PREMARATNE, U.; SAMARABANDU, J.; SIDHU, T.; BERESH, B.; TAN, J.-C. Evidence theory based decision fusion for masquerade detection in iec61850 automated substations. In: **Information and Automation for Sustainability, 2008. ICIAFS 2008. 4th International Conference on**. [S.l.: s.n.], 2008. p. 194–199.

PREMARATNE, U.; SAMARABANDU, J.; SIDHU, T.; BERESH, R.; TAN, J.-C. Security analysis and auditing of iec61850-based automated substations. **Power Delivery, IEEE Transactions on**, 2010. v. 25, n. 4, p. 2346–2355, 2010. ISSN 0885-8977.

PREMARATNE, U. K.; SAMARABANDU, J.; SIDHU, T. S.; BERESH, R.; TAN, J.-C. An intrusion detection system for iec61850 automated substations. **Ieee Transactions on Power Delivery**, 2010. v. 25, n. 4, p. 2376–2383, 2010.

RANSOM, D. L.; CHELMECKI, C. Using goose messages in a main-tie-main scheme. **IEEE Transactions on Industry Applications**, 2014. v. 50, n. 1, p. 17–24, Jan 2014. ISSN 0093-9994.

RIEDMILLER, M.; RPROP, I. **Rprop - Description and Implementation Details**. 1994.

RYU, S. F. C. **Component-based Software Defined Networking Framework. Build SDN Agilely**. 2014. <https://osrg.github.io/ryu/>. Online. Acessado 08/06/2016.

SAIKIA, D.; MALIK, N. **An Introduction to OpenMUL SDN Suite**. 2014. <http://www.openmul.org/uploads/1/3/2/6/13260234/openmul-sdn-platform.pdf>. Online. Acessado 08/06/2016.

SEVOV, L.; ZHAO, T.; VOLOH, I. The power of iec 61850: Bus-transfer and load-shedding applications. **IEEE Industry Applications Magazine**, 2013. v. 19, n. 1, p. 60–67, Jan 2013. ISSN 1077-2618.

- SIDHU, T.; YIN, Y. Modelling and simulation for performance evaluation of iec61850-based substation communication systems. **Power Delivery, IEEE Transactions on**, 2007. v. 22, n. 3, p. 1482–1489, 2007. ISSN 0885-8977.
- SILVA, I. da; SPATTI, D.; FLAUZINO, R. **Redes Neurais Artificiais para Engenharia e Ciências Aplicadas - Curso Prático**. [S.l.]: ARTLIBER, 2010. ISBN 9788588098534.
- SQLITE. **About SQLite**. 2014. <https://www.sqlite.org/about.html>. Online. Acessado 09/07/2016.
- STALLINGS, W. **Criptografia e segurança de redes: princípios e práticas**. [S.l.]: Pearson Prentice Hall, 2008. ISBN 9788576051190.
- STANFORD, J. U. **NetFPGA**. 2006. <http://www.netfpga.org/>. Online. Acessado 14/06/2013.
- SUČIĆ, S.; DRAGIČEVIĆ, T.; CAPUDER, T.; DELIMAR, M. Economic dispatch of virtual power plants in an event-driven service-oriented framework using standards-based communications. **Electric Power Systems Research**, 2011. v. 81, n. 12, p. 2108 – 2119, 2011. ISSN 0378-7796. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0378779611001957>>.
- SYDNEY, A.; NUTARO, J.; SCOGGIO, C.; GRUENBACHER, D.; SCHULZ, N. Simulative comparison of multiprotocol label switching and openflow network technologies for transmission operations. **IEEE Transactions on Smart Grid**, 2013. v. 4, n. 2, p. 763–770, June 2013. ISSN 1949-3053.
- TANENBAUM. **Computer networks**. 5th. ed. [S.l.]: Pearson Prentice Hall, 2011. ISBN 978-85-7605-924-0.
- TEN, C.-W.; HONG, J.; LIU, C.-C. Anomaly detection for cybersecurity of the substations. **Smart Grid, IEEE Transactions on**, 2011. v. 2, n. 4, p. 865–873, 2011. ISSN 1949-3053.
- TEN, C.-W.; LIU, C.-C.; MANIMARAN, G. Vulnerability assessment of cybersecurity for scada systems. **Power Systems, IEEE Transactions on**, 2008. v. 23, n. 4, p. 1836–1846, 2008. ISSN 0885-8950.
- TEN, C.-W.; MANIMARAN, G.; LIU, C.-C. Cybersecurity for critical infrastructures: Attack and defense modeling. **Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on**, 2010. v. 40, n. 4, p. 853–865, 2010. ISSN 1083-4427.
- THALER, P. **Usage of maxFrameSize in 802-3 and it's supplements**. Feb 1998. <http://www.ieee802.org/3/ac/public/maxframe.pdf>. Online. Acessado 09/07/2016.
- THOMAS, M.; ALI, I. Reliable, fast, and deterministic substation communication network architecture and its performance simulation. **Power Delivery, IEEE Transactions on**, 2010. v. 25, n. 4, p. 2364–2370, 2010. ISSN 0885-8977.
- TS'O, T.; HART, D.; KACUR, J. **Real-Time Linux**. 2016. https://rt.wiki.kernel.org/index.php/Main_Page. Online. Acessado 30/05/2016.

- TURNER, A. **Tcpreplay Pcap editing e replay tools for *NIX**. 2001. <http://tcpreplay.synfin.net/>. Online. Acessado 17/01/2013.
- URBAN, T. **Cacti - The Complete RRDTool-based Graphing Solution**. 2011. <http://www.cacti.net/>. Online. Acessado 01/08/2013.
- WALKIN, L. **Using the Open Source ASN.1 Compiler**. 2003. <http://lionet.info/asn1c/asn1c-usage.pdf>. Online. Acessado 30/05/2013.
- WANG, W.; LU, Z. Cyber security in the smart grid: Survey and challenges. **Computer Networks**, 2013. v. 57, n. 5, p. 1344 – 1371, 2013. ISSN 1389-1286. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1389128613000042>>.
- WEIDMAN, G. **Penetration Testing: A Hands-On Introduction to Hacking**. [S.l.]: No Starch Press, 2014. ISBN 9781593275648.
- WILLIAMS, A. **VMware Buys Nicira For \$1.26 Billion And Gives More Clues About Cloud Strategy**. 2012. <http://techcrunch.com/2012/07/23/vmware-buys-nicira-for-1-26-billion-and-gives-more-clues-about-cloud-strategy/>. Online. Acessado 08/06/2016.
- WU, S. X.; BANZHAF, W. The use of computational intelligence in intrusion detection systems: A review. **Applied Soft Computing**, 2010. v. 10, n. 1, p. 1 – 35, 2010. ISSN 1568-4946. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1568494609000908>>.
- WU, Y.; BIN, D.; ZHOU, X. Research on on-line interlocking of substation operation on wide area security defense. In: **Sustainable Power Generation and Supply, 2009. SUPERGEN '09. International Conference on**. [S.l.: s.n.], 2009. p. 1–4.
- XILINX. **Linux kernel release 4.4**. 2016. <https://github.com/Xilinx/linux-xlnx>. Online. Acessado 23/05/2016.
- XILINX, A. P. **Pre-built Zynq Linux Image/Release**. 2016. <http://www.wiki.xilinx.com/Linux>. Online. Acessado 23/05/2016.
- YAGHMOUR, K. **Building embedded Linux systems**. [S.l.]: O'Reilly, 2003. (O'Reilly Series). ISBN 9780596002220.
- YANG, H.-S.; KIM, S.-S.; JANG, H.-S. Optimized Security Algorithm for IEC 61850 based Power Utility System. **Journal of Electrical Engineering & Technology**, 2012. The Korean Institute of Electrical Engineers, 7, p. 443–450, 2012. Disponível em: <<http://www.dbpia.co.kr/Article/1922051>>.
- YOO, B.-K.; YANG, S.-H.; YANG, H.-S.; KIM, W.-Y.; JEONG, Y.-S.; HAN, B.-M.; JANG, K.-S. Communication Architecture of the IEC 61850-based Micro Grid System. **JOURNAL OF ELECTRICAL ENGINEERING & TECHNOLOGY**, 2011. 6, n. 5, p. 605–612, SEP 2011. ISSN 1975-0102.
- ZADEH, M.; SIDHU, T.; KLIMEK, A. Implementation and testing of directional comparison bus protection based on iec61850 process bus. **Power Delivery, IEEE Transactions on**, 2011. v. 26, n. 3, p. 1530 –1537, july 2011. ISSN 0885-8977.

ZILLGITH, M. **libIEC61850 open source library for IEC 61850**. 2016. <http://libiec61850.com/libiec61850>. Online. Acessado 06/04/2016.

ZULYADI; SEMBIRING, J. Design of ftts forecasting model using markov chain and p2amf framework case study: Farmer's terms of trade of smallholders estate crops sub-sector in riau. In: **2015 International Conference on Information Technology Systems and Innovation (ICITSI)**. [S.l.: s.n.], 2015. p. 1–6.