UNIVERSIDADE FEDERAL FLUMINENSE

YONA LOPES

SMARTFlow
SisteMa Autoconfigurável para Redes de Telecomunicações IEC 61850 com arcabouço OpenFlow

NITERÓI

2013

UNIVERSIDADE FEDERAL FLUMINENSE

YONA LOPES

SMARTFlow
SisteMa Autoconfigurável para Redes de Telecomunicações IEC 61850 com arcabouço OpenFlow

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Telecomunicações da Universidade Federal Fluminense como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Telecomunicações. Área de concentração: Comunicação de dados multimídia

Orientador:
CARLOS ALBERTO MALCHER BASTOS

Co-orientador:
NATALIA CASTRO FERNANDES

NITERÓI

2013

YONA LOPES

SMARTFlow - SisteMa Autoconfigurável para Redes de Telecomunicações IEC 61850 com arcabouço OpenFlow

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Telecomunicações da Universidade Federal Fluminense como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Telecomunicações. Área de concentração: Comunicação de dados multimídia

Aprovada em 19 dezembro de 2013.

BANCA EXAMINADORA

Prof^{o}. CARLOS ALBERTO MALCHER BASTOS, D.Sc.
Universidade Federal Fluminense (UFF) - Orientador

Prof^{a}. NATALIA CASTRO FERNANDES, D.Sc.
Universidade Federal Fluminense (UFF) - Coorientadora

Prof^{o}. JOBERTO SÉRGIO BARBOSA MARTINS, D.Sc.
Universidade Salvador(UNIFACS)

Prof^{o}. JOSÉ AUGUSTO SURUAGY MONTEIRO, D.Sc.
Universidade Federal de Pernambuco (UFPE)

Prof^{o}. RICARDO CAMPANHA CARRANO, D.Sc.
Universidade Federal Fluminense (UFF)

Niterói
2013

À minha família.

Agradecimentos

Agradeço, inicialmente, à minha família, especialmente aos meu pais, Yara e Cesar, pelo apoio em cada decisão que tomei na vida. A pessoa que sou hoje, agradeço a vocês.

Agradeço também ao meu marido Felipe, pela paciência e pelo apoio, estando ao meu lado em todas as horas que precisei.

Agradeço aos meus orientadores, Professor Malcher e Professora Natalia, pela amizade, ensinamentos, conselhos, exemplos, incentivo e orientação. Agradeço ao Professor Malcher pela oportunidade de realizar um sonho, tornando possível a minha saída do mercado e a conclusão desse trabalho. Agradeço a Professora Natalia, por toda a disponibilidade, atenção e ajuda sempre. Agradeço também aos Professores Joberto, Suruagy e Carrano pela disponibilidade, apoio e atenção.

Aos amigos que entenderam minha ausência, e sempre me incentivaram e apoiaram, fico grata. Agradeço, em especial, aos amigos do laboratório Gteccom, pelo acolhimento, apoio e carinho durante todo o tempo.

Por fim, um obrigado a todos que, embora não estejam citados aqui, me incentivaram, contribuindo de forma direta ou indireta, para o desenvolvimento deste trabalho.

Resumo

As redes elétricas inteligentes, também chamadas de *Smart Grids*, dependem de uma sólida base de comunicação. Apesar de normas como a IEC 61850 apresentarem soluções para alcançar a interoperabilidade, ainda existem problemas em aberto com relação à autoconfiguração e ao controle automatizado da rede de comunicação. Essa dissertação propõe uma solução eficiente de controle e gerenciamento autônomo de redes de comunicação para subestações baseadas na norma IEC 61850. A solução proposta, chamada de SMARTFlow, é baseada no uso de redes definidas por software para obter granularidade e flexibilidade no tratamento dos fluxos de dados. Com base nos arquivos de configuração do sistema elétrico de potência gerados de acordo com a norma IEC 61850, o SMARTFlow pró-ativamente define, calcula e configura os grupos *multicast* de camada 2, necessários para o encaminhamento de mensagens GOOSE, que são utilizadas para proteção da rede elétrica e, por essa razão, possuem rígidos requisitos de atraso. Além disso, o SMARTFlow define um novo modelo para as priorizações de tráfego, diferenciando os vários tipos de mensagens definidas na norma de acordo com os seus limiares para atraso, garantindo os requisitos mínimos de tempo na entrega das mensagens dentro das subestações. O SMARTFlow monitora a rede, definindo sob demanda a configuração de fluxos cliente-servidor, além de reconfigurar pró-ativamente todas as entradas de fluxo em caso de falhas na rede. A proposta foi implementada e testada utilizando o emulador Mininet e um gerador de tráfego desenvolvido no SCAPY, e se mostrou mais eficiente que o gerenciamento padrão das redes de subestação. O SMARTFlow reduziu em até 20 vezes o atraso em redes controladas pelo OpenFlow em sua forma tradicional e em até 44% a carga total da rede quando comparada a *switches* típicos. A proposta também demonstrou vantagens quando comparada com outras soluções de *multicast*, como o GMRP (*GARP Multicast Registration Protocol*).

Palavras-chave: IEC 61850, OpenFlow, *Software Defined Network* (SDN), *Multicast* na camada de enlace, Métodos de Recuperação de Falha, 802.1, SMARTFlow, autoconfiguração.

Abstract

Smart grids depend on a solid foundation on communications. Although standards like IEC 61850 propose solutions to achieve interoperability, auto configuration and automatic control of the communication network are still an open problem. This dissertation proposes an efficient solution for autonomic management and control of communication networks for substations based on IEC 61850. The proposed solution, called SMARTFlow, uses Software Defined Network in order to achieve granularity and flexibility in the handling of data flows. Based on electric power system configuration files, according to IEC 61850, SMARTFlow proactively define, calculate and set Layer 2 multicast groups, which are necessary to forward GOOSE messages and are used to protect electric network and, therefore have strong delay requirements. Moreover, SMARTFlow defines a a new model for traffic prioritization, distinguishing several types of messages defined in accordance to standard delay thresholds and ensuring the minimum delay for messages delivery within substations. SMARTFlow monitors the network, defines on-demand configuration of client-server flows, as well as proactively reconfigures all entries flows in case of network failures. The proposal was implemented and tested using Mininet and a traffic generator developed with SCAPY, and was more efficient than the standard network management substation. Solution SMARTFlow decreased up to 20 times the delay in networks controlled by traditional OpenFlow and up to 44 % the overall network load compared with typical switches. The proposal also have shown advantages when compared with other multicast solutions, such as GMRP (*GARP Multicast Registration Protocol*).

Palavras-chave: IEC 61850, OpenFlow, *Software Defined Network* (SDN), Layer 2 *Multicast*, 802.1, SMARTFlow, auto configuration.

Lista de Figuras

2.1 Exemplo de uma referência de nome de objetos do IEC 61850 e sua estrutura hierárquica. 15

2.2 Exemplo simples de modelagem de um IED [36]. 16

2.3 Exemplo de distribuição de Funções no *Intelligent Electronic Device* (IED). 17

2.4 Níveis de uma subestação e interfaces de comunicação entre estes níveis de acordo com a arquitetura proposta na IEC 61850 [37]. 18

2.5 Princípios de comunicação TPAA e MCAA da norma IEC 61850 [39]. . . . 19

2.6 Comunicação dentro de uma subestação via *Manufacturing Message Specification* (MMS), através da troca de mensagens entre o cliente e o servidor. 21

2.7 Estrutura do Pacote Ethernet (64 – 1518 bytes) 22

2.8 Tempos de Transmissão para eventos. Mensagens GOOSE [13] 23

2.9 Comunicação dentro de uma subestação via mensagens *Generic Object Oriented Substation Event* (GOOSE), com um IED notificando aos demais sobre algum alarme específico 24

2.10 Estrutura da *tag* [13, 40]. 25

2.11 Estrutura do Quadro Ethernet 26

2.12 Arquitetura de composição dos arquivos da linguagem *Substation Configuration Language* (SCL) [41]. 28

4.1 Comparação entre o modelo tradicional, onde o plano de controle reside no próprio equipamento, e o modelo programável do OpenFlow [56] 40

4.2 *Switch* OpenFlow. A Tabela de fluxos é controlada por um controlador remoto via o canal seguro [14] 42

4.3 Tabela de Fluxos. Uma entrada de fluxo, no OpenFlow 1.0.0 consiste em regras, contadores e ações [63] 44

4.4 Exemplo de entrada da tabela de fluxos do OpenFlow versão 1.0.0 [56, 63] 44

4.5 Fluxograma detalhando processamento dos fluxos através de um *switch* OpenFlow versão 1.0.0 [64] 45

4.6 Componentes do *switch* OpenFlow versão 1.1.0 [64] 46

4.7 Campos do cabeçalho no OpenFlow 1.1.0 [64] 47

4.8 Uma entrada de fluxo, no OpenFlow 1.1.0 consiste em regras, estatísticas e instruções [64] 47

4.9 Fluxos de pacote através do *switch* OpenFlow [64] 48

4.10 Fluxograma detalhando processamento dos fluxos através de um *switch* OpenFlow versão 1.1.0 [64] 49

4.11 Tabela de Grupos. Uma entrada de grupo, no OpenFlow 1.1.0 consiste nos campos identificador de grupo, tipo de grupo, contadores, e *action buckets* [64] 49

4.12 Switch OpenFlow Dedicado [56] 51

4.13 Switch OpenFlow Híbrido [56] 52

4.14 Comparação de desempenho entre o POX e NOX, com suas duas interfaces (C++ e Python) [71] 54

5.1 Estrutura do SMARTFlow, o qual é composto por um conjunto de aplicações de controle para a rede de comunicação das subestações baseadas em IEC 61850. 61

5.2 Mecanismo para recuperação de falha no SMARTFlow implementado em um *switch* OpenFlow 1.1 ou superior. 67

6.1 Topologias testadas 73

6.2 Ambiente de implementação do SMARTFlow 75

6.3 Fluxograma do *script mnteste* 77

6.4 Estrutura do Pacote GOOSE 78

6.5 Tela de captura de pacotes GOOSE no Wireshark 81

6.6 Atraso das mensagens GOOSE na rede com os componentes do SMARTFlow, em função do aumento do número de IEDs. 84

6.7 Atraso das mensagens GOOSE na rede com os componentes do SMART-Flow, em função do aumento do número de *switches*. 85

6.8 Tempo para descobrir a topologia (T1) 86

6.9 Tempo de configuração dos grupos *multicast*, onde T2 é o tempo para cálculo da árvore *multicast* e T3 o tempo para configurar os fluxos 87

6.10 Comparação do atraso das mensagens GOOSE na rede, ao se utilizar o SMARTFlow e o OpenFlow Nativo, assumindo IEDs uniformemente distribuídos entre os *switches*. 88

6.11 Carga de controle na rede após estabilização. 89

6.12 Carga total de dados na rede em função do aumento do número de IEDs. 90

Lista de Tabelas

2.1 Tipos de mensagens suportadas pelo padrão IEC 61850 [37]. 20

2.2 Faixa de endereços *multicast* recomendados [13, 40]. 24

2.3 Valores *default* atribuídos para Ethertype, APPID, VLAN IDs, e Prioridades [13]. 26

2.4 Exemplos de protocolo de redundância [19]. 30

5.1 Proposta para diferenciação do Tráfego em Rede IEC 61850. 68

6.1 Cenários de teste. 74

6.2 Parâmetros usados em cada experimento. 83

6.3 Comparação de métodos para recuperação de falha 92

6.4 Comparação entre as atuais soluções *multicast* e a proposta SMARFlow . 93

Lista de Abreviaturas e Siglas

API *Application Programming Interface* 41

BRP *Beacon Redundancy Protocol* 30

CID *Configured IED Description* 27

CFI *Canonical Format Indicator* 25

CRP *Cross-network Redundancy Protocol* 30

DA *Data Attributes* 14

DO *Data Objects* 14

DRP *Distributed Redundancy Protocol* 30

GOOSE *Generic Object Oriented Substation Event* 19

GMRP *GARP Multicast Registration Protocol* 3

HSR *High-availability Seamless Redundancy* 30

ICD *IED Capability Description* 27

IEC *International Electrotechnical Commission* 2

IED *Intelligent Electronic Device* 8

IGMP *Internet Group Management Protocol* 36

LAN *Local Area Network* 11

LD *Logical Device* 14

LN *Logical Node* 14

LLDP *Link Layer Discovery Protocol* 62

MAC *Media Access Control* 25

MCAA *Multicast Application Association* 19

MMS *Manufacturing Message Specification* 19

MPLS *Multiprotocol Label Switching* 46

MRP *Media Redundancy Protocol* 30

NAT *Network Address Translation* 43

PCP *Priority Code Point* 25

PRP *Parallel Redundancy Protocol* 30

ONF *Open Networking Foundation* 41

QoS *Quality of Service* 41

RSTP *Rapid Spanning Tree Protocol* 34

RTU *Remote Terminal Unit* 21

SAS Sistema de Automação de Subestações 2

SCADA *Supervisory Control and Data Acquisition* 12

SCD *Substation Configuration Description* 27

SCL *Substation Configuration Language* 13

SCTP *Stream Control Transmission Protocol* 47

SDN *Software Defined Network* 39

SEP Sistema Elétrico de Potência 1

SPF *Shortest Path First* 65

SSD *System Specification Description* 27

SSL *Secure Socket Layer* 42

SV *Sampled Values* 19

TC Transformador de Corrente 11

TCP *Transmission Control Protocol* 42

TCAM *Ternary Content Addressable Memory* 43

TCI *Tag Control Information* 25

TLV *Type Length Value* 51

TP Transformador de Potencial 11

TPAA *Two Party Application Association* 19

TPID *Tag Protocol Identifier* 25

VID *VLAN Identifier* 25

VLAN *Virtual Local Area Network* 3

XML *eXtensible Markup Language* 26

Sumário

1 Introdução **1**

1.1 Motivação e Objetivos 2

1.1.1 Objetivos Gerais 3

1.1.2 Objetivos Específicos 4

1.2 Principais contribuições 5

1.3 Organização do trabalho 6

2 Rede Elétrica, Smart Grids e a Norma IEC 61850[1] **7**

2.1 Uma breve introdução à Rede Elétrica e à Proteção 9

2.2 A Norma IEC 61850 12

2.2.1 Modelagem dos dispositivos 13

2.2.2 Modelagem da Comunicação 17

2.2.2.1 Mensagem *Manufacturing Message Specification* (MMS) . 20

2.2.2.2 Mensagens *Generic Object Oriented Substation Event* (GOOSE) e *Sampled Values* (SV) 21

2.2.3 Linguagem de configuração de Subestações (*Substation Configuration Language* - SCL) 26

2.3 Protocolos de redundância 29

3 O uso dos recursos de redes no contexto IEC 61850 **32**

3.1 Diferenciação do tráfego 32

3.2 Métodos de Recuperação de Falha 33

3.3 Comunicação multicast na camada de enlace 35

4 O Arcabouço OpenFlow **39**

4.1 O *Switch* OpenFlow 43

4.1.1 Tabela de Fluxos OpenFlow 43

4.1.2 Versão 1.1.0 46

4.1.3 Versão 1.2 50

4.1.4 Tipos de *switches* OpenFlow 51

4.2 O Controlador OpenFlow 52

4.2.1 Controlador de Referência OpenFlow 53

4.2.2 NOX e POX 53

4.2.3 Outros Controladores 54

4.3 Vantagens e Benefícios do OpenFlow 55

5 A proposta SMARTFlow **58**

5.1 O OpenFlow e o IEC 61850 59

5.2 Arquitetura 60

5.2.1 Aplicações de controle do SMARTFlow 61

5.3 Algoritmos de controle do SMARTFlow e configuração automática a partir do arquivo *Substation Configuration Description* (SCD) 64

5.4 Diferenciação do Tráfego 67

5.5 Vantagens e Desvantagens 69

6 Análise do SMARTFlow **71**

6.1 Cenários e parâmetros de teste 72

6.2 Ambiente de Implementação 74

6.2.1 Gerador de tráfego GOOSE 78

6.3 Experimentos realizados e resultados 82

6.3.1 Validação da Norma IEC 61850 com o modelo proposto 83

6.3.2 Desempenho do modelo proposto 85

6.3.3 Comparação do SMARTFlow com o OpenFlow Nativo 88
6.4 Análise qualitativa do SMARTFlow 91
6.4.1 Métodos de recuperação de Falha e o SMARTFlow 91
6.4.2 As soluções *multicast* camada 2 e o SMARTFLow 93

7 Conclusões **95**
7.1 Trabalhos Futuros 97

Referências **98**

Apêndice A – Exemplo da Saída do Gerador GOOSE **104**

Capítulo 1

Introdução

A energia elétrica é essencial para o desenvolvimento da sociedade. Com os avanços tecnológicos equipamentos como computador, televisão, aparelhos de som, condicionadores de ar, aquecedores e diversos outros estão cada vez mais presentes na casa dos cidadãos aumentando a demanda por energia.

O provimento da energia elétrica implica diretamente na qualidade de vida do usuário. Anos atrás, ficar sem energia durante um período de tempo não causava tanto impacto quanto causa nos tempos modernos. Atualmente, uma cidade sem energia é uma cidade imóvel, pois as luzes, os telefones, as máquinas, os computadores, tudo se apaga. A falta de energia, mesmo que por um período curto de tempo, acarreta em prejuízos enormes além de danos causados pela falta de serviços essenciais. No Brasil, uma auditoria do Tribunal de Contas da União mostrou que os apagões de 2001 e 2002 causaram um prejuízo de R$ 45,2 bilhões para os brasileiros [1]. Estatísticas semelhantes também são encontradas em outras regiões do mundo, como os Estados Unidos. Estima-se que o apagão de 2011 causou, apenas na região de San Diego, um prejuízo de $97 a $118 milhões de dólares, incluído gastos com comidas estragadas ($12 a $18 milhões), horas extras governamentais ($10 a $20 milhões) e perda de produtividade ($70 milhões) [2]. Estudos da Universidade de Berkeley apontam para uma estimativa de $80 bilhões de perdas anuais nos EUA devido às interrupções no setor elétrico [3].

O responsável por prover a qualidade do serviço elétrico oferecido aos consumidores, controlando e direcionando o fluxo de energia e fornecendo energia durante o maior tempo possível e com qualidade aos consumidores finais é o Sistema Elétrico de Potência (SEP), que possui uma infraestrutura complexa para gerar, transmitir e distribuir essa energia. Contudo, o SEP está sujeito a anormalidades e defeitos em sua operação. Para que falhas não gerem os apagões, e não prejudiquem o consumidor, sistemas de proteção, moni-

toração, controle e supervisão são indispensáveis. A forma com que os dispositivos do SEP interagem afeta diretamente esses sistemas. Para isso foram desenvolvidos diversos protocolos de comunicação, muitas vezes proprietários, aumentando a quantidade de protocolos no mercado e trazendo incompatibilidade entre dispositivos que implementem protocolos distintos. Deste problema, surgiu o incentivo para a construção de um padrão único que especifique os parâmetros e a forma como os dispositivos devem se comunicar, padronizando a comunicação no Sistema de Automação de Subestações (SAS). Assim, em 1995, foram estabelecidos três grupos de trabalho do Comitê Técnico TC57 (*Technical Committee 57*) da *International Electrotechnical Commission* (IEC), chamados de Grupos de Trabalho 10, 11 e 12 [4] que, mais tarde, deram origem à norma IEC 61850 [5]. A norma propõe uma solução unificada de comunicação para garantir a interoperabilidade entre dispositivos de diferentes fabricantes que permitem, dentre outros, a supervisão, o controle e a proteção em tempo real do SEP. Com isso pretende-se modernizar o sistema trazendo mais qualidade de serviço, confiabilidade e integração na rede elétrica.

Além disso, a norma IEC 61850 suporta uma comunicação automatizada e pavimenta o caminho para as *Smart Grids* ao fazer integrações entre sistemas, como monitoramento, proteção, medição e controle. As *Smart Grids* surgiram dessa necessidade inevitável de modernização dos sistemas de automação do SEP, tornando imprescindível a implantação de um sistema de comunicação "inteligente" com os centros de controle, supervisão e de medição das distribuidoras de energia para prover uma gestão eficiente da rede elétrica [6]. Um sistema elétrico inteligente comuta toda a oferta de energia através da rede de distribuição, gerenciando a demanda de energia através de um sistema de comunicação. Portanto, a inteligência da rede reside na capacidade dos dispositivos de se comunicar, trocando informações que permitem construir uma rede mais segura e mais eficiente.

Apesar da norma IEC 61850 apresentar soluções para a comunicação nesse sistema com uma modelagem embasada na interoperabilidade dos dispositivos, ainda existem problemas em aberto como será visto a seguir.

1.1 Motivação e Objetivos

Métodos não aplicados corretamente causam perda de desempenho na rede, além de dificultar expansões nas redes elétricas e de telecomunicações. Dentro das subestações, mesmo com a implantação da norma IEC 61850, ainda existem desafios e limitações.

As mensagem de camada de enlace padronizadas pela norma para serem enviadas

em um modelo *multicast*, devido ao comportamento típico de *switches*, são enviadas por todas as portas como *broadcast*. Esse comportamento traz uma sobrecarga tanto para a rede quanto para os dispositivos finais que recebem mensagens mesmo não estando no grupo [7]. Existem soluções, como o uso de filtros *multicast* como o *GARP Multicast Registration Protocol* (GMRP) na rede, que segundo [8, 7, 9] diminuem os atrasos em redes de subestação e reduzem o volume de dados recebidos por dispositivos [10]. Devido à restrição dos dispositivos, pois não implementam o protocolo, a configuração se torna menos dinâmica[1], o que faz com que na prática, algumas soluções optem pelo uso de redes locais virtuais (*Virtual Local Area Network*s (VLANs)) para segmentar a rede e minimizar o problema. Porém, como mostrado em [11, 9], os projetos que usam *multicast* ao invés de *broadcast* em VLANs evitam a sobrecarga e aumentam o desempenho na rede.

Além disso, a configuração das redes IEC 61850 atuais é feita de forma manual. Cada *switch* tem que ser configurado individualmente mesmo quando é utilizado um protocolo dinâmico. Isso acontece quando os dispositivos finais não implementam o protocolo, ou porque a solução é estática, o que demanda que cada dispositivo seja configurado manualmente com o endereço do grupo *multicast*. Com isso, não só o planejamento da rede como a configuração são trabalhosos e pouco flexíveis.

Outra questão importante está relacionada com o fato de que os sistemas de controle e monitoração dos *switches*, quando existentes, diferem por fabricantes e não são interoperáveis. Além disso, os sistemas de recuperação de falhas usados em redes de subestação ainda geram excesso de tráfego na rede ou têm um tempo de recuperação alto, além de serem restritos a topologias específicas.

Com relação à prioridade dos pacotes na rede, embora os atrasos máximos sejam bem definidos na norma, variando de 3ms à 1000ms, a prioridade atribuída a cada classe de mensagens tem pouca granularidade. De fato, são utilizados apenas dois níveis de prioridade na rede, dentre os oito possíveis níveis de prioridade [12, 13].

1.1.1 Objetivos Gerais

Essa dissertação visa melhorar o desempenho na comunicação dentro de subestações utilizando de forma eficiente os recursos de rede, controlar eficientemente as redes de dados de subestações elétricas, além de dinamizar e simplificar o planejamento e a configuração de

[1]De acordo com informações providas pelos fabricantes internacionais e nacionais de *switches* IEC 61850, devido ao fato dos IEDs não implementarem o protocolo, a configuração do GMRP é feita manualmente na tabela *multicast* estática nos *switches* conectados diretamente aos IED, com isso a propagação das mensagens GMRP se dá a partir desses *switches*

redes de subestação IEC 61850 tornado a rede autoconfigurável. Seus objetivos principais são o desenvolvimento de um encaminhamento apropriado de mensagens IEC 61850 e o controle eficiente de redes de dados de subestações elétricas.

Para alcançar esses objetivos, essa dissertação descreve o SisteMa Autoconfigurável para Redes de Telecomunicações IEC 61850 com arcabouço OpenFlow (SMARTFlow), que propõe o uso das redes definidas por software para fazer um melhor controle dos dados em tráfego dentro de uma subestação. O OpenFlow [14] é uma solução de redes definidas por software que possibilita um controle mais flexível e orientado às necessidades de cada sistema de comunicação específico, como é o caso das *smart grids* e da Norma IEC 61850. Com isso, torna-se possível experimentar novos métodos e novos algoritmos que garantam uma boa solução e aumentem o desempenho das redes em questão.

O SMARTFlow permite o controle do fluxo de cada tráfego da rede, escolhendo as rotas que os pacotes deverão seguir e o processamento que receberão independente de camadas. Com o arcabouço OpenFlow, que provê uma interface para o desenvolvimento de aplicações de controle de rede de telecomunicações, foi possível desenvolver um componente que calcula e configura as árvores *multicast* para encaminhamento das mensagens multicast de camada dois, de forma adequada e transparente para os dispositivos finais. Foi desenvolvido também, um componente para tratamento de falhas, que recalcula as árvores dinamicamente sempre que houver falhas na rede. Além disso, o SMARTFlow define um modelo para aplicação de prioridade nos diferentes tipos de mensagens na rede, que pode ser usado tanto em redes tradicionais quanto em redes definidas por *software*.

1.1.2 Objetivos Específicos

Os objetivos específicos do SMARTFlow são:

- criar pró-ativamente e dinamicamente árvores *multicast* de camada de enlace para o envio das mensagem IEC 61850 de restrição temporal rígida;
- propiciar uma configuração e um gerenciamento mais eficientes;
- permitir a autoconfiguração inicial da rede de telecomunicações com base nas informações obtidas com a modelagem proposta pelo IEC 61850;
- permitir uma recuperação de falhas mais inteligente que não gere tráfego extra na rede e atue em um tempo aceitável;

- garantir uma maior granularidade na definição das prioridades, aumentando a qualidade do serviço oferecido.

- ser uma solução transparente aos dispositivos finais.

1.2 Principais contribuições

Além de estudar alguns dos problemas de configuração e controle de redes de subestação, nessa dissertação, é proposto e modelado o SMARTFlow, que permite solucionar os problemas observados de forma eficiente e automática.

Um aspecto interessante dessa dissertação é que o SMARTFlow foi implementado utilizando o controlador de redes OpenFlow POX 0.1.0, na versão 1.0.0 do OpenFlow e pode ser utilizado em redes reais. A validação e os testes de desempenho da implementação da proposta foram feitos na ferramenta de emulação de redes Mininet [15] versão 2.

Os testes foram realizados com o auxílio de *scripts* escritos em Python, de um gerador de tráfego criado no Scapy [16], e de ferramentas como o tcpdump [17]. Foram desenvolvidos diversos cenários a fim de validar o uso do SMARTFlow, avaliando diferentes aspectos de desempenho como atraso na rede, carga total, carga de controle, tempo de cálculo da árvore, dentre outros. Os resultados se mostraram muito positivos quando comparados às soluções já existentes. O SMARTFLow manteve o atraso abaixo do limiar mais rígido de tempo, 3 ms, definido pela norma IEC 61850, se mostrado adequado para aplicação em subestações. Além disso, teve o atraso da rede diminuído em 20 vezes quando comparado aos componentes nativos do controlador OpenFlow, que implementam o comportamento típico de *switches* padrão.

As contribuições dessa dissertação estão principalmente relacionadas à:

- identificação do OpenFlow como arcabouço para resolver esses problemas, já que além de ser flexível possui uma visão global da rede como será detalhado nessa dissertação.

- Construção e implementação de:

  - um componente para criação transparente e automática de árvores *multicast* de camada de enlace. Até o momento, todas as soluções encontradas, como o GMRP, precisam fazer alterações nos IEDs de forma que esses elementos

implementem o protocolo. A proposta SMARTFlow pode ser implementada sem a necessidade de modificação dos IEDs tradicionais, e sem sobrecarregá-los.

  - um componente para tratamento de falhas sem duplicação de tráfego ou de elementos de rede, como acontece com os métodos tradicionais.
  - um componente para auto configuração da rede de telecomunicações com base nos dados obtidos nos arquivos de configuração da subestação, providos pela norma IEC 61850.

- definição de um modelo de priorização para os diferentes tipos de mensagem da rede, aumentando a qualidade do serviço oferecido.
- identificação de vantagens da proposta proativa SMARTFlow quando comparada à proposta reativa do OpenFlow e a *switches* padrão.
- identificação de pontos fracos na norma, problemas e desafios encontrados na comunicação.

1.3 Organização do trabalho

A dissertação está estruturada em oito capítulos da seguinte forma: no Capítulo 2, para embasamento teórico, são apresentados os conceitos de rede elétrica e da Norma IEC 61850. Neste capítulo, os conceitos relacionado a mensagens da norma, requisitos temporais, modelagem de comunicação e dispositivos de uma subestação, protocolos de redundância, dentre outros, são descritos. Em seguida, no Capítulo 3, é feita uma análise sobre os principais trabalhos relacionados ao uso dos recursos de rede no contexto IEC 61850, como diferenciação de tráfego, métodos de recuperação de falha e comunicação *multicast* na camada de enlace, itens de muita discussão na área [8, 7, 9, 18, 19, 20]. No Capítulo 4, os conceitos do arcabouço utilizado, o OpenFlow, são descritos. Nesse capítulo são mostrados os motivos para a escolha desse arcabouço, a arquitetura do OpenFlow e os trabalhos relacionados nessa área. A proposta SMARTFlow é apresentada no Capitulo 5, onde a arquitetura da proposta é detalhada, assim como seus componentes e algoritmos. Em seguida, o Capítulo 6 apresenta as ferramentas utilizadas para implementação da proposta, o ambiente implementação, a descrição dos experimentos e os principais resultados obtidos com o SMARTFlow, assim como a análise dos valores encontrados. Por fim, o Capítulo 7 conclui a dissertação, ressaltando os objetivos alcançados com as propostas. As principais vantagens e desvantagens do SMARTFlow são discutidas, assim como alguns trabalhos futuros que podem ser desenvolvidos.

Capítulo 2

Rede Elétrica, Smart Grids e a Norma IEC 61850[1]

O Sistema Elétrico de Potência (SEP) possui como objetivo primário fornecer energia aos consumidores finais. Este fornecimento é realizado através dos processos de geração, transmissão e distribuição de energia elétrica. Contudo, apesar da evolução tecnológica que o mundo vem sofrendo, nos últimos 100 anos, não houve mudanças revolucionárias na estrutura da rede de energia elétrica [21, 22].

A rede elétrica tradicional consiste em um modelo centralizado, onde a energia é gerada em grandes plantas de geração, normalmente distantes dos grandes centros, percorrendo as linhas de transmissão até chegar às proximidades dos consumidores, onde enfim é distribuída.

Nesse modelo, existem muitas desvantagens e restrições. O fluxo de energia é unidirecional e pouco automatizado [23]. Além disso, não há armazenamento de energia, ou seja, a energia tem que ser consumida assim que gerada. No sistema tradicional, o consumidor é desinformado e não participa do sistema, não provendo *feedback* sobre a qualidade da energia ou sobre a demanda de consumo. Com isso, o sistema tem que ser projetado para o horário de pico, ficando muito tempo ocioso tornando o sistema caro. O mercado de energia não é competitivo, o que leva a pouca inovação nas subestações e linhas de transmissão, e consequentemente os equipamentos utilizados e as tecnologias, algumas vezes, são os mesmos de 40 anos atrás [22].

As experiências têm mostrado que a obsolescência dos sistemas da rede do século 20 não se adapta mais às necessidades do século 21, que incluem não apenas um aumento na demanda por recursos, mas também um aumento na qualidade e na variedade dos

[1] Algumas partes deste Capítulo são baseadas no texto publicado em [6].

serviços providos [22].

Para resolver estes e outros problemas surgiram as redes elétricas inteligentes, ou *smart grids*. Esta é uma solução inovadora que proporciona uma arquitetura integrada para todos os componentes do sistema, incluindo geração, transmissão, distribuição e usuário. Isso significa dizer que, modernas técnicas de comunicação e de tecnologia da informação deverão ser usadas em conjunto com as rede elétricas para prover maior qualidade, eficiência e confiabilidade ao sistema. Nesse novo modelo, toda a inteligência e automação que antes só existiam em parte do sistema, como em subestações, deverão ser levadas para todo o sistema, chegando à casa dos consumidores. Por exemplo, medidores inteligentes poderão ser implantados, evitando fraudes e permitindo a comunicação entre a concessionária e o consumidor, viabilizando a administração de oferta e consumo com autonomia. Com a modernização da rede, novas aplicações serão possíveis, como o acompanhamento em tempo real do consumo, a geração de energia pelo consumidor, o armazenamento de energia e a auto-recuperação da rede elétrica em caso de falha. Além disso, o religamento e desligamento de energia serão feitos de forma automática.

Para se alcançar esses objetivos, a palavra chave é integração, onde a arquitetura é fortemente baseada em redes de telecomunicações com vantagens inerentes, tais como uma maior eficiência e confiabilidade para o sistema, permitindo a comunicação entre os muitos dispositivos inteligentes que agora fazem parte desse novo sistema. Esse novo sistema permite uma coleta exaustiva de dados para subsidiar algumas aplicações e soluções.

Deve-se observar que o aumento do número desses dispositivos inteligentes, que vão desde medidores inteligentes até relés modernos, aumenta significativamente a quantidade de dados. Toda essa integração, automatização, monitoramento e coleta de dados trazem, além de benefícios, muitos desafios. A diversidade de equipamentos que precisam coexistir só aumenta, cada um com requisitos distintos. Além disso, diversos protocolos de comunicação surgiram, trazendo problemas de incompatibilidade na comunicação entre equipamentos de diferentes fabricantes. Sendo assim, observa-se que o principal desafio das *smart grids* é permitir que todo esse sistema converse de forma interoperável com confiança e eficiência. Para solucionar esse problema de comunicação, umas das principais iniciativas é a norma IEC 61850, que propõe uma solução unificada de comunicação e aplicações para garantir a interoperabilidade entre dispositivos de diferentes fabricantes. A norma foi concebida para padronizar a comunicação no SAS, definindo assim, a comunicação entre os dispositivos eletrônicos inteligentes, *Intelligent Electronic Device*s (IEDs) [5]. Inicialmente, a norma era composta por 10 partes, mas, devido às

necessidades e desafios advindos das *smart grids*, sofreu extensões. Mais recentemente, seu nome foi alterado e seu objetivo foi ampliado passando a modelar toda a comunicação no sistema de automação, não apenas em subestações.

As partes mais recentes da norma tratam de assuntos inerentes às *smart grids* como a modelagem dos veículos elétricos, definida na parte 90-8 da norma [24], assim como a sua integração com a geração distribuída, com painéis fotovoltaicos e armazenamento de energia, os quais são detalhados na parte 7-420 da norma [25]. Além disso, assuntos como a troca de dados dos sincrofasores [26], a comunicação entre subestações [27] e a estrutura de comunicação para hidroelétricas [28] são também assuntos discutidos, assim como, várias outras partes que estão em produção. Com isso, percebe-se que a norma IEC 61850 é parte fundamental para o sucesso das *smart grids*, pois toda a modelagem desse vasto sistema é padronizado pela norma, assim como a comunicação e a nomeação dos dados.

2.1 Uma breve introdução à Rede Elétrica e à Proteção

O SEP tem como objetivo suprir seus consumidores com a maior confiabilidade e continuidade de serviço possível. Métodos de proteção e controle são premissas básicas para que esse sistema, composto por um conjunto de usinas, subestações, linhas de transmissão e outros equipamentos que possibilitam a geração, transmissão e distribuição de energia elétrica, funcione conforme o esperado. Esse sistema possui uma separação clara em suas estruturas de geração, transmissão e distribuição. Com isso, cada parte é atendida com soluções específicas em cada área. O sistema elétrico é composto por diversos elementos, dentre os quais temos os seguintes de maior importância:

1. Geração: na geração, as usinas são classificadas conforme os recursos que utilizam, podendo ser hidroelétricas, termoelétricas, eólicas, nucleares, etc. Nesta fase, certo tipo de energia é transformado em energia elétrica. No caso do uso de hidrelétricas, envolve o armazenamento de um fluido, normalmente água de rios, conversão de energia hidráulica do fluido em energia mecânica em uma turbina hidráulica e a conversão da energia mecânica em energia elétrica por um gerador elétrico.

2. Transmissão e Distribuição: condução da energia elétrica das usinas de geração até os consumidores. Após gerada, a tensão é aumentada para que se evite perdas, em seguida, a energia elétrica é transportada em altas tensões, por fim, é distribuída ao consumidor final em baixas tensões.

3. Subestação: subestações são responsáveis por aumentar ou diminuir a tensão na transmissão e na distribuição. Seu funcionamento é autônomo, normalmente suportados por sistemas tipo SCADA (*Supervisory Control and Data Acquisition*) para controle e supervisão remotos. Internamente, a subestação também desempenha funções de comutação, proteção e controle. Entre os problemas tratados por uma subestação, tem-se a interrupção de curtos-circuitos com o uso de disjuntores. Entre os tipos de subestações, destacam-se as subestações de transmissão, as quais conectam linhas de transmissão com voltagens iguais ou diferentes; as subestações de distribuição, as quais conectam linhas de transmissão com linhas de distribuição, além de regular a tensão; e as subestações coletoras, as quais são usadas na geração de energia eólica para ligar a geração com as linhas de transmissão. Assim, subestações têm um papel chave na confiabilidade e na qualidade do serviço elétrico oferecido aos clientes.

O SEP está sujeito a anormalidades e defeitos em sua operação, o que torna a proteção desse sistema um fator de extrema importância. É chamado de proteção o mecanismo do SEP que isola uma área ou um equipamento defeituoso de forma confiável e rápida interrompendo o menor trecho possível da rede, evitando, com isso, grandes perdas. É graças à proteção que as falhas no sistema não se propagam, evitando interrupções em grandes áreas e os apagões. Os principais equipamentos utilizados para realizar a proteção são:

- Relés: São elementos detetores-comparadores e analisadores, auxiliados pelo disjuntor. Cada relé, no sistema convencional, opera um equipamento elétrico, com o objetivo de sanar variações nas condições normais de operação deste equipamento ou do circuito o qual está ligado. Desta forma, promovem a retirada rápida de um elemento do sistema e indicam a localização e o tipo do defeito. Por exemplo, um relé de sobrecorrente tem como função abrir o circuito protegido quando a corrente que passa em seus contatos é maior que um valor pré-estabelecido.

- Disjuntores: São dispositivos de manobra que permitem ligar e desligar dois condutores que fazem parte de uma rede elétrica, seja comandado pelo relé ou pelo operador. Dessa maneira, são responsáveis por desconectar a área defeituosa do sistema isolando as falhas e pelos religamentos. A posição aberta corresponde a uma impedância infinita e posição fechada corresponde a um curto-circuito (impedância nula).

- Transformadores de Instrumentos: O transformador de instrumentos coleta correntes e tensões elevadas e as transforma em níveis adequados para utilização em equipamentos e dispositivos de medição e proteção. Isso é importante porque os relés e dispositivos não aceitam entradas de alta tensão ou de alto valor de corrente. O transformador de instrumento relacionado à transformação de potencial é chamado de Transformador de Potencial (TP), enquanto que o relacionado à corrente é chamado Transformador de Corrente (TC).

Em resumo, os TPs e TCs traduzem os parâmetros do sistema para níveis suportáveis pelos relés, de tal forma que eles possam ler os dados do sistema e identificar o defeito. Uma vez identificado um defeito, o relé envia um comando de abertura para os disjuntores, que, por sua vez, isolam a falta abrindo os terminais mais próximos para que não haja propagação do defeito para outros circuitos e equipamentos. Esse conjunto de ações é chamado de manobra. É necessário que as manobras sejam feitas da forma mais rápida possível para proteger o sistema. Nos esquemas de instalações convencionais, toda a comunicação para manobra necessita de uma instalação elétrica ponto-a-ponto. Desta forma, os sinais de intertravamento e proteção são enviados através de circuitos físicos dedicados, constituídos por fiação de cobre, interligando diretamente os circuitos de comando dos disjuntores.

Com a evolução dos relés de proteção, que passaram a possuir características microprocessadas, o SAS vem mudando para se adaptar a esse novo cenário. Os relés, passam a ter funções adicionais e, além da proteção, passam a controlar e registrar eventos, medidas, etc. Assim, os relés digitais modernos, agora intitulados IEDs, são capazes não só de acionar disjuntores mas também de enviar amostras de sinais de tensão e corrente e de funcionar como um nó na rede trocando mensagens. Com isso, a fiação de comando de cobre rígida é substituída por uma rede de comunicação local, *Local Area Network* (LAN).

Os IEDs, que passaram a possuir a capacidade de se comunicar, inicialmente usavam protocolos diferentes e proprietários. Com isso, os dispositivos de diferentes fabricantes não interoperavam, o que tornava difícil a comunicação na rede. Protocolos de comunicação como o Modbus e o DNP tornaram-se padrões industriais, tornando a rede mais confiável, porém a complexidade para integrar os dispositivos ainda era muito grande, fazendo da rede um sistema complexo [29].

Dessa necessidade, surgiu o incentivo para a construção de um padrão único que especificasse os parâmetros e a forma como os IEDs devem se comunicar. Assim, foi criada a norma IEC 61850 com o objetivo de modelar todo o SEP definindo regras e

padronizando a comunicação.

Além da proteção, tem-se outro elemento importante nesse sistema, a supervisão. Os sistemas de supervisão são responsáveis pelo controle e monitoramento da rede dentro e fora das subestações.

De forma prática, um sistema de supervisão engloba sensores, atuadores e um *software* que permite controlar e monitorar o sistema. O Sistema de Supervisão e Aquisição de Dados (*Supervisory Control and Data Acquisition* (SCADA)) faz exatamente esse papel, supervisionando, controlando, otimizando e gerenciando os sistemas de geração e transmissão de energia elétrica. Por essa razão, as operadoras de energia no Brasil utilizam o SCADA para realizar diversos tipos de medições[1].

Os dados coletados são analisados por especialistas que disparam medidas corretivas de contingência e de planejamento da rede. Uma grande parte dessa análise é automatizada com base em dados históricos, topologia da rede e experiências humanas [30]. Com a evolução das *smart grids*, o SCADA incorporará novos elementos inteligentes, tais como unidades de medição fasorial, relés inteligentes, novas tecnologias com utilização de fontes renováveis, armazenamento de energia em veículos elétricos (EV), etc. [31].

2.2 A Norma IEC 61850

A indústria de energia elétrica está passando por um período de grande mudança com o uso de avançadas tecnologias em um esforço para desenvolver uma rede mais inteligente que possa atender com sucesso aos desafios de hoje e do futuro nas *smart grid* [32]. A Norma IEC 61850 é uma plataforma aberta de proteção e automação de subestações, independente de fornecedores [33], de forma a padronizar os protocolos de comunicação para controle e proteção no sistema elétrico.

A Norma IEC 61850 modela a interconexão dos elementos de automação, representando em um plano lógico todos os elementos envolvidos na comunicação do sistema. Desta maneira, a norma possui um conjunto de funções que interoperam de forma distribuída, podendo estar alocadas em um ou mais IED conectados em rede. Este mesmo princípio é usado para integrar funções de medição, de controle e de proteção. Isto pos-

[1]Exemplos de medições são: diagramas fasoriais de tensões e correntes, queda de consumo, queda de demanda de energia, perfis de curvas de carga de potências ativas e reativas, sensor de abertura de porta da caixa do medidor, inconsistência de data e hora dos medidores, alarmes, inversão do circuito de corrente e de tensão e suspeita na medição sob supervisão e parâmetros específicos de elementos do sistema como, por exemplo, temperatura.

sibilita a substituição dos cabos de controle por redes de comunicação, reduzindo o custo global no comissionamento, na engenharia, no monitoramento, na manutenção e no diagnóstico [33]. A norma recomenda o uso de Ethernet com no mínimo 100Mbps de capacidade.

A norma também determina requisitos temporais rígidos para a comunicação. Esses requisitos estão detalhados na Seção 2.2.2.

A especificação de um SAS utilizando o padrão IEC 61850 é, até certo ponto, semelhante à especificação de um sistema convencional. Devem ser fornecidas informações sobre o diagrama unifilar da subestação[2] as funcionalidades requeridas, os requisitos de desempenho, as interfaces com o processo e com outros IEDs, além de protocolos, condições ambientais, índices de confiabilidade admitidos ou critérios de tolerância a falhas aceitáveis, etc. O uso da Norma IEC 61850, porém, demanda, adicionalmente, outros requisitos, como características do sistema de comunicação, além de toda a documentação em linguagem de configuração de subestações (*Substation Configuration Language* (SCL)) e procedimentos de teste específicos para o SAS que está sendo adquirido [33]. A linguagem de configuração adotada pela norma será descrita na Seção 2.2.3.

2.2.1 Modelagem dos dispositivos

Os dispositivos são representados de acordo com a sua função, ou seja, têm suas funcionalidades textualmente descritas. Desta forma, o modelo de dados especificado pela norma define atributos e funções dos dispositivos físicos de uma subestação elétrica.

O modelo de dados é baseado numa estrutura de dados orientada a objeto [34], utilizando os seguintes conceitos:

- Classe: representação de um conjunto de objetos com características afins.
- Objeto/instância de uma classe: serve para armazenar estados através de seus atributos e interagir com outros objetos através de mensagens;
- Atributos: são as características de um objeto;
- Métodos: são as funções implementadas nos objetos, como por exemplo, a capacidade de enviar e receber mensagens;

[2] O diagrama unifilar é uma representação simplificada das interligações entre equipamentos, onde os aspectos do circuito elétrico como localização dos elementos, percursos de uma instalação, condutores, distribuição da carga, proteções, dentre outros são contemplados.

- Herança: funcionalidade de uma classe se estender a outra classe, herdando seus métodos e atributos.

A norma define dispositivos físicos e lógicos. Um dispositivo físico é definido como um dispositivo que se conecta à rede através de um endereço específico. Dentro de cada dispositivo físico, pode haver um ou mais dispositivos lógicos (*Logical Device* (LD)). O dispositivo lógico especifica um grupo de nós lógicos (*Logical Node* (LN)) com as mesmas características, por exemplo controle, medição, nível de tensão, subestação ou vão que faz parte.

Cada nó lógico contém um ou mais objetos (*Data Objects* (DO)), compostos por atributos (*Data Attributes* (DA)). O objeto de um nó lógico representa um dado de uma função de automação e controle. Por exemplo, um objeto seria a posição de uma chave secionadora ou de um disjuntor. O atributo é o valor desse objeto, como o estado fechado ou aberto da posição de um disjuntor. Assim, seguindo o modelo de orientação a objeto, a norma disponibiliza uma visão hierarquizada para classificar as funções exercidas por cada dispositivo da rede, sendo iniciada pelo dispositivo físico até alcançar o atributo de dados.

A norma IEC 61850, padroniza grupos de nós lógicos [35], com intuito de agrupar funções de proteção e controle. Cada nó lógico possui uma denominação iniciada com a letra do grupo que faz parte, por exemplo:

- **P**XXX - Funções de Proteção
  - **P**TOC - Proteção de Sobrecorrente (***T**ime **O**ver**C**urrent*)
  - **P**DIS - Proteção de Distância (**DIS**tance)
- **C**XXX - Controle
  - **C**SWI - Controlador de Chaveamento (***SWI**tch*)
- **M**XXX - Medições
  - **M**MXU - Medição Operativa e Indicativa (***M**easurement **U**nit*)
- dentre outras

É dessa forma que, a funcionalidade do nó lógico é textualmente descrita, com a primeira letra indicando de qual grupo faz parte, e as restantes indicando seu papel.

A Figura 2.1 ilustra a estrutura hierárquica do modelo de dados e provê um exemplo de nomes de objetos para identificar uma instância de uma classe em um nível hierárquico.

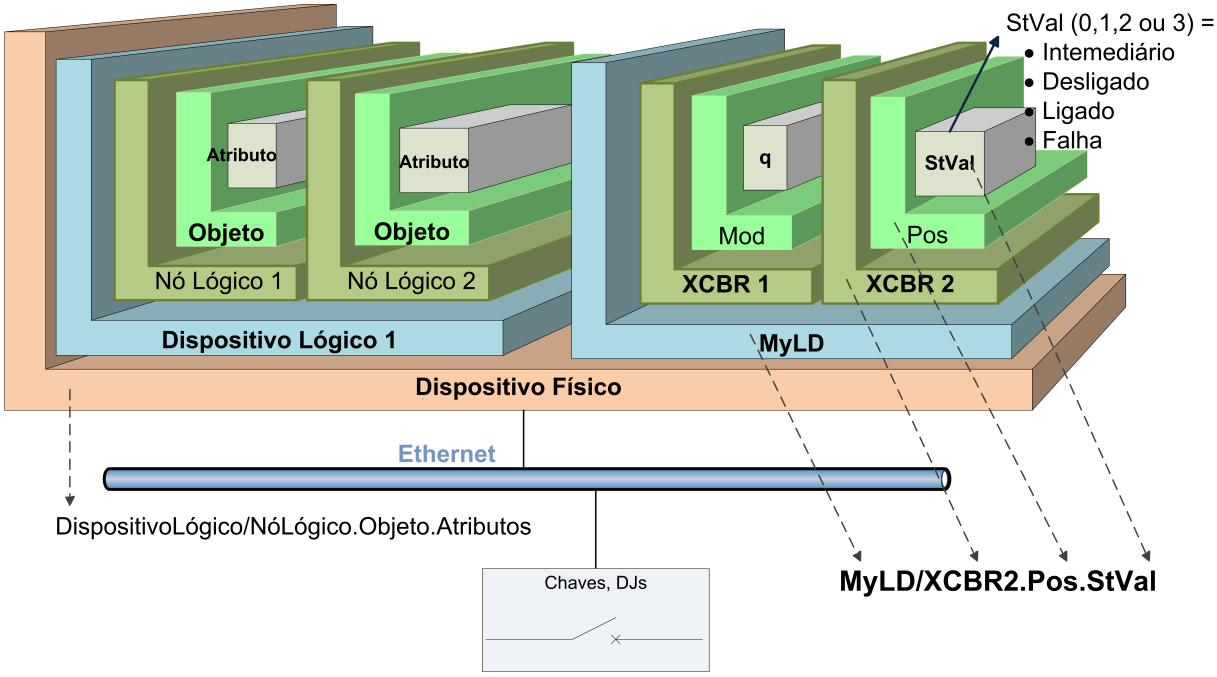


Figura 2.1: Exemplo de uma referência de nome de objetos do IEC 61850 e sua estrutura hierárquica.

De forma geral, cada objeto de dados tem um nome único. Esses nomes são determinados pela norma e funcionalmente ligados à finalidade do sistema de potência.

Com base no nome, é possível identificar por qual 'caminho' pode-se chegar até o dado, ou seja, em qual dispositivo físico, dispositivo lógico e nó lógico está o dado desejado. Nesse modelo de dados, qualquer dado pode ser diretamente acessado, usando para isso o seu caminho.

A Figura 2.1 apresenta um disjuntor que é modelado por um nó `XCBR` lógico que contém uma variedade de objetos, entre os quais, `Pos` para indicações associadas à posição do equipamento. O objeto `Pos` possui, por exemplo, o atributo `StVal`, que indica o estado atual do disjuntor (intermediário, aberto, fechado ou falha). Pode-se referenciar o nome de objetos da seguinte forma:

`DispositivoLógico/NóLógico.ObjetodeDados.AtributosdeDados`

Na figura, o termo `MyLD` representa o dispositivo lógico, o termo `XCBR2` representa o nó lógico, o termo `Pos` representa o objeto de dados e finalmente o termo `stVal` representa o atributo de dados.

Com isso tem-se:

`MyLD/XCBR2.Pos.StVal`

Então, o caminho acima, seria usado para acessar informações sobre o estado de um disjuntor.

Esse modelo de informação hierárquica é ilustrado também na Figura 2.2, onde mostra-se o nó lógico `XCBR` que é a raiz ao nível dos nós lógicos e a árvore completa desse nó com as referências a essa raiz [36].

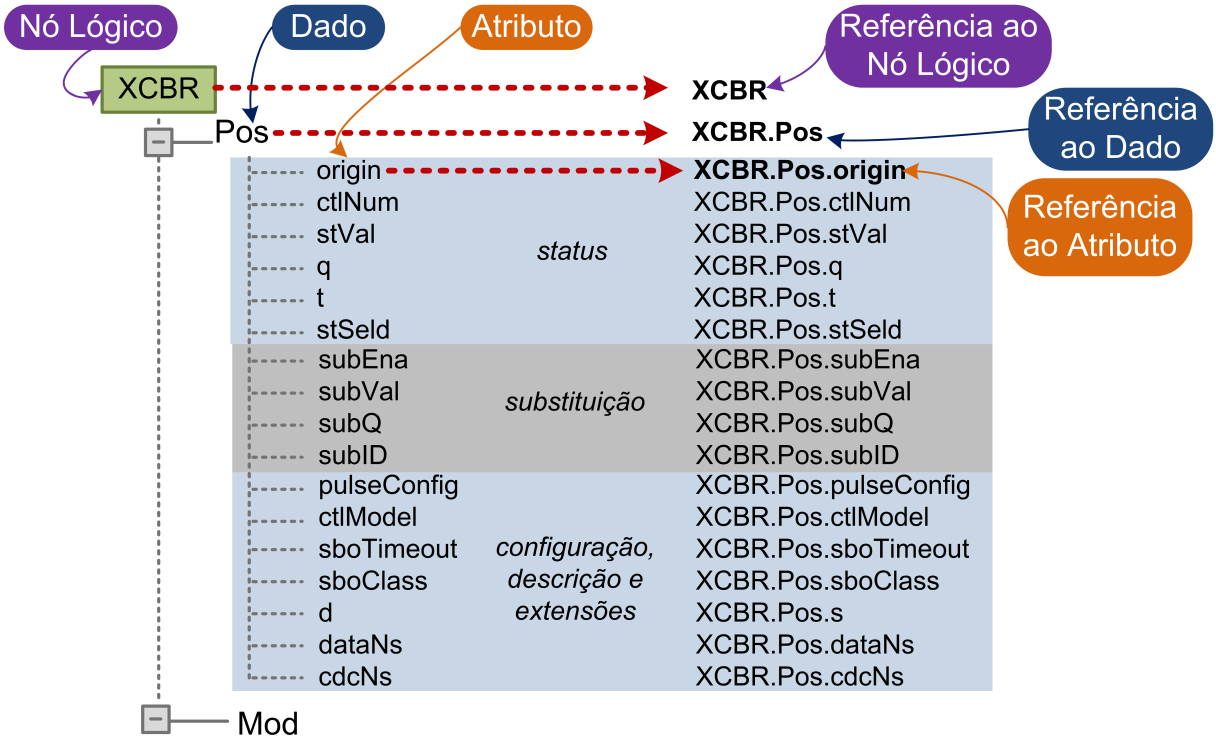
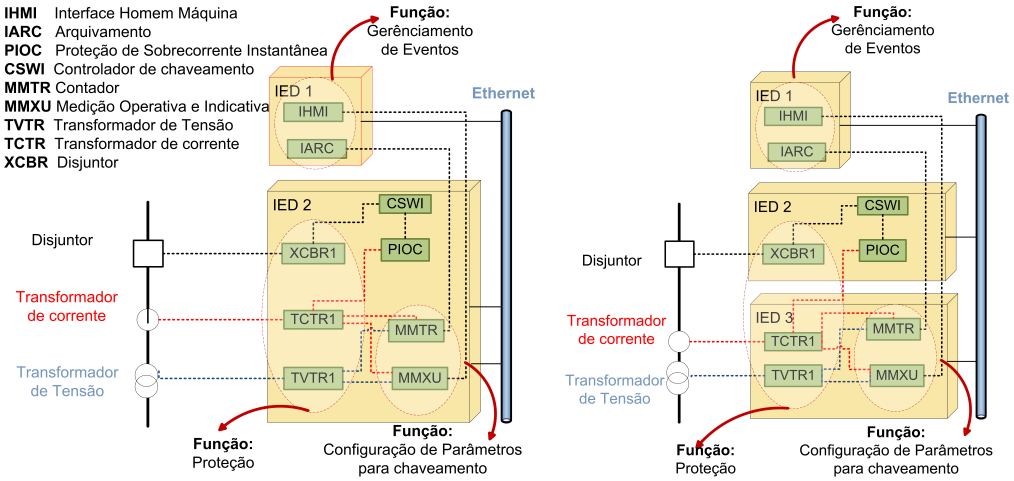


Figura 2.2: Exemplo simples de modelagem de um IED [36].

Para diferenciar nós lógicos com o mesmo nome, um sufixo e/ou prefixo podem ser usados. Por exemplo, para o nó lógico `XCBR` pode-se usar um sufixo de 1 a n pra diferenciar cada disjuntor, por exemplo, `XCBR1`...`XCBRn`. Podemos usar também um prefixo, por exemplo `Q0XCBR`, onde `Q0` é o prefixo, `XCBR` é o nó lógico do grupo Disjuntores. Além disso, pode-se usar o prefixo e o sufixo na mesma estrutura, por exemplo, `Q0XCBR1`. Com isso, representa-se logicamente todo o sistema.

Um conjunto de nós lógicos com características afins formam uma função do sistema. Essas funções se referem, em alto nível, a tarefas que devem ser executadas, sendo utilizadas para controlar, monitorar e proteger o sistema. Os IEDs são multifuncionais e, portanto, podem disponibilizar várias funções em um único elemento, como mostra a Fi-

gura 2.3(a). Além disso, uma função pode não estar localizada em um único dispositivo, mas distribuída entre vários dispositivos físicos, conforme Figura 2.3(b), se comunicando através da rede de telecomunicações.



(a) Distribuição em 2 IEDs. (b) Distribuição em 3 IEDs.

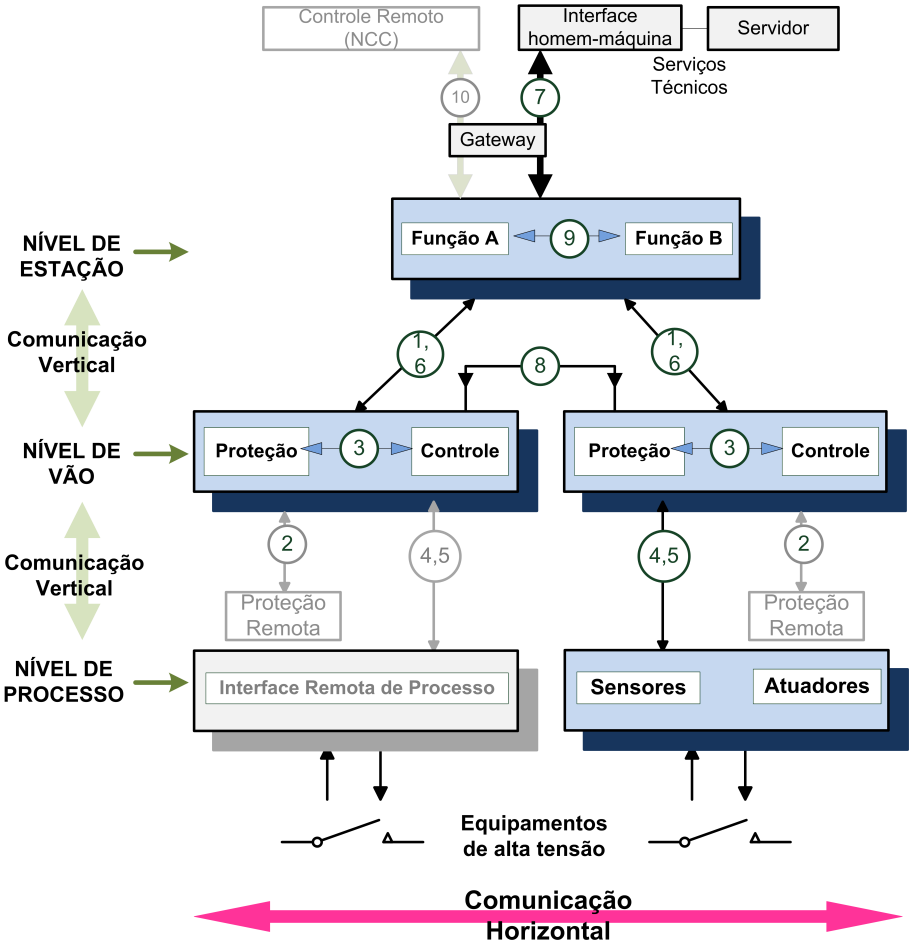
Figura 2.3: Exemplo de distribuição de Funções no IED.

2.2.2 Modelagem da Comunicação

Com relação à comunicação e ao sistema de automação de subestações, a norma recomenda que se estruture o sistema em diferentes níveis hierárquicos [37], a saber: estação (*station level*), vão (*bay/unit level*), ou processo (*process level*). Dessa forma, as funções em subestações como proteção, controle e supervisão, podem ser logicamente alocadas. Estes níveis compreendem:

- Funções de nível de processo: todas as funções que interagem com os dispositivos de nível de processo, tipicamente I/O remotos, transformadores, seccionadoras e disjuntores.
- Funções de nível de vão: todas as funções que interagem com os dispositivos típicos de nível de vão, como relés de proteção, medidores de energia, equipamentos de teleproteção e oscilógrafos.
- Funções de nível de estação: funções que utilizam dados de um vão ou de toda a subestação para interagir com o equipamento primário ou com o operador. Os dispositivos nesse nível compreendem computadores, a interface homem-máquina, e interfaces com enlaces para outras estações.

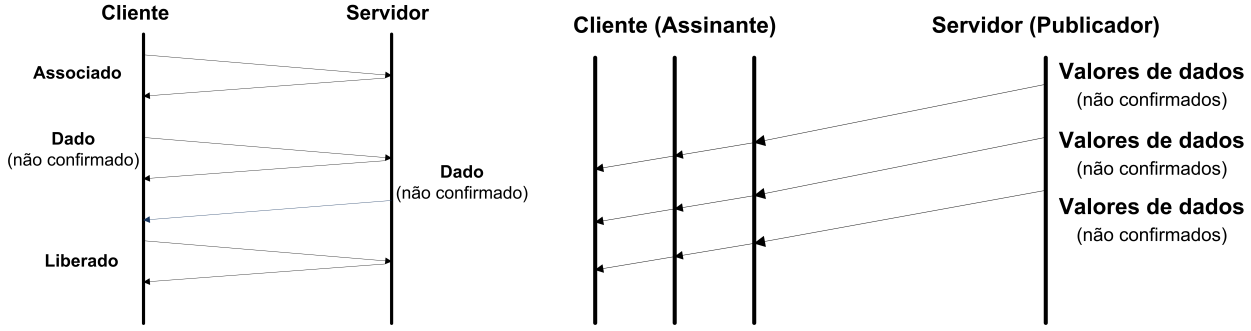
É importante observar que essa separação em níveis hierárquicos é lógica. Fisicamente, existe apenas um link físico por onde trafegam as informações dos barramentos de estação, vão e processo para uma implementação completa da IEC 61850 [38]. Estes níveis hierárquicos podem ser compreendidos através da interpretação da Figura 2.4, que ilustra a arquitetura de comunicação, assim como a separação lógica em níveis dos elementos e as principais interfaces entre esses.



1) Troca de dados de proteção entre os níveis de vão e de estação;
2) Troca de dados de proteção entre os níveis de vão e de proteção remota;
3) Troca de dados dentro do nível de vão;
4) Troca de dados instantânea do TC e TP entre os níveis de processo e de vão;
5) Troca de dados de controle entre os níveis de processo e de vão;
6) Troca de dados de controle entre os níveis de vão e de estação;
7) Troca de dados entre o nível de estação e a estação de trabalho remota do engenheiro;
8) Troca direta de dados entre diferentes níveis de vão, especialmente para funções rápidas como as de intertravamento;
9) Troca de dados dentro do nível de estação;
10) Troca de dados de controle entre os dispositivos e o centro de controle remoto.

Figura 2.4: Níveis de uma subestação e interfaces de comunicação entre estes níveis de acordo com a arquitetura proposta na IEC 61850 [37].

Além de especificar as interfaces, a norma também define o modo de comunicação realizado pelos dispositivos dentro do SAS para requisitar um serviço como pesquisa de objetos, manipulação de *logs*, leitura de informações, dentre outros. Os modelos de comunicação fornecem um mecanismo para controlar o acesso das instâncias de um dispositivo, podendo ser de dois tipos [39]: *Two Party Application Association* (TPAA) ou *Multicast Application Association* (MCAA), como ilustrado na Figura 2.5.



(a) TPAA, baseado em mensagens *unicast*.

(b) MCAA, baseado em mensagens *multicast*.

Figura 2.5: Princípios de comunicação TPAA e MCAA da norma IEC 61850 [39].

Observa-se que o modelo de comunicação TPAA transmite os pedidos de serviços e resposta através de uma troca bidirecional de informação ponto a ponto orientada a conexão. A conexão é confiável, dispondo de um controle de fluxo fim a fim. O modelo de comunicação MCAA permite uma troca de informação unidirecional entre um publicador (*Publisher*) e um ou mais assinantes (*subscriber*). O assinante deve ser capaz de detectar perdas ou duplicação da informação recebida.

O modelo TPAA é usado para a comunicação das mensagens *Manufacturing Message Specification* (MMS), descritas na Seção 2.2.2.1 e o MCAA para mensagens *Generic Object Oriented Substation Event* (GOOSE), descritas na Seção 2.2.2.2. As mensagens *Sampled Values* (SV) podem usar os dois modelos dependendo da aplicação [39], e também são descritas na Seção 2.2.2.2.

Esses modelos devem permitir que as garantias temporais de comunicação, estabelecidas pela IEC 61850, sejam atendidas. Essa garantia é imprescindível para a correta execução das funções dos dispositivos e do desempenho geral do sistema. Esses requisitos temporais são definidos de acordo com o tipo de mensagem e estão descritos na Tabela 2.1 que resume os tipos de mensagens disponibilizas pelo padrão. Suas respectivas classes de desempenho são [37]:

- Classe P1: Refere-se ao nível de vão de distribuição ou aos níveis cujo requisito

temporal não seja de alta criticidade;

- Classe P2: Aplica-se ao nível de vão de transmissão;
- Classe P3: Designada para o nível de vão de transmissão com características críticas de sincronização.

Tabela 2.1: Tipos de mensagens suportadas pelo padrão IEC 61850 [37].

Tipo	**Descrição**	**Exemplo**	**Classes**	**Mensagem e** Requisitos Temporais
1A	Mensagens Rápidas	*Trips*	P2 e P3	GOOSE (3ms)
			P1	GOOSE (10ms)
1B	Mensagens Rápidas (outras)	Comandos, Mensagens Simples	P2 e P3	GOOSE (20ms)
			P1	GOOSE (100ms)
2	Velocidade Média	Valores de Medidas	-	MMS (100ms)
3	Velocidade Baixa	Parâmetros	-	MMS (500ms)
4	Rajada de Dados	Saída de dados dos transformadores	P2 e P3	SV (3ms)
			P1	SV (10ms)
5	Transferência de Arquivos	Arquivos grandes	-	MMS($\geq$1000ms)
6A	Sincronização de Tempo A	Sincronização (*station bus*)	-	TimeSync
6B	Sincronização de Tempo A	Sincronização (*process bus*)	-	TimeSync
7	Mensagem de Comando	Comandos da estação HMI	-	MMS(500ms)

Por exemplo, para as mensagens do Tipo 1A (*trip*), por serem consideradas as mensagens rápidas mais importante na subestação, são definidos limites temporais de 3ms para as classe P2 e P3 e 10ms para a classe P1. As mensagens do Tipo 6A, utilizadas para sincronização de tempo "A", são definidas para um desvio temporal de módulo 1 ms no máximo. Já as mensagens do Tipo 6B, usadas para sincronização de tempo "B", toleram desvios temporais de módulo 4 μs e 1 μs, para as classe P2 e P3, e +/- 25 μs para a classe P1.

2.2.2.1 Mensagem MMS

A MMS é padronizada pela ISO 9506 com o objetivo de transferir dados em tempo real e informações de controle e supervisão entre dispositivos de rede e/ou aplicações. O padrão é desenvolvido e mantido pelo Comitê Técnico ISO 184 (TC184), e foi incorporado à norma

IEC 61850 [13]. Essa mensagem roda sobre TCP/IP ou OSI dependendo da aplicação. Em geral, a MMS atende os sistemas de aquisição de dados de um sistema de supervisão, como, por exemplo, o sistema SCADA. No SCADA, existe um ponto terminal, chamado de *Remote Terminal Unit* (RTU), o qual coleta medidas da rede elétrica e as transmite para o centro de operações, representado como o servidor na Figura 2.6. No centro de operações, existe um servidor do SCADA, acessível aos operadores através de uma interface homem-máquina. As mensagens trocadas entre o RTU e o servidor SCADA são do tipo MMS, pois não têm requisitos rígidos com relação a atrasos.

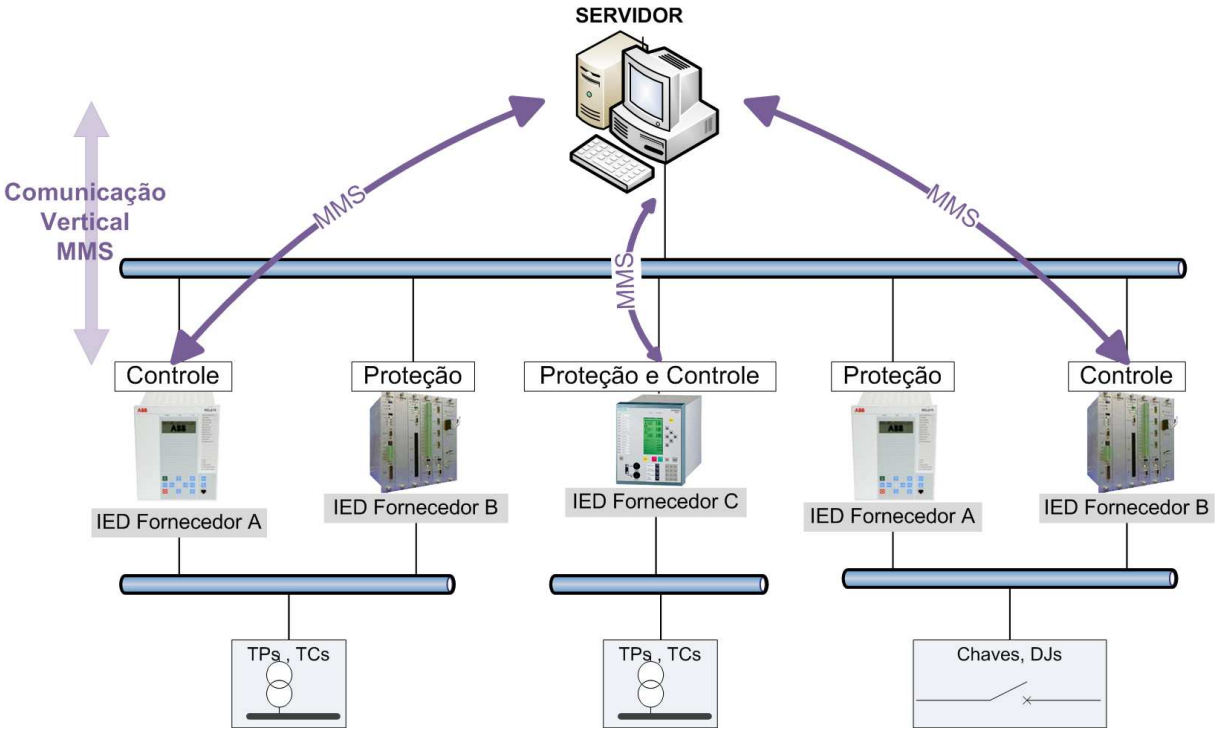


Figura 2.6: Comunicação dentro de uma subestação via MMS, através da troca de mensagens entre o cliente e o servidor.

Outros tipos de mensagens MMS são aquelas geradas pelos IEDs para a estação de controle, contendo dados para o planejamento e a administração da rede de produção. O conteúdo dessas mensagens abrange alarmes menos críticos, eventos, medições de parâmetros envolvidos no provimento de energia elétrica, comutações de elementos *backup* e de contingência etc.

2.2.2.2 Mensagens GOOSE e SV

A GOOSE é uma mensagem na qual conjuntos de dados de *status* ou valor são agrupados em um datagrama e transmitidos dentro de um período curto de tempo. As mensagens

GOOSE são orientadas a eventos, consequentemente, tendo o seu disparo de forma assíncrona. Essas mensagens são direcionada às aplicações de proteção em subestações. Como será visto a seguir, na Seção 2.2.2.2, as mensagens GOOSE têm um mecanismo de retransmissão para aumentar a confiabilidade na entrega.

As mensagens SV têm por objetivo possibilitar o envio dos sinais de tensão e corrente digitalizados, também com restrição rígida de tempo.

Para que se consiga manter o atraso na rede dentro dos limiares de tempo descritos na Tabela 2.1, as mensagens GOOSE e SV são mapeadas diretamente em Ethernet [13]. Um pacote Ethernet tem a estrutura mostrada na Figura 2.7, onde o campo ethertype indica o tipo de protocolo no pacote ethernet. Para GOOSE o valor $88B8$ e para SV o valor $88BA$, ambos em hexadecimal.

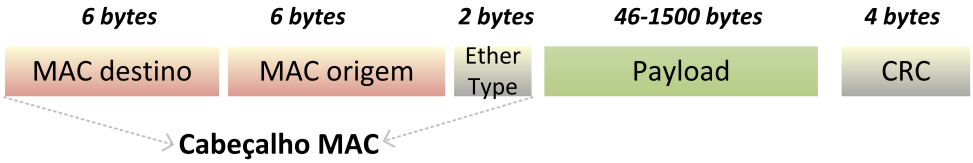


Figura 2.7: Estrutura do Pacote Ethernet (64 – 1518 bytes)

O envio de mensagens em camada de enlace resolve o problema do atraso pertinente às mensagens do tipo cliente-servidor de camada de rede, entretanto remove a confiabilidade que seria garantida por meio de estabelecimento de conexões e confirmações de recepção das mensagens. Para contornar a falta da confirmação de entrega do pacote e aumentar a confiabilidade nas mensagens GOOSE, a norma define um mecanismo baseado em temporização para reduzir o impacto das perdas de pacotes. Desta maneira, uma mesma mensagem GOOSE é enviada diversas vezes, aumentando progressivamente o intervalo entre as retransmissões, até que um novo evento ocorra, reiniciando o processo, ou se alcance um limite máximo de retransmissões. Esse procedimento é definido na parte 8-1 da norma [13] e exemplificado na Figura 2.8, onde:

- $T0$ - retransmissão em condições estáveis (tempo máximo, $Tmax$)
- $(T0)$ - retransmissão em condições estáveis sendo interrompida por um evento
- $T1$ - tempos de retransmissão mais curtos após um evento
- $T2, T3$ - incremento de tempo de retransmissão até alcançar $Tmax$ e retornar a uma condição estável.

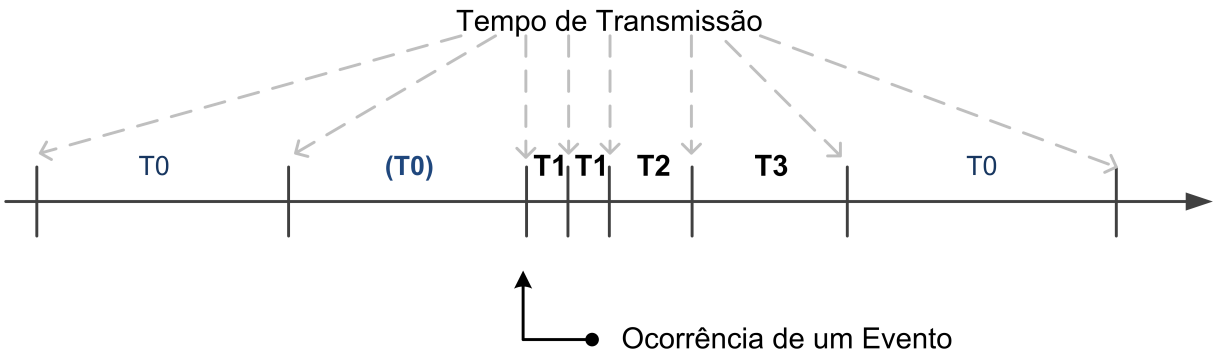


Figura 2.8: Tempos de Transmissão para eventos. Mensagens GOOSE [13]

Observa-se, na Figura 2.8, que os intervalos de retransmissão após a ocorrência de um evento são aumentados gradativamente até alcançar o tempo máximo T0 ($Tmax$), que representa a retransmissão já em condições estáveis do sistema. Assim, uma confiabilidade adicional é conseguida através da retransmissão dos mesmos dados, com aumento gradual de um campo da GOOSE chamado $SqNum$, que é um número de sequência, e do tempo de retransmissão. Cada mensagem carrega um parâmetro *timeAllowedToLive* que informa ao dispositivo receptor o tempo máximo de espera para a retransmissão seguinte. Dessa forma, se uma nova mensagem não for recebida dentro desse intervalo de tempo, o receptor assumirá que houve um problema na comunicação [13].

As mensagens GOOSE são trocadas entre IEDs para envio de alarmes críticos, contendo informações que permitem ao IED receptor tomar conhecimento da ocorrência de um novo evento, sabendo qual foi e quando ocorreu este evento para tomar uma ação apropriada. Neste tipo de comunicação, os IEDs trocam informações entre si utilizando *multicast*, no caso MCAA, seguindo o modelo publicador/assinante [13]. Dessa forma, o publicador envia as mensagens para rede e apenas os assinantes interessados, que formam um grupo *multicast* específico, abrem o pacote. Intitula-se comunicação horizontal essa troca de mensagens. Essa comunicação é ilustrada na Figura 2.9.

No caso da mensagem SV, a comunicação pode ser feita com *muticast* ou *unicast*, dependendo da aplicação[3].

Para que essa comunicação *multicast* ocorra, as mensagens precisam ser configuradas com o endereço *multicast* pelo qual elas irão se propagar. Com relação à recomendação da norma para estrutura desse endereço, seus anexos informativos B da parte 8-1 [13] e parte 9-2 [40] descrevem a estrutura do endereço *multicast* da seguinte forma:

[3]Essa dissertação, pelos seus objetivos, dará enfoque às mensagens GOOSE

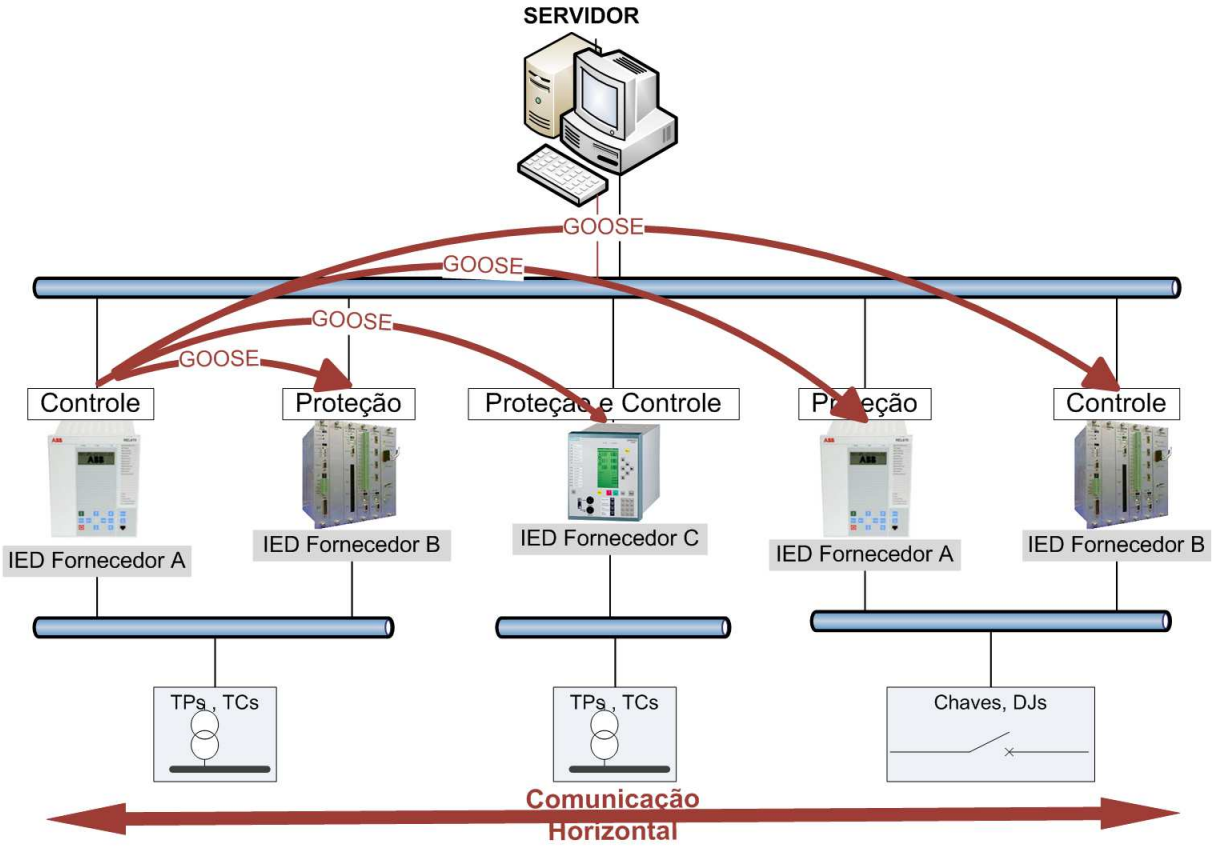


Figura 2.9: Comunicação dentro de uma subestação via mensagens GOOSE, com um IED notificando aos demais sobre algum alarme específico

- Os três primeiros octetos são atribuídos pelo IEEE com sendo 01-0C-CD.
- O quarto octeto deve ser 01 para GOOSE, e 04 para *Sample Values*.
- O valor 00-00-00-00-00-00 deve ser usado para indicar que o endereço *multicast* não foi configurado.
- Os dois últimos octetos devem ser usados como endereços individuais atribuídos pelo intervalo definido na Tabela 2.2.

Tabela 2.2: Faixa de endereços *multicast* recomendados [13, 40].

Serviço	**Endereço de Início** (hexadecimal)	**Endereço Final** (hexadecimal)
GOOSE	01-0C-CD-01-00-00	01-0C-CD-01-01-FF
Multicast Sampled Values	01-0C-CD-04-00-00	01-0C-CD-04-01-FF

Além do esquema de retransmissão para aumentar a confiabilidade na comunicação, a norma define, também, a marcação de prioridade, feita com base no IEEE 802.1Q [12],

para segmentar o tráfego crítico e de alta prioridade do tráfego de baixa prioridade [13, 40], visando um aumento no desempenho e na segurança do sistema.

O padrão 802.1Q especifica uma *tag* que é acrescida a um quadro *Media Access Control* (MAC) Ethernet, permitindo o uso de VLANs com diferentes prioridades nas mensagens. Dessa forma, torna-se possível a criação de redes virtuais entre os IEDs que se encontram conectados a uma mesma rede física. Para marcar o pacote com as *tags* do IEEE 802.1Q/802.1p, os seguintes campos, ilustrados na Figura 2.10, devem ser adicionados:

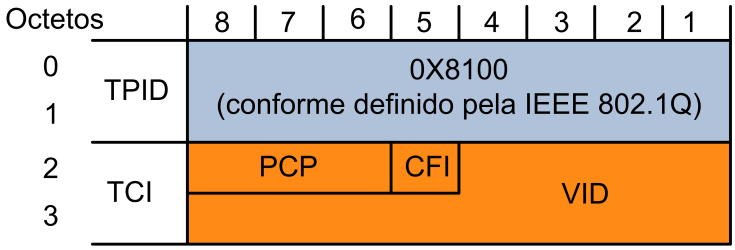


Figura 2.10: Estrutura da *tag* [13, 40].

- *Tag Protocol Identifier* (TPID): campo com 16 bits. Valor definido como $0x8100$, o que indica que o pacote contém *tags* IEEE 802.1Q/802.1p.
- *Tag Control Information* (TCI): campo com 16 bits. Contém os seguintes campos:
  - *Priority Code Point* (PCP): campo com 3 bits. É o código de prioridade IEEE 802.1Q que define oito níveis de prioridade (2^3), sendo o nível zero como mais baixo e sete como mais alto. Pode ser usado para dar prioridade a diferentes classes de tráfego, como GOOSE e SV. Os valores *default* estão descritos na Tabela 2.3, no caso, o valor quatro para ambas as mensagens.
  - *Canonical Format Indicator* (CFI): campo com 1 bit. Se o valor for zero, o endereço MAC está no formato canônico. Para essa norma e para *switches* Ethernet, o bit deve ser ajustado como zero[4].
  - *VLAN Identifier* (VID): campo com 12 bits, permitindo até 4094 VLANs. Especifica a qual VLAN aquele pacote pertence e seu uso é opcional. Um valor 0 indica que o quadro não pertence a qualquer VLAN. Neste caso, tem-se apenas a etiqueta de prioridade.

[4]Se definido o valor igual a um, seguirá o campo ethertype conforme ISO / IEC 8802-3. Essa forma é usada para ser compatível com Redes Token Ring.

Tabela 2.3: Valores *default* atribuídos para Ethertype, APPID, VLAN IDs, e Prioridades [13].

Uso	**Valor Ethertype** (hexadecimal)	**APPID**	**Prioridade** *Default*	**VID** *Default*
IEC 61850-8-1 GOOSE Type	88-B8	00	4	0
IEC 61850-9-2 Sampled Values	88-BA	01	4	0
IEC 61850-8-1 GOOSE Type 1A	88-B8	10	4	0

Ressalta-se que o campo VLAN ID igual a um está reservado para fins de gerenciamento no *switch* Ethernet e, portanto, não deve ser usado para GOOSE ou SV [13, 40]. Esses campos são acrescidos entre o campo de endereço MAC de origem e o campo ethertype, conforme ilustrado na Figura 2.11, usando um quadro GOOSE como exemplo.

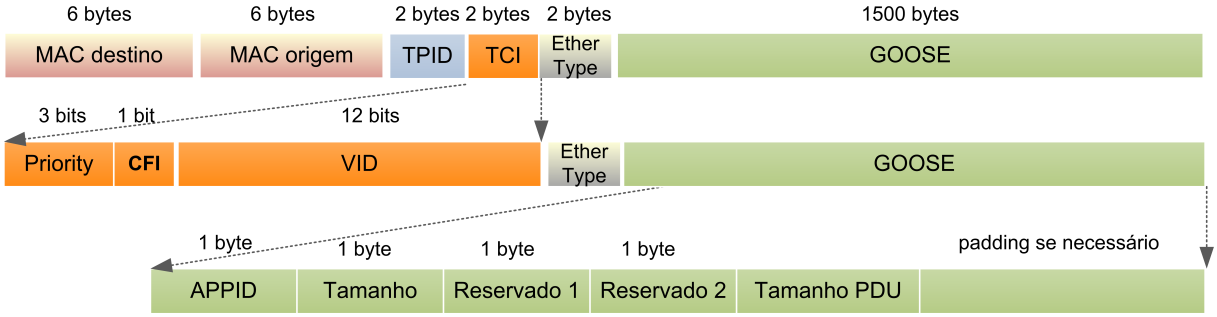


Figura 2.11: Estrutura do Quadro Ethernet

A norma define que todas as implementações que receberem GOOSE ou SV tem que ser capazes de entender qualquer VID, assim como as prioridades. Desta forma, devem ser capazes de processar mensagens que contenham, e que não contenham, informações IEEE 802.1. Nesse último caso, processa-se o campo ethertype [13] . Os valores atribuídos para o campo ethertype são mostrados na Tabela 2.3.

2.2.3 Linguagem de configuração de Subestações (*Substation Configuration Language* - SCL)

A norma IEC 61850 padroniza uma linguagem de descrição que norteia a configuração do sistema. Isto significa dizer que são configurados desde os canais de comunicação até a alocação de funções para os sistemas de automação. Ela é designada como Linguagem de Configuração de Subestação (*Substation Configuration Language* - SCL) [41]. A SCL é baseada na *eXtensible Markup Language* (XML) [41]. Desta forma, incorporou conceitos

de herança e referências abstratas de linguagens orientadas a objetos, se encaixando com a modelagem dos dispositivos descrita na norma. Seu objetivo principal é padronizar os atributos de configuração, ou seja, criar uma nomenclatura uniformizada, de maneira a permitir configurações de IEDs com maior segurança e confiabilidade. O intuito é manter a interoperabilidade, garantindo a troca de dados entre IEDs independente do fabricante.

A linguagem SCL é composta pelos seguintes arquivos de configuração [41], no que diz respeito à configuração dentro de subestações:

1. *IED Capability Description* (ICD): arquivo (.icd) que contém todas as características e funcionalidades do IED. Nele, estão descritas todas as funções que poderão ser utilizadas nos dispositivos. Este arquivo deve ser fornecido pelo fabricante do IED com o intuito de ser uma espécie de *template*;

2. *System Specification Description* (SSD): arquivo (.ssd) que contém a especificação completa de um sistema de automação de subestações, incluindo o diagrama unifilar para a subestação, os seus nós lógicos e o modelo de tipo de dados requeridos;

3. *Substation Configuration Description* (SCD): arquivo (.scd) é composto pela união do arquivo que especifica a capacidade dos IEDs (arquivo .icd) com o arquivo que especifica o sistema (arquivo .ssd). O arquivo .scd descreve detalhadamente a subestação no que tange a comunicação, e contém uma seção de configuração de comunicação e uma seção de descrição da subestação. Desta maneira, este arquivo pode conter, por exemplo, as seguintes informações: aonde está alocado cada nó lógico do sistema, endereços de rede, endereços de grupos *multicast*, etc;

4. *Configured IED Description* (CID): é o arquivo (.cid) que contém a descrição de configuração de um IED específico. Este arquivo possui as funções parametrizadas ou habilitadas pelo usuário no IED. É este arquivo que é configurado em cada IED individualmente.

A formatação em XML permite que a descrição da configuração de um IED seja passada a uma ferramenta de engenharia de aplicação e comunicação, no nível de sistema, e retorne com a descrição da configuração do sistema completo para a ferramenta de configuração do IED [38]. A Figura 2.12 exemplifica o processo de composição dos arquivos citados.

Nessa figura, é apresentado o configurador do sistema, o qual é uma ferramenta de gerenciamento para fazer configurações. No ambiente de trabalho da engenharia têm-se os seguintes passos numerados na figura:

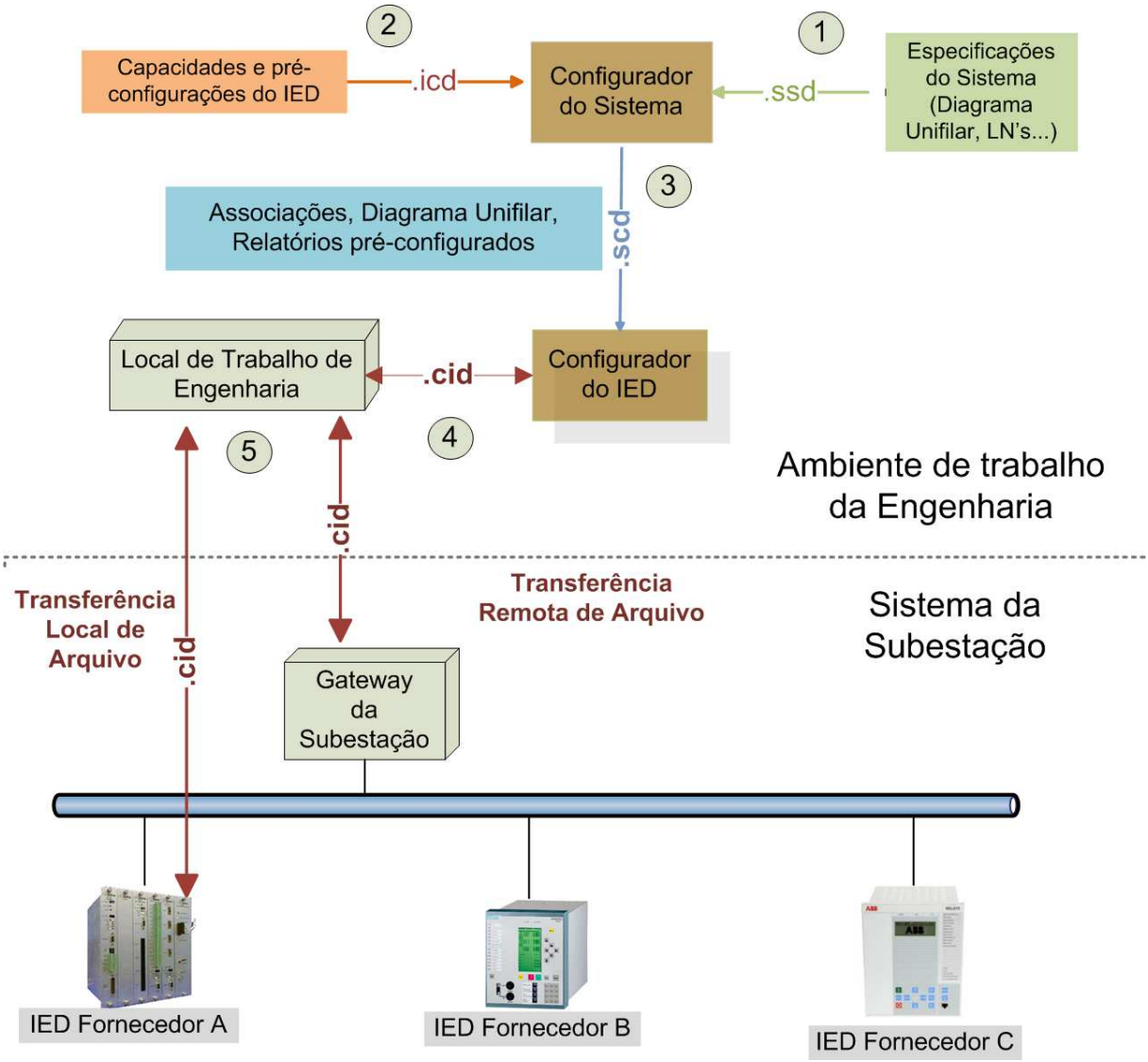


Figura 2.12: Arquitetura de composição dos arquivos da linguagem SCL [41].

- *Passo 1*: A configuração do sistema de automação, contendo o diagrama unifilar, os nós lógicos utilizados e o modelo de tipo de dados necessários são entregues à ferramenta de configuração do sistema (configurador do sistema), na forma do arquivo .ssd.

- *Passo 2*: O arquivo .icd contendo a capacidade dos IEDs com todas as características e funcionalidades destes deve também ser entregue ao configurador do sistema.

- *Passo 3*: De posse desses dois arquivos, .icd e .ssd, o configurador do sistema gera o arquivo .scd com o conteúdo descrito acima. Este provirá os recursos necessários para que a ferramenta de configuração do IED (configurador do IED) defina o arquivo CID com a configuração dos IEDs.

- *Passo 4*: O configurador do IED gera os arquivos .cid, os quais especificam os parâmetros com os quais o IED deve operar. Este, então, é enviado ao local de trabalho de engenharia.
- *Passo 5*: Por sua vez, o local de trabalho de engenharia distribui o arquivo de forma local (diretamente para o IED) ou de forma remota, enviando o arquivo para que um *gateway* o distribua.

A linguagem SCL, em seu escopo completo, permite descrever modelos que abordam:

- Estrutura do sistema primário (potência): relata a forma pela qual os equipamentos estão conectados e quais funções serão utilizadas;
- Sistema de comunicação: descreve como os IEDs serão conectados à rede, e em quais portas de comunicação;
- O nível de aplicação da comunicação: informa qual será o agrupamento de dados a ser transferido, a maneira pela qual os IEDs acionarão o envio e qual o serviço escolhido;
- A configuração de cada dispositivo lógico no IED, os nós lógicos com suas respectivas classes e tipo de dados, relatórios e conteúdo dos dados;
- Definições de tipo para cada instância de nó lógico;
- Relacionamento entre as instâncias dos nós lógicos e seus respectivos IEDs.

2.3 Protocolos de redundância

Com o que foi visto até o momento, a automação fez com que o SEP dependa, cada vez mais, da infraestrutura de redes e da comunicação entre os seus dispositivos. Devido a essa modernização causada pela inserção maciça da automação, aumentam as preocupações inerentes à redundância e ao restabelecimento da rede em caso de falha. A alta disponibilidade da rede tem sido um assunto muito discutido e é tratada na norma IEC 62439 [42]. Essa norma descreve alguns desses métodos, resumidos na Tabela 2.4. Cada um desses métodos apresentam características particulares de desempenho, e só podem ser empregados em determinado tipo de topologia de rede.

A Norma IEC 61850 optou por usar os métodos PRP e HSR, definidos na parte 3 da norma IEC 62439 [42]. Esses métodos são baseados na duplicação da informação

Tabela 2.4: Exemplos de protocolo de redundância [19].

Protocolo	**Solução**	**Tempo de Recuperação**
Cross-network Redundancy Protocol (CRP) [43]	IEC 62439-4	1s
Distributed Redundancy Protocol (DRP) [44]	IEC 62439-6	100ms
Media Redundancy Protocol (MRP) [45]	IEC 62439-2	30 ms até 500ms
Beacon Redundancy Protocol (BRP) [46]	IEC 62439-5	>4,8ms
Parallel Redundancy Protocol (PRP) [42]	IEC 62439-3	0s
High-availability Seamless Redundancy (HSR) [42]	IEC 62439-3	0s

transmitida, com o objetivo de fornecer a recuperação contínua em caso de uma falha única de um enlace entre *switches* da rede, ou de algum *switch* [42].

O PRP precisa de duas redes iguais, ou muito similares, funcionando ao mesmo tempo. Assim, cada IED da rede tem duas portas ethernet, que usam o mesmo endereço MAC, conectadas em duas LANs distintas. Isso significa que as redes devem ser completamente separadas e falhas em uma rede não podem interferir na outra. O IED de origem envia duas cópias do pacote, simultaneamente, uma para cada porta. Assim os dois pacotes passam por sua respectiva LAN até acharem o destino. O destinatário aceita o primeiro pacote e descarta o segundo, com base em um campo de sequência que é colocado em cada pacote quando enviado. Como benefício, esse método garante tempo de recuperação de 0 s se houver falha em apenas uma das redes. Porém, duplica-se o número de *switches* para sua implementação, já que são necessárias duas redes iguais para o funcionamento do método. Além disso, é necessário que os IEDs implementem esse protocolo. Ressalta-se que o processamento de cada IED receptor é duplicado, já que recebe quase sempre dois pacotes tendo que aceitar um e rejeitar o outro. Além disso, gasta-se mais energia para realizar o encaminhamento do mesmo pacote duas vezes, ao invés de re-rotear o pacote pela rota alternativa em caso de falha.

O HSR é uma evolução do PRP em que o mesmo mecanismo é aplicado, porém apenas em redes de IEDs em anel, não havendo a necessidade de duplicar a rede e seus componentes. Nesse método, os IEDs estão ligados por duas portas Ethernet, formando um anel. O IED de origem envia o mesmo pacote para ambas as portas, ou seja, um pacote para cada lado do anel. O destino deverá receber, em uma situação livre de falhas, dois pacotes idênticos, descartando o segundo assim como o PRP. Como benefício, esse método garante tempo de recuperação de 0 s se houver apenas uma falha no anel de IEDs. Além disso, reduz a quantidade de *switches* necessários na sua implantação já que os IEDs

que implementam esse protocolo são capazes de enviar pacotes de uma porta para a outra. Porém, esse método insere um atraso no encaminhamento, já que os pacotes passam por cada IED em um anel HSR, o que também insere uma limitação quanto a quantidade de IEDs na rede. Além disso, nesse cenário, apenas cerca de metade da largura de banda fica disponível. Isto porque todos os pacotes são enviados duas vezes sobre a mesma rede, mesmo quando não há nenhuma falha, o que pode sobrecarregar a rede. Outro ponto importante é que os *switches* embarcados em IEDs, necessários para a implementação do HSR, são menos eficientes que os *switches* padrão, apresentando maiores atrasos no encaminhamento de pacotes.

Ressalta-se que, embora atraente, a Norma IEC 61850 ainda apresenta alguns problemas relacionados à comunicação que afetam seu desempenho, como será apresentado no próximo capítulo.

Capítulo 3

O uso dos recursos de redes no contexto IEC 61850

Para garantir o bom desempenho das redes IEC 61850, alguns aspectos devem ser observados. O uso correto dos recursos de rede, o alto desempenho no encaminhamento de pacotes, a qualidade de serviço e a integração são algumas das palavras chave nesse contexto.

Mesmo com toda a padronização e todos os estudos para construir a melhor modelagem possível para os sistemas de proteção e controle, as soluções atuais ainda não exploram todas as vantagens e recomendações da norma, ou não usam bem os recursos de comunicação de dados disponíveis. Além disso, a norma ainda apresenta desafios no que diz respeito à comunicação na rede e ao desempenho desta, visto que, algumas recomendações podem ser melhoradas para alcançar um maior desempenho de rede. As seções que seguem tratam alguns desses aspectos.

3.1 Diferenciação do tráfego

Como foi descrito na seção anterior, na Tabela 2.3, o valor *default* de prioridade para os pacotes GOOSE e SV é definido como quatro para ambas as mensagens [13]. Assim, apesar de requisitos temporais distintos, essas mensagens têm por padronização a mesma priorização.

Na prática, as mensagens GOOSE, por exemplo, têm restrições temporais distintas. Para a mensagem tipo 1A (*trip*) na classe P2 e P3, que são aplicadas ao nível de vão de transmissão, essa restrição é de 3ms. Em contrapartida, o tipo 1B na classe P1, que é aplicada ao nível de vão de distribuição, tem como restrição 100ms. Ambos os tipos

são mensagens GOOSE e teriam a mesma prioridade na rede segundo a norma. Além disso, a mensagem SV tipo 4 também receberia a mesma priorização, conforme descrito na Tabela 2.1.

Todas as outras mensagens recebem como *default* na norma o valor zero para priorização. Com isso, apesar das oito possibilidades de priorização do padrão 802.1p, de zero a sete, apenas dois são usados.

Uma forma de aumentar a qualidade de serviço dessa rede, seria redistribuir a prioridade de acordo com a aplicação de cada mensagem. Esse assunto será tratado na Seção 5.4.

Até o momento, nenhum trabalho relacionado a esse aspecto foi encontrado.

3.2 Métodos de Recuperação de Falha

Como foi visto no capítulo anterior, os métodos HSR e PRP, que são intitulados como protocolos de recuperação de falha, na verdade não restabelecem caminhos e nem recuperam a rede em caso de falha, apenas duplicam os pacotes enviados na tentativa de garantir a entrega destes. É por esse motivo que admite-se que esses protocolos possuem tempo de recuperação 0s.

Para esses métodos funcionarem como o esperado, algumas premissas devem ser atendidas, como por exemplo:

- A topologia dos IEDs precisa estar em anel (HSR), ou precisa ser duplicada colocando uma segunda rede paralela à primeira (PRP).
- A falha só pode ocorrer de um lado do anel, ou em uma das redes paralelas, não havendo garantia em caso de falhas duplas.
- O processamento dos IEDs é duplicado já que recebem dois pacotes, tendo que aceitar um e rejeitar o segundo com base no campo de sequência que é colocado no pacote.
- A banda é diminuída pela metade já que os pacotes são duplicados.
- A rede pode sofrer uma sobrecarga com os pacotes de alta prioridade duplicados.

Como esses métodos só podem ser aplicados aos cenários e às topologias descritas, o projeto da rede fica engessado, além de não se alcançar o desempenho ideal nas redes.

Apesar disso, são soluções usadas na prática, já que no caso de uma falha em um dos canais de comunicação, nenhum pacote se perde, pois uma cópia seguiu por outro caminho. Além disso, a norma IEC 61850, em sua segunda edição, padronizou o uso do PRP e do HSR incluindo campos nos pacotes GOOSE e SV.

Existem alguns trabalhos que discutem o uso do PRP e do HSR em redes de subestação e suas premissas para implantação.

Tan e Luan discutem projetos de arquitetura para redes de subestação e como o PRP e o HSR, por exemplo, são aplicados nessas arquiteturas [47]. O autor ressalta a necessidade de implementação dos protocolos nos IEDs, ou em qualquer elemento que esteja na rede, incluindo controladores e *Merging Units*[1].

Ainda nesse sentido, Antonova et al. apresentam diferentes topologias e aplicações para automação de subestações utilizando o PRP e o HSR de forma que os requisitos rígidos de tempo da norma sejam atendidos [48]. Porém, o artigo também aponta a necessidade de duplicação da rede e/ou do pacote de forma que os IEDs recebam o primeiro pacote e descartem o segundo, o que também impõe um maior processamento nos IEDs.

Yong-hui et al. comentam o uso do PRP para melhorar a confiabilidade na rede e ressaltam as premissas para implementação do protocolo como a necessidade de duas redes paralelas [18].

Para melhorar o desempenho de redes que implementam o PRP e/ou o HSR, Goraj e Harada propõe soluções hibridas com PRP e HSR. Os autores ressaltam que o uso do protocolo HSR introduz uma sobrecarga significativa para todos os nós da rede e apresentam uma solução hibrida, com o uso do PRP nos IEDs e do HSR na rede como uma alternativa para tentar melhorar o desempenho da rede [49].

Souza Jr. et al. propõem uma solução para cenários nos quais os protocolos HSR e PRP não podem ser aplicados devido a restrições técnicas como quantidade de IEDs e/ou disponibilidade de *hardwares* compatíveis com estas tecnologias [50]. O artigo propõe o uso do protocolo MRP em uma topologia em anel, combinada com ilhas *Rapid Spanning Tree Protocol* (RSTP) [50].

Os artigos buscaram soluções para minimizar os problemas do PRP e do HSR. Para isso, escolhem um dos dois métodos como a melhor solução, propõem implementações híbridas ou recorrem a outros protocolos que apresentam um tempo de recuperação muito

[1] A Merging Unit(MU) é um elemento que recebe os valores amostrados dos transformadores de potencial e corrente e envia para a rede em forma de um pacote SV.

maior, o que não é indicado para redes IEC 61850. Abordagens propondo um novo método, para redes de camada de enlace, não foram encontradas.

O ideal seria um método que conseguisse um tempo de recomposição da rede de 0s ou próximo de 0s, sem sobrecarregar a rede nem seus equipamentos, sem a necessidade de uma topologia específica e sem a necessidade de implementação nos dispositivos finais. Assim, o método poderia ser aplicado a qualquer cenário, aumentando o desempenho da rede sem sobrecarregar a mesma. Esse é um dos assuntos tratados por essa dissertação, descrito na Seção 6.4.1.

3.3 Comunicação multicast na camada de enlace

A questão mais importante, segundo a norma [37], no planejamento da comunicação em uma subestação é a correta atribuição dos nós lógicos e o arranjo da rede de comunicação em si. Para isso, a norma define que a comunicação deve ser feita via *multicast*, em um modelo publicador/assinante para mensagens GOOSE e SV. Nesse modelo, o IED publicador envia as mensagens para a rede e os assinantes daquele grupo específico aceitam os pacotes. Assim, apenas os IEDs daquele determinado grupo *multicast* têm acesso à mensagem.

Switches de camada de enlace típicos, por *default*, quando recebem um pacote endereçado a um destino MAC *multicast*, enviam este pacote por todas as portas. É esse método que garante que o pacote vai chegar ao destino qualquer que seja o destinatário, pois, esses *switches* tratam o tráfego *multicast* como desconhecido ou *broadcast*. Porém, esse método consome banda no enlace, aumenta o atraso nos *switches* e introduz uma sobrecarga significativa nos IEDs [11, 9].

Para contornar este problema, os fluxos de dados *multicast* devem ser restritos apenas ao grupo em questão, sendo estabelecido um caminho para que o pacote siga. Assim, os pacotes seguem um caminho até os IEDs assinantes do grupo, sem serem enviados por toda a rede, evitando a perda de desempenho. Esse caminho é intitulado árvore *multicast*.

As árvores *multicast*, podem ser configuradas de forma estática ou dinâmica:

- **Estática:** cada dispositivo na rede é configurado manualmente com o endereço do grupo *multicast*. Isto significa que o endereço *multicast* é mapeado para a porta, ou mais de uma se for o caso, por onde os pacotes daquele grupo específico devem ser encaminhados. Se o estado da rede muda, não existe uma atualização automática

na tabela de encaminhamento. Qualquer configuração ou alteração deve ser feita manualmente, por esse motivo não é muito utilizada.

- **Dinâmica:** o gerenciamento do *multicast* é feito com protocolos específicos, que se encarregam do tráfego *multicast* encaminhando-o para os dispositivos que manifestaram interesse em recebê-lo. A árvore *multicast* é construída dinamicamente, e, em caso de atualização na rede, o algoritmo deve recalcular a árvore. Nesse caso, os IEDs deveriam implementar protocolos para serem capazes de manifestar interesse em sair, ou entrar em um grupo.

Existem diversos recursos e protocolos de camada 3 para a configuração dinâmica da árvore *multicast*. Porém, para o caso de mensagens da camada de enlace, como GOOSE e SV, existem poucas alternativas. Para camada de enlace, esses protocolos são também conhecidos como métodos para restrição de tráfego *multicast*, ou filtragem multicast. Atualmente, o protocolo *multicast* de camada 2 usado em redes IEC 61850 é o GMRP (*GARP Multicast Registration Protocol*) [51] [18] [52]. Além do GMRP, tem-se também o *Internet Group Management Protocol* (IGMP) *snooping*, que apesar de funcionar também em camada de enlace, precisa de recursos da camada de rede[2]. Com isso, não pode ser aplicado em redes puramente da camada de enlace.

O GMRP [51] é um protocolo *multicast* de camada de enlace completo que traz diversas vantagens como escalabilidade, alta velocidade de transmissão e alta confiabilidade. O GMRP, padronizado pelo IEEE 802.1D [51], é usado para manter o registro de informações *multicast* nos *switches*. Seu principio básico de funcionamento é baseado em mensagens intituladas *join* e *leave*, as quais o *host* deve ser capaz de enviar quando quiser entrar ou sair de um grupo.

O *switch* registra a porta pela qual recebeu a mensagem *join* e associa essa porta ao grupo *multicast* da mensagem, montando a sua tabela de encaminhamento. Em seguida, o switch envia a mensagem *join* via *broadcast* para toda a rede, garantindo que o novo membro do grupo *multicast* passe a ser conhecido por toda a rede. Todos os *switches* que suportam GMRP podem receber essa informação de outros *switches* para atualizar seu registro local. Os *switches* não precisam de nenhuma configuração, porém devem suportar o protocolo GMRP [18]. Com a tabela configurada e atualizada, quando o publicador envia mensagens *multicast*, o *switch* envia essas mensagens apenas para as portas necessárias para que a mensagem chegue a todos os membros do grupo [18].

[2]O IGMP snooping monitora o tráfego IGMP entre *hosts* e roteadores. Através do tráfego da camada de rede, o *switch* é capaz de aprender por qual porta deverá enviar o tráfego multicast.

Dessa forma, o GMRP é fortemente indicado para comunicação de mensagens GOOSE e SV em subestações [18, 10].

Existem trabalhos que mostram que o uso de *multicast* em redes de subestação simplifica o controle do tráfego e diminui os atrasos [8, 7, 9], além de reduzir o volume de dados recebidos pelos dispositivos [10].

Yong-hui et al. mostram as vantagens do uso do GMRP em redes de subestação. O trabalho dá ênfase às necessidades de estudo do barramento de processo, e, com isso, do uso das mensagens SV e sua integração em uma mesma rede com as mensagens GOOSE [18]. Os autores exploram as vantagens do GMRP com aplicações em laboratório e aplicações práticas.

Nesse mesmo contexto, XiCai et al. apresentam um exemplo do uso do GMRP em subestações chinesas para reduzir o volume de dados recebidos por dispositivos [10]. Os autores mostram que a prática nas subestações chinesas tem sido a aplicação do GMRP para o tráfego na rede e do IEEE 1588 para sincronização.

Muitas vezes, o GMRP é usado em conjunto com as VLANs (*Virtual Local Area Network*) para restringir o tráfego [8].

Ingram et al. combinam o uso de VLANs e filtro *multicast* para separar o tráfego por aplicação e por grupos de interesse, de forma que o tráfego *multicast* fique restrito apenas a uma determinada VLAN [8]. Com objetivo de simplificar a análise do tráfego por engenheiros, o artigo propõe um método para mapear as faixas de endereços *multicast* de acordo com informações específicas, como nível de tensão da linha de transmissão ou a qual vão da subestação o IED pertence.

Sivanthi e Goerlitz propõem uma abordagem sistemática para melhorar o uso de filtros *multicast* e VLANs agrupando caminhos comuns. Os autores apresentam um algoritmo que agrupa destinos que seguem o mesmo caminho na rede, ou seja, se dois grupos *multicast* tiverem a mesma árvore, o algoritmo colocará os dois no mesmo grupo *multicast* [7]. O algoritmo é usado tanto para VLANs quanto para filtros *multicast*. Nesse sentido, os autores ressaltam que a segmentação de tráfego baseada em filtros *multicast* pode ser usada por IEDs que não possuam a capacidade de enviar *frames* com *tags*. Já a segmentação baseada em VLANs deve ser usada em IEDs com essa capacidade, ou por *switches* que acrescentem essa marcação de acordo com a porta [7].

As propostas de uso do GMRP em redes de subestação, demandam que os fabricantes de IEDs implementem o protocolo em seus dispositivos [9, 18], o que aumenta o

processamento nos IEDs e também o custo do dispositivo.

Contudo, os IEDs, até o momento, não possuem a capacidade de implementar o protocolo e enviar mensagens *join* e *leave* para a rede. Com isso, parte da configuração tem que ser feita manualmente, na tabela *multicast* estática dos *switches*. Com isso, cada *switch* interligado ao IED passa a ter na sua tabela as informações de grupo *multicast* e porta configuradas manualmente, assim, quando o *switch* envia a mensagem de atualização para rede as informações do grupo *multicast* são atualizadas entre os *switches*. Com isso, a configuração da rede se torna trabalhosa e a maioria dos fabricantes recomenda a utilização de VLANs para limitar os domínios de *broadcast* e, assim, minimizar a sobrecarga da rede [53]. Portanto, apesar de estudos [11, 9] mostrarem que o uso de *multicast* ao invés de *broadcast* em VLANs evitam a sobrecarga, o *multicast* acaba não sendo utilizado na prática, reduzindo o desempenho da rede.

Outros trabalhos abordam o *multicast*, porém numa abordagem de camada de rede, para uso em outras aplicações. Seewald usa IP *multicast* para o transporte de dados entre PMUs (*Phasor Measurement Unit*) [54]. A arquitetura proposta usa o IP *Multicast* PIM-SSM (*Protocol Independent Multicast - Source Specific Multicast*) com o objetivo de prover um caminho de entrega ótimo para tráfego de baixa latência. Essa abordagem, porém, é restrita à camada de rede, não sendo adequada para mensagens GOOSE e SV. Cabe observar que o SMARTFlow também trata esse tipo de tráfego, sendo capaz de configurar reativamente ou pró-ativamente esses grupos de camada três.

O principal objetivo em [55] é a transmissão de mensagens GOOSE via modelo publicador/assinante em redes locais *Wireless*. O autor avalia a proposta usando o simulador OPNET e apresenta resultados dentro dos requisitos temporais da norma para mensagens *multicast* em redes locais sem fio. O SMARTFlow, além de ser implementado em redes cabeadas, pode também ser implementado em redes sem fio, pois existe suporte para embarcar o OpenFlow em pontos de acesso.

Um dos objetivos da proposta dessa dissertação, como será visto no Capítulo 5, é a criação automática de árvores *multicast* de camada de enlace para o envio de mensagens GOOSE e SV, assim como qualquer tráfego MAC *multicast*. A proposta é transparente para os IEDs, que não têm a necessidade de implementar diferentes protocolos.

Capítulo 4

O Arcabouço OpenFlow

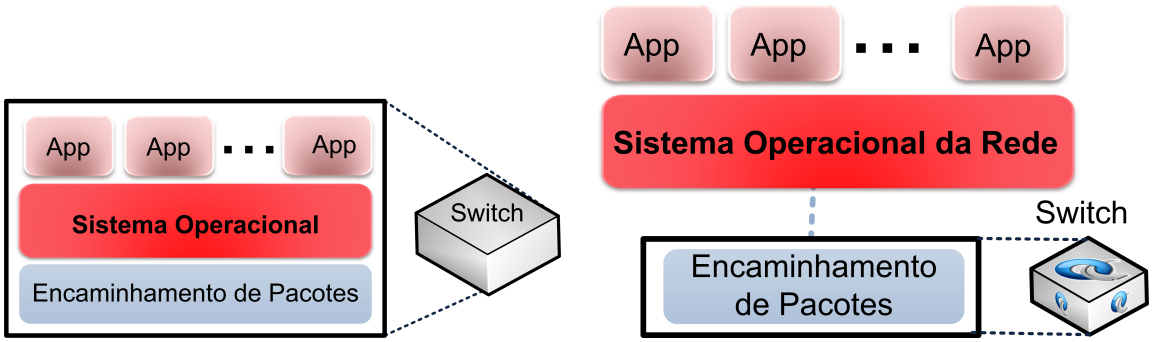
Muitos pesquisadores consideram que a infraestrutura de rede atual está "ossificada", não podendo ser modificada [14, 56]. Isso ocorre, pois existem muitos obstáculos para a implantação de inovações onde já existe uma base extensa, já instalada, de equipamentos e protocolos em produção. Experimentar novas tecnologias em redes em produção não é uma prática comum. Além disso, a estrutura dos equipamentos tradicionais é fechada para inovações [56]. Mckeown et al. [14] afirmam que não há praticamente nenhuma forma de se testar novos protocolos de rede, por exemplo, os novos protocolos de roteamento ou, alternativas ao IP, em ambientes suficientemente realistas para ganhar a confiança necessária para sua implantação generalizada. O resultado é que a maioria das novas ideias da comunidade de pesquisa não é devidamente testada e, por essa razão, acaba não sendo implementada nas redes em produção.

Nas redes atuais, cada vez mais, são demandados novos requisitos de rede, como o isolamento entre redes virtuais, o provimento de qualidade de serviço, a mobilidade, etc. A comunidade de redes vem atendendo bem a essas crescentes demandas, porém, com soluções individuais, que não lidam com a arquitetura como um todo [57]. Tem-se soluções bem definidas para casos específicos e isolados, porém alcançar a integração ainda é um desafio nas redes atuais.

A rede definida por *software* (*Software Defined Network* (SDN)) é uma proposta emergente que objetiva flexibilizar o provimento de novas funcionalidades na rede de forma estruturada, através da criação de um plano de controle via software que seja fisicamente separado do plano de dados. Com isso, o plano de dados reside nos equipamentos de rede, sejam eles *switches*, pontos de acesso ou roteadores, independente de fornecedor, com uma interface uniforme. O plano de controle fica externo ao equipamento de rede, podendo ser

implementado de forma centralizada ou distribuída. Nas aplicações mais comuns de SDN, o plano de controle é abrigado em uma máquina física ou virtual, de forma que tem-se um novo elemento na rede, responsável pela tomada de decisão e confecção das tabelas de encaminhamento. Dentro desse elemento reside o sistema operacional de rede que é o software que oferece uma interface de programação simples e uma visão topológica da rede, o que simplifica as tarefas de engenheiros de rede [58].

A Figura 4.1 compara a arquitetura programável do OpenFlow, ilustrada na Figura 4.1(b), com a arquitetura tradicional, onde o plano de controle e o plano de dados residem no mesmo equipamento, conforme Figura 4.1(a).



(a) Dispositivos de rede tradicionais. (b) Modelo Programável [56]

Figura 4.1: Comparação entre o modelo tradicional, onde o plano de controle reside no próprio equipamento, e o modelo programável do OpenFlow [56]

Com essa estrutura a implementação de aplicações de rede se torna mais fácil, pois é feita com base em uma visão central da rede e também permite uma visão de alto nível [58]. Desse modo, mantém-se o alto desempenho no encaminhamento de pacotes em *hardware* aliado à flexibilidade de se inserir, remover e especializar aplicações em *software* [59].

De acordo com [60], a indústria está adotando as SDNs em grande escala. Embora seja um método relativamente novo, analistas da indústria previram que o mercado de SDNs vai ultrapassar 200 milhões de dólares em 2013, e que além disso, crescem rapidamente para US $2 bilhões até 2016. Também de acordo com [60], as maiores operadoras de rede do mundo, como por exemplo, Google e a NTT Communications, já utilizam as SDNs. Atualmente existem mais de 60 produtos disponíveis no mercado que facilitam a transição para essa nova tecnologia. Novos produtos e implementações são anunciados a cada dia, o que indica que as organizações perceberam o profundo impacto que as SDNs podem trazer para os seus negócios.

Algumas das maiores empresas de redes, incluindo Deutsche Telekom, Facebook, Google, Microsoft, Verizon, e Yahoo!, criaram a *Open Networking Foundation* (ONF) [61] para padronizar e promover as SDNs e protocolos incluindo o OpenFlow. A ONF é uma fundação sem fins lucrativos, que possui quase 100 membros, entre estes incluem: Alcatel-Lucent, Cisco, Dell, Extreme, HP, Huawei, IBM, Juniper, NEC, Nokia Siemens. Vários fornecedores anunciaram produtos, tais como *switches* e controladores, e muitos mais são esperados no próximo meses [60, 61].

A plataforma OpenFlow, proposta pela Universidade de Stanford [14], é o principal exemplo da aplicação do conceito de SDNs. No OpenFlow, o plano de controle funciona em um servidor, capaz de se comunicar com todos os switches OpenFlow da rede, obtendo informações de estado e enviando comandos de configuração e encaminhamento.

Com isso a demanda pela validação de novas propostas de arquiteturas e protocolos de rede, incluindo as abordagens *clean slate*[1], sobre equipamentos comerciais em uma rede em produção poderia ser atendida.

As atividades de gerência e configuração da rede são feitas em cima da abstração da rede. Com isso, ao invés de se configurar cada dispositivo separadamente, os administradores de rede precisam somente manipular o seu mapa lógico. A arquitetura também suporta uma série de interfaces de programação de aplicativos (*Application Programming Interface*s (APIs)), que permitem a implementação de serviços de rede comuns como roteamento, *multicast*, políticas de acesso, engenharia de tráfego, *Quality of Service* (QoS), etc.

O OpenFlow tem como objetivo ser flexível para atender os seguintes requisitos [14]:

- Possibilidade de uso em implementação de baixo custo e de alto desempenho;
- Capacidade de suportar uma ampla gama de pesquisas científicas;
- Garantia de isolamento entre o tráfego experimental e o tráfego de produção;
- Consistência com a necessidade dos fabricantes não exporem o projeto de suas plataformas.

Para isso, utilizam-se as tabelas já existentes nos *switches* atuais. Define-se um protocolo padrão para determinar as ações de encaminhamento de pacotes em dispositivos

[1] A abordagem *Clean slate* visa substituir toda a arquitetura atual por uma nova, totalmente reconstruída, tendo como principal objetivo direcionar como será efetuado o desenho da nova internet [62].

de rede. Portanto, um *switch* OpenFlow precisa apresentar a capacidade de estabelecer um canal seguro com o controlador, de trocar dados e aceitar configurações por meio do protocolo OpenFlow e, ainda, de armazenar as configurações recebidas na(s) tabela(s) de fluxos [14]. A Figura 4.2 ilustra um *switch* OpenFlow, cujas partes representadas são descritas nas seções que seguem.

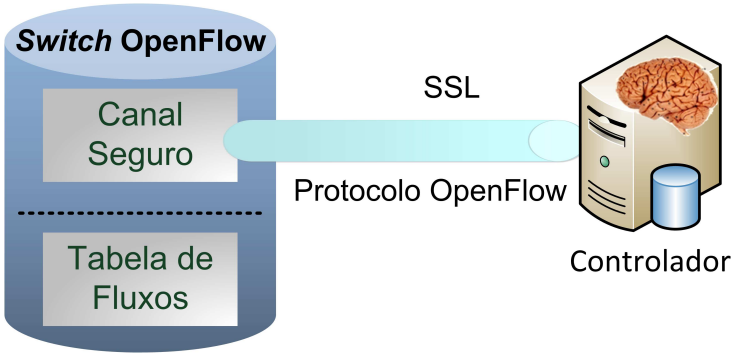


Figura 4.2: *Switch* OpenFlow. A Tabela de fluxos é controlada por um controlador remoto via o canal seguro [14]

No OpenFlow, um administrador pode controlar seus fluxos, escolhendo a rota que seus pacotes irão percorrer e como os pacotes do seu fluxo serão processados.

O canal seguro (*secure channel*) conecta o *switch* ao controlador da rede, permitindo que sejam trocados comandos e pacotes entre esses elementos [14]. Para que a rede não sofra ataques de elementos mal intencionados, o canal seguro garante confiabilidade na troca de informações entre o *switch* e o controlador. A interface de acesso recomendada é o protocolo *Secure Socket Layer* (SSL), embora também seja possível fazer a conexão via *Transmission Control Protocol* (TCP) nas implementações atuais [63].

Para fornecer uma forma aberta e padronizada para essa comunicação entre controlador e *switch*, foi desenvolvido o Protocolo OpenFlow. Ele especifica uma interface padronizada onde os administradores da rede podem configurar a tabela de fluxos, obter estatísticas da rede, entre outros [14]. Assim, mensagens são trocadas entre os equipamentos de rede e o controlador. O protocolo OpenFlow permite o uso de três tipos de mensagens [63, 64, 65, 66, 67]:

- Simétricas: são mensagens geradas, sem solicitação, do controlador para o *switch* e vice-versa. Exemplos são as mensagens `hello` e `echo`. A primeira é trocada entre o controlador e o *switch* na inicialização da rede, a segunda é usada principalmente para verificar se a conexão entre o *switch* e o controlador continua ativa e para identificação de latência e banda.

- Assíncronas: são mensagens enviadas pelo *switch* sem a solicitação do controlador. Informam eventos na rede, chegada de pacotes, erros e mudanças no estado do *switch*. Um exemplo desse tipo de mensagem é a `Packet In` que tem como objetivo notificar a chegada de um fluxo não classificado no *switch*.

- *Controller-to-switch*: são mensagens iniciadas pelo controlador usadas para gerenciar diretamente ou inspecionar o estado do *switch*. Estas mensagens permitem que o controlador configure o *switch*, modifique estados e entradas de fluxo, dentre outras características.

4.1 O *Switch* OpenFlow

A ideia por trás da implementação do OpenFlow em *switches* comerciais é simples. A maioria dos *switches* Ethernet e roteadores modernos possuem tabelas de encaminhamento tipicamente construídas a partir de *Ternary Content Addressable Memory*s (TCAMs), que são utilizadas para implementar diferentes funcionalidades, como Qualidade de Serviço (QoS), *Firewalls*, *Network Address Translation* (NAT) e coleta de estatísticas. Assim, a proposta do OpenFlow tem como ideia básica a instalação das tabelas de fluxo nessa área dos *switches* e roteadores. Contudo, como o *hardware* varia de fabricante para fabricante, cabe ao fabricante decidir qual a melhor forma de implementar todas as funções do OpenFlow sobre o hardware existente, assim como a disponibilização de um *firmware* compatível.

Ressalta-se que, uma interface bem definida entre o *switch* e o controlador incetiva a inovação e a concorrência. Isso ocorre, pois permite que diversos fabricantes desenvolvam seus *switches* bastando para isso obedecer as regras definidas pela interface. Além disso, permite que vários controladores sejam desenvolvidos desde que saibam como usar o protocolo.

4.1.1 Tabela de Fluxos OpenFlow

A tabela de fluxos é uma lista que possui um conjunto de ações para cada entrada de fluxo definida, informando ao *switch* como processar os pacotes que chegam. A versão 1.0.0 do OpenFlow [63], lançada em 2009, possui uma tabela única, e cada entrada na tabela de fluxos possui 3 campos: os campos de cabeçalho, conhecido como regra, os contadores e as ações [14], como mostrado na Figura 4.3.

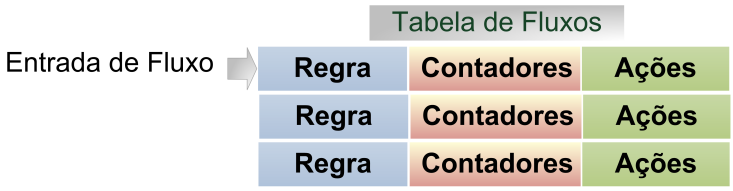


Figura 4.3: Tabela de Fluxos. Uma entrada de fluxo, no OpenFlow 1.0.0 consiste em regras, contadores e ações [63]

A regra é formada com base na definição do valor de um ou mais campos do cabeçalho do pacote. No OpenFlow 1.0.0 são disponibilizados 12 campos para serem usados no cabeçalho, conforme indicado na Figura 4.4. Desta forma, o fluxo pode ser identificado através da combinação de qualquer um desses campos, inclusive, com mais de um. É importante observar que o *switch* OpenFlow opera usando campos de diferentes camadas, o que garante uma alta flexibilidade no encaminhamento de pacotes.

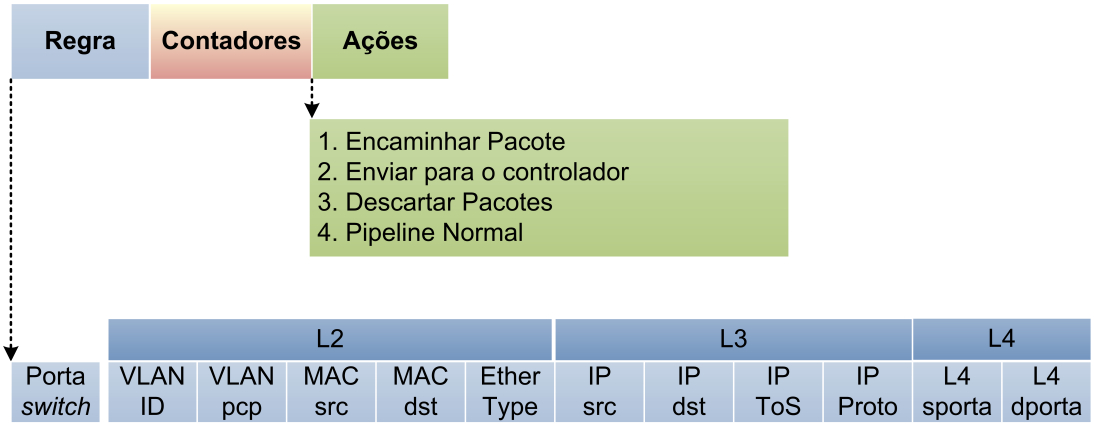


Figura 4.4: Exemplo de entrada da tabela de fluxos do OpenFlow versão 1.0.0 [56, 63]

Os contadores, que são as estatísticas do fluxo, mantém, entre outros, um registro do número de pacotes e *bytes* referentes a cada fluxo, o tempo decorrido desde a última vez que um pacote desse fluxo foi identificado pelo *switch* e o tempo desde a instalação do fluxo. O contador não possui indicador de *overflow*, com isso a contagem é reiniciada automaticamente ao alcançar o valor máximo [63].

Associa-se à regra um conjunto de ações a serem tomadas, que definem o modo como os pacotes devem ser processados e para onde devem ser encaminhados. As ações podem ser obrigatórias ou opcionais. Como ilustrado na Figura 4.4, as ações básicas que os *switches* OpenFlow na versão 1.0.0 [63] devem suportar, são as seguintes [14]:

1. Encaminhar os pacotes do fluxo para uma ou mais portas.

2. Encapsular os pacotes e encaminhá-los para um controlador, utilizando o canal seguro de comunicação. Essa ação acontece sempre que um pacote que chega ao *switch* não é compatível com nenhuma das regras de fluxo instaladas. Tipicamente, essa ação é executada no momento do envio do primeiro pacote de um novo fluxo, para que o controlador possa decidir, caso seja conveniente, qual entrada de fluxo deve ser adicionada à tabela.

3. Descartar os pacotes deste fluxo. Essa ação pode ser utilizada em aplicações de segurança para impedir, por exemplo, ataques de negação de serviço.

4. Encaminhar os pacotes do fluxo pelo pipeline normal do *switch*. Nessa ação um fluxo identificado como não sendo referente ao OpenFlow será processado normalmente nas camadas 2 e 3 do equipamento. Aplica-se apenas para *switches* OpenFlow híbridos, descritos na Seção 4.1.4.

O processamento dos fluxos nessa versão se dá conforme a Figura 4.5. Ao receber um pacote, o *switch* verifica o cabeçalho do pacote para ver se ele está definido em sua tabela de fluxos. Se estiver, o *switch* executa a ação especificada na tabela e atualiza os contadores referentes às estatísticas deste fluxo. Senão, encapsula o pacote e o envia para o controlador. O controlador então será responsável por adicionar uma entrada na tabela de fluxos do *switch*, identificando o fluxo referente a aquele pacote [68].

O OpenFlow também oferece suporte opcional ao *Spanning Tree* [51]. Caso o *switch* implemente passa a ser obrigatória em todas as portas físicas e opcional nas portas virtuais [63].

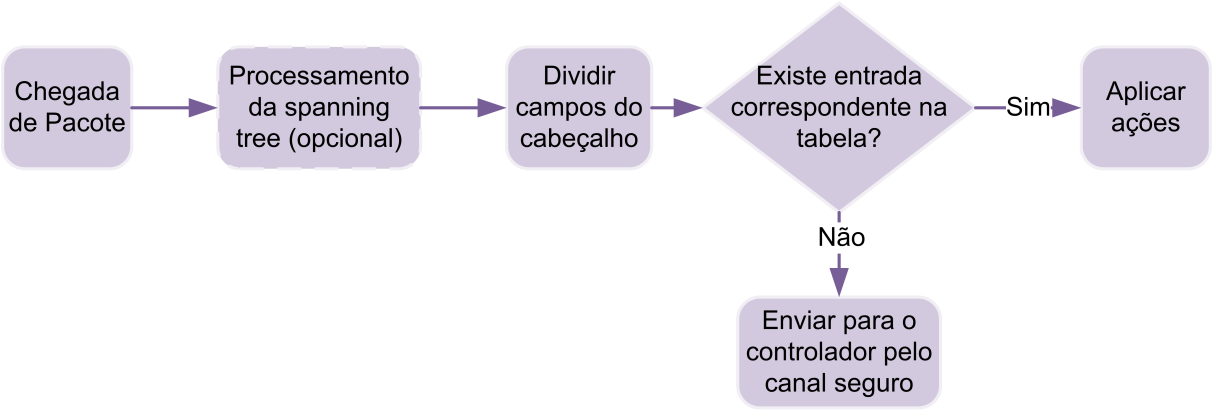


Figura 4.5: Fluxograma detalhando processamento dos fluxos através de um *switch* OpenFlow versão 1.0.0 [64]

Além disso, os pacotes são comparados com as entradas de fluxo com base em uma

priorização. Uma entrada que especifica uma correspondência exata é sempre a maior prioridade. Todas as entradas com campos coringa[2] têm uma prioridade associada. Se várias entradas possuírem a mesma prioridade, o *switch* é livre para escolher qualquer ordem.

Com as próximas versões do OpenFlow, disponíveis no site do ONF, novos conceitos foram introduzidos. A primeira versão, 0.2.0, foi lançada em 2008 [69] e desde então tem sofrido melhoras e extensões. A seguir são descritas as principais mudanças trazidas com as versões 1.1.0 [64] e 1.2 [65]. As versões 1.3 [66] e 1.4 [67] não são abordadas nesse trabalho.

4.1.2 Versão 1.1.0

A versão 1.1.0 [64], lançada em 2011, introduz o conceito de *pipeline* de tabelas com múltiplas tabelas de fluxo, assim como a inserção de uma tabela de grupos. A Figura 4.6 mostra os principais componentes do *switch* OpenFlow dessa versão.

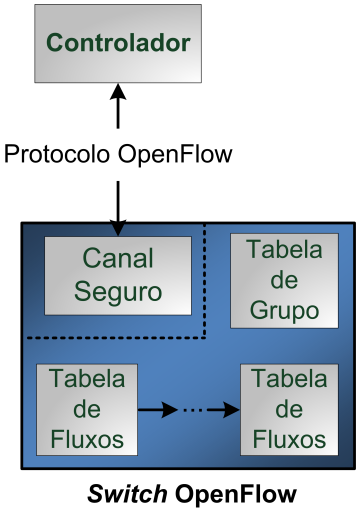


Figura 4.6: Componentes do *switch* OpenFlow versão 1.1.0 [64]

Essas alterações permitem uma definição ainda mais completa dos fluxos e do conjunto de ações associadas [64].

Além disso, nessa versão, são acrescidos novos campos no cabeçalho, conforme a Figura 4.7. O campo metadados é usado para troca de informações entre as tabelas. Os campos `Label` e `Classe` são campos para permitir a implementação do *Multiprotocol*

[2]Campos coringas, conhecidos como *wildcards*, não são especificados podendo conter qualquer valor. Dessa forma, o *switch* não se preocupa com o valor especificado no campo, apenas sabe se casou ou não.

Label Switching (MPLS). Os campos SCTP são as portas de origem e destino do *Stream Control Transmission Protocol* (SCTP)[3] como mais uma alternativa para implementação de protocolos da camada de transporte.

Figura 4.7: Campos do cabeçalho no OpenFlow 1.1.0 [64]

Com o suporte a múltiplas tabelas de fluxos, a especificação introduz o conceito de "instruções", com um significado mais complexo, onde um conjunto de ações pode ser modificado. Com isso, cada entrada de fluxo contém um conjunto de instruções que são executadas quando um pacote corresponde a essa entrada, conforme Figura 4.8.

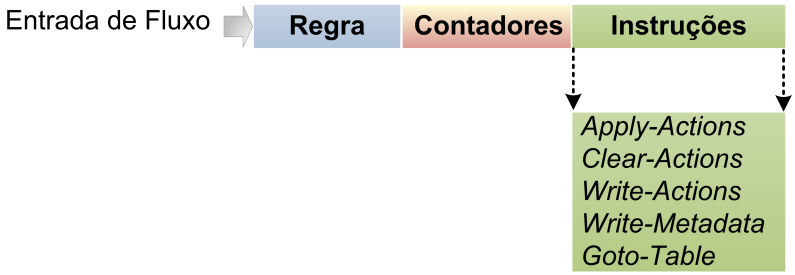


Figura 4.8: Uma entrada de fluxo, no OpenFlow 1.1.0 consiste em regras, estatísticas e instruções [64]

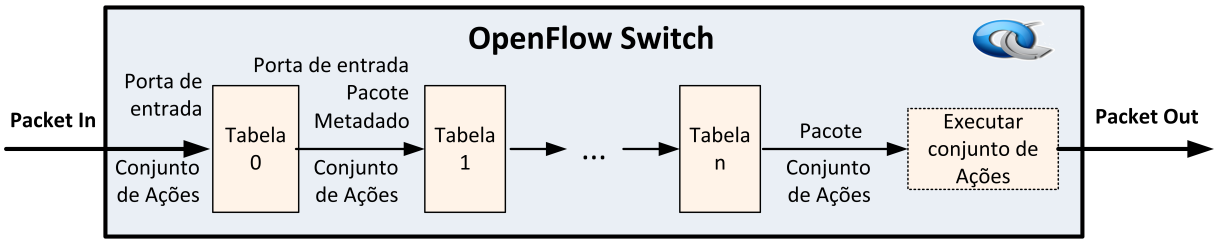
As instruções podem resultar em alterações no pacote, em um conjunto de ações e/ou um processamento no *pipeline* de tabelas. As instruções podem ser dos seguintes tipos [64]:

- `Apply-Actions <actions>` - Aplica as ações especificadas imediatamente sem modificar o *action set*[4]. Essas instruções podem ser usadas para modificar o pacote entre tabelas e realizar múltiplas ações do mesmo tipo.
- `Clear-Actions` - Apaga todas as ações no *action set* imediatamente.
- `Write-Actions <actions>` - Adiciona as ações especificadas no *action set*. Caso uma ação do mesmo tipo já exista no conjunto atual, essa instrução sobrescreve a ação.

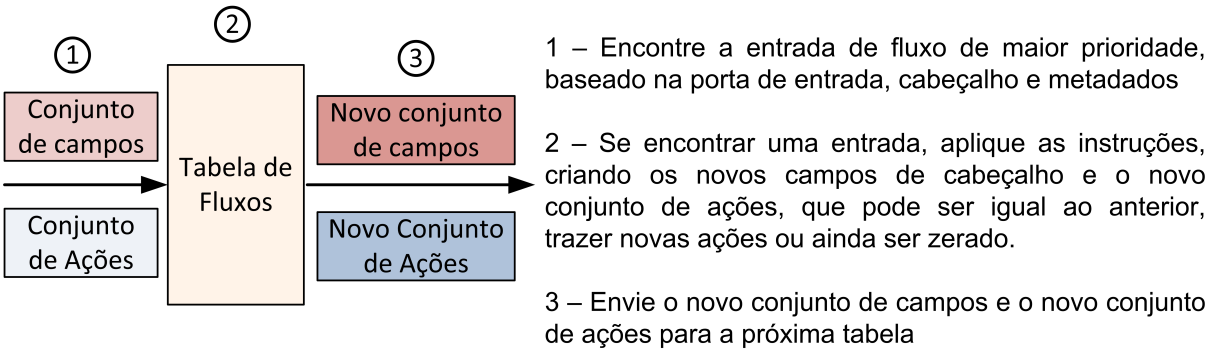
[3] O SCTP é um protocolo de transporte definido em 2000 pela RFC 2960 [70].
[4] O *action set* é o conjunto de ações a serem aplicadas a cada pacote.

- `Write-Metadata <metadata/mask>` - Escreve o valor do metadado no campo `Metadados`. A máscara especifica quais *bits* do metadado devem ser modificados.

- `Goto-Table <next-table-id>` - Indica qual a próxima tabela a ser consultada. O campo `table-id` é o identificador da tabela atual, o `next-table-id` precisa ser maior que o `table-id`. Os fluxos da última tabela não podem conter essa instrução, ou seja, se um conjunto de instruções não tem `Goto` significa que acabou o *pipeline* e que irá executar o `action set`.



(a) Exemplo de processamento do *pipeline* nos *switches* OpenFlow [64].



(b) Processamento do Pacote por tabela [64, 58]

Figura 4.9: Fluxos de pacote através do *switch* OpenFlow [64]

O processamento do pacote na entrada do *switch* também muda. Tem-se a disponibilidade de múltiplas tabelas interligadas. As tabelas de fluxo são numeradas de 0 a n. O processamento do *pipeline* nos *switches* OpenFlow define como os pacotes interagem com as tabelas de fluxo, conforme Figura 4.9 [64].

Dessa forma, quando o pacote chega, se houver uma regra correspondente na primeira tabela, o pacote é processado ali, senão, se houver um link para outra tabela, esse pacote será encaminhado para a segunda tabela. O processamento sempre começa pela tabela de número 0. As outras tabelas são usadas de acordo com o resultado da correspondência feita na tabela anterior [64]. Se o fluxo encontrar uma regra configurada, as instruções para aquele fluxo serão aplicadas. O detalhamento desse processo é ilustrado na Figura 4.10.

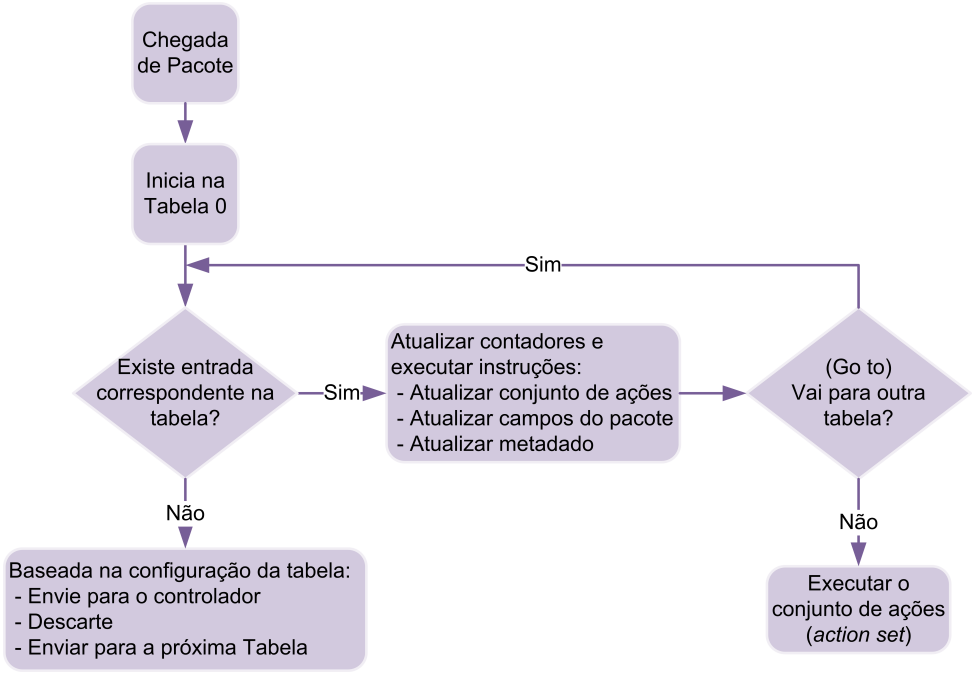


Figura 4.10: Fluxograma detalhando processamento dos fluxos através de um *switch* OpenFlow versão 1.1.0 [64]

Além disso, as entradas na tabela podem apontar para uma tabela de grupo. A tabela de grupo é projetada para realizar operações que são comuns a múltiplos fluxos. Dessa forma, ações são executadas para um grupo se for necessário, não apenas para um fluxo. Ações complexas de encaminhamento como múltiplos caminhos, agregação de enlaces e re-roteamento rápido são permitidas através desse mecanismo [69].

Uma tabela de grupos consiste em entradas de grupo ao invés de entradas de fluxo. A capacidade de um fluxo apontar para um grupo permite que o OpenFlow adicione novos métodos de encaminhamento, onde cada entrada de grupo é associada a quatro campos, conforme Figura 4.11 [64]. Onde [64]:

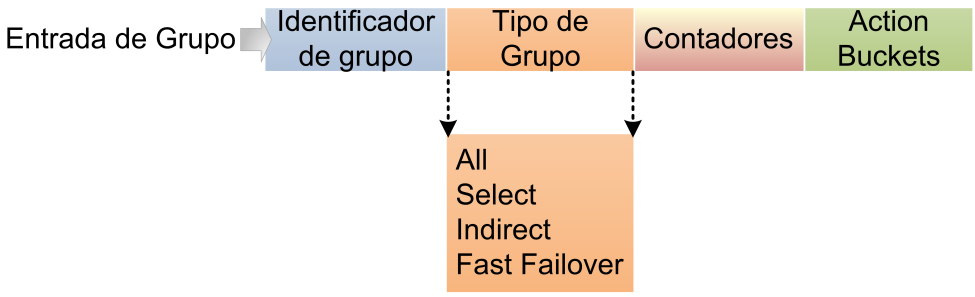


Figura 4.11: Tabela de Grupos. Uma entrada de grupo, no OpenFlow 1.1.0 consiste nos campos identificador de grupo, tipo de grupo, contadores, e *action buckets* [64]

- Identificador de grupo: número que identifica unicamente o grupo.
- Tipo de grupo: determina que tipo de ações esse grupo pode tomar. Podem ser de quatro tipos:
  - *all*: Executa todos os *buckets*[5] no grupo. Utilizado para o encaminhamento *multicast* e *broadcast*. O pacote é copiado para cada um dos *buckets*.
  - *select*: Executa um *bucket* no grupo. Os pacote são enviados para um único *backet* no grupo baseado em algum algoritmo como *hash* ou *round robin*.
  - *indirect*: Executa apenas um *bucket* específico no grupo. Permite que múltiplos fluxos ou grupos apontem para um único identificador de grupo. Um exemplo é o próximo salto no roteamento IP. Esse tipo de grupo é semelhante ao `all` porém com um único *bucket*.
  - *fast failover*: Executa o primeiro *bucket* ativo. Permite que o *switch* mude o encaminhamento sem a necessidade de comunicação com o controlador. Se não houver *bucket* ativo o pacote será descartado.
- Contadores: estatísticas atualizadas a cada vez que um pacote é processado na tabela de grupo.
- *Action buckets*: lista ordenada de *action buckets*, onde cada *action bucket* contém um conjunto de ações para serem executadas.

Com o conceito de *fast failover* é possível que a se recupere instantaneamente quando um caminho falhar, já que, no caso de falha no caminho principal o pacote, sem se comunicar com o controlador, já envia o pacote para o próximo caminho ativo.

4.1.3 Versão 1.2

Em dezembro de 2011, foi lançada a versão 1.2 [65] que inclui algumas características importantes como suporte ao IPv6. Os campos IPv6 adicionados foram:

- Endereço de origem
- Endereço de destino
- Número de protocolo

[5]Conjunto de ações para um grupo específico.

- Classe de tráfego
- ICMPv6 type
- ICMPv6 code
- Campos do cabeçalho do mecanismo de descoberta de vizinhos IPv6
- Labels de fluxos IPv6

Além disso, essa versão passa a suportar a estrutura *Type Length Value* (TLV) que traz maior flexibilidade para os protocolos atuais e futuros. A principal ideia com a estrutura TLV é que o pacote pode conter diversos tipos de dados. Isso acontece, pois os dados transmitidos codificados com o TLV são autoidentificáveis, pois possuem campos que definem o tipo dos dados, o comprimento e o valor dos dados.

Por fim, essa versão permite extensões experimentais por meio de campos dedicados determinados pela ONF.

4.1.4 Tipos de *switches* OpenFlow

É útil categorizar os *switches* em dois tipos: *Switch* OpenFlow dedicado e *Switch* OpenFlow híbrido:

1. ***Switch* OpenFlow dedicado:**
   Esse tipo de *switch*, exemplificado na Figura 4.12, não suporta o processo normal nas camada 2 e 3, é um *switch* puramente OpenFlow.

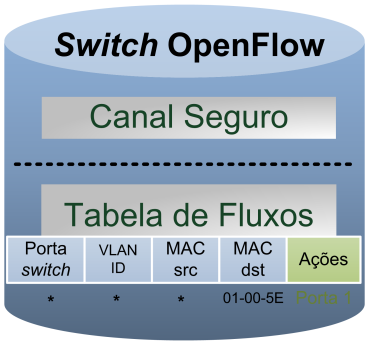


Figura 4.12: Switch OpenFlow Dedicado [56]

2. ***Switch* OpenFlow híbrido:**
   Consiste em *switches* ou roteadores com o OpenFlow habilitado como uma das

possíveis formas de encaminhamento. Portanto, as funcionalidades OpenFlow são adicionadas como um novo recurso, além do processamento normal de pacotes desses equipamentos. A Figura 4.13 mostra uma rede de *switches* comerciais com *switches* OpenFlow híbridos. Neste exemplo, todas as tabelas de fluxo são gerenciadas pelo mesmo controlador. O objetivo é permitir que os experimentos tenham lugar em redes de produção existentes junto com o tráfego regular e as aplicações. Portanto, para ganhar a confiança de administradores da rede, os *switches* híbridos devem isolar o tráfego experimental, processado pela tabela de fluxo, do tráfego de produção, processado como de costume, pelas camadas 2 e 3 do *switch* [14].

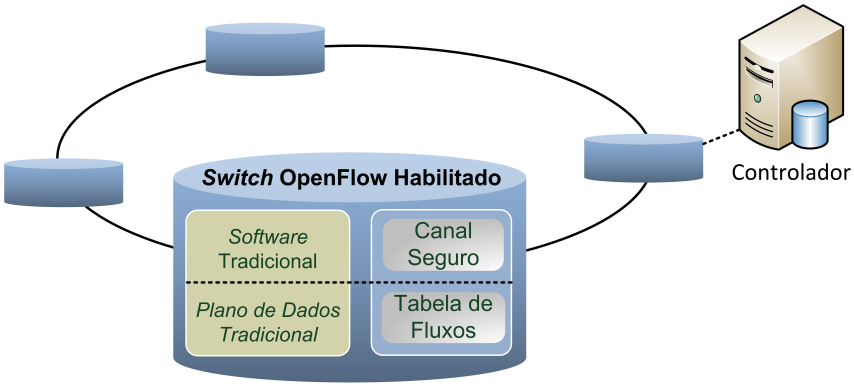


Figura 4.13: Switch OpenFlow Híbrido [56]

4.2 O Controlador OpenFlow

O controlador é uma plataforma para a construção de aplicações de gerenciamento e controle de rede, responsável por monitorar e tomar decisões [14]. Assim, diz-se que o controlador OpenFlow atua como um Sistema Operacional de rede [59], em analogia aos Sistemas Operacionais em computadores. Os sistemas operacionais oferecem uma interface de alto nível para as aplicações utilizarem os recursos de *hardware*, além de controlar a interação entre essas aplicações. Essa é a principal função do controlador OpenFlow no seu escopo.

Existem várias implementações de controladores disponíveis para a plataforma OpenFlow, pois a arquitetura da plataforma foi desenvolvida com o objetivo de incentivar a inovação. A seguir, são descritos alguns dos controladores mais relevantes, incluindo as primeiras propostas e os softwares mais utilizados atualmente.

4.2.1 Controlador de Referência OpenFlow

O controlador de referência OpenFlow, também chamado OVS-controller, é um controlador mais simples, que é distribuído com o OpenFlow. Seu principal objetivo é fornecer uma referência simples da implementação de um controlador OpenFlow. Normalmente, o OVS-controller é usado para verificar as tabelas de fluxo do *switch* OpenFlow. Este controlador gerencia qualquer versão do *switch* usando o protocolo OpenFlow [58].

4.2.2 NOX e POX

Um dos primeiros controladores OpenFlow, e ainda hoje um dos mais utilizados, é o NOX [68]. O NOX introduziu a ideia do sistema operacional de rede, que fornece uma interface centralizada e uniforme para programar toda a rede. Um controle e gerenciamento centralizados da rede são mais simples de realizar e mais eficientes do que uma abordagem distribuída [68].

As funções de gerenciamento e controle no NOX são feitas por aplicações do sistema operacional de rede. Isto é, funções de controle como monitoramento de dados, balanceamento de carga, roteamento, dentre outros, são aplicações que rodam no NOX.

A interface do NOX fornece automaticamente a visão geral da rede o que inclui a topologia. Baseada na topologia, os algoritmos de controle e gerenciamento podem exercer suas funções. Além disso, o NOX provê uma interface para coletar dados sobre os *switches* e fluxos [58].

O NOX pode configurar fluxos proativamente ou reativamente. Na primeira opção, o controlador pode configurar os fluxos antes que os pacotes comecem a trafegar pela rede. Na segunda opção, quando um pacote desconhecido chega ao *switch* este encaminha o cabeçalho do pacote para o controlador. Com base nos dados recebidos, o controlador pode calcular e configurar as regras adequadas para aquele fluxo. As duas abordagens podem ser usadas em conjunto.

Inicialmente, as aplicações do NOX eram escritas em Python ou $C++$ [68]. Da API Python do controlador NOX nasceu o POX. O controlador POX é recomendado para experimentos e fins educacionais, e resulta em uma interface mais elegante e simples do que o NOX [71], além de oferecer melhor desempenho que o NOX com Python, como é ilustrado na Figura 4.14.

O objetivo dos desenvolvedores é que o POX venha a substituir o NOX nos casos

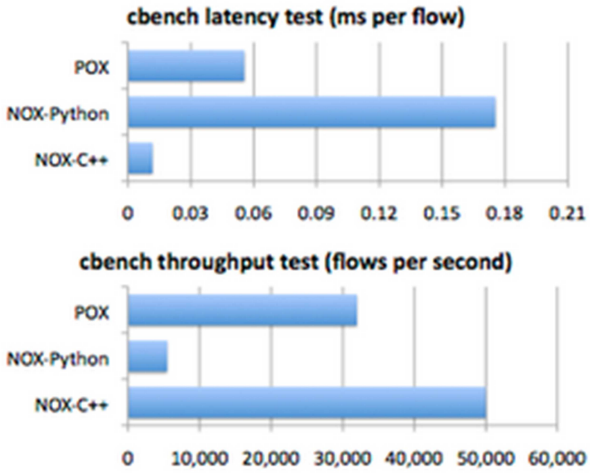


Figura 4.14: Comparação de desempenho entre o POX e NOX, com suas duas interfaces (C++ e Python) [71]

onde o desempenho do controlador não seja um requisito muito crítico, e especialmente nos casos onde a interface Python for utilizada. Nesse aspecto, o POX traz uma interface mais moderna e elegante.

Além do POX, outros controladores foram desenvolvidas com base no NOX, como é o caso do Jaxon, que é controlador OpenFlow baseado em Java que é NOX *dependent*. O Jaxon fornece uma interface Java que interage com NOX [72].

O Controlador SNAC é outro controlador derivado de NOX, que fornece uma interface Web que utiliza uma linguagem de definição flexível. O SNAC torna a tarefa de configuração de dispositivos e monitoração da rede mais fácil [58].

4.2.3 Outros Controladores

Existem muitos outros controladores além do NOX e do POX que apresentam algumas diferenças. Dentre esses, ressalta-se os seguintes controladores:

- **ONIX**

  É um controlador que é executado em um *cluster* de um ou mais servidores físicos [73]. As principais diferenças do Onix para NOX é que o Onix é mais confiável, escalável, e oferece uma API de programação mais flexível. O Onix provê uma interface que trata eficazmente eventos de falhas na rede. Para isso, uma das instâncias do Onix é responsável pela divulgação do estado atual da rede para as outras ins-

tâncias Onix no *cluster*. A escalabilidade é alcançada através da utilização de um conjunto de servidores, em vez de um único servidor, como no NOX. Os servidores no *cluster* se comunicam e são capazes de compartilhar o controle da rede. Em termos de flexibilidade de programação, o Onix permite aos projetistas implementar aplicações de controle distribuído.

- **Trema**
  É um *framework* para o desenvolvimento de controladores OpenFlow. As linguagens que este controlador suporta são C e Ruby.

- **Beacon e Floodlight e Maestro.**
  O Beacon é um controlador baseado na linguagem Java, que suporta operações baseadas em eventos e em *threads*. O Beacon pode rodar em diferentes plataformas. Devido as suas vantagens, foi a base para o desenvolvimento do controlador Floodlight, também baseado em java. Além do Beacon e o Floodlight, outro controlador escrito em Java é o Maestro.

4.3 Vantagens e Benefícios do OpenFlow

A principal vantagem do OpenFlow está na programabilidade por intermédio de inteligência em um plano de controle central e remoto. Esse plano de controle pode rodar em computadores comuns ou na nuvem e, por isso, é totalmente aberto a inovações. O protocolo OpenFlow separa o plano de controle do plano de dados, permitindo a utilização de controladores remotos baseados em servidores com sistemas operacionais e linguagens de programação comuns na indústria de TI. O software do controlador é responsável por definir o modo como os fluxos de pacotes são encaminhados e processados na rede. Isso permite que o controle sobre o plano de dados seja "terceirizado"a pesquisadores, sistemas de gerência, desenvolvedores, operadores de rede, plataformas de serviços e, até mesmo, aplicações finais, como, por exemplo, servidores de conteúdo ou serviços em nuvem. Dessa forma, o controle da rede deixa de estar embarcado nos equipamentos e limitado por implementações e padrões com mais de 15 anos de existência [59].

O OpenFlow não depende da forma como o software controla a rede, e oferece um serviço simples de encaminhamento multicamada (L1-L4) orientado a fluxos definidos por qualquer combinação de mais de 20 campos do cabeçalho padrão. Além disso, é compatível com os equipamentos utilizados nas redes atuais [59]. De fato, a interface do OpenFlow é transparente para usuários e outros equipamentos de rede, que podem continuar operando

da mesma forma.

Dentre as vantagens da redes OpenFlow podemos citar:

- Simplicidade e flexibilidade por ser mais programável - é possível programar as redes e não simplesmente configurá-las, o que significa uma quantidade muito mais vasta de possibilidades de ações e de tipos de tomada de decisão. Além disso a programação é centralizada, o que deixa os algoritmos mais simples [74].

- Diminuição da carga de controle - Ao remover a carga de processamento de controle dos *switches*, o OpenFlow permite que os switches concentrem-se no tráfego da rede de forma mais rápida [75, 74].

- Amplo suporte à inovação – Controladores e aplicações podem ser desenvolvidos de forma desassociada do fabricante, o que incentiva a inovação;

- Verificação de novos mecanismos - a validação de novas tecnologias em redes reais pode ser feita sem interromper o funcionamento da mesma, o que permite implementar e testar novos recursos sem prejudicar o tráfego de produção [75];

- Fabricantes não precisam expor o projeto de suas plataformas pois o OpenFlow pode ser incorporado em equipamentos de rede comerciais, atualmente, em operação, sem modificação do hardware e mediante uma atualização do *firmware*, garantindo o desempenho de tecnologias consolidadas no encaminhamento de pacotes IP/Ethernet, como, por exemplo, ASICs, FPGAs, etc [59]. A utilização do OpenFlow não diminui a competitividade entre os fabricantes de equipamentos de rede, que podem oferecer hardwares mais rápidos, além de novas funcionalidades de encaminhamento que vão além daquilo que foi definido pela especificação do OpenFlow;

- Gerenciamento - o controlador centralizado do OpenFlow permite que o administrador tenha uma visão centralizadas da rede, o que permite, dentre outras coisas, um melhor gerenciamento da rede. Ao permitir que os administradores vejam o fluxo geral de tráfego de forma mais clara, por exemplo as invasões e outros problemas são tratados de forma mais fácil [75].

Além disso, é compatível com novas tecnologias de rede, oferece uma maior integração, permite o uso de aplicações, permite o planejamento e a otimização de forma global [74], oferece suporte a recuperação da rede em tempo zero [64], provê suporte para a qualidade de serviço e para mobilidade, dentre outros benefícios.

Ressalta-se que, utilizar uma abordagem logicamente centralizada para realizar o encaminhamento *multicast* pode trazer diversas vantagens sobre a abordagem distribuída. Entre elas, a possibilidade de criar uma árvore de distribuição ótima para cada ocasião, devido à visão completa da topologia que um algoritmo centralizado possui. Também seria possível processar eventos de controle de grupo mais rapidamente sem criar inundações de mensagens como ocorre na abordagem distribuída.

O OpenFlow atualmente ainda está em fase de desenvolvimento. Apesar de nova, é uma tecnologia muito promissora e que já vem sendo adotada pelos principais fabricantes de equipamentos de rede.

Até o momento, não se conhece estudos relacionados ao uso do OpenFlow, que é uma tecnologia relativamente nova, em redes de subestações ou aplicado no contexto dos protocolos da norma IEC 61850. No entanto, existem trabalhos relacionados ao gerenciamento de rede e ao encaminhamento *multicast* para redes em geral [76, 77, 78].

Kotani et al. usam o OpenFlow para criar árvores *multicast* rapidamente usando caminhos redundantes [76]. Segundo os autores, construir árvores redundantes usando uma abordagem distribuída além de trabalhoso não é um método adequado. Para realizar essa tarefa com sucesso os autores escolheram o OpenFlow por possuir um controle centralizado da rede. A análise dos resultados mostra que o trabalho conseguiu uma baixa taxa de perda de pacotes quando os grupos mudam de árvore.

Marcondes et al. também implementam o OpenFlow para uma abordagem *multicast*. Neste trabalho o cálculo das rotas de cada fonte para cada destino possível é feito antecipadamente com o objetivo de reduzir o atraso na rede [77].

Silva et al. propõem o uso de OpenFlow para mobilidade e para o tráfego*multicast* com o objetivo de transmitir vídeo. A evolução mostrou uma redução no consumo de banda devido ao uso da abordagem proposta [78].

Capítulo 5

A proposta SMARTFlow

Conforme visto no Capítulo 3, apesar dos esforços de padronização, as redes baseadas na norma IEC 61850 ainda apresentam alguns problemas, que diminuem o desempenho da rede:

- o planejamento e a configuração da rede de telecomunicações ainda são feitos de forma manual;
- o controle da rede é ineficiente e, com isso, não é possível escolher o caminho que um pacote vai seguir;
- os pacotes *multicast* da camada de enlace são geralmente enviados por todas as portas do *switch*;
- os sistemas de recuperação de falhas recomendados pela IEC para subestações geram excesso de tráfego ou têm um tempo de recuperação alto; e
- com relação à prioridade dos pacotes na rede, embora os tempos sejam bem definidos na norma, com cada mensagem classificada em um tipo diferente, a prioridade que elas recebem tem pouca granularidade.

Para resolver esses problemas é proposto o SMARTFlow, que é um sistema autoconfigurável para redes de telecomunicações IEC 61850 baseado no arcabouço OpenFlow. Seus objetivos principais são o desenvolvimento de um encaminhamento apropriado de mensagens IEC 61850 e o controle eficiente de redes de dados de subestações elétricas. Além disso, com o SMARTFlow, é possível fazer a autoconfiguração dos grupos *multicast* de forma automática, toda vez que for inserido ou retirado um IED da rede, ou se alguma função lógica for migrada entre dispositivos físicos. Como a proposta é facilitar o

planejamento e a configuração da rede, é possível configurar automaticamente a rede de telecomunicações com a solução adequada aos requisitos da rede.

Dessa forma, os principais componentes do SMARTFlow, detalhados nas seções que seguem, são:

- um módulo de autoconfiguração inicial da rede de telecomunicações com base nos dados obtidos pelos arquivos de configuração da subestação padronizados pela norma IEC 61850;
- um módulo de criação automática de árvores *multicast* de camada de enlace para o envio de mensagens GOOSE e SV, as quais são atualizadas pró-ativamente sempre que o estado da rede muda;
- um módulo de recuperação de falhas através do uso de duas técnicas diferentes:
  - recálculo e atualização automática das rotas sempre que um dos enlaces da rede cair;
  - utilização do mecanismo *fast failover* do OpenFlow, caso a versão 1.1 ou superior do protocolo esteja disponível, para garantir a recuperação em tempo zero e sem sobrecarregar a rede com mensagens duplicadas;
- uma proposta de um modelo para aplicação de prioridade nos diferentes tipos de mensagens da rede, aumentando a qualidade do serviço oferecido.

Todos os serviços descritos são inovadores para as redes de subestação. Com exceção do modelo de priorização proposto nesse trabalho, tudo indica que, mesmo que boas práticas de gerência e controle de rede estivessem em uso, ainda assim não seria possível prover essas funcionalidades de forma mais eficiente.

5.1 O OpenFlow e o IEC 61850

Nessa dissertação, propõe-se utilizar as redes definidas por *software*, mais especificamente o OpenFlow, como arcabouço para resolver os problemas discutidos no Capítulo 3. Esse método possibilita um controle mais flexível e orientado às necessidades de cada sistema de comunicação, como é o caso das *smart grids* e da Norma IEC 61850. Com isso, torna-se possível experimentar novos métodos e novos algoritmos que garantam uma boa solução e aumentem o desempenho das redes. De fato, essa tecnologia traz ganhos por permitir

a implementação de todas as recomendações da norma e de recursos para otimizar a rede. Além disso, por existir um controlador centralizado, as redes OpenFlow oferecem flexibilidade de programação e uma visão unificada da rede, facilitando a implementação de novos algoritmos, a configuração e a gestão da rede. A visão global centralizada da rede torna a programação mais fácil.

O OpenFlow foi escolhido pois permite controlar o fluxo de cada tráfego da rede, escolhendo as rotas que seus pacotes deverão seguir e o processamento que receberão independente de camadas. Desta forma, torna-se possível experimentar esquemas de encaminhamento montando uma árvore *multicast* para encaminhar os pacotes de camada 2, da forma mais adequada possível e, principalmente, de forma padronizada pela norma IEC 61850, garantindo uma boa solução. Isso é possível pois o OpenFlow provê uma interface simples para o desenvolvimento de aplicações de controle de rede de telecomunicações. Dessa forma, pode-se desenvolver componentes que implementam algoritmos para criar e gerenciar os fluxos de maneira eficiente e simples, permitindo experimentar novos sistemas que aumentem o desempenho da rede.

Além disso, a manutenção preventiva da rede é realizada de uma maneira simples, uma vez que a migração de fluxos é simples em redes OpenFlow. Com o uso do arcabouço OpenFlow a virtualização da rede é simplificada: diferentes fabricantes podem implementar políticas para controlar a sua rede no mesmo *switch* sem interromper ou interferir com outros serviços.

5.2 Arquitetura

O SMARTFlow foi implementado no controlador OpenFlow POX. Por essa razão, a descrição detalhada da proposta será ilustrada com base nessa ferramenta. Contudo, a proposta poderia ser implementada em qualquer controlador OpenFlow que disponibilize um módulo para reconhecimento da topologia e outro para o encaminhamento padrão de pacotes sob demanda de tráfego sem priorização.

A arquitetura detalhada do SMARTFlow é ilustrada na Figura 5.1. O SMARTFlow é executado sobre um controlador, o qual é detalhado na Seção 5.2.1, que se comunica com os *switches* OpenFlow da LAN IEC 61850 via um canal seguro. Uma vez que o SMARTFlow é baseado em OpenFlow, a interface de comunicação entre controlador e switches é especificada pelo protocolo OpenFlow.

O SMARTFlow define um conjunto de aplicações, que são descritas a seguir na Se-

ção 5.2.1, as quais são responsáveis pela coleta da topologia e do estado da rede, além da coleta das informações sobre a subestação, obtidas através do arquivo SCD definido na norma IEC 61850. Com base nesses dados, aplicações específicas são chamadas para definir rotas pró-ativamente ou sob demanda, dependendo do tipo de tráfego, além de realizar o tratamento de falhas e o balanceamento de carga na rede. Uma das idéias principais é fazer um encaminhamento eficiente encontrando caminhos que não sobrecarreguem a rede, independentemente da topologia da subestação.

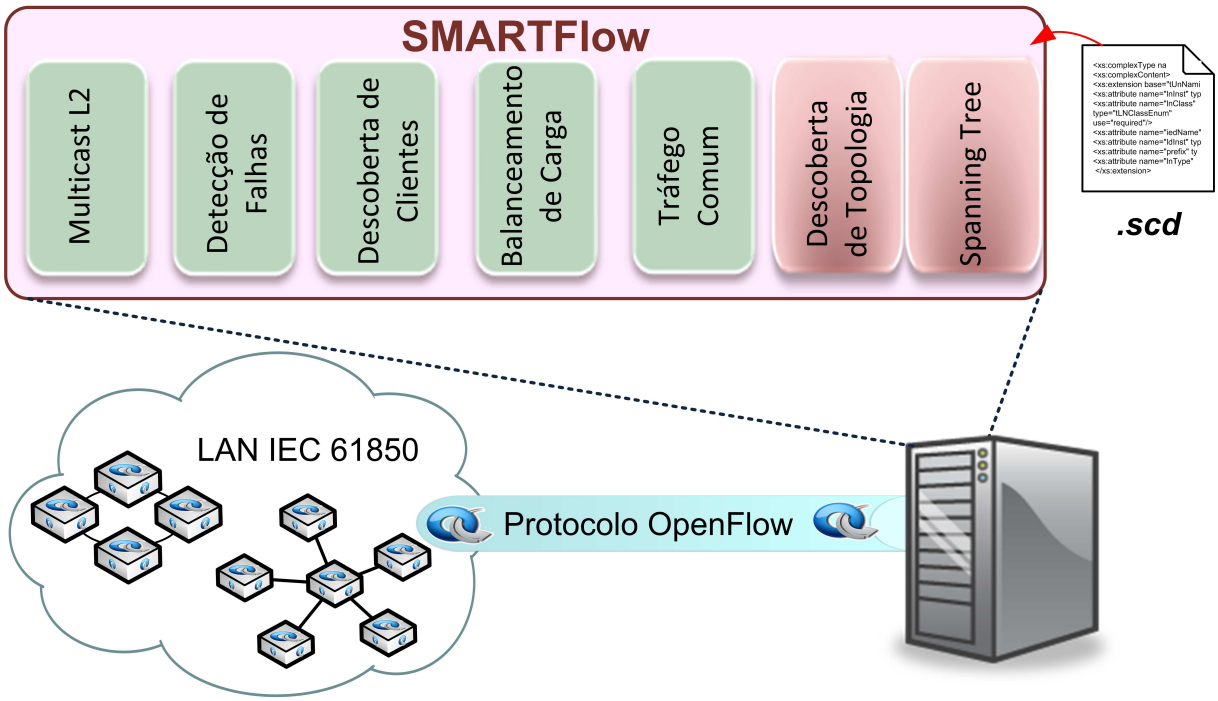


Figura 5.1: Estrutura do SMARTFlow, o qual é composto por um conjunto de aplicações de controle para a rede de comunicação das subestações baseadas em IEC 61850.

O controlador é um elemento central da rede que se comunica com todos os *switches* para configurar as tabelas de fluxo. O esquema do SMARTFlow pode ser implementado em qualquer controlador OpenFlow. Uma das opções é o POX, que foi utilizado para a implementação da proposta e pode ser implementado em um servidor comum. Os módulos nativos dos controladores OpenFlow utilizados em conjunto com o SMARTFlow são encontrados em todas as principais distribuições de controladores, por tratarem das funções básicas de reconhecimento da topologia e encaminhamento padrão de pacotes.

5.2.1 Aplicações de controle do SMARTFlow

No SMARTFlow, o plano de controle da rede de telecomunicações é responsável por monitorar e configurar a rede automaticamente a partir de parâmetros do arquivo SCD

da Norma. O SMARTFlow propõe o uso dos seguintes componentes, os quais todos foram desenvolvidos para o SMARTFlow com exceção do `Descoberta de Topologia` e do `Spanning Tree`, que são componentes nativos do OpenFlow [1]:

- `Descoberta de Topologia`: Este componente é nativo dos controladores e é essencial para o funcionamento dos outros componentes do SMARTFlow. O componente `Descoberta de Topologia` instrui cada um dos *switches* da rede a enviar mensagens *Link Layer Discovery Protocol* (LLDP) de modo que o controlador possa descobrir a topologia da rede. Quando um *switch* recebe um pacote LLDP, ele encaminha o cabeçalho do pacote para o controlador, pois a regra para esse fluxo não está definida. Com isso, o controlador pode inferir a conectividade dos enlaces combinando os dados de cada pacote LLDP recebido.

- `Spanning Tree`: Já conhecendo a visão geral da topologia da rede, pelo componente `Descoberta de Topologia`, pode-se, então, construir a *spanning tree*. O objetivo do componente *Spanning Tree* é calcular a árvore de cobertura da rede e, de acordo com os dados obtidos, desativar a primitiva de inundação em alguns enlaces específicos. Isso faz com que topologias que formem ciclos evitem *loops* infinitos no envio de mensagens em *broadcast* [2].

- `Tráfego Comum`: Este componente encaminha, sob demanda, o tráfego unicast de baixa prioridade das redes baseadas em IEC 61850, como por exemplo, mensagens *unicast* e *mensagens* MMS. O componente `Tráfego Comum` é uma versão do componente nativo do Openflow que funciona como um *switch* de aprendizagem de forma reativa. Assim, o componente identifica quais eventos de `Packet In` se referem a fluxos de baixa prioridade e chama o componente nativo para tratá-los. Isso significa dizer que, sempre que um novo fluxo *unicast* de baixa prioridade chegar a um *switch*, este dispositivo encaminhará o cabeçalho do primeiro pacote para o controlador, o qual acionará o módulo `Tráfego Comum` para calcular e definir as entradas de fluxo correspondentes com base no estado atual da rede.

- `Descoberta de Clientes`: Este componente captura os pacotes `Packet In` da rede e faz o mapeamento dos endereços MAC de todos os IEDS na rede automaticamente,

[1] Entende-se, nessa dissertação, por componentes nativos do OpenFlow, as aplicações de controle disponibilizadas pelos controladores OpenFlow que implementam a descoberta de topologia, o cálculo e configuração da *spanning tree* e o encaminhamento de pacotes na rede.

[2] A principal diferença da *Spanning Tree* do OpenFlow para a tradicional é que no OpenFlow as portas não são completamente desativadas. Apenas pacotes com a ação `FLOOD` não são encaminhadas pela porta, isso significa que o tráfego *unicast* e mensagens de controle, por exemplo, continuam passando pela porta.

armazenando em uma lista todos os endereços MAC dos IEDs, assim como a qual portas e *switches* estão ligados. Cada vez que um IED é acrescentado ou retirado da rede, esse componente atualiza essa lista. Os endereços MAC dos IEDs e servidores, na prática, podem ser obtidos do arquivo SCD da norma IEC 61850. Contudo, essa configuração inicial não é suficiente para detectar erros de ligação, ou ainda, mudanças na topologia geradas durante o funcionamento da rede. Assim, optou-se pelo uso de uma solução mais genérica. O componente proposto monitora os eventos da rede e mantém atualizada a lista que correlaciona o MAC e a porta de saída de cada *switch*.

- `Multicast L2`: Este componente implementa o algoritmo para calcular as árvores de distribuição para os grupos *multicast* de camada dois, necessários para as mensagens GOOSE e SV, o qual é descrito em detalhes na Seção 5.3. Utilizando como entrada o arquivo SCD, esse componente identifica os grupos *multicast* e, em seguida, com base nos dados das aplicações `Descoberta de Topologia` e `Descoberta de Clientes`, calcula as árvores *multicast* para cada grupo na rede IEC 61850 da subestação e configura pró-ativamente os respectivos *switches*. Essa configuração pró-ativa é importante porque as grupos *multicast* são utilizados por mensagens de alta sensibilidade a atrasos no IEC 61850. Com isso, a configuração pró-ativa impede que a entrega das mensagens seja atrasada pela consulta reativa ao controlador, o que seria o comportamento padrão do OpenFlow.

- `Detecção de Falhas`: Este componente é chamado sempre que ocorre um evento que indica quando um enlace está operacional ou não. Desta forma, sempre que um enlace cair ou um *switch* falhar, ele chama o componente `Multicast L2` e recalcula as árvores. O componente e seu algoritmo são detalhados na Seção 5.3 [3].

- `Balanceamento de carga`: Esse componente observa o estado da rede e distribui a carga dos fluxos *unicast* menos prioritários entre os enlaces, através do uso de migrações ao vivo [79, 80]. Com isso, sempre que se observa que um enlace está próximo de um certo limiar de uso, alguns fluxos são migrados para outros enlaces menos sobrecarregados. Essa migração é feita criando-se uma nova rota a partir do destino até a origem. A rota original é apagada apenas quando a nova rota já estiver pronta para uso, garantindo, assim, que não serão perdidos pacotes. Isso ajuda a melhorar o desempenho da rede e a minimizar a latência de resposta, o

[3] Cabe observar que o comportamento desse componente varia de acordo com a versão do OpenFlow em uso, de acordo com a disponibilidade ou não da tabela de grupo e do mecanismo de *fast failover*.

que é essencial para redes de comunicação que auxiliam mecanismos de proteção do SEP.

Desta forma, tem-se uma rede estável que realiza o cálculo antecipado de árvores *multicast* de acordo com os grupos definidos no arquivo SCD da norma, e que, além disso, controla os demais fluxos que são transmitidos pela rede, escolhendo as rotas que seus pacotes deverão seguir e o processamento que receberão independente de camadas. Além disso, com o componente `Detecção de Falhas`, o controlador recalcula as rotas sempre que houver uma falha como será detalhado na Seção 5.3. Como consequência, os pacotes são encaminhados apenas para os nós lógicos corretos de uma forma eficiente e simples, sem aumentar o tráfego na rede ou a latência desta. Isso significa que, por exemplo, as mensagens GOOSE tipo 1A (*Trip*), apresentadas na Seção 2.2.2 na Tabela 2.1, são apenas transmitidas ao grupo de interesse, por uma árvore *multicast* bem definida, evitando que o tráfego seja enviado para toda a rede desnecessariamente, evitando a sobrecarga nos *switches* e nos IEDs.

Além disso, o sistema é autoconfigurável, o que significa que sempre que a rede for iniciada ou que um elemento novo for conectado à rede, o sistema se configura sozinho. Uma vantagem grande, que traz ganhos no desempenho da rede, é que o cálculo é realizado proativamente, durante a inicialização da rede, evitando assim a necessidade de cálculos de rotas ou de mensagens de controle durante o funcionamento da rede. Essa é uma das diferenças básicas para os componentes nativos do OpenFlow, que tem um funcionamento reativo. O funcionamento proativo é possível em redes IEC 61850, pois são redes bem conhecidas e específicas, onde cada fluxo é conhecido, assim como cada grupo *multicast* envolvido.

5.3 Algoritmos de controle do SMARTFlow e configuração automática a partir do arquivo SCD

Os algoritmos de controle do SMARTFlow usam como base informações contidas no arquivo SCD da Norma. Como foi visto na Seção 2.2.3, o arquivo SCD pode conter, dentre outras informações, a que grupo *multicast* cada IED pertence, quais nós lógicos cada IED possui e os endereços de rede do mesmo. Dessa forma, torna-se possível fazer um mapeamento dos grupos *multicast* existentes na rede para o algoritmo calcular as árvores.

O algoritmo do componente `Multicast L2`, graças ao componente `Descoberta de`

`Topologia`, é capaz de ter uma visão completa da rede, podendo armazenar em uma lista, todos os enlaces e nós da rede. Além disso, o componente `Descoberta de Clientes` gera como saída um dicionário mapeando todos os endereços MAC dos IEDs da rede com as suas respectivas portas e *switches* conectados ao dispositivo. Esse dicionário é intitulado *mac_map*.

A descrição detalhada do componente `Multicast L2` é dada no Algoritmo 1. A lista de enlaces (E), a lista de nós (N), a lista de grupos *multicast* ($gruposmulticastgerados$) e o dicionário *mac_map* são as entradas desse algoritmo. A saída é o caminho completo da árvore *multicast*, intitulado *caminho_completo*, que o próprio algoritmo usa para configurar os fluxos em cada *switch* da rede.

Algoritmo 1: Algoritmo de cálculo de árvores *multicast* e estabelecimento de rotas

```
   Input: E, N, mac_map, gruposmulticastgerados
   Output: caminhos_completos
 1 for grupo in gruposmulticastgerados do
 2 |   end_GM = descobre_end(grupo)
 3 |   raiz = descobre_raiz(mac_map, grupo)
 4 |   sws_destino = descobre_dst(mac_map, grupo)
 5 |   melhores_rotas = SPF_multicast(raiz, N, E)
 6 |   for rota in melhores_rotas do
 7 |   |   if rota[dst] in sws_destino then
 8 |   |   |   caminhos_arvore.append(rota)
 9 |   |   end
10 |   end
11 |   arvore_multicast = remove_redundancias(caminhos_arvore)
12 |   caminhos_completos = acrescenta_portas(mac_map, arvore_multicast)
   |   instala_fluxos(caminhos_completos, end_GM, raiz)
   |   returncaminhos_completos
13 end
```

A lista de grupos *multicast* é percorrida para criação e configuração de cada árvore *multicast*, como mostrado na linha 1. A função *descobre_end*, na linha 2, descobre qual o endereço MAC do grupo *multicast*. A função *descobre_raiz*, na linha 3, com base no *map_mac* descobre qual o endereço MAC do nó raiz do grupo *multicast* e em qual *switch* está conectado. É importante observar que serão considerados como raiz todos os nós que possam emitir mensagens para o grupo *multicast*. A função *descobre_dst*, na linha 4, retorna uma lista com os *switches* que estão conectados a pelo menos um dos membros do grupo, com as respectivas portas.

Com os nós, os enlaces e a raiz do grupo *multicast* a função $SPF_multicast$, que roda o algoritmo *Dijkstra Shortest Path First* (SPF), calcula todas as rotas mais curtas

da raiz para cada outro nó restante da rede, possíveis nós receptores do grupo. Com isso, é criada uma lista de caminhos mais curtos, intitulada *melhores_rotas*.

Conforme descrito na linha 6, as rotas da lista *melhores_rotas* que tiverem como destino os nós receptores do grupo são armazenadas na lista *caminhos_arvore* que, ao final, contém os melhores caminhos do IED publicador até os IEDs receptores do grupo *multicast*. A lista *caminhos_arvore* é processada para remoção de redundâncias com a função *remove_redundancias*, que exclui qualquer caminho repetido na lista. Essa função gera a lista *arvore_multicast*, contendo o caminho pelo qual os pacotes deverão passar para alcançar os destinos daquele grupo, conforme linha 11. Para que as tabelas dos *switches* possam ser configuradas, além do caminho é necessário o acréscimo das portas pelas quais esse caminho está conectado. Essa tarefa é realizada pela função *acrescenta_portas*, que gera como saída a lista intitulada *caminhos_completos*, contendo uma lista para cada *switch* com o seu número de identificação e as portas por onde os pacotes serão encaminhados, ou seja, a tabela de fluxos que deverá ser configurada. Por fim, o controlador pode configurar as tabelas de fluxos nos *switches* pertencentes à lista *caminhos_completos* criando assim a árvore*multicast* do grupo em questão.

Outro componente de grande importância é o `Detecção de Falhas`. Conforme o Algoritmo 2, o componente escuta a rede até que seja detectada alguma mudança de estado, como a queda de um enlace. Nesse caso, o componente chama o `Multicast L2`, que recalcula as árvores *multicast* para os grupos afetados pela falha, restabelecendo a rede dinamicamente.

Algoritmo 2: Algoritmo de detecção de falhas e restauração da rede

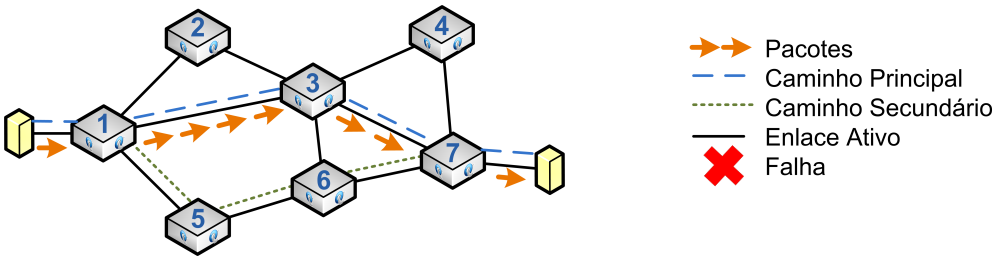
Input: *evento*,*E*, *N*, *mac_map*, *gruposmulticastgerados*, *caminhos_completos*
Output: *caminhos_completos*

1 $enlaces_afetados = busca_falhas(evento,E,N)$
2 $grupos_afetados, grupos_nao_afetados = busca_grupos(enlaces_afetados, caminhos_completos, gruposmulticastgerados)$
3 $novos_grupos = multi_tree(E, N, mac_map, grupos_afetados)$
4 $caminhos_completos = novos_grupos + grupos_nao_afetados$
5 $returncaminhos_completos$

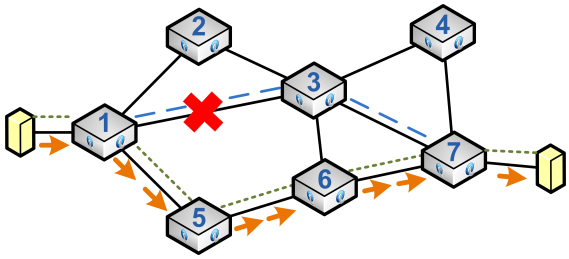
É importante notar que, se a versão do OpenFlow suportar o *Fast Failover*, então existem pequenas alterações nos Algoritmos 1 e 2. O Algoritmo 1, após calcular a árvore de distribuição para um par (endereço MAC multicast, raiz), irá aumentar os custos de todos os enlaces utilizados nessa árvore de cobertura e irá buscar uma nova árvore de distribuição na rede. Em seguida, essa nova árvore é adicionada como caminho para casos de falha na tabela de grupo dos *switches* OpenFlow. É importante observar que

o aumento do custo garante que uma segunda árvore será encontrada, mesmo que não existam caminhos totalmente disjuntos. As modificações no Algoritmo 2 ocorrem apenas para apagar a rota original que está com uma falha, de forma que a segunda opção se torne a rota principal. O Algoritmo, naturalmente, já buscará uma terceira opção de árvore de distribuição, para prevenir novas falhas na rede. Ressalta-se que, a rede deverá prover mais de um caminho, caso a rede só tenha um caminho possível, esse componente não se aplica.

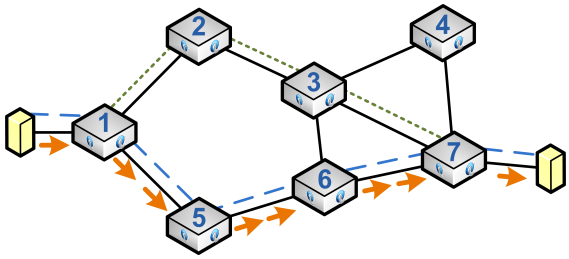
A Figura 5.2 dá um exemplo dessa situação.



(a) Situação normal com o *switch* encaminhando os pacotes pela árvore principal.



(b) Falha no enlace de 1 para 3. *Switch* encaminha os pacotes pela árvore secundária.



(c) Algoritmo já rodou e a rede está estabilizada com uma nova árvore secundária.

Figura 5.2: Mecanismo para recuperação de falha no SMARTFlow implementado em um *switch* OpenFlow 1.1 ou superior.

5.4 Diferenciação do Tráfego

Como foi visto na Tabela 2.3, o Padrão IEC 61850 define o valor quatro para prioridade das mensagens GOOSE e SV e o valor zero para as demais. Isso significa que, o campo PCP do quadro Ethernet (Figura 2.7) teria o valor *default* quatro para ambas as mensagens e seus diferentes subtipos. Assim, apesar de requisitos temporais distintos, como mostrado na Tabela 5.1, as mensagens 1A, 1B e 4, nas classes P1, P2 e P3, teriam a mesma priorização, como foi mostrado também na Seção 3.1.

O SMARTFlow propõe uma forma mais detalhada de priorização, onde a segmentação dos fluxos ocorre com mais precisão. Para isso, considera-se o tipo de mensagem, a

classe a que pertence, a sua aplicação e os seus requisitos temporais, aumentando assim a confiabilidade e o desempenho da rede. A priorização continua sendo feita seguindo o IEEE 802.1p, porém o campo PCP, que anteriormente tinha o valor fixado em zero ou quatro, usa mais níveis de prioridade, sendo o nível zero como mais baixo e sete como mais alto. Desta forma, ao invés de definir a mesma prioridade para mensagens GOOSE e SV, propõe-se especificar valores de prioridade diferentes, de acordo com a Tabela 5.1.

Tabela 5.1: Proposta para diferenciação do Tráfego em Rede IEC 61850.

Tipo	**Mensagem**	**Classe**	**Requisito Temporal**	**Prioridade *default***	**Prioridade Proposta**
1A	GOOSE	P2 e P3	3 ms	4	6
1A	GOOSE	P1	10 ms	4	5
1B	GOOSE	P2 e P3	20 ms	4	4
1B	GOOSE	P1	100 ms	4	3
2	MMS	-	100 ms	0	2
3	MMS	-	500 ms	0	1
4	SV	P2 e P3	3 ms	4	6
4	SV	P1	10 ms	4	5
7	MMS	-	500 ms	0	1

Isto garante maior flexibilidade, maior granularidade e maior qualidade de serviço para as redes IEC 61850. Por exemplo, mensagens GOOSE tipo 1B – classe P2, que têm requisitos temporais diferentes das mensagens GOOSE tipo 1A – classe P1, serão tratadas diferentemente, e assim por diante. Com esse sistema seriam usados 6 níveis de prioridade, dos 8 possíveis.

Com relação ao projeto como um todo, ganha-se mais flexibilidade com relação à segmentação da rede em redes virtuais (VLANs). Como as mensagens e aplicações já estariam priorizadas de forma adequada, sendo tratadas de acordo com seus tipos de aplicação e requisitos temporais, o uso de VLANs poderia se restringir a separar logicamente a rede. Por exemplo, cada fabricante poderia ter uma VLAN própria ao invés de usar este método para separar diferentes tipos de mensagem. Outro exemplo é a rede de uma concessionária, que, muitas vezes, possui a rede estruturada tendo aplicações distintas em cada VLAN. Por exemplo, uma VLAN para supervisão, uma para sincronização, e assim por diante. Quando a concessionária já possui esse projeto, torna-se difícil a colocação de redes IEC 61850 que tenham outra forma de segmentação em VLANs na mesma rede. Além disso, não existe uma modelagem padrão definida para o uso de VLANs em subestações IEC 61850. Assim, cada fabricante desenvolve o seu projeto de acordo com

suas próprias políticas, o que muitas vezes faz com que o principal objetivo da norma, a interoperabilidade, seja perdido.

Com o mecanismo de priorização proposto em união com a solução *multicast*, não se teria a necessidade de implantação de VLANs apenas para segmentar a rede em grupos. Ressalta-se que o mecanismo proposto é aplicado no SMARTFlow mas pode ser facilmente aplicado a redes IEC 61850 atuais em produção.

5.5 Vantagens e Desvantagens

As principais vantagens de usar o arcabouço do OpenFlow para *Smart Grids*, estão listados abaixo:

- O OpenFlow fornece uma interface de programação centralizada da rede, o que é ideal para redes de subestação. Isto facilita da configuração, da gestão e da manutenção da rede.
- A manutenção preventiva da rede é realizada de uma maneira simples, uma vez que a migração de fluxos é simples em redes OpenFlow.
- O uso do arcabouço OpenFlow simplifica a virtualização da rede. Diferentes fabricantes podem implementar políticas para controlar a sua rede no mesmo *switch* sem interromper ou interferir com outros serviços.
- A aplicação do arcabouço OpenFlow em redes IEC 61850 permite a validação de novas propostas em redes em produção. Novos protocolos, aplicações e recursos destinados a melhorar o desempenho da rede podem ser testados e validados em redes reais, sem afetar o tráfego em produção. A virtualização de redes simplifica a experimentação destes novos recursos de rede.
- O OpenFlow trabalha também com redes sem fio, pois pode ser implementado em pontos de acesso OpenFlow.

Além de trazer todas essas vantagens, intrínsecas ao OpenFlow, as principais contribuições do SMARTFlow são listadas abaixo:

- O SMARTFlow utiliza o arquivo SCD da Norma IEC 61850 para deixar mais eficientes os algoritmos do plano de controle.

- O SMARTFlow utiliza o *multicast* de camada dois de forma eficiente, ou seja, sem inundar a rede, com configuração automática dos switches, e de forma transparente para os dispositivos finais. A proposta calcula uma árvore *multicast* de forma que os quadros são enviados apenas para os nós de interesse sem sobrecarregar a rede maximizando o desempenho da LAN na subestação. É importante notar que nenhuma solução atual é capaz de realizar essas funções de forma combinada. De fato, os IEDs não têm suporte para os protocolos de multicast e todas as soluções acabam sendo baseadas em inundações controladas com o uso de VLANs.

- O SMARTFlow faz a configuração de caminhos automaticamente e proativamente, trazendo simplicidade e reduzindo a incidência de erros de configuração, principalmente quando houver mudanças na topologia da rede. Isto é essencial para subestações, onde não são toleradas falhas em nenhuma circunstância, já que falhas nas rede de telecomunicações podem interromper criticamente o serviço elétrico.

- O SMARTFlow propõe um componente para balanceamento de carga, que redireciona e restringe o tráfego em situações de sobrecarga de rede, a fim de otimizar a utilização de recursos, maximizar o desempenho, e minimizar o tempo de resposta.

- O SMARTFlow oferece a tolerância a falhas eficiente sem sobrecarregar a rede.

- O SMARTFlow, por possuir componentes proativos, reduz muito os tempos de atraso na rede e a carga de controle desta, aumentando substancialmente o desempenho da rede.

- O SMARTFlow propõe o uso mais consciente da priorização dos pacotes, melhorando a qualidade de serviço na rede.

É importante ressaltar que o OpenFlow está sendo proposto para este trabalho, porque além de todas as vantagens indicadas acima, já está sendo disponibilizado pelos fabricantes de equipamentos de rede [60]. Observa-se, contudo, que a tecnologia ainda é muito nova e precisa de testes industriais antes de ser adotada dentro de uma subestação.

Capítulo 6

Análise do SMARTFlow

As redes elétricas inteligentes dependem de uma base sólida de comunicação. Para construir essa base, requisitos essenciais como velocidade e confiabilidade devem ser atendidos, garantindo assim o bom desempenho das aplicações. Fornecer um serviço de rede melhor e mais previsível ajuda a fornecer um bom desempenho do sistema.

O objetivo do SMARTFlow é atender a esses requisitos de forma que possa ser implementado em subestações. Para isso, foi feito um estudo para avaliar a proposta no que diz respeito:

1. às garantias temporais na comunicação de dados da norma IEC 61850. Normalmente são requeridos valores de latência rígidos da ordem de milissegundos, no caso mais rigoroso chegando a 3ms como é o caso das mensagens *trip*. Por esse motivo, o SMARTFlow deve garantir que o atraso na rede não ultrapasse os limites estabelecidos pela norma, apresentados na Tabela 2.1 da Seção 2.2;

2. ao desempenho do SMARTFlow no que diz respeito aos tempos de descoberta da rede, de cálculo do algoritmo e de configuração dos fluxos, que precisam ser aceitáveis para implementação em subestações. Além disso, um bom desempenho nesses tempos garante também a rapidez no restabelecimento da rede em caso de falha;

3. ao desempenho do SMARTFlow, quando comparado com as aplicações padrão do OpenFlow e com um *switch* tradicional, no que diz respeito ao atraso das mensagens, carga de controle da rede e carga total de dados.

4. às características do SMARTFlow quando comparado aos métodos de recuperação de falha da norma IEC 61850;

5. às características do SMARTFlow quando comparado à soluções *multicast* de camada 2 usadas em subestações.

Para os itens 1, 2 e 3 foram desenvolvidos experimentos para avaliar o desempenho. Os cenários e os parâmetros escolhidos são descritos na Seção 6.1. Na Seção 6.2 o ambiente de implementação desenvolvido para esses experimentos é descrito. À seguir a Seção 6.3 descreve os experimentos realizados e quais foram os resultados encontrados. Para os itens 3 e 4 foi feita uma análise qualitativa que é descrita na Seção 6.4.

6.1 Cenários e parâmetros de teste

Para avaliar o desempenho do SMARTFlow é necessário definir que cenários precisam ser testados, assim como os motivos para escolha de certos parâmetros como quantidade de *switches* e IEDs na rede. Para isso, foram identificados os cenários típicos de subestação. Os requisitos de desempenho de uma rede em uma subestação dependem diretamente do tamanho da subestação e da sua importância no sistema. A norma IEC 61850 [5] classifica as subestações de acordo com seus tamanhos e funções da seguinte forma:

- Subestação de distribuição de pequeno porte (D1): Uma subestação de distribuição com até 5 equipamentos, como por exemplo, 4 alimentadores de painéis e um disjuntor de interligação de barras. Segundo a norma [5], essa subestação pode ser equipada com apenas um IED que possua a função de sobrecorrente. Este mesmo IED teria funções de controle, como controle do disjuntor por exemplo, e alarme para o centro de controle.

- Subestação de distribuição de médio porte (D2): É o tipo mais comum de subestação, tendo mais do que 5 elementos e menos do que 20. Desta forma, 3 IEDs seriam suficientes [5].

- Subestação de distribuição de grande porte (D3): Subestação de distribuição com mais do que 20 elementos.

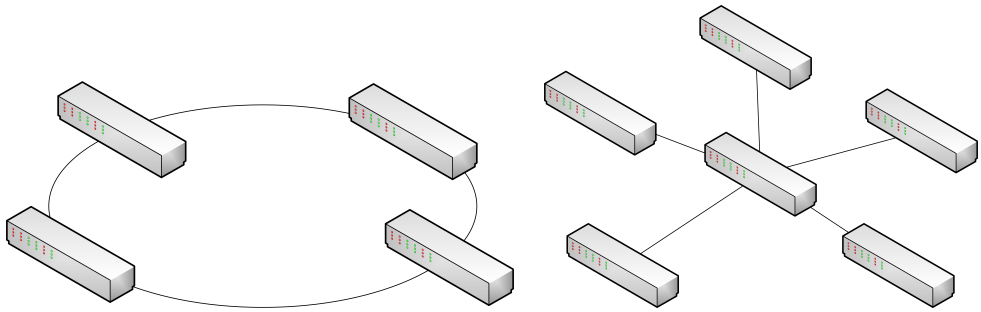
- Subestação de Transmissão de pequeno porte (T1): Uma subestação de transmissão com até 10 elementos.

- Subestação de Transmissão de grande porte (T2): Uma subestação de transmissão com mais do que 10 elementos.

Em todos os casos, a quantidade de IEDs depende muito do projeto e funções que serão utilizadas na subestação. Assumindo uma quantidade de 3 até 12 IEDs pode-se assumir que os testes englobam, senão todos, a maior parte dos cenários típicos em subestações[1]. Para cenários que não tiveram variação no número de IEDs, fixou-se o número em nove, para que os testes com nove *switches* tivessem pelo menos um IED conectado a cada *switch*.

Com relação à quantidade de *switches*, levou-se em consideração, na maioria dos cenários, o número de um a nove, e para os testes que não sofreram variação do número de *switches*, para que a rede ficasse bem distribuída, fixou-se a quantidade em cinco.

A topologia da LAN IEC 61850 é definida em função de vários fatores, dependendo do projeto, do número de IEDs a serem interligados, da disponibilidade desejada, da confiabilidade esperada, do seu custo operacional, dentre outros. As topologias mais comumente utilizadas em subestações são a anel e a estrela, por esse motivo esse trabalho levou em conta essas duas topologias. A Figura 6.1 ilustra essas topologias, descritas a seguir:



(a) Topologia em Anel (b) Topologia em Estrela

Figura 6.1: Topologias testadas

- Topologia em anel: Nesta topologia, os *switches* são interligados através de um circuito fechado, em série. Desta forma cada *switch* funciona como um repetidor, retransmitindo os sinais para o próximo *switch* da rede até que seja encontrado o destinatário.

- Topologia em estrela Neste tipo de rede os *switches* são diretamente ligados a um elemento central. Desta forma, toda informação passa pelo nó central da rede.

[1]Na prática, uma quantidade maior de IEDs pode ser instalada. Isso acontece, pois a capacidade multifuncional do IED, muitas vezes, não é explorada, e com isso, uma quantidade maior de IEDs é instalada mesmo sem necessidade.

Com o que foi descrito, e com base nos parâmetros escolhidos, foram definidos 4 cenários de testes que estão resumidos na Tabela 6.1. Quando a quantidade de elementos

Tabela 6.1: Cenários de teste.

Parâmetros	**Cenário 1**	**Cenário 2**	**Cenário 3**	**Cenário 4**
Quantidade de IEDs	3 a 12	9	9	9
Quantidade de *Switches*	5	1 a 9	3 a 9	1 a 9
Quantidade de grupos *Multicast*	5	5	7	5
Quantidade de eventos na rede elétrica	10	10	10	10

é variada, leva-se em consideração todos os valores intermediários. A quantidade de eventos na rede elétrica causa um aumento no tráfego como será visto adiante. Em todos os cenários foram testadas as topologias em anel e estrela.

6.2 Ambiente de Implementação

O ambiente foi emulado por meio de virtualização em um notebook com processador Intel Core i5-3210M, e 4GB de memória RAM. Os testes foram realizados com três instancias de máquinas virtuais simultâneas, cada uma delas com uma CPU virtual, 1024 MB de memória e executando o sistema operacional Ubuntu 11.10.

A Figura 6.2 resume o ambiente de testes do SMARTFlow. Neste esquema, como exemplo, é apresentada a topologia em Anel. Os componentes do SMARTFlow foram desenvolvidos em Python e implementados no controlador POX 0.1.0 na versão 1.0.0 do OpenFlow. Como foi visto na Seção 4.2, o POX é uma plataforma para o desenvolvimento e a prototipagem de *softwares* de controle de rede usando Python. Com isso, tem-se um sistema operacional de rede que fornece informações básicas para configuração e gerenciamento da rede.

Para esses experimentos, focou-se na entrega do tráfego de mensagens GOOSE nas redes de subestação, que tem grande restrição temporal. As mensagens GOOSE foram geradas com o gerador de tráfego, representado na Figura 6.2 como GTD, que é descrito a seguir, na Seção 6.2.1.

Foram implementados os componentes `Descoberta de Clientes`, `Tráfego Comum`, `Multicast L2` e `Detecção de Falhas`, todos descritos na Seção 5.2.1. Os componentes

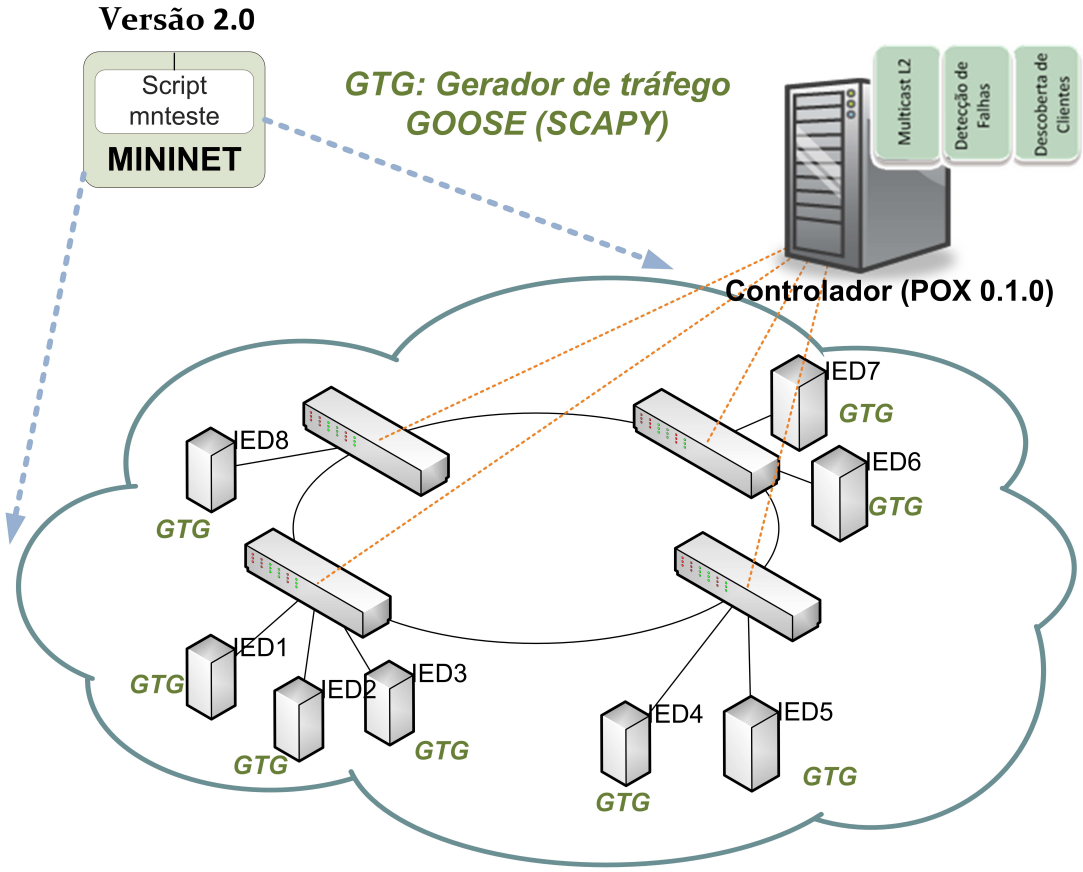


Figura 6.2: Ambiente de implementação do SMARTFlow

`Descoberta de Topologia` e `Spanning Tree` são componentes nativos do POX.

Os experimentos foram emulados usando o Mininet [15] versão 2. O Mininet é uma plataforma flexível para emulação de redes OpenFlow que permite a criação de *hosts*, *switches*, roteadores, controladores e enlaces. Assim pode-se criar a rede e escrever a lógica de controle para manipular seus pacotes além de permitir a interação com o Wireshark e com o POX. Foi criado um módulo no Mininet que constrói topologias LAN de subestações, todas distribuindo os IEDs uniformemente na rede. Este módulo também contém a classe que chama os componentes do SMARTFlow para controlar a rede. Como apresentado na Figura 6.2, foi desenvolvido o *script* `mnteste` no Mininet que é responsável por gerar todos os nós e interconectá-los, assim como por inicializar o controlador e coletar os resultados. O `mnteste` é descrito no Algoritmo 3, e ilustrado na Figura 6.3.

O mnteste utiliza a API do Mininet para:

1. construir a topologia de acordo com o arquivo de configuração do experimento contendo os parâmetros da simulação;

Algoritmo 3: Algoritmo mnteste

Input: $nswitches, nieds, tipo_topologia, neventos, tempo_sim, nrodadas, qtd_GM, controller, tipo_traf$

```
for rodada in range(nrodadas) do
    seta_parametros(tempo, rodadas)
    topo = cria_topologia(nswitches, nieds, tipo_topologia)
    net = Mininet(topo, controller)
    net.start()
    gruposmulticast = geraGM(topo, qtd_GM)
    iedsmininet = net.hosts
    interfacessw = interfaces_ativas(topo)
    for ied in iedsmininet do
        inicia_TCPDUMP(ied)
    end
    for i in interfacessw do
        inicia_TCPDUMP(i)
    end
    for ied in iedsmininet do
        for grupo in gruposmulticast do
            if ied.MAC() == grupo['ied_pub'] then
                geradorGOOSE(grupo['end_gm'], grupo['ied_pub'])
            end
        end
    end
    for evento in range(1, neventos + 1) do
        grupoAleatorio = random.choice(gruposmulticast)
        for i in iedsmininet do
            if i.MAC() == grupoAleatorio['ied_pub'] then
                geradorGOOSE_retransmissao(grupoAleatorio['end_gm'],
                grupo['ied_pub'], evento)
            end
        end
    end
    if trafego_fundo == 1 then
        gera_trafegoFundo(tipo_traf)
    end
    sleep(temp_sim)
    encerra_processos(geradorGOOSE_retransmissao, geradorGOOSE,
    gera_trafegoFundo, net, inicia_TCPDUMP)
end
```

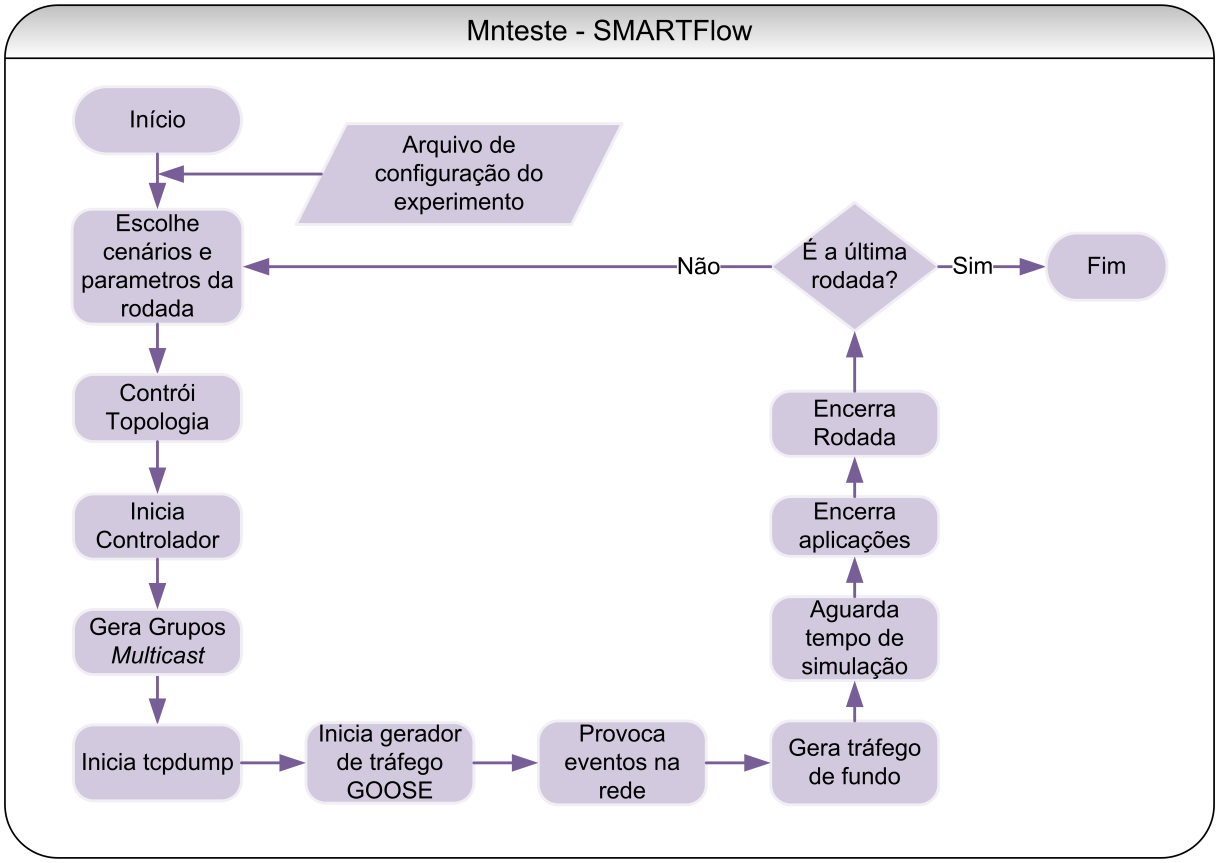


Figura 6.3: Fluxograma do *script mnteste*

2. iniciar o controlador da rede;
3. usar os IEDs criados na rede para gerar os grupos *multicast* do experimento;
4. iniciar o tcpdump para capturar todos os pacotes em todas as interfaces;
5. iniciar geradores de tráfego GOOSE em cada IED publicador da rede;
6. simular eventos na rede elétrica, que se refletem em um aumento do tráfego GOOSE;
7. iniciar outro tipo de tráfego quando necessário;
8. esperar o tempo de simulação;
9. encerrar todos os testes e iniciar nova rodada ou novo cenário.

Para esses experimentos, os grupos *multicast* foram gerados no *script mnteste* ao invés de serem retirados do arquivo .scd da norma IEC 61850. Essa escolha se deu apenas para simplificação do teste devido à quantidade de rodadas. Em uma situação prática, os grupos *multicast* seriam extraídos do arquivo .scd, assim como o tráfego seria gerado pelos próprios IEDs da rede em funcionamento. O publicador do grupo *multicast* e os

receptores são escolhidos de forma aleatória seguindo uma distribuição uniforme para, desta forma, obter resultados não tendenciosos.

As medidas extraídas envolvem atraso, tempos para estabilização da rede, tempos para instalação de fluxo, dentre outros. Essas medidas são discutidas na Seção 6.3. O tcpdump foi usado para capturar os pacotes permitindo que os dados fossem armazenados para serem posteriormente tratados. Para tratamento dos pacotes e para extração destas medidas, *scripts* escritos em Python foram usados como apoio.

6.2.1 Gerador de tráfego GOOSE

Como visto na Seção 2.2.2.2, as mensagens GOOSE e SV trabalham mapeadas diretamente na camada de enlace e têm campos e valores específicos. Para desenvolver um gerador de tráfego fiel aos campos dessas mensagens, é necessária uma ferramenta versátil que permita, por exemplo, trabalhar com campos Ethertype, 802.1p e 802.1Q.

Para gerar esse tráfego nos IEDs foi utilizada a ferramenta Scapy [16]. O Scapy é um programa poderoso de manipulação interativa de pacotes, escrito em Python. Com ele pode-se criar *scripts* para construir e decodificar os pacotes de um grande número de protocolos, enviá-los e capturá-los. O Scapy permite que se construa exatamente o pacote desejado, pois tem um modelo flexível. Portanto, novos campos podem ser criados e dispostos como se preferir e na camada desejada [16].

No caso do *script* desenvolvido, algumas funções e classes da camada 2 são usadas e foi criada uma classe para montar a 'parte GOOSE' do pacote.

Cada classe representa os atributos relativos a uma camada ou subcamada. Por exemplo, a classe `Ether()` contém os campos `src` e `dst` com os endereços MAC de origem e destino respectivamente, e o campo `type` com o tipo do pacote de camada 2.

Para implementar o gerador de mensagens GOOSE deve-se seguir a estrutura padronizada na parte 8-1 da norma [13], ilustrada na Figura 6.4, onde tem-se os seguintes campos[2]:

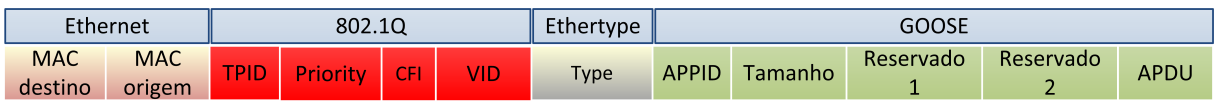


Figura 6.4: Estrutura do Pacote GOOSE

[2]O tamanho dos campos na Figura não obedecem o seu tamanho em bytes ou bits, são apenas ilustrativos.

- **Campos Ethernet:** com MAC de origem e MAC de destino. O endereço de destino deve seguir as recomendações da Tabela 2.2, estando entre 01-0C-CD-01-00-00 e 01-0C-CD-01-01-FF.
- **Campos 802.1Q:** Deve possuir os campos TPID, PCP, CFI e VID, que foram definidos na Seção 2.2.2.2.
- **Campo Ethertype:** Campo que indica que o pacote é GOOSE, possui o valor $0X88B8$.
- **Campos GOOSE:** Contém os seguintes campos:
  - APPID (*Application Identifier*): É usado para selecionar quadros Ethernet que contenham mensagens GOOSE e distinguir essas mensagens.
  - Tamanho: Número de octetos, a partir de APPID.
  - Reservado 1: Reservado para futuras padronizações.
  - Reservado 2: Reservado para futuras padronizações.
  - APDU (*Application Protocol Data Unit Octets*): 12 campos contendo informações referentes a mensagem GOOSE, como número sequência e conteúdos [13].

A implementação do gerador de tráfego GOOSE no Scapy foi feita conforme abaixo:

- **Campo Ethernet:**
  Para este campo foi usada a classe `Ether()`. O campo `dst` foi definido como sendo o endereço do grupo *multicast* do qual o IED fazia parte e para onde queria enviar pacotes. O campo `src` é o endereço MAC do próprio IED. Com isso tem-se:

  ```
  ethernet = Ether(dst='<end. destino>',src='<end. origem>')
  ```

  Observa-se que o campo `type` não será usado nessa camada como será visto mais adiante.
- **Campo 802.1Q:**
  A prioridade no Scapy é definida pela classe `Dot1Q()`. Essa classe possui os seguintes campos:
  - *type* - equivale ao campo TPID, definido sempre como 0x8100, para identificar que é um pacote que contém campos 802.1p.
  - *prio* - equivale ao campo PCP que define os oito níveis de prioridade possíveis, de zero a sete.

- *id* - equivale ao campo CFI, definido na norma sempre como 0 para indicar formato canônico.
- *vlan* - equivale ao campo VID que especifica a qual VLAN o pacote pertence. O valor zero indica que o quadro não pertence a qualquer VLAN, neste caso, tem-se apenas a etiqueta de prioridade.

Com isso tem-se:

```
prio = Dot1Q(type=0x8100,prio=4,id=0,vlan=0)
```

- **Campo Ethertype:**
  O campo Ethertype normalmente é definido dentro da função `Ether()`.Para adicionar esse campo na posição padronizada pela norma, ou seja, após as *tags* do 802.1Q, conforme Figura 6.4, foi criado um campo novo dentro da classe `Dot1Q()`, como mostrado no Algoritmo 4

Algoritmo 4: Alteração da Classe `Dot1Q()`

1 $prio = Dot1Q(type = 0x8100, prio = 4, id = 0, vlan = 0)\ prio.fields_desc[4] = XShortEnumField(\text{“}typegoose\text{”}, 0x88b8, ETHER_TYPES)$

- **Campo GOOSE:** Para esse campo foi necessária a criação de uma nova classe, contendo os campos APPID, Tamanho, Reservado1, Reservado2 e APDU, como mostrado no Algoritmo 5.

Algoritmo 5: Criação da Classe GOOSE

1 $classGoose(Packet) : name = \text{“}Goose\text{”}$
2 $fields_desc =$
$[ByteField(\text{“}APPID\text{”}, 0), ShortField(\text{“}Length\text{”}, 0), ByteField(\text{“}Reserved1\text{”}, 0),$
$ByteField(\text{“}Reserved2\text{”}, 0), XByteField(\text{“}APDU\text{”}, None)]$

Entre os 12 campos do *APDU* estão o `stNum` e o `sqNum`. O campo `stNum` indica que houve uma mudança, ou seja, que a retransmissão em condições estáveis foi interrompida por um evento. O campo `sqNum` é o número de sequência que reinicia com o `stNum`. Assim é possível identificar o pacote. Apenas para simplificação no teste, os campos reservados foram usados como campos `stNum` e `sqNum`. Os campos `APPID` e `Length` ficaram com valor *default*.

A validação dos pacotes gerados foi feita com a ferramenta Wireshark, através da qual foi possível observar a frequência do envio de mensagens e o formato dos pacotes. A Figura 6.5 mostra um exemplo de captura dos pacotes GOOSE gerados.

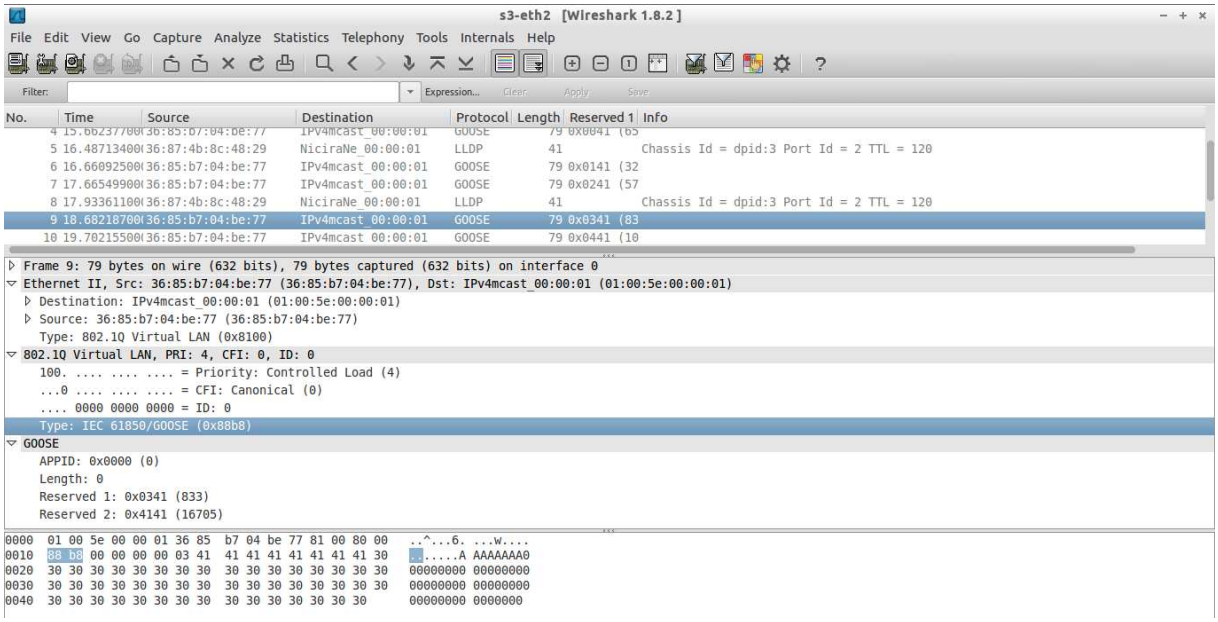


Figura 6.5: Tela de captura de pacotes GOOSE no Wireshark

Esses pacotes são enviados constantemente na rede de $Tmax$ (ciclo máximo) em $Tmax$, sendo $Tmax$ um parâmetro configurado pelo IED. Quando ocorre um evento na rede elétrica que precisa ser notificado a outros IEDs via GOOSE, esse tempo diminui para min_cycle (ciclo mínimo) e os intervalos de retransmissão vão aumentando, seguindo uma curva de transmissão, até alcançar $Tmax$ e retornar a retransmissão anterior já em condições estáveis. Como a norma IEC 61850 não define uma equação para a curva de transmissão, cada fabricante possui a sua equação. Este gerador de GOOSE obedeceu a curva de transmissão do relé de proteção da linha MiCOM fabricado pela Schneider Electric descrita em [19] e ilustrada na Equação 6.1. Todos os parâmetros descritos a seguir foram baseados nos testes apresentados em [19], com valores definidos pela linha de relés MICOM.

$$t_intervalo = min_cycle + (1 - increment)^{N-1} \tag{6.1}$$

Onde:

- $t_intervalo$:
  intervalo até o envio da próxima mensagem GOOSE, enquanto $t_intervalo < Tmax$.

- min_cycle:
  Ciclo mínimo definido como 10ms na linha MiCOM.

- $increment$:

pode ser configurado de 0 até 999. O valor utilizado foi 900, conforme [19].

- N:
  É o número de sequência da mensagem ($SqNum$) .

- $Tmax$:
  definido como 1000ms na linha MiCOM.

A curva de transmissão define o tempo em que a próxima mensagem será enviada. Quando $t_intervalo$ atinge o valor configurado para $Tmax$, o relé de proteção interrompe o envio das mensagens pela equação da curva de transmissão e passa a enviar as mensagens GOOSE em intervalos de tempos constantes [19]. O script do gerador que foi desenvolvido é capaz de modelar essas variações de intervalo. Um exemplo da saída do gerador é mostrado no Apêndice A.

6.3 Experimentos realizados e resultados

Essa seção descreve os experimentos realizados e quais foram os resultados obtidos, assim como a análise acerca destes. Os resultados obtidos nas simulações estão descritos em três partes:

A primeira, descrita na Seção 6.3.1, apresenta os resultados da implementação da proposta em subestações padronizadas pela norma no que diz respeito aos requisitos temporais, verificando se os tempos de resposta atendem as exigências da norma IEC 61850 nas topologias típicas de subestação. Em seguida, na Seção 6.3.2, os tempos de descoberta da rede, cálculo do algoritmo e configuração dos fluxos são analisados. Uma análise destes tempos permite validar o sistema em caso de falha na rede. Por fim, na Seção 6.3.3, será feita uma comparação do SMARTFlow com os componentes de referência do OpenFlow, a fim de verificar o atraso dos pacotes e a carga de controle da rede. Essa comparação ilustra o resultado de um controle feito de forma reativa, como os componentes nativos do OpenFlow, com o modelo proativo do SMARTFlow. Além disso, é feita uma comparação da carga total gerada na rede por um *switch* típico, um *switch* OpenFlow e um SMARTFlow.

Considerou-se, para todos os testes realizados, enlaces de 100 Mbps, pacotes GOOSE com 160 bytes[3], e um intervalo de confiança de 95% nos resultados. A duração dos ex-

[3]Petenel e Panazio [81], realizam experimentos práticos no laboratórios ABB Brasil, e, nos testes efetuados, o tamanho do maior dos pacotes medidos foi de 160 bytes, para pacotes GOOSE contendo

perimentos foi de 100 segundos para cada rodada, incluindo, neste tempo, a estabilização da rede, a configuração dos fluxos, a troca de mensagens e o tempo da simulação. Foram variados parâmetros como quantidade de IEDs na rede, quantidades de *switches* e quantidade de IEDs por grupo *multicast*, que continham no mínimo 2 IEDs e no máximo a quantidade de IEDs total da rede.

Esses parametros são resumidos na tabela abaixo:

Tabela 6.2: Parâmetros usados em cada experimento.

Enlaces	100 Mbps
Pacotes GOOSE	160 bytes
Intervalo de confiança	95 %
Topologias	Anel e Estrela
Duração do experimento	100 segundos

6.3.1 Validação da Norma IEC 61850 com o modelo proposto

Nesta seção, é analisado o atraso das mensagens GOOSE em uma rede IEC 61850. Para essa análise, foram montados 2 cenários: O cenário 1 e o cenário 2 da Tabela 6.1.

O cenário 1 considerou:

- uma LAN com cinco *switches*, variando o número de IEDs de 3 a 12.
- dois diferentes tipos de topologia dos *switches*: anel e estrela.
- cinco grupos *multicast*.
- dez eventos na rede elétrica, em momentos aleatórios, que refletem em um aumento do tráfego GOOSE;

Neste cenário, variou-se a quantidade de IEDs na rede a fim de medir o impacto no atraso das mensagens GOOSE enviadas na rede quando o número de IEDs, e consequentemente o tráfego na rede, é aumentado. Isto ocorre, pois como foi visto na Seção 6.2.1, mensagens GOOSE são enviadas constantemente na rede pelos publicadores dos grupos *multicast*. Como consequência, com o aumento de IEDs na rede, mais mensagens são enviadas aumentando o tráfego na rede.

valores de potência, e 156 bytes, para pacotes GOOSE que transportam sinal de trip de proteção [81]. Nessa dissertação considerou-se o maior valor.

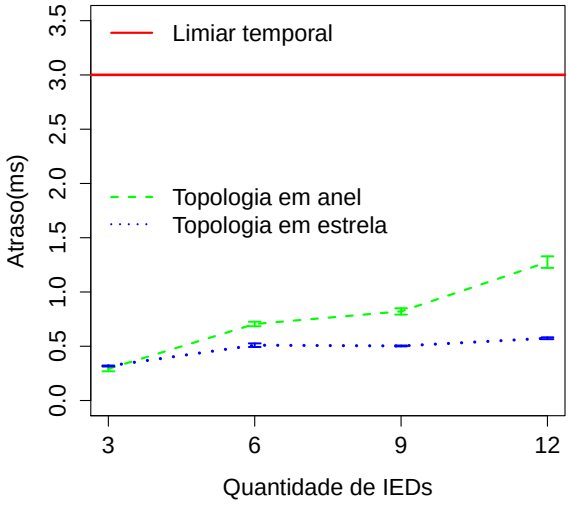


Figura 6.6: Atraso das mensagens GOOSE na rede com os componentes do SMARTFlow, em função do aumento do número de IEDs.

A Figura 6.6 mostra o atraso obtido nos testes de acordo com a carga gerada pelo gerador de GOOSEs iniciado em cada IED da rede. O atraso total, em ambos os cenários, foi calculado como sendo a média dos atrasos até todos os destinos, isto é, o atraso que cada pacote que sai do publicador sofre até chegar em cada receptor no grupo *multicast*.

Com base nesse gráfico, é possível observar que a topologia em estrela, tem um comportamento que se aproxima de uma constante se comparada a topologia em anel. Isso se explica pela própria arquitetura que limita a quantidade de saltos já que todos os nós são ligados a um elemento central. Observa-se também que o atraso das mensagens GOOSE na topologia em anel, aumenta com o aumento dos IEDs, mas se mantendo dentro dos limites estabelecidos pela norma. É importante ressaltar que, ambas as topologias, nos cenários testados, atendem ao requisito temporal mais crítico da norma: a mensagem *trip* (3ms), representada pela linha vermelha no gráfico.

O cenário 2 considerou:

- uma LAN com nove IEDs, e o número de *switches* variando entre um a nove.
- dois diferente tipos de topologia: anel e estrela.
- cinco grupos *multicast* .
- dez eventos na rede elétrica, em momentos aleatórios, que refletem em um aumento do tráfego GOOSE.

Neste cenário, variou-se a quantidade de *switches* na rede, impactando na quantidade de nós, e consequentemente na quantidade de tabelas de fluxo instaladas na rede, assim como impactando diretamente na quantidade de saltos na topologia em anel.

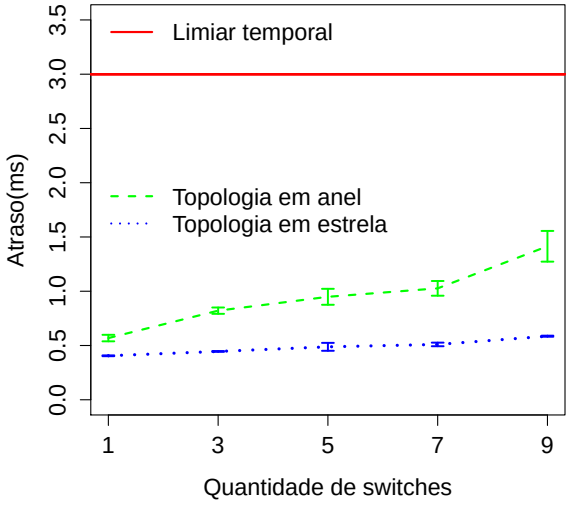


Figura 6.7: Atraso das mensagens GOOSE na rede com os componentes do SMARTFlow, em função do aumento do número de *switches*.

A Figura 6.7 mostra o atraso das mensagens GOOSE obtido nos testes de acordo com o aumento do número de *switches*. Observa-se aqui que, a topologia em estrela continua com um atraso pequeno, se aproximando de uma constante. Além disso, ambas as topologias, anel e estrela, continuam apresentando um ótimo resultado abaixo do limiar da norma.

Em ambos os casos o SMARTFlow se mostra eficiente para as topologias típicas de subestação, atendendo sem dificuldades os requisitos temporais mais críticos.

6.3.2 Desempenho do modelo proposto

Na primeira parte dos experimentos, foi demostrada a eficiência do SMARTFlow no que diz respeito às premissas da norma IEC 61850. O objetivo desta segunda parte é analisar o desempenho do SMARTFlow, no que diz respeito aos tempos de descoberta da rede, cálculo do algoritmo, e configuração dos fluxos, analisando a eficiência dos algoritmos propostos. Para isso, foi testado o cenário 3. Cada topologia, anel e estrela, contém nove IEDs, variando os número de *switches* na rede. Foram gerados fluxos GOOSE em todos os IEDs publicadores da rede, e sete grupos *multicast* foram criados. Os testes

com dois, cinco e sete grupos *multicast* obtiveram resultados muito próximos, por esse motivo foi escolhido o maior tempo dentre esses. Desta forma, dependendo da árvore *multicast* criada, cada *switch* pode ter de 0 a 7 fluxos de mensagens GOOSE. Os gráficos das Figuras 6.8 e 6.9 foram ilustrados separados apenas por serem de ordens diferentes, segundos e milissegundos.

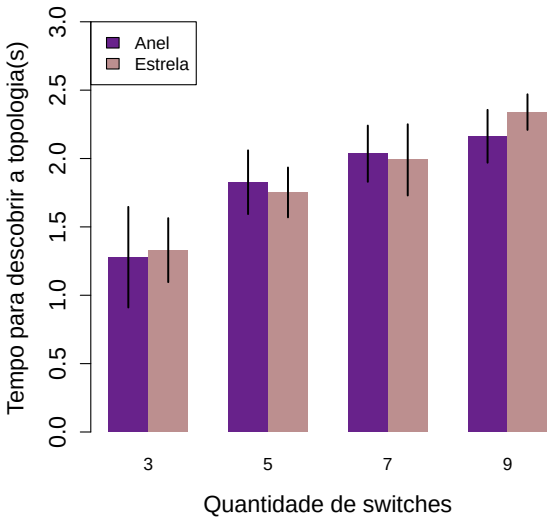


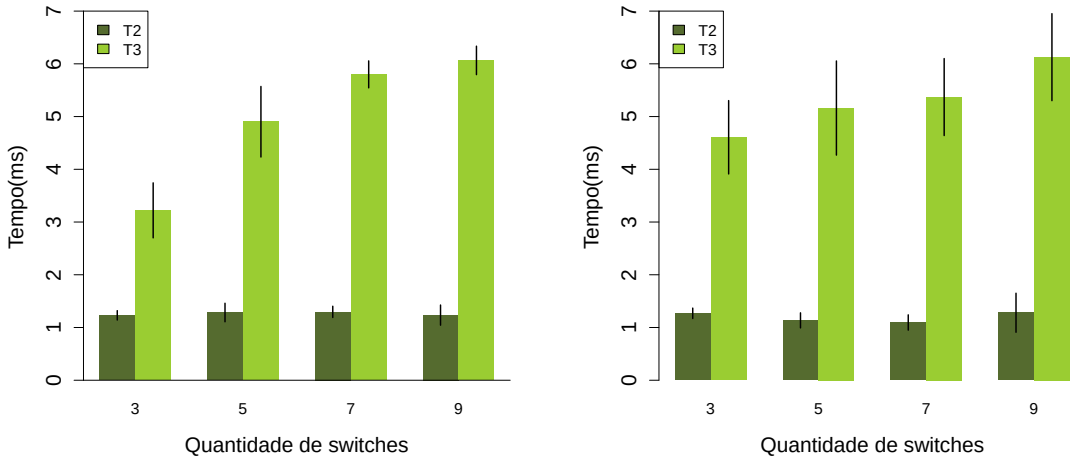
Figura 6.8: Tempo para descobrir a topologia (T1)

Observa-se na Figura 6.8, que o tempo T1 que é o tempo do processo de descoberta da topologia, foi da ordem de segundos. Verifica-se um maior atraso com o aumento da quantidade de nós na rede, comportamento este esperado, já que, quanto maior a rede, mais pacotes de controle têm que ser trocados e mais endereços, portas e localização têm que ser listados.

A Figura 6.9 apresenta os tempos T2 e T3. O tempo T2, que é o tempo que o algoritmo leva para calcular a árvore *multicast*, apresenta um comportamento quase constante, devido ao cenário, que possui poucos elementos e uma topologia fixada, o que leva a um tempo relativamente constante. Se a topologia tivesse, por exemplo, 500 nós provavelmente esse tempo não seria constante, pois o tempo tenderia a aumentar com o número de elementos.

O tempo T3, que é o tempo para configuração dos fluxos, é relativamente maior que o tempo T2, e tende a aumentar com o aumento da rede, já que com mais nós, mais configurações são feitas. Mesmo assim é da ordem de poucos milissegundos.

Ressalta-se que, o tempo T1 ilustrado na Figura 6.8, é da ordem de segundos, enquanto



(a) Topologia em Anel (b) Topologia em Estrela

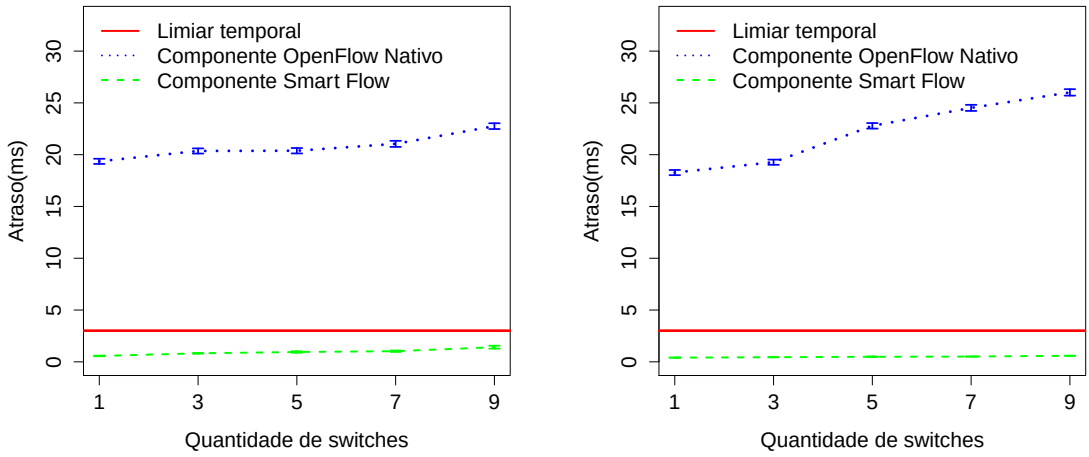
Figura 6.9: Tempo de configuração dos grupos *multicast*, onde T2 é o tempo para cálculo da árvore *multicast* e T3 o tempo para configurar os fluxos

os outros, Figura 6.9, são da ordem de milissegundos. Desta forma, o tempo de descoberta da rede é o mais lento durante o processo.

É importante ressaltar que, considerando uma rede SMARTFLow que não implemente o `fast failover`, em caso de falha, o tempo para restabelecer e calcular novas rotas não inclui T1. Quando uma falha acontece, seja a queda de um enlace ou de um equipamento, o SMARTFlow já conhece a rede e apenas recalcula todas as rotas e configura os fluxos. Assim apenas os tempos T2 e T3 causariam impactos. Desta forma, o tempo para restabelecimento da rede em caso de falha seria da ordem de milissegundos. Por exemplo, para uma topologia Anel com nove IEDs e cinco *switches*, onde T2=1,28ms e T3=4,9ms, em caso de falha, a rede se restabeleceria em 6,2 ms. Esse tempo seria um problema para mensagens muito rápidas de 3ms, porém para todas as outras, que estão acima de 10ms, não causaria impacto. Além disso, conforme será visto na Seção 6.4.1, em comparação com os outros métodos de recuperação de falha, esse tempo só perde para soluções que duplicam os pacotes na rede, diminuindo seu desempenho. Lembra-se também que, as novas versões do OpenFlow, já vêm com um mecanismo de `fast failover` que em caso de queda já é acionado imediatamente, fazendo com que esse tempo para restabelecer seja 0s sem duplicar os pacotes na rede.

6.3.3 Comparação do SMARTFlow com o OpenFlow Nativo

Uma LAN IEC 61850 de uma subestação é um cenário bem conhecido, o que trz muitas vantagens para o controle da rede. Uma vez que se conhece a rede, ou seja, que tipos de mensagens trafegam por ela, quantos e quais grupos se falam na rede, um cenário proativo pode ser implantado. Construir árvores *multicast* de camada 2 proativamente minimiza os tempos da rede e aumenta o desempenho desta, como é demostrado a seguir. Nessa seção são descritos três experimentos. O primeiro compara o atraso das mensagens GOOSE, o segundo a carga de controle e o terceiro a carga total de dados, nos 3 casos a comparação é feita quando a rede é controlada pelo SMARTFlow com a rede controlada pelos componentes nativos do OpenFlow. O último caso inclui a comparação com um *switch* tradicional como será visto adiante.



(a) Topologia em Anel

(b) Topologia em Estrela

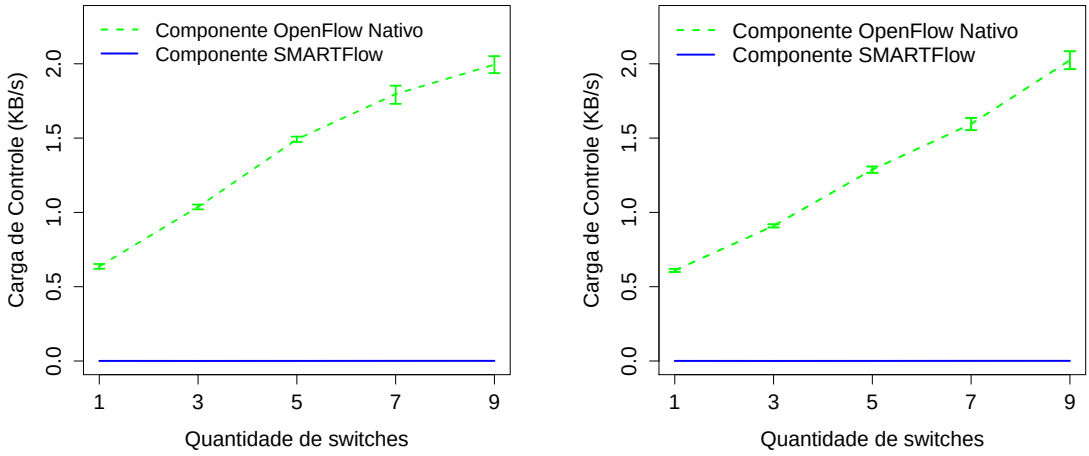
Figura 6.10: Comparação do atraso das mensagens GOOSE na rede, ao se utilizar o SMARTFlow e o OpenFlow Nativo, assumindo IEDs uniformemente distribuídos entre os *switches*.

Para isto, foi testado o cenário 4: duas topologias com cinco grupos *multicast* distintos e nove IEDs, variando o número de *switches* de um a nove em 20 rodadas. Foram estimulados dez eventos na rede, gerando tráfego GOOSE. O intervalo de confiança é de 95%, mas por ser muito pequeno, em alguns pontos do gráfico não se consegue ver a barra de erro, principalmente quando o tempo de aproxima de 1ms e 0ms.

Nota-se que o atraso na rede controlada pelo OpenFlow nativo é muito maior do que o atraso do SMARTFlow, que por exemplo, passa a ser de mais 20 vezes maior que o sistema SMARTFlow. Esse comportamento era esperado pela característica reativa dos

componentes nativos do OpenFlow, isto é, os fluxos são instalados reativamente sempre que um pacote novo chega ao *switch*. Como o comportamento natural de *switches* típicos é tratar o *multicast* como *broadcast*, cada pacote que chega ao *switch* é enviado ao controlador e quando este identifica que é um pacote *multicast*, configura uma entrada de fluxo para enviar o pacote para todas as portas de saída, o que, naturalmente, sobrecarrega a rede e aumenta o atraso dos pacotes.

Ainda com o mesmo ambiente descrito acima, cenário 4, analisa-se também a carga de controle gerada na rede pelos dois sistemas, conforme mostrado na Figura 6.11. Para isso, calculou-se toda a carga de pacotes de controle na rede, como pacotes LLDPs, *packet_ in*, *packet_ out*, etc. Como a rede tem um tempo muito pequeno para estabilizar, ou seja, descobrir a topologia, calcular as rotas e instalar os fluxos, e esse tempo só ocorre uma vez no inicio do teste, os cálculos foram feitos imediatamente após essa estabilização.



(a) Topologia em Anel

(b) Topologia em Estrela

Figura 6.11: Carga de controle na rede após estabilização.

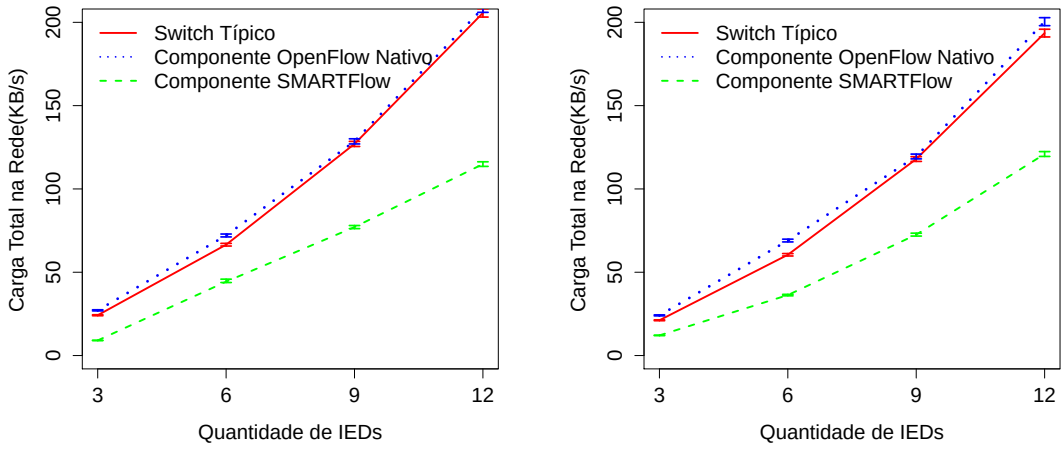
Apesar de parecer que o SMARTFlow não apresenta carga de controle, o mesmo tem valores muito próximos de 0. Essa pouquíssima carga de controle deve-se à natureza proativa do SMARTFlow, que por já conhecer a rede, após a estabilização, não troca mais pacotes com o controlador para o estabelecimento do tráfego GOOSE, a não ser em casos específicos como o de queda de um enlace. Já o OpenFlow nativo troca mensagens durante todo o tempo de forma reativa, o que aumenta consideravelmente a carga na rede.

Por fim, na Figura 6.12, foi avaliada a carga total incluindo a carga de controle no caso dos componentes OpenFlow. Além dos componentes anteriores, OpenFlow Nativo

e SMARTFlow, acrescenta-se um *switch* típico. Para emular os *switches* típicos, criou-se um componente apenas para instalar proativamente regras que tratam o tráfego *multicast* como *broadcast* configurando os fluxos proativamente. Desta forma, simula-se um *switch* comum, onde os fluxos já se encontram definidos encaminhando o pacote por todas as portas.

O experimento considerou os mesmos parâmetros do cenário 1:

- uma LAN com 5 *switches* e IEDs variando entre 3 e 12.
- dois diferente tipos de topologia: anel e estrela.
- cinco grupos *multicast*.
- dez eventos na rede elétrica, em momentos aleatórios, que refletem em um aumento do tráfego GOOSE.



(a) Topologia em Anel (b) Topologia em Estrela

Figura 6.12: Carga total de dados na rede em função do aumento do número de IEDs.

Observa-se na Figura 6.12, que a carga total de dados no OpenFlow Nativo é um pouco mais alta do que a carga do *switch* típico. Isso acontece pois, quando o componente nativo do OpenFlow emula *switches* típicos é acrescida à carga de inundação na rede a carga de controle para instalar o fluxo. Ressalta-se que, a carga de controle, para este cenário, foi muito pequena comparada ao valor total, com isso o comportamento das duas curvas foram muito próximos. Além disso, verifica-se que o SMARTFlow diminui a carga total na rede em até 44% para 12 IEDs na topologia em Anel. Isso acontece, pois o SMARTFlow

calcula uma árvore *multicast* para encaminhar os pacotes evitando a inundação natural de *switches* típicos. Quando o pacote chega ao *switch*, ao invés de ser encaminhado por todas as portas, é enviado apenas para a porta relativa a árvore *multicast*.

A análise realizada mostra que o SMARTFlow é eficiente e atende não só aos requisitos rígidos de tempo da norma IEC 61850 como garante o bom desempenho da rede com uma carga de controle baixa, o que não a sobrecarrega e a torna mais eficiente e confiável.

Além disso, em caso de falha, o SMARTFlow restabelece a rede rapidamente garantindo a alta disponibilidade desta. É importante ressaltar que os protocolos de redundância utilizados atualmente, comentados na Seção 2.3 , Tabela 2.4, apresentam tempos bem mais altos que o SMARTFlow com exceção do PRP e do HSR que apresentam tempo 0s. Porém esses protocolos apresentam restrições como a topologia a ser adotada além de duplicar todos os pacotes enviados na rede, enviando para dois caminhos, podendo interferir no desempenho da rede. Como será visto a seguir, na Seção 6.4.1, as novas versões do OpenFlow trarão benefícios muito maiores nesse sentido.

Portanto, verificou-se que o sistema proposto é eficiente, atendendo aos requisitos da norma, garantindo a disponibilidade da rede e seu bom desempenho. Além disso, notou-se uma pequena vantagem no uso de topologias em estrela se comparada à topologia em anel.

6.4 Análise qualitativa do SMARTFlow

Além dos experimentos realizados, foi feita uma análise qualitativa acerca de dois pontos importantes discutidos nessa dissertação:

- Os principais métodos de recuperação de falha usados atualmente nas redes IEC 61850, listados na Tabela 2.4 da Seção 2.3, e
- a comunicação *multicast* na camada 2, descrita no capítulo 3.

Nas próximas seções, a análise é descrita e os resultados são apresentados.

6.4.1 Métodos de recuperação de Falha e o SMARTFlow

Como foi visto, embora apresentem vantagens, os métodos para recuperação de falha muitas vezes não são a melhor opção, pois atendem a um dos requisitos mas deixam a desejar

no desempenho da rede como um todo. Comparando os métodos mais usados atualmente, o PRP e o HSR, com a recuperação de falha do componente `Detecção de Falhas`, assumindo OpenFlow sem *Fast Failover*, o tempo de recuperação que nos primeiros é dito como 0 s, certamente, é menor que o tempo do `Detecção de Falhas`.

Ressalta-se que, o HSR e o PRP apresentam algumas limitações:

- Só podem ser aplicados em topologias específicas, por exemplo, em topologias em anel de IEDs no caso do HSR.
- No caso do HSR, tem-se a limitação na quantidade de IEDs na rede.
- O processamento de cada IED é duplicado, já que recebe, quase sempre, dois pacotes tendo que aceitar um e rejeitar o outro.
- No caso do PRP, precisam de duas redes distintas e independentes, ou sejam, precisam do dobro de *switches* na rede.
- Podem sobrecarregar a rede pois duplicam todos os pacotes enviados.
- Os IEDs precisam implementar os protocolos.

A comparação é resumida na Tabela 6.3.

Tabela 6.3: Comparação de métodos para recuperação de falha

Protocolo	**Atraso**	**Aplicável a qualquer Topologia**	**Sem duplicação de tráfego ou equipamentos**	**Transparente aos IEDs**
SMARTFlow 1.0	<9 ms	Sim	Sim	Sim
SMARTFlow 1.1	0 s	Sim	Sim	Sim
PRP	0 s	Não	Não	Não
HSR	0 s	Não	Não	Não

Nesse ponto o SMARTFlow apresenta vantagens, pois é independente da topologia, não duplica pacotes na rede, evitando sobrecarga, não tem a necessidade de ser implementado nos IEDs e não necessita de duplicação dos componentes da rede. Porém, ressalta-se o HSR e o PRP são usados e testados em redes já em produção, o que é uma vantagem se comparado á esta proposta.

Destaca-se que, conforme indicado na tabela, as novas versões do OpenFlow permitem que o tempo de recuperação seja também 0 s, ao implementar o conceito de *Fast Failover* e instanciar caminhos distintos. Com isso o SMARTFlow com OpenFlow 1.1 ou superior é a melhor solução.

6.4.2 As soluções *multicast* camada 2 e o SMARTFLow

Nesta seção, realiza-se uma análise qualitativa comparando as características das soluções *multicast* discutidas no Capitulo 3, com as características da solução SMARTFlow. Os resultados são mostrados na Tabela 6.4, onde:

- *multicast* típico se refere ao *multicast* feito em típicos *switches* de camada 2.
- *multicast* estático se refere à forma estática de configuração do multicast.
- GMRP se refere à redes em que este protocolo é aplicado.
- SF 1.0 se refere à proposta SMARTFlow implementada no arcabouço do OpenFlow versão 1.0.0
- SF se refere à proposta SMARTFlow implementada no arcabouço do OpenFlow versão 1.1

Tabela 6.4: Comparação entre as atuais soluções *multicast* e a proposta SMARFlow

Características	***Multicast*** típico	***Multicast*** Estático	**GMRP**	**SF 1.0**	**SF**
Complexidade de configuração	Baixa	Alta	Média	Baixa	Baixa
Dependência de mensagens *Join* e *leave*	Não	Não	Sim	Não	Não
Consumo de Banda pelo tráfego de dados	Alto	Baixo	Baixo	Baixo	Baixo
Carga de Controle	Baixa	Baixa	Alta	Baixa	Baixa
Inundação da Rede	Sim	Não	Não	Não	Não
Controle aprimorado do tráfego	Não	Não	Não	Sim	Sim
Simplicidade e Flexibilidade	Baixa	Baixa	Baixa	Alta	Alta
Convergência Rápida	Sim	Não	Não	Sim	Sim
Tempos de atraso na rede	Alto	Baixo	Baixo	Baixo	Baixo

O consumo de banda pelo tráfego de dados em redes compostas por *switches* típicos é alto devido ao fato dos *switches* típicos tratarem o tráfego *multicast* como *broadcast*.

Uma vantagem do SMARTFlow sobre o GMRP é a ausência de mensagens de controle entre os *switches*. No GMRP, os *switches* são responsáveis por encaminhar pacotes e trocar mensagens de controle para montar a árvore *multicast*. Isto cria uma sobrecarga extra em termos de consumo de banda, pois o controle de mensagens, como *join* e *leave* ou atualizações de árvores, são enviadas através dos mesmos enlaces que o tráfego de dados. Além disso, no SMARTFlow, os IEDs são automaticamente incluídos na árvore *multicast* com base no arquivo SCD, de modo que não é necessário que os IEDs implementem protocolos *multicast* cliente ou que mensagens de atualização da árvore sejam enviadas frequentemente. Mudanças na topologia de rede são automaticamente detectadas no SMARTFlow, que desencadeia a reconfiguração das árvores *multicast*.

Capítulo 7

Conclusões

As *Smart Grids* e um dos seus principais padrões, a norma IEC 61850, ainda enfrentam diversos desafios para prover uma entrega confiável de energia aos consumidores. A norma IEC 61850 tem ganhado cada vez mais espaço, sendo implantada em novas subestações trazendo inúmeros benefícios como redução de custos e de erros humanos, automação, implementação de novas capacidade, dentre outros. Contudo, com a inovação vieram problemas a serem resolvidos.

Essa dissertação identificou e abordou alguns desses problemas, assim como propôs, desenvolveu e avaliou um serviço de gerenciamento e encaminhamento autoconfigurável para redes de subestações IEC 61850 baseada em uma técnica promissora que tem possibilitado um controle mais flexível e orientado às necessidades de cada sistema de comunicação específico, o OpenFlow.

A proposta, chamada SMARTFlow, SisteMa Autoconfigurável para Redes de Telecomunicações IEC 61850 com arcabouço OpenFlow, se destacou pelo alto desempenho que apresentou quando comparada aos mecanismos tradicionais e ao OpenFlow nativo. Os experimentos e as análises realizadas mostraram que o SMARTFlow é eficiente e atende não só aos requisitos rígidos de tempo da norma IEC 61850 como garante o bom desempenho tornando a rede mais eficiente e confiável.

Dentre os módulos da proposta, destaca-se o componente que calcula e configura dinamicamente as árvores *multicast* para encaminhamento de mensagens GOOSE e SV para grupos específicos na rede. Além do atraso com o uso deste componente, foram testadas também a carga de controle, que ficou próxima de zero para a proposta. Isso indica uma vantagem do modelo, já que no GMRP, os *switches* e IEDs são responsáveis por encaminhar pacotes e trocar mensagens de controle para montar a árvore *multicast*

durante todo o tempo, criando uma sobrecarga extra em termos de consumo de banda. Ainda nesse sentido, as soluções tradicionais precisam que os IEDs implantem protocolos específicos, o que não acontece no SMARTFlow, permitindo que os dispositivos sejam mais simples e tenham menos processamento, o que reduz custos.

A carga total da rede também ficou muito abaixo da carga gerada pelo OpenFlow Nativo e pelo *switch* típico, chegando a diminuir em até 44%.

Com relação ao algoritmo para cálculo da árvore *multicast*, o experimento mostrou que o tempo para o cálculo ficou em torno de 1 ms, o que é essencial para agilidade da rede, tanto na configuração automática inicial quanto para o restabelecimento em caso de falhas. Nesse caso, o maior tempo encontrado para calcular e instalar os fluxo foi de 9 ms. Levando em consideração que a proposta pode ser implementada em controladores mais robustos e rápidos, esse tempo poderia diminuir consideravelmente. Apesar de possuir um tempo maior que os métodos HSR e PRP, a proposta não tem as limitações desses métodos, pois não duplica os pacotes na rede, não tem a necessidade de ser implantada nos IEDs, não necessita de duplicação da rede e, consequentemente, duplicação dos dispositivos, e pode ser implementada em qualquer topologia, o que não acontece nos métodos citados.

Destaca-se que, como foi descrito anteriormente, a nova versão do OpenFlow irá permitir que o tempo de recuperação seja também zero ao implementar o conceito de *Fast Failover*, fazendo com que o SMARTFlow seja melhor do que os métodos citados também no tempo de recuperação.

Além disso, a proposta aumenta a qualidade de serviço na rede ao propor um modelo de priorização com mais granularidade e de acordo com as aplicações.

Os testes mostraram, também, que o atraso na rede controlada pelo OpenFlow em sua forma habitual chegou a ser 20 vezes maior do que a rede controlada pelo SmartFlow, passando de 20ms. Com isso, conclui-se que, embora o OpenFlow seja muito promissor com promessa de crescimento e implantação nos próximos anos, no estado atual, em sua forma nativa, não atende a todos os requisitos temporais das redes IEC 61850. Contudo, o atraso na rede controlada pela proposta SMARTFLow não ultrapassou 1,5ms, que é metade do valor mais rígido de tempo estabelecido pela norma IEC 61850 (3ms), provando que a proposta atende adequadamente os requisitos temporais dessa norma.

Outro ganho relacionado ao uso da técnica OpenFlow para criar redes de comunicação de subestações de próxima geração é a programabilidade dos *switches*, que proporciona maior controle e gestão da rede. O controle centralizado não só simplifica o desenvolvi-

mento de algoritmos mais eficientes para controle da subestações, mas também permite a configuração automática da rede de comunicação com base em dados do arquivo SCD. Portanto, o uso de OpenFlow torna mais fácil a configuração automática, além da monitoração do estado da rede, o que é uma tarefa essencial para o funcionamento da subestação.

A análise realizada mostra que o SMARTFlow atende a todos os requisitos das mensagens da norma IEC 61850 seguindo as recomendações padrão nos cenários testados. Além disso, o uso de *multicast* com um controle centralizado e com dados oriundos do arquivo SCD permitem que sejam usados IEDs mais simples e também reduz o tráfego de controle, o tempo de convergência dos algoritmos de controle e o atraso de entrega de dados, quando comparado com os protocolos habituais.

É importante ressaltar que o OpenFlow é uma tecnologia nova e está atualmente em desenvolvimento. Assim, a verdadeira implementação desta tecnologia em redes de subestação depende da liberação de software e hardware estáveis. Nessa dissertação verificou-se que o SMARTFlow é capaz de cumprir os requisitos impostos pela norma IEC 61850 ao usar as aplicações desenvolvidas como uma prova de conceito. De fato, essa tecnologia é uma nova tendência para redes que podem se beneficiar de um controle centralizado e deveria estar no foco de engenheiros de rede para os próximos anos.

7.1 Trabalhos Futuros

Como trabalhos futuros, pretende-se implantar a proposta em uma rede real OpenFlow, e em uma rede tradicional para uma análise mais profunda. Além disso, pretende-se estudar e implantar o *Fast Failover* assim que disponibilizado.

Uma outra questão é o estudo, desenvolvimento e implementação do SMARTFlow nos IEDs. Os IEDs possuem *switches* embarcados o que permite a construção de topologias em anel. Portanto, seria necessária a implementação do OpenFlow nesses *switches* embarcados e a realização de testes de desempenho utilizando o SMARTFlow.

Pode-se também investigar, implementar e avaliar a proposta em um contexto entre subestações, podendo inclusive estender a pesquisa para casos referentes às *Smart Grids*.

Por fim, pretende-se fazer um estudo mais detalhado acerca do modelo de priorização proposto nesse trabalho, detalhando cada critério utilizado, verificando quantos níveis são realmente necessários, definindo a disciplina de escalonamento e outras características necessárias para o estudo.

Referências

[1] D. Bancillon, *Apagão elétrico custou R$45,2 bilhões aos brasileiros*, Correio Brasiliense, http://www.correiobraziliense.com.br/app/noticia/economia/2009/07/15/internas_economia,126861/index.shtml, Acesso em julho de 2012.

[2] S. DiSavino, *Insight: Power reliability will cost Americans more*, Reuters, http://www.reuters.com/article/2011/09/13/us-utilities-sandiego-blackout-idUSTRE78C4UG20110913, Acesso em julho de 2012.

[3] A. Chen, *Berkeley Lab Study Estimates $80 Billion Annual Cost of Power Interruptions*, Research News Berkeley Labs, http://www.lbl.gov/Science-Articles/Archive/EETD-power-interruptions.html, Acesso em julho de 2012.

[4] I. Canales, P. Ibañez, J. Torres, E. Garcia, F. Cobelo, J. Urquiza, and J. Galletero, "INTERUCA Project: UCA interoperability for distributed control within electrical substations," in *Cigré 2004*, no. B5-204, 2004, pp. 1–8.

[5] International Electrotechnical Commission, "IEC 61850-1 Network and Systems in Substations - Introduction and Overview," IEC, Tech. Rep., 2003.

[6] Y. Lopes, R. H. Frazão, D. A. Molano, M. A. dos Santos, F. G. a. Calhau, C. A. M. Bastos, J. S. B. Martins, and N. C. Fernandes, "Smart Grid e IEC 61850 : Novos Desafios em Redes e Telecomunicações para o Sistema Elétrico," in *Minicursos do XXX Simpósio Brasileiro de Telecounicações*, 1st ed. (SBrT), Sociedade Brasileira de Telecomunicações, 2012, pp. 1–44. [Online]. Available: http://sbrt.org.br/sbrt2012/publicacoes/99346_1.pdf

[7] T. Sivanthi and O. Goerlitz, "Systematic real-time traffic segmentation in substation automation systems," in *Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on*, 2013, pp. 1–4.

[8] D. Ingram, P. Schaub, and D. Campbell, "Multicast traffic filtering for sampled value process bus networks," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, 2011, pp. 4710–4715.

[9] R. Moore, R. Midence, and M. Goraj, "Practical experience with IEEE 1588 high Precision Time Synchronization in electrical substation based on IEC 61850 Process Bus," in *Power and Energy Society General Meeting, 2010 IEEE*, 2010, pp. 1–4.

[10] Z. XiCai, W. ShuChao, X. Lei, and F. YaDong, "Practice and trend of DSAS in China," in *Advanced Power System Automation and Protection (APAP), 2011 International Conference on*, vol. 3, 2011, pp. 1762–1766.

[11] J. McGhee and M. Goraj, "Smart High Voltage Substation Based on IEC 61850 Process Bus and IEEE 1588 Time Synchronization," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 489–494.

[12] Institute of Electrical and Electronics Engineers, "802.1Q - Standard for Local and metropolitan area networks: Virtual Bridged Local Area Networks," IEEE, Tech. Rep., 2003.

[13] International Electrotechnical Commission, "IEC 61850-8-1: Specific communication service mapping - Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3," IEC, Tech. Rep., 2011.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[15] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop," in *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks - Hotnets '10.* ACM Press, 2010, pp. 1–6.

[16] *Scapy*, http://www.secdev.org/projects/scapy/, Accesso em novembro de 2013.

[17] *Documentation TCPDUMP*, http://www.tcpdump.org, Accessed in 2013 set.

[18] Y. Yong-hui, W. Lei-tao, and T. Yong-jian, "Research of network transmission of process bus based upon IEC 61850," in *Advanced Power System Automation and Protection (APAP), 2011 International Conference on*, vol. 2, 2011, pp. 1578–1582.

[19] P. A. S. Jr, N. d. C. Fernandez, and Gilberto Morgado, "Topologias de rede ethernet tipo anel para grandes sistemas elétricos baseados na norma IEC 61850," in *XI Seminário Técnico de Proteção e Controle*, 2012, pp. 1–7.

[20] A. Kieling, M. Schmitt, and A. Pereira, "Impacto do desempenho das soluções de gerenciamento de redes de comunicação nas mensagens GOOSE," in *XI Seminário Técnico de Proteção e Controle*, 2012, pp. 1–9.

[21] F. Li, W. Qiao, H. Sun, H. Wan, J. Wang, Y. Xia, Z. Xu, and P. Zhang, "Smart transmission grid: Vision and framework," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 168 –177, Sep. 2010.

[22] V. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. Hancke, "Smart grid technologies: Communication technologies and standards," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 529–539, Nov. 2011.

[23] P. Palensky and D. Dietrich, "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 381–388, 2011.

[24] International Electrotechnical Commission, "IEC 61850-90-8: IEC 61850 object models for electric mobility," IEC, Tech. Rep., 2012.

[25] ——, "IEC 61850-7-420: Basic communication structure - Distributed energy resources logical nodes," IEC, Tech. Rep., 2009.

[26] ——, "IEC 61850-90-5: Use of IEC 61850 to transmit synchrophasor information according to IEEE C37.118," IEC, Tech. Rep., 2012.

[27] ——, "IEC 61850-90-1: Use of IEC 61850 for the communication between substations," IEC, Tech. Rep., 2010.

[28] ——, "IEC 61850-7-510: Basic communication structure - Hydroelectric power plants - Modelling concepts and guidelines," IEC, Tech. Rep., 2012.

[29] S. Olson, "IEC 61850: Where are we headed with substation communications," in *Utility Automation & Engineering T&D*, 2005.

[30] L. T. de Meneses, "Automação da detecção de fraudes em sistemas de medição de energia elétrica utilizando lógica fuzzy em ambientes SCADA," Master's thesis, Universidade Federal do Rio Grande do Norte, RN, Brasil, Apr. 2011.

[31] A. Giani, E. Bitar, M. Garcia, M. McQueen, P. Khargonekar, and K. Poolla, "Smart grid data integrity attacks: Characterizations and countermeasures," *Cyber and Physical Security and Privacy*, pp. 232–237, Nov. 2011.

[32] A. Apostolov and M. Paulino, *Smart Grids - Redes Inteligentes*. Portal O Setor Elétrico, Jun. 2012, ch. Capítulo XII - Interfaces de comunicação no smart grid, pp. 22–32.

[33] A. C. Pereira, R. Abboud, R. Pellizzoni, E. Zanirato, and D. Caceres, "Sistemas de proteção e automação de subestações de distribuição usando a norma IEC 61850," in *Cigre 2009*, no. B5-22, 2009, pp. 1–8.

[34] N. Ziviani, *Projeto de Algoritmos: Com Implementações em Pascal e C.*, 2nd ed. Thomson Learning, 2004.

[35] International Electrotechnical Commission, "IEC 61850-7-1: Communication Network and Systems for Power Utility Automation. Basic Communication Structure - Compatible Logical Node Classes and Data Object Classes," IEC, Tech. Rep., 2010.

[36] ——, "IEC 61850-7-1: Communication Network and Systems for Power Utility Automation. Basic Communication Structure - Principles and Models," IEC, Tech. Rep., 2011.

[37] ——, "IEC 61850-5: Communication requirements for functions and device models," IEC, Tech. Rep., 2003.

[38] M. E. C. Paulino, I. P. Siqueira, and U. A. C. Chesf, "Requisitos para interoperabilidade de IED's e sistemas baseados na norma IEC 61850," in 10º *Seminário Técnico de Proteção e Controle - X STPC*, Oct. 2010, pp. 1–10.

[39] International Electrotechnical Commission, "IEC 61850-7-2: Communication Network and Systems for Power Utility Automation. Basic information and communication structure - Abstract communication service interface (ACSI)," IEC, Tech. Rep., 2010.

[40] ——, "IEC 618509-2: Specific communication service mapping (SCSM)-Sampled values over ISO/IEC 8802-3," IEC, Tech. Rep., 2011.

[41] ——, "IEC 61850-6: Configuration Description Language for Communication in Electrical Substations Related to IEDs," IEC, Tech. Rep., 2009.

[42] I. E. Commission, "IEC 62439-3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR)," in *Industrial Communication Networks - High Availability Automation Networks*, 2010, pp. 1–62.

[43] ——, "IEC 62439-4: Cross-network Redundancy Protocol (CRP)," in *Industrial Communication Networks - High Availability Automation Networks*, 2010.

[44] ——, "IEC 62439-6: Distributed Redundancy Protocol (DRP)," in *Industrial Communication Networks - High Availability Automation Networks*, 2010.

[45] ——, "IEC 62439-2: Media Redundancy Protocol (MRP)," in *Industrial Communication Networks - High Availability Automation Networks*, 2010.

[46] ——, "IEC 62439-5: Beacon Redundancy Protocol (BRP)," in *Industrial Communication Networks - High Availability Automation Networks*, 2010.

[47] J. cheng Tan and W. Luan, "Iec 61850 based substation automation system architecture design," in *Power and Energy Society General Meeting, 2011 IEEE*, July 2011, pp. 1–6.

[48] G. Antonova, L. Frisk, and J.-C. Tournier, "Communication redundancy for substation automation," in *Protective Relay Engineers, 2011 64th Annual Conference for*, April 2011, pp. 344–355.

[49] M. Goraj and R. Harada, "Migration paths for IEC 61850 substation communication networks towards superb redundancy based on hybrid PRP and HSR topologies," in *11th IET International Conference on Developments in Power Systems Protection (DPSP 2012)*. IET, 2012.

[50] P. A. S. Jr, N. d. C. Fernandez, and G. Morgado, "Topologias de rede ethernet tipo anel para grandes sistemas elétricos baseados na norma IEC 61850," in *XI Seminário Técnico de Proteção e Controle*, 2012, pp. 1–7.

[51] I. C. Society, "IEEE 802.1D - IEEE Standart for Local and metropolitan area networks - MAC Bridges," Tech. Rep., 2004.

[52] J. Technologies, *Network Protocol Handbook*, 4th ed., 2007.

[53] D. M. E. Ingram, P. Schaub, R. R. Taylor, and D. A. Campbell, "Network Interactions and Performance of a Multifunction IEC 61850 Process Bus," in *IEEE Transactions on Industrial Electronics*, 2013, pp. 5933–5942. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6381518

[54] M. Seewald, "Building an architecture based on IP-Multicast for large phasor measurement unit (PMU) networks," in *Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES*, 2013, pp. 1–5.

[55] S. Fateri, Q. Ni, G. Taylor, S. Panchadcharam, and I. Pisica, "Design and Analysis of Multicast-Based Publisher/Subscriber Models over Wireless Platforms for Smart Grid Communications," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, 2012, pp. 1617–1623.

[56] S. Seetharaman, "OpenFlow and SDN tutorial," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2012 and the National Fiber Optic Engineers Conference*, 2012, pp. 1–52.

[57] *What Is SDN All About, Then?*, http://www.noxrepo.org/2012/03/sdn/, Accesso em janeiro de 2013.

[58] N. C. Fernandes and L. C. S. Magalhaes, *Network Innovation Through Openflow and Sdn: Principles and Design. Chapter 5:Control and Management Software for SDNs: Conceptual Models and Practical View (ISBN 9781466572096).* Fei Hu. (Org.).

[59] C. E. Rothenberg, M. R. Nascimento, and M. R. Salvador, "OpenFlow e redes definidas por software : um novo paradigma de controle e inovação em redes de pacotes," *Control*, vol. 7, pp. 65–75, 2011.

[60] *Open Networking Summit 2013*, http://www.opennetsummit.org/why-sdn.html, Accesso em maio de 2013.

[61] *Open Networking Foundation*, https://www.opennetworking.org/, Accesso em maio de 2013.

[62] *Clean slate research program*, http://cleanslate.stanford.edu/. Stanford Univesity, Accesso em outubro de 2013.

[63] *OpenFlow Switch Specification, Version (Wire Protocol 0x01).* The OpenFlow Consortium, 2009. [Online]. Available: http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf

[64] *OpenFlow Switch Specification Version 1.1.0 (Wire Protocol 0x02).* The OpenFlow Consortium, 2011. [Online]. Available: https://www.opennetworking.org

[65] *OpenFlow Switch Specification Version 1.2 (Wire Protocol 0x03).* The OpenFlow Consortium, 2011. [Online]. Available: https://www.opennetworking.org

[66] *OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04).* The OpenFlow Consortium, 2012. [Online]. Available: https://www.opennetworking.org

[67] *OpenFlow Switch Specification Version 1.4.0 (Wire Protocol 0x05).* The OpenFlow Consortium, 2013. [Online]. Available: https://www.opennetworking.org

[68] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[69] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using openflow: A survey," *Communications Surveys Tutorials, IEEE*, no. 99, pp. 1–20, 2013.

[70] Stewart, R and Xie, Q and Morneault, K and Sharp, C, "Stream Control Transmission Protocol (SCTP)," Network Working Group- RFC 2960, Tech. Rep., 2000.

[71] *Introducing POX*, http://www.noxrepo.org/2012/03/introducing-pox/, Accesso em setembro de 2013.

[72] *Jaxon:Java-based OpenFlow Controller*, http://jaxon.onuos.org/, Accesso em novembro de 2013.

[73] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," Berkeley, CA, USA,, 2010, pp. 1–14.

[74] T. A. Limoncelli, "OpenFlow: A Radical New Idea in Networking," *Communications of the ACM*, vol. 55, no. 8, pp. 1–7, 2012.

[75] S. J. Vaughan-Nichols, "OpenFlow: The Next Generation of the Network?" *Computer*, vol. 44, no. 8, pp. 13–15, Aug. 2011.

[76] D. Kotani, K. Suzuki, and H. Shimonishi, "A design and implementation of OpenFlow controller handling IP Multicast with fast tree switching," in *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet (SAINT)*, 2012, pp. 60–67.

[77] C. Marcondes, T. Santos, A. Godoy, C. Viel, and C. Teixeira, "CastFlow: Clean-slate multicast approach using in-advance path processing in programmable networks," in *2012 IEEE Symposium on Computers and Communications (ISCC)*, 2012, pp. 94–101.

[78] F. d. O. Silva, M. A. Goncalves, J. H. de Souza Pereira, R. Pasquini, P. F. Rosa, and S. T. Kofuji, "On the analysis of multicast traffic over the entity title architecture," in *Networks (ICON), 2012 18th IEEE International Conference on*, 2012, pp. 30–35.

[79] Y. Wang, J. V. Der Merwe, and J. rexford, "Vroom: Virtual routers on the move," in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking*, 2007, pp. 1–7.

[80] D. M. F. Mattos, N. C. Fernandes, V. T. da Costa, L. P. Cardoso, L. H. M. K. Campista, M. E. M.and Costa, and O. C. M. B. Duarte, in *IEEE - International Conference on the Network of the Future (NOF)*, pp. 52–56.

[81] F. Petenel and C. Panazio, "Análise de uma rede Smart Grid usando a norma IEC 61850 e dados de medições," in *XXX Simpósio Brasileiro de Telecomunicações*, 2012, pp. 13–16.

APÊNDICE A – Exemplo da Saída do Gerador GOOSE

O gerador de tráfego GOOSE resulta na saída mostrada abaixo. É interessante notar que, essa saída proporciona um melhor entendimento dos campos do pacote GOOSE e da sua curva de retransmissão. Nota-se também que, seguindo a curva de transmissão da linha de relés MICOM [19], a retransmissão gera 13 pacotes e estabiliza novamente, voltando a enviar um pacote a cada segundo.

```
1  ..........Enviando pacotes a cada 1000 ms
2
3  Sent 1 packets.
4  ###[ Ethernet ]###
5    dst       = 01:00:5e:00:00:01
6    src       = 1a:1c:1f:00:37:4f
7    type      = 0x8100
8  ###[ 802.1Q ]###
9       prio      = 4
10      id        = 0
11      vlan      = 0
12      type      = 0x88b8
13 ###[ Goose ]###
14         APPID     = 0
15         len       = 0
16         stNum     = 0
17         sqNum     = 0
18 ###[ Raw ]###
19 Enviando pacote a cada 1000 ms
20
21 Sent 1 packets.
22 ###[ Ethernet ]###
23   dst       = 01:00:5e:00:00:01
24   src       = 1a:1c:1f:00:37:4f
```

```
25    type      = 0x8100
26 ###[ 802.1Q ]###
27       prio      = 4
28       id        = 0
29       vlan      = 0
30       type      = 0x88b8
31 ###[ Goose ]###
32          APPID     = 0
33          len       = 0
34          stNum     = 0
35          sqNum     = 1
36 ###[ Raw ]###
37 Enviando pacote a cada 1000 ms
38
39 Sent 1 packets.
40 ###[ Ethernet ]###
41    dst       = 01:00:5e:00:00:01
42    src       = 1a:1c:1f:00:37:4f
43    type      = 0x8100
44 ###[ 802.1Q ]###
45       prio      = 4
46       id        = 0
47       vlan      = 0
48       type      = 0x88b8
49 ###[ Goose ]###
50          APPID     = 0
51          len       = 0
52          stNum     = 0
53          sqNum     = 2
54 ###[ Raw ]###
55 Enviando pacote a cada 1000 ms
56
57 ... Pedaço omitido devido ao tamanho.
58
59 Sent 1 packets.
60 ###[ Ethernet ]###
61    dst       = 01:00:5e:00:00:01
62    src       = 1a:1c:1f:00:37:4f
63    type      = 0x8100
64 ###[ 802.1Q ]###
65       prio      = 4
66       id        = 0
67       vlan      = 0
```

```
68      type      = 0x88b8
69  ###[ Goose ]###
70          APPID     = 0
71          len       = 0
72          stNum     = 0
73          sqNum     = 56
74  ###[ Raw ]###
75
76  OCORREU UM EVENTO!!!!
77
78  Enviado o pacote zero do Evento 1
79
80  Sent 1 packets.
81  ###[ Ethernet ]###
82    dst       = 01:00:5e:00:00:01
83    src       = 1a:1c:1f:00:37:4f
84    type      = 0x8100
85  ###[ 802.1Q ]###
86      prio      = 4
87      id        = 0
88      vlan      = 0
89      type      = 0x88b8
90  ###[ Goose ]###
91          APPID     = 0
92          len       = 0
93          stNum     = 1
94          sqNum     = 0
95  ###[ Raw ]###
96
97  ——————————>>>>>>>>>>>>>>>>>>>>>>> Aqui comecei a retransmissão
98
99  t = 11.0
100 Sent 1 packets.
101 ###[ Ethernet ]###
102   dst       = 01:00:5e:00:00:01
103   src       = 1a:1c:1f:00:37:4f
104   type      = 0x8100
105 ###[ 802.1Q ]###
106     prio      = 4
107     id        = 0
108     vlan      = 0
109     type      = 0x88b8
110 ###[ Goose ]###
```

```
111           APPID      = 0
112           len        = 0
113           stNum      = 1
114           sqNum      = 1
115 ###[ Raw ]###
116
117 t = 11.9
118 Sent 1 packets.
119 ###[ Ethernet ]###
120   dst        = 01:00:5e:00:00:01
121   src        = 1a:1c:1f:00:37:4f
122   type       = 0x8100
123 ###[ 802.1Q ]###
124        prio       = 4
125        id         = 0
126        vlan       = 0
127        type       = 0x88b8
128 ###[ Goose ]###
129           APPID      = 0
130           len        = 0
131           stNum      = 1
132           sqNum      = 2
133 ###[ Raw ]###
134
135 t = 13.61
136 Sent 1 packets.
137 ###[ Ethernet ]###
138   dst        = 01:00:5e:00:00:01
139   src        = 1a:1c:1f:00:37:4f
140   type       = 0x8100
141 ###[ 802.1Q ]###
142        prio       = 4
143        id         = 0
144        vlan       = 0
145        type       = 0x88b8
146 ###[ Goose ]###
147           APPID      = 0
148           len        = 0
149           stNum      = 1
150           sqNum      = 3
151 ###[ Raw ]###
152
153 t = 16.859
```

```
154 Sent 1 packets.
155 ###[ Ethernet ]###
156   dst       = 01:00:5e:00:00:01
157   src       = 1a:1c:1f:00:37:4f
158   type      = 0x8100
159 ###[ 802.1Q ]###
160      prio      = 4
161      id        = 0
162      vlan      = 0
163      type      = 0x88b8
164 ###[ Goose ]###
165         APPID     = 0
166         len       = 0
167         stNum     = 1
168         sqNum     = 4
169 ###[ Raw ]###
170
171 ... Pedaço omitido devido ao tamanho.
172
173
174 t = 623.10662578
175 Sent 1 packets.
176 ###[ Ethernet ]###
177   dst       = 01:00:5e:00:00:01
178   src       = 1a:1c:1f:00:37:4f
179   type      = 0x8100
180 ###[ 802.1Q ]###
181      prio      = 4
182      id        = 0
183      vlan      = 0
184      type      = 0x88b8
185 ###[ Goose ]###
186         APPID     = 0
187         len       = 0
188         stNum     = 1
189         sqNum     = 11
190 ###[ Raw ]###
191
192 t = 1174.90258898
193 Sent 1 packets.
194 ###[ Ethernet ]###
195   dst       = 01:00:5e:00:00:01
196   src       = 1a:1c:1f:00:37:4f
```

```
197   type       = 0x8100
198 ###[ 802.1Q ]###
199       prio       = 4
200       id         = 0
201       vlan       = 0
202       type       = 0x88b8
203 ###[ Goose ]###
204          APPID      = 0
205          len        = 0
206          stNum      = 1
207          sqNum      = 12
208 ###[ Raw ]###
209 ——————————>>>>>>>>>>>>>> Aqui estabilizei a retransmissão com 13 pacotes
      enviados
210 Voltando a condição estável
```