Desenvolvimento de uma Plataforma para Gerenciamento e Configuração de Redes Auto-Organizadas

Maurício Amaral Gonçalves

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

Maurício Amaral Gonçalves

Desenvolvimento de uma Plataforma para Gerenciamento e Configuração de Redes Auto-Organizadas

Tese de doutorado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Pedro Frosi Rosa
Coorientador: Flávio de Oliveira Silva

Uberlândia
2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

G635
2021

Gonçalves, Mauricio Amaral, 1988-
Desenvolvimento de uma Plataforma para Gerenciamento e Configuração de Redes Auto-Organizadas [recurso eletrônico] / Mauricio Amaral Gonçalves. - 2021.

Orientador: Pedro Frosi Rosa.
Coorientador: Flávio de Oliveira Silva.
Tese (Doutorado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.te.2021.274
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Rosa, Pedro Frosi,1959-, (Orient.). II. Silva, Flávio de Oliveira,1970-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Tese de doutorado, 9/2021, PPGCO				
Data:	17 de maio de 2021	Hora de início:	09:00	Hora de encerramento:	12:50
Matrícula do Discente:	11613CCP003				
Nome do Discente:	Maurício Amaral Gonçalves				
Título do Trabalho:	Desenvolvimento de uma Plataforma para Gerenciamento e Configuração de Redes Auto-Organizadas				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Rafael Pasquini - FACOM/UFU, Rodrigo Sanches Miani - FACOM/UFU, José Gonçalves Pereira Filho - UFES, José Marcos Silva Nogueira - UFMG, Flávio de Oliveira Silva - FACOM/UFU (coorientador) e Pedro Frosi Rosa - FACOM/UFU orientador do candidato.

Os examinadores participaram desde as seguintes localidades: José Gonçalves Pereira Filho - Vitória/ES; José Marcos Silva Nogueira - Belo Horizonte/MG; Rafael Pasquini, Rodrigo Sanches Miani, Flávio de Oliveira Silva e Pedro Frosi Rosa - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Pedro Frosi Rosa, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Doutor.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Pedro Frosi Rosa**, **Professor(a) do Magistério Superior**, em 17/05/2021, às 15:40, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rodrigo Sanches Miani**, **Professor(a) do Magistério Superior**, em 17/05/2021, às 21:42, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rafael Pasquini**, **Professor(a) do Magistério Superior**, em 18/05/2021, às 06:50, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Flávio de Oliveira Silva**, **Professor(a) do Magistério Superior**, em 18/05/2021, às 12:08, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **José Gonçalves Pereira Filho**, **Usuário Externo**, em 18/05/2021, às 15:45, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **José Marcos Silva Nogueira**, **Usuário Externo**, em 21/05/2021, às 22:42, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2771964** e o código CRC **FA5A4FF3**.

Referência: Processo nº 23117.031675/2021-21 SEI nº 2771964

Dedico este trabalho à minha filha, Júlia.

Agradecimentos

Sinto-me honrado por ter tantas pessoas às quais devo agradecer pelo apoio na concretização deste trabalho: a todos vocês a minha eterna gratidão.

Inicio por agradecer a Deus por estar sempre ao meu lado e por me dar forças para concluir esta pesquisa.

Agradeço à minha filha, Júlia, pela inspiração e esperança em dias melhores; e à minha esposa, Valquíria, por me apoiar neste desafio e por me dar todo carinho, amor e atenção possível.

Agradeço ao meu pai, Maurício, pelos bons exemplos; à minha mãe, Zilda, pela inspiração; à minha irmã, Angélica, pelo apoio; e ao meu irmão, José Neto, pela alegria que ele renovou.

Agradeço ao meu orientador, Pedro, pelos ensinamentos, pela orientação e pela valorosa amizade conquistada. E agradeço aos meus caros amigos Natal, Daniel e Flávio, pelo companheirismo e pelas incontáveis contribuições a este trabalho.

Agradeço aos meus companheiros de trabalho: Diego, Raul, Johny, Marla, Ronaldo, Alan, Guilherme, Enilton, Rodrigo, Everson, Leonildo, Pedro, Leon, Alécio, Luciana, Fábio, Walace, Jean e Rafael. Sem o apoio e compreensão de vocês esse trabalho não seria possível.

Agradeço aos meus queridos amigos, compadres e companheiros de vida Alessandro e Nathália, pela amizade e compreensão pelos momentos de ausência; e à minha querida afilhada Cecília, a quem tanto estimo e quero bem.

Agradeço aos meus amigos de longa data da minha querida cidade natal: Alípio, Caio, Bruno, Pablo e Leonardo, pelo companheirismo e confiança em mim depositados. E aos meus companheiros da minha turma adotiva, a 71^a turma da curso de Engenharia Elétrica da UFU: Breno, Luiz Eduardo, Everton, Américo, Luiz, Carlos e Matheus, pelo acolhimento, amizade e incentivo.

Agradeço também a todos os colegas pesquisadores do grupo MEHAR, em especial aos meus '*goodfellas*' de projeto SONAr. Agradeço também a todos os meus professores, colegas de graduação e pós-graduação, e assistentes administrativos da FACOM.

"L'homme est condamné à être libre; condamné parce qu'il ne s'est pas lui créé lui-même, et par ailleurs cependant libre parce qu'une fois jeté dans le monde, il est responsable de tout ce qu'il fait."

Jean-Paul Sartre - L'Être et le Néant

Resumo

O gerenciamento das redes de computadores atuais é uma atividade complexa que requer grande esforço humano para atingir os níveis de disponibilidade e desempenho definidos pelas aplicações. Este cenário agrava-se com a introdução de novos componentes de controle por abordagens como SDN e NFV, que embora resolvam aspectos importantes para a flexibilização e segurança da rede introduzem novos pontos de falha. Essa complexidade na perspectiva dos ISPs representa um maior gasto em OPEX, enquanto na dos usuários representa uma maior possibilidade de quebra de SLA, o que em certos casos pode ocasionar riscos de vida e perda financeira. A automação do gerenciamento através dos conceitos de computação autônoma já foi aplicada com sucesso nas redes de telecomunicações através da especificação SON do 3GPP, e representa uma alternativa prática para as redes de computadores. Acredita-se que seja possível alcançar níveis elevados de automação do gerenciamento de rede a partir da utilização de componentes responsáveis pela coleta, análise, aprendizado, tomada de decisão e intervenção na rede. Nesse contexto o presente trabalho se dedica a especificar uma plataforma de gerenciamento de redes auto-organizadas para a automação de operações de configuração em redes definidas por *software*. Esta especificação é apresentada nesta tese com diferentes níveis de visão iniciando pela definição de modelos conceituais responsáveis pela abstração de redes auto-organizadas e serviços de comunicação; depois pela especificação de uma arquitetura para redes auto-organizadas, i.e. SONAr; pela implementação de um *framework* de auto-gerenciamento; e por fim, pela materialização de uma plataforma de auto-configuração. Este trabalho submete esta implementação a testes em cenários emulados que envolvem as operações de inicialização de redes, *plug-and-play* de dispositivos e aprovisionamento de serviços. Os resultados obtidos são promissores ao demonstrarem ganhos tanto qualitativos quanto quantitativos em relação às abordagens tradicionais.

Palavras-chave: Rede Autônomas. Redes Auto-Organizadas. Auto-Configuração. Redes definidas por Software. Funções de Rede Virtualizadas.

Abstract

Managing today's computer networks is a complex activity that requires great human effort to achieve the levels of availability and performance defined by applications. This scenario is aggravated by the introduction of new control components used in approaches such as SDN and NFV, which despite resolving important aspects for the flexibility and security of the network, introduce new points of failure. This complexity from the perspective of ISPs represents more spending on OPEX, and from the perspective of users it represents more risk of breaking the SLA, which in certain cases can cause life risks and financial losses. Management automation through autonomous computing concepts has already been successfully applied to telecommunications networks through the SON specification by 3GPP, and represents a practical alternative for computer networks. We believe that it is possible to achieve high levels of automation of network management through the use of components responsible for the collection, analysis, learning, decision making and intervention in the network. In this context, the present work aims at specifying a self-organized network management platform for the automation of configuration operations in software-defined networks. This specification is presented in this thesis with different levels of vision starting from the definition of conceptual models responsible for the abstraction of self-organized networks and communication services; then by specifying an architecture for self-organizing networks, i.e. SONAr; by implementing a self-management framework; and finally, by the materialization of a self-configuration platform. This work applies the framework implementation to tests in emulated scenarios involving network initialization operations, device plug-and-play and service provisioning. The results obtained are promising in demonstrating both qualitative and quantitative compared to traditional approaches.

Keywords: Autonomous Network. Self-Organizing Networks. Self-Configuration. Software-defined Networking. Network Functions Virtualization.

Lista de ilustrações

Lista de tabelas

Lista de siglas

3G	*3th Generation Mobile Network*
3GPP	*3rd Generation Partnership Project*
5G	*5th Generation Mobile Network*
5G-SA	*5th Generation Mobile Network Stand Alone*
ABM	*Auto-Boot Manager*
AC	*Autonomic Computing*
ACID	***A**tomicity, **C**onsistency, **I**solation e **D**urability*
ACL	*Autonomic Control Loop*
ACoE	*Alarms Collector Entity*
AMQP	*Advanced Message Queuing Protocol*
ANM	*Autonomic Network Management*
AP	*Address Provider*
API	*Application Programming Interface*
AR	*Augmented Reality*
ARP	*Address Resolution Protocol*
BGP	*Border Gateway Protocol*
BI	*Behavioral Intent*
BIIBS	*Behavior Inferred Intent-Based Service*
BOOTP	*Bootstrap Protocol*
BPMN	*Business Process Model and Notation*
BSD	*Berkeley Software Distribution*
BSLE	*Behavior Self-Learning Entity*
BSS	*Business Support System*
CAP	*Consistency, Availability and Partition tolerance*
CCIS	*Communications in Computer and Information Science*
CDN	*Content Delivery Network*
CLI	*Command Line Interface*
CLOSER	*International Conference on Cloud Computing and Services Science*
CMIP	*Common Management Information Protocol*
CMIS	*Common Management Information Service*

CMOT	*CMIP Over TCP/IP*
C-NIM	*Containerized Network Infrastructure Manager*
CoE	*Collecting Entity*
COMMAG	*IEEE Communications Magazine*
CORBA	*Common Object Request Broker Architecture*
CP	*Container Provider*
CPI	*Control Plane Interceptor*
CRAN	*Cloud Radio Access Network*
CRS	*Cognitive Radio System*
DBMS	*Database Management System*
DHCP	*Dynamic Host Configuration Protocol*
DI	*Declarative Intent*
DN	*Distinguished Name*
DNS	*Domain Name System*
DoS	*Denial of Service*
DPI	*Deep Packet Inspection*
DSLE	*Diagnostics Self-Learning Entity*
DSR	*Design Science Research*
DSRM	*Design Science Research Methodology*
EDA	*Event-driven Architecture*
ESB	*Enterprise Service Bus*
ESLE	*Evaluation Self-Learning Entity*
eTOM	*enhanced Telecom Operations Map*
ETSI	*European Telecommunications Standards Institute*
FAB	***F**ulfillment, **A**ssurance, **B**illing*
FACOM	Faculdade de Computação
FCAPS	***F**ault, **C**onfiguration, **A**ccounting, **P**erformance and **S**ecurity*
FDIBS	*Formally Described Intent-Based Service*
FTP	*File Transfer Protocol*
GDMO	*Guidelines for the Definition of Managed Objects*
GNS3	*Graphical Network Simulator-3*
HTB	*Hierarchy Token Bucket*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IA	Inteligência Artificial
IBN	*Intent-Based Networking*
IBS	*Intent-Based Service*
IBSA	*Intent-Based Service Activator*
IBSBD	*Intent-Based Service Behavior Detector*

IBSI	*Intent-Based Service Interpreter*
IBSM	*Intent-Based Service Model*
IBSNLT	*Intent-Based Service Natural Language Translator*
IBSO	*Intent-Based Services Ontology*
IBSRM	*Intent-Based Service Representation Model*
IBSPM	*Intent-Based Service Provisioning Model*
IBSV	*Intent-Based Service Validator*
ICMP	*Internet Control Message Protocol*
IDL	*Interface Definition Language*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IGMP	*Internet Group Management Protocol*
IoT	*Internet of Things*
IP	*Internet Protocol*
IPFIX	*Internet Protocol Flow Information Export*
IS	*Sistema de Informação*
ISC	*Internet Systems Consortium*
ISO	*International Organization for Standardization*
ISP	*Internet Service Provider*
IT	*Tecnologia da Informação*
JSON	*JavaScript Object Notation*
LAN	*Local Area Network*
LCoE	*Logs Collector Entity*
LLDP	*Link Layer Discovery Protocol*
LTE	*Long-Term Evolution*
MAC	*Medium Access Control*
MAN	*Metropolitan Area Network*
MANO	*Management and Network Orchestration*
MCoE	*Metrics Collector Entity*
MEC	*Mobile Edge Computing*
MEHAR	*Mondial Entities Horizontally Addressed by Requirements*
MIB	*Management Information Base*
ML	*Machine Learning*
MPLS	*Multi-Protocol Label Switching*
MTU	*Maximum Transmission Unit*
NAD	*Network Administration Dashboard*
NAT	*Network Address Translation*
NBI	*Northbound Interface*
NDB	*Network Database*

NDP	*Neighbor Discovery Provider*
NDIBS	*Natural Described Intent-Based Service*
NE	*Network Element*
NEM	*Network Event Manager*
NETCONF	*Network Configuration Protocol*
NETMOD	*Network Modeling*
NFV	*Network Functions Virtualization*
NFVI	*Network Function Virtualization Infrastructure*
NGOSS	*New Generation Operations Systems and Software*
NIM	*Network Infrastructure Manager*
NLP	*Natural Language Processing*
NM	*Network Management*
NSB	*Network Service Broker*
NSLE	*Natural Language Processing Self-Learning Entity*
OAM&P	***O**perations, **A**dministration, **M**aintenance **&** **P**rovisioning*
OF-CONFIG	*OpenFlow Management and Configuration Protocol*
OID	*Object Identifier*
OMG	*Object Management Group*
ONF	*Open Networking Foundation*
ONOS	*Open Network Operating System*
OPEX	*Operational Expenditure*
ORB	*Object Request Broker*
OS	*Operating System*
OSI	*Open Systems Interconnection*
OSLE	*Orchestration Self-Learning Entity*
OSPF	*Open Shortest Path First*
OSS	*Operations Support System*
OVSDB	*Open vSwitch Database*
OWL	*Web Ontology Language*
PaaS	*Platform as a Service*
PBNM	*Policy-Based Network Management*
PDU	*Protocol Data Unit*
PPGCO	Programa de Pós-Graduação em Ciência da Computação
PSLE	*Prediction Self-Learning Entity*
PhD	*Doctor of Philosophy*
QoE	*Quality of Experience*
QoS	*Qualidade de Serviço*
RDP	*Resource Data Provider*
REST	*Representational State Transfer*

RFB	*Reusable Functional Blocks*
RFC	*Request for Comments*
RIP	*Routing Information Protocol*
RPC	*Remote Procedure Call*
RSPAN	*Remote Switched Port Analyzer*
RTP	*Real-time Transport Protocol*
RTSP	*Real Time Streaming Protocol*
RUP	*Rational Unified Process*
SARSA	*State–Action–Reward–State–Action*
SBI	*Southbound Interface*
SCE	*Self-Configuration Entity*
SCoE	*Samples Collector Entity*
SDH	*Synchronous Digital Hierarchy*
SDK	*Software Development Kit*
SDN	*Software-defined Networking*
SDN2	*Self-Driving Network*
SDP	*Session Description Protocol*
SeCoMP	*Self-Configuration and Management Platform*
SHE	*Self-Healing Entity*
SLA	*Service Level Agreement*
SLE	*Self-Learning Entity*
SMTP	*Simple Mail Transfer Protocol*
SNMP	*Simple Network Management Protocol*
SOA	*Self-Organizing Network Architecture*
SOM	*Self-Organizing Map*
SONAr	*Self-Organizing Network Architecture*
SONAr-Framework	*Self-Organizing Network Framework*
SONeM	*Self-Organizing Network Model*
SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SoC	*Separation of Concerns*
SOE	*Self-Organizing Entity*
SON	*Self-Organizing Networks*
SOPE	*Self-Optimization Entity*
SPAN	*Switched Port Analyzer*
SPE	*Self-Protection Entity*
SSH	*Secure Shell*
STP	*Spanning Tree Protocol*
TCoE	*Topology Collector Entity*

TCP	*Transmission Control Protocol*
TDD	*Test-driven Development*
TI	Tecnologia da Informação
TMF	*Telemanagement Forum*
TMN	*Telecommunications Management Network*
TNSM	*Transactions on Network and Service Management*
TOM	*Telecoms Operation Map*
TSLE	*Tuning Self-Learning Entity*
UDP	*User Datagram Protocol*
UFU	Universidade Federal de Uberlândia
UML	*Unified Model Language*
URI	*Uniform Resource Identifier*
WAN	*Wide Area Network*
W3C	*World Wide Web Consortium*
WSDL	*Web Services Description Language*
VIM	*Virtualised Infrasctructure Manager*
VNF	*Virtualized Network Function*
VM	*Virtual Machine*
VoD	*Video on Demand*
VoIP	*Voice over IP*
VR	*Virtual Reality*
XML	*eXtensible Markup Language*
XSD	*XML Schema Definition*
YAML	*YAML Ain't Markup Language*
YANG	*Yet Another Next Generation*

Sumário

Capítulo 1

Introdução

O rápido crescimento da Internet, em parte impulsionado pelos avanços dos computadores de uma maneira geral, não foi acompanhado de uma evolução significativa em sua arquitetura (de Oliveira Silva, 2013). De fato, as proporções e as finalidades das redes atuais são muito distintas das idealizadas por *Baran* nos anos sessenta (BARAN, 1964). Pouco aprimoramento foi feito nos principais protocolos da Internet em mais de sessenta de história (PEREIRA, 2012) e as soluções aplicadas em sua concepção começam a se mostrar inadequadas para as necessidades das aplicações atuais, como por exemplo: a necessidade de suportar um número alto de nós imposta pela *Internet of Things* (IoT) (MADAKAM et al., 2015); a baixíssima latência exigida pelas aplicações de *Augmented Reality* (AR)/ *Virtual Reality* (VR) (BASTUG et al., 2017); e, disponibilidade de 99.999% requerida pelas redes futuras (LóPEZ et al., 2017) (e.g. *5th Generation Mobile Network* (5G)).

O endereçamento dos requisitos de alto desempenho e disponibilidade definidos por estas aplicações contribuem para o aumento da complexidade do gerenciamento de rede que hoje ainda depende de um alto nível de conhecimento técnico dos administradores (METZLER, 2011). O aumento da complexidade das redes computadores também guarda relação com a necessidade de utilização de soluções para contornar as limitações da especificação do IPv4, tais como o NAT (EGEVANG; FRANCIS, 1994), MPLS (FAUCHEUR et al., 2002), DNS (MOCKAPETRIS, 1989) e IGMP (FENNER, 2002)). Além disso, as redes de computadores passaram a fazer parte do *design* das soluções de *software*, sobretudo com a utilização de conceitos como CDN (SAROIU et al., 2002) e *Cloud Computing* (ANTONOPOULOS; GILLAM, 2010)), o que torna o gerenciamento ainda mais crítico.

De maneira geral a complexidade de gerenciamento torna os sistemas computacionais difíceis de alterar, manter e operar (BENSON; AKELLA; MALTZ, 2009). Essa dificuldade pode inviabilizar a utilização de um sistema por demandar mais investimento de tempo e dinheiro em sua manutenção, e em alguns casos, pode torná-lo mais susceptível a falhas e eventos adversos devido à imprevisibilidade do seu comportamento e existência

de pontos de falha e gargalos. Uma abordagem bem difundida para lidar com sistemas computacionais complexos consiste em torná-los autônomos por meio dos conceitos de computação autônoma (GANEK; CORBI, 2003).

Sistemas Autônomos contemplam pelo menos quatro propriedades fundamentais: auto-configuração, auto-cura, auto-otimização e auto-proteção (GANEK; CORBI, 2003). Através da auto-configuração, um sistema se adapta dinamicamente a mudanças no ambiente. A auto-cura diz respeito à capacidade de detecção, diagnóstico e reação a falhas com o objetivo de manter o sistema funcionando. A auto-otimização cuida do monitoramento de sistemas e de ajustes de recursos para melhorar o desempenho. E por fim, a auto-proteção refere-se à capacidade de antecipar, detectar, identificar e proteger o sistema. Além destas, existem outras propriedades tradicionalmente aplicadas em Sistemas Autônomos (SIFAKIS, 2019), como por exemplo: auto-orquestração (VERMESAN et al., 2017), que indica que um sistema deve ser capaz de ajustar as próprias políticas e orquestrar os seus componentes; auto-consciência (SADIGHI et al., 2018), que indica que um sistema deve ser capaz de coletar dados, realizar exames, interpretar resultados e diagnosticar problemas; e, auto-aprendizado (STEIN et al., 2018), que é a característica que torna um sistema apto a tomar decisões de forma autônoma através de uma análise de probabilidade com base em cenários anteriores.

O processo de implementação de uma rede autônoma passa pela abstração das propriedades fundamentais de sistemas autônomos e suporte à definição de comportamentos por meio de regras de negócio e políticas administrativas. A solução proposta nesta tese se baseia na utilização de *i) Software-defined Networking* (SDN) (MCKEOWN et al., 2008), que propõe a separação dos planos de Controle e de Dados, com o objetivo de flexibilizar a implementação de novas funcionalidades nas redes; e *ii) Network Functions Virtualization* (NFV) (CUI et al., 2012), que propõe a utilização de máquinas virtuais para a realização de funções de rede tradicionalmente executadas em *appliances*. SDN e NFV são consideradas duas das mais promissoras abordagens para as redes futuras (RAMIREZ-PEREZ; RAMOS, 2016), especialmente a SDN que já conta com amplo suporte da indústria através do protocolo *OpenFlow* (Cisco, 2013), (HP, 2013), (Huawei, 2013), (NoviFlow, 2013).

Self-Organizing Networks (SON) é um conceito explorado há anos na área de telecomunicações, que define elementos de software, protocolos e técnicas padronizadas para a auto-organização da rede com o objetivo de reduzir *Operational Expenditure* (OPEX) e viabilizar serviços de comunicação com requisitos de confiabilidade e mobilidade (3GPP, 2018c). As redes de telecomunicações também possuem uma separação entre o plano de Controle (chamado de Sinalização) e de Dados, que, assim como SDN, viabilizam a implementação de comportamentos por meio de *softwares* implantados em elementos de controle. Hoje SDN é considerada uma tecnologia chave para a implementação das redes de telecomunicação futuras (BLANCO et al., 2017), e trabalhos como Zhang, Xie e Yang

(2015) propõe inclusive a unificação das redes de dados e de telecomunicações por meio de uma única solução baseada em SDN.

A automação do gerenciamento das redes de computadores é um problema ainda em aberto mesmo considerando a aplicação da abordagem SDN (WICKBOLDT et al., 2015). Embora a ideia de separação dos planos de dados e de controle facilite a implementação de aplicações de gerenciamento acoplados à *Northbound Interface* (NBI) dos Controladores, estes mesmos componentes introduzem novos pontos de falha que requerem funções de gerenciamento específicas. Além disso, a capacidade de implantar unidades de *software third-party* através dos controladores torna a rede mais susceptível à *bugs* e *exploits* que podem comprometer a disponibilidade da rede (ABDELSALAM, 2018). Por este motivo muitos trabalhos se dedicam à explorar o *Network Management* (NM) sob o ponto de vista da propriedade de auto-cura, como por exemplo: o LegoSDN que propõe um *framework* para isolar e tolerar aplicações com falhas por meio de *sandboxes*, *checkpoints* e técnicas de restauração (CHANDRASEKARAN; TSCHAEN; BENSON, 2016); o SMaRtLight que propõe uma arquitetura para controladores tolerantes a falhas baseada na utilização de uma base de dados compartilhada pelas aplicações e distribuída na rede (BOTELHO et al., 2014); e diversos outros (FONSECA; MOTA, Fourthquarter 2017).

A propriedade de auto-configuração é pouco explorada nos estudos relacionados ao auto-gerenciamento de rede, sobretudo no universo das redes definidas por *software* (WOJCIECHOWSKI; SIERSZEń; STURGULEWSKI, 2016). Sharma et al. (2013) propôs uma solução para a configuração automática de dispositivos *OpenFlow* com base em DHCP e OF-CONFIG, e que diferentemente do trabalho proposto, não automatiza a inicialização dos elementos de controle. Patil, Gokhale e Hakiri (2015) propõe o InitSDN, um mecanismo de inicialização de redes SDN que se baseia na implementação de serviços de descoberta da topologia, divisão do plano de dados e distribuição do plano de controle, mas que opera sobre uma rede com elementos de rede pré-configurados, enquanto o presente trabalho requer apenas elementos com suporte à DHCP operando com estratégia *learning-switch*. Por fim, Katiyar et al. (2015) propõe uma abordagem para auto-configuração de *switches* SDN em redes híbridas (SDN e Legada) similar à empregada nesta tese, mas que não apresenta resultados experimentais e que não inclui elementos chaves propostos no presente trabalho, como por exemplo o *Control Plane Interceptor* (CPI) que facilita o *handover* entre controladores e o *plug-and-play* de novos dispositivos.

Para fornecer uma solução de gerenciamento autônomo para as redes de computadores com foco em auto-configuração o presente trabalho propõe a *Self-Organizing Network Architecture* (SONAr), uma arquitetura que se baseia em princípios de computação autônoma com componentes integrados por meio de troca de eventos e compartilhamento de dados. A SONAr define componentes responsáveis por um ciclo de gerenciamento que consiste na coleta e análise de dados, tomada de decisão, intervenção na rede e validação de resultados. Essa solução é construída e apresentada nesta tese por meio de diferentes

níveis de abstração, desde modelos conceituais até a materialização de uma plataforma de auto-configuração.

No primeiro nível de abstração da solução o presente trabalho propõe o *Intent-Based Service Model* (IBSM) – modelo para representação e aprovisionamento de serviços de comunicação com base em políticas e restrições com alto nível semântico; e, o *Self-Organizing Network Model* (SONeM) – modelo de organização da rede que propõe uma camada de gerenciamento que opera sobre as camadas de controle e de infraestrutura e que é responsável pela coleta de informações, interpretação de cenários, tomada de decisão e intervenção na rede. No segundo nível de abstração é especificada a SONAr com o objetivo de endereçar as propriedades de sistemas autônomos necessárias para a automação do gerenciamento de rede, tais como: auto-configuração, auto-cura, auto-otimização, auto-proteção, auto-aprendizado e auto-consciência. No terceiro nível de abstração é implementado o SONAr-Framework, uma solução prática que abstrai os componentes SONAr como microsserviços em um ambiente conteinerizado com suporte ao protocolo *OpenFlow*. Por fim, o SONAr-Framework é utilizado na implementação de uma solução *off-the-shelf* para a automação de operações de auto-configuração em redes definidas por *software*, i.e. *Self-Configuration and Management Platform* (SeCoMP).

Na parte final desta tese são detalhados os experimentos conduzidos em uma infraestrutura de rede emulada para a automação da *inicialização* da rede, *plug-and-play* de dispositivos, e configuração de serviços de comunicação. Os resultados demonstram a viabilidade da proposta, comparam diferentes técnicas e abordagens, e apresentam o comportamento satisfatório sob o ponto de vista da complexidade de tempo em relação ao crescimento da infraestrutura.

1.1 Motivação

A concepção de uma rede auto-gerenciável representa uma forma de lidar com a crescente complexidade das redes de computadores cujos procedimentos operacionais ainda são manuais ou semi-automatizados. O conceito de rede auto-organizada já é empregado há anos nas redes de telecomunicações mas ainda é pouco explorado nas redes de computadores. Com a eminente unificação destas redes, faz-se necessária uma abordagem de automação para as redes de computadores que eleve o nível de automação e simplifique a operação. A complexidade no gerenciamento da rede tem impacto direto na qualidade dos serviços de comunicação e gera um aumento no custo operacional que pode inviabilizar negócios e aplicações. Essas redes também são mais susceptíveis a falhas, são difíceis de otimizar e requerem um tempo longo para implantação de novos serviços e equipamentos.

A inicialização da rede, isto é, o processo que compreende: *i)* a descoberta de recursos de infraestrutura (e.g. servidores, dispositivos, *hosts*, aplicações, *appliances*, funções de rede virtualizadas); *ii)* o aprovisionamento de componentes de controle, gerenciamento e

de tratamento do plano de dados (e.g. alocação de recursos computacionais, criação de máquinas virtuais ou contêineres, inicialização dos componentes); e, *iii)* a configuração de recursos, componentes e serviços de comunicação; é uma tarefa complexa e que tradicionalmente requer um esforço operacional humano proporcional à dimensão da rede. Com o advento de IoT e *Smart Cities* espera-se crescimento substancial da infraestrutura de rede o que implica na necessidade de um mecanismo automatizado de inicialização, uma vez que a utilização de uma abordagem manual é inviável. Da mesma maneira, o suporte a novos recursos, sejam eles lógicos ou físicos, precisa ser automático para viabilizar a extensão da rede e prover o requisito de mobilidade de forma transparente.

O aprovisionamento de serviços mais flexíveis às necessidades emergentes das aplicações é um importante requisito para as redes futuras e pode ser alcançado através da propriedade de auto-configuração que é o foco do presente trabalho. Trata-se de um campo aberto e que carece de atenção, sobretudo na concepção de métodos com maior expressividade semântica para descrição e atendimento de requisitos comunicacionais. Com a análise de aplicações atuais e de novos protocolos (em sua maioria na camada de aplicação), percebe-se que há uma lacuna entre os requisitos comunicacionais e os serviços disponibilizados nativamente pela rede. De fato, a arquitetura TCP/IP disponibiliza serviços muito restritos para aplicações que se baseiam normalmente em uma escolha entre dois protocolos fim-a-fim na camada de transporte. Pode-se dizer que há pouca flexibilidade da rede para novos requisitos de aplicações, o que é imprescindível para a evolução das redes.

Os recentes avanços em SDN e NFV oferecem um ambiente profícuo para iniciativas como a deste projeto, o que viabiliza propostas mais disruptivas e liberta pesquisadores do ônus da complexidade de mais baixo nível. Além disso, a adoção dessas abordagens por importantes fabricantes de equipamentos, grandes empresas e diversos centros de pesquisa facilita os aspectos de implantação e minimiza riscos.

1.2 Hipótese

O presente trabalho define como hipótese:

> *"É possível melhorar a velocidade e flexibilidade das operações de gerenciamento através da automação da configuração dos elementos de infraestrutura."*

1.2.1 Problemas da Pesquisa

A hipótese acima está associada os seguintes problemas de pesquisa:

1. Como automatizar a inicialização da infraestrutura de rede e dos componentes de controle e gerenciamento?

2. Como permitir o acoplamento e desacoplamento automático de dispositivos na infraestrutura de rede?.

3. Como traduzir os requisitos de aplicações e corporações em instruções de configuração da infraestrutura de rede?

1.3 Objetivos

O objetivo geral deste trabalho é o de especificar uma plataforma de gerência para redes auto-organizáveis com foco nas operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços.

1.3.1 Objetivos Específicos

Para alcançar o objetivo geral citado acima, propõe-se os seguintes objetivos específicos:

1. Especificar modelos conceituais de redes autônomas com serviços flexíveis aos requisitos comunicacionais atuais e futuros;

2. Descrever uma proposta de arquitetura de redes auto-organizadas e investigá-la por meio de um estudo de caso com operações de configuração;

3. Implementar uma solução prática e flexível para a automação do aspecto de configuração no gerenciamento de rede;

4. Levantar requisitos comunicacionais funcionais e não-funcionais das aplicações, usuários e provedores de serviço;

5. Formalizar de maneira ontológica uma definição de serviço comunicacional flexível; e,

6. Investigar a efetividade a solução proposta por meio de experimentação com cenários que envolvam a configuração de componentes, dispositivos e serviços.

1.4 Contribuições

Este trabalho propõe uma solução com artefatos que representam contribuições independentes para aspectos relacionados ao gerenciamento de redes. Abaixo estes artefatos são descritos em termos de funcionalidades:

1. Modelo conceitual para representação e aprovisionamento de serviços de comunicação flexíveis e com definições de alto nível.

2. Arquitetura para concepção de redes auto-organizadas através de componentes responsáveis pela abstração das propriedades[1] de sistemas autônomos;

3. Plataforma de auto-configuração para a automação das operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços em redes definidas por *softwares*.

Este trabalho almeja ser uma base para trabalhos futuros, e dessa forma, os artefatos da solução listados como contribuições acima visam propiciar a extensão da pesquisa apresentada nesta tese e formalizar o tema de gerenciamento de redes auto-organizadas. Acredita-se que as discussões iniciadas aqui contribuam para a concepção de redes autônomas flexíveis, capazes de prover serviços adequados às necessidades dos usuários e que garantam princípios de disponibilidade e segurança sem intervenção humana.

1.5 Organização da Tese

O presente documento está organizado da seguinte forma:

❑ Capítulo 1 - *Introdução*

Introduz o tema de Gerenciamento de Rede e os conceitos básicos de Sistemas Autônomos; apresenta o problema abordado e propõe uma solução; descreve brevemente o estado da arte; apresenta a motivação do trabalho de maneira mais formal; define a hipótese; lista os problemas de pesquisa; define os objetivos da tese; lista as contribuições; e, detalha a organização desta tese.

❑ Capítulo 2 - *Fundamentação Teórica e Contextualização da Pesquisa*

Apresenta os conceitos de Gerenciamento de Rede, Computação Autônoma e Rede Auto-Organizada; discorre sobre o estado da arte e requisitos de auto-gerenciamento; e, introduz as tecnologias habilitadoras utilizadas na solução.

❑ Capítulo 3 - *Método de Pesquisa*

Apresenta a metodologia aplicada no desenvolvimento desse trabalho, explorando detalhes sobre a revisão da literatura, planejamento da pesquisa, ferramentas e técnicas de desenvolvimento.

❑ Capítulo 4 - *Proposição dos Modelos Conceituais Intent-Based Service Model (IBSM) e Self-Organizing Network Model (SONeM)*

Modela uma solução de alto nível de abstração para o problema de pesquisa baseada na proposição dos modelos conceituais IBSM e SONeM para a representação de serviços baseados em intenções e redes auto-organizadas.

[1] auto-configuração, auto-cura, auto-otimização, auto-proteção, auto-orquestração, auto-consciência e auto-aprendizado

- Capítulo 5 - *Visão Geral da Self-Organizing Network Archtecture (SONAr) e Aplicação na Automação de Operações de Configuração*

  Especifica a Arquitetura SONAr, detalhando componentes de gerenciamento e estratégias gerais de integração e implantação; e, apresenta um estudo de caso de automação dos principais processos de auto-configuração com base na arquitetura proposta.

- Capítulo 6 - *Implementação do Self-Organizing Framework (SONAr-Framework) e da Self-Configuration and Management Platform (SeCoMP)*

  Apresenta em detalhes o SONAr-Framework que materializa a SONAr e define novos componentes úteis o gerenciamento da rede; e, apresenta a SeCoMP que estende o SONAr-Framework para a automação das operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços em redes definidas por *software*.

- Capítulo 7 - *Experimentos e Análise dos Resultados*

  Descreve os experimentos; apresenta cenários de teste; detalha a infraestrutura utilizada; e, analisa os resultados obtidos;

- Capítulo 8 - *Conclusão*

  Conclui o trabalho, avalia os problemas de pesquisa, ressalta as suas contribuições e apresenta as perspectivas futuras.

Capítulo 2

Fundamentação Teórica e Contextualização da Pesquisa

Este capítulo almeja consolidar uma base de conhecimento para o entendimento da solução proposta nesta tese, o que é feito por meio da apresentação de conceitos, fundamentos, marcos históricos, tecnologias e trabalhos correlatos. A seção 2.1 apresenta detalhes sobre a disciplina de 'Gerenciamento de Rede', bem como conceitos, técnicas e perspectivas para redes futuras. A seção 2.2 discorre sobre os fundamentos de 'Computação Autônoma' e conceitua as propriedades de automação (também conhecida como propriedades *self-**). Nesta mesma seção é introduzido o conceito de 'Rede Auto-Organizada', que aplica os fundamentos de Computação Autônoma na automação do Gerenciamento de Rede; o SON, uma abordagem de automação de rede de telecomunicações proposta pelo 3GPP; e, as recentes iniciativas da indústria IBN e SDN2, propostas pelos fabricantes de equipamentos *Cisco* e *Juniper* respectivamente, que visam a automação e flexibilização das redes de computadores. A seção 2.2.4 levanta os principais requisitos de auto-gerenciamento de aplicações, redes e operadores futuros. A seção 2.3 apresenta detalhes sobre abordagens atuais que podem ser utilizadas na implementação de uma solução de auto-gerenciamento. Por fim, a seção 2.4 discorre sobre o estado da arte e apresenta trabalhos correlatos relacionados ao auto-gerenciamento e à auto-configuração.

2.1 Gerenciamento de Rede

As redes de computadores e de telecomunicações moldaram as organizações empresariais e a própria sociedade (CHARAN, 1991) permitindo a criação de novos negócios, flexibilizando a troca de informações e ampliando o alcance de sistemas e pessoas. Por outro lado, organizações e sociedade se tornaram dependentes de serviços de rede que passaram a ser utilizados nas mais diversas áreas, tais como: finanças, segurança e saúde. A indisponibilidade ou deterioração de serviços de rede podem ocasionar grandes perdas

materiais e riscos de vida, o que torna imprescindível a adoção de mecanismos para mitigar riscos e minimizar impactos negativos. Nesse contexto, o *Gerenciamento de Rede* é um mecanismo fundamental para manter a rede funcionando e prover serviços de comunicação em níveis e requisitos mínimos. Quando bem executado, o gerenciamento de rede ajuda a manter custos sob controle, otimiza a utilização de recursos de rede, melhora a qualidade de serviços e aumenta o retorno sobre o investimento; por outro lado, um gerenciamento não efetivo pode causar a degradação dos serviços, subutilização da infraestrutura, perda de investimento e redução de lucro (CLEMM, 2006).

O gerenciamento facilita o acesso a serviços de rede, bem como garante os requisitos comunicacionais, o que livra o usuário final de detalhes técnicos como protocolos e tecnologias. Isso quer dizer que a alocação de recursos, a operação da rede e o aprovisionamento do serviço, atividades contempladas pelo gerenciamento de rede, ocorrem de forma transparente para usuário (CLEMM, 2006). De fato, com a melhoria dos serviços de rede, usuários têm se preocupado cada vez menos com a rede e mais com as aplicações, que por sua vez têm definido novos requisitos comunicacionais, tais como baixa latência, alta disponibilidade e alto volume de dados (BERGSTRA; BURGESS, 2011).

O *Gerenciamento de Rede* lida com diferentes aspectos, desde o nível técnico que envolve operações de campo tais como a implantação e configuração de equipamentos; até o nível corporativo que envolve interesses estratégicos e acordos comerciais. Essas diferentes visões sobre o gerenciamento são tratadas na literatura como *Dimensões de Gerenciamento*, o que permite uma análise completa sobre o tema de *Gerenciamento de Rede*. O Apêndice A apresenta uma análise sobre as dimensões propostas por Clemm (CLEMM, 2006) e Hegering (HEGERING, 1999).

2.1.1 Definição de Gerenciamento de Rede

Existem diferentes definições de *Gerenciamento de Rede* na literatura que variam de acordo com pontos de vista. Em geral, o gerenciamento de rede contempla os processos de planejamento, controle, provisionamento, coordenação e monitoramento de recursos de rede (DING, 2016). As atividades desses processos variam de acordo com o tipo de rede, foco de gerenciamento e nível organizacional.

Hegering (1999) define o *Gerenciamento de Rede* como o conjunto de medidas que garantem a eficiência e efetividade das operações de sistema conectados e recurso de rede em harmonia com objetivos corporativos.

Clemm (2006) define *Gerenciamento de Rede* como o conjunto de técnicas, ferramentas e procedimentos utilizados nas atividades de operação, administração, manutenção e aprovisionamento de sistemas conectados em rede.

Subramanian (2010) descreve *Gerenciamento de Rede* como a operação, administração, manutenção e aprovisionamento da rede e de seus serviços com objetivos definidos em

diferentes perspectivas, tais como garantia de qualidade (na perspectiva do usuário) e controle de custos (na perspectiva do provedor).

2.1.2 Padrões, Modelos e *Frameworks* de Gerenciamento de Rede

O primeiro padrão de *Gerenciamento de Rede* surgiu com a ISO 7498-4 (ISO/IEC, 1989) que especificou um *Framework* de Gerenciamento como parte do Modelo de Referência *Open Systems Interconnection* (OSI) (DAY; ZIMMERMANN, 1983) que define mecanismos para monitoramento, controle e coordenação de sistemas conectados divididos em cinco áreas funcionais. Este modelo de gerenciamento ficou conhecido como FCAPS (***F**ault, **C**onfiguration, **A**ccounting, **P**erformance and **S**ecurity*) e se tornou um padrão para a indústria e academia (STEVENSON, 1995). As cinco áreas definidas pelo FCAPS são:

- *Gerenciamento de Falhas* (F): engloba detecção, isolamento, notificação, correção e documentação de problemas na interconexão de sistemas;
- *Gerenciamento de Configuração* (C): compreende a definição de parâmetros de configuração, alocação de recursos, coleta de dados e implantação de funcionalidades;
- *Gerenciamento de Contabilização* (A): prevê mecanismos para medir o uso de recursos e custos envolvidos;
- *Gerenciamento de Desempenho* (P): permite avaliar o comportamento dos recursos e a efetividade dos serviços de comunicação; e,
- *Gerenciamento de Segurança* (S): aplica políticas de segurança e identifica/reporta eventos relevantes.

O Gerenciamento de Redes de Telecomunicações se consolidou com a recomendação ITU-T M.3010 (ITU-T, 1992) – parte da série M.30 – que propôs o TMN (*Telecommunications Management Network*), que é uma especificação que abrange diversos aspectos relacionados às redes de telecomunicações, passando pelas áreas de planejamento, implantação, operação e administração. A principal contribuição do TMN na área de gerenciamento está na definição de uma terminologia amplamente aceita e na proposição de um modelo de referência organizado em camadas que são listadas abaixo (PRAS et al., 1999):

- *Gerenciamento de Elemento*: gerenciamento de elementos de rede com o objetivo em mantê-los funcionando de forma individual;
- *Gerenciamento de Rede*: gerenciamento de um conjunto de elementos de rede com o objetivo de manter a rede de uma maneira geral funcionando;

- *Gerenciamento de Serviço*: gerenciamento dos serviços providos pela rede com o objetivo de garantir o correto funcionando de acordo com os requisitos especificados;
- *Gerenciamento de Negócio*: gerenciamento do ponto de vista da aplicação prática da rede com o foco em questões de negócio como: cobrança, custo, retorno e atendimento ao cliente;
- *Elemento de Rede*: camada de mais baixo nível na qual o elemento em si é responsável por seu funcionamento e precisa coletar métricas, reportar erros e prover dados para o gerenciamento em níveis superiores.

FCAPS e TMN são comumente referenciados de maneira complementar como ilustrado na Figura 1 que apresenta os conceitos de ambas as abordagens correlacionados através de camadas ortogonais de uma pirâmide.

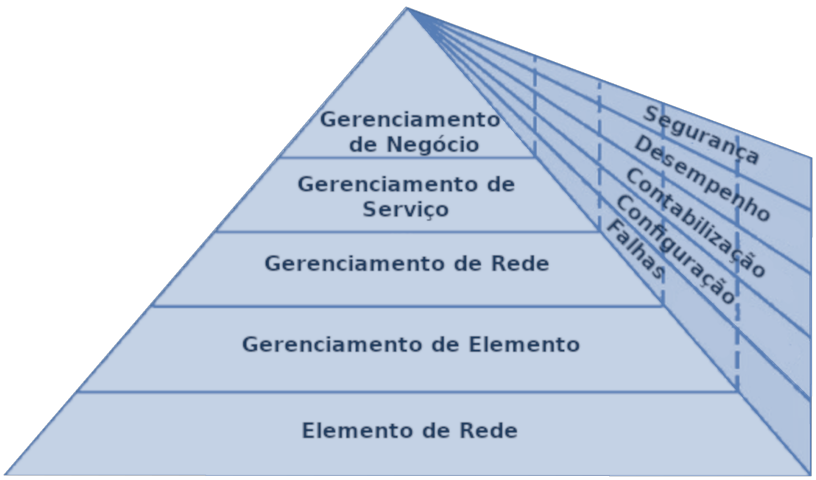


Figura 1 – Visão Tridimensional do Modelo de Gerenciamento TMN refinado com o FCAPS.

Fonte: Adaptado de Clemm (2006).

Uma alternativa ao FCAPS amplamente aceita por provedores de serviços de telecomunicação (CLEMM, 2006), é o OAM&P (***O****perations,* ***A****dministration,* ***M****aintenance &* ***P****rovisioning*) que especifica o gerenciamento de rede baseado em quatro funções (KU, 1998):

- *Operação* (O): consiste em manter a rede funcionando e garantir os níveis de serviços acordados através de monitoramento, diagnóstico e correção de problemas;
- *Administração* (A): responsável por gerir recursos de rede, sejam eles físicos (dispositivos, servidores, cabos) ou lógicos (endereços IP, vlans, *softwares*), controlar custos de operação, planejar alterações na infraestrutura e garantir o retorno sobre o investimento;

- *Manutenção* (M): responsável por prevenir incidentes através de técnicas como gerenciamento de obsolescência, atualização/aquisição de recursos de software/hardware e reconfigurações preventivas ou programadas;

- *Aprovisionamento* (P): responsável por preparar a infraestrutura de rede para prover um serviço de comunicação através de funções como configuração, instalação, alocação e inicialização.

A Tabela 1 apresenta de maneira geral como as funções do FCAPS e do OAM&P se relacionam. Nesta Figura o 'X' indica que há forte relação entre as funções, '(X)' indica que há alguma relação entre as funções, e '–' indica que há pouca ou nenhuma relação entre as funções.

	F	**C**	**A**	**P**	**S**
O	(X)	–	–	(X)	–
A	–	–	X	(X)	(X)
M	X	(X)	–	X	X
P	–	X	–	–	–

Tabela 1 – Relação entre o FCAPS e o OAM&P.

Fonte: Adaptado de Clemm (2006).

O TMF (*Telemanagement Forum*)[1] (CLEMM, 2006) propôs um *framework* para o gerenciamento de redes de telecomunicações conhecido como *Telecoms Operation Map* (TOM) (TMF, 1998) que define o ciclo de vida de gerenciamento em três estágios conhecidos como FAB (***F**ulfillment, **A**ssurance, **B**illing*). Cada estágio tem seus próprios requisitos definidos de acordo com a camada de gerenciamento:

- *Cumprimento* (F): responsável pelas atividades relacionadas à implantação do serviço, tais como alocação de recursos, configuração de dispositivos e instalação da infraestrutura;

- *Garantia* (A): responsável por assegurar que o serviço atende a requisitos acordados através de atividades como monitoramento, correção e otimização;

- *Cobrança* (B): responsável pela contabilização da utilização do serviço de maneira a permitir a cobrança adequada.

A Tabela 2 apresenta de maneira geral como as funções do FCAPS e do FAB se relacionam. Nesta Figura são utilizados os valores 'X', '(X)' e '–' para indicar nível de relação entre as funções (do maior para o menor) da mesma maneira que na Tabela 1.

[1] TMF é um consórcio de companhias de telecomunicação criado em 1988

	F	**C**	**A**	**P**	**S**
F	–	X	–	–	–
A	X	–	–	X	X
B	–	–	X	–	–

Tabela 2 – Relação entre o FCAPS e FAB.

Fonte: Adaptado de Clemm (2006).

Anos depois da especificação do TOM, o TMF propôs o NGOSS (*New Generation Operations Systems and Software*) (STRASSNER et al., 2004), um *framework* com a missão de especificar uma arquitetura agnóstica para provisão de sistemas de *Operations Support System* (OSS) e *Business Support System* (BSS). NGOSS utiliza uma versão aprimorada do TOM chamada de *enhanced Telecom Operations Map* (eTOM) (KELLY, 2003). Essa melhoria, no entanto, não adicionou novidade ao FAB.

2.1.3 Protocolos e Técnicas de Gerenciamento e Comunicação

O *Common Management Information Protocol* (CMIP) é um protocolo definido pela ISO 9596 (ISO/IEC, 1991) como parte integrante do *Framework* de Gerenciamento do Modelo de Referência OSI (DAY; ZIMMERMANN, 1983) que provê os serviços especificados pelo *Common Management Information Service* (CMIS) (ISO/IEC, 1990): GET, SET, CREATE, DELETE, ACTION, EVENT_REPORT de "objetos gerenciados", que são descritos por *Guidelines for the Definition of Managed Objects* (GDMO) e identificados por *Distinguished Names* (DNs). O CMIP também provê mecanismos de autorização, controle de acesso, *logs* de segurança e emissão de eventos flexíveis, e é considerado uma opção mais completa (e complexa) de protocolo de gerenciamento (BLACK, 1994). Este protocolo se tornou o padrão para as redes de telecomunicações depois da recomendação ITU-T X.700 (ITU-T, 1993) e faz parte da especificação do *Telecommunications Management Network* (TMN). Existe também uma variação voltada para redes de computadores conhecida como *CMIP Over TCP/IP* (CMOT), mas que acabou sendo pouco utilizada devido a popularidade do SNMP (apresentado a seguir).

O *Simple Network Management Protocol* (SNMP) é considerado o protocolo padrão para o gerenciamento de dispositivos de redes TCP/IP (STALLINGS, 1998), e por este motivo, ele é amplamente suportado for equipamentos e sistemas finais nestas redes. Este protocolo foi especificado formalmente pela *Internet Engineering Task Force* (IETF) na RFC 1122 (BRADEN, 1989). Os sistemas que suportam SNMP atuam como servidores com suporte aos métodos GET e SET, e são aplicados sobre um conjunto de informações (chave/valor) definidas por *Management Information Base*s (MIBs) e identificadas por *Object Identifiers* (OIDs). O SNMP também a utilização de eventos chamados de "SNMP Traps", que permitem o envio de *Protocol Data Unit*s (PDUs) dos dispositi-

vos gerenciados aos sistemas de gerenciamento sem que uma solicitação (GET) tenha sido feita. Este protocolo aplica mecanismos básicos de autorização (diferentemente do CMIP), que se baseiam na definição de "*communities*" (identificadores de grupos/perfís que servem como chaves de acesso). Sendo assim, para buscar uma informação de um dispositivo com suporte ao SNMP é necessário escolher um OID (de acordo com a MIB), obter uma *community* com acesso a esse OID e utilizar o método GET. A versão básica de SNMP é conhecida por SNMPv1; já a segunda versão que traz melhorias de desempenho e segurança e altera a sintaxe de algumas primitivas (e.g. SNMP *Traps*) é a SNMPv2c; e por fim, a terceira e mais recente versão é a SNMPv3 que trouxe ainda mais melhorias aos aspectos de segurança com a introdução de técnicas de criptografia e uma melhor estratégia de autenticação/autorização (STALLINGS, 1998).

O *Common Object Request Broker Architecture* (CORBA) é uma solução da *Object Management Group* (OMG) voltada para a comunicação em sistemas distribuídos heterogêneos, e que por este motivo, foi muito empregada pelos primeiros sistemas de gerência (SIEGEL; D., 2000). O CORBA é basicamente um mecanismo de *Remote Procedure Call* (RPC) que permite a utilização de métodos de sistemas externos com uma visão local. Para isso, o CORBA define o *Object Request Broker* (ORB), um módulo de *middleware* que atua como intermediário entre os clientes e os objetos. A definição das interfaces do CORBA é papel da *Interface Definition Language* (IDL), que facilita a criação automática de classes/estruturas comumente chamadas de "*stubs*". Embora tenha se popularizado rapidamente nos primeiros anos de sua proposição, o CORBA logo se tornou uma tecnologia ultrapassada e considerada "obscura" (HENNING, 2006), especialmente com o advento dos *web-services*.

O *Simple Object Access Protocol* (SOAP) é um protocolo para troca de informações entre sistemas distribuídos proposto pelo *World Wide Web Consortium* (W3C) (BOX et al., 2000) muito utilizado na implementação de *web-services*, especialmente por ser mais simples que o CORBA e por rodar sobre HTTP. O SOAP se popularizou com a proposição da *Service-Oriented Architecture* (SOA) (PERREY; LYCETT, 2003), que propõe a utilização do *Enterprise Service Bus* (ESB), um barramento para interconexão de sistemas que define uma interface padronizada, o que diminui o acoplamento entre sistemas e reforça princípios de coesão e reaproveitamento. A sintaxe do SOAP é definida através de *eXtensible Markup Language* (XML) (BRAY et al., 2000), que também foi proposta pelo W3C. Por isso, as estruturas de dados utilizadas no SOAP são definidas via *XML Schema Definition* (XSD), e os serviços são especificados via *Web Services Description Language* (WSDL).

O padrão *Representational State Transfer* (REST) proposto na Tese de PhD de *Roy Fielding* (FIELDING, 2000) simplifica a utilização de *web-services* através de convenções de *Uniform Resource Identifier* (URI) e da aplicação direta dos métodos e recursos do *Hypertext Transfer Protocol* (HTTP). Os dados transmitidos via REST tradicional-

mente são descritos com *JavaScript Object Notation* (JSON) (CROCKFORD, 2006), uma notação textual para representação de estrutura de dados através de uma abordagem "chave/valor"; aspectos de segurança são tratados diretamente pelo protocolo HTTP (ou HTTPS); os métodos são os mesmos definidos pelo HTTP (e.g. GET, POST, PUT e DELETE); e, os códigos de retorno também (e.g. *200 OK*, *403 Forbiden*, *404 Not Found* e *500 Internal Server Error*). REST hoje é um padrão para a integração de sistemas e implementação de APIs *web* (YASMIN; TIAN; YANG, 2020), sendo utilizado amplamente por aplicativos *Android* e *iOS* (MESFIN et al., 2016); provedores de serviços como *Facebook*, *Google* e *Amazon* (BATTLE; BENSON, 2008); e, na comunicação com sistemas de controle da infraestrutura como *Open Network Operating System* (ONOS) (BERDE et al., 2014) e *OpenStack* (PEPPLE, 2011).

O *Network Configuration Protocol* (NETCONF) é um protocolo especializado na configuração e gerenciamento de dispositivos de rede proposto pelo IETF na RFC 4741 (ENNS, 2006). A solução criada pelo NETCONF permite a instalação, manipulação e remoção de configurações nos dispositivos através de operações descritas em XML, JSON ou YAML (BEN-KIKI; EVANS; INGERSON, 2009), tais como *get-config*, *edit-config*, *delete-config*, *lock* e *close-session*. Essas operações foram propostas pelo *Network Modeling Working Group* (NETMOD) na RFC 6020 através da YANG (BJORKLUND, 2010), uma linguagem independente de fabricante para modelagem dados e operações de configuração. Esta linguagem, no entanto, requer conhecimento de detalhes sobre os equipamentos, e por isso se difere da proposta nesta tese que tem maior expressividade semântica (IBSRM – seção 4.1.1). Recentemente foi proposto o RESTCONF na RFC 8040 (BIERMAN et al., 2017), uma solução baseada na especificação YANG e no NETCONF com uma abordagem REST.

O protocolo *OpenFlow* proposto *Mckeown* (MCKEOWN et al., 2008) propõe uma abordagem SDN inspirada em técnicas de comutação por circuitos tradicionalmente utilizadas nas redes de telecomunicações (KUROSE; ROSS, 2006). Este protocolo permite a definição de fluxos nos elementos de comutação para tratativa e redirecionamento de dados. Essa flexibilidade facilita a implantação de configurações e fornece uma abordagem de alto nível para o controle da infraestrutura. O *OpenFlow*, no entanto, permite apenas a configuração de fluxos o que torna necessária a utilização de outras soluções para configuração de *interfaces*, *bridges* e *policies* dentre outros. Por este motivo foi proposto o *OpenFlow Management and Configuration Protocol* (OF-CONFIG) (ONF, 2014) pela *Open Networking Foundation* (ONF) que estende o NETCONF para aplicação em *switches OpenFlow*. Esta solução altera a modelagem YANG para incluir elementos específicos da solução OpenFlow, e especifica os *OpenFlow Configuration Points* responsáveis pela configuração dos dispositivos.

O *Open vSwitch Database Management Protocol* (PFAFF; DAVIE, 2013) é um protocolo para o gerenciamento de switches virtuais com suporte à *Open vSwitch Data-*

base (OVSDB). O *Open vSwitch* (PFAFF et al., 2015), que utiliza OVSDB e suporta o protocolo de gerenciamento homônimo, tem em sua implementação tanto o cliente (que permite a alteração da base local pelo próprio elemento), quanto servidor (que permite receber chamadas remotas para alterar a base local). Dessa maneira o gerenciamento de *Open vSwitches* pode ser realizado através de uma abordagem cliente/servidor a partir de um sistema de gerenciamento.

Mesmo com todos os protocolos listados acima há casos nos quais são utilizados sistemas de gerência ou aplicações WEB disponibilizadas pelos fabricantes de equipamentos (e.g. *OmniVista* da *Alcatel* (SOLOMON, 2008)). Normalmente estes sistemas são restritos aos equipamentos do fabricante e fornecem algum tipo interface de integração (e.g. CORBA, SOAP ou REST). Quase sempre é possível também acessar os equipamentos diretamente através de programas *Command Line Interface* (CLI), seja de maneira local: via SERIAL ou USB; ou remota: via *Secure Shell* (SSH) (LEHTINEN; LONVICK, 2006) ou Telnet (O'SULLIVAN, 1971). O problema nestas abordagens está na dependência de conhecimento técnico dos administradores de rede; necessidade de intervenção manual que inviabiliza a aplicação em redes de larga escala e aumenta o risco de falha; e, dificuldade de gerenciamento de redes heterogêneas [2].

2.1.4 Gerenciamento de Redes Futuras

O gerenciamento de redes futuras enfrenta grandes desafios, sobretudo com os requisitos e indicadores de desempenho definidos pelo 5G (GUPTA; JHA, 2015). A expectativa é que a rede comece a ser utilizada em áreas mais complexas com requisitos críticos como *e-health*, *smart cities*, *self-driving cars* e *smart grids*; e que seja ainda mais explorada em áreas como finanças, segurança, *e-commerce*, *streaming* e *cloud computing*.

Internet of Things (ATZORI; IERA; MORABITO, 2010) introduz novos desafios ao gerenciamento, uma vez que amplia a dimensão da infraestrutura e a complexidade no controle e processamento. Nesse tipo de aplicação, assim como na *Edge Computing* (SATYANARAYANAN, 2017), a rede precisa ser capaz de alocar elementos de software para processamento de dados, o que requer uma abordagem de gerenciamento específica.

A concepção de uma rede 5G requer a evolução de três campos: *Radio*, Operação e Gerenciamento (LóPEZ et al., 2017). No trabalho de López et al. (2017) fica claro o papel do gerenciamento como mecanismo para garantia dos requisitos de alto desempenho e disponibilidade; viabilização de uma solução para controle e aprovisionamento de elementos físicos e virtualizados; redução de custos operacionais e de investimento; melhoria da qualidade do serviço e da própria receita; e, redução da complexidade e de riscos à operação.

De maneira geral, pode-se afirmar que as abordagens SDN e NFV facilitam o gerenciamento da rede, uma vez que permitem o controle da infraestrutura com uma visão

[2] Redes com equipamentos e tecnologias de diferentes fornecedores, modelos e versões

logicamente centralizada bem como a orquestração das funções de rede de maneira virtualizada (BOURAS; KOLLIA; PAPAZOIS, 2017). Por esse motivo, acredita-se que as redes futuras serão concebidas através das abordagens SDN e NFV com foco nos requisitos de segurança, escalabilidade e robustez (ZAIDI et al., 2018).

O gerenciamento de redes SDN também apresenta desafios, uma vez que um elemento controlador representa um ponto de falha (AZAB; FORTES, 2017). Além disso, elementos de controle definem requisitos *carrier-grade* como consistência, disponibilidade e tolerância a particionamento, o que como definido pelo Teorema CAP (*Consistency, Availability and Partition tolerance*) de Brewer (BREWER, 2000) é um problema intratável. A manutenção de um plano de Controle estável e a saúde dos componentes de controle estão diretamente relacionados ao estado da rede, por isso, para implementar uma solução de gerenciamento com base em SDN e NFV deve-se primeiro criar mecanismos para garantir que os componentes definidos por essas abordagens estejam sempre operacionais.

A configuração de dispositivos em redes SDN já conta com ferramentas e protocolos úteis ao gerenciamento, como por exemplo o OF-CONFIG, apresentado na seção anterior, que pode ser utilizado na configuração de *switches OpenFlow*; e, o *AdVisor* que é uma ferramenta útil para a inicialização de redes *OpenFlow* (WICKBOLDT et al., 2015).

Existem projetos em desenvolvimento na área de gerenciamento para redes futuras, sendo que dentre eles destacam-se o COGNET e o SELFNET (LóPEZ et al., 2017). Estes projetos são abordados em detalhes na seção 2.4, em especial o SELFNET devido a sua similaridade com a SONAr. Todos estes trabalhos propõem abordagens de gerenciamento autônomo (*self-management*), o que se tornou um consenso nas propostas para redes futuras.

2.2 Computação Autônoma e Redes Auto-Organizadas

O crescimento vertiginoso e a convergência das redes criam cenários que se mostram inviáveis de serem gerenciados humanamente (LóPEZ et al., 2017). Se considerarmos que IoT deve adicionar trilhões de coisas a estes cenários (VERMESAN et al., 2017), então é o momento, ou passou do momento, de se ter uma rede autônoma. Redes autônomas nos moldes preconizados pela Computação Autônoma ainda é um sonho, mas o *3rd Generation Partnership Project* (3GPP) (3GPP, 2018a) cunhou o conceito de 'Rede Auto-Organizada' (*Self-Organizing Networks* (SON)) que é pertinente ao momento da computação que vivemos. A seção 2.2.1 faz uma introdução à Computação Autônoma, a seção 2.2.2 introduz o conceito de Redes Auto-Organizadas, a seção 2.2.3 apresenta as Propriedades de uma Rede Auto-Gerenciada, a seção 2.2.2.1 apresenta o conceito de SON do 3GPP e a seção 2.2.2.2 abrange *Self-Driving Network* (SDN2) e *Intent-Based Networking* (IBN).

2.2.1 Conceituação e Especificação de Computação Autônoma

Computação Autônoma (*Autonomic Computing*) foi proposta pela IBM (HORN, 2001) como uma solução para lidar com a complexidade de sistemas computacionais. Para isso, os sistemas devem ser "inteligentes" e capazes de se auto-gerenciar de acordo com objetivos administrativos preestabelecidos. O termo Computação Autônoma é uma analogia ao 'sistema nervoso' (*Autonomic Nervous System*), que é responsável por importantes funções do corpo (e.g. respiração e irrigação sanguínea) sem uma intervenção consciente. De maneira semelhante, um sistema autônomo deve ser capaz de prover funções básicas para o seu funcionamento sem intervenção humana.

Ganek e Corbi (2003) exploram essa abordagem ao definir as características de um sistema autônomo:

- ❏ deve ser capaz de se configurar (e reconfigurar) diante mudanças de contexto programadas ou aleatórias;
- ❏ deve sempre procurar maneiras de otimizar o seu comportamento;
- ❏ deve ser capaz de se recuperar de eventos rotineiros ou extraordinários que possam causar o seu mau funcionamento;
- ❏ deve resistir a ameaças externas e impedir o uso indevido.

Essas características foram propostas pelos autores como fundamentos de computação autônoma, e são descritas em detalhes neste trabalho, sendo elas: auto-configuração (*self-configuration*), auto-cura (*self-healing*), auto-otimização (*self-optimization*) e auto-proteção (*self-protection*).

O nível de automação de um sistema pode variar desde a administração manual até a autônoma (MCLARNON et al., 2011). Pesquisadores da IBM definiram cinco níveis de automação (IBM, 2002):

- ❏ *manual* – todas as tarefas são realizadas por administradores;
- ❏ *gerenciado* – administradores utilizam ferramentas para a realização de atividades específicas;
- ❏ *preditivo* – o sistema é capaz de analisar dados e recomendar alterações para administradores;
- ❏ *adaptativo* – o sistema é capaz de se gerenciar, mas ainda precisa dos administradores no tratamento de situações mais complexas; e
- ❏ *autônomo* – o sistema é completamente capaz de se gerenciar com base em políticas definidas por administradores.

Os fundamentos de computação autônoma, também chamados de propriedades *self-**, representam características básicas que devem ser contempladas por sistemas autônomos. Diversas propriedades foram propostas com a evolução destes sistemas, sendo as quatro principais definidas pela IBM (GANEK; CORBI, 2003):

- *self-configuration* – sistemas autônomos que contemplam propriedades de auto-configuração:
  - adaptam-se automaticamente a ambientes dinâmicos,
  - têm a capacidade de definir o seu comportamento em tempo de execução, e
  - permitem a extensão de funcionalidades e sua distribuição sem intervenção humana;
- *self-healing* – sistemas autônomos que contemplam propriedades de auto-cura:
  - monitoram, diagnosticam e corrigem falhas,
  - predizem problemas e tomam decisões preventivas para evitar falhas, e
  - são resilientes a falhas e se mantêm sempre disponíveis;
- *self-optimization* – sistemas autônomos que contemplam propriedades de auto-otimização:
  - monitoram e ajustam recursos de maneira automática,
  - permitem a alocação, balanceamento e distribuição de recursos sem intervenção humana, e
  - analisam a utilização e tomam medidas para evitar o desperdício de recursos;
- *self-protection* – sistemas autônomos que contemplam propriedades de auto-proteção:
  - antecipam, detectam e se protegem de ataques,
  - aplicam técnicas para autorização de acesso a recursos e identificação de intrusos, e
  - identificam falhas de segurança e aplicam correções/atualizações de forma automática.

Além dos quatro fundamentos básicos citados, outros foram propostos na literatura (RAMIRO; HAMIED, 2011; VERMESAN et al., 2017; SADIGHI et al., 2018; STEIN et al., 2018). Dentre eles destacam-se:

- *self-orchestration* – sistemas autônomos que contemplam propriedades de auto-orquestração:
  - definem os seus fluxos de execução de maneira dinâmica,
  - interpretam cenários e decidem sobre as melhores abordagens, e
  - priorizam componentes e funcionalidades de acordo com o contexto;
- *self-awareness* – sistemas autônomos que contemplam propriedades de auto-consciência:
  - têm uma noção fiel a respeito do seu atual estado,

- permitem a extração de dados e realização de testes em seus componentes, e
- monitoram e registram informações referentes às suas utilizações;

❑ *self-learning* – sistemas autônomos que contemplam propriedades de auto-aprendizado:

- aprendem com diagnósticos e tratativas anteriores,
- analisam cenários, correlacionam dados e preveem mudanças, e
- interpretam requisitos de alto nível e traduzem em ações práticas.

2.2.2 Conceituação e Levantamento de Redes Auto-Organizadas

As redes de computadores atuais têm níveis de automação que variam entre o manual e o gerenciado. Isso ocorre devido à complexidade em se automatizar as tarefas administrativas e pelas barreiras culturais na administração de redes (BARRETT et al., 2004). Administradores tendem a ver de forma negativa a perda de controle e a dependência de ferramentas no processo administrativo (DUEZ; ZULIANI; JAMIESON, 2006). Bainbridge (1983) demonstra que nem sempre a automação é a solução mais apropriada, pois às vezes ela mesma pode aumentar a complexidade de sistemas o que ele chama de "*ironia da automação*". Esses fatores somados à *ossificação da rede* (MOREIRA et al., 2009) foram determinantes para a não automação das redes de computadores. Deste modo, a ideia de redes autônomas cedeu lugar ao conceito cunhado pelo 3GPP denominado 'Rede Auto-Organizada'.

2.2.2.1 *Self-Organizing Networks* (SON)

O conceito de *Self-Organizing Networks* (SON) surgiu no *Release 8* do *3GPP* (3GPP, 2018a) na proposta de *Long-Term Evolution (LTE)* (ASTELY et al., 2009) para 'gerenciamento de rádio na telefonia móvel', cujo objetivo é criar mecanismos para lidar com a complexidade de redes de telecomunicações através da automação de tarefas de planejamento, configuração, gerenciamento, otimização e cura. Essas redes contam com um plano de Sinalização separado do plano de Dados (de maneira similar à separação dos planos de Controle e de Dados em SDN), e com elementos de controle na borda da rede (de maneira similar às ferramentas administrativas das redes atuais). Através da padronização de protocolos e automação de processos, as redes móveis se tornaram extremamente flexíveis e permitem a implantação de funcionalidades complexas. A ideia por trás do SON é a automação do gerenciamento de redes usando funções de auto-organização. Os principais casos de SON estão relacionados ao gerenciamento de *eNodeB*, de maneira a permitir que novas *eNodeB* sejam adicionadas automaticamente à topologia e com processos de otimização/recuperação sem intervenção manual.

A função de *Auto-Estabelecimento* (ou Auto-configuração) tem a função de alocar automaticamente novos recursos de rede (uma eNodeB por exemplo). Este conceito é

especificado na referência 3GPP (2018b), na qual são descritos aspectos de segurança e procedimentos de interconexão com redes internas e externas. A Auto-configuração utilizada para alocação de uma nova eNodeB, é realizada quando a eNodeB é inserida na topologia considerando que a rede já esteja operacional no momento da inserção.

A *Auto-Cura*, especificada pela referência 3GPP (2018e), destina-se a resolver ou mitigar falhas automaticamente. A função de Auto-cura se divide basicamente em duas partes: monitoramento e recuperação. O monitoramento é responsável por detectar falhas de software/hardware e a recuperação por corrigir essas falhas automaticamente. Ações comuns de recuperação são: reinicialização do sistema, restauração de *backups*, realização de *fallbacks*, reconfiguração etc.

A função de *Auto-Otimização* visa melhorar a utilização de recursos da rede e a qualidade dos serviços de comunicação conforme especificado na referência 3GPP (2018d). A Auto-otimização monitora dados de gerenciamento, como métricas de desempenho, notificações, alarmes de falhas, e define quais ações devem ser realizadas para melhoria da comunicação. Ações comuns de melhorias nas redes de celulares são balanceamento de carga, *handover*, reparametrização etc.

Os conceitos de Auto-Organização definidos pela 3GPP foram desenvolvidos e implantados por fornecedores em todo o mundo. Note-se que estes conceitos foram aplicados na borda da rede, comumente referido como rede celular. O núcleo da rede não possui padrões ou arquiteturas preparadas para lidar com esses conceitos. Por este motivo, iniciativas como o SONAr (proposta nesta tese) são necessárias.

2.2.2.2 *Self-Driving Network* (SDN2) e *Intent-Based Networking* (IBN)

Self-Driving Network (SDN2) (KOMPELLA, 2019) é o nome da iniciativa da fabricante de equipamentos *Juniper* que visa conceber redes com gerenciamento "*zero touch*" através da automação de processos de operação e administração para reduzir a complexidade da rede, facilitar a implantação de serviços, otimizar a utilização de recursos e aumentar a resiliência. Esta iniciativa visa tornar as redes capazes de preverem e de se adaptarem automaticamente a mudanças ambientais, reduzir custos operacionais, minimizar riscos e melhorar a qualidade da experiência de usuários finais. O termo é uma analogia ao *self-driving car*, que é uma tecnologia para carros que funciona sem intervenção humana, apenas com inteligência artificial. Da mesma forma, a SDN2 propõe redes autônomas através de técnicas como aprendizado de máquina, automação de processos, intenções declarativa, telemetria e *big data*.

Intent-Based Networking (IBN) (LALIBERTE, 2018) é o nome dado para a iniciativa de auto-gerenciamento da fabricante de equipamentos *Cisco* que propõe o uso de intenções declarativas para guiar o comportamento da rede. Essas "intenções" representam requisitos comunicacionais com alto nível de abstração e precisam sere traduzidas, ativadas e garantidas pela rede. Isso requer estratégias de monitoramento contínuo e correções

guiadas com aprendizado de máquina. A ideia é descrever `"qual"` o comportamento esperado da rede e não *"como"* a rede deve agir para garantir este comportamento. Isso é consistente com a definição da IBM de alto nível de automação (MCLARNON et al., 2011), que propõe que a interação entre administradores e a rede deve ser definida por meio de políticas em alto nível.

A ideia de gerenciamento com base na interpretação de intenções, que é abordada em ambas as propostas, se baseia no princípio de computação com o modelo declarativo (FAHLAND et al., 2009). Neste modelo, a lógica de computação é expressa sem detalhes sobre o fluxo de controle, e sim com base na definição de comportamentos. Por outro lado o modelo imperativo, que é aplicado nas estratégias de gerenciamento convencionais, descreve computação de maneira mais básica em termos de instruções que alteram o estado do sistema e o fluxo de execução. Com base no modelo declarativo surgiu o conceito de "Intenção Declarativa" (DI) que consiste em uma definição de requisitos e comportamentos por meio de intenções descritas em linguagem natural ou formal.

As DIs no IBN são definidas como restrições de nível de serviço, políticas ou requisitos de TI. Essa abordagem define um ciclo para interpretação de intenções como medidas práticas de configuração e verificação contínua da infraestrutura para validar o cumprimento das intenções. Caso uma intenção não esteja sendo atendida, o ciclo é reiniciado com a reinterpretação da intenção. No SDN2, as DIs são definidas por políticas e restrições, como largura de banda e latência mínima. Essas intenções também são tratadas por um *loop* de controle fechado (como no IBN), que é baseado na análise com telemetria em tempo real e na tomada de decisão baseado na previsão, recomendação, tendência e correlação de dados.

2.2.3 Propriedades *Self-** no Contexto do Gerenciamento de Redes

Os fundamentos de computação autônoma (seção 2.2.1) aplicados no gerenciamento de sistemas autônomos são comumente referidos como 'Propriedades *Self-**' (BERNS; GHOSH, 2009). Há inúmeras propriedades *Self-** listadas na literatura conforme levantado por Berns e Ghosh (2009), por isso este trabalho se restringe às propriedades utilizadas na concepção da proposta SONAr apresentada na seção 5.1, são elas: Auto-Configuração (seção 2.2.3.1), Auto-Cura (seção 2.2.3.2), Auto-Otimização (seção 2.2.3.3), Auto-Proteção (seção 2.2.3.4), Auto-Orquestração (seção 2.2.3.5), Auto-Aprendizado (seção 2.2.3.6) e Auto-Consciência (seção 2.2.3.7).

Nesta seção serão apresentadas as Propriedades *Self-** analisadas sob o ponto de vista da automação do gerenciamento das redes de computadores e de telecomunicações.

2.2.3.1 Auto-Configuração

A propriedade de 'Auto-Configuração' (*Self-Configuration*) (3GPP, 2018b; WOJCIECHOWSKI; SIERSZEń; STURGULEWSKI, 2016) se refere à capacidade da rede de se auto-configurar e de se auto-adaptar a mudanças no ambiente. A rede deve ser capaz de se auto-inicializar sem intervenção humana, suportar novos dispositivos com uma abordagem *Plug-and-Play* e gerenciar serviços conforme políticas definidas por administradores. A configuração deve ser ótima e adaptável a mudanças de contexto.

Um importante aspecto da automação da configuração de uma rede está no processo de inicialização da rede, também chamado de *Bootstrapping* (CANINI et al., 2017). Normalmente, este é um processo manual que requer um alto nível de conhecimento técnico. Nas redes atuais, por exemplo, a implantação de um novo *switch* em um anel *Metro-Ethernet* requer a replicação de todas as VLANs nas novas portas "tronco" e a configuração, atualização e padronização do novo dispositivo. De maneira similar, nas redes SDN, é necessário criar regras de fluxo de controle em cada *switch*, antes mesmo de inicializar o controlador pra prover o controle *in-band*.

Uma rede com capacidade de auto-configuração deve ser capaz de configurar os serviços de comunicação de acordo com as suas características de maneira automática. Nos serviços convencionais de acesso à Internet por exemplo, esse processo normalmente envolve a configuração de *VLANs*, criação de *policies* de QoS e regras de roteamento nos equipamentos de *core*, distribuição e acesso. Nas abordagens SDN e NFV, a configuração de um serviço se baseia na criação de regras de fluxos nos *switches* e instanciação de funções de rede virtualizadas conforme a necessidade.

Uma rede auto-configurável deve suportar a implantação de novos equipamentos e serviços de forma dinâmica com uma abordagem similar ao *Plug-and-Play* dos sistemas operacionais. Quando um novo equipamento é adicionado à rede, ele precisa passar pelo processo de inicialização e de configuração. De maneira similar, caso um equipamento seja removido, ou tenha as suas características alteradas, a rede precisa se adaptar através da reconfiguração dos serviços.

2.2.3.2 Auto-Cura

A propriedade de 'Auto-Cura' (*Self-Healing*) (VILCHEZ; YAHIA; CRESPI, 2014; THORAT et al., 2015; 3GPP, 2018e) se refere à capacidade da rede de corrigir (ou evitar) falhas de maneira a garantir a sua disponibilidade. Estas falhas podem estar relacionadas a eventos aleatórios (e.g. rompimento de um cabo) ou previsíveis (e.g. ocorrência de um *loop* na rede). Eventos de falha podem ser tratados por meio de medidas pró-ativas (e.g. redundância de *links*, balanceamento de carga e *backup* de configurações), e/ou reativas (e.g. redirecionamento de tráfego e reinicialização de componentes). A tratativa pró-ativa de eventos previsíveis envolve também a previsão e diagnóstico de cenários de

falha antes que eles ocorram. No exemplo citado anteriormente (ocorrência de *loop* na rede), é possível identificar o erro antes que os serviços sejam afetados através da: análise da configuração (e.g. teste de rotas/fluxos e verificação de protocolos de controle de *loop*), avaliação do padrão de crescimento na utilização de um *link* (e.g. identificação de comportamento não linear) e correlação de alarmes (e.g. notificações de uso excessivo de memória e descarte de pacotes).

Redes com capacidade de auto-cura precisam garantir que a comunicação entre os próprios componentes de controle esteja sempre disponível. Mesmo nas redes convencionais é necessário manter a conectividade com todos os equipamentos para que eles possam ser gerenciados. Assim, a rede deve estar preparada para cenários mais complexos como a indisponibilidade de equipamentos e rompimento de conexões, e deve ser capaz de reconfigurar as regras de fluxo de controle (no caso de SDN) e de roteamento (no caso das redes convencionais).

2.2.3.3 Auto-Otimização

A propriedade de 'Auto-Otimização' (*Self-Optimization*) (RAMIRO; HAMIED, 2011; RAZAVI; KLEIN; CLAUSSEN, 2010) diz respeito à capacidade de auto-ajustar a configuração da rede com o objetivo de obter melhorias de desempenho ou redução de custo. De uma maneira geral, essa funcionalidade está diretamente relacionada à qualidade de serviço uma vez que impacta no custo e na garantia de requisitos comunicacionais. A capacidade de auto-otimização pressupõe uma rede em estado operacional, e por isso os fluxos de tratamento devem ser empregados apenas quando a rede está livre de ataques e erros.

Para otimizar a utilização dos recursos de infraestrutura a rede deve ter um alto nível de consciência (seção 2.2.3.7), de maneira a identificar situações como congestionamento de *links*, subutilização de dispositivos e degradação de serviços. Para a redução de OPEX, a rede deve ser capaz de estimar custos relacionados à utilização de equipamentos, como: consumo energético, valor do ativo e acordos comerciais. Para a melhoria de *Qualidade de Serviço* (QoS) a rede deve monitorar a utilização dos recursos e classificá-los quanto a capacidade de atender a requisitos como: baixa latência e alta vazão. A 'Auto-Otimização' deve ser sempre uma análise da relação custo/benefício levando em consideração políticas definidas por administradores.

2.2.3.4 Auto-Proteção

A propriedade de 'Auto-Proteção' (*Self-Protection*) (HARIRI et al., 2005; BEHRINGER et al., 2015) diz respeito à capacidade da rede de evitar ataques, invasões e usos indevidos de sua infraestrutura ou dos ambientes interconectados. Ataques e invasões representam uma grande ameaça para as redes atuais e futuras (BUCHANAN, 2016), pois podem afetar não apenas usuários mas também os próprios componentes e recursos de

rede. Por este motivo, uma rede com capacidade de auto-proteção precisa ser capaz de detectar e de se proteger de ameaças antes mesmo que elas ocorram, e caso isso não seja possível, ela deve ser capaz de mitigar os impactos negativos. Exemplos de problemas que acometem as redes são *Denial of Service* (DoS), *phishing* e invasões de *hosts*.

Outro importante aspecto referente à propriedade de 'Auto-Proteção' está na capacidade de evitar o seu uso indevido, e para isso a rede deve identificar os usuários e monitorar o *Service Level Agreement* (SLA). Por exemplo, um *Internet Service Provider* (ISP) pode restringir o uso da rede apenas para usuários que contrataram um plano de *Internet* e que estejam com pagamento em dia. Por isso, caso seja detectado um usuário desconhecido ou inadimplente utilizando a rede (por algum erro de configuração), a rede deve ser capaz de bloquear o seu acesso.

Por fim, uma rede com capacidade de auto-proteção deve conhecer os seus próprios recursos para evitar que dispositivos de terceiros mal intencionados sejam acoplados automaticamente à topologia pelos processos de *bootstrapping* e *plug-and-play*. Para isso pode ser necessário utilizar mecanismos de autenticação e autorização para atestar a autenticidade e permitir que os novos recursos sejam conectados aos demais.

2.2.3.5 Auto-Orquestração

A propriedade de 'Auto-Orquestração' (*Self-Orchestration*) (SCHULZ, 2016; VERMESAN et al., 2017) diz respeito à capacidade da rede de gerir os seus componentes e priorizar os seus fluxos de execução. Essa propriedade é necessária para organizar e sincronizar os processos de gerenciamento da rede, resolver conflitos e delegar responsabilidades de acordo com uma análise sobre o estado da rede. Uma rede "auto-orquestrada" é capaz de coordenar o acesso à sua infraestrutura, e dessa maneira, torna-se mais segura e estável. Essa capacidade é especialmente necessária em arquiteturas de rede com componentes distribuídos, visto que eles podem "concorrer" pelos mesmos estímulos (e.g. níveis de utilização e alarmes).

Uma rede com capacidade de 'Auto-Orquestração' precisa ter consciência do seu estado (seção 2.2.3.7), e precisa ser capaz de analisar situações e tomar decisões sem intervenção humana o que pode provido por meio de mecanismos de auto-aprendizado (seção 2.2.3.6). Isso é necessário pois a rede auto-orquestrada precisa decidir quais as melhores linhas de execução para um determinado cenário, e isso nem sempre é claro. Por exemplo, uma rede com um *link* sobrecarregado pode precisar acionar um fluxo de cura (seção 2.2.3.2) caso seja detectada uma falha ou erro de configuração, ou um fluxo de proteção (seção 2.2.3.4) caso seja detectado um ataque em curso, ou mesmo um fluxo de otimização (seção 2.2.3.3) caso seja constatada a utilização normal. Note-se que em todos os casos o "sintoma" é o mesmo: um link congestionado. Dessa forma, a rede precisa aplicar uma análise de primeiro nível para tomar a decisão mais assertiva antes mesmo que os fluxos de tratamento específico de cada propriedade sejam acionados.

2.2.3.6 Auto-Aprendizado

A propriedade de 'Auto-Aprendizado' (*Self-Learning*) (STERRITT, 2002; ZANDER; NGUYEN; ARMITAGE, 2005; LAWYER et al., 2005) diz respeito à capacidade da rede de aprender a analisar dados e tomar decisões sem intervenção humana. Essa característica é útil para interpretar requisitos comunicacionais, diagnosticar problemas/oportunidades, priorizar fluxos de trabalho, predizer cenários e escolher tratativas, atividades essas que são tradicionalmente desempenhadas pelos administradores de rede.

Redes com capacidade de 'Auto-Aprendizado' implementam técnicas de Inteligência Artificial (IA) (WANG; LU, 2019) para prover aprendizagem guiada por administradores, ou inferidas a partir da observação dos dados. O tipo de técnica varia de acordo com a aplicação do aprendizado. Exemplo: a base de conhecimento para diagnóstico de problemas pode ser mais facilmente criada a partir da análise de decisões tomadas por administradores interpretadas como sintomas; já a predição de cenários deve se basear em uma análise estatística com base em cenários anteriores em relação ao tempo.

Essa propriedade é essencial para a concepção de uma abordagem de gerenciamento com alto nível de autonomia, pois ela é a principal responsável pela substituição da intervenção humana. Entretanto, a implementação dessa propriedade não é trivial pois requer a definição de modelos com boa capacidade de representação para os cenários de rede e a aplicação de técnicas de processamento que normalmente são complexas e lentas.

De certo modo, a propriedade de 'Auto-Aprendizado' representa um dos mecanismos possíveis para concepção de uma abordagem de rede com capacidade de 'Auto-Consciência' (seção 2.2.3.7) de maneira adaptativa.

2.2.3.7 Auto-Consciência

A propriedade de 'Auto-Consciência' (*Self-Awareness*) (GELENBE, 2009; SADIGHI et al., 2018) se refere à capacidade da rede de monitorar e compreender o seu estado. Essa "consciência" é provida através da aplicação de funções como coleta de dados, realização de testes, inspeção de cabeçalhos, análise de comportamento e captura/correlação de alarmes. A capacidade da rede de se auto-avaliar é imprescindível para a habilitação de funções de cura e otimização de maneira autônoma (SMIRNOV et al., 2009), pois a tomada de decisão, assim como o diagnóstico de problemas e de oportunidades, só e possível de ser realizada com base na análise de dados. É importante notar que desvios na avaliação do estado da rede podem causar decisões erradas, e assim, comprometer (ou agravar) este estado.

Nas redes atuais a interpretação do estado operacional, análise de causa-raiz e planejamento de mudanças são realizados normalmente de maneira manual (HEGERING, 1999) ou assistida com o auxílio de ferramentas de relatórios e de correlação de eventos (FRIEDBERG et al., 2015). Para isso são utilizadas soluções como as propostas pelo *NetFlow*

(CLAISE, 2004), *sFlow* (PHAAL; PANCHEN; MCKEE, 2001), *Internet Protocol Flow Information Export* (IPFIX) (QUITTEK et al., 2004), *Switched Port Analyzer* (SPAN) e *Remote Switched Port Analyzer* (RSPAN) (SVOBODA et al., 2015), que fornecem mecanismos para coleta de dados por agentes externos.

Conceber uma rede com capacidade de 'Auto-Consciência' requer a utilização de técnicas que simulem a interpretação humana, e é nesse contexto que técnicas de IA se mostram promissoras (MAHMOUD, 2007). Essa propriedade aplicada no contexto das redes se ampara em duas atividades básicas: coleta e análise de dados. Enquanto a coleta já conta com diversas técnicas e ferramentas (como as mencionadas anteriormente), o grande desafio está na análise dos dados.

2.2.4 Requisitos de Auto-Gerenciamento para Redes Futuras

Aplicações futuras podem ser desafiadoras, pois podem exigir requisitos complexos e eventualmente conflitantes. Por exemplo, uma aplicação pode exigir alta disponibilidade, processamento distribuído e consistência de dados, que de acordo com o Teorema CAP (BREWER, 2000) é um problema intratável. Do ponto de vista das redes, requisitos como segurança, confiabilidade, latência, vazão e largura de banda podem habilitar (ou inviabilizar) novas aplicações. Os níveis e tipos de requisitos suportados pelas redes atuais são de certa maneira as grandes barreiras para a evolução das aplicações. Embora já existam alguns exemplos de aplicações modernas como computação nas nuvens e IoT, de maneira geral essas aplicações ainda se encontram em estágio de protótipo aguardando melhorias na capacidade e no gerenciamento da rede.

Com o crescimento da rede e maior complexidade do gerenciamento, a própria Indústria de Telecomunicações passou a demandar requisitos mais complexos como: menor tempo de aprovisionamento, maior disponibilidade e melhores formas de tarifação. A forma como o gerenciamento se dá impacta diretamente no custo operacional (OPEX) do provedor e pode até mesmo inviabilizar o negócio. Essas empresas precisam criar mecanismos para prover serviços com os níveis requeridos pelas aplicações e contratados pelos clientes, o que por si já representa novos requisitos de gerenciamento.

Nas seções seguintes serão apresentados detalhes sobre os requisitos que a rede deve suportar de maneira a viabilizar as aplicações futuras (seção 2.2.4.1) e atender às necessidades da indústria de telecomunicações (seção 2.2.4.2).

2.2.4.1 Aplicações da Internet do Futuro e Requisitos de QoS

A Internet do Futuro está sendo desenvolvida como uma resposta aos requisitos e à complexidade das novas aplicações – algumas das quais já estão sendo exploradas em um escopo limitado. Alguns bons exemplos são aplicações de VR/AR, controle de *drones* (junto com transmissão de vídeo 4K), Internet tátil, jogos na nuvem e carros autônomos.

	Requisitos	
	IMT-Advanced (4G)	IMT-2020 (5G)
Taxa de Transmissão	1 Gbps (*download*) 500 Mbps (*upload*)	20 Gbps (*download*) 10 Gbps (*upload*)
Latência no Plano de Dados	10ms	eMBB: 4ms URLLC: 1ms
Latência no Plano de Controle	100ms	20ms

Tabela 3 – Comparação entre os requisitos de latência e transmissão do 4G e 5G.

Fonte: Adaptado de Simsek et al. (2016).

Todos os exemplos citados exigem um determinado nível de largura de banda, *throughput* e latência, o que pode ser crucial para aplicações multimídia com interação em tempo real ou menos importante para aplicações convencionais de compartilhamento de arquivos por exemplo. Nestes casos mais particulares, os sentidos humanos devem ser levados em consideração. Por exemplo, as interações auditivas precisam de um tempo de resposta de 100 ms, enquanto a reação visual requer cerca de 10ms, e o *feedback* tátil demanda um atraso menor que 1 ms (ida e volta) (SIMSEK et al., 2016).

Para uma aplicação VR/AR de imersão total e realidade indistinguível seria necessário um *throughput* de 5,2 Gb/s para cada usuário. Em uma abordagem mais realista seria necessário suportar um *throughput* de 100-200 Mb/s por usuário com uma latência máxima de 13 ms para prover uma experiência de imersão satisfatória para a maioria das pessoas (BASTUG et al., 2017).

Outra questão importante é a confiabilidade da rede. Algumas aplicações típicas de 5G exigem uma taxa de falha inferior à ordem de grandeza de 10^{-7}, o que corresponde a 3,17 segundos de indisponibilidade por ano (SIMSEK et al., 2016).

Pode-se observar que as redes atuais – incluindo as versões 4G mais recentes – não atendem os requisitos definidos pelas aplicações futuras. De fato, os requisitos definidos pelo 5G são muito mais restritos do que os suportados pelo 4G, especialmente em termos de latência de ponta a ponta, como mostrado na Figura 3.

Para suportar essas novas demandas, técnicas como *Mobile Edge Computing* (MEC) e *Network Slicing* (NS) possivelmente serão utilizadas nas redes futuras. No entanto, acredita-se que para casos específicos, algumas dessas técnicas poderão ser inviáveis. Um exemplo é o uso de um robô na execução de uma cirurgia remota, onde o MEC não faz sentido uma vez que o provedor do serviço é humano e não pode ser simplesmente deslocado para a borda da rede. Em redes futuras, esse serviço deveria ser trabalhado por uma aplicação com foco na redução de latência e do *jitter* e na melhoria da confiabilidade.

Em geral, os principais requisitos para uma rede de nova geração em relação às aplicações futuras são: flexibilidade, plasticidade, confiabilidade, boa relação custo-benefício

e facilidade de (re)configuração, implementação e controle.

2.2.4.2 Principais Requisitos da Indústria de Telecomunicações

Implementar uma rede auto-gerenciada não é uma tarefa fácil. Do ponto de vista da operadora de rede existem vários desafios tais como: prover compatibilidade com tecnologias legadas, definição de estratégias de implantação, aplicação de políticas de segurança e viabilização de casos de negócio com as aplicações futuras como os listados em Zaidi et al. (2018). Os principais requisitos apresentados pelos operadores de rede podem ser divididos entre as propriedades de sistemas autônomos descritas na seção 2.2.1: auto-configuração, auto-cura, auto-otimização e auto-proteção. Sem esses requisitos, será difícil – se não impossível – implantar com êxito as aplicações da Internet do Futuro em uma rede comercial.

- *Auto-Configuração* – configurar elementos de rede com as tecnologias atuais exige um grande esforço técnico e um profundo conhecimento da infraestrutura. Essa atividade faz parte do dia a dia de qualquer provedor de serviços de comunicação e é executada sempre que um novo NE é aprovisionado, substituído ou restaurado. Com a perspectiva de crescimento da infraestrutura das redes futuras e com a maior oferta de tecnologias de acesso, a configuração de elementos, assim como a inicialização da rede, deve se tornar ainda mais complexa. De maneira semelhante, a configuração de serviços oferece grandes desafios técnicos aos operadores, tanto pela necessidade de implantação de mecanismos de garantia de requisitos quanto pela atividade operacional de configuração da infraestrutura. Sendo assim, assume-se que a inicialização, o aprovisionamento de serviços e o *Plug-and-Play* são requisitos de configuração dos operadores das redes futuras;

- *Auto-Cura* – manter os serviços de comunicação com classe 6 de disponibilidade (99,9999%) é a maior prioridade dos operadores. Os serviços de comunicação devem ser tratados como recursos básicos, tais como a água e energia, devido ao impacto econômico e risco de vida associados à sua ausência. Não por acaso, a maior parte do custo com a operação de redes convencionais é destinada ao monitoramento, diagnóstico, tratativa e prevenção de problemas de rede. Nas redes atuais, esta é uma tarefa extremamente complexa que exige a alocação de equipes técnicas em escalas de plantão 24/7. Assim como na 'Auto-Configuração', o crescimento e a maior complexidade da infraestrutura de rede contribuem de forma negativa para a 'Auto-Cura'. Sendo assim, acredita-se que a capacidade de identificar problemas e restabelecer serviços de forma autônoma são requisitos de interesse dos operadores das redes futuras;

- *Auto-Proteção* – implantar políticas de segurança, identificar intrusos, autenticar usuários/aplicações e aplicar atualizações/*patches* são só algumas das medidas mais

básicas necessárias para proteger redes. Acredita-se que os operadores futuros precisarão implementar mecanismos de inteligência artificial para análise de tráfego e padrões de comportamento para identificar ameaças e impedir que elas ocorram. Nas redes atuais, a tratativa de ataques normalmente é realizada pelas mesmas equipes técnicas responsáveis pela 'Auto-Cura' depois que a rede já foi comprometida. Dessa forma, requisitos como coleta/monitoramento de dados, realização de diagnóstico e emprego de tratativas automáticas são requisitos essenciais para os provedores das redes futuras;

- *Auto-Otimização* – otimizar a utilização da rede talvez seja a atividade de maior interesse dos provedores de serviço, uma vez que ela melhora os lucros e viabiliza os negócios. Otimizar significa que serão necessários menos investimentos com a expansão da infraestrutura e que mais clientes poderão ser atendidos com os mesmos recursos. A otimização também impacta na qualidade do serviço, o que pode tanto auxiliar no cumprimento de SLAs quanto ajudar na retenção e conquista de clientes. Acredita-se que provedores das redes futuras tenham como requisitos a criação de mecanismos capazes de analisar a configuração e o comportamento da rede e propor ajustes ou reconfigurações de maneira a economizar recursos.

2.3 Tecnologias Habilitadoras para Auto-Gerenciamento de Redes

O SDN (COX et al., 2017) surgiu como uma maneira de flexibilizar o controle de redes de computadores, o que é útil na implementação de funções de operação e manutenção. SDN facilita a Auto-organização da rede através da abstração de um plano de Controle logicamente centralizado e separado do plano de Dados; utilização de um modelo aberto que incentiva a customização e a evolução colaborativa; e, abstração simples que permite a implementação de novas funções e protocolos de maneira rápida e flexível (COX et al., 2017).

De maneira complementar, NFV (ETSI, 2013) propõe a abstração de funções de rede como elementos de *software* virtualizados denominados *Virtualized Network Functions* (VNFs). Tradicionalmente essas funções são implementadas como *appliances* o que dificulta o aprovisionamento por limitações físicas. Além disso, esses *appliances* normalmente se baseiam em padrões fechados, o que dificulta a customização e evolução das funções de rede. Por outro lado, o aprovisionamento de VNFs consiste em instanciar contêineres ou máquinas virtuais em posições específicas da rede, o que normalmente já é uma função automatizada. As funções de tratamento do plano de Dados podem ser alocadas sob demanda de maneira a garantir os requisitos das aplicações. Componentes de controle também podem ser instanciados conforme a necessidade através do NFV, o que

pode ser útil tanto para a inicialização da rede, distribuição/balanceamento do controle e aplicação de intervenções para cura/otimização/proteção.

Ambos SDN e NFV encabeçam um movimento que ficou conhecido como *network softwarization* que se assemelham de certo modo ao movimento de migração de implementações de *hardware* para *software* pelo qual os computadores passaram ainda nos anos sessenta. De maneira geral, acredita-se que esse movimento engloba as principais tecnologias habilitadoras para o auto-gerenciamento de redes de computadores que deverão se basear em elementos de *software* para controlar, gerenciar e processar dados da rede. Não por acaso, a maioria das propostas para as redes futuras propõe a utilização de pelo menos uma dessas abordagens. Zhang, Xie e Yang (2015), por exemplo, propõe uma arquitetura para redes 5G baseada em SDN e NFV e justifica a escolha pela separação dos planos de Dados e de Controle pela abstração de funções de rede virtualizadas. As seções seguintes apresentarão maiores detalhes sobre essas tecnologias.

2.3.1 Redes Definidas por Software

Software-defined Networking (SDN) (LANTZ; HELLER; MCKEOWN, 2010) é uma abordagem para redes de computadores, que cria uma interface lógica aberta e bem definida para o controle da rede, como é apresentado na Figura 2. Essa camada abstrai a rede e fornece serviços que permitem a sua reconfiguração em tempo de execução. Por esse motivo, SDN é considerado uma ferramenta de inovação para pesquisas sobre as redes futuras, o que também se deve à distinção e organização entre funções de controle e de encaminhamento/processamento.

SDN torna possível a separação entre os planos de Dados e Controle, que podem ser programados de maneira independente da infraestrutura subjacente. Isso simplifica drasticamente o processo de administração de redes, por definir uma camada que abstrai os diferentes tipos de equipamentos, e flexibiliza a implantação de comportamentos de maneira pró-ativa ou reativa, através de um software logicamente centralizado responsável pela tramitação da comunicação.

2.3.1.1 *OpenFlow*

O *OpenFlow* (MCKEOWN et al., 2008) é a materialização da abordagem SDN, e implementa a separação dos planos de Controle e Dados através do conceito de fluxo e do protocolo aberto homônimo. A orquestração no nível de Controle é realizada pelo Controlador *OpenFlow*, um software extensível que dá suporte ao protocolo *OpenFlow*, e que permite a criação de comportamentos de maneira flexível como proposto pelo SDN. Dessa maneira, torna-se possível a implementação de algoritmos sensíveis ao contexto de utilização, estado da rede e às requisições de sistemas externos de maneira pouco intrusiva.

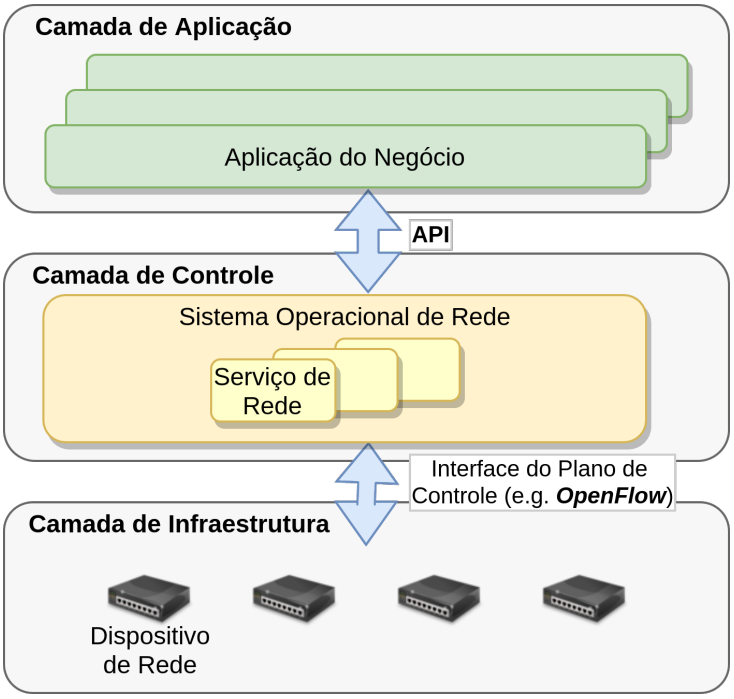


Figura 2 – Abstração da rede em camadas na visão da abordagem SDN.

Fonte: Adaptado de McKeown et al. (2008).

O Controlador *OpenFlow* realiza configurações de regras e 'rotas' nos elementos de rede de forma a guiar o tratamento das primitivas no plano de Dados. Essas regras definem padrões a serem utilizados como filtros de primitivas e ações a serem tomadas, como por exemplo: encaminhar a primitiva para uma determinada porta ou alterar campos do cabeçalho. Elas são armazenadas em tabelas de fluxos de maneira similar às tabelas de comutação nos *switches* convencionais.

O *OpenFlow* versão 1.0 define a utilização de apenas uma tabela de fluxo no *switch* conforme apresentado na Figura 3, enquanto nas versões posteriores foram adicionadas múltiplas tabelas que são analisadas sequencialmente. O objetivo dessa estratégia é diminuir a latência da rede através de uma otimização no processo de identificação de fluxos. As versões 1.1 e 1.2 apresentaram também mudanças nas ações possíveis para o tratamento dos fluxos. A versão 1.3 apresenta importantes evoluções, sobretudo na capacidade de criação de regras para identificação de fluxos de maneira flexível, e não mais é limitada aos protocolos da Arquitetura TCP/IP. Essa nova estratégia é chamada de *OpenFlow eXtended Match* (OXM) (FERNANDES; ROTHENBERG, 2014), e fornece um importante apoio para abordagens disruptivas, por facilitar o suporte a novos protocolos por *switches* e *controladores OpenFlow*.

Atualmente verifica-se um grande crescimento no número de elementos de redes com suporte ao *OpenFlow*. Grandes fabricantes de equipamentos, como Cisco, NEC, Juniper e HP têm investido em pesquisas sobre SDN e oferecem equipamentos com suporte ao protocolo *OpenFlow* (Cisco, 2013; HP, 2013; Huawei, 2013; NoviFlow, 2013).

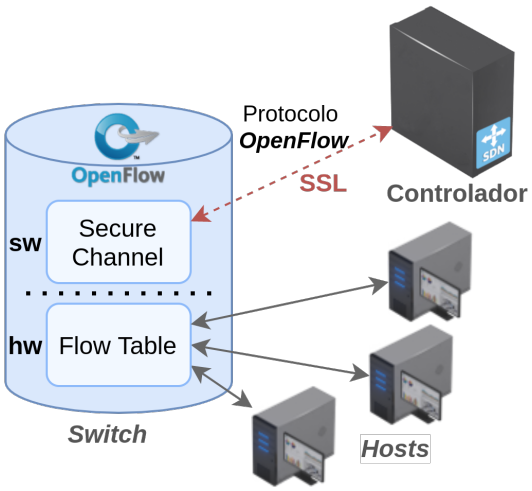


Figura 3 – Especificação em alto-nível do *OpenFlow Switch* 1.0.

Fonte: Adaptado de McKeown et al. (2008).

2.3.2 Virtualização das Funções de Rede

O NFV (CUI et al., 2012) se alinha com os avanços da pesquisa de *virtualização* ao propor a implementação de funções de rede como elementos de software virtualizados. Essa abordagem visa preencher a lacuna deixada pelos elementos com funções bem definidas na arquitetura atual, como *firewalls* e *load-balancers*, na abordagem SDN. Dessa maneira, torna-se possível alocar "máquinas virtuais" para a realização de funções de rede de maneira flexível, por se basear em software, e escalável, por ser virtualizado.

A abordagem NFV viabiliza a utilização de servidores comuns para a execução de funções de rede que tradicionalmente seriam tratadas por *appliances* como apresentado na Figura 4, o que otimiza a utilização de hardware e economiza recursos. Essa abordagem também se beneficia de todas as vantagens da virtualização de servidores, como elasticidade e resiliência, e de *cloud computing*, como processamento compartilhado e disponibilizado como serviço.

As funções de redes definidas pela abordagem NFV podem ser mais facilmente alteradas, assim como a pilha de funções de rede utilizadas no tratamento do plano de dados, pois as ligações entre as máquinas virtuais são lógicas, diferentemente da abordagem utilizada nas redes atuais. A questão do desempenho no acesso à placa de rede pelas máquinas virtuais vem sendo aprimoradas pelos principais fabricantes de componentes, como a Intel com o *Data Plane Development Kit* (DPDK), que possibilita um alto desempenho no processamento de primitivas de rede através do acesso direto ao hardware.

A Figura 5 apresenta o *framework* arquitetural da abordagem NFV, padronizado pelo *European Telecommunications Standards Institute* (ETSI). Pode-se observar pela figura que essa abordagem propõe separação entre as camadas de Orquestração e Infraestrutura assim como a SDN propõe a separação dos planos de Controle e de Dados. A ideia é

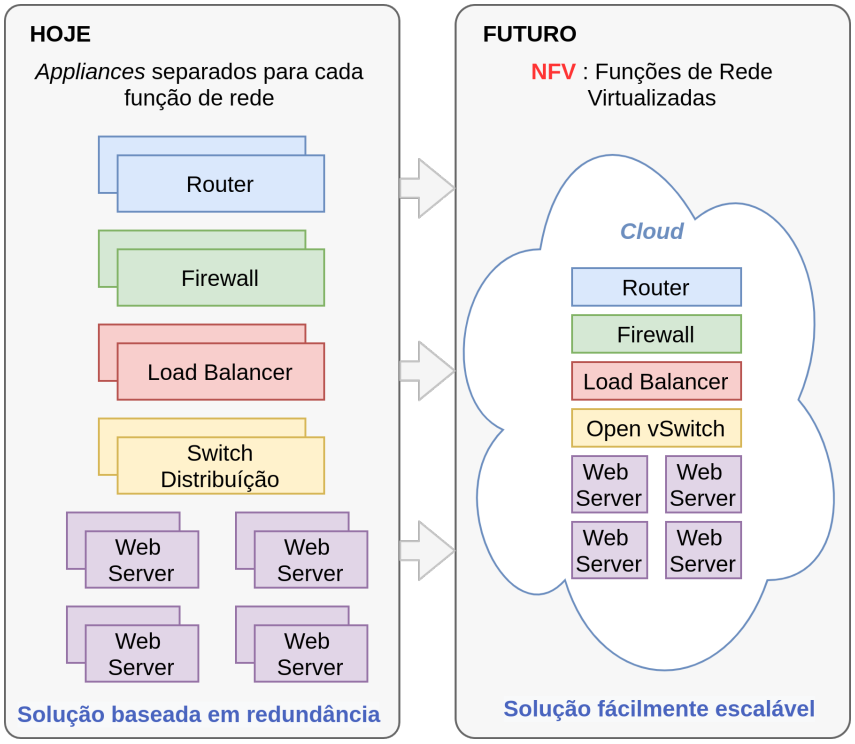


Figura 4 – Comparação entre a abordagem tradicional e a NFV para o aprovisionamento de funções de rede.

Fonte: Adaptado de Cui et al. (2012).

conceber os recursos computacionais (processamento, armazenamento e comunicação) de modo virtualizado através do *Network Function Virtualization Infrastructure* (NFVI). O *Virtualised Infrasctructure Manager* (VIM) é responsável nessa arquitetura por gerenciar recursos virtuais e de *hardware*, enquanto o VNF *Manager* gerencia as funções de rede virtualizadas.

Os componentes de controle de uma rede SDN podem ser fornecidos com uma abordagem NFV através da NFVI. Da mesma maneira, os próprios componentes do NFV podem ser virtualizados na NFVI. No entanto, como a rede é essencialmente um sistema distribuído, gerir esses componentes é uma tarefa complexa e mantê-los disponíveis é essencial para a manutenção do estado operacional da rede (HAN et al., 2015).

Pode-se afirmar que a arquitetura NFV padrão (ETSI, 2013) possui diversos componentes que carecem de automação para gerenciamento. Esses componentes podem ser observados na Figura 5, que propõe diversos elementos novos como: VNF Manager, VIM e *Orchestrator*.

2.3.3 Camada de Gerenciamento

A proposição de um 'Plano de Controle' logicamente centralizado e separado do 'Plano de Dados' é o ponto fundamental da abordagem SDN. Essa ideia, no entanto, não é de fato nova, uma vez que uma abordagem semelhante já foi aplicada na telefonia através

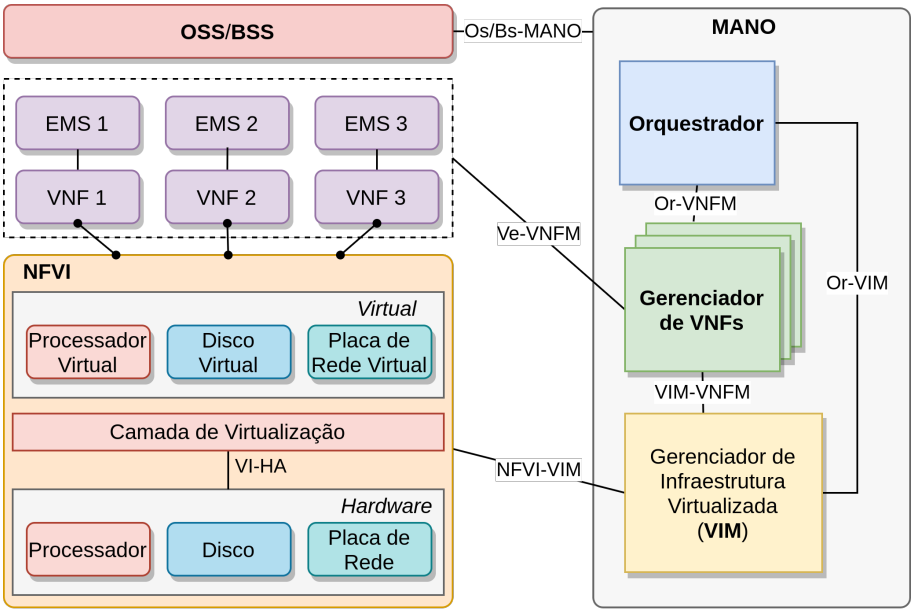


Figura 5 – Arquitetura da abordagem NFV.

Fonte: Adaptado de Cui et al. (2012).

da separação entre o tráfego de sinalização e o tráfego de voz/dados [PFR citar ITU-T Q.700]. Acredita-se que essa abordagem possa ser aplicada às redes de computadores, tornando-as mais flexíveis e programáveis e facilitando a automação da gestão, operação e controle da rede.

Na abordagem SDN, as funções de gerenciamento normalmente são providas por elementos na 'Camada de Controle', o que em tese é uma disfunção dessa camada, ou na 'Camada de Aplicação' que é uma camada genérica para a programação das redes através de elementos de *software*. Por esse motivo, alguns trabalhos como (WICKBOLDT et al., 2015) propõe uma camada específica para o gerenciamento, separada das camadas de controle e de aplicação.

Abdallah et al. (2018), por exemplo, defende a utilização de uma 'Camada de Gerenciamento' para a organização da rede e gerenciamento baseado no FCAPS. Abdallah et al. (2018) define que certos aspectos devem ser tradados na 'Camada de Controle', por exemplo, *Configuration*, *Accouting* e *Performance*; já os aspectos de *Fault* e *Security* necessariamente são atribuições da 'Camada de Gerenciamento'.

O presente trabalho abstrai as redes auto-organizadas com uma visão em camadas, bem como a abordagem SDN, mas com uma camada de gerenciamento específica que inclui componentes responsáveis pela coleta de dados, análise, previsão, diagnóstico, tomada de decisão, intervenção na infraestrutura de rede.

2.4 Estado da Arte

O objetivo geral deste trabalho está em propor uma solução de auto-gerenciamento com foco na propriedade de auto-configuração para as redes de computadores, o que representa dois temas principais correlacionados: o auto-gerenciamento (um tema mais amplo) e a auto-configuração (um tema mais específico). Desta maneira, a presente tese divide os trabalhos correlatos em duas categorias: os relacionados ao auto-gerenciamento de redes (seção 2.4.1) e aqueles com foco em operações de auto-configuração (seção 2.4.2).

2.4.1 Estado da Arte do Auto-Gerenciamento de Rede

A implementação de soluções para Auto-Gerenciamento de Rede esbarra em desafios como a configuração da infraestrutura que é naturalmente distribuída e heterogênea, e aprovisionamento de serviços com altos indicadores de desempenho requeridos pelas aplicações atuais. Com os avanços tecnológicos na camada de controle, sobretudo com a proposição do SDN (MCKEOWN et al., 2008) e NFV (CUI et al., 2012) há melhores perspectivas para a automação das redes, uma vez que os elementos de *software* introduzidos por essas abordagens podem ser mais facilmente gerenciados. A grande vantagem para o gerenciamento viabilizada pela abordagem SDN está na concepção de uma camada de controle logicamente centralizada, sobre a qual funções de gerenciamento podem ser implementadas (COX et al., 2017). Por este motivo, a maior parte dos trabalhos apresentados nesta seção utilizam pelo menos uma dessas abordagens, inclusive as propostas para o *core* do 5G. Ao final desta seção todas as propostas são organizadas na Tabela 4 que traz informações como: último ano de atividade; tipo de rede na qual a abordagem foca; o tipo de solução proposta / as tecnologias chaves; e por fim, as principais características que definem a proposta.

Tsagkaris et al. (2015) propõe o conceito de *Autonomic Network Management* (ANM) em redes SDN, que consiste na aplicação de técnicas *Autonomic Computing* (AC) para automação de funções de gerenciamento e controle. O ponto central neste trabalho está na utilização do conceito de *Autonomic Control Loop* (ACL) para monitoramento, análise e execução de procedimentos na rede. Nos trabalhos seguintes ao ANM o autor se dedicou ao auto-gerenciamento do rádio de redes de telecomunicações através do conceito de *Self-Organizing Maps* (SOMs) na implementação de *Cognitive Radio Systems* (CRSs) (TSAGKARIS; BANTOUNA; DEMESTICHAS, 2012) e mais recentemente propôs o ANM para o gerenciamento de rádio em redes 5G (PSAROMANOLAKIS et al., 2020).

Neves et al. (2016) propõe o SELFNET, uma arquitetura para orquestração, administração e controle de acesso em SDN e NFV. Este trabalho propõe a aplicação de técnicas de IA na automação de funções de gerenciamento de rede através de elementos sensores e atuadores implantados no plano de Controle e no plano de Dados com a missão de executar ações computacionais autônomas (monitoramento e recuperação). A mais recente

publicação do projeto SELFNET é o *Deliverable 7.3* em 2018 (CELDRÁN et al., 2018), que apresenta pequenos experimentos e testes unitários com os componentes da solução.

Bianchi et al. (2016) propõe o SUPERFLUIDITY, um projeto integrante do Horizon2020 (COMMISSION et al., 2015) financiado pela União Europeia que tem como objetivo projetar uma arquitetura de rede 5G com foco em desempenho, flexibilidade, agilidade e portabilidade. Esse projeto propõe a decomposição de funções e serviços de rede em blocos reutilizáveis implantados na infraestrutura chamados de *Reusable Functional Blocks* (RFB). Os RFBs propostos pelo SuperFluidity são similares às VNFs propostas pelo NFV, mas com a vantagem de serem reutilizáveis e passíveis de serem combinados para a concepção de funções de rede complexas. Este projeto propõe gerenciamento de rede através da orquestração de RFBs e tecnologias de acesso em um nível de detalhamento similar ao proposto pelo NFV. A mais recente publicação com resultados do SUPERFLUIDITY ocorreu 2018 (SOLDANI et al., 2018), e assim como no SELFNET, apresenta resultados preliminares comparando técnicas e elementos de *hardware* (e.g. processadores). Trabalhos como P5G (SHOJAFAR et al., 2017) aplicam o SUPERFLUIDITY com êxito no gerenciamento de funções para 5G e apresentam resultados similares aos apresentados nos relatórios deste projeto.

Xu et al. (2016) propõe o COGNET, uma arquitetura para o gerenciamento de rede com capacidades cognitivas voltada para o 5G com base na arquitetura proposta pelo NFV. Essa abordagem propõe três novos componentes, além daqueles definidos pelo NFV: *Data Collector*, responsável pela coleta e processamento de dados; *CogNet Smart Engine* que aplica técnicas de *Big Data* para processamento de dados obtidos da rede; e, *Policy Manager*, responsável por analisar dados gerados pela *CogNet Smart Engine* e sugerir políticas aplicáveis pelos componentes do MANO. O CogNet é um projeto parte integrante do Horizon2020 assim como o SELFNET e o SUPERFLUIDITY, e em sua última publicação ocorrida no ano de 2016 se limitou a apresentar a arquitetura, sem necessariamente implementá-la ou testá-la.

Abdallah et al. (2018) propõe uma arquitetura que contempla os aspectos do modelo FCAPS em redes SDN. Neste trabalho é proposto um componente Gerenciador acoplado ao Controlador via NBI no qual é implantada a maior parte das aplicações de gerenciamento. A distribuição sugerida pelo autor consiste em: aplicações de gerenciamento de Falha e Segurança implantadas no Controlador; e, aplicações de gerenciamento de Configuração, Contabilização e Desempenho implantadas no Gerenciador. O autor apresenta resultados básicos comparando o desempenho em três cenários: *i)* com as aplicações divididas entre o Controlador e Gerenciador como definido pela arquitetura proposta; *ii)* com todas as aplicações implantadas no controlador (abordagem SDN mais tradicional); e, *iii)* com todas as aplicações implantadas no Gerenciador (abordagem com os componentes de gerencialmente totalmente separados do controle).

Ferreira et al. (2020) propõe o ArchSDN, uma arquitetura para organização do plano

Proposta	Dados Gerais				Principais Características
	Ano	Foco	Solução	Tecnologias	
ANM (Tsagkaris et al., 2015)	2015	Telecom	Conceitual	SDN; ACL; SOM; CRS	gerenciamento de rede autônomo; *loop* de controle (abordagem ativa); proposta não implementada e nem experimentada.
SELFNET (NEVES et al., 2016)	2018	Telecom	*Framework*	SDN; NFV; IA; Cloud Computing	gerenciamento de rede autônomo; sensores e atuadores nos planos de controle e dados; foco no CORE 5G; resultados preliminares com testes unitários.
SUPERFLUIDITY (BIANCHI et al., 2016)	2018	Telecom	*Framework*	NFV; SDN; MEC; CRAN	aprovisionamento de serviços *on-the-fly*; funções de rede virtualizadas em blocos reutilizáveis; inclui resultados preliminares.
COGNET (XU et al., 2016)	2016	Telecom	Arquitetura	NFV; *Big-Data*; ML	rede com capacidades cognitivas; "extensão" da arquitetura NFV; proposta não implementada e nem experimentada.
Abdallah et al. (2018)	2018	SDN	*Arquitetura*	SDN, FCAPS, OF-CONFIG	gerenciamento de redes SDN; aplicações FCAPS; gerenciador na NBI do controlador; testes preliminares de desempenho.
ArchSDN (FERREIRA et al., 2020)	2020	SDN	*Arquitetura*	SDN, SARSA, MPLS	gerenciamento híbrido do plano de controle SDN; divisão da infraestrutura em setores; cálculo de rotas com SARSA; teste de viabilidade e resultados básicos.

Tabela 4 – Tabela Comparativa com os Trabalhos Correlatos ao tema de Auto-Gerenciamento.

Fonte: Elaborado pelo autor (2021).

de controle de redes SDN com uma abordagem de controle híbrido [3] na qual a infraestrutura é dividida em setores e distribuída pelos controladores SDN. O ArchSDN propõe uma implementação própria de controlador SDN que se conecta ao Gerenciador Central para organização dos setores e cálculo de configurações multi-setores. O ArchSDN propõe a utilização do algoritmo de *Deep-Learning State–Action–Reward–State–Action* (SARSA) (ZHAO et al., 2016) análise e predição de cenários. Os resultados apresentados em Ferreira et al. (2020) demonstram a efetividade no cálculo de caminhos considerando as comunicações dentro de um único setor, entre dois setores e entre múltiplos setores.

[3] Controle Híbrido: *leaderless* (organização horizontal) e *leader-based* (organização vertical)

2.4.2 Estado da Arte da Auto-Configuração de Rede

A propriedade de Auto-Configuração aplicada em redes de computadores se baseia em, pelo menos, três operações básicas: inicialização (*bootstrapping*) da rede, *plug-and-play* de recursos e aprovisionamento de serviços. Embora esses aspectos já tenham sido contemplados na seção anterior, a presente seção apresenta trabalhos com maior foco na Auto-configuração. Considerando-se que as abordagens SDN e NFV despontam como as principais filosofias a serem utilizadas pelas redes futuras, os trabalhos apresentados nesta seção prioritariamente baseiam-se na automação de redes concebidas com essas propostas. Similarmente à seção anterior, ao final desta seção todas as propostas são organizadas na Tabela 5 que traz informações como: último ano de atividade; tipo de rede na qual a abordagem foca; o tipo de solução proposta / as tecnologias chaves; e por fim, as principais características que definem a proposta.

Sharma et al. (2013) apresenta um processo para a configuração automática de dispositivos *OpenFlow*. Essa solução se baseia em duas etapas: *i)* atribuição de um endereço/identificador para o dispositivo, o que pode ser realizado por protocolos como o DHCP ou ferramentas como OF-CONFIG; e, *ii)* estabelecimento da sessão *OpenFlow* entre o dispositivo e o controlador, que envolve a configuração do dispositivo através de protocolos como OVSDB ou mesmo CLI. Entretanto, Sharma et al. (2013) não contempla aspectos essenciais para o *bootstrapping* da rede tais como o aprovisionamento dos controladores e a utilização de mecanismos para controle de *loops*. Do ponto de vista do *plug-and-play* de dispositivos, Sharma et al. (2013) apenas automatiza a configuração do dispositivo até a conexão com o controlador, deixando aspectos como reconfiguração do plano de controle e rebalanceamento de fluxos do plano de dados de lado. Além disso, Sharma et al. (2013) apresenta como resultado uma análise de tempo em relação a inicialização de *switches* com diferentes distâncias (em *hops*) em relação ao controlador em um ambiente simulado.

Patil, Gokhale e Hakiri (2015) propõe o InitSDN, um mecanismo de inicialização de redes SDN com suporte a controladores distribuídos. Ele define uma arquitetura com componentes responsáveis pela a automação dessa operação, são eles: *Network Discovery & Topology Service*, responsável pela descoberta de dispositivos através do protocolo LLDP; *Network Hypervisor*, responsável pela divisão da rede em dois *slices* (dados e controle); e, *Control Plane Synchronization*, responsável por prover a estratégia de sincronização entre os controladores. O fluxo de execução do InitSDN tem como requisito uma rede já descoberta e o acesso aos *switches* e não apresenta experimentos e resultados.

Katiyar et al. (2015) propõe uma abordagem para auto-configuração de *switches* SDN em redes híbridas (SDN e Legada). O fluxo proposto por essa abordagem consiste na seguinte sequência de passos: *i)* dispositivos SDN iniciam a solicitação de configurações; *ii)* o *AutoConf Server* recebe a solicitação e inicia o processo de configuração; *iii)* o *SDN Switch Locator* inicia uma busca pelo novo *switch* na infraestrutura e o *Intermediate Switch Configurator* cria fluxos de acesso ao novo *switch* através dos nós intermediários;

iv) o *AutoConf Server* troca informações com o *switch* e envia configurações; e, *v)* a sessão *OpenFlow* é estabelecida entre o *switch* e o Controlador. Katiyar et al. (2015) não considera o aprovisionamento dos componentes de controle e gerenciamento e nem o aprovisionamento de serviços e não apresenta resultados práticos.

Al-Jawad (AL-JAWAD et al., 2018; AL-JAWAD, 2018) propõe LearnQoS um *framework* para prover garantia de QoS definido por políticas em redes SDN. São utilizados componentes para monitoramento de dados de acordo com políticas de QoS aplicadas, e caso uma violação seja detectada, técnicas de adaptação baseadas na reconfiguração de fluxos e no aprovisionamento de funções de rede são realizadas. O autor propõe a utilização de Redes Neurais para aprendizado e verificação dos requisitos de QoS. Os resultados apresentados se baseiam em uma análise de latência, perda de pacotes e vazão de dados em um experimento de *streaming* de vídeo na qual uma quebra no contrato de QoS é artificialmente induzida.

Yousafzai e Hong (2020) propõe o SmartSON, um *framework* para a interconexão entre provedores e consumidores de serviços de comunicação em redes auto-organizadas. O autor propõe a utilização de uma estrutura Pub/Sub (BAEHNI; EUGSTER; GUERRAOUI, 2004), na qual provedores podem anunciar serviços como *Cloud Storages* e *Knowledge Databases*, e negociá-los através da tecnologia *block-chain* (SWAN, 2015) inspirado no algoritmo da cripto-moeda *Ethereum* (DANNEN, 2017). Trata-se de uma estratégia de auto-configuração apenas para o aprovisionamento de serviços, e sendo assim, que não contempla os aspectos de inicialização da rede e *plug-and-play* de recursos. Yousafzai e Hong (2020) apresenta resultados preliminares comparando os custos (e.g. tempo, memória e processamento) envolvido na negociação de serviços em um ambiente simulado.

Proposta	Dados Gerais				Principais Características
	Ano	Foco	Solução	Tecnologias	
Sharma et al. (2013)	2013	SDN	Método	SDN; *OpenFlow*; DHCP; LLDP	configuração automática de *switches OpenFlow*; DHCP *Server* modificado; controle *in-band*; testes simulados.
InitSDN (PATIL; GOKHALE; HAKIRI, 2015)	2015	SDN	Conceitual	SDN; *OpenFlow*	inicialização do plano de controle; controle *in-band*; particionamento e sincronização de informações; não inclui experimentos e resultados.
Katiyar et al. (2015)	2015	SDN	Método	SDN; *OpenFlow*; DHCP; LLDP	auto-configuração de *switches* em redes híbridas; DHCP *Server* modificado; controle *in-band*; não inclui resultados.
LearnQoS (AL-JAWAD et al., 2018)	2018	SDN	*Framework*	PBNM; SDN; Redes Neurais;	gerenciamento autônomo de serviços e QoS; componentes na camada de aplicação; redes neurais para a análise do QoS; resultados comparando transmissão de vídeo com e sem QoS.
SmartSON (YOUSAFZAI; HONG, 2020)	2020	Telecom	*Framework*	NFV; *blockchain*; SON;	auto-configuração de serviços; abordagem baseada em *blockchain*; estratégia *Pub/Sub*; requer uma rede em estado operacional; experimentos com negociação simuladas.

Tabela 5 – Tabela Comparativa com os Trabalhos Correlatos ao tema de Auto-Configuração.

Fonte: Elaborado pelo autor (2021).

Capítulo 3

Método de Pesquisa

Este capítulo apresenta o percurso metodológico utilizado para responder os problemas de pesquisa e alcançar os objetivos definidos no Capítulo 1. A primeira seção introduz o método de pesquisa e as seções seguintes detalham as etapas da pesquisa de acordo com a *Design Science Research Methodology* (DSRM) (PEFFERS et al., 2007). Os aspectos conceituais relacionados ao método de pesquisa, incluindo detalhes sobre a DSRM, são descritos no Apêndice B.

3.1 Método de Pesquisa do Trabalho

O presente trabalho utiliza a *Design Science Research Methodology* (DSRM) (PEFFERS et al., 2007) como metodologia para o *design* de uma solução que resolve o problema de automação do gerenciamento e da configuração de redes de computadores. O ponto de partida escolhido de acordo com processo proposto por esta metodologia (descrito em detalhes no Apêndice B) foi a 'Identificação do Problema e Motivação' o que fornece uma base sólida para a proposição dos artefatos que compõe a solução. Nesta etapa foi realizado um levantamento bibliográfico com o objetivo de fomentar uma base conceitual sobre redes de computadores e de telecomunicações, gerenciamento de rede e sistemas autônomos, para enfim proceder com a escolha dos problemas de pesquisa que são relacionados ao auto-gerenciamento de redes de computadores com enfoque na propriedade de auto-configuração.

Desta maneira, verificou-se que a pesquisa desenvolvida nesta tese é de 'Natureza Aplicada', uma vez que ela se propõe a resolver um problema prático que pode ser aplicado de maneira imediata na especificação de redes futuras, como por exemplo na proposta de *5th Generation Mobile Network Stand Alone* (5G-SA) que carece de uma solução arquitetural para o núcleo da rede (5GPPP, 2019). Trata-se de um tema relevante mesmo para as redes atuais, o que pode ser verificado ao considerar a importância do gerenciamento de rede na manutenção do estado operacional da rede e o aumento da complexidade de gerencia-

mento em decorrência do crescimento da infraestrutura e utilização de novas tecnologias (BENSON; AKELLA; MALTZ, 2009). Dentre os inúmeros problemas em abertos relacionados à automação do gerenciamento de rede destacam-se as operações de configuração que visam o estabelecimento do estado operacional, capacidade de extensão e aprovisionamento de serviços de comunicação (WOJCIECHOWSKI; SIERSZEń; STURGULEWSKI, 2016).

A segunda etapa do processo de pesquisa da metodologia DSRM proposto por (PEFFERS et al., 2007) é a definição de 'Objetivos da Solução' que consiste em identificar os artefatos necessários para a solução. Dessa maneira, foram definidos 'Objetivos Exploratórios' através de uma análise sobre o estado da arte. Estes objetivos visaram gerar conhecimento sobre o tema de auto-gerenciamento com ênfase na propriedade de auto-configuração (WOJCIECHOWSKI; SIERSZEń; STURGULEWSKI, 2016). Para isso o presente trabalho propõe soluções em diferentes níveis de abstração: *i)* modelos conceituais para organização da rede e aprovisionamento de serviços; *ii)* especificação de um arquitetura para redes auto-organizadas; *iii)* implementação de um *framework* de auto-gerenciamento; e, *iv)* implementação de uma plataforma de auto-configuração que automatiza as operações de *bootstrapping*, *plug-and-play* e aprovisionamento de serviços. Nesta tese também é proposta uma ontologia para representação de serviços com alta expressividade semântica e uma lista de requisitos comunicacionais levantadas por meio do processo de 'Engenharia de Requisitos'.

Na etapa de '*Design e Desenvolvimento*' os artefatos descritos como objetivos foram criados, e por estarem fortemente relacionados (a plataforma utiliza o *framework* que implementa a arquitetura que aplica os modelos), optou-se por uma estratégia de validação por experimentação. Este processo iniciou-se pela definição de dois modelos conceituais: o primeiro batizado de *Intent-Based Service Model* (IBSM) que especifica mecanismos e estruturas responsáveis pela abstração de serviços orientado por intenções; e o segundo *Self-Organizing Network Model* (SONeM) que define camadas e subcamadas responsáveis pelo gerenciamento da rede. Em seguida foi especificada a *Self-Organizing Network Architecture* (SONAr) uma arquitetura que abstrai os modelos IBSM e SONeM para a concepção de redes auto-organizadas baseada em conceitos de computação autônoma. Por fim, os principais conceitos da SONAr foram materializados por um *framework* de auto-gerenciamento batizado de *Self-Organizing Network Framework* (SONAr-Framework) que é aplicado na implementação da *Self-Configuration and Management Platform* (SeCoMP), uma plataforma de auto-configuração com foco automação das operações de inicialização, *plug-and-play* e configuração de serviços em redes SDN com suporte aos protocolos *OpenFlow* e OVSDB.

Na etapa seguinte que compreende a 'Demonstração' da solução, foram definidos cenários de experimentação próximos da realidade relacionados às operações de configuração com o objetivo de validar a SeCoMP por meio do método de 'prova de conceito'. Foi

utilizada uma estratégia de emulação de rede com *switches* virtuais (com suporte aos protocolos *OpenFlow*, OVSDB, DHCP, LLDP, STP e SNMP) e servidores com suporte a acomodação dos componentes de controle e gerenciamento com uma abordagem de microsserviços e abstraídos como contêineres. Foram realizados experimentos com diferentes técnicas em topologias de rede com 1, 2, 4, 8, 32, 64 e 128 *switches*.

Na etapa de 'Avaliação' foram adotadas ambas as abordagens 'qualitativa' e 'quantitativa'. Os ganhos qualitativos se referem à maior expressividade semântica e flexibilidade na definição de serviços comunicacionais; garantia de QoS de maneira automática; e, simplificação das operações de configuração que passam a exigir menos (ou nenhum) conhecimento técnico por parte dos administradores de rede. Os ganhos quantitativos se referem à uma análise do tempo de configuração das operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços com a solução proposta neste trabalho considerando diferentes técnicas e redes com variadas proporções.

Por fim, na etapa de 'Comunicação' a presente pesquisa se dedicou a realização de publicações e escrita desta tese. Foi publicado um artigo com resultados preliminares na edição 2020 do *International Conference on Cloud Computing and Services Science* (CLOSER) e foi submetido um artigo com os resultados desta tese ao IEEE *Transactions on Network and Service Management* (TNSM) no tópico *Smart Management of Future Softwarized Networks*. Além disso foram realizadas contribuições para outras quatro publicações encabeçadas por outros membros do grupo de pesquisa.

Os principais métodos utilizados no curso deste trabalho foram: (a) 'levantamento bibliográfico', utilizada com o objetivo para a construção de uma base de conhecimento sobre o assunto e levantamento do estado da arte; (b) 'levantamento de informações', na qual profissionais com notório saber foram entrevistados para aprimoramento da solução; (c) 'especificação técnica', por meio da proposição e documentação dos modelos conceituais, da arquitetura e do *framework* para redes auto-organizadas apresentada neste trabalho; (d) 'estudo de caso', através da exploração de cenários com foco nas operações de configuração; (e) 'prova de conceito', através da implementação do *framework* e da versão básica de uma plataforma com foco na automação das operações de configuração; e, (f) 'análise *ex-post-facto*', por meio da análise dos resultados obtidos com testes preliminares e refinamento da solução.

3.2 Etapa 1: Identificação do Problema e Motivação

Segundo Zanella (2006) a pesquisa é fundamentada e construída de maneira a resolver ou esclarecer um problema. Para Gil (2008) "*problema é qualquer questão não solvida e que é objeto de discussão, em qualquer domínio do conhecimento*". Dessa maneira, o processo de formulação do problema é etapa primordial para a pesquisa e pode ser realizado a partir das recomendações básicas de Gil (2008): o problema deve ser formulado

como uma pergunta; o problema tem que ter dimensão viável; e, o problema deve ser claro e preciso. De acordo com Zanella (2006), o problema necessita de uma resposta "*provável, suposta ou provisória*", que é a hipótese da pesquisa. Lakatos e Marconi (2003) denominam a resposta principal de um problema como *hipótese básica*, e as respostas complementares que viabilizam a solução como *hipóteses secundárias*.

3.2.1 Tema de Pesquisa

O processo adotado por este trabalho para a formulação do tema de pesquisa ocorreu através de ciclos de refinamentos, i.e. *stepwise refinement* (BUTLER, 1996), com o objetivo de permitir o aprofundamento do tema de pesquisa de maneira gradual e efetiva. Dessa forma, este trabalho definiu as redes de computadores como objeto de pesquisa. A escolha se deu de maneira subjetiva considerando a experiência e interesse do pesquisador, o que é considerado por Moresi et al. (2003) como um ponto de partida para a definição de um assunto de pesquisa. Com base neste objeto, o presente trabalho se dedicou a explorar aspectos conceituais e práticos relacionados às redes de computadores e de telecomunicações através do método de 'levantamento bibliográfico' e estudo de livros conceituados sobre o tema como a publicação de Kurose e Ross (2006).

Na primeira etapa deste processo pôde-se observar que os principais protocolos destas redes (IP, TCP, UDP e ICMP) pouco evoluíram nos últimos quarenta anos, o que pôde ser concluído pelo baixo número de RFCs publicadas neste período, ao mesmo tempo que o contexto de utilização das redes passou por uma mudança drástica (PEREIRA, 2012). Pôde-se observar também o papel fundamental das redes de computadores e de telecomunicações na sociedade que se tornaram imprescindíveis para aspectos sociais, humanos e econômicos (CHARAN, 1991). Nesse sentido o gerenciamento de rede desempenha um papel fundamental, por lidar com as características ultrapassadas da arquitetura vigente, ao mesmo tempo em que é cobrado para (tentar) garantir os requisitos de alto desempenho e disponibilidade das aplicações atuais (CLEMM, 2006; FETTWEIS, 2014). Assim, o presente trabalho definiu o 'Gerenciamento de Rede' como assunto de pesquisa, por entender este como tema relevante, de interesse do grupo de pesquisa, com problemas em aberto e de aplicação prática.

O Gerenciamento de Rede (NM – *Network Management*) é responsável por manter a rede operacional, por aplicar políticas administrativas e por aprovisionar serviços de comunicação. A primeira utilização desse termo ocorreu no final da década de oitenta através do padrão FCAPS (STEVENSON, 1995). Considerando-se que a análise do *status quo* da matéria sobre gerenciamento de redes tradicionais é fundamental para a exata compreensão do problema sendo modelado, a segunda etapa do processo consistiu na utilização dos método de 'levantamento bibliográfico' e 'levantamento de informações'. Este trabalho se amparou na bibliografia tradicional relacionada gerenciamento de rede – *Integrated Management of Networked Systems* (HEGERING, 1999), *Network Management Funda-*

mentals (CLEMM, 2006) e *Network Management: Principles and Practice* (SUBRAMANIAN, 2010). No curso deste levantamento verificou-se a necessidade de complementar as informações acerca do assunto através da consulta de padrões ISO (UNION, 1989; ISO/IEC, 1991; ISO/IEC, 1990), e RFC's de protocolos e *frameworks* de gerenciamento (BRADEN, 1989; ENNS, 2006; BJORKLUND, 2010; BIERMAN et al., 2017). Para estabelecimento do 'estado da arte' o presente trabalho partiu do capítulo publicado sobre o tema por Ding (2016) e se aprofundou em trabalhos mais específicos (QUITTEK et al., 2004; AL-JAWAD, 2018; KIM; FEAMSTER, 2013). Já o levantamento de informações se deu a partir de entrevistas com profissionais responsável pela operação e administração das redes *metro-ethernet*, SDH e 3G da Algar Telecom, na qual pôde-se constatar a real complexidade de gerenciamento imposta pelas ferramentas e técnicas atuais.

Considerando o levantamento do 'estado da arte' e de 'informações da indústria' sobre o gerenciamento de rede verificou-se que grande parte do custo de provedores de acesso e empresas de TI, hoje em dia, está em atividades operacionais (OPEX), que requerem uma mão de obra muito especializada e que introduzem riscos inerentes à intervenção humana. Esse problema se agrava ao observarmos tendências de redes futuras como o 5G e aplicações de rede como IoT. Verificou-se também uma tendência de trabalhos correlatos ao afirmarem que uma abordagem de gerenciamento manual não suportaria os indicadores de desempenho e serviços requeridos por essas tecnologias (PAUL; PAN; JAIN, 2011). Assim, a presente pesquisa optou pelo tema de 'Redes Auto-Organizadas', termo este que foi cunhado pelo 3GPP (2018c) para a automação do plano de sinalização das redes de telecomunicações da terceira geração.

'Redes Auto-Organizadas' são sistemas distribuídos autônomos, e por este motivo este trabalho se dedicou a compreender os conceitos de 'Computação Autônoma' (HORN, 2001) e como eles podem ser empregados na concepção de 'Sistemas Autônomos' através do suporte às propriedades *self-** [1] (BERNS; GHOSH, 2009). A 'Computação Autônoma' é uma estratégia que propõe eliminar (ou limitar) a necessidade de intervenção humana no gerenciamento de sistemas que pode ser utilizada na concepção de redes auto-organizadas. Entretanto, a adaptação dos conceitos de computação ao universo das redes não é tarefa trivial devido à natureza distribuída das redes e ao fato de que elas são ao mesmo tempo pré-requisitos para a comunicação entre os elementos que as controlam.

Por se tratar de um tema amplo, no último ciclo este trabalho decidiu por focar no aspecto específico de configuração como 'estudo de caso' para a solução proposta. A propriedade de configuração tem a missão de tornar a rede operacional, bem como, suportar novos recursos e serviços de maneira automática. Assim, ela engloba as operações de inicialização de rede, *plug-and-play* de dispositivos e aprovisionamento de serviços que foram definidos como pontos focais deste trabalho. A escolha do aspecto de configuração

[1] e.g. auto-configuração, auto-cura, auto-otimização, auto-proteção, auto-orquestração, auto-aprendizado e auto-consciência

de maneira autônoma foi uma escolha natural que simboliza o ponto de partida para o auto-gerenciamento, uma vez que é a partir da auto-configuração que a rede alcança o seu estado operacional e se torna útil para a comunicação. Além disso, essa escolha se pautou na observação do 'estado da arte' no qual constatou-se a inexistência de uma solução prática de auto-gerenciamento com foco em configuração.

3.2.2 Estado da Arte

O 'Estado da Arte' permite contextualizar a pesquisa, definir o estágio atual de trabalhos correlatos, e justificar a relevância do trabalho. A maneira como esta pesquisa realizou o processo de estabelecimento formal do estado da arte iniciou pelo método de 'levantamento bibliográfico' com propostas implantadas ou concebidas pela indústria. Dentre as propostas avaliadas a que tem maior destaque é o SON (3GPP, 2018c), que automatiza os aspectos de configuração, cura e planejamento em redes de telecomunicações e foi utilizado na implantação do 3G. Além disso, SDN2 (KOMPELLA, 2019) e IBN (LALIBERTE, 2018) são propostas amplamente divulgadas que demonstram o interesse da indústria em avançar no aspecto de automação do gerenciamento de rede e flexibilização na utilização dos serviços de comunicação. Ambos os trabalhos, no entanto, são conceituais e por isso não podem ser aplicados de fato na automação das redes atuais.

O estado da arte a respeito do tema de gerenciamento de rede apresentado nesta tese foi aprofundado a partir da utilização de um *survey* sobre requisitos de operação e gerenciamento para redes futuras publicado por López et al. (2017). A partir desta publicação foi realizado um trabalho investigativo de 'levantamento bibliográfico' onde foram encontrados outros trabalhos relacionados à automação do gerenciamento de rede, dentre eles destacam-se: Neves et al. (2016) que propõe o SELFNET como uma solução de automação do gerenciamento com base em SDN e NFV para o núcleo do 5G; Xu et al. (2016) que propõe o COGNET como arquitetura com componentes acoplados ao plano de controle responsáveis pelo gerenciamento com capacidades cognitivas; e, Ferreira et al. (2020) que propõe ArchSDN, uma abordagem de distribuição e auto-orquestração do plano de controle.

Com o enfoque dado à propriedade de configuração definido de acordo com o refinamento do tema de pesquisa descrito na seção anterior, o presente trabalho mais uma vez iniciou o levantamento bibliográfico para estabelecimento formal do estado da arte a partir de um trabalho de *survey*. Neste caso o artigo usado como referência foi o Wojciechowski, Sierszeń e Sturgulewski (2016), e em seguida foram considerados outros trabalhos mais recentes encontrados através de um processo investigativo de 'levantamento bibliográfico' a partir de publicações conceituadas (e.g. *Springer Series*) e ferramentas de indexação acadêmica (e.g. *Google Scholar*). Assim, a presente pesquisa avaliou vários trabalhos que se relacionam à automação da configuração, sobretudo em redes definidas por *software*. Dentre estes trabalhos destacam-se: Patil, Gokhale e Hakiri (2015) que propõe o InitSDN,

uma solução para auto-inicialização do plano de controle com uma estratégia *in-band*; Al-Jawad et al. (2018) que propõe o LearnQoS, uma solução que aplica conceitos de redes neurais na configuração e garantia de políticas de QoS; e, Yousafzai e Hong (2020) que propõe o SmartSon, uma solução para auto-configuração de funções de rede e serviços de comunicação a partir de uma abordagem *pub/sub* com *block-chain*.

3.2.3 Problemas e Hipótese de Pesquisa

O principal objetivo da primeira etapa da DSRM (PEFFERS et al., 2007) consiste na identificação de um ou mais problemas de pesquisa e formulação das hipóteses de maneira a guiar o desenvolvimento da solução. Este processo baseou-se na análise das questões em aberto relacionadas ao tema de *'automação das operações de configuração em redes auto-organizadas'* através de três diferentes perspectivas: *(i)* provedores de serviço, *(ii)* usuários finais, e *iii)* desenvolvedores de aplicações. Na perspectiva *(i)*, a auto-configuração da rede é um habilitador do negócio que ajuda a mitigar riscos e reduzir custos; na perspectiva *(ii)*, a auto-configuração representa uma melhor experiência com maior agilidade na implantação e melhores níveis de disponibilidade; por fim, na perspectiva *iii)* a auto-configuração atua de maneira a garantir que os requisitos comunicacionais definidos por suas aplicações sejam atendidos. Dessa maneira, o presente trabalho definiu problemas com foco na automação, organização e flexibilidade da rede.

O método aplicado na identificação dos problemas de pesquisa foi similar ao método utilizado na escolha do tema (descrito na seção 3.2.1), de maneira que os ciclos de refinamentos envolveram não apenas a definição temas mais específico a cada iteração, mas também a identificação de problemas mais específicos relacionados ao tema e formulação de hipóteses. Assim ao ser definido o tema de 'Gerenciamento de Rede' como foco do trabalho, foi definido como problema de pesquisa a questão sobre *'Como reduzir a complexidade do gerenciamento de rede?'*, ao que se respondeu com a hipótese de que *'Conceitos de computação autônoma são capazes de reduzir a complexidade do gerenciamento de rede'*. Com o refinamento do tema de pesquisa que passou a focar no aspecto de 'Auto-Gerenciamento de Rede' o problema levantado foi *'Como automatizar as operações de gerenciamento de redes'*, ao que se respondeu com a hipótese de que *'Componentes responsáveis pela coleta, análise, aprendizado, tomada de decisão e intervenção na rede são capazes de automatizar as principais operações de gerenciamento de rede sem intervenção humana.'*. Ao especializar ainda mais o tema no contexto da propriedade de auto-configuração o problema definido que se tornou o alvo do presente trabalho foi *'Como flexibilizar a configuração de serviços de comunicação e dinamizar/automatizar os processos de inicialização e extensão da rede'*, ao que se respondeu com a hipótese definida na seção 1.2: *'É possível melhorar a velocidade e flexibilidade das operações de gerenciamento através da automação da configuração dos elementos de infraestrutura.'*.

Considerando-se as diferentes perspectivas envolvidas na comunicação e o problema/hipótese geral do trabalho foram identificados três problemas de pesquisa mais específicos à serem respondidos (conforme descrito na seção 1.2.1): *i) 'Como automatizar a inicialização da infraestrutura de rede e dos componentes de controle e gerenciamento?'*, *ii) 'Como permitir o acoplamento e desacoplamento automático de dispositivos na infraestrutura de rede?'* e *iii) 'Como traduzir os requisitos de aplicações e corporações em instruções de configuração da infraestrutura de rede?'*. Pode-se observar que cada um dos problemas utilizados responde à um aspecto da propriedade de configuração no âmbito das redes auto-organizadas, que é o tema focal do presente trabalho.

3.3 Etapa 2: Definição de Objetivos da Solução

Os objetivos da solução representam aspectos que desejam-se alcançar de maneira a resolver os problemas de pesquisa e/ou comprovar a hipótese. (HEVNER; CHATTERJEE, 2010) define que os objetivos podem ser: *i)* quantitativos, através de uma especificação sobre como a nova solução deve ser melhor que a atual; ou, *ii)* qualitativos, através da especificação de novos artefatos que suportam a solução proposta. Considerando o problema central tratado nesta tese que se refere *'automação das operações de configuração em redes auto-organizadas'*, os objetivos escolhidos são em sua maioria qualitativos e se referem à concepção de artefatos capazes de resolver aspectos relacionados ao problema, tais como o suporte à serviços flexíveis com definição de alto nível e automação do gerenciamento da rede.

De maneira a planejar os objetivos da pesquisa foi especificada uma solução preliminar de alto nível com base na levantamento bibliográfico, estabelecimento do estado da arte, investigação do problema, entrevista com administradores/operadores de redes reais e discussão com outros pesquisadores do mesmo tema. Dessa maneira, definiram-se artefatos mais genéricos a partir dos quais foram definidos artefatos mais próximos dos especificados como objetivos da solução.

Essa estratégia serviu como base para o planejamento da solução proposta por esta pesquisa que partiria da concepção de 'modelos conceituais' para representação de aspectos fundamentais do problema relacionados ao gerenciamento e configuração da rede. Com base nos modelos resultantes planejou-se a especificação de 'arquitetura conceitual' para a concepção de redes auto-organizadas com um maior nível de detalhes sobre componentes, fluxos de trabalho e técnicas de integração. Com o objetivo de aprofundar a solução no tema de auto-configuração foi planejado também um 'estudo de caso' que envolveria a automação de operações de configuração. Por fim, planejou-se uma etapa de implementação que consistiria na construção de um *framework* de gerenciamento como base para a implementação de uma plataforma de auto-configuração com foco nos casos elencados no 'estudo de caso' com operações de configuração. Considerando a necessidade

de validação da pesquisa planejou-se uma etapa de experimentação desta *plataforma* que serviria como 'prova de conceito'.

3.3.1 Objetivos Geral e Objetivos Específicos

Segundo Peffers et al. (2007) os objetivos definidos por uma pesquisa devem ser capazes de serem inferidos a partir da especificação do problema, e para isso pode-se amparar no levantamento do estado da arte e aprofundamento do tema de pesquisa. Neste trabalho pôde-se inferir que o objetivo geral seria o de *'especificar uma plataforma de gerência para redes auto-organizáveis com foco nas operações de inicialização da rede, plug-and-play de dispositivos e aprovisionamento de serviços'*, e os principais objetivos específicos seriam: *i)* modelar serviço de comunicação formalmente; e, *ii)* especificar uma solução arquitetural para a organização da rede. Estes dois objetivos se relacionam aos principais aspectos contemplados pelo problema de pesquisa: o gerenciamento da infraestrutura e configuração de serviços.

A partir da especificação preliminar de uma solução de alto nível para o problema tratado por este trabalho pôde-se abstrair os requisitos da solução ainda nos estágios iniciais do projeto de pesquisa. Planejou-se uma solução baseada em um conjunto de artefatos, para os quais foram definidos objetivos específicos de maneira a especificá-los e construí-los. Dessa forma, os objetivos específicos definidos nesta tese na seção 1.3.1 representam de maneira genérica estes artefatos. Os objetivos definidos foram: *1) 'especificar modelos conceituais de redes autônomas com serviços flexíveis aos requisitos comunicacionais atuais e futuros'*; *2) 'descrever uma proposta de arquitetura de redes auto-organizadas e investigá-la por meio de um estudo de caso com operações de configuração'*; *3) 'implementar uma solução prática e flexível para a automação do aspecto de configuração no gerenciamento de rede'*; *4) 'levantar requisitos comunicacionais funcionais e não-funcionais das aplicações, usuários e provedores de serviço'*; *5) 'formalizar de maneira ontológica uma definição de serviço comunicacional flexível'*; e, *6) 'investigar a efetividade a solução proposta por meio de experimentação com cenários que envolvam a configuração de componentes, dispositivos e serviços'*. Deve-se destacar que os objetivos específicos da solução foram revistos durante a etapa de *design* da solução, de maneira a incluir, alterar ou remover objetivos de acordo com o amadurecimento da solução proposta por este trabalho.

3.4 Etapa 3: *Design* e Desenvolvimento

Um dos princípios básicos para a realização de uma DSR, que é representado pela *Guideline 1* de Hevner e Chatterjee (2010), é que o *design* deve ser centrado em artefatos. Em outras palavras, a solução para o problema de pesquisa deve ser especificada em

termos de artefatos, de maneira que ao construí-los a solução é materializada. Essa ideia se assemelha ao conceito de 'Separação de Preocupações' (SoC) proposto por *Dijkstra* em seu famoso artigo de 1982 *"On the role of scientific thought"* (DIJKSTRA, 1982), que é o princípio básico por trás do *design* de sistemas modulares. A "filosofia" do SoC consiste no pensamento de que diante da impossibilidade de responder à todas as questões relacionadas ao problema simultaneamente, deve-se tratar cada aspecto isoladamente mas com uma visão holística, de maneira que ao se responder/resolver uma dessas questões contribui-se com uma parte da solução do todo.

A divisão da solução em artefatos permitiu um melhor planejamento das atividades através de uma definição mais clara dos objetivos e de marcos de "entrega". Acredita-se também que os artefatos criados nesta pesquisa possam contribuir para o estado da arte de maneira independente, servindo como base para outras pesquisas. Por exemplo, a definição ontológica de serviço de comunicação descrita pela IBSO pode ser utilizada por outras soluções independente da utilização dos conceitos definidos pelo IBSM, SONeM e SONAr. Os detalhes de *design* bem como as questões relacionadas ao desenvolvimento dos artefatos da solução são apresentados nesta seção.

3.4.1 Modelo de Rede Auto-Organizada

Modelos têm sido utilizados desde as primeiras redes de computadores, sendo o 'Modelo OSI' (DAY; ZIMMERMANN, 1983) o mais conhecido e influente modelo conceitual de rede. Este modelo representa sistemas interconectados de maneira geral e estabelece princípios funcionais para o tratamento da comunicação por meio de camadas. O 'Modelo OSI' serviu de como referência para a especificação da *Internet* o que resultou no 'Modelo *Internet*' (também conhecido como 'Modelo TCP/IP') que é especificado indiretamente na proposta do *Internet Protocol* (IP) (POSTEL, 1981) e que define quatro camadas para organização dos protocolos de controle e tratamento de dados (KUROSE; ROSS, 2006), são elas *'Enlace' (layer 2)* – responsável pelo envio de *frames* para dispositivos na mesma rede; 'Internet'/'Rede' (layer 3) – responsável pelo envio de *pacotes* entre redes interconectadas; 'Transporte' (layer 4) – responsável pelo endereçamento dos processos comunicantes e tratamento dos *segmentos* nos sistemas finais; e, 'Aplicação' (layer 7) – responsável pelo tratamento de aspectos relacionados ao uso da rede.

A abordagem SDN (MCKEOWN et al., 2008) (seção 2.3.1) propõe um modelo que define a separação da comunicação em dois planos: o de *controle* que inclui todas instruções de definição de fluxos, políticas de QoS, e primitivas de protocolos como ARP, LLDP e STP; e, o de *dados* que é utilizado para o tráfego da informação em si. A abordagem SDN também apresenta um modelo de organização da rede com base em três camadas: *infraestrutura*, que engloba os dispositivos, servidores e recursos de *hardware* ou virtualizados utilizados para a transmissão da informação; *controle*, que engloba os componentes de controle da rede, tais como o MANO (proposto pelo NFV) e o 'Controlador' (proposto

pelo SDN); e, *aplicação*, que a contempla os elementos de *software* capazes de alterar o comportamento de rede.

Ao analisar os modelos apresentados acima, verifica-se o aspecto de 'gerenciamento' não foi devidamente tratado por nenhuma delas, sobretudo a matéria de 'auto-gerenciamento'. De fato, o FCAPS (ISO/IEC, 1989) (seção 2.1.2), um *framework* de gerenciamento de rede que é utilizado ainda hoje como um padrão para as soluções de gerenciamento, só foi proposto anos mais tarde da proposição dos modelos OSI e *Internet*. Observa-se também que a SDN propõe uma camada genérica para as aplicações definição de comportamento da rede que pode ser utilizada para o gerenciamento ou quaisquer outras funções, o que tem sido revisto na literatura correlata através da proposição de uma 'Camada de Gerenciamento' (seção 2.3.3).

Neste contexto o presente trabalho propõe o *Self-Organizing Network Model* (SONeM), um modelo pra representação de redes auto-organizadas que tem enfoque no aspecto de auto-gerenciamento. Esta definição se dá por meio do método de conceituação com base na aplicação da 'Teoria do Conceito' na 'Modelagem Conceitual' proposta por Wand et al. (1995). O SONeM é um modelo conceitual que abstrai as rede auto-organizada a partir de uma abordagem de camadas. As quatro camadas propostas pelo SONeM são: *1) Infraestrutura*, que compreende elementos habilitadores da comunicação como *switches*, roteadores, *appliances* e servidores; *2) Controle*, que engloba os componentes utilizados no controle da rede (e.g. Controlador SDN); *3) Gerenciamento*, que considera todos os componentes necessários para a automação do gerenciamento da rede; e, *4) Administração*, que representa o ambiente de administração da rede.

A 'Camada de Gerenciamento' proposta pelo SONeM é o aspecto fundamental deste modelo, que permite a observação de detalhes sobre a integração com os recursos de infraestrutura, componentes de controle e ferramentas administrativas. O SONeM trata o aspecto de 'auto-gerenciamento' através da definição de subcamadas e de um fluxo de automação baseados em funções de coleta, análise e intervenção. O SONeM é descrito em detalhes na seção 4.2.

3.4.2 Modelo de Representação e Aprovisionamento de Serviços

Wand et al. (1995) propõe a 'teoria dos atos de fala' como método da linguística para modelagem conceitual. Essa estratégia permite uma abstração de alto-nível para a modelagem conceitual, uma vez que a preocupação sobre o modelo considera mais o aspecto semântico que sintático de sua representação. Pode-se observar uma similaridade entre este conceito (atos de fala) e a técnica de representação declarativa (FAHLAND et al., 2009) que permite uma modelagem conceitual como foco em declarações (*statements*). Uma forma comum de exemplificar a representação declarativa está em compará-la com

a imperativa da seguinte maneira: enquanto método declarativo se preocupa em definir "o que", o imperativo "como".

O conceito de intenção também guarda relação com o método resultado da aplicação da 'teoria dos atos de fala' pela apreensão da motivação da comunicação. A utilização de intenções para guiar a execução de um sistema autônomo representa o maior nível de automação de acordo com a definição de Ganek e Corbi (2003). Dessa maneira, o presente trabalho se dedicou a encontrar meios para viabilizar a definição de comportamentos de rede baseados em intenções. Uma possibilidade para representação de intenções é aplicação do conceito de 'intenção declarativa' (DI), que se refere à representação de intenções por meio da abordagem declarativa (LALIBERTE, 2018). Esse conceito se mostra especialmente útil na representação de serviços com alta expressividade semântica.

Neste contexto, o presente trabalho propõe o *Intent-Based Service Model* (IBSM), um modelo conceitual de serviços baseados em intenções que se divide em outros dois modelos: *i) Intent-Based Service Representation Model* (IBSRM), modelo para representação de serviços baseados em intenções; e, *ii) Intent-Based Service Provisioning Model* (IBSPM), modelo para aprovisionamento de serviços baseados em intenções. O modelo de representação (IBSRM), se baseou em outros dois artefatos da solução para a sua concepção: a 'Ontologia de Serviço Baseado em Intenções' (IBSO) e o 'Levantamento de Requisitos Comunicacionais'. Através da proposição do IBSO tornou-se possível uma representação formal para a definição de serviço baseado em intenção, o que é enumerada como uma das possibilidades de representação propostas pelo IBSRM. Outras duas opções se referem à utilização de linguagem natural, o que requer a interpretação das intenções através de técnicas de NLP (CHOWDHURY, 2003); e, inferência por comportamentos, que se baseia no conceito proposto neste trabalho de 'intenção comportamental' inspirado na abordagem imperativa, na qual propõe-se identificar a necessidade de serviços por meio de detecção de padrões de comportamento e mudanças de contexto.

O modelo de aprovisionamento de serviços baseados em intenções (IBSPM), é um modelo conceitual que se aproxima da definição de 'Arquitetura Conceitual', uma vez que define componentes conceituais e fluxos de trabalho o que é característico deste tipo de representação. O IBSPM define um fluxo de aprovisionamento cíclico que se baseia nas atividades de definição, interpretação, configuração, validação e manutenção do serviço. O IBSPM também define componentes conceituais de alto nível para a execução deste fluxo e inclui componentes específicos para aspectos adicionais como a interpretação de linguagem natural e detecção de comportamentos na rede. Os principais componentes definidos por este modelo são: *i)* interpretador, responsável por transformar as definições de serviços em instruções de configuração; o *ii)* ativador, responsável por implantar as configurações na rede; e, o *iii)* validador, responsável por validar se a configuração foi efetiva ao prover o serviço. O IBSM é descrito em detalhes na seção 4.1.

3.4.3 Ontologia de Serviço Baseado em Intenções

Ontologia refere-se à vertente da filosofia que remonta a metafísica aristotélica responsável pelo estudo do ser enquanto ser, ou seja, da natureza inerente a todos os seres (DUTRA; IBERTIS, 2003). O termo tem origem do grego *ontos* (ser) e *logos* (estudo, discurso, palavra) e é definido por (HARPER et al., 2001) como "*estudo dos seres e da essência das coisas*". A ontologia como método de modelagem conceitual é utilizada para representação do domínio do objeto de estudo e é tradicionalmente descrita através da *Web Ontology Language* (OWL) (HORRIDGE; BECHHOFER; NOPPENS, 2007), uma linguagem formal pra definição ontológica que se baseia na descrição de classes, indivíduos, atributos, propriedades, relações e restrições através de axiomas.

No processo de *design* e desenvolvimento do *Intent-Based Service Representation Model* (IBSRM), que é o modelo responsável pelo aspecto de representação dos *Intent-Based Services* (IBSs), identificou-se a necessidade de uma definição formal do conceito de *serviço baseado em intenção* o que se deu pela construção de uma ontologia descrita em OWL através do *Protégé* (HORRIDGE et al., 2012), ferramenta desenvolvida pela Universidade de *Stanford* para edição visualização e análise de ontologias.

Noy, McGuinness et al. (2001) sugere como método para a concepção de uma ontologia 7 passos/diretrizes: *1) "determine o domínio e o escopo da topologia"*; *2) "considere a reutilização de ontologias já existentes"*; *3) "enumere os termos importantes da ontologia"*; *4) "defina as classes e a hierarquia de classes"*; *5) "defina as propriedades das classes"*; *6) "definição das restrições das propriedades"*; e, *7) "crie os indivíduos"*. Através da aplicação destes passos o presente trabalho concebeu uma representação ontológica de serviço baseado em intenção conforme os conceitos definidos pelo IBSRM que foi denominada de *Intent-Based Services Ontology* (IBSO). Essa ontologia define a classe *Serviço* como uma subclasse de *Coisa* que é formada basicamente por propriedades do tipo *Função*, *Política* e *Restrição*. A IBSO é descrita em detalhes na seção 4.1.1.3.

3.4.4 Levantamento de Requisitos Comunicacionais

Segundo Vazquez e Simões (2016) a 'Engenharia de Requisitos' consiste no *"uso sistemático e repetitivo de técnicas para cobrir as atividades de obtenção, documentação e manutenção de um conjunto de requisitos de software que atendam aos objetivos de negócio e que sejam de qualidade"*. Este processo desempenha um passo importante no desenvolvimento de um IS, pois representa o processo pelo qual apreende-se as funcionalidades de um sistema (requisitos funcionais) e como ele deve se comportar (requisitos não-funcionais). De maneira geral, a 'Engenharia de Requisito' representa o processo inicial executado na 'Engenharia de *Software*', seja no desenvolvimento em estratégia sequencial (usualmente referido como 'cascata') ou iterativo (e.g. *Rational Unified Process* (RUP)) (VAZQUEZ; SIMÕES, 2016).

Sommerville (2011) descreve um processo genérico de 'Engenharia de Requisitos' por meio das atividades: *a)* Compreensão do domínio; *b)* Coleta de requisitos; *c)* Classificação; *d)* Resolução de Conflitos; *e)* Definição de Prioridades; e, *f)* Verificação de Requisitos. Este processo foi aplicado no contexto da comunicação em rede de maneira a levantar os requisitos das aplicações, usuários, provedores de serviços e desenvolvedores de sistemas. O resultado deste processo é um modelo conceitual expresso de maneira linguística por atos de fala declarativos (*speech act theory*) (WAND et al., 1995) que captura as funcionalidades e comportamentos esperados das redes de computadores e de telecomunicações. O 'Levantamento de Requisitos Comunicacionais' é definido como um artefato da solução proposta neste trabalho e é utilizado na concepção da representação ontológica de serviço provida pela IBSO.

No **passo *a*** foi definido como domínio a *'comunicação em rede'*, o que se relaciona diretamente aos problemas e objetivos especificados nas etapas anteriores da pesquisa. No **passo *b*** foi realizado um procedimento de coleta de requisitos com base na análise do 'estado da arte', 'bibliografia tradicional', 'entrevista com profissionais da área' e 'observação do objeto de estudo'. Nesta etapa foram definidos requisitos funcionais genéricos de maneira que as especificidades fossem definidas por requisitos não-funcionais.

Os requisitos funcionais definidos de maneira resumida são: *'comunicação unicast intra-domain'* (RF01); *'comunicação unicast inter-domain'* (RF02); *'comunicação multicast intra-domain'* (RF03); *'comunicação multicast inter-domain'* (RF04); *'comunicação broadcast'* (RF05); *'comunicação anycast intra-domain'* (RF06); e, *'comunicação anycast inter-domain'* (RF07). E os requisitos não-funcionais: *'largura de banda garantida'* (RNF01); *'largura de banda limitada'* (RNF02); *'largura de banda dinâmica'* (RNF03); *'utilização limitada por tempo'* (RNF04); *'utilização limitada por volume de dados'* (RNF05); *'utilização limitada por custo/valor'* (RNF06); *'canal livre de erros'* (RNF07); *'canal com latência mínima'* (RNF08); *'canal com baixo custo'* (RNF09); *'canal resiliente à tentativa de ataques'* (RNF10); *'canal otimizado de acordo com a aplicação'* (RNF11); *'restrição de uso por dispositivo'* (RNF12); *'restrição de uso por aplicação'* (RNF13); *'restrição de uso por tipo de conteúdo'* (RNF14); *'restrição de uso por tecnologia'* (RNF15); e, *'restrição de uso por território'* (RNF16).

No **passo *c*** os requisitos funcionais foram classificados por dois critérios: *tipo* e *abrangência*. Quanto ao *tipo* foram definidas as classes: *i)* requisitos de comunicação *unicast* (RF01 e RF02), permitem a comunicação entre pares de entidades; *ii)* requisitos de comunicação *multicast* (RF03 e RF04), permitem a comunicação em grupo de entidades; *iii)* requisitos de comunicação *broadcast* (RF05), permite a comunicação entre uma entidade e todas as demais entidades de um domínio; e, *iv)* requisitos de comunicação *anycast* (RF06 e RF07), permitem a comunicação entre uma entidade e uma partição de um grupo de entidades. Quanto à *abrangência* foram definidas as classes: *i)* requisitos de comunicação *intra-domain* (RF01, RF03, RF05 e RF06), permitem a comunicação entre entidades de

um mesmo domínio; e, *ii)* requisitos de comunicação *inter-domain* (RF02, RF04 e RF07), permitem a comunicação entre entidades de domínios distintos.

Neste mesmo passo os requisitos não-funcionais foram classificados quanto ao tipo de controle: *i)* requisitos de controle de alocação (RNF01, RNF02 e RNF03), permitem o controle sobre a largura de banda e a forma de alocação; *ii)* requisitos de controle de uso (RNF04, RNF05 e RNF06), permitem o controle sobre os critérios de limitação do uso; *iii)* requisitos de controle de canal (RNF07, RNF08, RNF09, RNF10 e RNF11), permitem o controle sobre as funções de tratamento do plano de dados e as características à serem priorizadas na alocação de recursos de infraestrutura para o estabelecimento do canal; e, *iv)* requisitos de controle de acesso (RNF12, RNF13, RNF14, RNF15 e RNF16), permitem o controle sobre os critérios de restrição e acesso e de escolha de recursos da infraestrutura para o estabelecimento do canal.

O **passo *d*** que envolve a resolução de conflitos entre os requisitos se mostrou necessário neste processo diante a possibilidade da combinação destes requisitos de maneira a representar cenários mais complexos. Definiu-se que os requisitos funcionais classificados como *requisitos de comunicação multicast* e *requisitos de comunicação anycast* podem ser combinados de maneira há possibilitar uma comunicação que seja simultaneamente *intra-domain* e *inter-domain*. Dentre os requisitos não comunicacionais verificou-se não haver nenhum conflito entre os requisitos classificados como *requisitos de controle de alocação*, pois pode-se encontrar cenários onde todos os requisitos dessa classe são utilizados simultaneamente, por exemplo, pode-se querer definir uma banda mínima, e máxima e um comportamento dinâmico de alocação entre estes limites.

Os requisitos não-funcionais classificados como *requisitos de controle de uso* são em tese conflitantes, pois definem estratégias diferentes para limitação do uso do canal. Os requisitos não-funcionais classificados como *requisitos de controle de acesso* não são conflitantes e em tese podem ser combinados em requisitos de acesso mais sofisticados. Pode-se por exemplo restringir dispositivos (através do MAC), aplicações (através de portas), conteúdo (a partir de amostras), tecnologia (por marca, modelo ou protocolo) e território (por regiões geográficas) utilizando os requisitos *RNF12*, *RNF13*, *RNF14*, *RNF15* e *RNF16* respectivamente. Já os requisitos não-funcionais classificados como *requisitos de controle de canal* embora sejam conflitantes acredita-se que eles possam ser combinados a partir de uma definição de prioridade. Pode-se definir por exemplo a utilização de um canal com tratamento de erros (*RNF07*), e ao mesmo tempo (com menor prioridade) com menor latência (*RNF08*) ou menor custo (*RNF09*). Pode-se também definir que a rede deva optar por um canal otimizado para uso geral através do *RNF11* ou por um canal com foco em segurança através do *RNF10*.

O **passo *e*** que consiste na priorização dos requisitos foi executado repetidas vezes durante as iterações do processo de 'Engenharia de Requisitos'. Dessa maneira, muitos requisitos definidos nas primeiras iterações foram excluídos por representarem aspectos

já cobertos por outros requisitos ou por representarem funcionalidades/comportamentos pouco relevantes ao problema modelado. Por fim, o **passo *f*** que consiste na verificação dos requisitos se deu por meio da técnica de 'estudo de caso'. A ideia adotada foi a de representar cenários de comunicação reais por meio dos requisitos levantados. Diversos 'casos de uso' foram utilizados nesta etapa. Dois exemplos são: *1) transmissão de vídeo ao vivo para múltiplos dispositivos conectados pela Internet*: *RF04* (comunicação *multicast inter-domain*), *RNF01* (alocação de banda mínima) + *RNF03* (alocação de banda dinâmica) combinados e *RNF08* (canal com latência mínima); e, *2) requisições de compra e venda entre um robô de software e um broker balanceado de sistema financeiro na Internet*: *RF07* (comunicação *anycast inter-domain*), *RNF03* (alocação de banda mínima), *RNF10* (canal seguro) + *RNF07* (canal confiável) + *RNF08* (canal com latência mínima) e *RNF12* (restrição de dispositivos) + *RNF13* (restrição de aplicação).

3.4.5 *Design* da Solução

Segundo Wand et al. (1995) o *design* transforma o modelo conceitual em um modelo mais próximo da implementação que inclui informações sobre os componentes do sistema e suas interfaces. Olivé (2007) define o conceito de 'Arquitetura Conceitual' como uma representação de um sistema em alto nível que inclui informações sobre os componentes e suas integrações. Bass, Clements e Kazman (2003) define de maneira mais ampla o conceito de 'Arquitetura de *Software*' como a estrutura do sistema que é composta de seus componentes de *software* e das relações entre eles. Assim, este trabalho utilizou de conceitos e ferramentas da 'Engenharia de Software' para o *design* de uma 'Rede Auto-Organizada' abstraída como uma 'Arquitetura Conceitual'.

3.4.6 Arquitetura de Rede Auto-Organizada

O presente trabalho propôs a *Self-Organizing Network Architecture* (SONAr), uma arquitetura para a concepção de 'Redes Auto-Organizadas' que define um conjunto de componentes de gerenciamento responsáveis pela coleta, análise e intervenção na rede. A SONAr é resultado do processo de *design* aplicado ao SONeM e IBSM. A influência do SONeM no *design* da SONAr se manifesta na aplicação da 'Camada de Gerenciamento' proposta por este modelo, inclusive considerando as mesmas definições de interfaces, tanto superiores quanto inferiores; e na proposição de componentes que atendem aos princípios funcionais definidos pelas subcamadas de gerenciamento. A influência do IBSM na SONAr está no suporte especificado nesta arquitetura a IBSs de acordo com a e definição IBSO; e incorporação dos componentes conceituais responsáveis pelo fluxo de aprovisionamento de serviços definidos pelo IBSM.

A SONAr se inspirou no conceito de *Event-driven Architecture* (EDA) (MICHELSON, 2006) para a integração dos componentes de gerenciamento de maneira a prover o princípio

de 'baixo acoplamento' (BOURQUE et al., 1999), que é um medida pela qual atesta-se a baixa relação de interdependentes entre os módulos de um sistema. Diferente de uma arquitetura tradicional que se baseia no padrão cliente/servidor, a EDA se baseia no padrão *pub/sub* e propõe uma solução centrada na manipulação de eventos. A SONAr define um *broker* de eventos (NEM), ao qual todas as entidades se conectam. Essas entidades se inscrevem em tópicos relevantes ao fluxo que elas implementam, por exemplo, uma entidade de cura pode se inscrever em tópicos de eventos de congestionamento de *links*. O NEM provê também uma abstração de RPC por meio de eventos de *callback*, o que se mostrou especialmente útil na sincronização dos fluxos de configuração entre as entidades de configuração (SCE) e de coleta de topologia (TCoE) propostos nesta tese.

Como as redes são naturalmente sistemas distribuídos, uma arquitetura de rede necessariamente precisa ponderar sobre esse aspecto, sobretudo em relação ao teorema CAP que descreve a incapacidade de prover consistência, disponibilidade e tolerância a particionamento de maneira simultânea (BREWER, 2000). Ao considerar que toda comunicação em rede é passível de falha, por motivos além do controle da rede tais como a indisponibilidade de *links* e equipamentos, o aspecto de particionamento se torna indispensável. Por isso, Kleppmann (2015) afirma que a única escolha possível em relação às propriedades do teorema CAP, está entre consistência e disponibilidade. No *design* da SONAr são definidos dois componentes distribuídos que permitem a integração das entidades de coleta, aprendizado e organização: *i)* a base de dados (NDB), que é responsável por armazenar métricas, eventos, configurações e a topologia dentre outras informações; e o barramento de serviços (NEM), que é responsável por gerir os eventos trocados pelos componentes. Após reflexão chegou-se à conclusão que para a maioria das informações e eventos de rede é mais importante o aspecto de disponibilidade que o de consistência, uma vez que as informações de rede têm uma natureza volátil, e por isso, perdem valor rapidamente. Para casos específicos, como no registro de serviços, faz-se necessária uma solução com garantias ACID, e isso requer a priorização do aspecto consistência. Assim, o NDB como especificado pela SONAr envolve pelo menos duas soluções para o armazenamento de informações: uma "CA" e outra "CP'.

A SONAr foi especificada através da *Unified Model Language* (UML) (BOOCH; RUMBAUGH; JACOBSON, 2006), uma linguagem visual que permite a especificação de sistemas através de diagramas semanticamente ricos. Os diagramas utilizados foram: *i) Diagrama de Componente*, utilizado para explicitar todos os componentes envolvidos na operação de uma rede auto-organizada com enfoque nos componentes de auto-gerenciamento; *ii) Diagrama de Atividades*, utilizado na descrição de fluxos genéricos para operações de configuração, cura, proteção e otimização; e, *iii) Diagrama de Implantação*, utilizado para evidenciação do posicionamento dos componentes na infraestrutura e definição das interfaces de integração.

3.4.7 Estudo de Caso com Operações de Configuração

'Estudo de Caso' é um método de investigação com o objetivo qualitativo de explorar, descrever ou explanar algo relacionado ao caso em estudo (CHETTY, 1996). De maneira geral, o 'Estudo de Caso' permite o aprofundamento em um aspecto específico com o objetivo de explorar um conceito em um contexto limitado. Trata-se de um método amplamente utilizado para exemplificação e/ou especificação de uma estratégia de aplicação de uma solução em um cenário específico.

O presente trabalho se dedicou a propor uma solução para a automação do gerenciamento de rede, o que se demonstrou tema muito vasto para o escopo de uma tese de doutorado. Desta maneira, foi definido o direcionamento do trabalho para a automação das operações de configuração o que acredita-se ser um ponto de partida natural para a automação do gerenciamento de rede. Considera-se a propriedade de configuração um aspecto chave, pois é a partir das operações de configuração que a rede se torna operacional e sensível a mudanças na infraestrutura e definição de serviços. Dessa maneira, não há como falar de cura, otimização ou proteção, sem que o aspecto de configuração tenha sido endereçado.

Neste contexto, foi proposto como artefato um 'Estudo de Caso com Operações de Configuração' que explora a aplicação da SONAr na automação das operações de *bootstrapping* da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços em redes SDN. Para isso, os processos foram mapeados e exemplificados através de *Business Process Model and Notation* (BPMN), uma notação simples de alto valor semântico para representação de processos de negócio (DIJKMAN; DUMAS; OUYANG, 2008). Essa notação tradicionalmente é utilizada por ISPs para mapeamento dos processos operacionais, administrativos e comerciais, e por isso foi escolhida para exploração do estudo de caso, tanto por se tratar de uma abordagem conhecida para o público alvo, como por fornecer uma especificação com maior aplicabilidade.

3.4.8 Implementação

Segundo Wand et al. (1995), a implementação tem o objetivo de transformar uma especificação de um solução conceitual em um sistema de fato. Em outras palavras, a implementação trata do desenvolvimento de elementos de *software* e/ou *hardware* que compõe a solução e podem ser utilizados de maneira prática e objetiva. Essa etapa tradicionalmente inicia por um planejamento de atividades o que tem relação direta com o método de desenvolvimento. Neste trabalho foi utilizado o SCRUM, um método ágil que permite a definição de atividades com alta granularidade de maneira incremental (SOMMERVILLE, 2011). No SCRUM as entregas são realizadas a cada 2 ou 4 semanas, o que exige um planejamento coeso das atividades no início de cada ciclo de desenvolvimento (SPRINT).

3.4.9 *Framework* de Auto-Gerenciamento

O presente trabalho definiu como artefato o *Self-Organizing Network Framework* (SONAr-Framework), um *framework* de gerenciamento que materializa a especificação SONAr através da implementação de seus conceitos mais básicos como a *Network Database* (NDB) e o *Network Event Manager* (NEM). O SONAr-Framework foi desenvolvido na linguagem de programação JAVA em sua versão 8, e se amparou no *Spring-Boot* (versão 2.1.2) um *framework JAVA* que facilita o desenvolvimento de aplicações através de bibliotecas que implementam importantes aspectos de desenvolvimento tais como injeção de dependências, exposição de *web-services* e manipulação de bases de dados.

O desenvolvimento da solução proposta nesta tese como um *framework* de 'auto-gerenciamento' seguiu uma estratégia de 'Projeto de Desenvolvimento Estruturante', que é um projeto que visa iniciar os projetos de sistemas, estruturar o código e implementar aspectos básicos para o seu desenvolvimento. A ideia de um projeto estruturante é muito utilizada em sistemas complexos, sobretudo nos desenvolvidos por equipes com muitos desenvolvedores, pois permite a implementação de aspectos básicos e comuns que viabilizam a quebra das atividades para execução paralela (atividades sem relação de dependência). Nesse processo foram realizadas as seguintes atividades: *i)* configuração do sistema de versionamento; *ii)* planejamento da estrutura projetos; *iii)* criação dos projetos e configuração das dependências; *iv)* criação de *scripts* de *build*, teste, *deployment* e execução dos componentes; *v)* instalação e configuração de sistemas auxiliares (e.g. DBMS, *broker* de eventos e controlador SDN); *vi)* implementação das bibliotecas de integração; *vii)* implementação de módulos básicos para o projeto (e.g. modelo, funções úteis); e, *viii)* criação de um arquétipo de componente.

O SONAr-Framework foi especificado como um sistema modular, assim como a própria especificação SONAr, e por isso o projeto de código-fonte do SONAr-Framework também é dividido em módulos especificados com *Maven* 3 (MILLER; VANDOME; MCBREWSTER, 2010) organizados hierarquicamente em três níveis: *nível 1* – projeto pai '*sonar*' que define as dependências, as versões e as configurações básicas dos *plugins*; *nível 2* – projetos de agrupamento de acordo com a classificação dos componentes especificados; e, *nível 3* – projetos "geradores de artefatos" responsáveis pela implementação das entidades e componentes da solução.

Os principais grupos de artefatos (unidades de *software*) definidos no projeto modular '*sonar*' são: *i)* self-organizing-entities – artefatos executáveis responsáveis pela abstração das SOEs definidas na seção 5.1.2; *ii)* self-learning-entities – artefatos executáveis responsáveis pela abstração das SLEs definidas na seção 5.1.3; *iii)* collecting-entities – artefatos executáveis responsáveis pela abstração das CoEs definidas na seção 5.1.4; *iv) core-libs* – bibliotecas com código comum compartilhado pelos demais projetos; e, *v) client-libs* – bibliotecas de integração com os componentes auxiliares (e.g. NEM e NDB), com elementos de infraestrutura e com componentes de controle.

A maioria dos projetos de módulos do SONAr-Framework geram artefatos executáveis que podem ser implantados em um servidor de aplicações (arquivo *.war*), executados diretamente através do *Spring-Boot* (arquivo *.jar*) ou utilizados para a criação de contêineres através de imagens *Docker* geradas automaticamente com o *plugin dockerfile-maven-plugin*. O *Auto-Boot Manager* (ABM) que é o componente responsável pelo aprovisionamento dos componentes de controle e gerenciamento, utiliza a abstração de contêineres o que facilita e dinamiza este processo. Os componentes são aprovisionados a partir de um *software* próprio de gestão de contêineres *Docker* implementados neste trabalho e embarcado junto ao SONAr-Framework que é chamado *Containerized Network Infrastructure Manager* (C-NIM).

Embora o SONAr-Framework por si só não automatize o gerenciamento da rede, acredita que o que ele, como um *framework* de fato, permita que novas funcionalidades e aplicações sejam implementadas por outros trabalhos. Esta solução inclusive já foi utilizada por trabalhos com foco em auto-cura e auto-configuração de *eNodeBs* conforme apresentado na seção 8.6. Mais detalhes sobre o SONAr-Framework são apresentados na seção 6.1 e o código fonte do SONAr-Framework pode ser obtido do *GitLab* da FACOM-UFU – <http://gitlab.facom.ufu.br/sonar>.

3.4.10 Plataforma de Auto-Configuração

Com base no estudo de caso apresentado na seção 5.2 que foca na automação das operações de inicialização da rede, *plug-and-play* de dispositivo e aprovisionamento de serviços; e na implementação do SONAr-Framework o presente trabalho especificou como um artefato de solução a *Self-Configuration and Management Platform* (SeCoMP), uma plataforma de 'auto-configuração' de redes definidas por *software*. O processo de desenvolvimento desta plataforma ocorreu de maneira iterativa de acordo com o avanço dos experimentos de maneira a corrigir problemas de *design* e de desenvolvimento e comparar diferentes as estratégias detalhadas no Capítulo 7. Considerando o escopo de experimentação definidos pelos cenários propostos na seção 7.2 alguns módulos tiveram algumas de suas funcionalidades simplificadas, como por exemplo, a escolha de melhores rotas realizada pela SCE utilizou um algoritmo de menor caminho baseado no número de saltos.

Nesta abordagem a extensão de 'auto-configuração' implementada na SeCoMP foi incorporada SONAr-Framework, de maneira a viabilizar a implementação de outras propriedades que dependam do estabelecimento do estado operacional da rede (e.g. auto-cura). Considera-se que a solução de auto-gerenciamento provida pelo SONAr-Framework em um contexto mais amplo ainda esteja em desenvolvimento, sendo que na implementação atual essa solução foca apenas nos aspectos inerentes à auto-configuração (por causa da incorporação da SeCoMP). Mais detalhes sobre a SeCoMP são apresentados na seção 6.2 e o código fonte já incluso no SONAr-Framework pode ser obtido do *GitLab* da FACOM-UFU – <http://gitlab.facom.ufu.br/sonar>.

3.4.11 Artefatos da Solução

A realização da etapa de *design e desenvolvimento* conforme proposto pela DSRM (PEFFERS et al., 2007) resultou em um conjunto de artefatos que tratam diferentes aspectos da *'automação das operações de configuração em redes auto-organizadas'*. Estes artefatos são listados na Figura 6 com múltiplos níveis de abstração de acordo com a abordagem descrita por Wand et al. (1995) que aplica a modelagem conceitual ao processo de desenvolvimento de ISs. A Figura 6 descreve relações de materialização/herança e de composição entre os artefatos por meio da notação de 'Diagrama de Classes' especificados no UML (BOOCH; RUMBAUGH; JACOBSON, 2006).

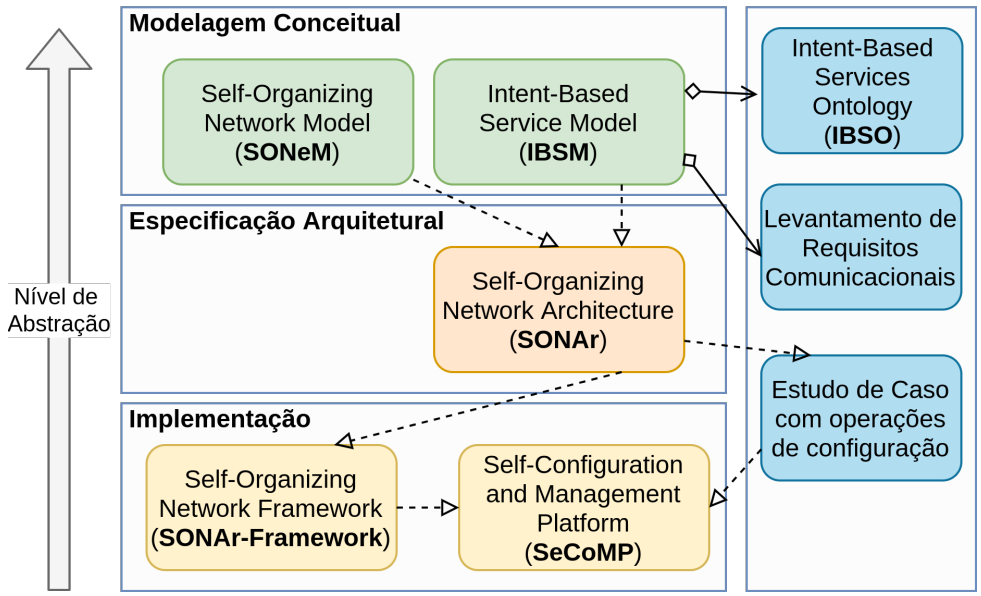


Figura 6 – Visão geral dos artefatos da solução proposta neste trabalho.

No nível de 'Modelagem Conceitual' o presente trabalho definiu dois artefatos principais: *1) Self-Organizing Network Model* (SONeM) – utilizado com o objetivo de abstrair o conceito de rede auto-organizada, especificar uma camada de gerenciamento e propor um fluxo básico de gerenciamento autônomo; e, *2) Intent-Based Service Model* (IBSM) – utilizado para a abstração de serviços de comunicação a partir de intenções, sejam elas declaradas ou inferidas, e especificação de métodos de representação e aprovisionamento para estes serviços. Para especificação do IBSM verificou-se a necessidade de uma definição formal do conceito de serviço orientado por intenções, o que foi provido pelo *Intent-Based Services Ontology* (IBSO), uma representação ontológica em OWL de *Intent-Based Services* (IBSs). Essa especificação abstrai serviços de comunicação por meio da definição de uma função e de um conjunto políticas que estão relacionadas aos requisitos funcionais e não-funcionais da comunicação. Dessa maneira, o presente trabalho também gerou como artefato um *'Levantamento de Requisitos Comunicacionais'* a partir do processo de Engenharia de Requisitos. Este levantamento se baseou em uma análise sobre a forma e o motivo pelo qual a rede é utilizada na perspectiva de usuários, aplicações e ISP. Deve-se

destacar que o 'Levantamento de Requisitos Comunicacionais' representa um conjunto de requisitos gerais iniciais de maneira a permitir o desenvolvimento do IBSM e da IBSO, mas que pode (e deve) ser modificado de acordo com as especificidades de cada rede.

No nível de 'Especificação Arquitetural' o presente trabalho definiu como artefato a *Self-Organizing Network Architecture* (SONAr), uma arquitetura que abstrai redes auto-organizadas e automatiza o gerenciamento de rede a partir de entidades responsáveis pela coleta de dados (CoEs); aprendizado e análise (SLEs); e, tomada de decisão, intervenção e organização da rede (SOEs). A SONAr materializa conceitos definidos pelo SONeM como a concepção de uma 'Camada de Gerenciamento', e pelo IBSM como o suporte à IBSs. Ainda no nível arquitetural, o presente trabalho concebeu como artefato um '*Estudo de Caso com Operações de Configuração*' através da aplicação conceitual da SONAr.

Por fim, no nível de 'Implementação' o presente trabalho definiu como artefato o *Self-Organizing Network Framework* (SONAr-Framework), um *framework* de gerenciamento de rede que abstrai os componentes SONAr por meio de microsserviços conteinerizados; e a *Self-Configuration and Management Platform* (SeCoMP), uma plataforma construída com base no SONAr-Framework que implementa os fluxos de trabalho propostos pelo '*Estudo de Caso com Operações de Configuração*' em redes SDN.

3.5 Etapa 4: Demonstração

Segundo Peffers et al. (2007) é na etapa de demonstração que é verificada a eficácia dos artefatos na resolução dos problemas de pesquisa. Este processo pode envolver diferentes estratégias, tais como: experimentação, simulação, prova conceitual e estudo de caso. Jr, Chen e Purdin (1990) propõe o uso da estratégia de experimentação na demonstração da solução proposta por uma pesquisa de sistemas de informação. Segundo o autor, é por meio da experimentação que é feita uma relação entre a teoria e prática o que pode se basear em duas perspectivas: "olhando para trás", de maneira a demonstrar o percurso da pesquisa com a visão do pesquisador; ou "olhando para frente", com o objetivo de resolver os problemas relacionados à aceitação da solução e transferência tecnológica com a visão dos possíveis usuários. Na visão de (HEVNER; CHATTERJEE, 2010), a demonstração por experimentação é uma das estratégias de avaliação possíveis para a validação do *design* da solução e pode se basear em experimentos controlados ou simulações.

A demonstração da solução proposta neste trabalho se baseou na experimentação por meio de emulação com o artefato *Self-Configuration and Management Platform* (SeCoMP), que aplica em certa medida todos os outros artefatos desenvolvidos no curso desta pesquisa. A SeCoMP implementa os principais fluxos de configuração no contexto das redes SDN, o que permite a resolução dos problemas de pesquisa e comprovação da hipótese levantada neste trabalho. Os cenários definidos foram baseados na realidade dos provedores de acesso, e os dados colhidos com os experimentos visaram a avaliação

objetiva da solução a partir de uma análise de desempenho baseado em tempo.

A emulação de rede como método de experimentação é uma estratégia importante para as soluções de rede, haja vista a dificuldade de experimentação em redes de grandes proporções. A diferença entre simulação e emulação está na profundidade da abstração: enquanto a simulação representa de maneira artificial um comportamento específico, a emulação reconstrói a solução de maneira completa em menor escala. Assim, uma rede emulada utiliza imagens de *switches*/roteadores reais, e aplica técnicas do próprio *kernel* para a criação de interfaces virtuais de rede. A emulação utilizada nos experimentos foi feita via GNS3 (NEUMANN, 2014) se baseou em topologias com dimensões entre 1 e 128 dispositivos criados como contêineres *Docker* (MERKEL, 2014) customizados com uma imagem de *Open vSwitch* (PFAFF et al., 2015).

3.5.1 Definição dos Cenários de Experimentação

Para este trabalho foram levantados cenários de experimentação comuns para provedores de acesso relacionados às operações de auto-configuração e tratados com uma abordagem SDN/*OpenFlow*. Esses cenários são descritos abaixo:

- ❏ *Inicialização de uma Rede SDN/OpenFlow*: iniciar recursos de infraestrutura e componentes de controle em uma rede com suporte a tecnologias *OpenFlow*;
- ❏ *Plug-and-Play de switches OpenFlow*: detectar e configurar novos *switches OpenFlow* automaticamente; e
- ❏ *Aprovisionamento de serviço para comunicação de VoIP*: configurar um serviço com QoS para garantia de banda e priorização de tráfego de uma comunicação VoIP.

3.5.2 Planejamento dos Experimentos

A etapa de planejamento de experimentos é fundamental para avaliar a qualidade e a relevância dos resultados, e consequentemente da pesquisa. Os experimentos precisam ser descritos de maneira clara com o objetivo de guiar a execução da experimentação com padrões de controle e qualidade. Estes experimentos são descritos nesta seção por uma sequências de passos que é detalhada no Capítulo 7. Abaixo são enumerados os passos de cada experimento de maneira simplificada:

Passo-a-passo da '*Inicialização de uma Rede SDN/OpenFlow*' com a SeCoMP

- ❏ *Passo 1* – Criar a topologia de rede através da ferramenta de emulação;
- ❏ *Passo 2* – Iniciar componente de auto-inicialização;
- ❏ *Passo 3* – Executar uma aplicação de coleta de resultados;

- ❏ *Passo 4* – Iniciar todos os elementos de rede da topologia;
- ❏ *Passo 5* – Aguardar a execução da inicialização da rede e coletar os resultados.

Passo-a-passo do '*Plug-and-Play de switches OpenFlow*' com a SeCoMP

- ❏ *Passo 1* – Criar a topologia de rede através da ferramenta de emulação;
- ❏ *Passo 2* – Iniciar o componente de auto-inicialização;
- ❏ *Passo 3* – Iniciar todos os elementos de rede da topologia;
- ❏ *Passo 4* – Executar uma aplicação de coleta de resultados;
- ❏ *Passo 5* – Aprovisionar um novo *switch* na topologia;
- ❏ *Passo 6* – Aguardar a execução do *Plug-and-Play* e coletar os resultados.

Passo-a-passo do '*Aprovisionamento de serviço para comunicação de VoIP*' com a SeCoMP

- ❏ *Passo 1* – Criar a topologia de rede através da ferramenta de emulação;
- ❏ *Passo 2* – Iniciar o componente de auto-inicialização;
- ❏ *Passo 3* – Iniciar todos os elementos de rede da topologia;
- ❏ *Passo 4* – Aprovisionar quatro *hosts* e ligá-los à topologia;
- ❏ *Passo 5* (exclusivo para a abordagem SONAr): Aprovisionar um serviço de comunicação via barramento utilizando os IPs dos *Hosts* 1 e 3, e com política de garantia de banda de 64Kbps;
- ❏ *Passo 6* – Executar uma aplicação receptora para a comunicação de VoIP no `host3` e uma aplicação emissora no `host1` com taxa de 64Kbps por 50 segundos;
- ❏ *Passo 7* (simultaneamente ao *Passo 5*) – Aguardar 20 segundos e executar uma aplicação receptora de tráfego do tipo *Torrent* no `host4` e uma emissora no `host2` com taxa de 1Gbps por 30 segundos;
- ❏ *Passo 8* – Coletar os dados nas aplicações receptoras hospedadas nos *Hosts* 3 e 4.

3.5.3 Informações coletadas nos experimentos

Diferentes técnicas e parâmetros foram testados em cada experimento, e para cada teste foram coletados resultados específicos. Para isso foram implementadas aplicações específicas capazes de monitorar tempos de execução e uso do enlace. Abaixo uma breve descrição sobre os dados coletados em cada experimento:

Informações coletadas com o experimento de '*Inicialização de uma Rede SDN/OpenFlow*'

- ❏ *Tempo de Descoberta*: parâmetro que registra o tempo gasto para descoberta e validação de recursos da infraestrutura;

- ❏ *Tempo de Roteamento*: parâmetro que registra o tempo gasto no cálculo do melhor caminho para o estabelecimento dos fluxos de controle e gerenciamento;
- ❏ *Tempo de Configuração*: parâmetro que registra o tempo gasto para configuração dos recursos de infraestrutura e componentes de controle; e
- ❏ *Tempo de Processamento/Atraso*: parâmetro que registra a diferença entre o tempo de inicialização e a finalização do processo excluindo os tempos de Descoberta, Roteamento e Configuração.

Informações coletadas com o experimento de '*Plug-and-Play de switches OpenFlow*'

- ❏ *Tempo de Inicialização do Switch*: parâmetro que registra o tempo gasto para o *boot* e execução do *script* de inicialização do *switch*;
- ❏ *Tempo de Atraso/Processamento*: parâmetro que registra o tempo gasto acumulado para a comunicação entre os componentes, a reação a eventos e a espera para sincronização;
- ❏ *Tempo de Cálculo de Rota*: parâmetro que registra o tempo gasto no cálculo do melhor caminho para acessar o *switch* conectado;
- ❏ *Tempo de Configuração de Rota*: parâmetro que registra o tempo gasto para aplicar as novas regras de fluxo na topologia;
- ❏ *Tempo de Descoberta*: parâmetro que registra o tempo gasto para descobrir e validar o novo *switch*; e
- ❏ *Tempo de Configuração*: parâmetro que registra o tempo gasto para configurar o novo *switch*.

Informações coletadas com o experimento de '*Aprovisionamento de serviço para comunicação de VoIP*'

- ❏ *Taxa de Recepção VoIP*: parâmetro que registra o *bitrate* percebido na recepção de tráfego do tipo *Voice over IP* (VoIP); e
- ❏ *Taxa de Recepção Torrent*: parâmetro que registra o *bitrate* percebido na recepção de tráfego do tipo *Torrent*.

3.6 Etapa 5: Avaliação

Segundo (PEFFERS et al., 2007) a avaliação é uma maneira de observar e mensurar o quão bem os artefatos suportam a solução do problema de pesquisa. Dessa maneira, a avaliação consiste em comparar os objetivos da solução em relação aos resultados observados com a análise da demonstração dos artefatos. Pode-se avaliar os artefatos e identificar ganhos sob duas perspectivas: *i)* ganhos qualitativos, se referem à melhorias

que não podem ser mensuradas, tais como percepção do usuário em relação aos serviços de comunicação em rede; e, *ii)* ganhos quantitativos, se referem à melhorias objetivas que podem ser expressas em valores, por exemplo o ganho de tempo da abordagem de *plug-and-play* proposta nesta tese em relação à abordagem tradicional.

Deve-se atentar para fato de que a análise dos resultados experimentais não é a única maneira de avaliação de um artefato. O Apêndice B na seção B.6 apresenta possibilidades de métodos de avaliação segundo a visão de Hevner e Chatterjee (2010) que foram utilizados nesta etapa. O presente trabalho optou por avaliar a solução proposta, bem como os artefatos, através de diferentes métodos, tais como estudo de caso, simulação, testes funcionais e estruturais, análise arquitetural e análise argumentativa. A presente seção detalha o processo para a aplicação destes métodos.

3.6.1 'Estudo de Caso' com a propriedade de Auto-Configuração

Após a especificação da SONAr o presente trabalho realizou um estudo de caso com o objetivo de aplicar os conceitos desta arquitetura em um contexto prático relacionado ao tema central da tese: a *'automação das operações de configuração em redes auto-organizadas'*. Para isso, os componentes da SONAr foram explorados de maneira conceitual na automação das operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços de comunicação. O resultado deste estudo foi planejado como um artefato da solução e foi utilizado como base da implementação da SeCoMP.

Com a realização do estudo de caso verificou-se a aplicabilidade da SONAr, e por consequência do IBSM e SONeM. Esse estudo demonstrou de maneira objetiva através de fluxos mapeados como processos administrativos com a notação BPMN (DIJKMAN; DUMAS; OUYANG, 2008), que é possível implementar de maneira automática as principais operações de configuração ao aplicar os conceitos propostos pela SONAr. Mais detalhes sobre o estudo de caso são descritos na seção 5.2.

3.6.2 'Simulação' de cenários de configuração automatizados

Este trabalho optou por simular cenários reais de configuração através de uma rede emulada com GNS3 (NEUMANN, 2014), na qual foram implantados os componentes de gerenciamento implementados pelo SONAr-Framework/SeCoMP como microsserviços conteinerizados. Essa simulação foi a estratégia adotada para a demonstração do trabalho que é descrita na seção 3.5. Os detalhes sobre os experimentos conduzidos são apresentados no Capítulo 7.

Os dados colhidos durante os experimentos foram armazenados em planilhas eletrônicas e serviram como fonte para a elaboração de gráficos que indicam: *i)* tempo gasto em cada etapa do processo de bootstrapping/*plug-and-play*; e, *ii* largura de banda utilizada

pelos serviços ativos. Além disso foi utilizado o método *ex-post-facto* para a interpretação dos resultados, refinamento do SONAr-Framework/SeCoMP e conclusões da pesquisa.

Ganhos Qualitativos

Os resultados obtidos com a execução dos experimentos demonstram a viabilidade da SeCoMP e a sua efetividade na melhoria na experiência do usuário (QoE), redução de custo operacional e mitigação de riscos. Além disso, os resultados demonstram que é possível descrever serviços por intenções e alcançar comportamentos práticos de rede. Por isso, considera-se também como ganho qualitativo, a flexibilidade e facilidade na utilização da rede e definição de serviços de comunicação.

Ganhos Quantitativos

Os resultados coletados demonstram que o tempo gasto para inicialização da rede e *plug-and-play* de recursos é muito inferior que o dispendido normalmente com intervenções manuais. Estes tempos usados como referência são estimados através de dados reais de uma empresa provedora de acesso e entrevista com especialistas. Além disso os dados demonstram que a QoS oferecida pela SONAr foi capaz de garantir a largura de banda necessária para comunicações VoIP durante todo o experimento.

3.6.3 'Testes Funcionais e Estruturais' com os componentes do *framework* de auto-gerenciamento e da plataforma de auto-configuração

O processo de desenvolvimento do *framework* de auto-gerenciamento, i.e. SONAr-Framework, e da plataforma de auto-configuração, i.e. SeCoMP, se guiou por boas práticas de desenvolvimento de *software*, dentre elas o *Test-driven Development* (TDD) (BECK, 2003), que propõe o desenvolvimento das funcionalidades de maneira incremental e guiados pela especificação de testes unitários. Beck (2003) descreve de maneira simplificada o TDD como "*uma maneira de gerir o medo durante a programação*", pois a partir dessa técnica adquire-se mais confiança de que o código atende aos cenários especificados. O TDD também gera confiança nos desenvolvedores pois ele diminui o risco de alterações por estimular uma cobertura de testes mais completa.

Assim, durante todo o curso de desenvolvimento da solução apresentada nesta tese pôde-se atestar por meio de testes unitários estruturais o funcionamento das funções implementadas pelos componentes SONAr. Além destes, foram realizados testes funcionais de maneira a garantir que os módulos do SONAr-Framework e da SeCoMP funcionassem como planejado diante estímulos específicos (e.g. interceptação de mensagens DHCP) e que as interfaces definidas por estes componentes através do padrão *pub/sub* e RPC

com evento de *callback* estivessem configuradas de maneira correta. Por fim, foram realizados testes de integração de maneira manual, com o objetivo de comprovar que a solução funcionasse como um todo. Nestes testes foram simulados cenários relacionados aos experimentos conduzidos e descritos no Capítulo 7.

3.6.4 'Análise Arquitetural' dos componentes do *framework* de auto-gerenciamento e da plataforma de auto-configuração

Na implementação do SONAr-Framework e da SeCoMP o presente trabalho se dedicou a uma análise arquitetural no sentido de validar a implementação dos componentes em relação à especificação SONAr. Para isso, foram avaliados os fluxos implementados pelos componentes de coleta de dados de topologia (TCoE) e de configuração (SCE), onde pôde-se verificar que não houve variação em relação aos fluxos genéricos propostos na especificação SONAr realizada na seção 5.1.7.

Além disso, na proposição da SONAr realizou-se uma análise sobre como os componentes SONAr se encaixariam dentro dos principais padrões, modelos e *frameworks* de gerenciamento enumerados na seção 2.1.2: FCAPS (ISO/IEC, 1989), OAM&P (KU, 1998) e FAB (TMF, 1998). O resultado dessa análise é descrito na seção 5.1.6 onde é demonstrado que o conjunto de componentes SONAr atende aos princípios definidos por estes padrões de maneira completa.

3.6.5 'Análise Argumentativa' sobre a aplicação da solução na resolução do problema

A análise argumentativa faz parte do percurso natural do processo de pesquisa e publicação. Desta maneira, realizou-se uma análise argumentativa sobre como cada um dos artefatos propostos pela solução pode ser utilizado na *'automação das operações de configuração em redes auto-organizadas'*. Neste processo procurou-se amparar a análise na comparação com outras abordagens, ou mesmo em relação à ausência de abordagem, o que é característico do gerenciamento de redes de computadores.

O resultado da análise argumentativa sobre a efetividade dos artefatos é apresentado de maneira mais clara nas seções 8.1 e 8.2. Realizou-se uma argumentação similar nos trabalhos publicados e submetidos listados na seção 8.6. Os argumentos utilizados nestes trabalhos foram avaliados por bancas de publicações conceituadas e considerados aceitáveis para a publicação.

3.7 Etapa 6: Comunicação

Um dos princípios básicos na realização de uma pesquisa, talvez tão importante quanto o desenvolvimento da pesquisa em si, reside em sua comunicação. Hevner e Chatterjee (2010) descreve em sua *Guideline 7* que uma DSR deve ser apresentável tanto para audiências técnicas quanto gerenciais. Para as audiências técnicas deve-se prover detalhes suficientes para permitir que os artefatos propostos pela solução sejam recriados. Já audiências gerenciais precisam de informações em alto-nível suficientes para o entendimento da solução e para a identificação dos recursos necessários para a sua construção ou implantação no ambiente da organização.

Peffers et al. (2007) define como objetivos da comunicação da pesquisa a descrição do problema e de sua importância, a proposição dos artefatos, a discussão sobre a sua utilidade e ineditismo, o rigor metodológico do *design* e a efetividade da solução. Trata-se de uma estrutura similar à empregada na maioria dos trabalhos acadêmicos, incluindo esta tese.

Neste trabalho a comunicação foi realizada com foco apenas em audiências técnicas por meio da escrita e apresentação de artigos e desta tese.

3.7.1 Escrita de Artigos sobre auto-gerenciamento

A escrita de artigos científicos é um método bem difundido para estruturação da pesquisa por fornecer oportunidades de crítica por pares, cooperação com outros pesquisadores e por servir como um 'entregável' para o projeto. Este método força a formalização da pesquisa e ajuda a torná-la mais relevante.

Este trabalho gerou um artigo publicado na edição de 2020 do *International Conference on Cloud Computing and Services Science* com o título de *'Bootstrapping and Plug-and-Play Operations on Software Defined Networks: A Case Study on Self-Configuration Using the SONAr Architecture'*. Este artigo explora os conceitos apresentados neste tese de maneira preliminar e apresenta resultados de inicialização e *plug-and-play* em cenários de rede simplificados com topologias livre de *loops*. Este trabalho foi indicado ao *Best Paper Award* e foi convidado para publicação de uma versão estendida no *Communications in Computer and Information Science*.

A pesquisa apresentada nesta tese em seu estágio final foi condensada em um artigo submetido na edição 20201 do *journal* da IEEE *Transactions on Network and Service Management* com o título de *'Automating Configuration Operations on SDN Networks through the SONAr Management Platform'*. Diferentemente da publicação no CLOSER 2020, o artigo submetido ao TNSM representa uma versão mais madura da solução que contempla cenários mais complexos de experimentação (sem restrição de *loop*) e experimento com aprovisionamento de serviços com QoS de VoIP.

Pode-se afirmar que o presente trabalho influenciou de maneira direta ou indireta outros trabalhos do grupo de pesquisa que contemplam aspectos como auto-cura do plano de controle e auto-configuração de *eNodeBs*. Dentre os artigos de colaboração (sem primeira autoria) publicados ou submetidos estão: *1) 'A Self-healing Platform for the Control and Management Planes Communication in Softwarized and Virtualized Networks'* – CLOSER 2020; *2) 'Self-healing in the Scope of Software-basedComputer and Mobile Networks'* – CCIS 2021; *3) 'Network Self-configuration for Edge Elements using Self-Organizing Networks Architecture (SONAr)'* – CLOSER 2021; e, *4) 'Management Slices: Enabling Self-management for SDN and NFV Communication Layers'* – COMMAG 2021.

3.7.2 Escrita da Tese de Doutorado

A escrita da tese de doutorado consiste em um trabalho formal para a comprovação da validade da pesquisa que, assim como na escrita do artigo, ajuda a organizar e estruturar a pesquisa. Além disso, a tese permite a exposição da pesquisa para uma banca de pesquisadores com notório saber que contribuem para o avanço e melhoria da qualidade do trabalho.

A presente tese seguiu o modelo aprovado pelo colegiado do Programa de Pós-Graduação em Ciência da Computação (PPGCO) da Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia (UFU). O método utilizado nesta pesquisa se baseou na DSRM (PEFFERS et al., 2007) desenvolvida com a estratégia de modelagem conceitual aplicados ao *design* e desenvolvimento de soluções de IT apresentando por (WAND et al., 1995), o que se manifesta de maneira clara na divisão da solução proposta nesta tese em três capítulos: Capítulo 4, que apresenta a solução através de modelos conceituais; Capítulo 5, que apresenta a solução através de uma arquitetura; e, Capítulo 6, que apresenta a solução através de um *framework* de auto-gerenciamento e de uma plataforma de auto-configuração. Os demais capítulos seguem uma estrutura tradicional de trabalhos científicos.

Capítulo 4

Proposição dos Modelos Conceituais *Intent-Based Service Model* (IBSM) e *Self-Organizing Network Model* (SONeM)

Este capítulo almeja apresentar uma solução para concepção de Redes Auto-Organizadas com alto nível de abstração através de modelos conceituais. Considerando as perspectivas para o gerenciamento das redes futuras apresentadas na seção 2.1.4 e os requisitos de auto-gerenciamento da indústria e das aplicações de rede levantados na seção 2.2.4 acredita-se que tal solução requer: *i)* definição de mecanismos para a abstração em alto nível de serviço de comunicação que sejam flexíveis às necessidades das aplicações atuais e futuras; e, *ii)* criação de estratégias para gerenciamento de recursos de infraestrutura e serviços de comunicação de forma autônoma. Para resolver *i* é apresentado na seção 4.1 o *Intent-Based Service Model* (IBSM) que propõe um modelo para representação e aprovisionamento de serviços de comunicação definidos por intenções descritas ou inferidas. Para *ii* é apresentado na seção 4.2 o *Self-Organizing Network Model* (SONeM), que propõe um modelo de organização da rede em camadas com funções de monitoramento, análise e intervenção. A principal contribuição do SONeM está na especificação de uma camada gerenciamento que pode ser aplicada tanto em redes tradicionais [1] quanto futuras [2] e que considera mecanismos para a coleta/análise de dados e configuração da infraestrutura. Os modelos propostos neste capítulo são utilizados para a concepção de uma Arquitetura de Conceituação e Levantamento de Redes Auto-Organizadas que é apresentada em detalhes no capítulo 5. A principal diferença entre a SONeM e esta arquitetura está no nível de abstração: enquanto a SONeM define camadas e fluxos, a arquitetura proposta

[1] Redes Tradicionais: e.g. redes de computadores TCP/IP
[2] Redes Futuras: e.g. redes SDN

no capítulo 5 define componentes de gerenciamento e suas integrações.

4.1 IBSM: Modelo de Serviços Baseado em Intenções

O *Intent-Based Service Model* (IBSM) é um modelo de representação e aprovisionamento de serviços de comunicação baseados em intenções definidas por análise comportamental, representação formal ou descrição em linguagem natural. O uso de intenções na definição desses serviços permite definir comportamentos de rede e características do canal de comunicação em alto nível, ou seja, sem detalhes técnicos, tais como: parâmetros de configuração, definição de rotas e alocação de recursos.

O conceito de *Serviço*, de maneira geral, está ligado à ideia de desempenhar uma atividade ou suprir uma demanda. O serviço de comunicação é responsável pelo gerenciamento da comunicação em conformidade com os requisitos comunicacionais das aplicações. No contexto das Rede Auto-Organizadas esses serviços são providos pela própria rede que também é responsável por identificá-los (de forma passiva ou ativa) e por traduzi-los em soluções práticas de acordo com as características da infraestrutura. A representação de serviços proposta pelo IBSM se baseia na percepção sobre as intenções por trás das comunicações, e são provenientes das seguintes indagações:

- Qual o tipo de comunicação será realizado e qual a motivação da comunicação?
  Exemplo: *multicast* para *streaming* de vídeo dentro de um domínio;
- Como a comunicação deve ser tratada e quais os requisitos da comunicação devem ser contemplados?
  Exemplo: com criptografia ponta a ponta e priorizando um canal com baixa latência;
- A qual comunicação o serviço se aplica e em qual contexto ele deve ser executado?
  Exemplo: comunicações provenientes de um IP específico nos finais de semana ou depois das 18 horas.

O IBSM define um Modelo de Representação de Serviços Baseados em Intenções (IBSRM) que é descrito na seção 4.1.1, e um Modelo de Aprovisionamento de Serviços Baseados em Intenções (IBSPM) que é apresentado na seção 4.1.2. O IBSRM conceitua os Serviços Baseados em Intenções (IBSs) e define mecanismos de representação com base na interpretação de Intenções Declarativas (*Declarative Intents*) e análise de comportamentos de rede. O IBSPM propõe componentes conceituais responsáveis pela definição, interpretação, configuração e validação de serviços na rede.

4.1.1 IBSRM - Modelo de Representação de Serviços Baseados em Intenções

O *Intent-Based Service Representation Model* (IBSRM) é um modelo de representação de serviços com base em intenções detectadas ou declaradas. Ele se baseia no Modelo Declarativo que consiste na definição de um objetivo sem se ater aos mecanismos para alcançá-lo. Em outras palavras, o Modelo Declarativo se preocupa com "o quê" e não com o "como". Serviços com base nesse modelo são representações de alto nível que capturam as intenções dos usuários e administradores. A definição dos fluxos e mecanismos para o aprovisionamento de serviços fica a cargo da própria rede.

4.1.1.1 Conceituação de *Intent-Based Services* (IBSs)

Serviços de Comunicação representam as maneiras como a rede pode ser utilizada para alcançar um determinado objetivo, seja assistir um *Video on Demand* (VoD) pelo celular ou controlar um braço robótico em uma cirurgia a distância. Os serviços de comunicação das redes atuais são pouco flexíveis e requerem a utilização de protocolos na camada de aplicação que supram deficiências das camadas de rede e transporte como é o caso do protocolo RTP que adiciona um controle mínimo de fluxo ao UDP úteis para o streaming de áudio e vídeo. Na arquitetura TCP/IP a rede oferece apenas um serviço de transferência de pacotes *straight-forward* que permite a escolha entre um canal simples sem nenhum controle de fluxo (UDP) e um canal com controle de fluxo e entrega garantida (TCP). Acredita-se que os serviços de comunicação oferecidos pela rede possam ser mais flexíveis através da escolha adequada de equipamentos/tecnologias, técnicas de comutação/roteamento e políticas de tratamento/priorização. Além disso, a rede pode implantar funções de rede de maneira a tratar o plano de dados de acordo com a necessidade da comunicação de maneira transparente para as aplicações.

Os *Intent-Based Services* (IBSs) são serviços de comunicação flexíveis que representam as intenções da comunicação sem detalhes tecnológicos sobre como os requisitos dessa comunicação devem ser supridos, o que torna a definição de serviço de comunicação mais genérica, pois quem decide como atender à necessidade de comunicação é a própria rede e não os usuários/aplicações. Isso quer dizer que um mesmo serviço pode ser implantado em diferentes tipos de rede, com diferentes tecnologias e protocolos sem que os sistemas finais tenham consciência disso. O conceito de IBS proposto nesta tese se baseia no Modelo Declarativo em contraposição ao Modelo Imperativo tradicionalmente utilizado na definição dos serviços de comunicação.

IBS é um conceito abstrato que pode tanto ser utilizado para descrever capacidade (e.g. largura de banda, *throughput* e volume de dados); comportamentos (e.g. correção de erros, baixa latência, baixo custo, livre de ataques); restrições de segurança (e.g. restrição de equipamentos, tecnologias ou localizações); limites de utilização (e.g. limitação por

tempo, uso ou custo); ou mesmo, tipo de comunicação (e.g. *unicast*, *broadcast*, *anycast* ou *multicast*). O IBS também ser utilizado para definir contextos de utilização, sejam eles referentes ao estado da rede, tipo de conteúdo, volume de tráfego, momento (e.g. data, hora, período e etc) ou detalhes da informação (e.g. origem, destino, protocolos e etc). Dessa maneira, o conceito de IBS é bastante flexível, e pode ser utilizado com diferentes níveis de detalhes, desde para definir uma comunicação genérica até para definir uma comunicação de um conteúdo e origem/destino específicos, em um determinado momento/frequência, com restrições de utilização e com requisitos de confiabilidade, largura de banda e latência.

Embora o conceito de IBS tenha se baseado no Modelo Declarativo, ele não se restringe à definição prévia, seja ela formal ou informal, do serviço, mas também à capacidade da rede de inferir as necessidades comunicacionais dos sistemas interconectados. Isso quer dizer que a rede pode atuar de maneira ativa inspecionando o plano de dados e definindo políticas de maneira a melhorar a *Quality of Experience* (QoE) mesmo sem uma definição prévia de serviço. A definição de IBSs inferidos torna a rede ainda mais flexível às necessidades das aplicações, uma vez que não exige nenhum detalhe (mesmo em alto nível) para viabilizar a comunicação com requisitos e restrições adequados. Na prática, a ideia de IBS inferido se baseia na premissa de que é possível detectar comportamentos de rede e gerar um IBS descrito passível de ser interpretado e implantado pela rede de forma autônoma.

4.1.1.2 Levantamento de Requisitos/Desafios Comunicacionais

A *Comunicação* consiste na troca de mensagens entre interlocutores com um objetivo básico e com um conjunto mínimo de requisitos. Esses requisitos definem as características do canal de transmissão em consonância com a natureza da comunicação. Para representação de IBSs faz-se necessária a correta análise e modelagem destes requisitos comunicacionais o que é feito nesta tese através de Engenharia de Requisitos, sendo o objetivo básico da comunicação abstraído como *Requisitos Funcionais* e as características do canal a ser utilizado na comunicação como *Requisitos Não-Funcionais*. Através das etapas do processo de *Engenharia de Requisitos* é possível definir um conjunto de requisitos comunicacionais básicos que podem ser combinados e/ou estendidos para a concepção de canais de comunicação mais sofisticados. Os requisitos (que também podem ser abordados como desafios) que foram levantados neste processo são listados abaixo:

Requisitos Funcionais

- **RF01** - *A rede deve possibilitar a comunicação entre um par de entidades (**unicast**) de um mesmo domínio (**intra-domain**).*

Essa funcionalidade permite a comunicação entre quaisquer pares de sistemas finais em uma rede local.

- **RF02** - *A rede deve possibilitar a comunicação entre um par de entidades (**unicast**) de domínios distintos (**inter-domain**).*
  Essa funcionalidade permite a comunicação entre quaisquer pares de sistemas finais em redes distintas.

- **RF03** - *A rede deve possibilitar a comunicação em grupo de entidades (**multicast**) de um mesmo domínio (**intra-domain**).*
  Essa funcionalidade permite a comunicação dentro de grupos de sistemas finais pertencentes a uma rede local.

- **RF04** - *A rede deve possibilitar a comunicação em grupo de entidades (**multicast**) de domínios distintos (**inter-domain**).*
  Essa funcionalidade permite a comunicação dentro de grupos de sistemas finais pertencentes a diversas redes.

- **RF05** - *A rede deve possibilitar a comunicação entre uma entidade e todas as outras entidades de um mesmo domínio (**broadcast**).*
  Essa funcionalidade permite a comunicação entre um sistema final e todos os outros sistemas finais de uma rede local.

- **RF06** - *A rede deve possibilitar a comunicação entre uma entidade e uma ou mais entidades dentre diversas de um mesmo domínio (**anycast intra-domain**).*
  Essa funcionalidade permite a comunicação entre um sistema final e um ou mais sistemas finais em um grupo de sistemas finais de uma rede local.

- **RF07** - *A rede deve possibilitar a comunicação entre uma entidade e uma ou mais entidades dentre diversas de domínios distintos (**anycast inter-domain**).*
  Essa funcionalidade permite a comunicação entre um sistema final e um ou mais sistemas finais em um grupo de sistemas finais de redes distintas.

Requisitos Não-Funcionais

- **RNF01** - *A largura de banda alocada para uma comunicação deve ser garantida.*
  Ao estabelecer uma comunicação entre sistemas finais deve ser possível definir qual a largura de banda mínima necessária.

- **RNF02** - *A largura de banda alocada para uma comunicação deve ser limitada.*
  Ao estabelecer uma comunicação entre sistemas finais deve ser possível definir o máximo de largura de banda que pode ser alocada.

- **RNF03** - *A largura de banda alocada para uma comunicação deve ser dinâmica.*
  Ao estabelecer uma comunicação entre sistemas finais a rede deve prover uma largura de banda conforme a necessidade.

- **RNF04** - *A utilização dos enlaces deve ser limitada por tempo.*
  A rede deve limitar o uso da infraestrutura utilizando como métrica o tempo acumulado de alocação do canal.

- **RNF05** - *A utilização dos enlaces deve ser limitada por dados.*
  A rede deve limitar o uso da infraestrutura utilizando como métrica a somatória da quantidade de dados trafegados por um canal.

- **RNF06** - *A utilização dos enlaces deve ser limitada por custo.*
  A rede deve limitar o uso da infraestrutura utilizando como métrica um cálculo de custo sobre a infraestrutura levando em consideração consumo energético, custo operacional, custo de investimento, tempo e percentual de ocupação.

- **RNF07** - *O canal estabelecido para a comunicação deve ser confiável (livre de erros).*
  Ao estabelecer o canal para uma comunicação a rede deve optar por enlaces com menor incidência de erro, e caso eles ocorram, a rede deve utilizar mecanismos para correção ou retransmissão das mensagens.

- **RNF08** - *O canal estabelecido para a comunicação deve ser rápido (latência mínima).*
  Ao estabelecer o canal para uma comunicação a rede deve optar por enlaces com menor latência acumulada e que levam em consideração os atrasos em decorrência de erros, ordenação e remoção de duplicatas. A rede também deve avaliar e aplicar técnicas úteis para diminuir a latência como marcação de pacotes, adequação de MTU e compactação de conteúdo.

- **RNF09** - *O canal estabelecido para a comunicação deve ser barato (baixo custo).*
  Ao estabelecer o canal para uma comunicação a rede deve optar por enlaces com menor custo energético, operacional e de infraestrutura. A rede deve também ativar/desativar elementos conforme a necessidade, e deve classificar fontes de energia por impacto ambiental.

- **RNF10** - *O canal estabelecido para a comunicação deve ser seguro (resiliente a tentativas de ataque).*
  Ao estabelecer o canal para uma comunicação a rede deve optar por enlaces melhor ranqueados quanto ao número de ataques. A rede deve ser capaz de implantar políticas e funções de rede de maneira a garantir princípios de confidencialidade, integridade e disponibilidade para a comunicação.

- **RNF11** - *O canal estabelecido para a comunicação deve ser otimizado (customizado por aplicação).*
  Ao estabelecer o canal para uma comunicação a rede deve procurar por mecanismos para otimizar a comunicação tais como compactação, codificação e de adaptação do conteúdo. Na aplicação em comunicações multimídia por exemplo, a rede deve ser capaz de identificar a necessidade do receptor e a largura de banda disponível e eventualmente recodificar o conteúdo de acordo com a necessidade.

- **RNF12** - *O canal de comunicação deve ser restrito a certos dispositivos.*
  A rede deve ser capaz de identificar e autorizar os sistemas finais de uma comunicação, e restringir os dispositivos de comutação e de tratamento que podem ser utilizados no estabelecimento do canal.

- **RNF13** - *O canal de comunicação deve ser restrito a certas aplicações.*
  A rede deve ser capaz de impedir comunicações entre aplicações desconhecidas ou maliciosas através de uma implementação similar aos filtros de pacotes convencionais.

- **RNF14** - *O canal de comunicação deve ser restrito a certos tipos de conteúdo.*
  A rede deve ser capaz de inferir qual o tipo de conteúdo está sendo transmitido em uma comunicação, e permitir (ou impedir) que a comunicação ocorra.

- **RNF15** - *O canal de comunicação deve ser restrito a certas tecnologias.*
  A rede deve estabelecer o canal de comunicação utilizando determinados dispositivos, técnicas e protocolos.

- **RNF16** - *O canal de comunicação deve ser restrito a certos territórios.*
  A rede deve estabelecer o canal de comunicação utilizando dispositivos restritos a uma determinada região ou partição de rede.

Os Requisitos Comunicacionais Funcionais e Não-Funcionais listados acima representam desafios para a comunicação em rede levantados por meio de 'pesquisa bibliográfica' e 'observação' das aplicações atuais. Os detalhes sobre o processo de Engenharia de Requisitos que envolve o 'Levantamento', 'Análise' e 'Resolução de Conflitos' são descritos na seção 3.4.4.

Além disso, os serviços propostos pelo IBSM não se restringem à lista de requisitos listados acima que por ora servem apenas como exemplos para o desenvolvimento da solução. Espera-se que o conjunto de requisitos seja flexível de maneira a representar as capacidades da rede e as necessidades das aplicações independente de tecnologia. Isso quer dizer que determinadas redes podem definir os seus próprios requisitos comunicacionais suportados e disponibilizá-los para a descrição de serviços pelas aplicações.

4.1.1.3 Definição Ontológica de um *Intent-Based Service*

A abstração de um serviço por meio da construção de uma ontologia permite a observação de suas partes e a utilização como representação formal. Dessa maneira, a presente tese trata o problema de representação de IBSs através de uma descrição em OWL, o que permite especificar classes, indivíduos, propriedades e relações através de axiomas.

A *Intent-Based Services Ontology* (IBSO) (Apêndice C) descreve IBSs através de OWL e define 'Serviço' como uma relação entre função, contexto e requisitos. Como apresentado na Figura 7 a IBSO define a classe *'Service'* que deriva da classe *'Thing'* e é composta pelas propriedades: *superService* (referência à 'zero ou um' indivíduo da classe *'Service'*) que representa um auto-relacionamento de herança; *function* (referência à um indivíduo da classe *'Function'*) que representa de maneira básica o tipo de comunicação à ser realizada de acordo com os 'Requisitos Comunicacionais Funcionais' suportados pela rede; *policy* (referência à um ou mais indivíduos da classe *'Policy'*) que representa as características do canal de transmissão e os requisitos para o tratamento da comunicação de acordo com os 'Requisitos Comunicacionais Não-Funcionais' (indivíduos da classe *'Requirement'*) suportados pela rede; *filter* (referência à um ou mais indivíduos da classe *'Filter'*) que representam os contexto de aplicação do serviço através de restrições (indivíduos da classe *'Restriction'*) combinadas de maneira lógica; *modifier* representado por uma *string* para especificar o modificador do serviço (e.g. final, concreto e abstrato); e *statement* representador por uma ou mais *strings* para declaração da finalidade do serviço em linguagem natural.

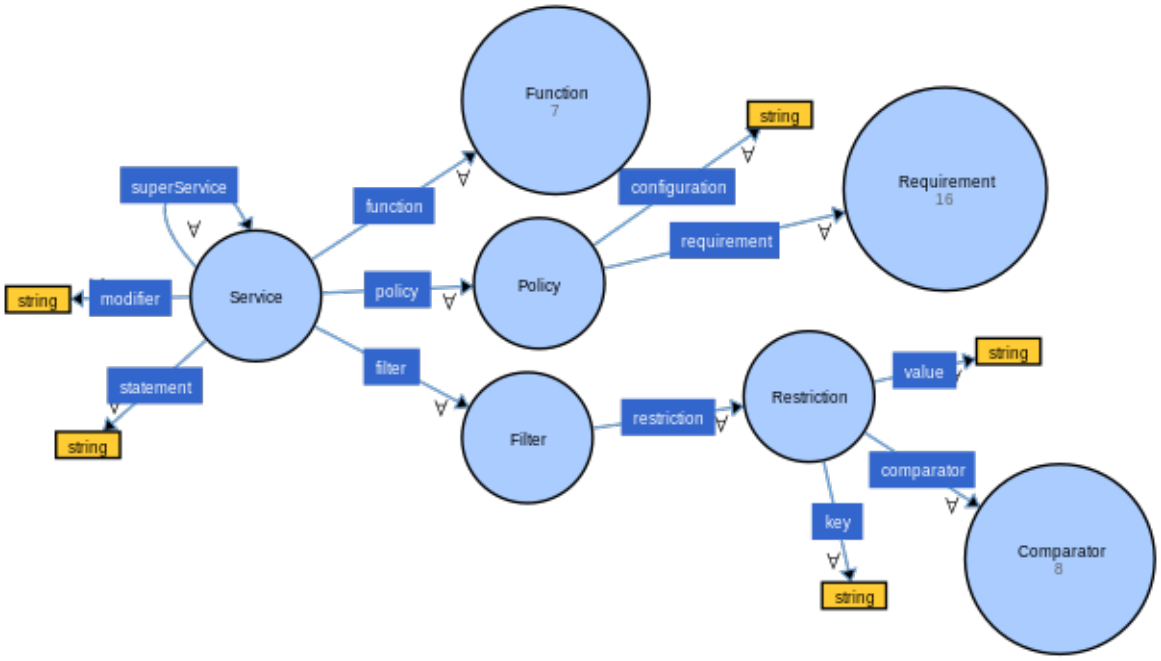


Figura 7 – Representação gráfica da *Intent-Based Services Ontology* (IBSO).

Fonte: Elaborado pelo autor (2021).

Abaixo uma breve descrição sobre as classes que compõe a IBSO:

- ***Service*** – um indivíduo da classe *Service* representa é um mecanismo para aprovisionamento de canais de comunicação com uma função bem definida, que atende a um conjunto de políticas e que se aplica em um contexto específico. O conceito de *serviço* apresentado nesta tese se baseia em uma combinação de requisitos comunicacionais, sendo os funcionais tratados como *funções* e os *não-funcionais* como *políticas*. Além disso, o serviço proposto define mecanismos de identificação de comunicação chamados de *filtros* e pode estender outros serviços de maneira a representar situações mais específicas;

- **Function** – um indivíduo da classe *Function* representa uma finalidade de uma comunicação e está relacionada à definição dos *Requisitos Comunicacionais Funcionais*. A *função* define o tipo e a motivação da comunicação, e pode ser abstraída em diferentes granularidades. Por exemplo, pode-se definir como função a comunicação *unicast* entre duas entidades de um mesmo domínio, ou, de maneira mais específica, pode-se definir como função essa mesma comunicação restrita a conteúdos multimídia. O conjunto de funções deve ser flexível para o suporte das aplicações futuras;

- ***Filter*** – um indivíduo da classe *Filter* representa um mecanismo de definição de contexto de atuação e é utilizado na identificação e dimensionamento da comunicação. Em geral um filtro é formado por uma expressão lógica combinando campos do cabeçalho (protocolos convencionais ou customizados), serviços de integração (REST, SOAP e SQL), consultas na rede (DNS, ARP e NAT), datas/períodos, tipo de conteúdo e métricas de rede. A flexibilidade de possibilidades na criação dos filtros faz com que os serviços sejam aplicados de forma dinâmica de acordo com a variação no tempo, estado da rede e conteúdo da comunicação;

- ***Policy*** – um indivíduo da classe *Policy* representa uma característica de um canal de comunicação, uma restrição ou uma estratégias de tratamento de dados de uma comunicação. Eles são reflexos diretos dos *Requisitos Comunicacionais Não-Funcionais*, e podem ser combinadas para a concepção de políticas mais complexas, e consequentemente, canais de comunicação mais específicos;

- ***Restriction*** – um indivíduo da classe *Restriction* representa uma condição de forma unitária através das propriedades *'key'*, *'comparator'* e *'value'* e que é utilizada como expressão lógica de filtro combinada com outras restrições através do operador 'E', e que por isso, tem uma relação 'muitos para um' em relação aos indivíduos da classe *'Filter'*;

- ***Requirement*** – um indivíduo da classe *Requirement* representa de forma direta os Requisitos Comunicacionais Não-Funcionais suportados pela rede no estilo de uma

estrutura de dados de enumeração e é utilizado na composição de indivíduos da classe *'Policy'* em uma relação 'um para um';

- ***Comparator*** – um indivíduo da classe *Comparator* representam uma forma de comparação entre valores (e.g. 'igual', 'diferente', 'maior que' e 'em') úteis para a concepção de expressões lógicas e que contemplam uma relação 'um para um' com os indivíduos da classe *'Restriction'*.

Conforme apresentado na Figura 8 a IBSO propõe indivíduos iniciais para as classes '*Function*', '*Requirement*' e '*Comparator*'. As funções escolhidas representam de maneira direta os Requisitos Comunicacionais Funcionais levantados na seção anterior, e por isso, o IBSO propõe os indivíduos: *'unicast_intradomain'*, *'unicast_interdomain'*, *'multicast_intradomain'*, *'multicast_interdomain'*, *'broadcast'*, *'anycast_intradomain'* e *'anycast_interdomain'*. O requisitos representam de maneira direta os Requisitos Comunicacionais Não-Funcionais levantados na seção anterior, e por isso, o IBSO propõe os indivíduos: *'guaranteed_bandwidth'*, *'limited_bandwidth'*, *'dynamic_bandwidth'*, *'time_limited'*, *'usage_limited'*, *'cost_limited'*, *'error_free_channel'*, *'fast_channel'*, *'cheap_channel'*, *'safe_channel'*, *'optimal_channel'*, *'restricted_by_devices'*, *'restricted_by_applications'*, *'restricted_by_content'* e *'restricted_by_tecnology'*, *'restricted_by_territory'*. E por fim, os indivíduos comparadores propostos pela IBSO são: *'equals'*, *'not_equals'*, *'greater_than'*, *'lesser_than'*, *'equals_or_greater_than'*, *'equals_or_lesser_than'*, *'in'*, *'not_in'*.

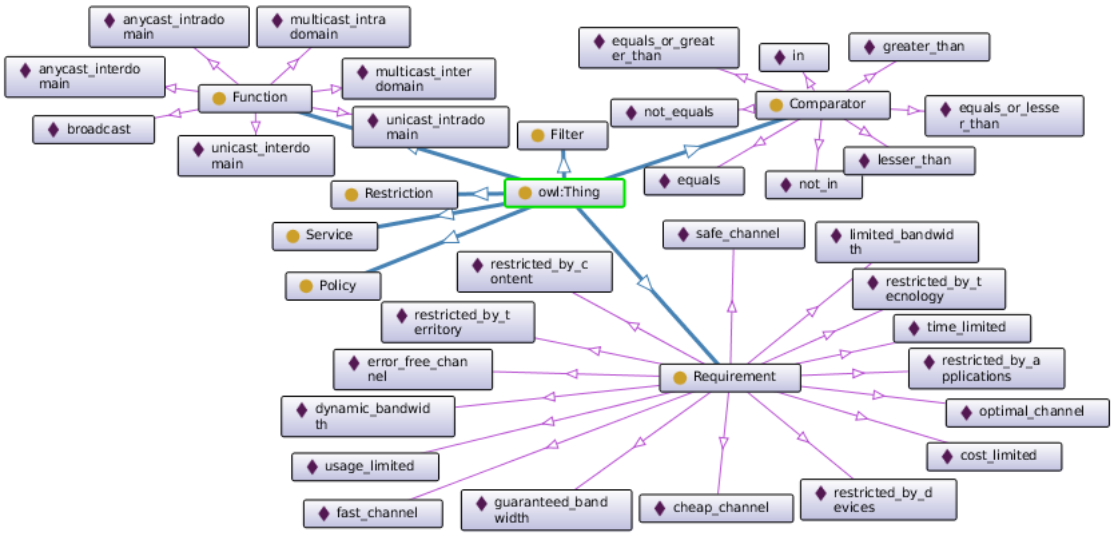


Figura 8 – Classes e Indivíduos da *Intent-Based Services Ontology* (IBSO).

Fonte: Elaborado pelo autor (2021).

A Figura 9 apresenta de maneira clara como as classes propostas pela IBSO se relacionam através de uma visão simplificada de um Diagrama de Classes geradas a partir do arquivo OWL e traduzido para português.

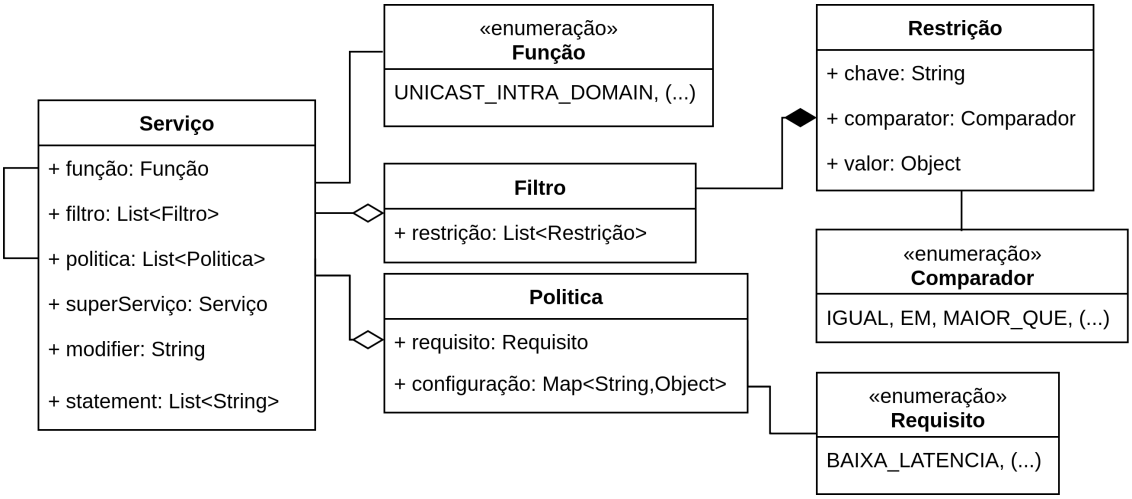


Figura 9 – Diagrama de Classes simplificado baseado na IBSO.

Fonte: Elaborado pelo autor (2021).

4.1.1.4 Classificação de um *Intent-Based Service*

A *granularidade* e a *dimensão* de um IBS variam de acordo com a abrangência do seu *filtro*, por exemplo, pode-se ter serviços individuais para cada aplicação (cenário de maior granularidade e menor dimensão), ou pode-se ter um único serviço por sistema final (cenário de menor granularidade e maior dimensão). Dessa maneira, pode-se definir *subserviços* que estendam os filtros de serviços de maior dimensão de maneira a aplicar uma tratativa diferente em um contexto mais específico. Por exemplo, pode-se abstrair um *link* genérico de acesso à internet como um serviço e o tratamento diferenciado das comunicações multimídia trafegadas nesse *link* como um *subserviço*. Um subserviço é como uma extensão de um serviço genérico, assim todas as políticas e filtros são herdados e combinados com as do subserviço. Da mesma maneira, um superserviço é um serviço mais genérico que é estendido por subserviços.

IBSs podem ser classificados como *abstratos* ou *concretos*. Um 'IBS Abstrato' define uma categoria de serviços por meio de políticas e filtros que serve como base para outros subserviços e que não pode ser aprovisionado diretamente na rede. Esse tipo de serviço é útil para definir padrões de serviços e políticas administrativas que devem ser aplicadas a um conjunto de serviços simultaneamente. 'IBSs Concretos' são serviços convencionais que podem ser aprovisionados diretamente na rede. Um serviço também pode ser definido como *final* para impedir que novos subserviços sejam definidos. Dessa forma, não faz sentido um serviço ao mesmo tempo 'final' e 'abstrato'.

IBSs também podem ser classificados quanto ao momento de aprovisionamento da seguinte forma:

- **Pré-configurados**: serviços configurados previamente, antes da comunicação se iniciar;

- **Sob-demanda**: serviços configurados apenas quando necessário, logo no início da comunicação;

- **Programados**: serviços configurados em determinados contextos, como por exemplo: momento, estado da rede ou tipo de conteúdo.

Esses tipos de serviços se complementam na flexibilização da comunicação em rede, pois permitem a configuração da rede com diferentes estratégias: estática (pré-configurados) ou dinâmica (sob-demanda ou programado). Pode-se definir superserviços pré-configurados e subserviços programados para determinados contextos. Serviços sob-demanda normalmente são definidos como subserviços pois requerem que a comunicação já esteja em curso para a identificação dos requisitos comunicacionais. Neste caso, um serviço genérico é configurado para permitir o tráfego de dados sem tratamento especial, e após detecção de um padrão ou conteúdo específico, um subserviço é aplicado para tratar a comunicação de maneira mais específica. Outra forma de aprovisionar serviços sob-demanda está em detectar tentativas de comunicação no primeiro dispositivo ao qual o sistema final está conectado, o que dispensaria a definição de um superserviço.

4.1.1.5 Representação de um *Intent-Based Service*

Pode-se representar serviços de diferentes formas, sejam formais ou informais, desde que os detalhes do serviço (função, políticas e filtros) possam ser extraídos ou inferidos. O IBSM define três formas de representação:

- FDIBS (*Formally Described Intent-Based Service*): serviços descritos por meio de uma linguagem formal facilmente interpretável por sistemas computacionais;

- BIIBS (*Behavior Inferred Intent-Based Service*): serviços inferidos por meio de uma análise comportamental com dados coletados da rede;

- NDIBS (*Natural Described Intent-Based Service*): serviços descritos por meio de uma linguagem natural através de afirmativas (*statements*) ou texto livre;

Os Serviços Descritos por Linguagem Natural (NDIBSs) e os Serviços Descritos Formalmente (FDIBSs) são baseados no conceito de *Declarative Intent* (DI) que apresenta intenções através de um modelo de representação declarativo, sendo NDIBSs representados por uma linguagem informal, e FDIBSs representados por uma linguagem formal. Os Serviços Inferidos por Comportamentos (BIIBSs) são representados de maneira abstrata a partir de padrões de comportamento com base no conceito de *Behavioral Intent* (BI). Deve ser possível traduzir tanto NDIBSs quanto BIIBSs para FDIBSs, pois são estes os serviços interpretados pela rede.

Formally Described Intent-Based Services **(FDIBSs)**

A representação de FDIBSs se baseia na utilização de uma representação formal que pode ser tanto codificada (a partir de primitivas especificadas por um protocolo) ou textual (de acordo com *schemas* e definições com a incluída no Apêndices C).Essa forma de representação é a mais próxima da rede, e por isso, deve ser interpretável por sistemas computacionais de maneira inequívoca.

Abaixo são apresentados exemplos com abstrações de FDIBSs:

- Descrição em OWL de acordo com o *Intent-Based Services Ontology* (incluído no Apêndice C):

```
(...)
  <!-- Service -->
  <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm/services/example">
    <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#Service"/>
    <service:filter rdf:resource="http://ufu.br/mehar/ibsrm/filters/0000000001_001_001"/>
    <service:function rdf:resource="http://ufu.br/mehar/ibsrm/functions/unicast_interdomain"/>
    <service:policy rdf:resource="http://ufu.br/mehar/ibsrm/policies/guaranteed_bandwidth_10mb"/>
  </owl:NamedIndividual>
  <!-- Policies -->
  <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm/policies/guaranteed_bandwidth_10mb">
    <policy:requirement rdf:resource="http://ufu.br/mehar/ibsrm/requirements/guaranteed_bandwidth"/>
    <policy:configuration>{bandwith=10mb}</policy:configuration>
  </owl:NamedIndividual>
  <!-- Filters and Restrictions-->
  <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm/filters/0000000001_001_001">
    <filter:restriction rdf:resource="http://ufu.br/mehar/ibsrm/restrictions/0000000001_001_001_01"/>
    <filter:restriction rdf:resource="http://ufu.br/mehar/ibsrm/restrictions/0000000001_001_001_02"/>
  </owl:NamedIndividual>
  <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm/restrictions/0000000001_001_001_01">
    <restriction:comparator rdf:resource="http://ufu.br/mehar/ibsrm/comparators/equals"/>
    <restriction:key>ip_src</restriction:key>
    <restriction:value>172.168.0.45</restriction:value>
  </owl:NamedIndividual>
  <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm/restrictions/0000000001_001_001_02">
    <restriction:comparator rdf:resource="http://ufu.br/mehar/ibsrm/comparators/in"/>
```

```
26          <restriction:key>day_of_week</restriction:key>
27          <restriction:value>[&amp;quot;sat&amp;quot;,&amp;quot;sun&amp;
              quot;]</restriction:value>
28      </owl:NamedIndividual>
29    (...)
```

❏ Descrição em JSON criado a partir de um *schema* baseado na IBSO;

```
1  {
2      "function":"unicast_interdomain",
3      "policy":[
4          {
5              "requirement":"guaranteed_bandwidth",
6              "configuration":{
7                  "bandwidth":"10MB"
8              }
9          }
10     ],
11     "filter":[
12         {
13             "restriction":[
14                 {
15                     "key":"ip_src",
16                     "comparator":"equals",
17                     "value":"172.168.0.45"
18                 },
19                 {
20                     "key":"day_of_week",
21                     "comparator":"in",
22                     "value":[
23                         "sat",
24                         "sun"
25                     ]
26                 }
27             ]
28         }
29     ]
30 }
```

Em ambas as representações é definido um serviço pra comunicação *unicast* e *interdomain*, com largura de banda garantida de 10MB que deve ser ativo aos finais de semana para comunicações a partir do IP 172.168.0.45.

Behavior Inferred Intent-Based Services **(BIIBSs)**

A representação de BIIBSs se baseia em uma comportamental da rede. Para isso, mecanismos específicos precisam analisar dados coletados dos recursos de infraestrutura, tais como métricas e amostras, de maneira a detectar padrões de conteúdo, tráfego e comportamento.

Estes serviços são sempre tratados como serviços sob-demanda, uma vez que dependem de estímulos externos. Abaixo são apresentados alguns exemplos de BIIBSs:

- ❏ Detecção de fluxo TCP (com a primeira primitiva do *handshake*) ou UDP (com a transmissão da primeira primitiva da comunicação), verificação de portas conhecidas, análise de conteúdo (com identificação de protocolos de aplicação). Dessa forma poderia ser inferido um serviço com filtro baseado nos campos de endereçamento (IPs e portas de origem e destino); função de acordo com o protocolo e o cabeçalho da primitiva de rede (ex. *unicast intra-domain*); e políticas de acordo com o protocolo de aplicação (ex. RTP/RTSP – baixa latência e banda garantida, FTP e SMTP – canal seguro e confiável);

- ❏ Interceptação de consulta ao DNS, identificação da aplicação de destino e coleta de detalhes (função e políticas) através do registro SRV. Um serviço poderia ser inferido sempre que uma consulta ao DNS é detectada (o que normalmente é realizado antes da comunicação). Nesse caso a base de requisitos comunicacionais do serviço poderia ser o próprio DNS através do registro do tipo SRV (ou outro registro específico);

- ❏ Inspeção do conteúdo de uma comunicação em curso e identificação de tipo. Isso poderia ser utilizado para prover subserviços específicos de acordo com o conteúdo. Por exemplo, ao detectar um conteúdo multimídia a rede infere requisitos mais comuns de comunicações com esse tipo de conteúdo;

***Natural Described Intent-Based Services* (NDIBSs)**

A representação de NDIBSs se baseia na utilização de linguagem natural para definição de requisitos comunicacionais e contexto da comunicação. Para isso são necessários mecanismos de *Natural Language Processing* (NLP), capazes de traduzir uma descrição informal de um serviço em uma descrição formal (tradução para FDIBS).

A descrição de intenções em linguagem natural facilita a utilização da rede por usuários com baixo conhecimento técnico e torna serviços auto-descritivos. Abaixo são apresentados alguns exemplos de abstração de NDIBSs:

- ❏ "Quero acessar o *Netflix* todos os dias após o trabalho (19h)"
  Na interpretação dessa descrição a rede precisa identificar a origem, o destino e as variáveis de contexto temporal (dias de semana entre 19h e 06h) pra abstrair os filtros; considerando o destino (*Netflix* no caso) a rede precisa identificar qual o tipo de comunicação a ser utilizada (*unicast inter-domain* por exemplo); e quais as políticas mais comuns pra essa comunicação (baixa latência e banda garantida por exemplo);

- ❏ " – a comunicação deve ter largura de banda mínima de 10Mb;"
  " – devem ser utilizados apenas recursos de infraestrutura localizados no Brasil;"
  " – deve-se priorizar a economia de energia;"

" – a comunicação ocorrerá no primeiro dia útil de cada mês;"

No exemplo acima, o serviço é descrito por afirmações (*statements*), que é uma forma de representação mais facilmente traduzível para uma descrição formal. Pode-se detectar a necessidade de políticas para garantia de banda, restrição de recursos e foco em economia energética. O filtro ainda é obtido pela análise da origem e contexto temporal (primeiro dia útil de cada mês). Como uma função não foi claramente definida, a rede pode optar por prover todos os tipos de funções disponíveis.

4.1.2 IBSPM - Modelo de Aprovisionamento de Serviços Baseados em Intenções

O *Intent-Based Service Provisioning Model* (IBSPM) é um modelo para aprovisionamento de serviços que se baseia em cinco etapas básicas (apresentadas na Figura 10):

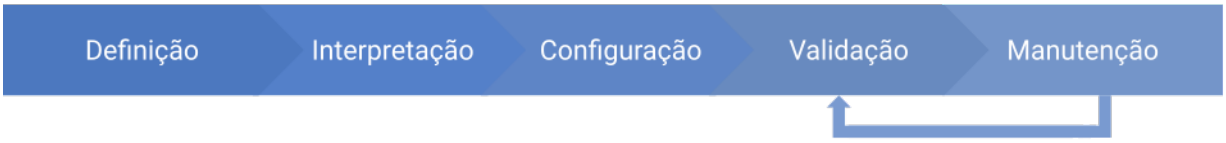


Figura 10 – Fluxo geral de aprovisionamento de serviços definido pelo IBSPM.

Fonte: Elaborado pelo autor (2021).

- **Definição**: abstrai intenções detectadas ou declaradas como serviços;
- **Interpretação**: traduz o serviço em instruções de configuração;
- **Configuração**: aplica as configurações necessárias para a ativação do serviço;
- **Validação**: verifica se o serviço atende às políticas definidas por intenções;
- **Manutenção**: corrige erros na configuração caso os níveis de serviço não sejam atingidos;

Essas etapas são desempenhadas pelos componentes apresentados na Figura 11. A *Definição* se dá por meio da detecção de comportamento (via IBSBD), tradução de linguagem natural (via IBSNLT) ou descrição formal. A *Interpretação* é realizada pelo IBSI que traduz os serviços descritos formalmente em instruções para a configuração da infraestrutura. A *Configuração* fica a cargo do IBSA que é quem efetivamente executa as instruções de configuração, e a *Validação* é realizada pelo IBSV que analisa dados e verifica se os níveis de serviço estão em acordância com as definições do serviço. A *Manutenção* fica a cargo tanto do IBSI quanto do IBSA que reavaliam os serviços para reestabelecer os níveis acordados. Esses componentes são descritos abaixo:

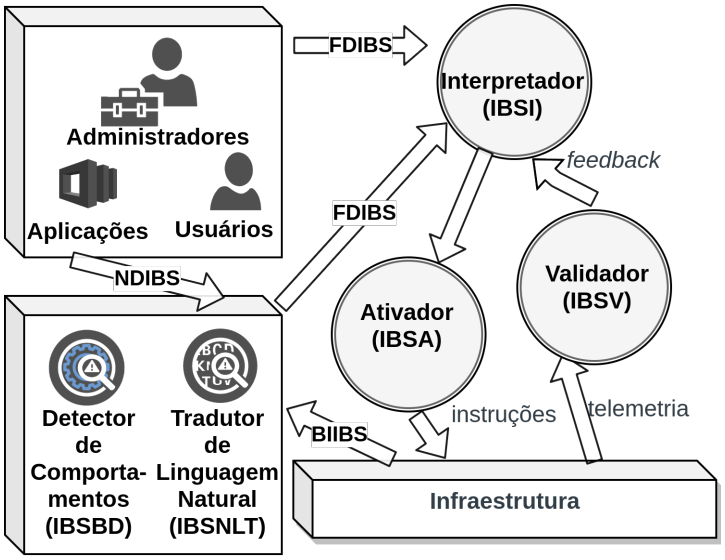


Figura 11 – Representação dos componentes conceituais utilizados no aprovisionamento de serviços de acordo com o IBSPM.

Fonte: Elaborado pelo autor (2021).

Interpretador de IBSs **(IBSI)**

O IBSI é o mecanismo responsável por identificar as melhores estratégias para atender às "intenções" nas quais os serviços se baseiam respeitando as regras de negócio e os objetivos administrativos. Ele define quais os recursos devem ser alocados e como eles devem ser configurados para o aprovisionamento do serviço. A interpretação de serviços pode ser construída através de inteligência artificial com base no *feedback* provido pelo IBSV, de maneira avaliar as decisões tomadas, testar novas abordagens e flexibilizar a configuração de acordo com o contexto (momento, região, tecnologias e estado da rede);

Ativador de IBSs **(IBSA)**

O IBSA é o mecanismo responsável por implantar configurações na infraestrutura conforme a especificação do IBSI. A implantação de serviços pode se dar diretamente na infraestrutura ou através de componentes de controle (como controladores SDN e orquestradores NFV).

Validador de IBSs **(IBSV)**

O IBSV é o mecanismo responsável por monitorar serviços usando telemetria e fornecer *feedback* ao IBSI de maneira a guiar a construção da base de conhecimento para interpretação de intenções. Ele é responsável por avaliar a eficácia do aprovisionamento e por garantir que o serviço atenda aos níveis acordados.

***Detector de Comportamentos para Inferência de IBSs* (IBSBD)**

O IBSBD é o mecanismo responsável pela detecção de comportamentos de rede por meio da análise de padrões de uso e de dados. Ele é responsável por detectar a necessidade de serviços; inferir as políticas, os filtros e a função; e por solicitar o aprovisionamento do serviço ao IBSI.

***Tradutor de Linguagem Natural para Representação de IBSs* (IBSNLT)**

O IBSNLT é o mecanismo responsável por traduzir serviços baseados em intenções descritas em linguagem natural para uma representação formal. Para isso ele aplica algoritmos de processamento de linguagem natural (NLP) com uma base de conhecimento construída com o *feedback* do IBSV. A interpretação de serviços definidos com linguagem natural simplifica a configuração da rede por eliminar a necessidade de conhecimentos técnicos.

4.2 SONeM: Modelo de Referência para Redes Auto-organizadas

O *Self-Organizing Network Model* (SONeM) propõe uma abstração conceitual de Rede Auto-Organizada através de camadas conforme é apresentado na Figura 12. Dessas camadas apenas três efetivamente fazem parte da rede (gerenciamento, controle e infraestrutura). A principal camada proposta por este modelo é a de 'Gerenciamento' que engloba todos os componentes responsáveis pela a automação do gerenciamento da rede. Essa camada de divide em outras três subcamadas responsáveis pela coleta e análise de dados, tomada de decisão e intervenção na rede.

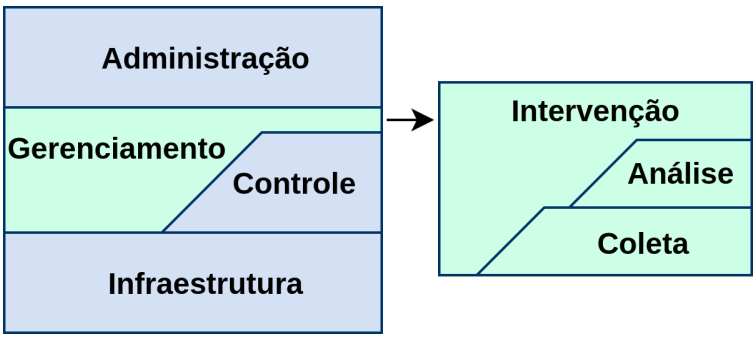


Figura 12 – Abstração da rede em camadas na visão do Modelo de Referência SONeM.

Fonte: Elaborado pelo autor (2021).

Abaixo uma breve descrição sobre as camadas do SONeM:

- **Infraestrutura** (*Infrastructure Layer*) – engloba todos os recursos de rede físicos e lógicos. Exemplo: *switches*, roteadores, servidores, funções de rede e *appliances*;

- **Controle** (*Control Layer*) – engloba componentes de controle de redes legadas e de propostas pra redes futuras. Exemplo: controladores SDN, NFV MANO, ferramentas de configuração e gerências de rede;

- **Gerenciamento** (*Management Layer*) – engloba componentes responsáveis pelo auto-gerenciamento da rede. Exemplo: componentes propostos pelo SONAr.

  - **Coleta** (*Collection Sublayer*) – engloba componentes responsáveis pela coleta de dados de dados na infraestrutura e nos demais componentes. Exemplo: *Collecting Entities* (CoEs) propostas pelo SONAr;
  - **Análise** (*Analysis Sublayer*) – engloba componentes responsáveis pela análise dos dados coletados, realização de diagnósticos e previsão de cenários. Exemplo: *Self-Learning Entities* (SLEs) propostas pelo SONAr;
  - **Intervenção** (*Intervention Sublayer*) – engloba componentes responsáveis pela tomada de decisão e configuração da infraestrutura e dos demais componentes de acordo com as propriedades suportadas pela rede (e.g. configuração, cura, otimização e proteção). Exemplo: *Self-Organizing Entities* (SOEs) propostas pelo SONAr;

- **Administração** (*Administration Layer*) – engloba ferramentas administrativas e integrações com outros sistemas da organização. Exemplo: sistemas de OSS/BSS, *Dashboard* de Administração da Rede (NAD) e ferramentas de monitoramento de SLA.

4.2.1 Termos e Conceitos Relevantes

Para a proposição do SONeM são utilizados termos e conceitos chaves úteis para o entendimento da proposta. Estes termos são definidos abaixo:

- Recurso – refere-se aos elementos físicos (dispositivos, servidores, cabos...) e lógicos (aplicações, funções, protocolos, vlans, IPs...) que compõem a infraestrutura de rede;

- Componente – refere-se aos elementos responsáveis pelo controle e gerenciamento da rede (entidades SONAr, controladores SDN...);

- Dados de Rede – refere-se aos diferentes tipos de informações extraídas de recursos e componentes da rede: métricas, *logs*, amostras, resultados de testes e informações de topologia;

- Fluxo de Controle e Gerenciamento – refere-se ao tráfego de primitivas entre recursos de infraestrutura e componentes de controle e de gerenciamento respectivamente. A expressão "estabelecer o fluxo de controle e gerenciamento" se refere à configuração de um canal de comunicação (com rotas e fluxos) para ser utilizado pelos componentes da rede;

4.2.2 Camada de Gerenciamento

A 'Camada de Gerenciamento' é o ponto central do SONeM. Ela define uma interface superior (*Northbound Interface* (NBI)) com a 'Camada de Administração' de maneira a permitir tarefas como: configuração de serviços de comunicação, consulta do estado da rede, correlação de dados, diagnóstico de problemas, parametrização de componentes e etc. Para isso a NBI da Camada de 'Gerenciamento' deve disponibilizar uma API de serviços para integração com sistemas de OSS/BSS e ferramentas administrativas.

A interface inferior (*Southbound Interface* SBI) da Camada de 'Gerenciamento' permite a comunicação com componentes de controle ('Camada de Controle') e com recursos de infraestrutura ('Camada de Infraestrutura'). Isso flexibiliza o gerenciamento de rede, pois permite a adoção do SONeM mesmo em redes tradicionais nas quais o controle é distribuído. Ao mesmo tempo, a possibilidade de comunicação diretamente com os recursos de infraestrutura implementação de componentes especializados nas tecnologias destes recursos de forma independente, o que simplifica e acelera o desenvolvimento de novos componentes/*plugins* e diminui o *overhead* e a latência na comunicação entre as camadas de infraestrutura e gerenciamento. Além disso, com essa abordagem os mesmos componentes de gerenciamento podem gerir redes com diferentes abordagens de controle e tecnologias de infraestrutura de maneira transparente para usuários e administradores da rede.

A 'Camada de Gerenciamento' se divide em três subcamadas responsáveis pela abstração do fluxo de auto-gerenciamento proposto pelo SONeM e descrito na seção 4.2.3. Dessa maneira, é definida a 'Subcamada de Coleta' responsável por coletar dados como métricas, *logs*, alarmes e etc provenientes de recursos de infraestrutura ou de componentes de controle/gerenciamento. A 'Subcamada de Coleta' atua diretamente na SBI da 'Camada de Gerenciamento' com o objetivo de coletar dados provenientes das camadas de infraestrutura, controle e da própria camada de gerenciamento. Os componentes desta camada desempenham um papel chave na abstração da propriedade de sistemas autônomos de 'auto-consciência', uma vez que permite que os demais componentes façam uma análise adequada do estado da rede.

A 'Subcamada de Análise' é responsável por correlacionar os dados coletados pelos componentes de coleta com o objetivo de detectar anomalias na utilização da rede, diagnóstico de falhas, oportunidades de melhoria, previsão de cenários e avaliação de recursos. O caminho natural para a implementação de componentes desta subcamada está na utili-

zação de técnicas de Inteligência Artificial (IA), mas acredita-se outras abordagens como uma abordagem empírica também sejam possíveis. Os componentes desta subcamada atuam de maneira complementar com os componentes da subcamada de coleta para a abstração da propriedade de 'auto-consciência', pois permitem que a rede seja capaz de se auto-analisar. Além disso, quando se baseiam em técnicas de IA os componentes dessa subcamada abstraem também a propriedade de 'auto-aprendizado' o que flexibiliza a implementação da solução para os mais variados contextos.

Por fim, a 'Subcamada de Intervenção' engloba os principais componentes responsáveis pelo gerenciamento da rede. Estes componentes precisam tomar decisões com base nas informações coletadas e analisadas pelos componentes das outras subcamadas; escolher um plano de ação; e, atuar diretamente sobre os recursos da infraestrutura e componentes de controle/gerenciamento de maneira a viabilizar as principais propriedades de autônomos (e.g. 'auto-configuração', 'auto-cura', 'auto-otimização' e 'auto-proteção'). Esta subcamada é responsável pela implementação dos fluxos de tratamentos específicos de cada propriedade, bem como pela orquestração destes fluxos de maneira a manter a rede sempre operacional.

4.2.3 Fluxo de Auto-Gerenciamento

O fluxo básico de 'Auto-Gerenciamento' proposto pelo SONeM é exemplificado na Figura 13 em três estágios: *i)* coleta de dados; *ii)* análise de cenários; e, *iii)* tomada de decisão e intervenção na rede.

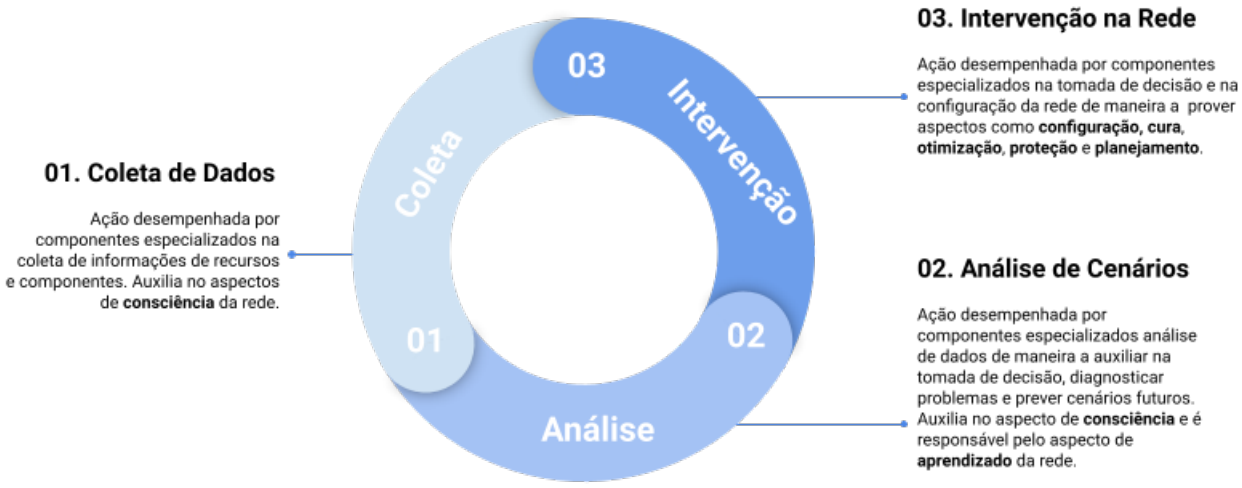


Figura 13 – Representação simplificada do fluxo de auto-gerenciamento proposto pelo SONeM.

Fonte: Elaborado pelo autor (2021).

A primeira etapa consiste em prover a propriedade de auto-consciência através do monitoramento e coleta de dados da rede. Estes dados podem ser obtidos através técnicas

específicas de acordo com as tecnologias e decisões de implementação. Pode-se optar, por exemplo, por uma estratégia de *pooling* para coletar métricas de fluxos em um equipamento, ou pode-se aguardar um *push* de dados do próprio dispositivo.

A segundo etapa consiste na análise dos dados de rede e na identificação de problemas ou oportunidades; detecção de padrões de conteúdo e comportamento; previsão de cenários e prospecção de recursos; e, correlação de dados e eventos. Essa etapa normalmente se dá pela abstração da propriedade de 'auto-aprendizado' que é flexibiliza o gerenciamento autônomo da rede.

A terceira etapa que fecha o ciclo é a de tomada de decisão e intervenção na rede. Os cenários detectados na etapa anterior tais como degradação da rede, evidências de ataques e oportunidades de otimização são tratados nessa etapa onde são executados os componentes responsáveis pelas principais propriedades de auto-organização: auto-configuração, auto-cura, auto-otimização, auto-proteção e auto-orquestração.

Capítulo 5

Visão Geral da *Self-Organizing Network Architecture* (SONAr) e Aplicação na Automação de Operações de Configuração

Este capítulo almeja apresentar a *Self-Organizing Network Architecture* (SONAr) uma Arquitetura de Rede Auto-Organizada que propõe a automação do gerenciamento por meio de componentes de coleta de dados, análise de cenários, tomada de decisão e intervenção na rede de acordo com os princípios do SONeM proposto na seção 4.2). Na seção 5.2 é apresentada uma solução com base no IBSM (seção 4.1) e no SONAr (proposto neste capítulo) para automação das operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços em redes auto-organizadas.

5.1 SONAr: Arquitetura para Redes Auto-organizadas

Self-Organizing Network Architecture (SONAr) é uma arquitetura para redes auto-organizadas inspirada no conceito de SON (seção 2.2.2.1) que propõe o uso de entidades especializadas nas propriedades de auto-organização apresentadas na seção 2.2.3. De maneira geral a SONAr visa fornecer recursos *self-** [1] para as redes de computadores com o objetivo de viabilizar uma abordagem de organização sem intervenção humana através de conceitos de computação autônoma aplicados. SONAr possui componentes responsáveis por cada aspecto da gestão com uma abordagem autônoma, desde a coleta e análise de informações até a tomada de decisão e intervenção na rede. Embora a SONAr

[1] e.g. auto-configuração, auto-cura, auto-otimização, auto-proteção, auto-orquestração, auto-aprendizado e auto-consciência

tenha sido projetado com foco nas redes SDN/NFV, o que a torna também uma candidata para aplicação no núcleo do 5G, ela também se destina a lidar com redes e protocolos legados.

A SONAr utiliza uma abordagem de divisão em camadas conforme definido no modelo de referência SONeM. A camada de gerenciamento engloba quatro grupos de entidades/-componentes:

- ❏ auto-organizadoras (seção 5.1.2) – responsáveis por implementar fluxos de trabalho para configurar, curar, proteger e otimizar serviços e recursos de rede;
- ❏ auto-aprendizado (seção 5.1.3) – responsáveis por analisar os dados de rede (e.g. utilização e *link status*) para apoiar a tomada de decisão sem intervenção humana;
- ❏ coletoras (seção 5.1.4) – responsáveis por coletar e monitorar recursos de infraestrutura, componentes de controle e outras entidades; e,
- ❏ auxiliares (seção 5.1.5) – responsáveis por prover funções que viabilizem a operação dos demais componentes, como por exemplo: base de dados e *broker* de eventos.

A Figura 14 apresenta uma visão mais ampla da arquitetura, com exemplos de cada camada e apresentação dos componentes SONAr. Nesta figura a camada de gerenciamento é posicionada acima da camada de controle e da camada de infraestrutura simultaneamente (e não apenas acima da camada de controle). Essa decisão de *design* foi tomada para flexibilizar o plano de gerenciamento que pode acessar diretamente recursos de infraestrutura ou mesmo assumir as funções da camada de controle.

A Figura 15 apresenta uma visão da arquitetura com foco nos detalhes de integração. No projeto de arquitetura utilizado na concepção da SONAr foram considerados aspectos de *design patterns* de desacoplamento e coesão de componentes. Dessa maneira, cada componente da arquitetura tem uma função específica e é integrado aos demais a partir do compartilhamento da base de dados que é distribuída e pela troca de eventos sem definições rígidas. A abordagem de integração da SONAr se baseia no conceito de EDA, o que permite que os componentes de gerenciamento atuem de maneira independente com um fluxo de trabalho que se baseia no consumo e na publicação de eventos. Por exemplo, as entidades coletas buscam dados na infraestrutura e publicam eventos que são recepcionados pelas entidades de análise/aprendizado. Estas por sua vez realizam um diagnóstico e publicam um novo evento que é consumido pelas entidades de intervenção/organização. Por fim, estas entidades aplicam um fluxo de tomada de decisão e intervenção na rede de maneira a reagir ao diagnóstico realizado. O compartilhamento de dados também permite uma estratégia de gerenciamento distribuído "*stateless*", uma vez que cada entidade sempre utiliza os dados mais recentes na base compartilhada.

As entidades que compõe o SONAr são essencialmente elementos conceituais de gerenciamento que podem ser abstraídos como *appliances*, aplicações SDN, funcionalidades de

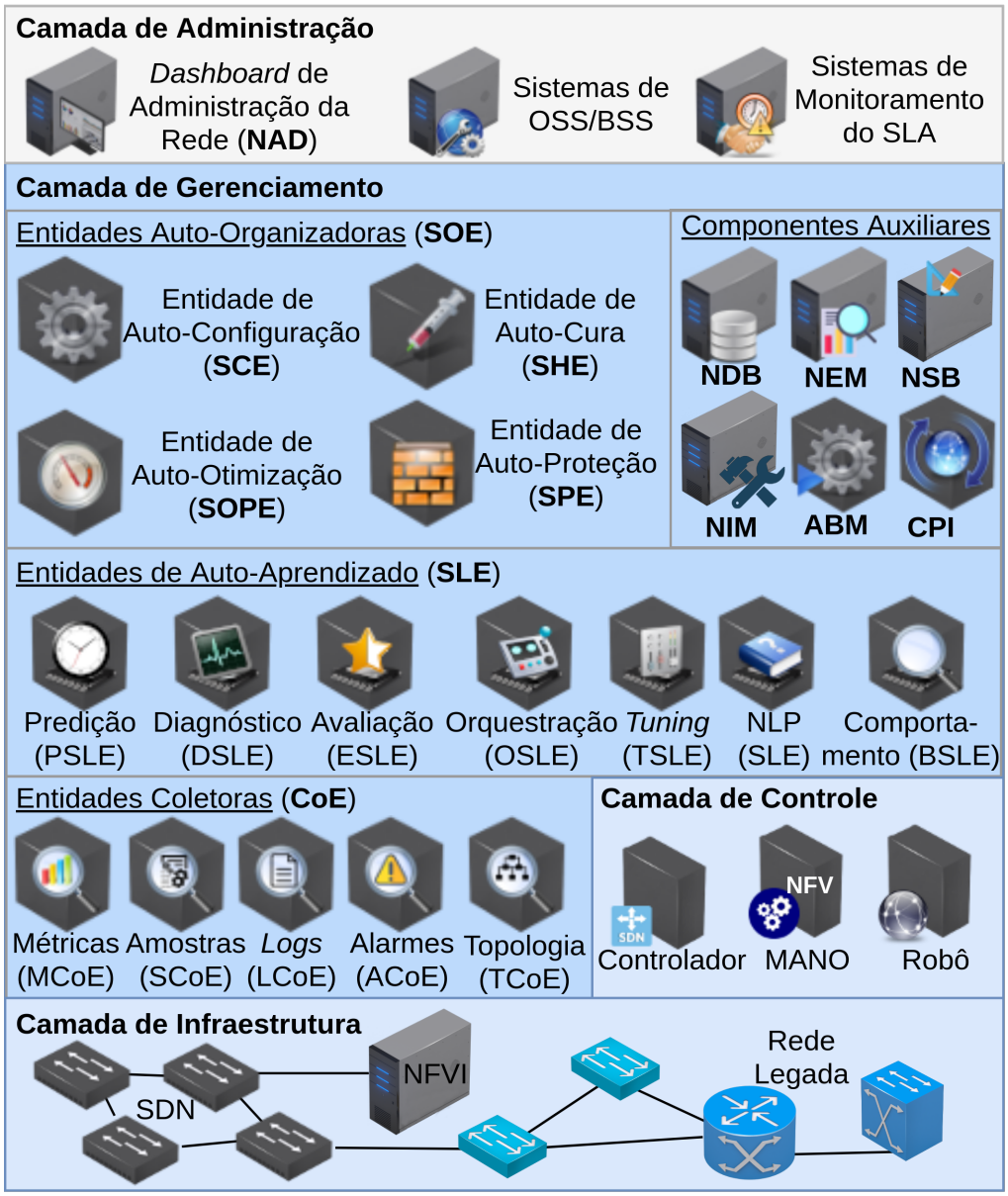


Figura 14 – Ilustração em alto-nível da Arquitetura SONAr.

Fonte: Elaborado pelo autor (2021).

roteadores, funções de rede virtualizadas/conteinerizadas, ferramentas de gerenciamento e etc. Os detalhes de implementação e a forma de abstração dessas entidades variam de acordo com as tecnologias disponíveis, o porte da rede e os objetivos/políticas definidas pelos administradores.

5.1.1 Modelos de Referência

O SONAr materializa o IBSM (seção 4.1) através da utilização do modelo de representação de serviços IBSRM que é suportado por todas as entidades; e, do modelo de aprovisionamento de serviços IBSPM que é abstraído pelas entidades: NSLE (seção

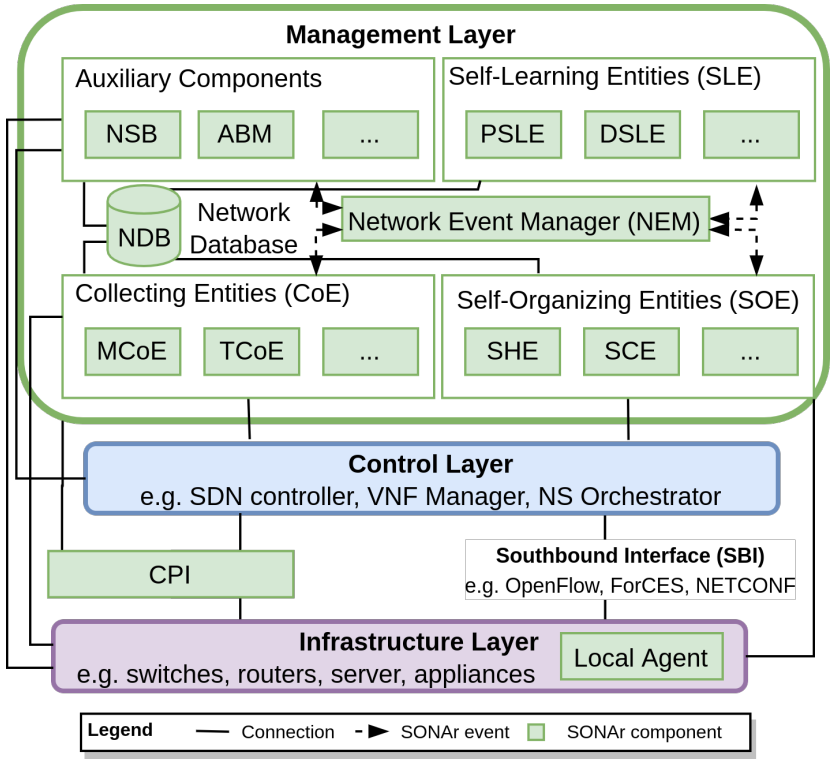


Figura 15 – Mapeamento de Relações entre os Componentes SONAr.

Fonte: Elaborado pelo autor (2021).

5.1.3.5), BSLE (seção 5.1.3.6), SCE 5.1.2.1) e MCoE 5.1.4.2). Todas as entidades SONAr trabalham com a abstração de políticas como representação formal de intenções. As entidades auto-organizadoras são especialmente responsáveis pelas etapas de interpretação, configuração e manutenção de serviços, e as coletoras e de auto-aprendizado pela validação do serviço. O SONAr também define componentes para tradução de serviços descritos em linguagem natural ou inferidos por análise comportamental através do uso de técnicas de inteligência artificial assim como proposto pelo IBSM. Além disso o SONAr utiliza o conceito de subserviços, serviços abstratos e serviços programados de com o objetivo de prover comunicações de maneira mais flexível às necessidades dos usuários.

O SONAr também materializa o SONeM através dos componentes da 'Camada de Gerenciamento' cujas funções estão diretamente relacionadas aos princípios das subcamadas de intervenção, análise e coleta. Na 'Subcamada de Coleta' estão as 'Entidades Coletoras' que buscam informações nos recursos da infraestrutura e componentes de controle/gerenciamento (assim como definido pela SBI da 'Camada de Gerenciamento' do SONeM). As 'Entidades de Auto-Aprendizado' atendem às definições da 'Subcamada de Análise' através de uma abordagem autônoma construída com técnicas de inteligência artificial. Por fim, as 'Entidades Auto-Organizadoras' atuam de acordo com os princípios da 'Subcamada de Intervenção', uma vez que operam com base nas informações coletadas e analisadas pelas demais entidades de maneira a prover requisitos de configuração, cura, otimização e proteção. Além disso, o SONAr também define uma NBI que permite

integração com sistemas de OSS/BSS e inclusive propõe o *Network Administration Dashboard* (NAD), uma ferramenta administrativa que permite a administração de alto nível da rede.

5.1.2 *Self-Organizing Entities* (SOEs)

As *Self-Organizing Entities* (SOEs) – Entidades Auto-Organizadoras – são entidades que efetivamente gerenciam a rede. Elas são responsáveis por implementar os fluxos de configuração, cura, proteção e otimização de redes auto-organizadas de maneira sincronizada e efetiva. Elas são as responsáveis por transformar os dados de rede e os resultados de análise em medidas concretas que alterem o comportamento da rede de acordo com o contexto e a necessidade, com o objetivo básico de manter a rede sempre funcionando e com níveis aceitáveis de disponibilidade e desempenho. Essas entidades abstraem de maneira direta as propriedades *self-** e implementam os componentes da 'Subcamada de Intervenção' conforme definido pelo SONeM.

As SOEs (assim como as SLEs) atuam sempre diante o estímulo de eventos, o que reforça os princípios de coesão e desacoplamentos defendidos pela SONAr. Isso facilita a integração entre essas entidades e as de análise e coleta, uma vez que não há definição de uma interface formal. Essa característica permite também em casos específicos a utilização de SOEs do mesmo 'tipo' mas com implementações específicas para determinados contextos e tecnologias. Por exemplo, pode-se ter duas entidades de auto-configuração coexistindo sendo uma especializada em redes SDN e outra em redes com roteamento BGP. Para tal comportamento é necessário que as entidades de auto-orquestrem o que pode ser feito através da abordagem de eventos na qual cada entidade evita a utilização de tópicos mais genéricos de maneira a evitar conflitos com outras entidades. Por exemplo, uma entidade de configuração de redes SDN deveria se inscrever nos tópicos de eventos específicos a dispositivos com suporte à esta abordagem e não a eventos de quaisquer tipos de dispositivos.

O problema de orquestração de SOEs não se restringe a sincronizar entidades do mesmo tipo, mas também entidades com as mais variadas funções. Em certos casos, como por exemplo na detecção de degradação da qualidade de um serviço, não é claro qual entidade deve atuar sem antes ser realizada uma análise adequada. Desta maneira, a SONAr propõe a utilização de uma entidade de análise/aprendizado específica para a orquestração da rede (OSLE – seção 5.1.3.7), que é capaz de atuar sobre eventos genéricos, realizar diagnósticos e publicar eventos mais específicos. Considera-se uma prática ruim que as SOEs se inscrevam em tópicos genéricos (o que pode eventualmente ser barrado na implementação do *broker* de eventos (NEM – seção 5.1.5.2), considerando a existência de uma entidade de análise que pode atuar de maneira a guiar as entidades de organização/intervenção em uma melhor decisão. Com base no exemplo da percepção de degradação, pode-se afirmar que as entidades especializadas em cura, proteção e otimiza-

ção deveriam se inscrever em tópicos mais específicos relacionados às questões inerentes de cada uma, como por exemplo, detecção de um *link down*, detecção de um ataque em curso ou detecção de uma configuração subótima da rede para o contexto atual.

A Figura 16 apresenta as entidades auto-organizadoras propostas pela SONAr. São elas:

- *Self-Configuration Entity* (SCE): responsável pela propriedade de auto-configuração (seção 5.1.2.1);
- *Self-Healing Entity* (SHE): responsável pela propriedade de auto-cura (seção 5.1.2.2);
- *Self-Protection Entity* (SPE): responsável pela propriedade de auto-proteção (seção 5.1.2.3);
- *Self-Optimization Entity* (SOPE): responsável pela propriedade de auto-otimização (seção 5.1.2.4);

Essas entidades implementam as etapas de *interpretação*, *configuração* e *manutenção* de serviços propostas pelo IBSM (seção 4.1) através da abstração dos componentes IBSI e IBSA (seção 4.1.2). São elas as responsáveis pela interpretação e aplicação das políticas definidas pelos serviços.

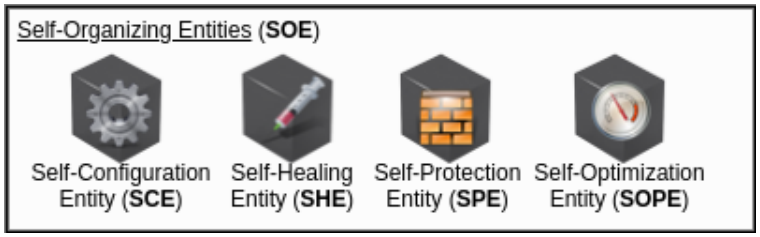


Figura 16 – Representação das Entidades Auto-Organizadoras da SONAr.

Fonte: Elaborado pelo autor (2021).

5.1.2.1 *Self-Configuration Entity* (SCE)

A *Self-Configuration Entity* (SCE) – Entidade de Auto-Configuração – é responsável pela abstração da propriedade de auto-configuração de redes auto-organizadas (seção 2.2.3.1). Essa entidade torna a rede operacional e provê serviços de comunicação de maneira transparente de acordo com a necessidade. Exemplos de atividades desempenhadas pela SCE incluem alocação, instalação, configuração e inicialização de recursos. Dentre as funcionalidades da SCE destacam-se:

- inicialização da rede (*bootstrapping*): torna a rede operacional a partir da configuração do plano de controle/gerenciamento, provisão recursos lógicos, configuração

de serviços de comunicação, configuração de recursos de rede e distribuição dos componentes;

- *plug-and-play* de recursos: permite a detecção e utilização de novos recursos de forma automática, estende os planos de controle e de gerenciamento, reavalia e reconfigura serviços;

- configuração de serviços de comunicação: provê a configuração de serviços de comunicação de acordo com os requisitos dos usuários, administradores e aplicações.

O estabelecimento do plano de controle é um ponto crucial para a auto-configuração da rede ao mesmo tempo em que é uma atribuição da própria SCE. Isso significa que a SCE deve ser capaz configurar recursos e serviços utilizando um plano de controle que ela mesma é responsável por configurar. O próprio aprovisionamento da SCE é uma parte da inicialização da rede e por esse motivo a SONAr propõe o ABM (seção 5.1.5.6), um componente que auxilia no aprovisionamento dos componentes de controle e gerenciamento para a inicialização da rede. A SCE também é responsável pela configuração inicial dos serviços e fluxos de controle/gerenciamento de acordo com o estado atual da rede, e todas as intervenções adicionais relacionadas a mudanças de contexto tais como otimização, cura e proteção ficam a cargo das demais entidades auto-organizadoras.

A SCE também é responsável pela autorização dos recursos de rede (tanto no *bootstrapping* quanto no *plug-and-play*) e dos usuários dos serviços de comunicação. Nas arquiteturas tradicionais quando um novo dispositivo é conectado e configurado ele passa a ser contemplado por protocolos como ARP, RIP e OSPF sem nenhum mecanismo de autorização. A SCE precisa executar um processo de autorização com os recursos de rede (*switches*, roteadores, servidores, *appliances*...) antes de efetivamente utilizá-los no aprovisionamento de serviços de comunicação o que pode ser feito através técnicas como troca de chaves e uso de certificados digitais.

5.1.2.2 SHE - *Self-Healing Entity*

A *Self-Healing Entity* (SHE) – Entidade de Auto-Cura – é responsável pela abstração da propriedade de auto-cura de redes auto-organizadas (seção 2.2.3.2). Para isso a SHE monitora o estado dos recursos de rede, dos serviços de comunicação e dos componentes da arquitetura de maneira a detectar e corrigir anomalias automaticamente. As técnicas de recuperação usuais executadas pela SHE incluem reinicialização, reinstalação, reconfiguração, realocação e restauração de recursos, serviços e componentes.

A manutenção do plano de controle é um aspecto fundamental da auto-cura, pois mesmo nas redes convencionais é necessário manter a conectividade entre todos os equipamentos de rede como premissa para a automação do gerenciamento. Por isso, a SHE tem a missão de lidar com cenários como indisponibilidade de equipamentos e rompimento

de conexões, e deve ser capaz de reconfigurar as regras de fluxo de controle (no caso de SDN) e de roteamento (no caso das redes atuais).

Da mesma maneira a SHE deve monitorar o estado dos serviços de comunicação implantados e reconfigurar os fluxos/rotas utilizados sempre que necessário. A SHE deve conseguir detectar degradação do serviço através de uma análise com base na taxa de erros, número de perdas e latência por exemplo, e reconfigurar os serviços de maneira a atender as políticas e níveis requeridos. Para isso os alarmes gerados pelos equipamentos e gerências de rede também precisam ser analisados, correlacionados e refletidos em ações práticas na configuração de rede.

Uma estratégia interessante para a auto-cura é a de utilização de pontos de restauração antes da alteração da configuração (similar a *backups*). Dessa maneira pode-se restaurar a rede a um estado anterior a uma falha. O registro das intervenções (e construção de uma base de conhecimento) também é útil na análise da causa raiz, diagnóstico de problemas/oportunidades e escolha de estratégias para resolução de problemas futuros.

5.1.2.3 SPE - *Self-Protection Entity*

A *Self-Protection Entity* (SPE) – Entidade de Auto-Proteção – é responsável pela abstração da propriedade de auto-proteção de redes auto-organizadas (seção 2.2.3.4). Para isso a SPE monitora dados de rede em busca de padrões característicos de ataques e invasões. Em uma abordagem mais pró-ativa, a SPE detecta e remove brechas de segurança antes que elas sejam utilizadas. O comportamento padrão da SPE se baseia nas atividades de isolamento, recuperação e prevenção de recursos alvos de ataques.

Essa entidade é responsável pela aplicação de políticas de segurança de acordo com as características e necessidades organizacionais. Isso quer dizer que a SPE pode ser usada para filtrar conteúdos impróprios ou restringir acessos em uma região ou organização. A SPE também é responsável pela identificação dos usuários e autorização de uso de serviços. Caso o usuário seja considerado não autorizado os fluxos de comunicação utilizados por ele são bloqueados. Isso elimina perdas por fraudes, torna a rede mais segura para os usuários e aumenta a disponibilidade dos serviços de comunicação.

A SPE também é responsável garantir a proteção de componentes das camadas de controle e de gerenciamento, e não apenas de infraestrutura. Um ataque bem-sucedido em um controlador SDN por exemplo forneceria ao invasor controle sobre toda infraestrutura. A SPE deve garantir que esses componentes só sejam acessíveis por pessoas/sistemas autorizados. Todas as tentativas suspeitas de acesso a esses componentes devem ser impedidas, e caso haja indícios de "infecção" os componentes devem ser restaurados.

Na abordagem reativa a primeira etapa consiste em isolar os componentes e recursos comprometidos, evitando que o ataque chegue aos demais. Após o isolamento, a SPE recupera o componente ou recurso atacado através de técnicas como: realocação, reinstalação, reconfiguração, restauração e reinicialização. Caso exista apenas uma suspeita de

tentativa de ataque mas sem sinais de degradação a SPE pode adotar uma abordagem preventiva impedindo que os serviços/recursos sejam afetados. Na abordagem pró-ativa a SPE é responsável pelo constante monitoramento das configurações de maneira a garantir as políticas de segurança definidas pelos administradores.

5.1.2.4 SOPE - *Self-Optimization Entity*

A *Self-Optimization Entity* (SOPE) – Entidade de Auto-Otimização – é responsável pela abstração da propriedade de auto-otimização de redes auto-organizadas (seção 2.2.3.3). Para isso ela monitora o estado da rede (componentes, recursos e serviços) em busca de oportunidades de otimização e aplica configurações para melhoria do desempenho ou redução de custos. Técnicas comuns de otimização da SOPE são: reconfiguração de serviços (incluindo mudança de rotas), ativação/desativação de recursos de acordo com a necessidade e *tuning* de recursos e componentes.

A primeira parte do trabalho da SOPE é identificar as oportunidades de melhoria o que pode ser realizado por meio do componente DSLE (seção 5.1.3.3) que aplica inteligência artificial na correlação de dados e diagnóstico de problemas/oportunidades de rede. As oportunidades detectadas devem ser utilizadas sempre que a rede estiver em estado operacional (sem degradação).

A SOPE é a responsável pela minimização dos custos através da otimização da utilização da infraestrutura. Dessa maneira essa entidade tem um papel fundamental na viabilização do negócio uma vez que ela atua de maneira a melhorar a relação custo/-benefício da rede. A economia de recursos pode também gerar reflexos na qualidade de serviço, uma vez que a rede passa a operar sempre em cenários com carga balanceada e controlada. A SOPE também é responsável por monitorar as políticas de restrição de uso dos serviços implantados de maneira a garantir que as configurações estejam sendo efetivas e bloquear serviços que tenham ultrapassado estes limites.

5.1.3 *Self-Learning Entities* (SLEs)

As *Self-Learning Entities* (SLEs) – Entidades Auto-Aprendizado – são entidades responsáveis pela análise dos dados coletados da infraestrutura e dos componentes de controle/gerenciamento de maneira a suportar a tomada de decisão das SOEs. Elas são responsáveis diretas pela abstração da propriedade de auto-aprendizado da rede (discutida na seção 2.2.3) e cumprem os princípios definidos pela 'Subcamada de Análise' do SONeM. Elas fornecem mecanismos baseados em Inteligência Artificial (IA) (IA) que viabilizam o gerenciamento da rede sem intervenção humana. Essas entidades usam bases de conhecimento que podem ser construídas com *deep-learning* através de uma análise de ação/consequência com as decisões tomadas pelos administradores, ou através da inferência com análise de cenários de rede. Essas bases podem ser importadas de empresas

especializadas em gerenciamento de rede autônomo em uma abordagem similar às atuais bases de antivírus (o que facilita a implantação em novas redes).

As SLEs implementam algoritmos de aprendizado de máquina para fornecer funcionalidades de previsão de cenários, otimização de parâmetros, diagnóstico de problemas/oportunidades, classificação de recursos, identificação de padrões de tráfego/conteúdo, controle de concorrência entre as entidades de organização e tradução de intenções descritas em linguagem natural. O processamento dessas atividades com técnicas de IA pode envolver o uso massivo de capacidade de processamento de acordo com a quantidade de dados e a complexidade de treinamento da *engine* e, portanto, precisam ser abordadas por mecanismos específicos com execuções assíncronas de maneira a não impactar negativamente no tempo de resposta das entidades auto-organizadoras. A funcionalidade de previsão, por exemplo, requer estratégias de representação discreta, armazenamento, classificação, busca e análise estatística de cenários.

Essas entidades processam dados coletados pelas entidades coletoras (seção 5.1.4) e publicam os resultados dessas análises em banco de dados (através NDB descrito na seção 5.1.5.1) e via eventos (através do NEM descrito na seção 5.1.5.2). Elas desempenham as atribuições da etapa de *validação* proposta pelo IBSM (seção 4.1) atuando como uma abstração do componente IBSV (4.1.2). A Figura 17 apresenta as entidades de auto-aprendizado propostas pelo SONAr:

- PSLE: responsável pela previsão de cenários e estimativa de uso de recursos (seção 5.1.3.1);
- TSLE: responsável pelo ajuste de parâmetros na configuração de recursos e de componentes (seção 5.1.3.2);
- DSLE: responsável pela correlação de dados/eventos e diagnóstico de problemas/oportunidades (seção 5.1.3.3);
- ESLE: responsável pela avaliação e classificação de recursos quanto aos requisitos comunicacionais (seção 5.1.3.4);
- NSLE: responsável pela tradução de serviços baseados em intenções descritas em linguagem natural e monitoramento dos níveis serviço para construção da base de conhecimento (seção 5.1.3.5);
- BSLE: responsável pela detecção de padrões de conteúdo e comportamento através da análise de métricas e amostras (*payloads* ou cabeçalhos) (seção 5.1.3.6); e,
- OSLE: responsável pela orquestração das entidades auto-organizadoras a partir de designação de diagnósticos às entidades auto-organizadoras. (seção 5.1.3.7);

Novas entidades de auto-aprendizado podem ser propostas de maneira a aprimorar o gerenciamento da rede. Entretanto, essas entidades devem ser coesas, devem realizar um processamento paralelo à execução da rede, não devem intervir diretamente na configuração, e devem utilizar os mesmos mecanismos de integração que as demais entidades.

Figura 17 – Representação das Entidades de Auto-Aprendizado da SONAr.

Fonte: Elaborado pelo autor (2021).

5.1.3.1 PSLE - *Prediction Self-Learning Entity*

A *Prediction Self-Learning Entity* (PSLE) – Entidade de Auto-Aprendizado para Predição – é responsável pela predição de cenários de rede com base no estado atual, informações sobre serviços programados e base de conhecimento. Essas previsões auxiliam na tomada de decisões e no planejamento da rede.

Um possível caminho de implementação para a PSLE está na aplicação de algoritmos de classificação cenários de rede com base em uma representação discreta composta de métricas, serviços, tempo e topologia. O parâmetro utilizados nesta técnica poderiam ser auto-ajustados pela TSLE (seção 5.1.3.2) para fornecer uma análise de melhor qualidade para cada uma das características (*features*). Essa classificação pode ser utilizada também como método de avaliação de cenários.

Nesta abordagem, o indivíduo que representa um cenário de rede pode ser construído como um conjunto de classificações obtidas por meio da clusterização de características obtidos com a aplicação do algoritmo de classificação. Estes indivíduos podem ser submetidos à um algoritmo de probabilidade condicional (e.g. *Inferência Bayesiana*) responsável por calcular a probabilidade de ocorrência de uma hipótese à medida que mais dados se tornam disponíveis.

5.1.3.2 TSLE - *Tuning Self-Learning Entity*

A *Tuning Self-Learning Entity* (TSLE) – Entidade de Auto-Aprendizado para Otimização de Parâmetros – é responsável pelo teste, avaliação e escolha de parâmetros para a configuração de recursos e componentes em tempo de execução. Os parâmetros de configuração podem ir de simples valores booleanos até escolhas mais sofisticadas como definição de uma técnica.

Uma abordagem de implementação desta entidade está no monitoramento da qualidade dos cenários em relação a um conjunto de parâmetros submetidos a um algoritmo de classificação. Nessa abordagem o indivíduo pode ser composto por um conjunto de parâmetros definidos em um estilo de valor-chave. A abstração da TSLE pode se basear em um modelo probabilístico de otimização (e.g. *Otimização Bayesiana*) capaz de propor parâmetros ideais que variam ao longo do tempo e que são influenciados pelos cenários da rede já observados.

Estes parâmetros também podem ser utilizados para definir os limites de atuação de cada SOE de maneira a guiar a orquestração dos fluxos de organização. Pode-se definir, por exemplo, um limite de utilização que caracteriza um problema como responsabilidade da entidade de auto-otimização e outro superior que designa o problema para a entidade auto-cura.

5.1.3.3 DSLE - *Diagnostics Self-Learning Entity*

A *Diagnostics Self-Learning Entity* (DSLE) – Entidade de Auto-Aprendizado para Diagnóstico – é responsável por analisar e correlacionar dados de rede para a garantia dos níveis de serviço, detecção de problemas de segurança, identificação de erros/falhas e acompanhamento da utilização de recursos por meio diagnósticos de problemas/oportunidades tratados por SOEs específicas.

A função da DSLE consiste em definir o conjunto de características obtidas através do processamento dos dados de rede (e.g. alarmes, métricas, amostras e *logs*) relacionados a problemas conhecidos tais como a subutilização de *links*, degradação dos níveis de serviço e suspeita de tentativa de ataques. Essa abordagem se assemelha ao diagnóstico médico que se baseia em anamnese e exames clínicos ou laboratoriais para realização de um diagnóstico (problema a ser tratado), escolha de um tratamento (e.g. remédios, cuidados e procedimentos) e exploração de um prognóstico (previsão da evolução do problema). De maneira similar a DSLE avalia os dados de rede como sintomas e atua realização de diagnósticos de problemas (ou oportunidades) para guiar as entidades de organização no tratamento e de predição no prognóstico.

A implementação da DSLE pode se basear no uso técnicas de inteligência artificial para a classificação de cenários assim como as entidades de análise/aprendizado exploradas nas seções anteriores. Para isso a DSLE deve utilizar um modelo de dados discreto baseado na clusterização de características que representam um cenário de rede. A base de conhecimento da DSLE pode ser alimentada a cada diagnóstico correto para viabilização de uma abordagem sem intervenção humana.

5.1.3.4 ESLE - *Evaluation Self-Learning Entity*

A *Evaluation Self-Learning Entity* (ESLE) – Entidade de Auto-Aprendizado para Avaliação de Recursos – é responsável por avaliar, classificar e comparar recursos utilizando

como critérios os requisitos comunicacionais definidos na seção 4.1.1.2. Por exemplo, ao considerar o requisito de *latência mínima* um dispositivo que processa e encaminha mensagens mais rapidamente tem uma classificação mais alta. Esses valores podem ser utilizados como pesos em algoritmos multiobjetivos para planejamento e implantação de serviços.

A classificação dos recursos é alterada à medida que mais dados se tornam disponíveis. No início a escolha dos recursos para o aprovisionamento ou manutenção de um serviço é arbitrária uma vez que todos os recursos possuem a mesma avaliação, mas após um tempo de amadurecimento as escolhas se tornam mais assertivas. Além disso, a própria utilização de um recursos pode afetar o seu desempenho e consequentemente a sua avaliação. Por exemplo, um dispositivo pode ter uma boa avaliação em relação ao requisito de latência mínima, mas ao começar ser utilizado no aprovisionamento de diversos serviços com esse mesmo requisito a latência pode aumentar devido a maior concorrência. Assim, recursos são constantemente reavaliados, e cabe à ESLE prover a avaliação no momento atual, no contexto histórico, valor médio e desvio padrão.

5.1.3.5 NSLE - *Natural Language Processing Self-Learning Entity*

A *Natural Language Processing Self-Learning Entity* (NSLE) – Entidade de Auto-Aprendizado para Processamento de Linguagem Natural – é responsável pela tradução de serviços baseados em intenções descritas em linguagem natural. A NSLE também tem a missão de avaliar dados para aferição dos níveis de serviço e melhoria da base de conhecimento para aplicação de NLP. Esta é uma abstração direta de um componente proposto pelo IBSM (seção 4.1), o IBSNLT (seção 4.1.2) que atua na etapa de *definição* de serviço de acordo com o IBSPM. Ela aplica técnicas de NLP para a tradução da descrição informal para uma representação formal que é inequívoca e mais facilmente interpretável por sistemas computacionais.

A NSLE pode inferir políticas, funções e filtros de um serviço com base em intenções declarativas expressas em linguagem natural. Essa representação pode se basear em texto livre ou conjunto de afirmações (*statements*). A base de conhecimento pode ser construída por meio de uma aprendizagem guiada ou através da análise de serviços na base que contenham descrições. O aspecto de análise de *feedback* pode ser realizado em uma abordagem similar à empregada em outras SLEs através da avaliação de serviços de acordo com os requisitos definidos.

5.1.3.6 BSLE - *Behavior Self-Learning Entity*

A *Behavior Self-Learning Entity* (BSLE) – Entidade de Auto-Aprendizado para Detecção Comportamentos – é responsável pela detecção padrões de uso da rede e identificação dos tipos de conteúdo sendo trafegados de maneira a viabilizar a utilização de serviços sob-demanda inferidos por comportamentos. Esta entidade é uma versão estendida

de um componente de detecção de comportamentos IBSBD (seção 4.1.2) proposto pelo IBSM (seção 4.1) e que atua na etapa de *definição* de serviço segundo o IBSPM.

A detecção de padrões ocorre de maneira similar à utilizada pela DSLE, mas no caso da BSLE não são identificados problemas e sim padrões de utilização particularmente úteis na configuração de serviços (desempenhada pela SCE seção 5.1.2.1). Esses padrões também servem para ativar serviços específicos configurados de acordo com variáveis contextuais como por exemplo a média de utilização, horário e tipo de conteúdo.

A BSLE é responsável por identificar os tipos de e por inferir as intenções de uma comunicação pode meio de análise comportamental, o que é feito através do processamento de amostras do *payload*, campos de cabeçalhos e métricas diversas. A detecção de conteúdos pode se dar pela identificação de padrões, por exemplo, conteúdos multimídia normalmente são transmitidos com RTP sobre UDP, ou, sempre que por 'gatilhos', por exemplo, sempre que for detectada uma mensagem RTSP com o método PLAY deve ser avaliada informação de SDP para identificação do canal e configuração automática do serviço.

5.1.3.7 OSLE - *Orchestration Self-Learning Entity*

A *Orchestration Self-Learning Entity* (OSLE) – Entidade de Auto-Aprendizado para Orquestração – é responsável pela orquestração dos fluxos de organização da rede através de uma análise estatística sobre quais as melhores tratativas pra cada diagnóstico. De maneira geral essa entidade atua de forma a transformar eventos genéricos (e.g. *link* congestionado) em eventos mais específicos (e.g. *link* sobrecarregado por um alto número de serviços simultâneos) guiando assim a execução das SOE.

A OSLE atua de maneira complementar à entidade de diagnósticos (DSLE) e tem uma integração com o *broker* de serviços para atuar sempre que duas entidades se inscrevem no mesmo tópico. Cabe a OSLE interceptar estas mensagens e decidir sobre quais entidades receberão o evento para que iniciem os seus fluxos de execução.

Em um cenário ideal os diagnósticos realizados pela DSLE são precisos e cada SOE só se inscreve em tópicos individuais, o que dispensaria o uso de uma entidade específica de orquestração como a OSLE. Neste caso nos referimos às SOEs como entidades de organização auto-orquestradas.

5.1.4 *Collecting Entities* (CoEs)

As *Collecting Entities* (CoEs) – Entidades Coletoras – são entidades responsáveis pela coleta de dados oriundos dos recursos de infraestrutura e componentes de controle e gerenciamento. Elas são essenciais para a suporte da rede à propriedade de auto-consciência (discutida na seção 2.2.3) e cumprem os princípios definidos pela 'Subcamada de Coleta' do SONeM. Exemplos de dados coletados incluem *logs* de aplicações/dispositivos, amos-

tras de comunicação, informações de topologia, resultados de testes, métricas e alarmes. Esses dados são coletados em diversos níveis de abstração, tais como: dispositivos, portas, fluxos, serviços e redes. De maneira geral essas entidades são responsáveis por prover insumo para a etapa de *validação* proposta pelo IBSM (seção 4.1).

As CoEs são responsáveis não apenas pela coleta mas também pela primeira análise dos dados coletados e pela notificação dos componentes de gerenciamento através de eventos. Além de realizar notificações relacionadas à coleta dos dados, as CoEs podem, por exemplo, notificar as demais entidades quando a utilização de um *link* alcançar os marcos específicos como 0%, 25%, 50%, 75%, 90%, 95% e 100%. Isso diminui o tempo de reação dos componentes de gerenciamento às mudanças de estado através de uma análise de 'nível 1' realizada de maneira antecipada na própria entidade de coleta.

A implementação das CoEs varia de acordo com as características dos recursos e componentes que eles suportam. Pode-se definir, por exemplo, CoEs específica pra coleta de dados de equipamentos por marcas, modelos, ano de fabricação, tecnologias e versões de *software*. Os coletores também podem ser especializados em tecnologias/protocolos específicos suportados pelos recursos/componentes, por exemplo, pode-se utilizar SNMPv2 com suporte à IF-MIB ou um robô de *software* via CLI para coleta de dados com comandos específicos de acordo com a necessidade;

A Figura 18 apresenta as entidades coletoras propostas pelo SONAr:

- TCoE: responsável pela coleta de informações de topologia como dispositivos e links disponíveis (seção 5.1.4.1);
- MCoE: responsável pela coleta de métricas de recursos como dispositivos, portas, servidores, e dos componentes de controle e gerenciamento (seção 5.1.4.2);
- ACoE: responsável pela coleta de alarmes enviados pelos recursos e componentes (seção 5.1.4.3);
- SCoE: responsável pela coleta de amostras de fluxos de comunicação através de inspeção de conteúdo e análise de cabeçalhos (seção 5.1.4.4);
- LCoE: responsável pela coleta (ou recepção) e indexação de *logs* de todos os recursos e componentes (seção 5.1.4.5);

5.1.4.1 TCoE - *Topology Collector Entity*

A *Topology Collector Entity* (TCoE) – Entidade Coletora de Informações de Topologia – é responsável pela auto-consciência da rede em relação a sua topologia. Essa entidade provê informações sobre quais recursos (e.g. dispositivos, servidores...) estão disponíveis e sobre como eles estão conectados. Essa entidade é fundamental para detectar novos recursos, e assim viabilizar o procedimento de *plug-and-play* provido pela SCE (seção

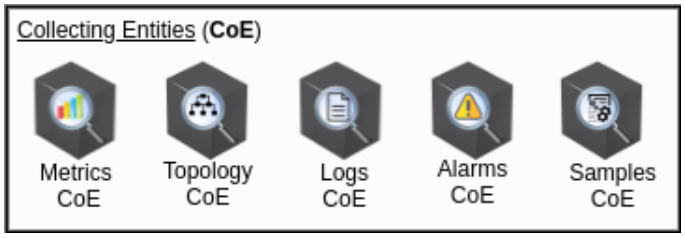


Figura 18 – Representação das Entidades Coletoras da SONAr.

Fonte: Elaborado pelo autor (2021).

5.1.2.1), e também para detectar a indisponibilidade de recursos, o que pode envolver um procedimento de auto-cura desempenhado pela SHE (seção 5.1.2.2).

Em geral o funcionamento da TCoE envolve a utilização de algoritmos e protocolos de descoberta que verificam a rede periodicamente em busca de mudanças. A TCoE é responsável pela atualização das informações de topologia (via NDB – seção 5.1.5.1), e pela notificação das demais componentes de gerenciamento por eventos (via NEM – seção 5.1.5.2).

5.1.4.2 MCoE - *Metrics Collector Entity*

A *Metrics Collector Entity* (MCoE) – Entidade Coletora de Métricas – é responsável pela auto-consciência da rede sobre o seu estado através de valores numéricos ou percentuais coletados dos recursos e componentes. Essa entidade monitora características variáveis tais como utilização, capacidade, memória e processamento. Essas informações são avaliadas pela MCoE que notifica as demais componentes de gerenciamento quando marcos forem alcançados ou quando valores sofrerem alterações acima de um limite. As métricas podem ser coletadas tanto de recursos físicos (e.g. dispositivos, servidores, links...) quanto lógicos (e.g. fluxos, serviços...).

A MCoE é uma entidade fundamental para o funcionamento da rede, pois a visão de estado que ela provê é utilizada por todas as demais entidades. Ela é responsável pela atualização das métricas (via NDB – seção 5.1.5.1), e pela notificação das demais componentes de gerenciamento por eventos (via NEM – seção 5.1.5.2).

Em geral o funcionamento da MCoE envolve a utilização de algoritmos e protocolos de coleta (ativo) ou recepção de métricas (passivo). Em abordagens ativas a coleta de métricas ocorre de maneira similar à coleta de informações de topologia. Os valores são consultados periodicamente através de técnicas suportadas pelos recursos. Em abordagens passivas são necessárias entidades com suporte a protocolos específicos (como sFlow e IPFIX) e recursos configurados com endereços alcançáveis.

5.1.4.3 ACoE - *Alarms Collector Entity*

A *Alarms Collector Entity* (ACoE) – Entidade Coletora de Alarmes – é responsável pela auto-consciência da rede sobre o seu estado através da coleta de alarmes que atuam como "sintomas" informados por agentes externos. Alarmes são eventos publicados pelos próprios recursos ou por componentes de controle utilizados para alertar sobre situações anormais.

Depois da coleta de dados, a ACoE realiza procedimentos para a sincronização e correlação de alarmes 'nível 1'. A sincronização é necessária pois alarmes representam um estado momentâneo e perdem sua relevância com o passar do tempo. Assim, essa entidade precisa descartar alarmes irrelevantes ou antigos. Além disso, a ACoE remove duplicatas e agrupa alarmes claramente relacionados (referentes ao mesmo recurso por exemplo).

A coleta de alarmes pode ocorrer de maneira ativa ou passiva, assim como a coleta de métricas. Após o processamento (sincronização, filtro e correlação), a ACoE armazena as informações (via NDB – seção 5.1.5.1), e notifica demais componentes de gerenciamento por eventos (via NEM – seção 5.1.5.2).

5.1.4.4 SCoE - *Samples Collector Entity*

A *Samples Collector Entity* (SCoE) – Entidade Coletora de Amostras – é responsável pela auto-consciência da rede em relação aos dados trafegados através da coleta de amostras. Essas amostras são úteis para detectar usos indevidos e para ativar serviços específicos de acordo com as necessidades dos usuários. Amostras são informações referentes às primitivas transmitidas pela rede, e se dividem em amostras de cabeçalho e conteúdo.

As amostras fornecem uma visão atual sobre o conteúdo e auxiliam na identificação da "intenção" da comunicação. Elas flexibilizam a implantação de serviços programados para contextos específicos relacionados aos tipos de conteúdo. Além disso elas fornecem informações úteis para o aprendizado de rede e ajudam na detecção de anomalias, ataques e usos indevidos de recursos.

A coleta de amostras se baseia no emprego de estratégias de *Deep Packet Inspection* (DPI), espelhamento de porta e *sniffing* de rede. As informações coletadas são armazenadas (via NDB – seção 5.1.5.1) e enviadas para as demais componentes de gerenciamento por eventos (via NEM – seção 5.1.5.2).

5.1.4.5 LCoE - *Logs Collector Entity*

A *Logs Collector Entity* (LCoE) – Entidade Coletora de *Logs* – é responsável pela auto-consciência da rede através da coleta, indexação e processamento de *logs* provenientes de seus recursos e componentes. A utilização de *logs* é uma estratégia comum em sistemas de computação para o armazenamento de informações sobre a execução dos componentes de

software. Eles podem ser implementados com diferentes níveis (e.g. ERROR, DEBUG, INFO, WARNING...), de forma estruturada ou em linguagem natural, e com base em arquivos locais ou servidores de *logs* remotos. Essas informações são uteis para rastrear situações de exceção e para detectar fluxos de execução.

A coleta de *logs* pode ocorrer de forma ativa ou passiva (assim como em outras CoEs). De maneira ativa a entidade utiliza serviços de consulta, protocolos de transferência de arquivos e monitores para detectar novos *logs* disponíveis. De maneira passiva os recursos e componentes são configurados para a utilização da entidade como um servidor de *logs* remoto. Depois de coletados *logs* a LCoE realiza um procedimento de indexação para facilitar a consulta e viabilizar o processamento pelas demais entidades. Na etapa de tratamento 'nível 1' são buscados padrões pré-definidos nos *logs* que podem ser avaliados pelas SLEs como sintomas de problemas de rede.

A LCoE armazena os *logs* coletados temporariamente (via NDB – seção 5.1.5.1) e notifica as demais entidades quanto um padrão é detectado (via NEM – seção 5.1.5.2).

5.1.5 Componentes de Gerenciamento Auxiliares

O SONAr define componentes de gerenciamento que auxiliam as entidades auto-organizadoras, coletoras e de auto-aprendizado a desempenharem os seus papéis na auto-organização da rede. A Figura 19 apresenta os componentes auxiliares propostos pelo SONAr:

- NDB: responsável pelo armazenamento dos dados de rede (seção 5.1.5.1);
- NEM: responsável pela integração dos componentes de gerenciamento por meio de troca de eventos e procedimentos remotos (seção 5.1.5.2);
- NSB: responsável por permitir a implantação de serviços de comunicação na rede pelos usuários e administradores (seção 5.1.5.3);
- NAD: responsável por prover uma interface administrativa em alto nível para a rede (seção 5.1.5.4);
- NIM: responsável por prover interfacear a comunicação com recursos de infraestrutura e componentes de controle; (seção 5.1.5.5);
- ABM: responsável por inicializar os componentes de gerenciamento e controle (seção 5.1.5.6);
- CPI: responsável pela interceptação da comunicação entre as camadas de infraestrutura e de controle (seção 5.1.5.7);

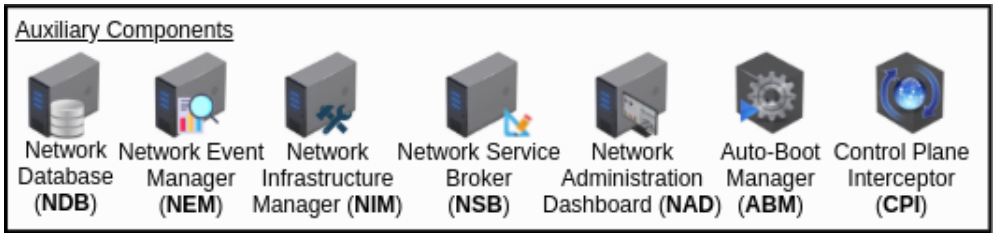


Figura 19 – Representação dos Componentes Auxiliares da SONAr.

Fonte: Elaborado pelo autor (2021).

5.1.5.1 NDB - *Network Database*

A *Network Database* (NDB) – Base de Dados de Rede – armazena todas as informações disponíveis rede, tais como: dados coletados pelas CoEs (seção 5.1.4); parâmetros e eventos usados pelos componentes de controle e gerenciamento; serviços de comunicação e suas configurações; bases de conhecimento e informações geradas pelas SLEs (seção 5.1.3); e, mensagens interceptadas pelo CPI (seção 5.1.5.7).

A NDB é distribuída para garantir alta disponibilidade e escalabilidade. Além disso, o tempo de resposta para consultas deve ser baixo de maneira a melhorar o tempo de reação das entidades SONAr, sobretudo as que atuam nas operações de cura e proteção. Pode-se também combinar bancos relacionais e não relacionais (desde que distribuídos) para a abstração da NDB de acordo com tipo de dado o que pode impactar positivamente no desempenho e na confiabilidade das informações armazenadas.

5.1.5.2 NEM - *Network Event Manager*

O *Network Event Manager* (NEM) – Gerenciador de Eventos da Rede – fornece um mecanismos *publish/subscribe* e de RPC para integração dos componentes de gerenciamento. Ele provê o desacoplamento entre os componentes de gerenciamento, que embora tenham dependências entre si não conhecem detalhes sobre a implementação uns dos outros. Isso facilita a alocação e substituição de componentes conforme a necessidade.

Os eventos são categorizados em tópicos e podem ser processados de forma síncrona ou assíncrona através de uma fila de mensagens (persistentes ou não) e procedimentos remotos. Todos os componentes do SONAr utilizam eventos para se comunicar, solicitar serviços ou notificar mudanças de estados. A escolha sobre quais tópicos devem ser contemplados por cada entidade é um detalhe de implementação e varia de acordo com a disponibilidade das entidades de auto-aprendizado (seção 5.1.3), sobretudo a entidade de auto-orquestração (seção 5.1.3.7). Em um cenário com gerenciamento orquestrado pela OSLE essa entidade se inscreve nos tópicos pertinentes à auto-organização da rede de maneira geral, e de acordo com um fluxo de orquestração as entidades auto-organizadoras são acionadas. Por outro lado, em um cenário onde as próprias entidades tratam a concorrência, todas elas se inscrevem nos tópicos pertinentes para a sua execução.

O funcionamento de boa parte das entidades de auto-aprendizado consiste em analisar e publicar eventos de alto nível com os resultados do processamento. Assim, entidades auto-organizadoras podem se inscrever em tópicos alimentados pelas entidades de auto-aprendizado para operar a rede.

5.1.5.3 NSB - *Network Service Broker*

O *Network Service Broker* (NSB) – *Broker* de Serviços da Rede – fornece uma interface para o gerenciamento dos serviços de comunicação implantados na rede. Trata-se de um sistema utilizado por usuários e administradores para criação, remoção, alteração, bloqueio e desbloqueio de serviços. Esse componente é o principal ponto de integração do auto-gerenciamento com agentes externos e permite implantar comportamentos específicos na rede com um alto nível de abstração.

O NSB é responsável por acionar a NSLE (seção 5.1.3.5) para interpretação dos serviços baseados em intenções descritas em linguagem natural e pela notificação da SCE (seção 5.1.2.1) para configuração da rede. Erros no processamento como falha na tradução, indisponibilidade de recursos ou erro de configuração são notificados aos administradores pelo NAD (seção 5.1.5.4).

O NSB opera com serviços pré-configurados. Serviços sob-demanda inferidos por comportamento de rede são tratados diretamente pela SCE através de tópicos publicados pela BSLE (seção 5.1.3.6) e mensagens interceptadas pelo CPI (seção 5.1.5.7). A ativação ou desativação de serviços programados ocorre de maneira automática (sem intervenção do NSB) através da ação da SCE.

5.1.5.4 NAD - *Network Administration Dashboard*

O *Network Administration Dashboard* (NAD) – *Dashboard* de Administração da Rede – é uma ferramenta administrativa utilizada para interagir com a camada de gerenciamento que permite:

- ❏ visualizar dados de rede: métricas, *logs*, alarmes, amostras informações de topologia e resultados de testes;
- ❏ gerenciar serviços: interface com o NSB para criação, remoção, alteração, bloqueio e desbloqueio de serviços;
- ❏ definir parâmetros e objetivos: definir os valores de parâmetros utilizados pelos componentes de gerenciamento e os objetivos corporativos de alto nível como redução de custos e restrições de segurança;
- ❏ guiar a construção de bases de conhecimento: auxilia no aprendizado guiado das SLEs;

- ❏ monitorar eventos: acompanhar eventos trocados entre os componentes através de informações sobre tópicos, conteúdos, emissores e receptores;

- ❏ monitorar decisões tomadas pelas SLEs: avaliar, confirmar ou bloquear diagnósticos, previsões, traduções, avaliações, otimizações, inferências e orquestrações realizadas via IA;

- ❏ realizar diagnósticos manuais: correlação de dados ou intervenção manual para o diagnóstico de problemas/oportunidades e construção guiada da base de conhecimento da DSLE;

- ❏ acompanhamento dos níveis de serviço: monitoramento dos serviços implantados e acompanhamento de impactos de problemas;

5.1.5.5 NIM - *Network Infrastructure Manager*

O *Network Infrastructure Manager* (NIM) – Gerenciador da Infraestrutura de Rede – é o componente responsável por abstrair as diferentes tecnologias suportadas pelos recursos de infraestrutura e componentes de controle através de uma interface padronizada para as demais entidades SONAr. O NIM provê serviços genéricos, tais como: *configurarFluxo* – traduzido em configurações práticas de acordo com as tecnologias disponíveis (e.g. *OpenFlow*, MPLS, *vlan...*); e *configurarBandaGarantida* – gera configurações de *policies* de acordo com o OS, modelo e fabricantes dos equipamentos (e.g. *Cisco ASR 1000 – IOS XE Gibraltar 16.12.1a*, *Juniper MX480 – JUNOS Software Version 15.1R7-S8...*).

O NIM pode ser abstraído como uma API e não necessariamente como um componente de fato, o que permite um melhor desempenho das entidades na integração com recursos de infraestrutura e componentes de controle; e, melhora a disponibilidade devido a eliminação de um ponto de falha. Independente da abordagem, acredita-se que a utilização do NIM como uma representação direta da SBI flexibiliza ainda mais a SONAr para atuação em redes dos mais variados tipos, inclusive nas redes legadas.

5.1.5.6 ABM - *Auto-Boot Manager*

O *Auto-Boot Manager* (ABM) – Gerenciador de Auto-Inicialização da Rede – é uma solução responsável por instanciar, configurar e inicializar componentes de controle e gerenciamento através de ações como: alocação de recursos; instanciação de funções virtualizadas/conteinerizadas; instalação de aplicações; configuração inicial de recursos; e, estabelecimento de fluxos de controle e gerenciamento.

O ABM é responsável pela inicialização dos componentes básicos do SONAr no servidor local de maneira a possibilitar a descoberta de recursos e distribuição de componentes realizadas respectivamente pelas entidades TCoE (seção 5.1.4.1) e SCE (seção 5.1.2.1).

Este componente pode ser implementado com um comportamento padrão ou pode seguir um plano de inicialização (com descrição de recursos e instruções de distribuição).

5.1.5.7 CPI - *Control Plane Interceptor*

O *Control Plane Interceptor* (CPI) – Interceptador do Plano de Controle – é um componente que funciona como um *proxy* entre o plano de dados e o plano de controle. As primitivas de controle são capturadas e analisadas pelo interceptador, e repassadas para processamento pelas entidades SONAr por meio de eventos (via NEM). Certas primitivas configuradas no CPI são especialmente úteis para os componentes de gerenciamento, como é o caso de mensagens do tipo PACKET_IN do protocolo *OpenFlow* que são úteis para funções de *plug-and-play* de recursos e configuração de serviços sob-demanda. Após o processamento o CPI pode encaminhar as mensagens interceptadas para o componente de controle ao qual ela é destinada, impedir que a mensagem continue ou encaminhar para outro componente de controle. Impedir que mensagens sejam processadas pelo plano de controle pode ser útil para evitar ataques ou impedir sobrecarga dos componentes de controle. O encaminhamento para outros destinatários é útil tanto para a auto-otimização por meio de balanceamento de carga, quanto para auto-cura por permitir a recuperação de um componente com problema de maneira imperceptível para os recursos de infraestrutura.

O CPI deve atuar de maneira transparente para os componentes de controle e recursos de infraestrutura. A ideia é que os objetos interceptados não tenham consciência da existência do interceptador. Para isso, o CPI deve ser implementado com uma estratégia de *transparent proxy*, ou deve assumir o endereço do componente de controle para os recursos de infraestrutura. Outra opção é encaminhar uma cópia das primitivas para o CPI através de regras de roteamento/fluxo implantadas nos dispositivos de comutação.

5.1.6 Relação entre as Entidades SONAr e os Modelos de Gerenciamento da Literatura

As entidades SONAr tratam do tema de gerenciamento de rede, o que foi explorado em detalhes na seção 2.1. Pode-se observar que há certas similaridades entre as funções de cada entidade e os aspectos de gerenciamento propostos por cada modelo/*framework* apresentado na seção 2.1.2, por este motivo a presente tese realiza uma análise para a correlação dos conceitos que é organizada na Tabela 6 na qual é marcado um 'x' quando a entidade se relaciona à um dos aspectos do Modelo/*Framework* em análise. São utilizados na comparação o FCAPS (*F: Fault, C: Configuration, A: Accounting, P: Performance* e *S: Security*); o OAM&P (*O: Operations, A: Administration, M: Maintenance* e *P: Provisioning*); e, o FAB (*F: Fulfillment, A: Assurance* e *B: Billing*).

		FCAPS (STEVENSON, 1995)					OAM&P (KU, 1998)				FAB (TMF, 1998)		
		F	C	A	P	S	O	A	M	P	F	A	B
SOEs	SCE		x							x	x		
	SHE	x					x					x	
	SPE					x	x					x	
	SOPE				x		x					x	x
SLEs	PSLE	x		x	x	x	x	x	x			x	
	DSLE	x		x	x	x	x	x	x			x	x
	ESLE		x		x			x	x	x	x		
	OSLE	x	x		x	x	x			x	x	x	
	TSLE				x			x				x	
	NSLE		x					x		x	x		
	BSLE		x							x	x		
CoEs	MCoE	x	x	x	x	x	x	x	x	x	x	x	x
	SCoE	x	x			x	x	x				x	
	LCoE	x				x	x	x	x			x	
	ACoE	x			x	x	x	x	x			x	
	TCoE	x	x				x	x		x	x	x	
Auxiliares	NDB	x	x	x	x	x	x	x	x	x	x	x	x
	NEM	x	x	x	x	x	x	x	x	x	x	x	x
	NIM	x	x	x	x	x	x	x	x	x	x	x	x
	NAD	x	x	x	x	x	x	x	x	x	x	x	x
	NSB		x					x		x	x		
	ABM		x							x	x		
	CPI	x	x			x	x				x	x	

Tabela 6 – Relação entre as SOEs e os principais Modelos/*Frameworks* de Gerenciamento.

Fonte: Elaborado pelo autor (2021).

5.1.6.1 FCAPS : *Fault, Configuration, Accounting, Performance and Security*

O aspecto *Fault* do FCAPS é a principal função da entidade de cura (SHE) na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de predição (PSLE), diagnóstico (DSLE) e orquestração OSLE. De maneira geral todas as entidades de coleta auxiliam de alguma maneira nesse aspecto: métricas (MCoE), amostras (SCoE), *logs* (LCoE), alarmes (ACoE) e topologia (TCoE). Dentre os componentes auxiliares atuam a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o *dashboard* de administração (NAD) e o interceptador do plano de controle (CPI).

O aspecto *Configuration* do FCAPS é a principal função da entidade de configuração (SCE) na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de avaliação de recursos/técnicas (ESLE),

orquestração (OSLE), detecção de comportamentos (BSLE) e tradução de serviços descritos em linguagem natural (NSLE). A entidade de coleta que trabalha de maneira mais direta na viabilização deste aspecto é a de topologia (TCoE), mas a de métricas (MCoE) pode afetar a avaliação dos recursos e a de amostras (SCoE) pode impactar na inferência de serviços. Dentre os componentes auxiliares atuam a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o barramento de serviços (NSB), gerenciador de *boot* automático (ABM), o *dashboard* de administração (NAD) e o interceptador do plano de controle (CPI).

O aspecto *Accounting* do FCAPS não é tratado diretamente na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de predição (PSLE) e diagnóstico (DSLE). A entidade de coleta de métricas (MCoE) é a que mais se relaciona com esse aspecto. Dentre os componentes auxiliares atuam a base de dados (NDB), o *broker* de eventos (NEM), o *dashboard* de administração (NAD) e o gerenciador de infraestrutura (NIM).

O aspecto *Performance* do FCAPS é a principal função da entidade de otimização (SOPE) na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de predição (PSLE), diagnóstico (DSLE), avaliação de recursos/técnicas (ESLE), otimização de parâmetros (TSLE) e orquestração (OSLE). As entidades de coletas mais relacionadas são as de métricas (MCoE) e alarmes (ACoE). Dentre os componentes auxiliares atuam a base de dados (NDB), o *broker* de eventos (NEM), o *dashboard* de administração (NAD) e o gerenciador de infraestrutura (NIM).

O aspecto *Security* do FCAPS é a principal função da entidade de proteção (SPE) na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de predição (PSLE), diagnóstico (DSLE) e orquestração OSLE. Quase todas as entidades de coleta são úteis neste aspecto: métricas (MCoE), amostras (SCoE), *logs* (LCoE) e alarmes (ACoE). Dentre os componentes auxiliares atuam a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o *dashboard* de administração (NAD) e o interceptador do plano de controle (CPI).

5.1.6.2 OAM&P : *Operations, Administration, Maintenance & Provisioning*

O aspecto *Operations* do OAM&P está muito relacionado às entidades de cura (SHE), otimização (SOPE) e proteção (SPE) no que se refere à organização da rede. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de predição (PSLE), diagnóstico (DSLE) e orquestração OSLE. De maneira geral todas as entidades de coleta auxiliam de alguma maneira nesse aspecto: métricas (MCoE), amostras (SCoE), *logs* (LCoE), alarmes (ACoE) e topologia (TCoE). Dentre os componentes auxiliares atuam a

base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o *dashboard* de administração (NAD) e o interceptador do plano de controle (CPI).

O aspecto *Administration* do OAM&P não está diretamente relacionada à nenhuma entidade de organização. Em relação às entidades de análise pode-se observar algum nível de correlação com as de predição (PSLE), diagnóstico (DSLE), avaliação de recursos/-técnicas (ESLE), otimização de parâmetros (TSLE) e tradução de serviços descritos em linguagem natural (NSLE). De maneira geral todas as entidades de coleta auxiliam de alguma maneira nesse aspecto: métricas (MCoE), amostras (SCoE), *logs* (LCoE), alarmes (ACoE) e topologia (TCoE). Dentre os componentes auxiliares atuam a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o *dashboard* de administração (NAD) e o barramento de serviços (NSB).

O aspecto *Maintenance* do OAM&P também não está diretamente relacionada à nenhuma entidade de organização. Dentre as entidades de análise há uma forte relação com a entidade diagnóstico (DSLE), predição (PSLE) e avaliação de recursos/técnicas (ESLE). Dentre as entidades de coleta as que se relacionam de alguma forma com este aspecto são as de métricas (MCoE), alarmes (ACoE) e *logs* (LCoE). Dentre os componentes auxiliares atuam a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM) e o *dashboard* de administração (NAD).

O aspecto *Provisioning* do OAM&P está intimamente relacionado à entidade de configuração (SCE) na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de avaliação de recursos/técnicas (ESLE), orquestração (OSLE), detecção de comportamentos (BSLE) e tradução de serviços descritos em linguagem natural (NSLE). As entidades de coleta que trabalham na viabilização deste aspecto são as de topologia (TCoE) e de métricas (MCoE). Todos os componentes auxiliares se relacionam com este aspecto: a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o barramento de serviços (NSB), gerenciador de *boot* automático (ABM), o *dashboard* de administração (NAD) e o interceptador do plano de controle (CPI).

5.1.6.3 FAB : *Fulfillment, Assurance, Billing*

O aspecto *Fulfilment* do FAB está diretamente relacionado à entidade de configuração (SCE) na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de avaliação de recursos/técnicas (ESLE), orquestração (OSLE), detecção de comportamentos (BSLE) e tradução de serviços descritos em linguagem natural (NSLE). As entidades de coleta que trabalham na viabilização deste aspecto são as de topologia (TCoE) e de métricas (MCoE). Todos os componentes auxiliares se relacionam com este aspecto: a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o barramento de serviços (NSB), gerenciador de *boot* automático (ABM), o *dashboard* de administração (NAD) e o interceptador

do plano de controle (CPI).

O aspecto *Fulfilment* do FAB está diretamente relacionado à entidade de configuração (SCE) na subcamada de organização/intervenção. Dentre as entidades de análise as que mais se relacionam com este aspecto são as de avaliação de recursos/técnicas (ESLE), orquestração (OSLE), detecção de comportamentos (BSLE) e tradução de serviços descritos em linguagem natural (NSLE). As entidades de coleta que trabalham na viabilização deste aspecto são as de topologia (TCoE) e de métricas (MCoE). Todos os componentes auxiliares se relacionam com este aspecto: a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o barramento de serviços (NSB), gerenciador de *boot* automático (ABM), o *dashboard* de administração (NAD) e o interceptador do plano de controle (CPI).

O aspecto *Assurance* do FAB se relaciona em algum nível com as entidades de cura (SHE), proteção (SPE) e otimização (SOPE) no nível organizacional. Na subcamada de análise/aprendizado as entidades que mais se relacionam são as de diagnóstico (DSLE), predição (PSLE), orquestração (OSLE) e otimização de parâmetros (TSLE). De maneira geral todas as entidades de coleta estão relacionadas com este aspecto: métricas (MCoE), amostras (SCoE), *logs* (LCoE), alarmes (ACoE) e topologia (TCoE). Dentre os componentes auxiliares estão relacionados a base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM), o *dashboard* de administração (NAD) e o interceptador do plano de controle (CPI).

O aspecto *Billing* do FAB se relaciona em algum nível com a entidade de otimização que considera aspectos de limitação da utilização na definição do serviço. No que se refere à subcamada de análise/aprendizado a entidade de diagnóstico (DSLE) é a única que tem alguma relação com este aspecto, uma vez que a degradação do serviço pode impactar na cobrança. Dentre as entidades de coleta a que se relaciona de fato com este aspecto é a de métricas (MCoE). Dentre os componentes auxiliares os que apresentam algum nível de relação com esta propriedade são as base de dados (NDB), o *broker* de eventos (NEM), o gerenciador de infraestrutura (NIM) e o *dashboard* de administração (NAD).

5.1.7 Fluxos de Organização das Entidades SONAr

A SONAr pode ser aplicada na automação de diversas operações de gerenciamento, desta maneira a presente seção propõe fluxos de trabalhos de alto nível (simplificados e genéricos) para casos de uso práticos envolvendo as principais propriedades de sistemas autônomos no contexto das redes (seção 2.2.3) e as entidades auto-organizadoras (seção 5.1.2) propostas pelo SONAr. Os casos de uso apresentados são:

1. *Configuração* : *Plug-and-Play* de um Recurso de Infraestrutura (seção 5.1.7.1);
2. *Configuração* : Configuração de um *Intent-Based Service* (seção 5.1.7.2);
3. *Configuração* : Inicialização da 'rede' (componentes, fluxos e serviços) (seção 5.1.7.3);

4. *Cura* : Correção de um *Intent-Based Service* que não está de acordo com as políticas definidas (seção 5.1.7.4);
5. *Cura* : Restabelecimento de um componente de controle/gerenciamento com problema (seção 5.1.7.5);
6. *Cura* : Recuperação da rede com um dispositivo com problema (seção 5.1.7.6);
7. *Proteção* : Tratamento de suspeita (ou diagnóstico) de tentativa de ataque (seção 5.1.7.7); e,
8. *Otimização* : Otimização da configuração de recursos, componentes e serviços (seção 5.1.7.8).

5.1.7.1 *Configuração* : *Plug-and-Play* de um Recurso de Infraestrutura

A Figura 20 apresenta o fluxo básico de execução da SCE na implementação da funcionalidade de *plug-and-play* para recursos de rede. A primeira parte do trabalho da SCE está em prover recursos de endereçamento (e.g. IPs, vlans, *labels*...) e instruções básicas de configuração em uma abstração similar ao do *Bootstrap Protocol* (BOOTP). Depois a SCE precisa garantir que os recursos sejam acessíveis, o que é feito através da configuração de rotas/fluxos nos dispositivos de rede entre a entidade e o novo recurso. A identidade do novo recurso é verificada a partir da troca de certificados, e em caso de sucesso a SCE configura o novo recurso de acordo com tipo. Após a configuração o recurso se torna disponível para as outras entidades.

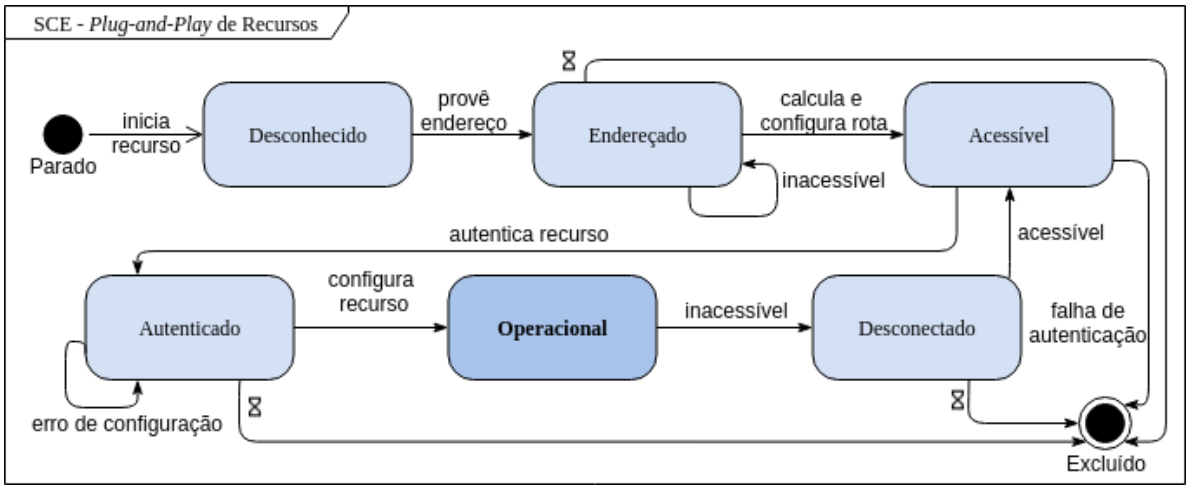


Figura 20 – Fluxo SCE para Implantação de Recursos.

Fonte: Elaborado pelo autor (2021).

5.1.7.2 *Configuração* : Configuração de um *Intent-Based Service*

A Figura 21 apresenta o fluxo básico de aprovisionamento de serviços de comunicação conforme especificado pelo IBSM (seção 4.1). Serviços pré-configurados e sob demanda

são tratados de maneira similar pela SCE. Caso eles sejam descritos com "intenções" em linguagem natural o primeiro passo é interpretá-los para identificar os requisitos comunicacionais. Depois a SCE precisa avaliar a rede e planejar o aprovisionamento do novo serviço a partir de um cálculo multiobjetivo que leva em consideração os requisitos definidos. Por fim, caso um plano de implantação seja possível o serviço é configurado e passa a ser tratado pelas outras entidades. Um serviço também pode ser alterado, e nesse caso, ele precisa passar por todo o fluxo de aprovisionamento novamente. Caso o serviço seja bloqueado a configuração é removida da rede, mas o planejamento é mantido. Depois do desbloqueio o plano de implantação é utilizado novamente para configurar o serviço.

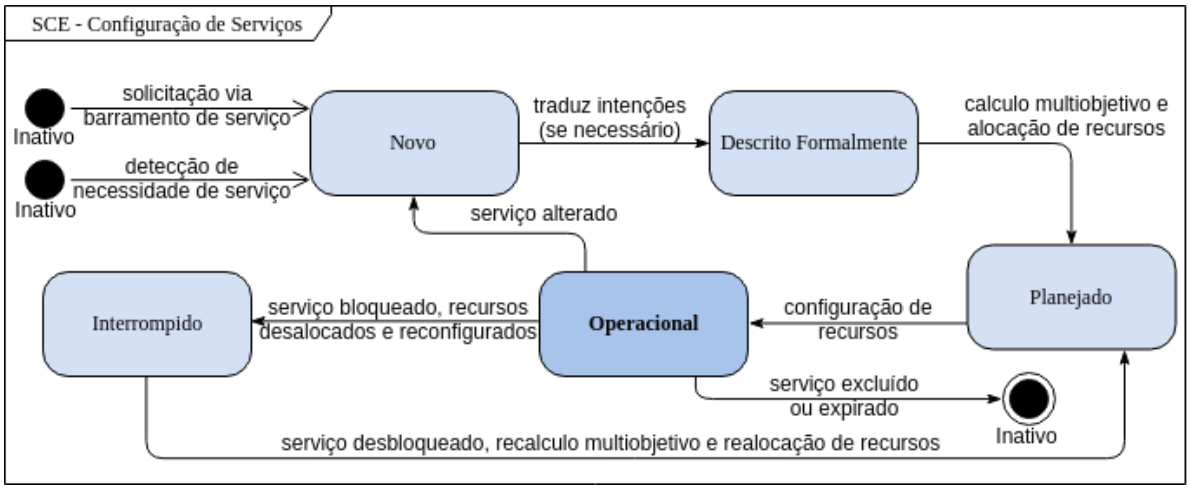


Figura 21 – Fluxo SCE para Aprovisionamento de Serviços.

Fonte: Elaborado pelo autor (2021).

5.1.7.3 *Configuração* : Inicialização da 'rede' (componentes, fluxos e serviços)

A Figura 22 apresenta um fluxo parcial da inicialização da rede sob a perspectiva da SCE. Esse fluxo inicia em um estado no qual os componentes SONAr estão funcionando e os recursos de rede já foram descobertos. A SCE realiza um planejamento de rotas/fluxos para o estabelecimento do plano de controle e de distribuição dos componentes SONAr. Caso o plano de implantação seja possível a SCE configura os dispositivos de maneira a prover conectividade pra rede, e distribui componentes entre os servidores disponíveis. Por fim, a SCE verifica o estado dos recursos, aplica configurações e implanta serviços de comunicação.

5.1.7.4 *Cura* : Correção de um *Intent-Based Service* que não está de acordo com as políticas definidas

A Figura 23 apresenta o fluxo básico de execução da SHE para o monitoramento e correção de serviços de comunicação. O fluxo inicia após a configuração do serviço. Quando

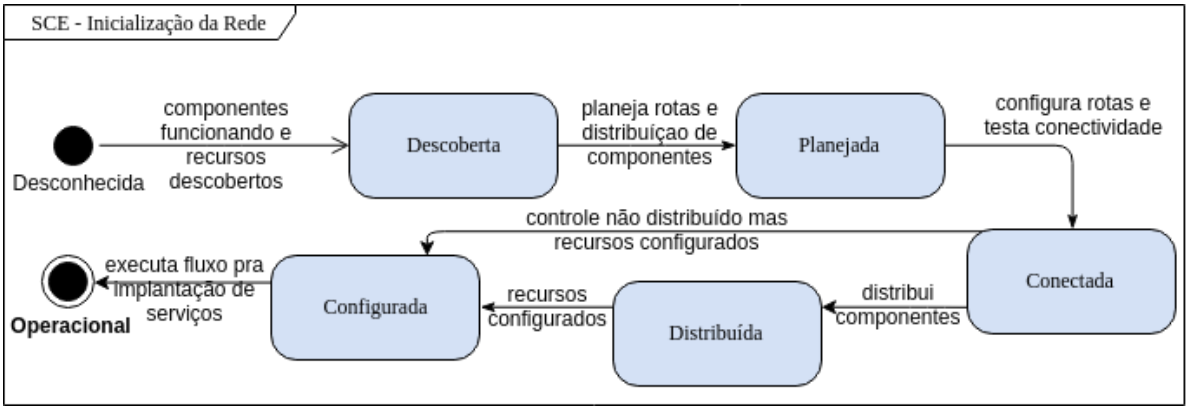


Figura 22 – Fluxo SCE para Inicialização da Rede.

Fonte: Elaborado pelo autor (2021).

é diagnosticada uma degradação dos níveis de um serviço a SHE realiza a reconfiguração desse serviço com novos recursos/fluxos e verifica a efetividade da tratativa. Caso os níveis acordados tenham se estabilizado o serviço é considerado operacional, e em caso contrário uma nova tentativa de reconfiguração é realizada. Caso um serviço esteja indisponível a SHE solicita a configuração pelo fluxo normal provido pela SCE.

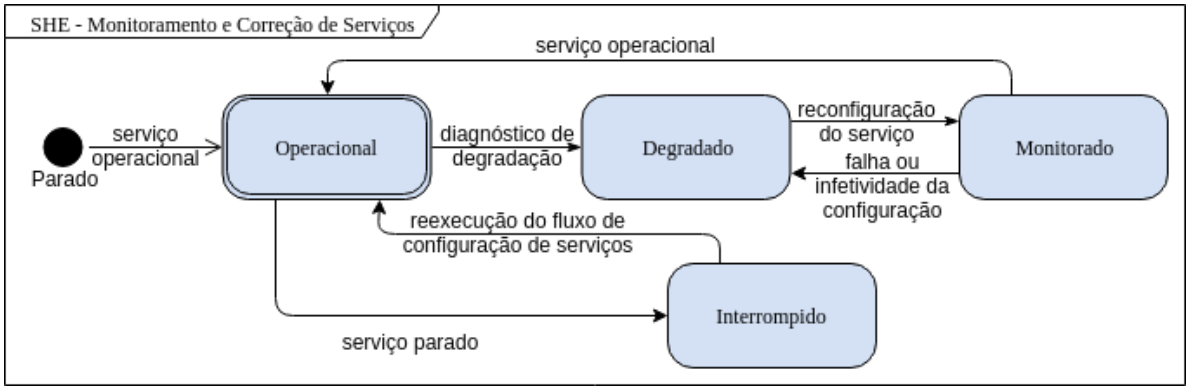


Figura 23 – Fluxo SHE para Monitoramento e Correção de Serviços.

Fonte: Elaborado pelo autor (2021).

5.1.7.5 *Cura* : Restabelecimento de um componente de controle/gerenciamento com problema

A Figura 24 apresenta o fluxo básico de monitoramento e restabelecimento de componentes. O fluxo inicia após a implantação, inicialização e configuração dos componentes. Quando é detectada a falha de um componente a SHE pode optar por duas abordagens: isolar o componente ou tentar recuperá-lo imediatamente. A primeira abordagem é mais segura uma vez que a SHE configura os demais componentes e recursos de maneira a ignorar o componente defeituoso e evitar o comprometimento das funções de controle e de gerenciamento. Nesse caso, após isolar o componente a SHE pode substituí-lo (a

partir da realocação e reimplantação) ou tentar recuperá-lo. Na segunda abordagem a recuperação é feita sem isolamento do componente, o que é uma solução mais simples e rápida. Em ambas as abordagens o procedimento de recuperação se baseia em técnicas de restauração, reconfiguração e reinicialização.

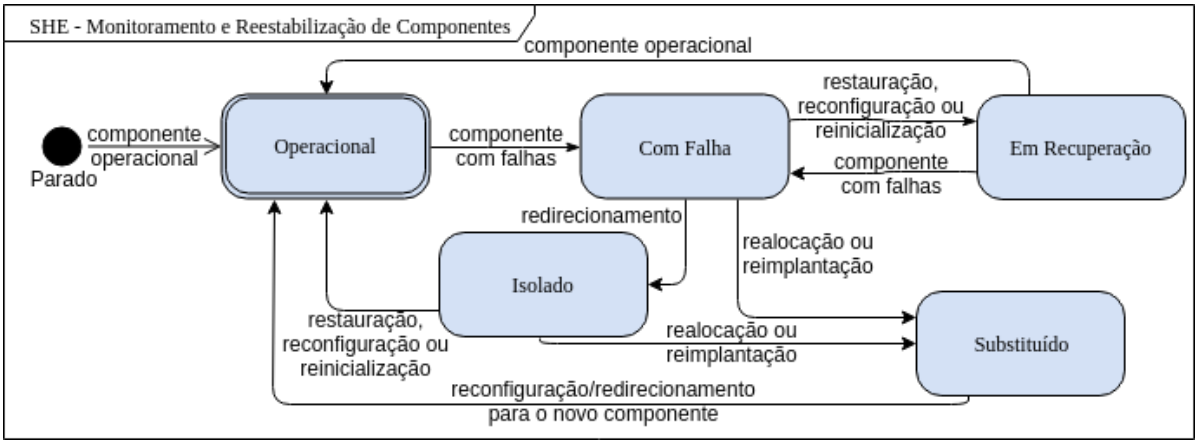


Figura 24 – Fluxo SHE para Monitoramento e Reestabilização de Componentes.

Fonte: Elaborado pelo autor (2021).

5.1.7.6 *Cura* : Recuperação da rede com um dispositivo com problema

A Figura 25 apresenta um fluxo para o monitoramento e reconfiguração de dispositivos. O fluxo inicia após os dispositivos se tornarem operacionais e o plano de controle ter sido estabelecido. A SHE pode lidar com três situações: dispositivo inacessível, com falhas ou com *link down*. Quando inacessível, a SHE deve isolar o dispositivo a partir da reconfiguração dos serviços e do plano de controle. Ao voltar ser acessível o dispositivo é reconfigurado pelo procedimento de *plug-and-play* de forma automática pela SCE. Quando o dispositivo está com falha a SHE também reconfigura os serviços e o plano de controle, mas o dispositivo entra em um estado de recuperação. O processo de recuperação de dispositivos é similar ao aplicado na recuperação de componentes: se baseia em restauração, reconfiguração e reinicialização do sistema. No caso de *link down* a SHE apenas reconfigura serviços e fluxos de controle caso estes utilizem a porta caída.

5.1.7.7 *Proteção* : Tratamento de suspeita (ou diagnóstico) de tentativa de ataque

A Figura 26 apresenta o fluxo básico de execução da SPE no tratamento reativo/preventivo de problemas de segurança. O fluxo se inicia quando a rede se torna operacional. Quando a SPE detecta um ataque em curso (eventualmente com degradação de rede) os recursos e componentes alvos do ataque são imediatamente isolados através de bloqueio de tráfego, desconfiguração ou até mesmo *shutdown* de recursos/componentes. Nesse estado

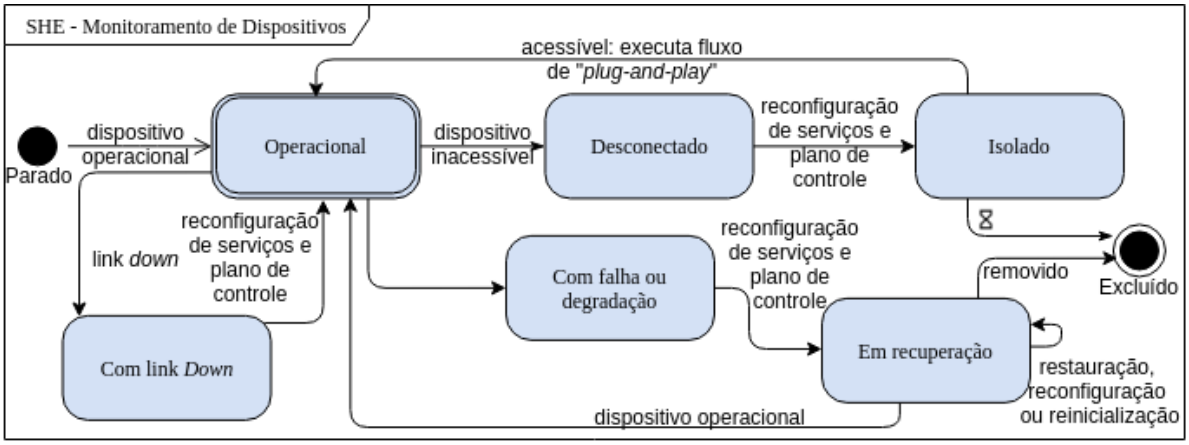


Figura 25 – Fluxo SHE para Monitoramento e Recuperação de Dispositivos.

Fonte: Elaborado pelo autor (2021).

a SPE inicia a tratativa para reestabelecimento do componente/recurso afetado até que a ameaça seja eliminada. Quando a SPE detecta apenas uma suspeita de ataque (sem degradação da rede) ele inicia um procedimento de recuperação dos componentes/recursos afetados de forma preventiva.

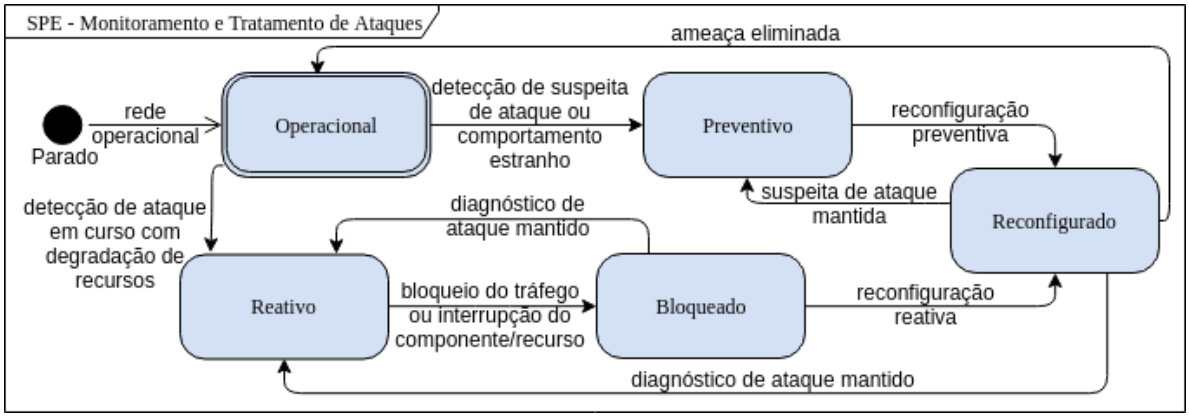


Figura 26 – Fluxo SPE para Monitoramento, Detecção e Tratamento de Ataques.

Fonte: Elaborado pelo autor (2021).

5.1.7.8 *Otimização* : Otimização da configuração de recursos, componentes e serviços

A Figura 27 apresenta o fluxo básico de execução da SOPE para a otimização de recursos, componentes e serviços. O fluxo inicia quando o alvo da otimização está operacional. Se detectada uma oportunidade de melhoria a SOPE realiza uma tentativa de reconfiguração de otimização. Caso a reconfiguração não surta efeito novas tentativas são realizadas até que não haja mais alternativas. Nesse caso a configuração original é restaurada. Caso alguma das reconfigurações demonstrem otimização o alvo é considerado otimizado.

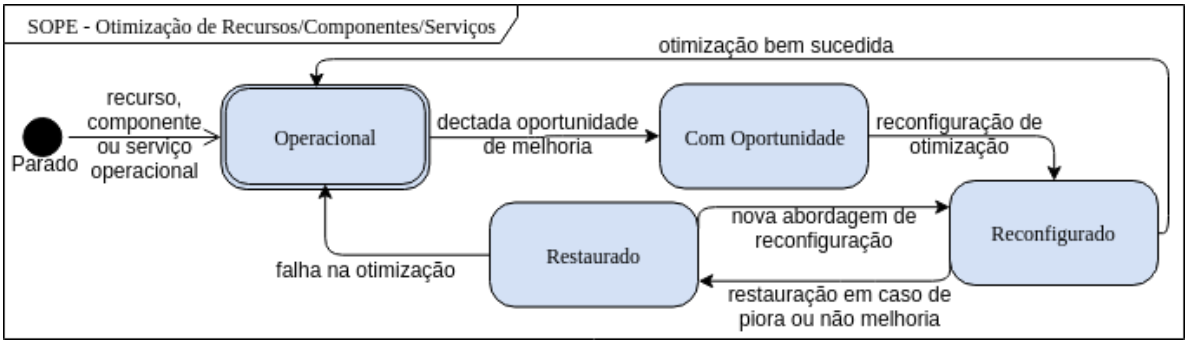


Figura 27 – Fluxo SOPE para otimização de Recursos, Componentes e Serviços.

Fonte: Elaborado pelo autor (2021).

5.2 Estudo de Caso: Auto-Configuração através da SONAr

A propriedade de auto-configuração sob o ponto de vista dos sistemas autônomos (seção 2.2.1) determina que um sistema deva se adaptar dinamicamente a mudanças ambientes de acordo com políticas administrativas e regras de negócio. Do ponto de vista das redes auto-organizadas (seção 2.2.3.1), essa propriedade está relacionada à capacidade da rede de se auto-inicializar, de aprovisionar serviços de comunicação conforme a necessidade e de suportar novos recursos e componentes de forma automática.

O IBSRM (proposto nesta tese na seção 4.1.1) é a parte do IBSM que propõe uma maneira flexível de representar serviços de comunicação baseados em intenções descritas por políticas (ou inferidas por comportamentos), e com ativação condicionada a definições contextuais mais amplas que incluem filtros pouco usuais como: estado da rede, momento e tipo de conteúdo. O IBSPM (proposto nesta tese na seção 4.1.2) é a parte do IBSM que propõe um estratégia de aprovisionamento de serviços que se baseia nas etapas de definição, interpretação, configuração, validação e manutenção. O IBSPM também define componentes conceituais responsáveis por cada etapa do fluxo de aprovisionamento e inclui a especificação de componentes adicionais responsáveis pela tradução de serviços descritos em linguagem natural e inferidos por comportamentos detectados na rede. O SONeM (proposto nesta tese na seção 4.2) propõe uma separação das funções de gerenciamento das camadas de controle e aplicação a partir de uma nova camada de gerenciamento responsável pela abstração de um fluxo cíclico de coleta de dados, análise de cenários e intervenção na rede.

Os modelos conceituais supracitados foram utilizados na concepção da SONAr (proposta nesta tese na seção 5.1), uma arquitetura para redes auto-organizadas que contempla propriedades de auto-configuração, auto-cura, auto-otimização, auto-proteção, auto-aprendizado e auto-consciência. Os aspectos de auto-configuração trabalhados nessa arquitetura ficam a cargo de dois componentes de maneira mais direta: o *Auto-Boot Ma-*

nager (seção 5.1.5.6), que é responsável pela inicialização e distribuição dos componentes de controle e gerenciamento, e a *Self-Configuration Entity* (seção 5.1.2.1), que provê a configuração de serviços, recursos e componentes.

A Figura 28 apresenta em destaque os componentes SONAr relacionados à propriedade de auto-configuração e contemplados no estudo de caso apresentado nesta seção. São eles:

- ❏ SCE (seção 5.1.2.1) – responsável pela configuração de recursos e componentes;
- ❏ ABM (seção 5.1.5.6) – responsável pela inicialização e distribuição dos componentes;
- ❏ ESLE (seção 5.1.3.4) – avaliação de recursos para cálculo multiobjetivo;
- ❏ NSLE (seção 5.1.3.5) – tradução de serviços descritos em linguagem natural;
- ❏ BSLE (seção 5.1.3.6) – detecção de comportamentos para serviços sob demanda;
- ❏ TCoE (seção 5.1.4.1) – descoberta da topologia e monitoramento de novos recursos;
- ❏ MCoE (seção 5.1.4.2) – coleta de métricas para avaliação de recursos;
- ❏ SCoE (seção 5.1.4.4) – coleta de amostras para detecção de comportamentos;
- ❏ NDB (seção 5.1.5.1) – armazenamento de topologia, serviços e configurações;
- ❏ NEM (seção 5.1.5.2) – troca de eventos entre os componentes;
- ❏ NSB (seção 5.1.5.3) – criação, alteração, remoção, bloqueio e desbloqueio de serviços;
- ❏ NIM (seção 5.1.5.5) – gestão e interface com os recursos de infraestrutura e componentes de controle;
- ❏ CPI (seção 5.1.5.7) – interceptação de mensagens para detecção de comportamento e inicialização da rede.

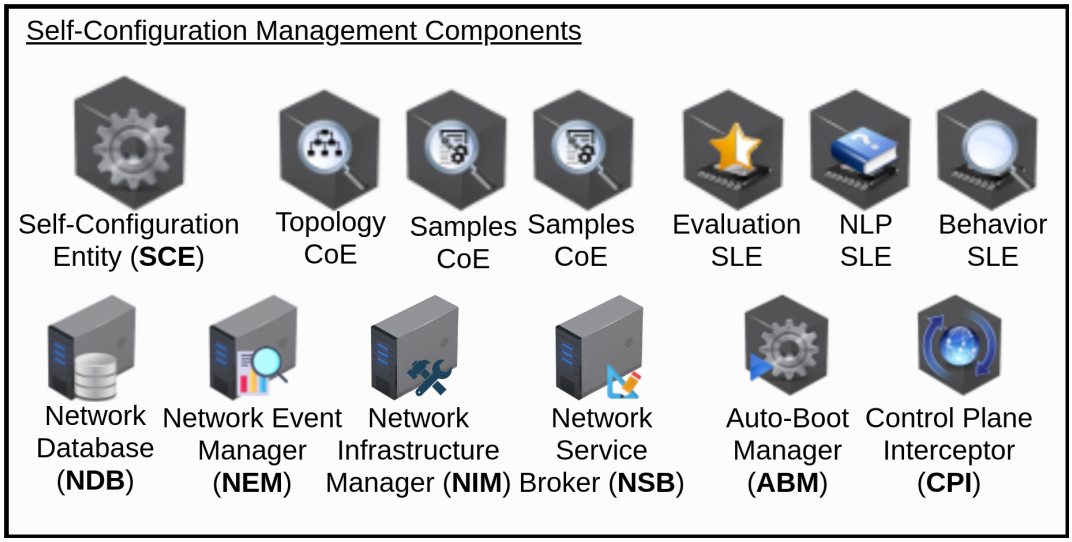


Figura 28 – Componentes SONAr relacionados à propriedade de auto-configuração.

Fonte: Elaborado pelo autor (2021).

Nas seções seguintes serão apresentados detalhes sobre uma proposta que automatiza operações de configuração da rede com base na arquitetura SONAr e o modelo IBSM.

Essa proposta é um estudo de caso para os conceitos defendidos nesta tese, e pretende demonstrar a viabilidade da solução na garantia da propriedade de auto-configuração.

5.2.1 Estudo de caso de Inicialização da Rede

A inicialização da rede, também conhecida como *bootstrapping*, é uma tarefa distribuída entre os componentes de gerenciamento/controle e recursos de infraestrutura que visa tonar a rede operacional. Nesse processo os recursos precisam ter autonomia para executar tarefas básicas de auto-configuração de maneira a se tornarem acessíveis antes mesmo que os componentes de gerenciamento entrem em ação. Quando acessíveis cabe à rede a tarefa de descobrir os recursos da infraestrutura e definir as melhores regras de fluxo/roteamento para prover controle e gerenciamento. Mas antes disso, os próprios componentes precisam ser instanciados, inicializados e configurados. Quando a rede detecta recursos que podem ser utilizados para a instanciação de novos elementos de controle e gerenciamento (servidores de contêineres ou de máquinas virtuais por exemplo), ela deve optar por utilizá-los ou não, e em caso afirmativo os novos componentes precisam ser aprovisionados e rotinas de distribuição precisam ser executadas (configuração de *endpoints*, reconfiguração de recursos/componentes e sincronização de dados).

O fluxo de inicialização da rede que já foi contemplado na seção 5.1.7.3, é apresentado de maneira mais detalhada nas figuras abaixo. A Figura 29 divide o processo de inicialização da rede em três subprocessos: inicialização dos componentes de gerenciamento e de controle, descoberta inicial e configuração de recursos e aprovisionamento inicial dos serviços de comunicação.

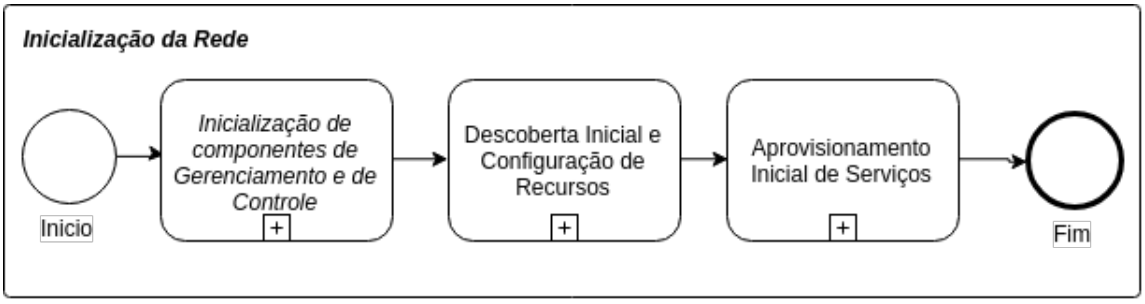


Figura 29 – Fluxo BPMN para Inicialização da Rede: Visão Geral.

Fonte: Elaborado pelo autor (2021).

5.2.1.1 Inicialização dos Componentes de Gerenciamento e de Controle

A *Inicialização dos Componentes de Gerenciamento e de Controle* é o processo responsável por verificar o estado dos componentes de controle e de gerenciamento; instanciá-los caso seja necessário; e, configurá-los de acordo com as políticas administrativas, características da infraestrutura e requisitos de distribuição.

A Figura 30 apresenta em detalhes esse processo que é desempenhado pelo ABM (seção 5.1.5.6). Ele executa o subprocesso de aprovisionamento de componentes simultaneamente para o NDB (seção 5.1.5.1) e NEM (seção 5.1.5.2), que são componentes básicos para o gerenciamento de rede com SONAr, e componentes diversos da camada de controle tais como gerências de equipamentos e controladores SDN. Além disso, o ABM aprovisiona componentes com funções de rede específicas como, por exemplo, servidores DHCP/BOOTP, o que é especialmente útil para a auto-configuração da rede. Depois que o NDB se torna operacional, o ABM cria a base e popula com dados iniciais caso necessário. Na segunda etapa do processo o ABM aprovisiona os demais componentes de gerenciamento utilizados na auto-configuração definidos no início da seção 5.2.

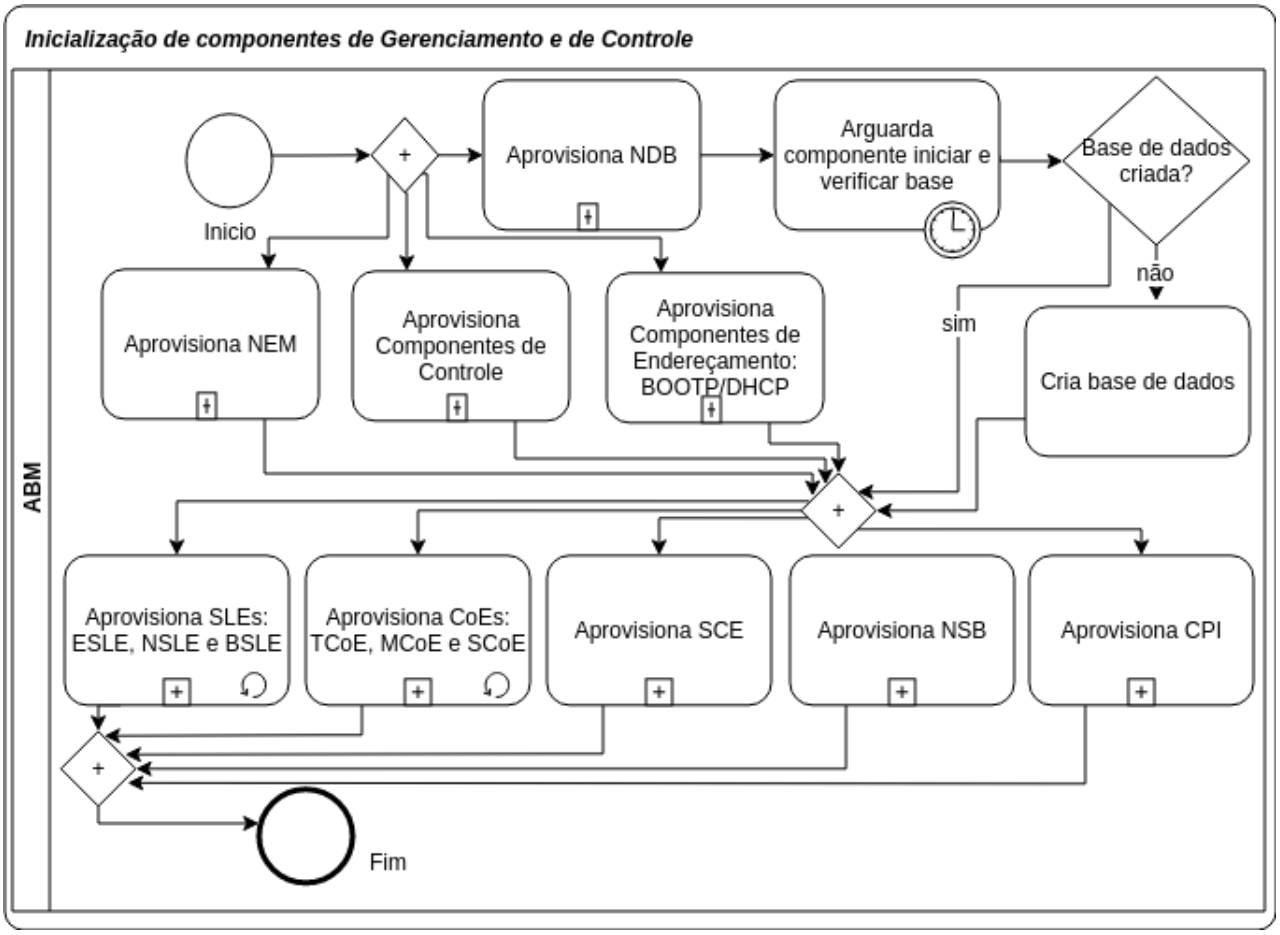


Figura 30 – Fluxo BPMN para Inicialização da Rede: Inicialização dos componentes de gerenciamento e de controle.

Fonte: Elaborado pelo autor (2021).

O subprocesso de *Aprovisionamento de Componentes* é apresentado na Figura 31, na qual pode ser observado que o ABM instancia o componente se ele não existir, inicializa se ele estiver parado, aplica uma configuração básica e registra o componente. A instanciação varia de acordo com as tecnologias utilizadas, por exemplo criação de um contêiner, instalação de um *software*, alocação de uma máquina virtual e etc.

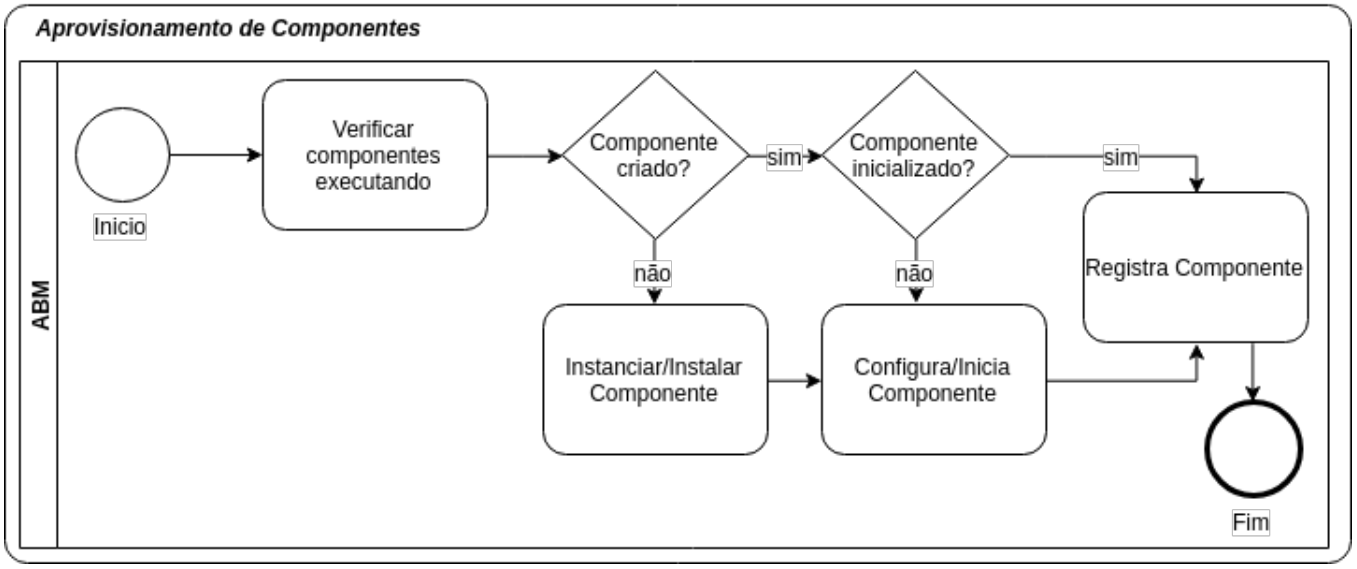


Figura 31 – Fluxo BPMN para Inicialização da Rede: Aprovisionamento de Componentes.

Fonte: Elaborado pelo autor (2021).

5.2.1.2 Descoberta inicial e configuração de recursos

A *Descoberta inicial e configuração de recursos* é o processo responsável por verificar descobrir os recursos disponíveis na infraestrutura, autorizá-los, configurá-los de acordo com o tipo, estabelecer rotas/fluxos de controle e instanciar novos componentes de maneira a distribuir o gerenciamento.

A Figura 32 apresenta em detalhes esse processo que se divide em duas etapas. A primeira é desempenhada pela TCoE (seção 5.1.4.1) que emprega mecanismos para a descoberta de recursos, como por exemplo: busca em largura a partir do servidor local; consulta de dados registrados nos componentes de controle; e, coleta de dados em recurso com endereço designado automaticamente pela rede. Depois de descoberta a topologia é validada de maneira a identificar inconsistências, e os recursos são autorizados para evitar ataques e restringir a utilização. Por fim, a TCoE notifica a rede sobre a conclusão da etapa de descoberta inicial de recursos.

A segunda etapa é desempenhada pelo SCE (seção 5.1.2.1) que emprega mecanismos para a configuração de recursos e distribuição dos componentes. A SCE aplica uma tratativa específica, de acordo com o tipo do recurso. Caso ele seja um dispositivo de comutação (*switch*, roteador...), ele é utilizado no estabelecimento dos fluxos de controle e gerenciamento; caso seja um servidor SONAr, a SCE decide, de acordo com regras definidas pelos administradores, sobre a distribuição do controle e gerenciamento (que é executado de fato pelo ABM); e caso ele seja outro tipo de recurso (e.g. *firewall*, *load-balancer*...), a SCE aplica uma configuração padrão de maneira a permitir a sua utilização no aprovisionamento de serviços futuros.

Na configuração dos dispositivos de comutação a SCE verifica as definições dos administradores e tecnologias suportadas pelo dispositivo para decidir sobre a necessidade de utilização de um componente de controle e um interceptor do plano de controle (CPI). Caso seja necessário, primeiro a SCE configura os dispositivos para se registrarem nesses componentes (controlador ou interceptor), e só depois aplica de fato as regras para o estabelecimento do fluxo de controle e gerenciamento. É importante observar que a SCE precisa obedecer a uma ordem específica na configuração dos dispositivos de maneira a manter a conectividade com todos os recursos durante todo o tempo.

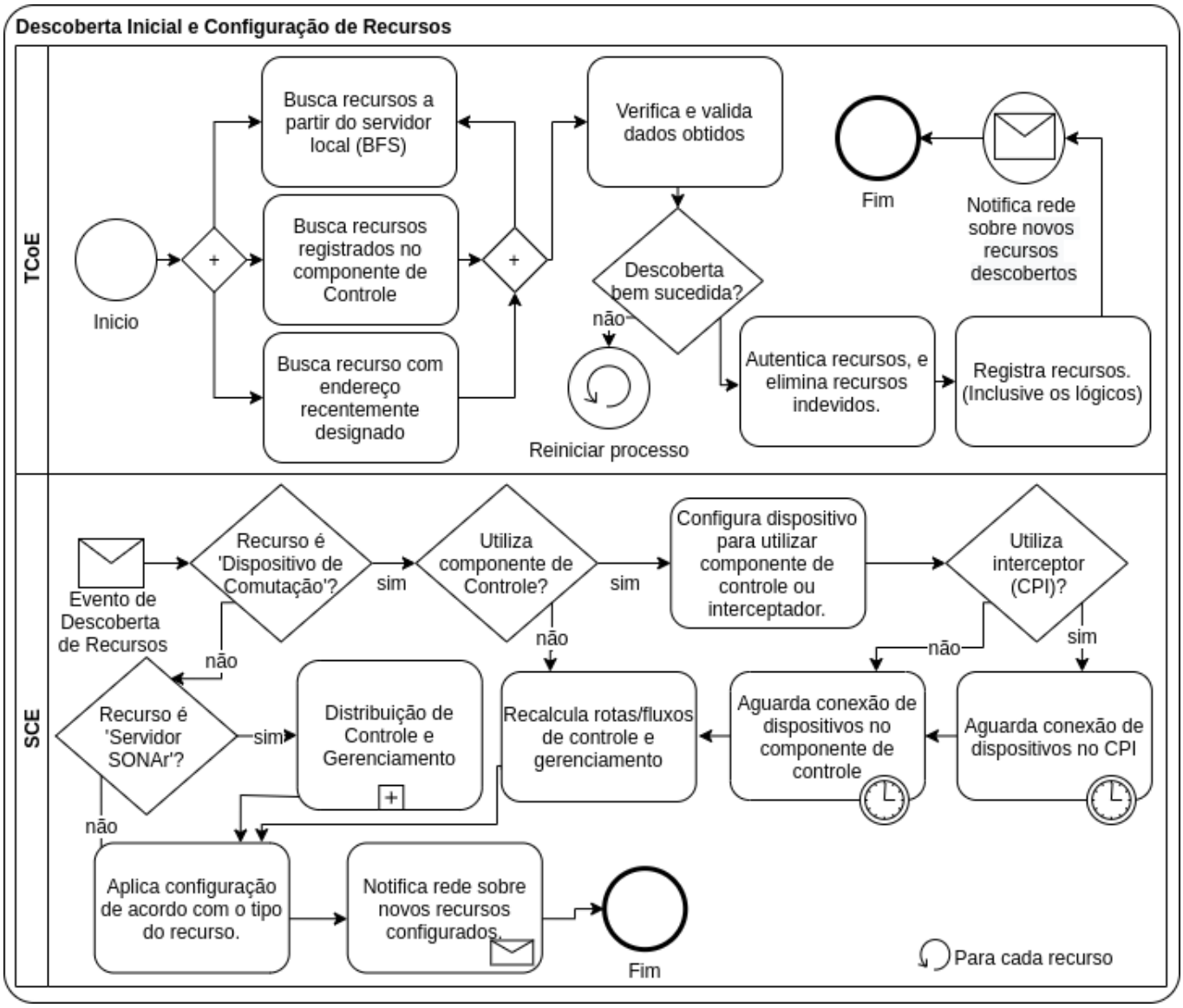


Figura 32 – Fluxo BPMN para Inicialização da Rede: Descoberta inicial e configuração de recursos.

Fonte: Elaborado pelo autor (2021).

O subprocesso de *Distribuição de Componentes de Controle e Gerenciamento* é apresentado na Figura 33, na qual pode ser observado que o ABM executa o processo de

inicialização dos componentes de gerenciamento e de controle (definido anteriormente) no novo servidor SONAr, sincroniza as bases de dados (NDB) e os gerenciadores de eventos (NEM), e reconfigura todos os componentes para reconhecerem a nova estrutura.

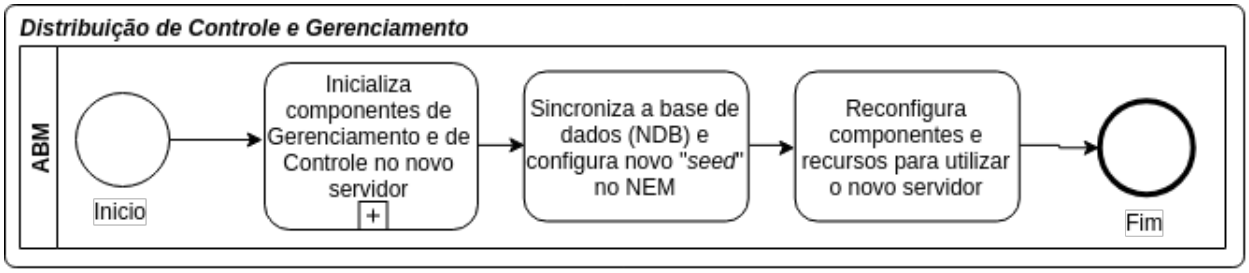


Figura 33 – Fluxo BPMN para Inicialização da Rede: Descoberta inicial e configuração de recursos.

Fonte: Elaborado pelo autor (2021).

5.2.1.3 Aprovisionamento Inicial de Serviços

O *Aprovisionamento inicial dos serviços* é o processo responsável pela configuração dos serviços de comunicação de acordo com políticas, funções e filtros conforme definido pelo IBSM (seção 4.1). Essa função é desempenhada pela SCE (conforme apresentado na Figura 34), que consulta os serviços registrados, elabora um plano de configuração considerando todos os serviços, identifica os serviços ativos de acordo com os filtros e aplica os planos de configuração na infraestrutura. Na implantação dos serviços a SCE pode implantar funções de rede virtualizada/conteinerizadas e/ou configurar os componentes de controle, o que é definido no planejamento da configuração.

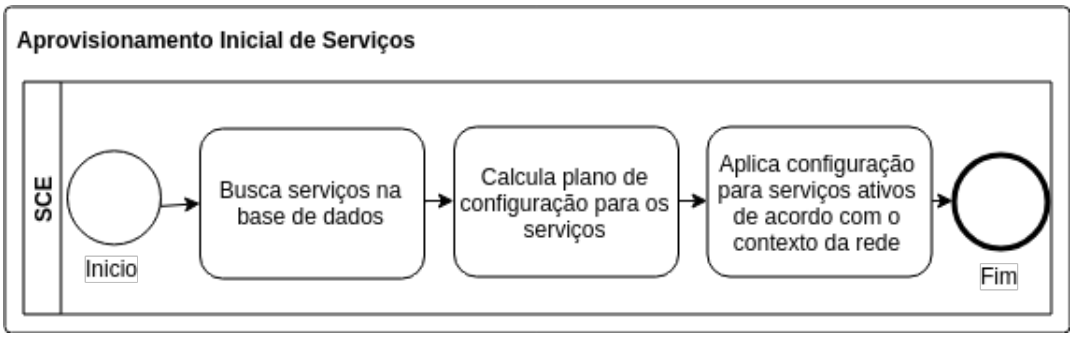


Figura 34 – Fluxo BPMN para Inicialização da Rede: Aprovisionamento inicial dos serviços registrados.

Fonte: Elaborado pelo autor (2021).

5.2.2 Estudo de Caso de *Plug-and-Play* de Recursos

O *Plug-and-Play de recursos* é o processo que viabiliza e extensão e retração automática da infraestrutura de rede, e assim como o processo de inicialização da rede é

essencialmente uma tarefa distribuída entre os recursos plugados e os componentes de controle e gerenciamento. Os recursos precisam se tornar acessíveis antes de serem efetivamente plugados na rede, por isso é necessário que eles se configurem de forma autônoma com instruções básicas de endereçamento e habilitação de funcionalidades. Mesmo depois de endereçados os recursos podem ainda não ser acessíveis, e por isso a rede executa uma tarefa para identificação do ponto de interconexão (porta na qual o recurso foi plugado), e configuração de rotas/fluxos específicas para acessar os novos recursos. De acordo com o tipo de recurso a rede define uma configuração padrão e registra as características do novo recurso para utilização posterior na melhoraria da qualidade dos serviços ou para a distribuição dos planos de controle/gerenciamento. Caso por exemplo o novo recurso identificado seja um *link*, a rede pode optar por redistribuir o tráfego de maneira a utilizar a nova rota.

O fluxo de *plug-and-play* de recursos já foi contemplado na seção 5.1.7.1, mas é apresentado de maneira mais detalhada na Figura 35 que define esse processo sob quatro perspectivas: *i)* na visão da entidade de auto-configuração (SCE – seção 5.1.2.1); *ii)* na visão da entidade coletora de informações de topologia (TCoE – seção 5.1.4.1); *iii)* na visão de um componente/função de rede para atribuição de endereço/configuração de forma automática (DHCP/BOOTP); e, *iv)* na visão do próprio recurso plugado.

5.2.2.1 *Plug-and-Play* na Perspectiva do Recurso de Infraestrutura

O processo apresentado na Figura 35 é responsável pelo *plug-and-play* de um recurso endereçável. Parte da responsabilidade dessa tarefa está no próprio recurso de infraestrutura que precisa se tornar acessível para a ação dos componentes de controle e gerenciamento. A primeira parte do fluxo é voltada para a inicialização do recurso. Para isso é necessário executar um *script* básico de configuração e definir um endereço (de forma estática ou dinâmica). O recurso também deve permitir ser gerenciado remotamente, e para isso ele deve se conectar aos componentes de controle e/ou suportar protocolos para configuração e consulta de dados.

5.2.2.2 *Plug-and-Play* na Perspectiva do Servidor de Endereços

O processo apresentado na Figura 35 define a utilização de servidores de endereço e configuração com protocolos como o DHCP e BOOTP. Eles fornecem um mecanismo bem difundido para iniciar o processo de *plug-and-play*, pois permitem interceptar a conexão de novos recursos ainda nos estágios inicias. Nesta solução os servidores além de prover endereços e configurações básicas também notificam os componentes SONAr sobre os novos recursos disponíveis na rede.

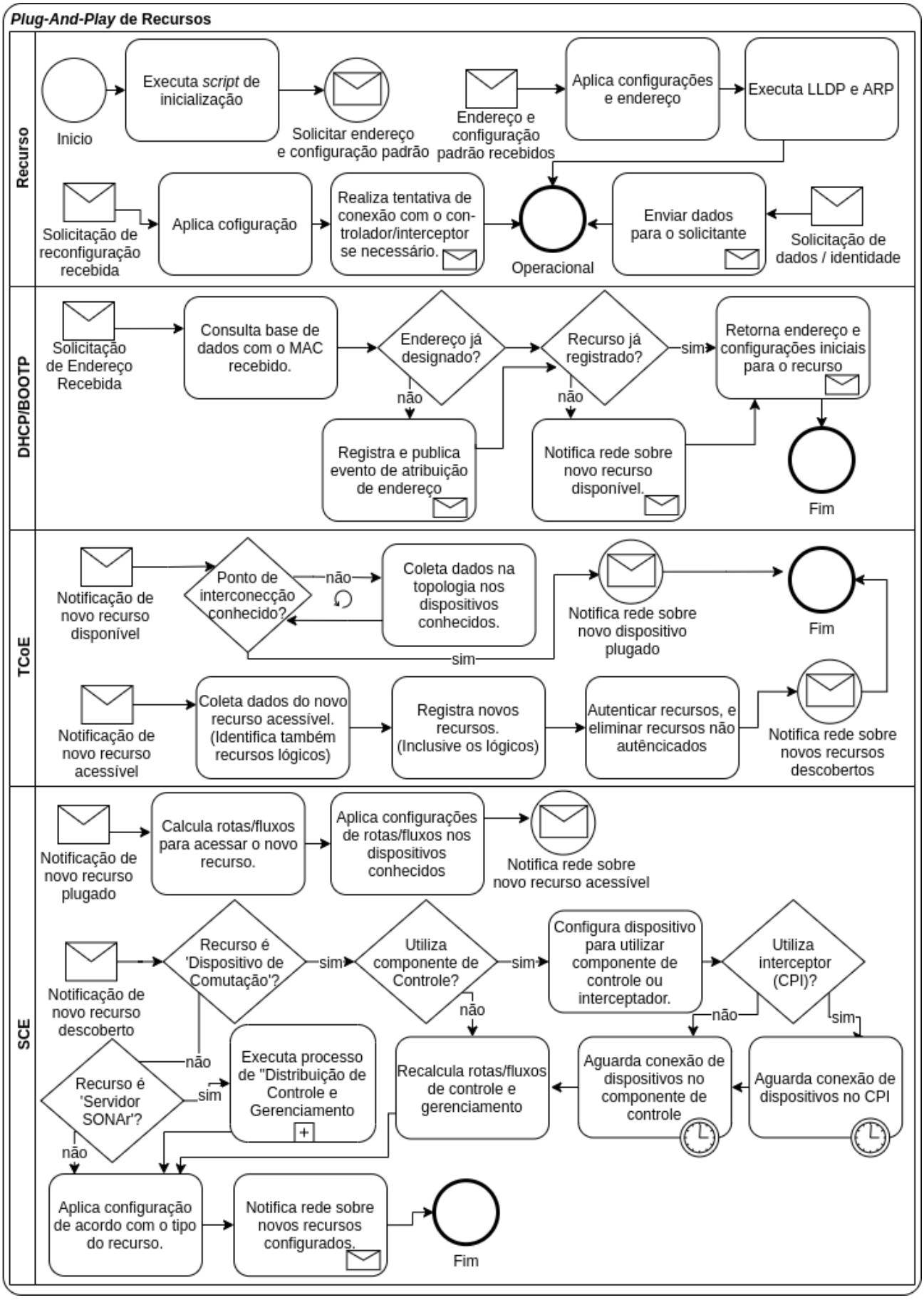


Figura 35 – Fluxo BPMN para *Plug-and-Play* de Recursos.

Fonte: Elaborado pelo autor (2021).

5.2.2.3 *Plug-and-Play* na Perspectiva da Entidade Coletora de Informações de Topologia

A TCoE (seção 5.1.4.1) apresentada na Figura 35 é responsável pela descoberta do novo recurso. Para isso, o novo recurso precisa ser acessível o que requer a execução de uma rotina para identificação do ponto de interconexão de dispositivos. Depois que a SCE (seção 5.1.2.1) configura uma rota/fluxo para acessar o novo recurso a TCoE realiza a coleta de informações, valida os dados obtidos e autoriza o novo recurso antes de notificar a rede. Caso a validação ou a autorização falhem o processo é reiniciado até que um *timeout* seja alcançado.

5.2.2.4 *Plug-and-Play* na Perspectiva da Entidade de Auto-Configuração

A SCE (seção 5.1.2.1) é responsável pela configuração dos novos recursos e adequação das configurações já implantadas relacionadas aos planos de controle e gerenciamento. No processo apresentado na Figura 35 a SCE atua em dois momentos: na configuração de rotas/fluxos para prover conectividade com o novo recurso, e na efetiva configuração do novo recurso. A primeira tarefa visa configurar um caminho ótimo que ligue os servidores SONAr até o dispositivo ao qual o novo recurso foi conectado. Na segunda tarefa o recurso já foi descoberto e a SCE aplica configurações específicas de acordo com o tipo de recurso. Dois tipos de recurso são tratados de maneira particular nesse momento: dispositivos de comutação e servidores de componentes. Caso o recurso seja um dispositivo de comutação a SCE verifica a necessidade de utilização de um componente de controle e um interceptor do plano de controle (CPI – seção 5.1.5.7) e configura os dispositivos caso necessário. Além disso a SCE reconfigura os fluxos de controle e de gerenciamento de maneira a garantir que todos os recursos permaneçam acessíveis. Caso o recurso seja um servidor de componentes a SCE verifica a necessidade de distribuição dos planos de controle e gerenciamento, e em caso afirmativo executa o processo de distribuição de componentes definidos na seção 5.2.1 que trata a inicialização da rede.

5.2.3 Estudo de Caso de Configuração de Serviços

A *Configuração de serviços* é o processo para a configuração de canais de comunicação conforme a necessidade dos usuários. Na seção 4.1.1.5 foram apresentados os diferentes tipos de serviços trabalhados nessa tese classificados quanto à forma de representação, são eles: serviços descritos formalmente (FDIBSs), serviços inferidos por comportamentos (IBSs) e serviços descritos por linguagem natural (NDIBSs). Esses serviços ainda podem ser classificados como pré-configurados, sob-demanda ou programados conforme apresentado na seção 4.1.1.4. No suporte da rede a esses serviços de diferentes tipos de aprovisionamento e formas de representação são necessários componentes específicos definidos pelo SONAr.

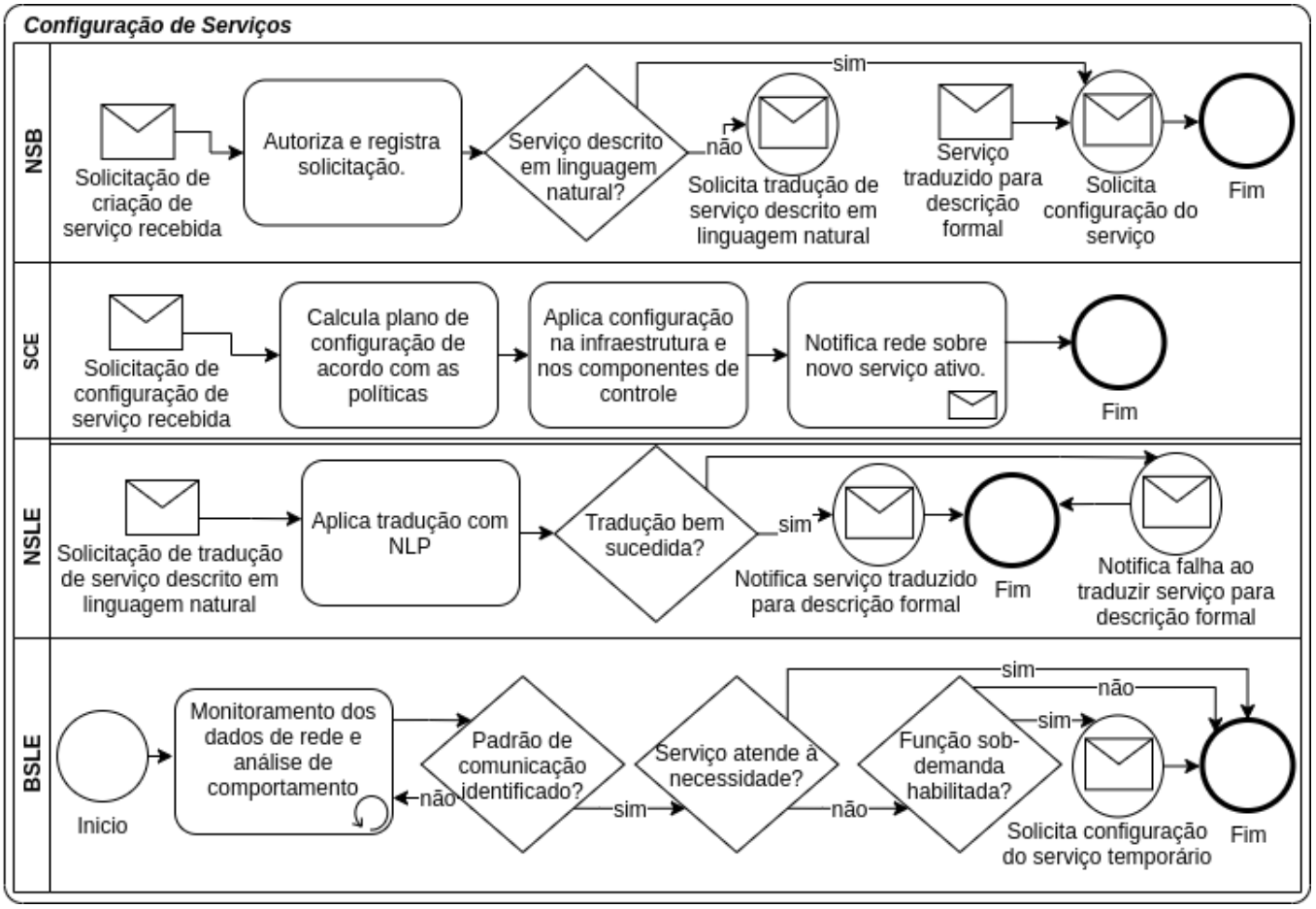


Figura 36 – Fluxo BPMN para Configuração de Serviços.

Fonte: Elaborado pelo autor (2021).

A Figura 36 apresenta um fluxo geral proposto para a implantação desses serviços que se divide em diferentes perspectivas. Essas perspectivas são descritas nas seções seguintes.

5.2.3.1 Configuração de Serviços na Perspectiva do *Broker* de Serviços da Rede

O NSB (seção 5.1.5.3) é responsável por receber solicitações para a configuração de serviços oriundos de sistemas de BSS/OSS, aplicações dos usuários ou *dashboard* de administração (e.g. NAD – seção 5.1.5.4). A primeira etapa desempenhada pelo NSB após receber a solicitação é verificar as credenciais utilizadas de maneira a prevenir ataques. Depois de autorizado o NSB verifica a forma utilizada para representação do serviço requisitado, e se for descrição em linguagem natural (NDIBS) o NSB solicita a tradução do serviço para uma descrição formal (FDIBS). Por fim, o NSB solicita a configuração do serviço via NEM (seção 5.1.5.2).

Perspectiva da Entidade de Auto-Aprendizado para Processamento de Linguagem Natural

A NSLE (seção 5.1.3.5) aplica algoritmos de processamento de linguagem natural (NLP) para traduzir a descrição informal em uma descrição formal. Ele utiliza de uma base de conhecimento construída com *deep-learning* ou com aprendizado guiado que identifica as melhores políticas e funções de acordo com as necessidades descritas na representação do serviço. Após o processamento, a NSLE notifica a rede quanto ao sucesso ou falha na tradução do serviço.

5.2.3.2 Configuração de Serviços na Perspectiva da Entidade de Auto-Aprendizado para Detecção Comportamentos

A BSLE (seção 5.1.3.6) monitora o uso da infraestrutura de maneira a detectar padrões de comportamento específicos e tipos de conteúdo trafegados. De acordo com os comportamentos detectados pode ser necessário configurar serviços sob-demanda (inferidos por comportamentos – BIIBS) ou programados para contextos específicos de acordo com o filtro do serviço.

5.2.3.3 Configuração de Serviços na Perspectiva da Entidade de Auto-Configuração

A SCE (seção 5.1.2.1) é a entidade responsável por configurar os serviços de comunicação na rede. Para isso a SCE executa um cálculo multiobjetivo considerando as políticas definidas pelo serviço com base nas avaliações dos recursos de infraestrutura obtidas via ESLE (5.1.3.4). O processamento multiobjetivo deve ser parametrizado de maneira a tornar flexível a elaboração de um plano de implantação de serviço que contemple as políticas definidas e que seja implantável com as tecnologias disponíveis. Os parâmetros nos quais esse processamento se baseia são otimizados através da TSLE (5.1.3.2), que ajuda na escolha de protocolos, técnicas e tecnologias. Depois da elaboração do plano, a SCE inicia a etapa de implantação do serviço, e para isso são executadas rotinas de configuração, instalação, instanciação e inicialização nos recursos da infraestrutura.

Capítulo 6

Implementação do *Self-Organizing Network Framework* (SONAr-Framework) e da *Self-Configuration and Management Platform* (SeCoMP)

Este capítulo tem por objetivo apresentar detalhes sobre a especificação e implementação de um *Framework* para a automação do gerenciamento de redes auto-organizadas baseadas na SONAr (seção 5.1) que é aplicado no desenvolvimento de uma plataforma de auto-configuração com foco nas operações de inicialização da rede, *plug-and-play* de recursos e aprovisionamento de serviços de comunicação. Essas operações que já foram especificadas como processos na seção 5.2, são contempladas nessa plataforma que corresponde ao tema central deste trabalho. A implementação dessa solução por meio de um *framework* tem por objetivo flexibilizar a proposta desta tese para aplicação na automação de diversas outras operações de auto-gerenciamento.

O *Self-Organizing Network Framework* (SONAr-Framework) e a SeCoMP propostas por esta tese são utilizadas para comprovação das soluções conceituais apresentadas nos capítulos anteriores servindo como "protótipos de prova de conceito" (MILLER, 1990). Esta abordagem é amplamente utilizada na literatura correlata, e fornece um mecanismo simplificado e eficiente para testar ideias e conceitos. Dessa maneira, as decisões de desenvolvimento foram guiadas pelos experimentos, os processos foram simplificados e alguns componentes foram omitidos. Aspectos *carrier-grade* serão contemplados em projetos futuros para a efetiva aplicação da solução proposta nesta tese em cenários reais.

No processo de desenvolvimento do SONAr-Framework foi definida uma estratégia na qual os componentes de controle e gerenciamento foram implementados como microsser-

viços (NAMIOT; SNEPS-SNEPPE, 2014) e implantados como contêineres (MERKEL, 2014). De fato, nos experimentos conduzidos até os *switches* foram customizados e implantados como contêineres. Além disso, essa solução se baseou nos conceitos de SDN e NFV explorados nas seções 2.3.1 e 2.3.2, o que provê uma grande flexibilidade para o gerenciamento da rede. O SONAr-Framework suporta naturalmente o protocolo *OpenFlow* (apresentado na seção 2.3.1.1), que simplifica o acesso aos dispositivos de comutação e implantação de regras de fluxos, o que é especialmente útil para a implementação da SeCoMP. Além disso, o *OpenFlow* é amplamente suportado por diversos fabricantes de dispositivos e vem sendo amplamente utilizado na academia desde a sua proposição em 2008.

Nas próximas seções serão descritos os detalhes de projeto dos componentes do SONAr-Framework que representam a base da solução de auto-gerenciamento materializada no curso dessa pesquisa; e a SeCoMP através da definição complementar à especificação do SONAr-Framework de componentes específicos relacionados à propriedade de auto-configuração no contexto das redes auto-organizadas.

6.1 SONAr-Framework: *Framework* de Gerenciamento para Redes Auto-Organizadas

O *Self-Organizing Network Framework* (SONAr-Framework) é uma solução de auto-gerenciamento que implementa a especificação SONAr (seção 5.1) e que abstrai entidades através microsserviços conteinerizados. A SONAr fornece uma especificação de componentes de gerenciamento flexível na qual as entidades podem ser implantadas de diferentes maneiras, por exemplo: como aplicações do Controlador, como funções de rede virtualizadas ou mesmo como funcionalidades embarcadas nos roteadores. O presente trabalho propõe uma abstração destes componentes como contêineres *Docker* o que simplifica o gerenciamento e utilização da solução. Todos os contêineres estão ou serão disponibilizados no *Docker Hub* – <https://hub.docker.com/u/meharsonar>. Os microsserviços disponibilizados são implementados na linguagem JAVA através do *framework* de desenvolvimento *Spring-Boot* e o código-fonte do SONAr-Framework é disponibilizado no *GitLab* da FACOM-UFU – <http://gitlab.facom.ufu.br/sonar>.

O SONAr-Framework se baseia na premissa de existência de elementos de infraestrutura nos quais pode-se aprovisionar os componentes de gerenciamento e de controle conteinerizados bem como funções de rede para o tratamento do plano de dados. Estes elementos, convencionados como *Servidores SONAr*, são responsáveis por acomodar as entidades e componentes auxiliares da SONAr e desempenham um papel fundamental da automação da inicialização da rede através do execução automática do *script* de *startup* que faz parte do *Auto-Boot Manager*. A especificação do SONAr-Framework conta ainda com outros elementos de *software* além dos já especificados pela SONAr com objetivos

diversos, tais como: *i)* disponibilizar informações para a coleta realizada pelas CoEs; *ii)* suporte à requisição de endereços (e.g. via DHCP); e, *iii)* gestão da infraestrutura conteinerizada dentre outros.

A Figura 37 apresenta uma visão geral da composição das peças de *software* necessárias tanto no 'Servidor SONAr' quanto nos elementos de rede como um todo. O suporte aos protocolos LLDP, SNMP, DHCP e ARP é um requisito para a utilização do SONAr-Framework em sua versão atual. Além disso, o 'Servidor SONAr' define a utilização de um provedor de contêineres para a acomodação das SOEs, SLEs, CoEs e componentes auxiliares SONAr. Estes contêineres podem ser gerados automaticamente através do código-fonte com *scripts* disponibilizados junto ao código, ou podem ser baixados automaticamente via *Internet* a partir *Docker Hub*. Os demais elementos de *software* que compõe o 'Servidor SONAr' apresentados na Figura 37.

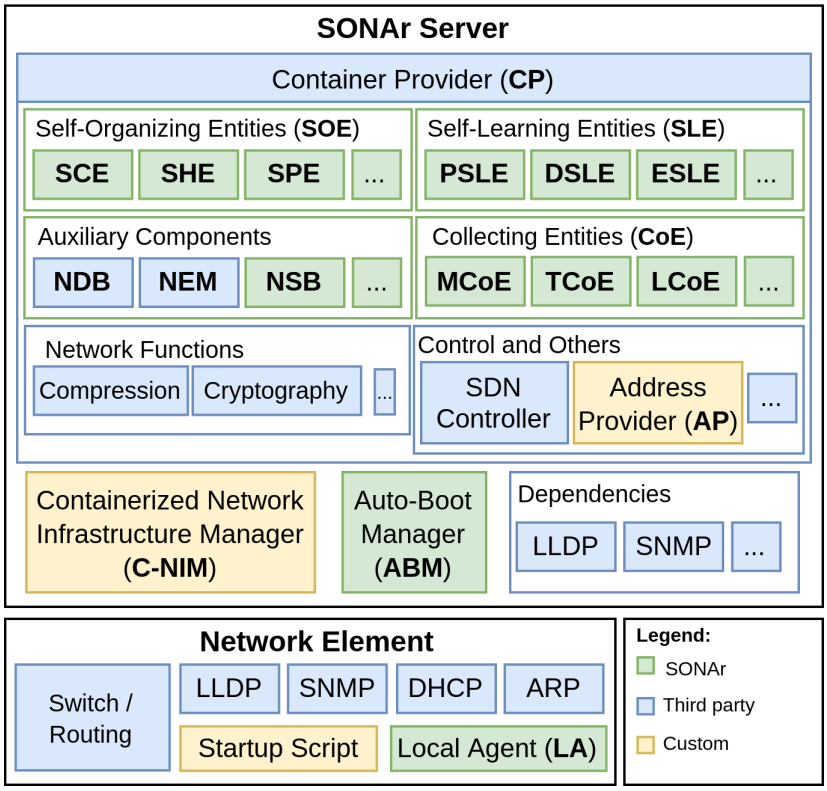


Figura 37 – Visão Geral do SONAr-Framework.

Fonte: Elaborado pelo autor (2021).

'Servidores SONAr' são essencialmente máquinas hospedeiras (físicas ou virtuais) com capacidade de processamento e armazenamento que podem ser utilizadas para a implantação de componentes. Essas máquinas devem ser autorizadas durante a inicialização ou extensão da rede, e devem suportar protocolos de rede úteis ao gerenciamento como LLDP e SNMP.

6.1.1 Componentes definidos pelo SONAr-Framework

Na implementação do SONAr-Framework foram definidos componentes responsáveis por aspectos práticos da solução além dos já especificados pela SONAr, tais como o aprovisionamento de contêineres e a disponibilização de dados. Estes componentes são descritos abaixo:

- *Containerized Network Infrastructure Manager* (C-NIM): responsável por gerir os componentes e funções de rede conteinerizadas (seção 6.1.1.4);
- *Address Provider* (AP): responsável por prover endereços e configurações básicas automaticamente quando solicitado pelos recursos de infraestrutura endereçáveis (seção 6.1.1.5);
- *Container Provider* (CP): responsável por prover a abstração de contêineres (seção 6.1.1.1);
- *Resource Data Provider* (RDP): responsável por prover informações do servidor (seção 6.1.1.2); e,
- *Neighbor Discovery Provider* (NDP): responsável por coletar informações sobre os recursos diretamente conectados ao servidor (seção 6.1.1.3).

Esses elementos de *software* são soluções de implementação para a viabilizar a construção de um Servidor SONAr com todas as características necessárias para a acomodação dos componentes definidos por essa arquitetura. Em cenários mais simples como a SONAr aplicada diretamente em um elemento de rede ou em um controlador SDN, essas funções normalmente já são implementadas de forma nativa. Por exemplo, um roteador tradicionalmente suporta LLDP, SNMP e DHCP de forma nativa.

A Figura 38 apresenta uma visão mais detalhada sobre a integração do 'Servidor SONAr' com os elementos de rede na versão atual do SONAr-Framework. Nesta figura podem ser observadas as entidades de coletas representadas pela caixa CoE que estabelece uma comunicação com os elementos através de técnicas/protocolos específicos e que inicia o fluxo definido pelo SONeM. Pode-se observar também a utilização do CPI como um *proxy* para o Controlador SDN. O NDB e NEM são componentes fundamentais para o SONAr-Framework e estabelecem comunicação com todos os demais componentes, inclusive com as SLEs que não atuam diretamente nos elementos de rede. Por fim, a figura inclui uma representação para as SOEs, elementos principais de auto-organização da SONAr, e o AP, componente para provisão de endereços que pode ser abstraído como um servidor DHCP.

6.1.1.1 *Container Provider* (CP)

O *Container Provider* (CP) é responsável por prover a abstração de contêineres para a rede. Contêineres são utilizados pelo *Containerized Network Infrastructure Manager*

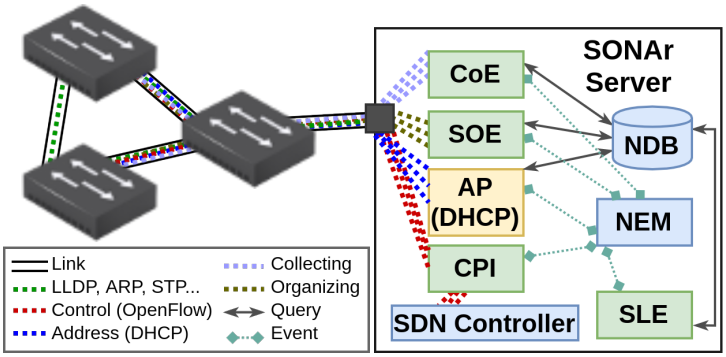


Figura 38 – Exemplo de integração entre o Servidor SONAr e os Elementos de Rede.

Fonte: Elaborado pelo autor (2021).

(seção 6.1.1.4) no aprovisionamento de funções de rede e componentes de controle e gerenciamento. Segundo (MERKEL, 2014) um contêiner pode ser visto de maneira superficial como uma versão leve de uma máquina virtual, e mais especificamente como uma unidade padrão de software que empacota seu código fonte e todas as suas dependências em um executável simples sem uma camada completa de virtualização.

Neste trabalho o CP foi provido pelo *Docker* (MERKEL, 2014) em sua versão 18.06.2-ce. O *Docker* é uma solução de *Platform as a Service* (PaaS) com virtualização no nível do sistema operacional que utiliza técnicas como *cgroups*, *namespaces* e *OverlayFS* pra criar contêineres leves e independentes. O *Docker* também é amplamente utilizado na indústria e academia e fornece uma interface administrativa flexível para a utilização pelo C-NIM.

6.1.1.2 *Resource Data Provider* (RDP)

O *Resource Data Provider* (RDP) é o agente local responsável por extrair e expor dados dos recursos. Ele coleta dados de acordo com o tipo de recurso, tais como: uso de banda, uso de memória, uso de processamento, informações sobre portas e componentes, e dados sobre vizinhos. Essas informações são processadas e expostas por serviços com contratos bem definidos para acesso remoto.

Neste trabalho o RDP foi provido pelo *net-snmp* (SNMP, 2019) em sua versão 5.7.3. O SNMP é um protocolo para coleta e organização de informações sobre dispositivos gerenciados em redes IP. O *net-snmp* é uma suíte de *softwares* com licença BSD utilizados para implementação do protocolo SNMP (v1, v2c e v3) distribuído nativamente em diversas distribuições UNIX como FreeBSD, Solaris e OSX. O *net-snmp* é um dos projetos mais populares do *sourceforge*, um site para compartilhamento de *softwares* gratuitos (SOURCEFORGE, 2019), e é amplamente utilizado na academia e indústria.

6.1.1.3 *Neighbor Discovery Provider* (NDP)

O *Neighbor Discovery Provider* (NDP) é uma aplicação capaz de identificar os recursos de infraestrutura vizinhos, bem como quais os enlaces aos quais eles estão conectados. Essa é uma informação importante para a definição de rotas de forma automática, e principalmente para o estabelecimento dos fluxos de controle e gerenciamento. As informações geradas pelo NDP tradicionalmente são expostas a agentes externos via RDP, por isso é necessário que ambas as soluções sejam integradas.

Neste trabalho o NDP foi provido pelo *lldpd* (LLDPD, 2019) em sua versão 1.0.3. O *Link Layer Discovery Protocol* (LLDP) é um protocolo de descoberta no nível da camada de enlace independente de fabricante definido pela norma IEEE 802.1AB (CONGDON, 2002). Junto ao protocolo, que é amplamente suportado pela maioria dos dispositivos de rede, foi proposta também uma MIB (LLDP-MIB) que é o mecanismo para definição de dados utilizados pelo SNMP. O *lldpd* é uma iniciativa da comunidade para implementação da especificação IEEE 802.1AB com licença ISC que permite fácil integração com o *net-snmp* (utilizado como RDP neste projeto).

6.1.1.4 *Containerized Network Infrastructure Manager* (C-NIM)

O *Containerized Network Infrastructure Manager* (C-NIM) é uma aplicação inspirada no *Virtualised Infrasctructure Manager* (VIM) que faz parte do *Management and Network Orchestration* (MANO) e que abstrai parte das funcionalidades do *Network Infrastructure Manager* (NIM) proposto pela SONAr. Ele é responsável por gerir a infraestrutura de rede de maneira a prover componentes, funções de rede e aplicações do usuário como contêineres distribuídos entre os servidores SONAr. Ele lida diretamente com o CP no aprovisionamento de contêineres, e é responsável por instanciar os componentes de controle e gerenciamento de acordo com as requisições do ABM.

Neste trabalho o C-NIM foi implementado como uma aplicação escrita com a linguagem de programação *JAVA 8* e com base no *framework* de desenvolvimento *Spring-Boot* versão 2.1.2. O código fonte dessa aplicação pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/custom-components/sonar-containerized-infrastructure-manager>. O C-NIM expõe *web-services* baseados em REST para listar, registrar, instalar, iniciar, parar e remover contêineres. Ele se integra ao CP (provido pelo *Docker*) através da API *docker-client* versão 8.16.1 disponibilizada pelo *Spotify*. Essa API simplifica a utilização do *Docker* por aplicações externas.

O C-NIM implementado neste trabalho pode ser substituído por outras soluções mais robustas, como por exemplo o *Kubernetes* que é desenvolvido pela *Red Hat* e é uma plataforma *open source* que automatiza as operações envolvendo contêineres *Linux*. A decisão de implementar uma versão própria está relacionada à simplificação e customização da solução às necessidades do SONAr-Framework.

6.1.1.5 *Address Provider* (AP)

O *Address Provider* (AP) é um elemento de *software* autônomo capaz de prover configurações iniciais solicitadas pelos recursos, tais como: endereços, configuração de DNS, e *scripts* de inicialização. Essa abordagem já é amplamente difundida através dos protocolos BOOTP e DHCP, sendo este último protocolo utilizado na implementação do AP proposto neste trabalho. O código fonte dessa aplicação pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/custom-components/sonar-dhcp-server>.

Neste trabalho o AP foi implementado como uma aplicação escrita com a linguagem de programação *JAVA 8* e com base no *framework* de desenvolvimento *Spring-Boot* versão 2.1.2. A implementação do protocolo DHCP foi desenvolvida de forma nativa (sem API específica). O AP implementado é basicamente um DHCP *Server* que provê endereços IPv4 e é integrado ao SONAr-Framework via NEM e NDB. Sempre que um novo recurso solicita um endereço o AP verifica se o endereço está livre na base de dados (via NDB), e designa um endereço ao novo recurso e notifica os demais componentes via NEM.

6.1.1.6 Controlador SDN/*OpenFlow*

O *Controlador SDN/OpenFlow* é responsável por fornecer uma interface programável de alto nível para acessar e controlar os dispositivos de rede. Embora o uso de componentes de controle não seja obrigatório, a versão atual do SONAr-Framework suporta apenas *switches OpenFlow*, e por isso foi utilizado o *ONOS* versão 2.2.0 (ON.LAB, 2014) um controlador SDN com suporte ao *OpenFlow* que é flexível e amplamente aceito pela academia. O suporte ao protocolo *OpenFlow* permite a abstração de fluxos na provisão de canais de comunicação, o que vem de encontro à definição de serviços proposto pelo IBSM.

Neste trabalho o *ONOS* foi utilizado através de um contêiner personalizado cuja definição pode ser acessada pelo link <http://gitlab.facom.ufu.br/sonar/docker/tree/master/controller/onos/v1>. Ao iniciar o *ONOS* a aplicação "*org.onosproject.openflow-base*" é iniciada automaticamente.

6.1.2 Componentes de Gerenciamento SONAr

Esta seção apresenta os componentes propostos pela SONAr implementados pelo SONAr-Framework com o objetivo de fornecer uma base para a implementação de rotinas de auto-gerenciamento específicas para cada propriedade de automação[1]. A implementação destes componentes na versão atual do SONAr-Framework visa viabilizar a construção da SeCoMP, uma plataforma de auto-configuração e gerenciamento de redes definidas por *software*, e a realização dos experimentos propostos no capítulo 7. Dessa forma, o

[1] auto-configuração, auto-cura, auto-otimização e auto-proteção

SONAr-Framework implementa apenas componentes fundamentais definidos pela SONAr, deixando de lado componentes específicos responsáveis pela implementação dos fluxos de gerenciamento (e.g. SCEs, SLEs e CoEs). Entretanto, uma porção destes componentes necessários para a automação de operações de auto-configuração foi implementada pela SeCoMP e incorporada ao SONAr-Framework de maneira a permitir o estabelecimento do estado operacional da rede de maneira automática. Os componentes SONAr implementados pelo SONAr-Framework proposto nesta tese são listados abaixo:

- *Network Database* (NDB): responsável pelo armazenamento de dados de rede (seção 6.1.2.1);
- *Network Event Manager* (NEM): responsável pela comunicação entre os componentes a partir da abstração de eventos (seção 6.1.2.2); e,
- *Control Plane Interceptor* (CPI): responsável pela interceptação da comunicação entre recursos da camada de infraestrutura e componentes da camada de controle (seção 6.1.2.3).

6.1.2.1 *Network Database* (NDB)

A *Network Database* (NDB) – já apresentada na seção 5.1.5.1 – é o componente SONAr responsável pelo armazenamento dos dados de rede de maneira distribuída. Pode-se utilizar diferentes implementações de acordo com o tipo de dado sendo manipulado, por exemplo, uma abordagem relacional pode ser indicada para armazenamento de estruturas de dados mais complexas e uma abordagem não relacional para armazenamento de dados no estilo chave/valor.

Neste trabalho a NDB foi abstraída através do *Cassandra* versão 3.11.4 (ALAPATI, 2018), um banco distribuído não relacional mantido pela Apache e com o foco em altos volumes de dados. O *Cassandra* foi desenvolvido pelo *Facebook* para otimizar os mecanismos de buscas, mas logo se tornou um *software open-source* mantido pela comunidade, e hoje é um dos dez DBMSs mais utilizados (DB-ENGINES, 2019).

A imagem de contêiner utilizada na definição do NDB através do *Cassandra* é uma imagem personalizada e pode ser acessada pelo link <http://gitlab.facom.ufu.br/sonar/docker/tree/master/database/cassandra/v2>. O *schema* de banco de dados aplicado pelo ABM define objetos do tipo: *controller*, para registro dos componentes de controle; *element* e *port*, para armazenamento de informações de topologia; *service* e *configuration*, para armazenamento de definição de serviços e suas configurações; e, *property*, para armazenamento de propriedades de sistema.

6.1.2.2 *Network Event Manager* (NEM)

O *Network Event Manager* (NEM) – já apresentada na seção 5.1.5.2 – é o componente SONAr responsável por interfacear a comunicação entre componentes de gerenciamento

através da troca de eventos. O NEM permite notificar componentes sobre determinadas situações tais como detecção de novos dispositivos e diagnóstico de degradação da comunicação; e executar procedimentos remotos em um estilo RPC como, por exemplo, para solicitação de configuração de fluxos de controle.

Neste trabalho o NEM foi abstraído através do *rabbitMQ* versão 3.8 (BOSCHI; SANTOMAGGIO, 2013), um *software open-source* para troca de mensagens que implementa o protocolo *Advanced Message Queuing Protocol* (AMQP) desenvolvido pela *Pivotal Software*. O *rabbitMQ* é uma solução leve com suporte à *JAVA* e amplamente utilizado na academia e na indústria. Foi utilizada uma imagem customizada que pode ser acessada pelo link <http://gitlab.facom.ufu.br/sonar/docker/blob/master/message/rabbitmq/v1>.

6.1.2.3 *Control Plane Interceptor* (CPI)

O *Control Plane Interceptor* (CPI) – já apresentado na seção 5.1.5.7 – é o componente SONAr responsável por interceptar as mensagens entre os componentes de controle e os recursos de infraestrutura. Na implementação do *framework* proposto nesta tese foi desenvolvida uma aplicação na linguagem de programação *JAVA 8* e com base no *framework Spring-Boot* versão 2.1.2. O código fonte dessa aplicação pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/custom-components/sonar-controller-interceptor>.

O CPI suporta interceptação de tráfego *OpenFlow* através de um *proxy* TCP desenvolvido com *netty* versão 4.1.31 (MAURER; WOLFTHAL, 2015), um *framework JAVA* projetado para lidar com eventos de rede assíncronos; e com a biblioteca *openflowj* versão 3.5.539, disponibilizada pelo projeto *floodlight* e que suporta as versões 1, 2 e 3 do protocolo *OpenFlow*.

6.1.3 Bibliotecas do SONAr-Framework

De maneira a garantir o princípio de desacoplamento para os componentes da camada de gerenciamento o SONAr-Framework define um conjunto de bibliotecas que interfaceiam a comunicação com os recursos de infraestrutura, com os componentes de controle e com os próprios componentes de gerenciamento. Essa abordagem permite a abstração de um componente SONAr de maneira simplificada, por exemplo, o NDB-Client pode fornecer uma implementação que se baseia no uso de estruturas de dados em memória de maneira transparente para os componentes usuários da NDB. Da mesma forma, o NIM-Client não requer a utilização de um componente de gerenciamento de infraestrutura formal como definido pela SONAr na especificação do NIM (seção 5.1.5.5), o que melhora o desempenho na comunicação entre os componentes de gerenciamento e os recursos de infraestrutura. As bibliotecas disponibilizadas pelo SONAr-Framework são:

- *Network Database Client* (NDB-Client): responsável por interfacear a comunicação com a NDB (seção 6.1.3.1);
- *Network Event Manager Client* (NEM-Client): responsável por interfacear a comunicação com o NEM (seção 6.1.3.2);
- *Network Infrastructure Manager Client* (NIM-Client): responsável por interfacear a comunicação com componentes de controle e recursos, como por exemplo: *switches*, controladores SDN e servidores (seção 6.1.3.3); e,
- *SONAr-Model & SONAr-Util*: bibliotecas para definição do modelo de dados e compartilhamento de funções úteis (seção 6.1.3.4).

6.1.3.1 *Network Database Client* (NDB-Client)

O NDB-Client é uma API que abstrai e interfaceia a comunicação entre os componentes SONAr e a NDB. Essa API esconde detalhes da implementação, tais como tecnologias, *schemas* e parâmetros de configuração. Além disso, essa API provê acesso à base de forma distribuída, sem que os componentes percebam. O código fonte pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/blob/master/client-apis/network-database-client>.

Na implementação do *framework* apresentado nesta tese o NDB-Client foi construído na linguagem de programação *JAVA 8*, através da API oficial do *cassandra* versão 3.1.4. Essa API disponibiliza quatro serviços básicos: TopologyService (responsável por manipular os dados de topologia), ServiceService (responsável por manipular os dados de serviços), o PropertyService (responsável por manipular parâmetros e propriedades do sistema), e o ConfigurationService (responsável pelo armazenamento de configurações).

6.1.3.2 *Network Event Manager Client* (NEM-Client)

O NEM-Client é uma API que permite a troca de mensagens, execução de procedimentos remotos e envio de eventos através do NEM. Assim como o NDB Client, essa API provê acesso ao NEM de forma distribuída sem que os componentes percebam. O código fonte pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/blob/master/client-apis/network-event-manager-client>.

No *framework* apresentado nesta tese o NEM-Client foi implementado com a linguagem de programação *JAVA 8* e com a biblioteca cliente do *rabbitMQ* com suporte ao protocolo AMQP. Foi utilizada troca de mensagens assíncronas divididas em tópicos. Alguns exemplos de tópicos são: "*sonar.topology.element.added*", "*sonar.boot.finish.configuration*", "*sonar.pnp.finish.channel-configuration*", "*sonar.interceptor.packet-in.dhcp*" e "*sonar.sce.callback.pnp.config*".

6.1.3.3 *Network Infrastructure Manager Client* (NIM-Client)

O NIM-Client é uma API que interfaceia a comunicação entre os componentes de gerenciamento e os recursos de infraestrutura e componentes de controle. Ele provê uma camada genérica e flexível para acessar recursos endereçáveis, tais como dispositivos e servidores, e recursos lógicos como o C-NIM e o Controlador SDN. Essa API permite por exemplo que o ABM solicite o aprovisionamento de componentes em um servidor SONAr, ou que a SCE solicite a configuração de um controlador em um *switch OpenFlow*, ou ainda que a TCoE consulte quais os dispositivos estão conectados a um controlador. Toda a comunicação proveniente da camada de gerenciamento em direção às camadas de infraestrutura e de controle passa por essa API, que convenientemente esconde detalhes como protocolos, tecnologias e modelos de equipamentos.

O código fonte pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/client-apis/network-infrastructure-manager-client> e foi implementado com a linguagem de programação *JAVA 8*. Na versão implementada nesta tese foram utilizadas as bibliotecas: *ovsdb-client* versão 1.0.1 que é disponibilizada pela *vmware* para configuração de equipamentos com o protocolo OVSDB; o *snmp4j* versão 2.7.0 utilizada na realização de consultas de informações em recursos de infraestrutura através do protocolo SNMP; e, *openflowj* versão 3.5.539 que é disponibilizada junto ao projeto *floodlight* e que implementa o protocolo *OpenFlow* versão 1, 2 e 3.

6.1.3.4 *SONAr-Model & SONAr-Util*

SONAr-Model é uma biblioteca desenvolvida na linguagem de programação *JAVA 8* que abstrai o modelo de dados que é utilizado pelos componentes como classes *JAVA*. Todos os componentes SONAr definem o *SONAr-Model* como dependência, o que facilita a comunicação dentro do sistema e reduz código redundante. O código fonte pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/core-libs/sonar-model>.

SONAr-Util é uma biblioteca desenvolvido na linguagem de programação *JAVA 8* que provê classes úteis ao desenvolvimento de componentes SONAr. Dentre as utilidades destacam-se: conversão de datas, processamento de *strings*, manipulação de endereços IP, padronização de endereços MAC e interpretação de protocolos da arquitetura TCP/IP. O código fonte pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/core-libs/sonar-util>.

6.2 SeCoMP: Plataforma para Gerenciamento e Configuração de Redes Auto-Organizadas

A *Self-Configuration and Management Platform* (SeCoMP) é uma solução de auto-configuração para concepção de redes auto-organizadas definidas por *softwares*. O objetivo desta plataforma é o de automatizar as operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços conforme especificado na seção 5.2. A implementação do SeCoMP se guiou pela execução dos experimentos propostos no capítulo 7, e dessa forma, passou por diversas alterações de maneira a prover melhorias do desempenho e correção de situações não previstas (e.g. atraso no processamento LLDP). Considerando apenas os cenários utilizados pelos experimentos optou-se por uma simplificação da implementação do fluxo de auto-configuração proposto pelo estudo de caso apresentado na seção 5.2, na qual alguns componentes SONAr não foram implementados: *i)* ESLE (seção 5.1.3.4), *ii)* NSLE (seção 5.1.3.5), *iii)* BSLE (seção 5.1.3.6), *iv)* SCoE (seção 5.1.4.4) e *v)* MCoE (seção 5.1.4.2). Os componentes SONAr implementados pela SeCoMP proposta nesta tese são listados abaixo:

- *Auto-Boot Manager* (ABM): responsável pela inicialização dos componentes de controle/gerenciamento, funções de rede básicas e aplicações úteis no servidor SONAr (seção 6.2.1);
- *Self-Configuration Entity* (SCE): responsável pela implementação do comportamento de auto-configuração na rede (seção 6.2.2);
- *Topology Collector Entity* (TCoE): responsável pela coleta de informações de topologia dos recursos de infraestrutura (seção 6.2.3);
- *Network Service Broker* (NSB): responsável por prover uma interface para gerenciamento de serviços de comunicação (seção 6.2.4).

Além destas implementações foram necessárias alterações nos componentes e bibliotecas do SONAr-Framework. Dentre os sub-projetos que foram mais alterados destacam-se, como esperado, os projetos de: *i) core-libs*, que compreende as bibliotecas compartilhadas com funções úteis e definição do modelo de dados; *ii) client-libs*, que são as bibliotecas utilizadas nas interfaces com os componentes de controle/gerenciamento (e.g. NDB-*Client*) e recursos de infraestrutura (e.g. NIM-*Client*); e, *iii) custom-components*, que se refere às implementações de sistemas customizadas pelo SONAr-Framework (e.g. DHCP Server). Além dessas alterações o CPI também precisou ser revisto de maneira a implementar os fluxos de auto-configuração que o envolvem. O comportamento prático do CPI é descrito brevemente na seção 6.2.5.

A implementação da SeCoMP foi incorporada à implementação do SONAr-Framework de maneira a viabilizar o estabelecimento do estado operacional da rede, e consequentemente, viabilizar a implementação de outras soluções de gerenciamento com enfoque em

propriedades como cura e otimização. O código fonte da SeCoMP combinado com o SONAr-Framework pode ser obtido através do link <http://gitlab.facom.ufu.br/sonar>.

6.2.1 *Auto-Boot Manager* (ABM)

O *Auto-Boot Manager* (ABM) – já apresentado na seção 5.1.5.6 – é o principal componente SONAr responsável pela inicialização da rede, pois é ele quem aprovisiona e configura os componentes de controle e de gerenciamento. Na SeCoMP o ABM é tratado como um *script* de inicialização e uma aplicação na linguagem de programação *JAVA 8*. A aplicação parte do ABM utiliza o *framework* de desenvolvimento *Spring-Boot* versão 2.1.2, que facilita o desenvolvimento de aplicações em *JAVA* através de bibliotecas para a injeção de dependências, exposição de *web-services* e manipulação de bases de dados. O código fonte da aplicação que abstrai parte do ABM na SeCoMP pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/auto-boot-manager>.

O *script* de inicialização verifica o estado e inicia se necessário o *Container Provider* (CP), o *Resource Data Provider* (RDP), o *Neighbor Discovery Provider* (NDP), o *Containerized Network Infrastructure Manager* (C-NIM) e a própria aplicação ABM. Esse *script* é executado junto à inicialização do 'Servidor SONAr', o que automatiza a inicialização da rede subjacente.

A aplicação que compõe o ABM executa basicamente o fluxo definido em 5.2.1. Ela inicia por checar junto ao C-NIM (seção 6.1.1.4) quais são os componentes ativos, pra só então começar processo de instanciação, inicialização e alocação de componentes. Os primeiros componentes aprovisionados são o NDB (seção 6.1.2.1), o NEM (seção 6.1.2.2) e os componentes de controle (controlador SDN/*OpenFlow* – seção 6.1.1.6). O ABM aguarda até que os componentes iniciem, verifica se o *schema* da base de dados foi criado, e em caso contrário cria o *schema* e popula com dados iniciais de acordo com arquivos de configuração específicos por tecnologia. Os dados de acesso ao NDB, NEM e componentes de controle são utilizados no aprovisionamento dos próximos componentes, são eles: o AP (seção 6.1.1.5), o CPI (seção 6.1.2.3), a TCoE (seção 6.2.3), SCE (seção 6.2.2) e o NSB (seção 6.2.4).

6.2.2 *Self-Configuration Entity* (SCE)

A *Self-Configuration Entity* (SCE) – já apresentada na seção 5.1.2.1 – é o componente SONAr responsável efetivamente por prover funcionalidades relacionadas à propriedade de auto-configuração para a rede. Essa entidade é a que configura os recursos de infraestrutura para tornar a rede operacional e para prover serviços de comunicação adequados às necessidades dos usuários. Os fluxos de execução da SCE apresentados de maneira simplificada nas seções 5.1.7.1, 5.1.7.2 e 5.1.7.3, e de maneira mais detalhada no estudo de caso apresentado na seção 5.2, foram implementados de maneira programática com

pequenas alterações em uma aplicação desenvolvida na linguagem de programação *JAVA 8* e com base no *framework Spring-Boot* versão 2.1.2. O código fonte dessa aplicação pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/self-organizing-entities/self-configuration-entity>.

A SCE atua sob estímulos oriundos do NEM (seção 6.1.2.2) que são enviados pela TCoE (seção 6.2.3) e pelo NSB (seção 6.2.4). Na operação de inicialização da rede a SCE aguarda a notificação de que a rede foi descoberta para só então iniciar o procedimento de configuração. A primeira parte dessa operação consiste em identificar as melhores rotas para o estabelecimento dos fluxos de controle e de gerenciamento que englobem todos os recursos endereçáveis. Para isso a SCE aplica o algoritmo de *Dijkstra* para o cálculo dos melhores caminhos usando o 'Servidor SONAr' no qual a entidade foi aprovisionada como ponto de origem. Esse procedimento gera um grafo com caminhos ótimos e sem ciclos representado por uma árvore com raiz no servidor. É importante definir fluxos sem ciclos para evitar *loops* na rede, uma vez que protocolos de controle de *loop* como o *Spanning Tree Protocol* (STP) precisam ser removidos para que a rede possa utilizar caminhos redundantes de acordo com as decisões tomadas pelas entidades auto-organizadoras. Antes de alterar qualquer configuração na rede, a SCE verifica as configurações registradas na base de dados de maneira a detectar se as novas configurações introduzirão *loops* ou *black-holes*, e caso isso ocorra as configurações são corrigidas.

Ainda na operação de inicialização da rede, depois que a SCE calcula as configurações necessárias para o estabelecimento dos fluxos de controle e de gerenciamento ela configura todos os *switches* via OVSDB (através do NIM-Client – seção 6.1.3.3) para utilizarem um controlador SDN (ou um CPI, o que é definido via configuração no momento da execução do ABM), configura uma *bridge* a ser utilizada no plano de controle e desabilita os protocolos de controle de *loop* (e.g. STP). A SCE aguarda até que os dispositivos se conectem ao CPI e/ou Controlador SDN e depois registra o *datapath* e os *ids* das portas na base de dados (através do NDB-Client – seção 6.1.3.1). Nesse ponto a SCE verifica os recursos endereçáveis remanescentes e emprega uma tratativa de acordo com o tipo: se for um servidor SONAr a SCE faz uma requisição RPC ao ABM local (via NEM), que se comunica com o ABM remoto para a realização da distribuição; caso seja um servidor de funções de rede, a SCE também via ABM realiza o aprovisionamento dos componentes no novo servidor e depois realiza a distribuição do gerenciamento; e, caso seja outro tipo de recurso a SCE não realiza nenhuma configuração adicional. Ao fim da configuração dos fluxos de controle, a SCE inicia a etapa de configuração dos serviços onde são lidos os registros na base de dados e executadas rotinas definidas pela operação de aprovisionamento de serviços. Após o término das configurações a SCE notifica os demais componentes via NEM sobre o êxito ou fracasso do processo.

Na operação de *plug-and-play* de recursos a SCE atua em dois momentos: quando o novo recurso é plugado e quando ele é descoberto. Ao ser plugado o recurso pode não ser

acessível, o que requer que a SCE calcule rotas/fluxos que liguem os servidores SONAr ao novo recurso de maneira similar ao cálculo realizado no estabelecimento dos planos de controle e de gerenciamento apresentado anteriormente. Depois que o recurso se torna acessível a TCoE executa a descoberta do novo recurso utilizando SNMP, e notifica a rede sobre o novo recurso descoberto. Nesse momento a SCE entra em ação novamente e configura o novo recurso de maneira similar à aplicada na operação de inicialização da rede. De acordo com o tipo do recurso a SCE aplica configurações específicas: se for um dispositivo de comutação a SCE configura o controlador (ou o CPI), espera até o dispositivo se conectar e aplica configurações e fluxos básicos; se for um servidor SONAr ou servidor de funções de rede a SCE solicita o aprovisionamento dos novos componentes e distribuição dos planos; e caso seja de outro tipo a SCE não aplica configurações específicas na versão atual da plataforma proposta por esta tese.

Na operação de aprovisionamento de serviços a SCE recebe solicitação de criação de serviços via NSB (seção 6.2.4). A SCE inicia o plano de configuração através da análise das políticas definidas pelos requisitos de controle de acesso (RNF12, RNF13, RNF14, RNF15 e RNF16), de maneira a filtrar os recursos elegíveis para a implantação do serviço. De acordo com as demais políticas utilizadas na especificação do IBS a SCE identifica quais as funções de rede são necessárias e quais os recursos suportam essas funções. Com base nessas informações a SCE calcula os possíveis conjuntos de recursos que contemplam a implantação de todas as funções de rede necessárias, e realiza cálculos de melhores caminhos considerando todos os trechos que passem pelos recursos especificados. Se nenhuma função de rede for necessária, é realizado apenas um cálculo de melhor caminho através dos dispositivos de comutação e os sistemas finais.

A avaliação de um recurso r_i sobre a sua capacidade de atender uma política p_j é dada pela Equação 1 (função *rating*), onde $r_i \in$ R, tal que R é o conjunto de recursos disponíveis na rede representado por $\{r_1,r_2...r_n\}$ onde n é o tamanho do conjunto R; e, $p_j \in$ P, tal que P é o conjunto de políticas suportadas pela rede representado por $\{p_1,p_2...p_m\}$ onde m é o tamanho do conjunto P. O valor da avaliação de um recurso em relação a uma determinada política varia entre 0 e *MAX_RATE*, sendo 0 usado quando o recurso não atende à política, 1 se atende ou não interfere na política, e de 1 a *MAX_RATE* de acordo com a avaliação normalizada provida pela ESLE (seção 5.1.3.4). *MAX_RATE* é um parâmetro otimizável pela TSLE (seção 5.1.3.2) que determina a nota máxima que um recurso pode receber em relação a uma política. Na implementação realizada neste trabalho *MAX_RATE* inicia com o valor 5.

$$rating(r_i, p_j) = \begin{cases} 0 & \text{se } r_i \text{ não atende } p_j \\ 1 & \text{se } r_i \text{ atende ou não impacta em } p_j \\ > 1 \text{ e} \leq \text{MAX\_RATE} & \text{de acordo com a avaliação de } r_i \text{ sobre } p_j \end{cases} \quad (1)$$

Com base na função *rating* pode-se avaliar os *links* entre dois recursos r_i e r_j através

da Equação 2 (função *eval*) que avalia tanto os recursos nas pontas (r_i e r_j) quanto o *link* em si que é representado por representado por $edge(r_i, r_j)$. a função *eval* composta pelo produto das avaliações de r_i, r_j e $edge(r_i, r_j)$ em relação a todas as políticas definidas pelo serviço. Se um dos recursos não suportar uma das políticas definidas toda a avaliação é zerada, e em caso contrário o *score* do enlace aumenta. Essa equação aplica uma raiz de $3 \times P$ como forma de normalização inspirada no cálculo de média geométrica. P é o número de políticas definidas pelo serviço e utilizadas na avaliação. O valor final da avaliação está contido entre 0 e *MAX_RATE*, de maneira similar à função *rating*.

$$eval(r_i, r_j) = \sqrt[3\times P]{\prod_{x=1}^{P} rating(r_i, p_x) * rating(r_j, p_x) * rating(edge(r_i, r_j), p_x)} \quad (2)$$

A função que calcula os pesos das arestas para cálculo do melhor caminho é definida pela Equação 3 (função *weight*). Nela, r_i e r_j são dois recursos dentre os disponíveis na rede para aprovisionamento do serviço de acordo com as restrições definidas pelo pré-processamento das políticas. Caso não exista ligação entre os recursos, ou a avaliação deles seja igual a zero, ou ambos não possam ser utilizados para a comutação do tráfego o peso da aresta na matriz de adjacências é ∞. Caso contrário, o peso pode ser calculado através da função *eval* em relação ao enlace só que invertida: quanto maior a avaliação menor o peso. Para isso é subtraída a avaliação do enlace do valor máximo de avaliação definido pela variável *MAX_RATE*. Adicionalmente é utilizada a variável *HOP_COST* iniciado com o valor 1 e otimizável com TSLE, como maneira de evitar "saltos" desnecessários no cálculo de rotas.

$$weight(r_i, r_j) = \begin{cases} \infty \ , \ \begin{cases} \text{se ambos } r_i \text{ e } r_j \text{ não são dispositivos de comutação} \\ \text{se não há ligação entre } r_i \text{ e } r_j \\ \text{se } eval(r_i, r_j) \leq 0 \end{cases} \\ HOP\_COST + (MAX\_RATE - eval(r_i, r_j)) \ , \text{ em caso contrário} \end{cases} \quad (3)$$

O melhor caminho é calculado através de um método inspirado no algoritmo de *Dijkstra* que calcula o caminho mais curto (com menor peso de acordo com a função *weight*) mas com restrições em relação aos servidores de funções de rede. Por exemplo, se necessário para o aprovisionamento do serviço o caminho deve passar por, pelo menos, um servidor de funções de rede para que seja utilizada uma função específica para processamento no plano de dados.

6.2.3 *Topology Collector Entity* (TCoE)

A *Topology Collector Entity* (TCoE) – já apresentada na seção 5.1.4.1 – é o componente SONAr responsável pelo monitoramento e coleta de informações sobre a topologia de rede. Esse componente é primordial para as operações de *inicialização da rede* e *plug-and-play de*

recursos conforme apresentado nos fluxos definidos na seção 5.2. A implementação desses fluxos no que se refere ao comportamento da TCoE foi realizada de forma programática com base na troca de eventos via NEM. O código foi desenvolvido na linguagem de programação *JAVA 8* através do *framework Spring-Boot* versão 2.1.2 e pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/Collector-entities/topology-Collector-entity>.

A TCoE faz consultas periódicas na topologia através do NIM-Client (seção 6.1.3.3) que na implementação utilizada nesta tese utiliza o protocolo SNMP para consulta de dados diretamente nos dispositivos via LLDP-MIB. A TCoE também suporta integração com o Controlador SDN, de maneira a validar a conexão dos dispositivos recém-configurados ou importar informações de topologia.

Na operação de inicialização da rede a TCoE tem a missão de descobrir os recursos disponíveis na rede, autenticá-los e validá-los antes da execução da SCE. Para isso a rede inicia em um estado "parado" onde todas as demais entidades aguardam para executar. A TCoE autoriza os recursos descobertos através da troca de certificados digitais ou usuário/senha de acordo com a variável *RESOURCE_AUTH_METHOD*, de maneira que recursos não autorizados sejam removidos da topologia. A validação ocorre de acordo com critérios também definidos por configuração, como por exemplo: *i)* dispositivos devem formar um grafo conexo; *ii)* deve-se obter um número mínimo de dispositivos; *iii)* não pode haver falha em nenhuma descoberta; *iv)* o processo não pode terminar com recursos desconhecidos; e, *v)* um número máximo de tentativas não foi ultrapassado. Ao final desse processo a TCoE notifica a SCE via NEM sobre a conclusão da etapa de descoberta de recursos para inicialização da rede.

Na operação de *plug-and-play* de recursos a TCoE atua em três momentos: ao receber a notificação do novo dispositivo plugado via AP (seção 6.1.1.5), ao detectar um vizinho desconhecido conectado a um dispositivo de comutação, e ao receber a notificação de que há um novo dispositivo acessível via SCE (seção 6.2.2). Nos primeiros dois casos o recurso plugado pode ainda não ser acessível, por isso a missão da TCoE é descobrir o ponto de interconexão que indica a porta para a qual um fluxo (ou rota) deve ser configurado de maneira a permitir o acesso ao novo recurso. Ao identificar o ponto de interconexão a TCoE notifica a SCE via NEM, que por sua vez realiza a configuração. No terceiro caso o recurso já é acessível, por isso cabe à TCoE autorizar o novo recurso e realizar a descoberta para coletar detalhes como portas, memória, processadores e funcionalidades. Nesse momento podem ser identificados também recursos lógicos como aplicações e funções de rede. Caso o novo recurso não passe pelo procedimento de autorização ele é registrado em uma *blacklist* temporariamente para prevenir novas tentativas de *plug-and-play*, e as configurações de rotas/fluxos são removidas para prevenir ataques.

6.2.4 *Network Service Broker* (NSB)

O *Network Service Broker* (NSB) – já apresentado na seção 5.1.5.3 – é o componente SONAr responsável prover mecanismos para gerenciamento de serviços de comunicação para usuários e administradores da rede. Na plataforma proposta nesta tese o NSB foi desenvolvido na linguagem de programação *JAVA 8* através do *framework Spring-Boot* versão 2.1.2. O código fonte dessa aplicação pode ser acessado pelo link <http://gitlab.facom.ufu.br/sonar/sonar-all/tree/master/custom-components/network-service-broker>.

Os *web-services* disponibilizados pelo NSB foram construídos com *Spring-MVC* em uma abordagem REST, e autenticados através do *Spring-Security* configurado com método *BASIC* (baseado em usuário e senha). Os serviços recebem um JSON para descrição do serviço de acordo com um *schema* baseado na IBSO.

O NSB suporta as operações de criação, remoção, consulta, bloqueio e desbloqueio de serviços. De acordo com a operação o NSB notifica os componentes SONAr com tópicos específicos via NEM. Após o processamento das solicitações a SCE notifica a rede sobre o resultado através da publicação de eventos. Todas as funções disponibilizadas pelo NSB são assíncronas, exceto a de consulta. O NSB, no entanto, faz verificações antes de publicar os eventos para atestar se a representação de serviço é válida, se o serviço realmente existe (exceto na criação), e se a SCE está disponível. Caso alguma das verificações falhe, uma resposta é retornada de forma síncrona.

6.2.5 *Control Plane Interceptor* (CPI)

O *Control Plane Interceptor* (CPI) – já apresentado de maneira conceitual na seção 5.1.5.7 e prática na seção 6.1.2.3 – é um componente chave para a implementação dos fluxos de configuração por permitir a interceptação de mensagens trocadas entre os componentes de controle e os recursos de infraestrutura. Este componente foi refinado no desenvolvimento da SeCoMP de maneira a permitir o uso do CPI como descrito no estudo de caso apresentado na seção 5.2.

Durante a configuração dos *switches OpenFlow* a SCE define o CPI como controlador ao invés do próprio controlador SDN. Sendo assim, a primeira parte do CPI consiste em atuar como um *proxy*: aceitar requisições dos *switches* e abrir requisições para o controlador. Caso esse *proxy* não seja transparente, como é o caso da implementação utilizada no *framework* apresentado nesta tese, o controlador SDN registra todos os *switches* com o mesmo IP, o que é facilmente contornável através do uso do *datapath* como mecanismos de identificação.

Depois de estabelecida a comunicação entre os *switches* e os controladores através do CPI, todas as primitivas de controle de ambas as direções passam pelo CPI que antes de retransmiti-las filtra mensagens de interesse e envia para os componentes SONAr via NEM. É o caso das primitivas do protocolo DHCP enviadas ao plugar um novo disposi-

tivo (antes da implantação de regras pra acessar o novo recurso), que são repassadas pelo *switch OpenFlow*. O CPI envia essas primitivas para o AP (seção 6.1.1.5) e evita que elas cheguem ao controlador. As respostas provenientes do AP são enviadas via chamada de procedimento remoto (RPC) pelo CPI (utilizando troca de mensagens com o NEM) como se o próprio controlador as estivesse enviando. O CPI também pode filtrar mensagens consideradas indevidas (com tentativas de ataques por exemplo) e pode coletar métricas, informações de topologia e analisar as configurações realizadas pelas aplicações no controlador.

Capítulo 7

Experimentos e Análise dos Resultados

Este capítulo apresenta detalhes sobre os experimentos realizados com a *Self-Configuration and Management Platform* (SeCoMP) desenvolvida neste trabalho e apresentada na seção 6.2. A SeCoMP estende o SONAr-Framework para a automação de operações de auto-configuração em redes SDN/*OpenFlow* e foi construída de maneira a viabilizar os experimentos descritos na seção 7.4. Estes experimentos visam comprovar a viabilidade de utilização dos conceitos propostos pela SONAr para a automação do gerenciamento de rede e demonstrar os ganhos obtidos com a abordagem de auto-configuração implementada pela SeCoMP.

7.1 Método para a Avaliação

O método utilizado para avaliação sobre a viabilidade da SONAr e sobre os ganhos da SeCoMP na automação das operações de auto-configuração consistem em:

- Definição de cenários de testes (seção 7.2) específicos para cada uma das operações de auto-configuração trabalhadas nesta tese;
- Construção de uma infraestrutura de rede emulada (seção 7.3) que permita a abstração dos cenários de testes, implantação da SeCoMP e realização dos experimentos;
- Realização dos experimentos e coleta de dados para análise dos resultados de acordo com os cenários de teste (seção 7.4);
- Interpretação e análise dos resultados obtidos com os experimentos (seção 7.5).

Neste trabalho são apresentadas comparações entre diferentes técnicas disponíveis na SeCoMP para a automação das operações de auto-configuração.

7.2 Cenários de Experimentação

Esta seção apresenta os cenários de testes envolvendo operações de configuração em redes SDN/*OpenFlow* automatizadas através da SeCoMP.

7.2.1 Cenário de Inicialização de uma Rede SDN/*OpenFlow*

Descrição

Este cenário consiste em iniciar uma rede de computadores composta por dispositivos de comutação com suporte ao protocolo *OpenFlow*, servidores para aprovisionamento de componentes de controle/gerenciamento, e *hosts* genéricos. Os dispositivos dessa rede possuem apenas configurações iniciais como o suporte aos protocolos DHCP, LLDP, SNMP, ARP e STP. Não há nenhum componente de controle ou de gerenciamento aprovisionado na rede. Os *switches* operam apenas com o comportamento de *learning-switch* suportado pelo protocolo ARP pois não há nenhuma rota ou fluxo pré-configurado.

Resultado Esperado

Espera-se que ao final da execução:

- ❏ um ou mais controladores com suporte ao protocolo *OpenFlow* tenham sido aprovisionados;
- ❏ todos os dispositivos de comutação tenham sido configurados e estejam conectados a pelo menos um controlador;
- ❏ e, os fluxos de controle e de gerenciamento tenham sido devidamente calculados e configurados (de maneira ótima e livres de *loops* ou *black-holes*).

Procedimento

Para realizar esse cenário através da SeCoMP deve-se:

- ❏ iniciar todos os recursos endereçáveis (servidores, *hosts* e dispositivos de comutação); e,
- ❏ executar o procedimento de inicialização caso ele não tenha sido executado automaticamente ao iniciar o servidor (de acordo com a configuração).

7.2.2 Cenário de *Plug-and-Play* de *switches OpenFlow*

Descrição

Este cenário consiste em detectar e configurar novos *switches* automaticamente à medida que eles se tornam disponíveis (são inicializados e conectados à topologia). Para

isso, é pré-requisito que uma rede como a definida no cenário anterior já esteja operacional, ou seja, que componentes de controle e gerenciamento tenham sido aprovisionados, que os dispositivos de comutação tenham sido configurados, e que a comunicação através da malha de rede seja possível. O novo dispositivo possui apenas configurações iniciais como suporte aos protocolos DHCP, LLDP, SNMP, ARP e STP. Além disso, não existem rotas/fluxos pré-configuradas para acessar o novo dispositivo antes da execução do *plug-and-play*.

Resultado Esperado

Espera-se que ao final da execução:

- ❏ o novo dispositivo tenha sido configurado e esteja conectado a um controlador;
- ❏ e, que os fluxos de controle/gerenciamento tenham sido reconfigurados para evitar *loops* e garantir a conectividade.

Procedimento

Para realizar esse cenário através da SeCoMP deve-se:

- ❏ iniciar a rede através dos procedimentos definidos no cenário anterior;
- ❏ implantar/aprovisionar um novo dispositivo; e,
- ❏ conectar o dispositivo à topologia e iniciá-lo.

7.2.3 Cenário de Aprovisionamento de serviço para comunicação de VoIP

Descrição

Este cenário consiste em configurar um serviço que permita a comunicação de uma aplicação de VoIP entre dois sistemas finais com os requisitos de garantia de banda e latência mínima. Para isso, é pré-requisito que uma rede como a definida nos cenários anteriores já esteja operacional. A comunicação VoIP só deve iniciar depois que o serviço for aprovisionado. Por fim, o serviço deve funcionar mesmo que exista concorrência por parte de outras aplicações que demandem uma maior largura de banda.

Resultado Esperado

Espera-se que ao final da execução uma comunicação VoIP entre dois *hosts* específicos na rede seja possível independente das demais comunicações concorrentes.

Procedimento

Para realizar esse cenário através da SeCoMP deve-se:

- ❏ iniciar a rede através dos procedimentos definidos no primeiro cenário;
- ❏ definir de maneira formal o serviço a ser aprovisionado;
- ❏ solicitar o aprovisionamento do serviço via NSB;
- ❏ iniciar as aplicações nos *hosts* e realizar a comunicação VoIP;

7.3 Infraestrutura de Experimentação

A infraestrutura de rede utilizada nos experimentos foi emulada com GNS3 versão 2.1.16 (NEUMANN, 2014), um *software* amplamente utilizado na indústria e academia capaz de emular redes de computadores através de máquinas virtuais, contêineres *Docker* e imagens de dispositivos reais de fabricantes como CISCO, HP e Juniper. No GNS3 os elementos de rede emulados são chamados de *appliances*, e podem ser customizados ou baixados da loja do GNS3 (acessível pelo link <https://www.gns3.com/marketplace/appliances>).

Os *appliances* customizados utilizados nesta tese estão disponíveis em <http://gitlab.facom.ufu.br/sonar/gns3/tree/master/appliances>. Dentre eles destacam-se:

- ❏ *Switch-OVS*: utiliza uma imagem de contêiner *Docker* customizada que pode ser acessada pelo link <http://gitlab.facom.ufu.br/sonar/docker/blob/master/switch/openvswitch/v4/Dockerfile>. Essa imagem tem como base o *alpine* 3.9, uma distribuição de *Linux* pequena muito utilizada em sistemas embarcados, e utiliza o *Open vSwitch* versão 2.10.1, um *switch* virtual de código aberto com suporte ao protocolo *OpenFlow*. São utilizados também na composição dessa imagem o *net-snmp* versão 5.7.3, o *lldpd* versão 1.0.3 e o dhclient versão 4.3.3, responsáveis respectivamente pelo suporte aos protocolos SNMP, LLDP e DHCP. Por fim, é definido um *script* de inicialização (acessível pelo link <http://gitlab.facom.ufu.br/sonar/docker/blob/master/switch/openvswitch/v4/scripts/boot-normal.sh>), que cria a base OVSDB (base de dados do *Open vSwitch*), inicia o ovsdb-server (sistema gerenciador da base OVSDB), inicia o ovs-vswitchd (*daemon* do *Open vSwitch*), cria uma *bridge* com todas as portas, habilita o protocolo STP em todas as portas, inicia o *net-snmp*, solicita endereço pra *bridge* via dhclient e inicia o *lldpd*.

- ❏ *SONAr-Server*: utiliza uma imagem de máquina virtual para o *VirtualBox* versão 6.0 com *Ubuntu* Server 16.04, 8194 MB de RAM, 22 GB de HD e um processador com 8 núcleos operando a 2,00 GHz. No *SONAr-Server* foi instalado o JDK 1.8.0-151, o *net-snmp* 5.7.3 e o *lldpd* 1.0.3 (com suporte ao SNMP). Além disso, foi instalado o *Docker* versão 18.06.2-ce, que é o mecanismo pelo qual os componentes especificados

pelo SONAr-Framework (no qual a SeCoMP se baseia) são aprovisionados. Todas as imagens de componentes já vêm instaladas no servidor, o que torna a inicialização da rede mais rápida. Essas imagens, no entanto, podem ser recompiladas no próprio servidor ou atualizadas via *Docker Hub*, um repositório de imagens de contêineres próprio do *Docker*. Por fim, o *SONAr-Server* é disponibilizado com o Auto-Boot Manager (*script* e *jar* executável) responsável por iniciar uma rede SONAr.

- *Client-Host*: utiliza uma imagem de contêiner *Docker* que pode ser acessada pelo link <http://gitlab.facom.ufu.br/sonar/docker/blob/master/host/client-term/Dockerfile>. Essa imagem tem como base o *Ubuntu* versão 16.04, e inclui os utilitários: *iperf3* versão 3.0.11, *tcpdump* versão 4.9.2, *net-tools* versão 1.60.2, *telnet* versão 0.17 e *traceroute* versão 2.0.21.

A Figura 39 apresenta detalhes sobre o ambiente utilizado nos experimentos conduzidos pelo presente trabalho. A infraestrutura consiste em uma VM com *Ubuntu* versão 18.10, 16 GB de RAM, 40 GB de HD e um processador com 8 núcleos operando a 2,00 GHz na qual foram instalados o *Docker* versão 18.06.2-ce, *wireshark* versão 2.4.1 (um *software* para inspeção de tráfego de rede) e o GNS3 versão 2.1.16. No GNS3 foi criado um projeto (disponível em <http://gitlab.facom.ufu.br/sonar/gns3/tree/master/projects>) com uma rede similar à apresentada na Figura 39. No desenho de rede foram utilizados os *appliances* definidos anteriormente: *SONAr-Server*, *Switch-OVS* e *Client-Host*.

Os experimentos foram realizados em redes de diferentes proporções de maneira a simular desde cenários básicos de redes domésticas até cenários mais complexos de grandes ISPs. Assim, foram criados projetos no GNS3 com topologias com 1, 2, 4, 8, 16, 32, 64 e 128 *switches*. A Figura 40 apresenta a topologia utilizada no experimento com 32 *switches*. Em todos os cenários o SONAr-Server foi conectado ao primeiro switch (*ovsw01*). No cenário com 1 *switch*. apenas o *ovsw01* e o SONAr-Server foram alocados. Nos cenários com 2, 4 e 8 *switches*, os dispositivos adicionais (de 2 a 8) foram inseridos em um anel (*Anel01*) que foi ligado ao primeiro *switch*. Nos cenários com mais *switches* (16, 32, 64 e 128) foi criado um anel central com 6 *switches* (*Core*) ligado à um dos *switches* do primeiro anel, e os demais foram organizados em anéis de até no máximo 8 *switches* ligados ao anel central (e.g. *Anel02* e *Anel03*).

7.4 Especificação dos Experimentos

Foram realizados três experimentos com base nos cenários descritos na seção 7.2 e na infraestrutura proposta na seção 7.3 com o objetivo de provar a viabilidade da SeCoMP e os ganhos com a automação das operações de auto-configuração. Para tal, os componentes da SeCoMP foram adequados para suportar *Open vSwitches* (e protocolos específicos como o OVSDB), para operar integrados ao *ONOS* como Controlador SDN,

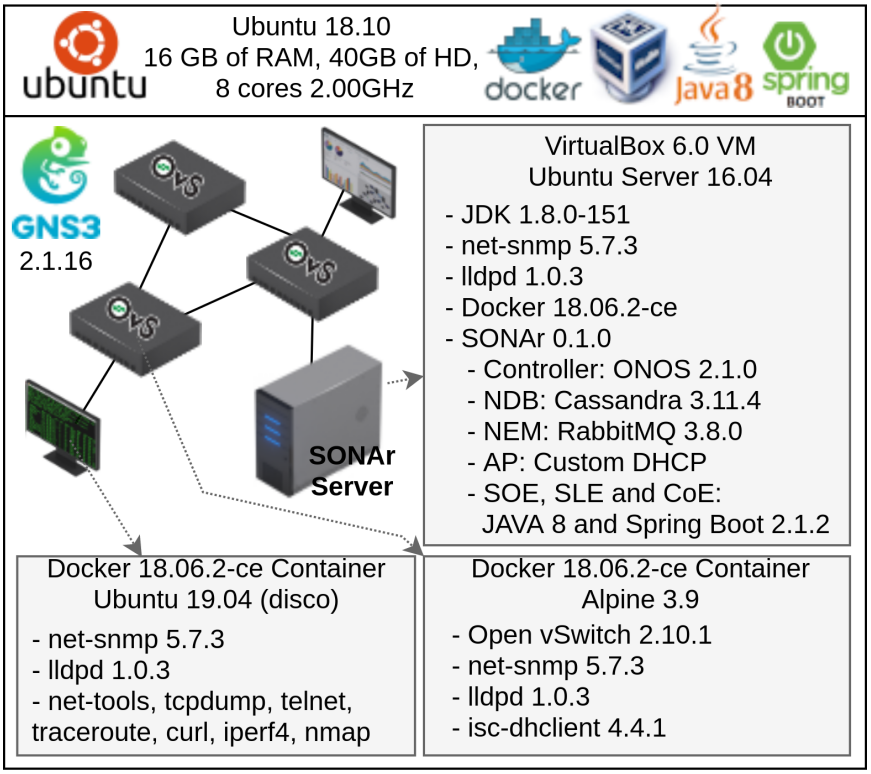


Figura 39 – Ambiente de Experimentação da SeCoMP.

Fonte: Elaborado pelo autor (2021).

e para suportarem o protocolo *OpenFlow* na definição de rotas/fluxos. Os experimentos conduzidos compreendem:

- ❏ Inicialização de uma Rede SDN/*OpenFlow*: visa demonstrar como a SONAr pode ser utilizado para automação do processo de *bootstrapping* de uma rede SDN;
- ❏ *Plug-and-Play* de *switches OpenFlow*: demonstra como a SONAr provê suporte à extensão da infraestrutura de rede de forma automática quando um novo switch *OpenFlow* se torna disponível;
- ❏ Aprovisionamento de serviço para comunicação de VoIP: apresenta um exemplo de aplicação da SONAr na configuração de serviços com política de garantia de banda e latência mínima de forma automática.

7.4.1 Experimento de Inicialização de uma Rede SDN/*OpenFlow*

A inicialização de uma rede SDN/*OpenFlow* envolve o aprovisionamento de um ou mais controladores, a configuração de *switches* com endereços e configurações iniciais, o cálculo de rotas de fluxo para garantia da comunicação entre o controlador e os *switches*, e por fim, a configuração destes fluxos via protocolo *OpenFlow*. Em uma rede com a

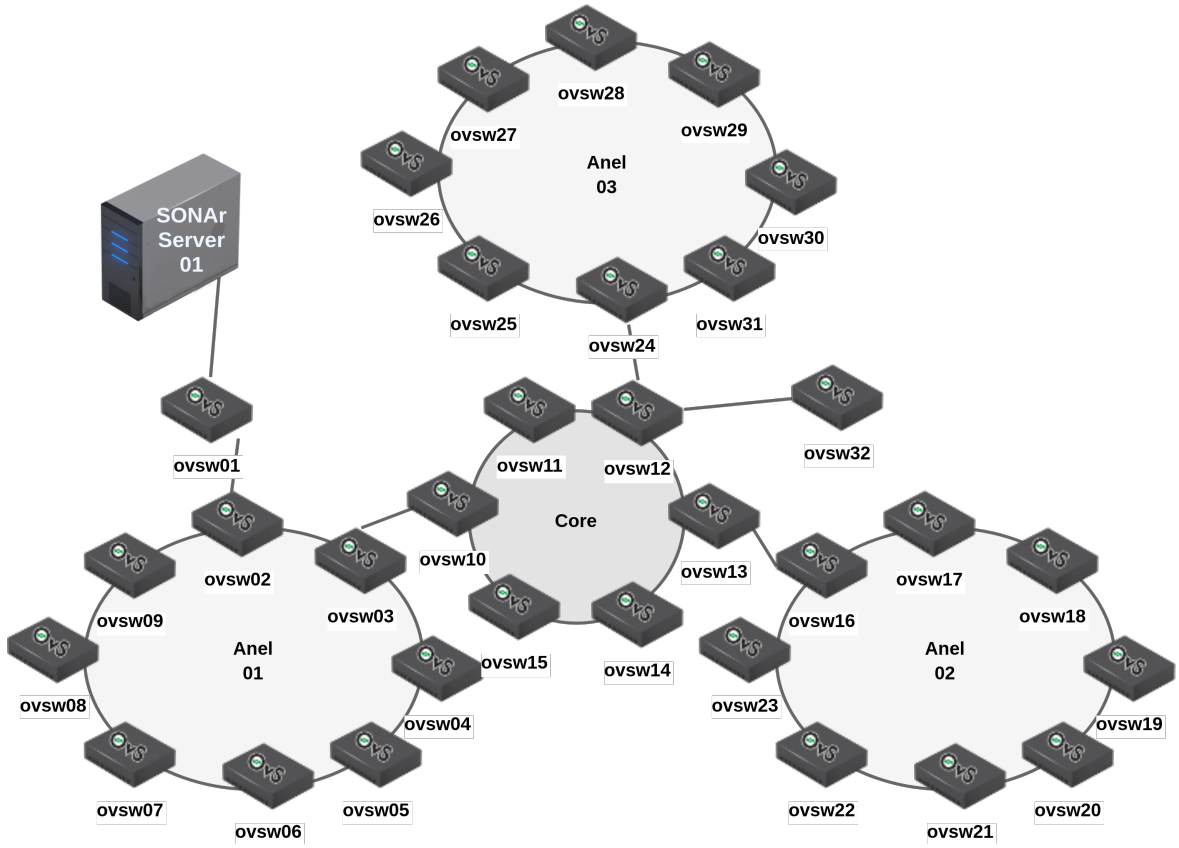


Figura 40 – Topologia utilizada no experimento com 32 switches.

Fonte: Elaborado pelo autor (2021).

arquitetura SONAr, é necessário também aprovisionar os componentes de gerenciamento (e.g. entidades auto-organizadoras, entidades coletoras e entidades de auto-aprendizado) de maneira a automatizar esse processo de inicialização.

Para validar a capacidade da SeCoMP de inicializar uma rede com essas características foi proposto o presente experimento, no qual redes de diferentes proporções (1, 2, 4, 8, 16, 32, 64 e 128 *switches*) foram emulados com o GNS3. O processo de inicialização foi dividido em etapas (descoberta, roteamento, configuração e atraso/processamento) e testes foram conduzidos de maneira a analisar o tempo gasto em cada etapa. Os resultados foram coletados com uma aplicação específica desenvolvida em *JAVA 8* com o SONAr SDK (conjunto de bibliotecas do SONAr-Framework descritas na seção 6.1.3), através da análise de eventos relacionados à operação de inicialização da rede via NEM. Os dados foram analisados e plotados através do *Google Sheets*.

Durante os experimentos algumas alterações nos componentes de gerenciamento foram necessárias de maneira a melhorar o desempenho ou prover novas funcionalidades, o que gerou três sub-experimentos distintos:

- **Experimento 1.1** : *Boot em Ordem Prefixa* – cenário no qual os fluxos de controle calculados com um algoritmo de 'Árvore Geradora Mínima' são configurados a par-

tir dos nós mais próximos (raiz da árvore – *switch* com acesso direto ao Servidor SONAr);

- **Experimento 1.2** : *Boot em Ordem Posfixa* – cenário no qual os fluxos de controle calculados com um algoritmo de 'Árvore Geradora Mínima' são configurados a partir dos nós mais distantes (folhas da árvore); e,
- **Experimento 1.3** : *Boot em Ordem Posfixa com CPI* – cenário no qual o componente interceptor do plano de controle é utilizado e os fluxos de controle calculados com um algoritmo de 'Árvore Geradora Mínima' são configurados a partir dos nós mais distantes (folhas da árvore).

A divisão deste experimento em três visa identificar o *overhead* proveniente da utilização do CPI, e os ganhos em relação à ordem da configuração dos *switches*, o que ocorre pelo fato de que ao configurar um determinado *switch* é necessário aguardar a implantação de todos os fluxos antes de prosseguir na configuração dos próximos *switches* acessíveis a partir do que está sendo configurado. Isso se deve ao fato de que ao configurar o protocolo *OpenFlow* na *bridge* do *switch* o comportamento padrão de *learning-switch* é desabilitado impedindo que dispositivos vizinhos sejam acessados sem que os fluxos tenham sido implantados. Ao começar pelos *switches* mais distantes da 'Arvore Geradora Mínima' que representa o fluxo de controle ('Ordem Posfixa') todos os *switches* podem ser acessados pelo comportamento *learning-switch* padrão, sem necessidade de aguardar a implantação dos fluxos *OpenFlow* como ocorre na 'Ordem Prefixa'.

Em cada sub-experimento foram realizados pelo menos oito testes (um para cada rede). Em cada teste a configuração dos *switches* foi zerada e os componentes foram removidos do servidor, de maneira a testar a inicialização em redes do zero.

Cada teste consiste na sequência de passos abaixo:

- *Passo 1* – Criar a topologia de rede no GNS3: criar/adicionar os contêineres com a imagem do *switch* modificado com base no *Open vSwitch*; criar/adicionar uma máquina virtual com uma imagem do Servidor SONAr; e, ligar as portas entre os recursos (*switches* e *servidor*) criando uma das topologias descritas na seção 7.3;
- *Passo 2* – Iniciar o Servidor SONAr e o ABM: iniciar a máquina virtual que representa o Servidor SONAr e executar o *script* de inicialização do ABM (seção 6.2.1) que inicia o C-NIM (seção 6.1.1.4), o NDP (seção 6.1.1.3), o RIP (seção 6.1.1.2), e a própria aplicação ABM que por sua vez inicia a TCoE (seção 6.2.3), a SCE (seção 6.2.2), o CPI (seção 6.1.2.3), o NSB (seção 6.2.4), o AP (seção 6.1.1.5) e o Controlador SDN (seção 6.1.1.6);
- *Passo 3* – Executar a aplicação de coleta de resultados: essa aplicação se inscreve nos tópicos que indicam o início e a finalização das etapas de descoberta, roteamento e configuração específicas da inicialização de rede;

- *Passo 4* – Iniciar todos os *switches* da topologia: iniciar os contêineres com os *switches* via GNS3;
- *Passo 5* – Aguardar a execução da inicialização de rede e coletar os resultados: a inicialização da rede ocorre de maneira automática após a inicialização dos *switches*.

7.4.2 Experimento de *Plug-and-Play* de *switches OpenFlow*

A operação de *plug-and-play* de um *switch* inicia pela identificação do evento de conexão, o que pode se dar de maneira ativa a partir de procedimentos de descoberta periódicos, ou passiva a partir de eventos ou requisições (e.g. DHCP). Em alguns casos, como na utilização de *OpenFlow*, o procedimento de descoberta só consegue alcançar o novo *switch* depois de terem sido criados fluxos que permitam essa comunicação. Por isso, nestes casos deve-se inicialmente identificar o ponto de interconexão do novo *switch*, calcular rotas que permitam o tráfego de dados até este ponto, e implantar os fluxos nos dispositivos já controlados de acordo com as rotas escolhidas. Depois de se tornar acessível o novo *switch* passa por um procedimento de descoberta onde são coletadas características e capacidades. Já no estado descoberto o *switch* passa por um procedimento de configuração para habilitar o uso do protocolo *OpenFlow* na *bridge*, setar o *endpoint* do controlador mais adequado (ou CPI), remover a configuração de controle de *loop* (e.g. STP) e aplicar configurações básicas. Por fim, depois de se conectar ao controlador o novo *switch* se torna disponível para a rede e a operação é concluída.

De maneira a provar como uma rede SONAr suporta naturalmente este operação foi proposto um experimento com redes de diferentes proporções (1, 2, 4, 8, 16, 32, 64 e 128 *switches*) emuladas com GNS3. Além disso, o procedimento foi dividido em diversas etapas: inicialização do *switch*, cálculo de rota, configuração de rota, descoberta e configuração. Foram conduzidos testes para verificar o tempo gasto em cada etapa durante o *plug-and-play* de um recurso. Esses tempos foram obtidos através de uma aplicação específica desenvolvida com o SONAr SDK (seção 6.1.3), que se inscreve em tópicos relacionados à inicialização e finalização dessas etapas. Os dados foram analisados e plotados como gráficos através do *Google Sheets*.

Algumas alterações foram realizadas na SeCoMP durante o experimento de maneira melhorar o desempenho ou testar a solução em situações específicas. O objetivo dessa divisão é verificar o *overhead* em decorrência da utilização do CPI, e os ganhos em desempenho (tempo) por postergar a etapa de descoberta. Essa abordagem foi chamada neste trabalho de "descoberta atrasada", e define que o procedimento de descoberta deve ocorrer depois de realizada a configuração do novo dispositivo, de maneira a permitir a execução do LLDP. Os sub-experimentos realizados são descritos abaixo:

- **Experimento 2.1** : *Plug-and-Play sem interceptor e com "descoberta padrão"* – cenário no qual o CPI não é utilizado, o que dificulta a detecção de um novo *switch*,

uma vez que as primitivas DHCP não são interceptadas;

- **Experimento 2.2** : *Plug-and-Play com interceptor e com "descoberta padrão"* – cenário no qual o CPI é utilizado, o que facilita a detecção de um novo *switch* através da interceptação de primitivas DHCP;

- **Experimento 2.3** : *Plug-and-Play com interceptor e com "descoberta atrasada"* – cenário com CPI é utilizado (o que facilita a detecção de um novo *switch*) e com a tática de "descoberta atrasada" (que diminui o número de tentativas falhas em decorrência do tempo de processamento do LLDP).

Em cada sub-experimento foram realizados pelo menos 8 testes (um para cada dimensão rede). Em cada teste foi utilizado um novo *switch*, de maneira a testar o *plug-and-play* de um *switch* do zero. A distância em saltos na qual o novo *switch* foi aprovisionado é igual a $(\log_2 D) + 1$ onde D é o número de dispositivos na topologia.

Os testes consistem na sequência de passos seguinte:

- *Passo 1* – Criar topologia e iniciar rede de acordo com os passos 1, 2 e 4 da inicialização de rede (seção 7.4.1): a rede deve estar em estado operacional, com todos os componentes de controle e gerenciamento executando;

- *Passo 2* – Executar a aplicação de coleta de resultados: essa aplicação se inscreve nos tópicos que indicam o início e a finalização das etapas de inicialização, cálculo de rota, configuração de rota, descoberta e configuração de um *switch*;

- *Passo 3* – Aprovisionar um novo *switch* na topologia: criar um novo *switch* como um contêiner via GNS3, ligá-lo a uma distância específica do Servidor SONAr (em *hops*) de acordo com o teste, e iniciá-lo;

- *Passo 4* – Aguardar a execução do *plug-and-play* do novo *switch* e coletar os resultados: a detecção e configuração do novo *switch* ocorre de maneira automática.

7.4.3 Experimento de Aprovisionamento de serviço para comunicação de VoIP

O aprovisionamento de um serviço requer uma forma de representação e um mecanismo para a sua interpretação e configuração. Além disso deve ser possível identificar as instâncias de comunicações (fluxos) alvos desse serviço. Esse processo normalmente é realizado de forma manual por um técnico ou administrador de rede, mas a SONAr prevê componentes específicos para receber as solicitações (NSB - seções 5.1.5.3 e 6.2.4) e para interpretar/configurar o serviço sobre a infraestrutura (SCE - seções 5.1.2.1 e 6.2.2) inspirados pelo IBSPM que descreve componentes conceituais básicos para o aprovisionamento de IBSs.

Ao considerar a abordagem de representação de um IBS proposta pela IBSRM (seção 4.1.1) para abstração de um serviço pra comunicação VoIP é necessário definir pelo menos uma política: a de garantia de banda (IBSM/RNF1 – seção 4.1.1.2). Além disso pode-se filtrar os fluxos através de características da camada de transporte tradicionalmente relacionadas ao VoIP, por exemplo protocolo UDP e *range* de portas 16384-32768. Um serviço VoIP normalmente envolve uma comunicação *unicast* intra/inter-domain (IBSM/RF01 e IBSM/RF02 – seção 4.1.1.2). No presente experimento o serviço aprovisionado foi descrito de maneira formal (FDIBS – seção 4.1.1.5) através das funções, políticas e filtros definidos com as restrições/características citadas acima.

Esse experimento foi realizado com uma topologia simples com três *switches* abstraídos como contêineres através de uma imagem modificada de *Open vSwitch*. Foram ligados a estes três *switches* uma máquina virtual com uma imagem de Servidor SONAr, e outros quatro contêineres como hospedeiros, sendo dois para uma comunicação com alto consumo de banda de conteúdo do tipo *Torrent*, e outros dois para uma comunicação de VoIP. O objetivo do experimento é verificar a qualidade da comunicação de uma aplicação VoIP entre os dois hospedeiros quando outra comunicação de uma aplicação *Torrent* começa a competir pelo mesmo enlace. A SONAr configura automaticamente políticas de QoS para garantir uma largura de banda mínima e manter a comunicação ativa de 64kbps (valor definido para VoIP). Nesse experimento foram realizados dois testes, um com uma abordagem sem nenhum tipo de QoS e outro com a SeCoMP que aplica uma política QoS automaticamente. Os principais ganhos da abordagem SONAr em relação à uma abordagem de configuração de QoS tradicional está na flexibilidade da definição de serviço em alto-nível e na automação da configuração.

Os testes consistem na sequência de passos seguinte:

- *Passo 1* – Criar topologia e iniciar rede de acordo com os passos 1, 2 e 4 da inicialização de rede (seção 7.4.1): a rede deve estar em estado operacional, com todos os componentes de controle e gerenciamento executando. No teste sem SONAr, a TCoE e a SCE não devem ser iniciados e os fluxos precisam ser criados diretamente na aplicação administrativa do controlador, já no teste com SONAr deve-se solicitar ao NSB a configuração de um serviço específico para o tratamento da comunicação VoIP;

- *Passo 2* – Aprovisionar os contêineres hospedeiros: criar, ligar à topologia e inicializar quatro contêineres criados uma imagem customizada com o *iperf3*;

- *Passo 3* – Executar o *iperf3* por 50 segundos como receptor no *host3* e como emissor de tráfego VoIP (RTP/UDP porta 16384 e ToS 0xC0) no *host1* com taxa de 64Kbps;

- *Passo 4* (simultaneamente ao Passo 3) – Aguardar 20 segundos e executar o *iperf3* por 30 segundos como receptor no *host4* e como emissor de tráfego *Torrent* (TCP

porta 6881) no *host2* com taxa de 1Gbps;

- *Passo 5* – Coletar os dados nas instancias de *iperf3* receptoras (*hosts* 3 e 4).

Os dados coletados no Passo 5 foram analisados/plotados através do *Google Sheets*.

7.5 Análise dos Resultados Experimentais

Esta seção apresenta os resultados obtidos em cada um dos experimentos (e sub-experimentos) apresentados na seção 7.4, bem como considerações sobre as técnicas, valores e comportamentos obtidos.

7.5.1 Resultados do Experimento de Inicialização de uma Rede SDN/*OpenFlow*

Como apresentado na seção 7.4.1 o experimento de inicialização foi dividido em outros três: *i)* com configuração em ordem prefixa (a partir dos nós mais próximos); *ii)* com configuração em ordem posfixa (a partir dos nós mais distantes); e, *iii)* com o CPI implantado entre os *switches* e o Controlador SDN e com configuração em ordem posfixa (a partir dos nós mais distantes). Em cada sub-experimento foram conduzidos testes com oito topologias de dimensões variadas (1, 2, 4, 8, 16, 32, 64 e 128 *switches*). Foram realizadas três execuções de testes por topologia em cada sub-experimento, o que corresponde a setenta e duas execuções no total (vinte e quatro por sub-experimento). Apenas a execução com melhor resultado em cada sub-experimento/topologia foi utilizada. Verificou-se uma variação média de 7.6% entre as execuções escolhidas e as demais descartadas.

Foram realizadas quatro medições por execução: *1)* 'Descoberta' (tempo gasto para descobrir e validar todos os *switches* na infraestrutura); *2)* 'Roteamento' (tempo gasto no cálculo dos melhores caminhos para acesso aos *switches* e estabelecimento dos fluxos de controle e gerenciamento); *3)* 'Configuração (tempo gasto para realizar a configuração dos *switches* via OVSDB e implantar os fluxos via Controlador SDN); e, *4)* 'Processamento/Atraso' (diferença de tempo entre a inicialização e a finalização do processo excluindo os tempos de Descoberta, Roteamento e Configuração).

A Figura 41 apresenta os resultados de todos os sub-experimentos em uma única visão e permite verificar o comportamento de crescimento linear alcançado com todas as abordagens. Pode-se observar os ganhos das abordagens 'posfixas' (Experimento 1.1 e 1.3) em relação à abordagem 'prefixa' (Experimento 1.2). Isso se deve ao fato de que com essa abordagem o componente de configuração (SCE) não precisa aguardar que os fluxos sejam configurados antes de prosseguir com configuração dos demais, o que ocorre pois existe uma relação de dependência entre o *switch* anterior e o posterior de maneira que os componentes de controle/gerenciamento sejam capazes de acessar o *switch* posterior.

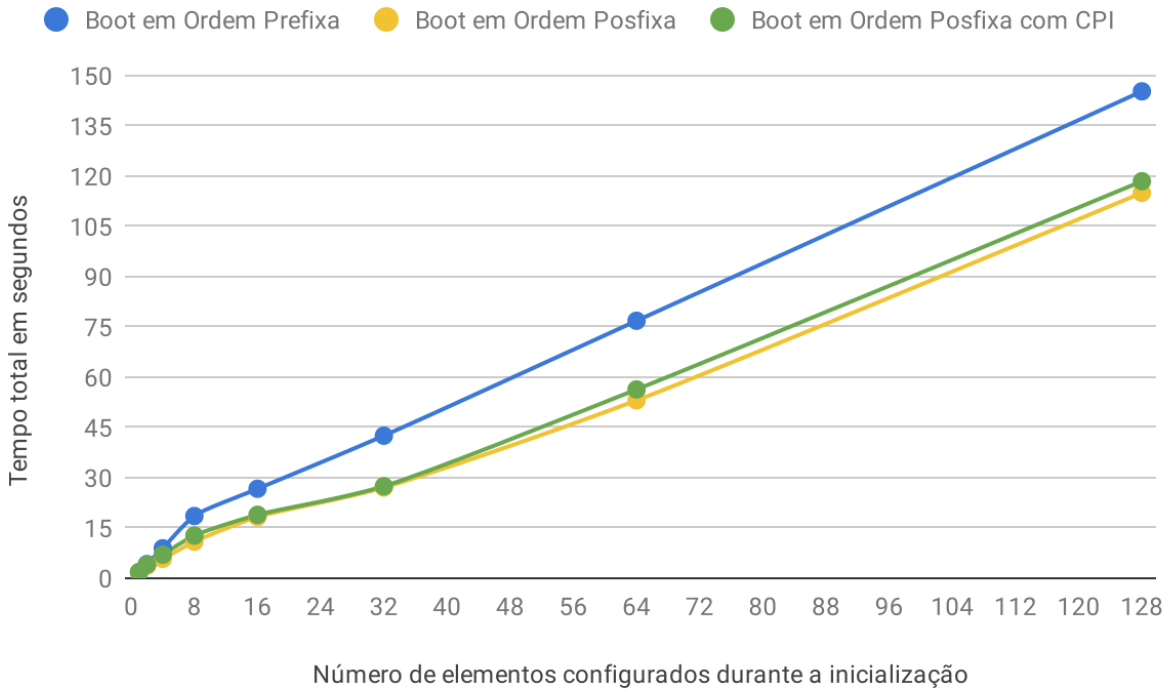


Figura 41 – Comparação entre todas as estratégias de inicialização da rede experimentadas.

Fonte: Elaborado pelo autor (2021).

Os resultados demonstram que a estratégia de configuração posfixa leva uma vantagem média ponderada pela quantidade de *switches* de 37.41% em relação à prefixa.

Ambas as estratégias de configuração, 'prefixa' e 'posfixa', baseiam-se no cálculo de uma árvore geradora mínima que representa a relação de dependências entre os elementos de rede. Sendo assim, quanto mais profunda for essa árvore (maior número de níveis) mais lento é o processo de inicialização de rede. Isso se torna mais perceptível na implementação da SeCoMP uma vez que o algoritmo implementado configura de maneira simultânea (com processamento *multithreading*) todos os elementos do mesmo nível. Mais níveis representam mais iterações no procedimento de configuração, e consequentemente, mais tempo de configuração. A topologia de rede escolhida para os experimentos tem uma organização em anel o que é usual para redes críticas[1] por garantir maior disponibilidade através da redundância de *links*. Por esse motivo, deve-se observar que alterações na disposição dos elementos da rede podem impactar nos resultados fornecidos nesta seção.

Pode-se observar também na Figura 41 o impacto no tempo total de inicialização da rede ao ser utilizado o CPI, um componente que opera como um *proxy* entre os *switches* e o Controlador SDN. O uso do CPI não representa uma vantagem para a operação de inicialização da rede, mas permite a melhoria de outras operações como o *plug-and-play* de dispositivos (auto-configuração) e a recuperação de controladores (auto-cura).

[1] e.g. redes de provedores de *Internet* e redes de *data centers*

A abordagem com CPI (Experimento 1.3) apresenta um tempo ligeiramente maior em todos os cenários em relação à abordagem sem CPI (Experimento 1.2). Esse acréscimo é esperado, pois toda a comunicação de controle passa pelo CPI que analisa a informação, notifica os componentes de gerenciamento e transmite ao Controlador. O comportamento da abordagem com CPI se manteve linear e não se distanciou muito do comportamento apresentado pela abordagem sem CPI, o que demonstra que o *overhead* introduzido pelo CPI é baixo.

A Figura 42 apresenta de maneira detalhada o tempo gasto em cada estágio da abordagem com configuração em ordem posfixa e com CPI (Experimento 1.3). Pode-se observar que a etapa de 'Configuração' é a que apresenta um comportamento de crescimento mais significativo em relação ao número de *switches* na topologia, o que é esperado tendo em vista que o processo envolve atividades mais demoradas como o acesso a cada *switch* de maneira individual e a configuração de fluxos por *switch*. Todavia, o crescimento do tempo necessário para a execução da rotina de configuração é linear, assim como o crescimento geral do tempo de inicialização apresentado exposto na Figura 41. O tempo de 'Roteamento' e o 'Atraso acumulado' são pequenos se comparados ao tempo de 'Configuração' em todos os cenários, especialmente nos com maior número de *switches* na topologia. Ambas as etapas apresentam um comportamento de crescimento linear, o que faz sentido ao se considerar que quanto mais *switches* mais lenta é a execução do algoritmo de roteamento e mais informação são trafegadas e impactam no atraso geral acumulado. Por fim, o tempo de 'Descoberta' é superior aos tempos de 'Atraso' e 'Roteamento' e inferior ao tempo de 'Configuração'. O crescimento de tempo apresentado nesta etapa é linear e tende a sofrer menos influência em relação ao número de *switches* na topologia devido ao processamento *multithread* utilizado pelo TCoE e pela forma como as topologias foram organizadas.

De maneira geral, o estágio de 'Configuração' é o que demanda mais tempo na inicialização da rede e é o estágio mais influenciado pela dimensão da infraestrutura. Como pode ser observado na Figura 43 que apresenta o percentual de tempo gasto em cada estágio do Experimento 1.3, o tempo de 'Configuração' representa cerca de 52,9% do tempo de inicialização, seguido pelo tempo de 'Descoberta' que representa 29.3%, tempo de 'Atraso/Processamento' que representa cerca de 16.6%, e por fim, tempo de 'Roteamento' que representa apenas 1.3% do tempo de inicialização. O tempo de 'Roteamento' é baixo pois a rotina utilizada não depende de nenhum processo assíncrono e nem de integrações com outros componentes.

A Figura 44 apresenta duas visões com uma análise do tempo de configuração por *switch* em todos os experimentos. Na primeira pode-se observar que o tempo é mais regular nas estratégias de configuração em ordem posfixa (Experimento 1.2 e 1.3) em todos os cenários. Na estratégia em ordem prefixa o tempo de configuração por *switch* é maior em todos os cenários, especialmente nos que utilizam uma topologia com 16 ou

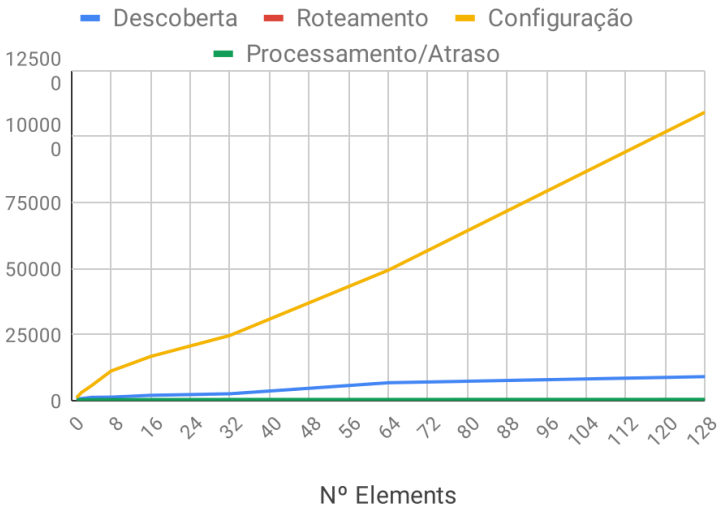


Figura 42 – Tempo gasto em cada estágio do Experimento 1.3.

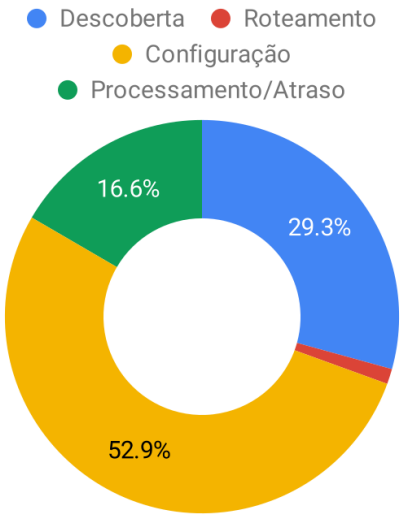


Figura 43 – Percentual de tempo gasto em cada estágio do Experimento 1.3

Fonte: Elaborado pelo autor (2021).

menos switches. Nas topologias acima de 64 *switches* o tempo de configuração é mais uniforme em todas as abordagens mas apresenta ligeira desvantagem para abordagem com configuração em ordem prefixa. Isso fica mais evidente na segunda visão, onde pode-se observar que as curva que representam as abordagens de configuração em ordem posfixa são próximas em cenários com topologias acima de 16 *switches*, enquanto a curva que representa a abordagem de configuração em ordem prefixa permanece superior em todos os cenários. A estabilização do crescimento do tempo de configuração por *switch* que pode ser observado com mais clareza no segundo gráfico da Figura 44 demonstra o comportamento de crescimento linear em função da dimensão da topologia para a solução de inicialização de rede implementada pela SeCoMP.

Os resultados apresentados acima atendem ao planejamento realizado na seção 7.2.1: *i)* um controlador SDN é automaticamente aprovisionamento pelo ABM; *ii)* todos os *switches* são descobertos com SNMP, configurados via OVSDB e conectados ao Controlador SDN via *OpenFlow*; e, *iii)* os fluxos de controle e de gerenciamento são automaticamente calculados e implantados pela SCE através da NBI do Controlador SDN. Verifica-se que o tempo total de inicialização alcançado nos experimentos é menor do que o tempo normalmente exigido por abordagens mais tradicionais que ainda se amparam em atividades técnicas manuais ou semiautomáticas realizadas por administradores de rede. Além disso, a automação da inicialização da rede garante maior padronização, eficiência e segurança, pois elimina os riscos inerentes à intervenção humana.

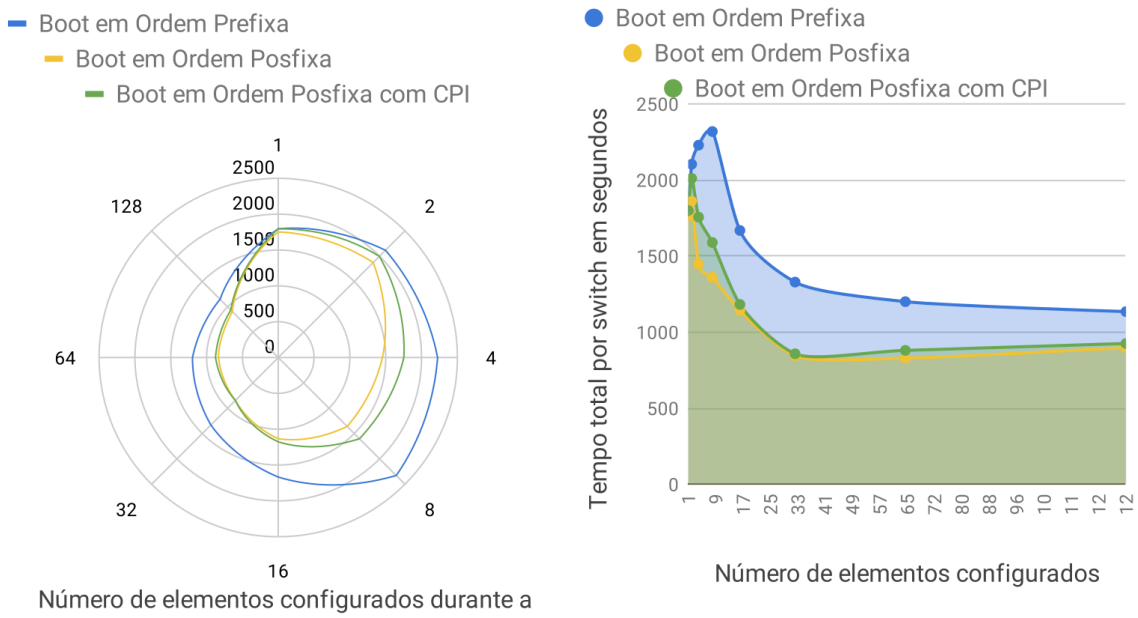


Figura 44 – Tempo de configuração acumulado por *switch*

Fonte: Elaborado pelo autor (2021).

7.5.2 Resultados do Experimento de *Plug-and-Play* de Dispositivos de Comutação

Como apresentado na seção 7.4.2 o experimento de inicialização foi dividido em outros três: *i)* sem interceptor; *ii)* com interceptor; e, *iii)* com o interceptor e com a estratégia de "descoberta atrasada". Em cada sub-experimento foram conduzidos testes envolvendo topologias de diferentes proporções (1, 2, 4, 8, 16, 32, 64 e 128 *switches*). Em cada combinação de sub-experimento e topologia foram realizadas três execuções, sendo que apenas a execução com menor tempo total foi utilizada. Excluindo-se as execuções com erros causados por *bugs* (que foram corrigidos durante a etapa de experimentação), não verificou-se grande variação entre os resultados obtidos com os sub-experimentos com interceptor (*ii* e *iii*). O experimento sem interceptor, por outro lado, apresentou maior variação (cerca de 5.3%) o que pode ser explicado pela necessidade de descoberta de toda a topologia, e por isso, existência de mais variáveis envolvidas no resultado final.

Foram realizadas seis medições por teste: *1)* 'Inicialização do *Switch*' (tempo gasto na execução do *script* de inicialização do *switch*: inicia com a inicialização do contêiner e finaliza com a designação de um endereço via DHCP); *2)* 'Atraso/Processamento' (indica o tempo gasto na comunicação entre os componentes, o tempo de reação aos eventos e o tempo de espera para execução de uma ação externa: compreende a diferença entre o tempo total de *plug-and-play* e a somatória de tempos das outras etapas); *3)* 'Cálculo de Rota' (tempo gasto no cálculo do melhor caminho para acessar o novo *switch*); *4)* 'Configuração de Rota' (tempo gasto para aplicar as novas regras de fluxo na topologia);

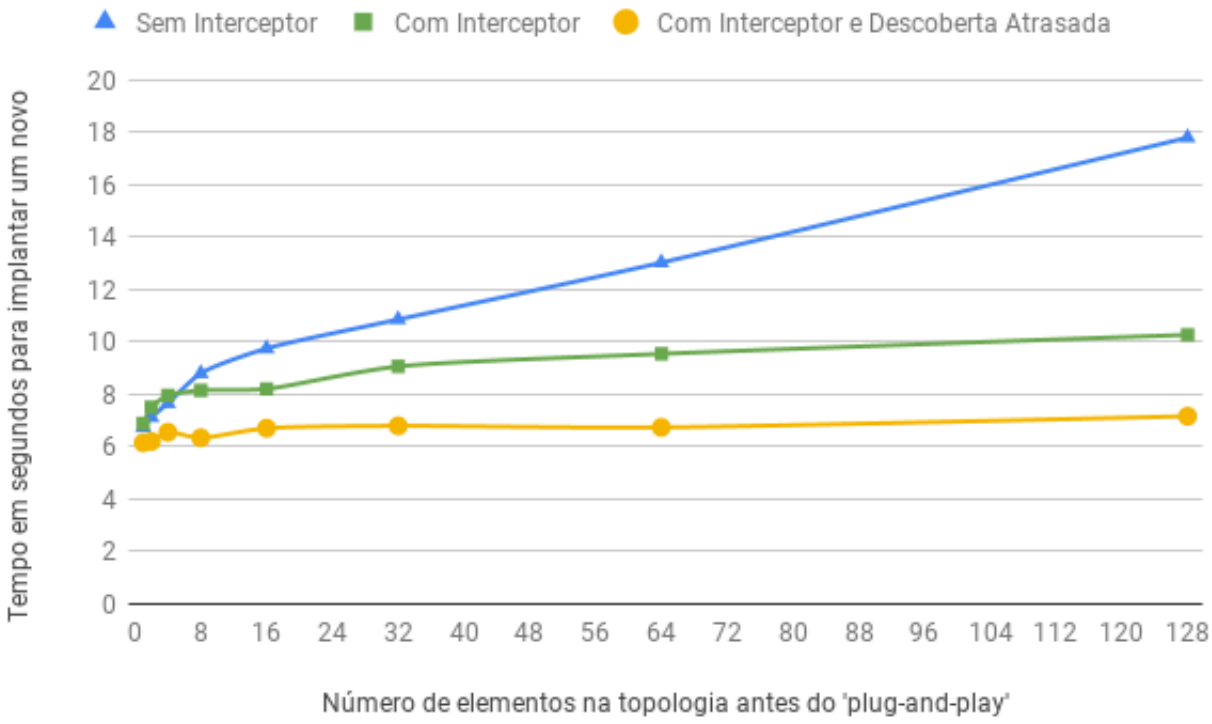


Figura 45 – Comparação entre todas as estratégias de *plug-and-play* experimentadas.

Fonte: Elaborado pelo autor (2021).

5) 'Descoberta' (tempo gasto para descobrir e validar o novo *switch*); e, *6)* 'Configuração' (tempo gasto para realizar a configuração do novo *switch* via OVSDB e implantar os fluxos via Controlador SDN).

A Figura 45 apresenta o tempo acumulado gasto na execução da operação de *plug-and-play* de um novo *switch* em redes de diferentes proporções com as três abordagens propostas (Experimento 2.1, 2.2 e 2.3). Pode-se verificar que a abordagem sem interceptor (Experimento 2.1) demanda um tempo consideravelmente superior se comparado às outras abordagens. Na comparação com a abordagem com interceptor e com estratégia de "descoberta atrasada" no pior cenário (128 *switches*) o tempo da abordagem sem interceptor é cerca de 2.4 vezes maior, o que se deve à dificuldade de detectar o ponto de interconexão do novo *switch*. As abordagens com interceptor (Experimento 2.2 e 2.3) levam vantagem pois a detecção do novo *switch* e a descoberta do ponto de interconexão são possíveis de maneira praticamente instantânea através da análise dos dados interceptados.

A linha que representa a abordagem sem interceptor no gráfico apresentado na Figura 45 demonstra uma maior tendência de crescimento à medida que mais nós são inseridos na topologia, contudo o crescimento é linear. Já as linhas que representam as abordagens com interceptor demonstram uma tendência de estabilização, pois como não é necessário realizar um processo de descoberta completo pra detecção do ponto de interconexão a dimensão da rede não impacta no tempo total da operação. Por fim, percebe-se que a

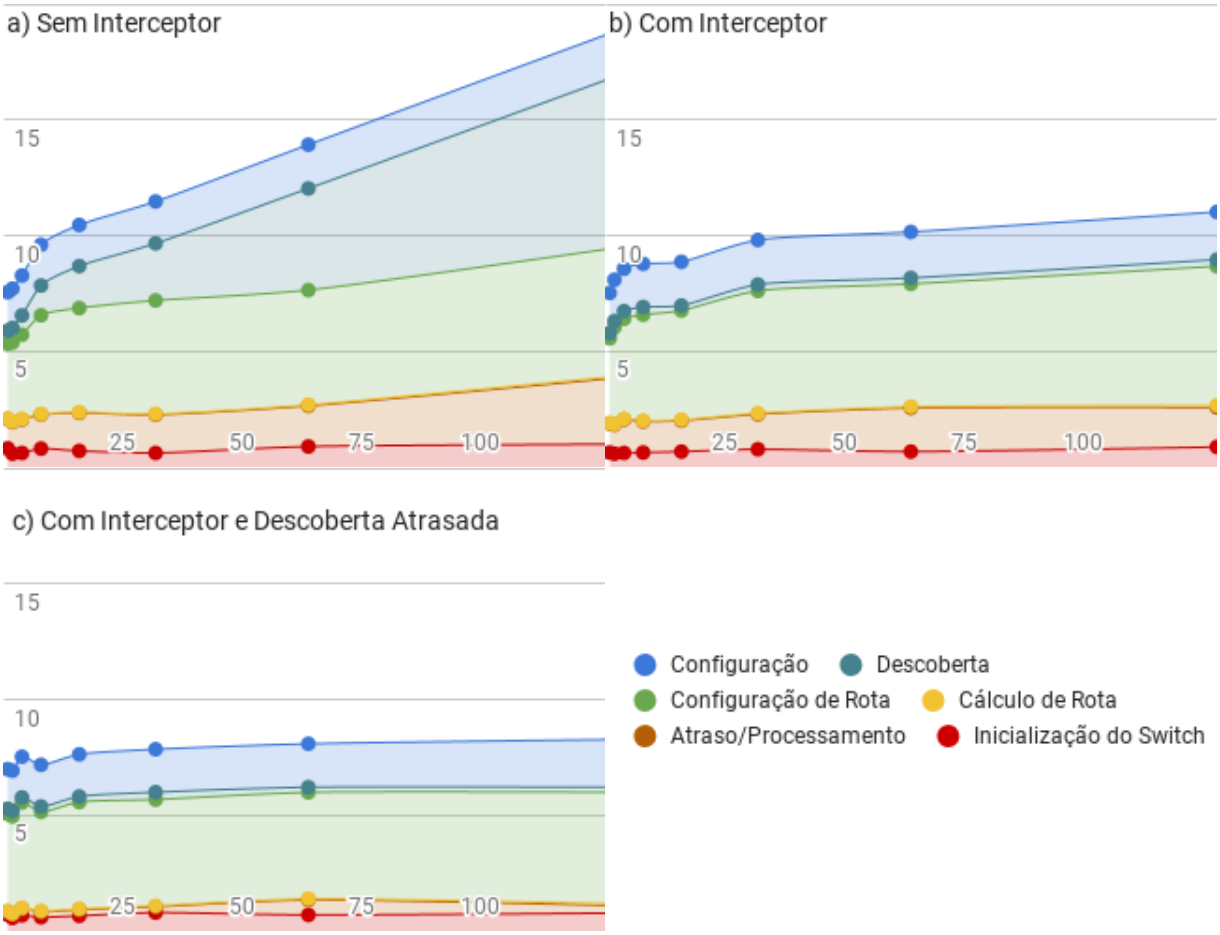


Figura 46 – Tempo gasto em cada estágio do *plug-and-play* de um novo *switch* em todos os experimentos.

Fonte: Elaborado pelo autor (2021).

estratégia de 'descoberta atrasada' que consiste em realizar o processo de 'Descoberta' depois do processo de 'Configuração' demonstra ser capaz de acelerar o *plug-and-play* pois diminui a chance de falha de 'Descoberta' em decorrência do atraso na inicialização do *snmpd* e *ovsdb-server* e execução dos protocolos ARP, STP e LLDP no novo *switch*.

A Figura 46 apresenta em detalhes o tempo gasto em cada estágio do processo de *plug-and-play* de um novo *switch* em todos os sub-experimentos: *a)* experimento 2.1 – abordagem sem interceptor; *b)* experimento 2.2 – abordagem com interceptor; e, *c)* experimento 2.3 – abordagem com interceptor e com estratégia de "descoberta atrasada". Os dados apresentados nesta figura demonstram de maneira clara que a abordagem *c* (Experimento 2.3) tem o melhor desempenho, e abordagem *a* (Experimento 2.1) o pior.

Considerando o gráfico *a* da Figura 46 que apresenta o tempo gasto em cada estágio do experimento sem interceptor, verifica-se que a etapa de 'Descoberta' apresenta um comportamento de crescimento linear em função do número de dispositivos na rede. Isso ocorre pois ao não utilizar o CPI o TCoE precisa vasculhar toda a topologia em busca do ponto de interconexão do *switch* plugado. Pode-se observar também que o tempo de

'Configuração de Rota' é afetado pela dimensão da infraestrutura, o que guarda relação com o fato de que na utilização da SeCoMP sem CPI a SCE precisa fazer requisições via NBI do Controlador pra detectar a conexão do novo *switch* (depois de realizado o *handshake* do protocolo *OpenFlow*), e para isso são realizadas múltiplas requisições em uma estratégia de "espera ocupada" sendo que a cada requisição são retornados dados de toda a topologia.

Nos resultados apresentados de maneira visual no gráfico *b* da Figura 46 verifica-se que a etapa de 'Descoberta' é realizada de maneira bem mais rápida que na abordagem anterior. Isso ocorre pois com o CPI a TCoE sabe exatamente o ponto de interconexão do novo *switch* o que dispensa uma descoberta completa da topologia. Os demais tempos são similares aos obtidos com o Experimento 2.1. De maneira geral o tempo gasto em cada etapa apresentou pouco ou nenhum crescimento na comparação com topologias de diferentes proporções.

Ao analisar os resultados dos Experimentos 2.1 e 2.2 verifica-se que o tempo de 'Atraso/Processamento' é alto em ambas as abordagens. Isso se deve ao fato de que os novos *switches* precisam executar rotinas com os protocolos ARP, STP e LLDP para estarem aptos a se comunicarem e conhecerem os *switches* vizinhos. Realizar a etapa de 'Descoberta' logo após configurar os fluxos de acesso pode fazer com que as primeiras tentativas falhem, até que os fluxos definidos por estes protocolos sejam concluídos. Além disso é necessário aguardar a inicialização de *daemons* no *switch* que são importantes para o processo de 'Descoberta', tais como o *snmpd*, *lldpd* e *ovsdb-server*. Com o objetivo de diminuir este tempo de espera a presente tese propõe uma estratégia chamada de "descoberta atrasada" na qual o novo *switch* só é descoberto depois de ser configurado. Essa estratégia só funciona com o CPI, pois ela requer a detecção do ponto de interconexão do novo *switch* antes da realização do procedimento de descoberta. O gráfico *c* da Figura 46 apresenta os resultados obtidos com essa estratégia em um novo experimento (Experimento 2.3). Pode-se observar que embora o tempo da maioria das etapas nos experimentos com e sem a estratégia de "descoberta atrasada" sejam parecidos, o tempo de 'Atraso/Processamento' diminuiu visivelmente no Experimento 2.3.

Os resultados apresentados acima atendem ao planejamento realizado na seção 7.2.2: *i)* o novo elemento de rede é automaticamente descoberto através do CPI que captura as primitivas de DHCP encaminhadas ao Controlador SDN via protocolo *OpenFlow*, ou através da TCoE que realiza um procedimento de descoberta via SNMP com base no protocolo LLDP diretamente nos elementos de rede; e, *ii)* os fluxos de controle e de gerenciamento são automaticamente recalculados e implantados pela SCE através da NBI do Controlador SDN. Os resultados demonstram ganhos em tempo e facilidade de utilização se comparados com abordagens mais tradicionais (manuais ou semiautomáticas). Verifica-se que a utilização do CPI auxilia na operação de *plug-and-play* de novos *switches* por permitir a interceptação de mensagens destinadas ao Controlador, sobretudo as de

PacketIn[2]. Além disso, observa-se que ao postergar o processo de descoberta pode-se obter uma redução média de mais de 50% no tempo de 'Atraso/Processamento'.

7.5.3 Resultados do Experimento de Aprovisionamento de Serviço de VoIP

Como apresentado na seção 7.4.3 o experimento de aprovisionamento de um serviço para comunicação VoIP se baseou na comparação de resultados com dois testes: o primeiro com uma rede não baseada na SONAr, que não aplica nenhum tipo de QoS para priorização de tráfego, e com dispositivos operando com uma estratégia *learning-switch*; e o segundo com uma rede SONAr e com serviços específicos configurado pra cada comunicação conforme definido pelo IBSM. Naturalmente o responsável por priorizar o tráfico e prover os ganhos apresentados nesta seção é o próprio mecanismo de QoS, no entanto, a SeCoMP facilita a utilização de políticas de QoS através da abstração de serviço do IBSRM e do modelo de aprovisionamento de serviços especificados pelo IBSPM e materializado pela SONAr. Os resultados visam demonstrar a viabilidade e efetividade da configuração de QoS de maneira automática provida pela SeCoMP.

A SCE, entidade responsável diretamente pelo aprovisionamento dos serviços, ao receber e validar a requisição de configuração do novo serviço realiza um cálculo para escolha dos enlaces a serem utilizados. Depois a SCE escolhe ao menos uma porta para aplicar políticas de QoS, idealmente uma porta comum às todas as comunicações ativas. Nesse experimento foi aplicado QoS em todas as portas de entrada e saída utilizadas na comunicação entre as duas pontas. Para isso a SCE acessa os *switches* com OVSDB e verifica se a porta já possui QoS. Em caso negativo, a SCE configura a política de QoS na porta com a estratégia *Linux-HTB* (mecanismo de QoS nativo do *Linux*). Nessa abordagem são definidas filas (*queues*) para o processamento do tráfego, sendo a *fila 0* utilizada por padrão e as demais apenas quando definidas de forma explícita. Para utilizar uma fila específica com o protocolo *OpenFlow* deve-se configurar um fluxo com uma *action* chamada *set_queue* que permite definir a fila a ser utilizada para uma comunicação.

A fila padrão (*fila 0*) é configurada com os campos *priority 1* e *max-rate* igual à capacidade máxima aferida para o enlace menos a somatória dos campos *min-rate* das demais filas acrescidas de 5% de margem (parâmetro auto-ajustável via TSLE). Embora não seja necessário ajustar o *max-rate* da *fila 0* para garantir o *min-rate* das demais filas a SCE aplica esse cuidado adicional na garantia da banda mínima para os serviços aprovisionados. Para cada novo serviço a SCE cria uma nova fila com prioridade de acordo com as políticas definidas, sendo 10 para política de banda mínima, e 5 para política de banda máxima. Neste experimento a prioridade da fila para tratar a comunicação VoIP foi 10, pois apenas uma política foi definida: a de banda mínima. Além disso a SCE utiliza

[2] *PacketIn*: Primitiva definida em todas as versões do protocolo *OpenFlow* para encaminhamento de um pacote recebido para o Controlador.

o parâmetro *min-rate* para definir o mínimo de banda necessário para a comunicação por essa fila. Neste experimento o *min-rate* configurado foi de 67Kbps que é igual aos 64kbps definidos pelo serviço mais 5% de margem que também é auto-ajustável via TSLE.

A Figura 47 apresenta os resultados obtidos com o teste no qual os componentes SONAr não atuam na automação do gerenciamento, e por isso, nenhuma política de QoS foi configurada. Podem ser observadas duas comunicações: a primeira em azul representando o tráfego VoIP durante os primeiros 50 segundos; e a segunda em vermelho representando o tráfego de *Torrent* que inicia 20 segundos após o início da primeira comunicação. Os resultados demonstram que sem uma política de QoS a comunicação VoIP é severamente afetada, o que na prática inviabilizaria a comunicação em curso. Isso pode ser observado de maneira clara a partir de 20 segundos na Figura 47, na qual verifica-se que a linha em azul (comunicação VoIP) começa a oscilar abaixo dos 64kbps, o que é um requisito para comunicações deste tipo segundo o cenário descrito na seção 7.2.3.

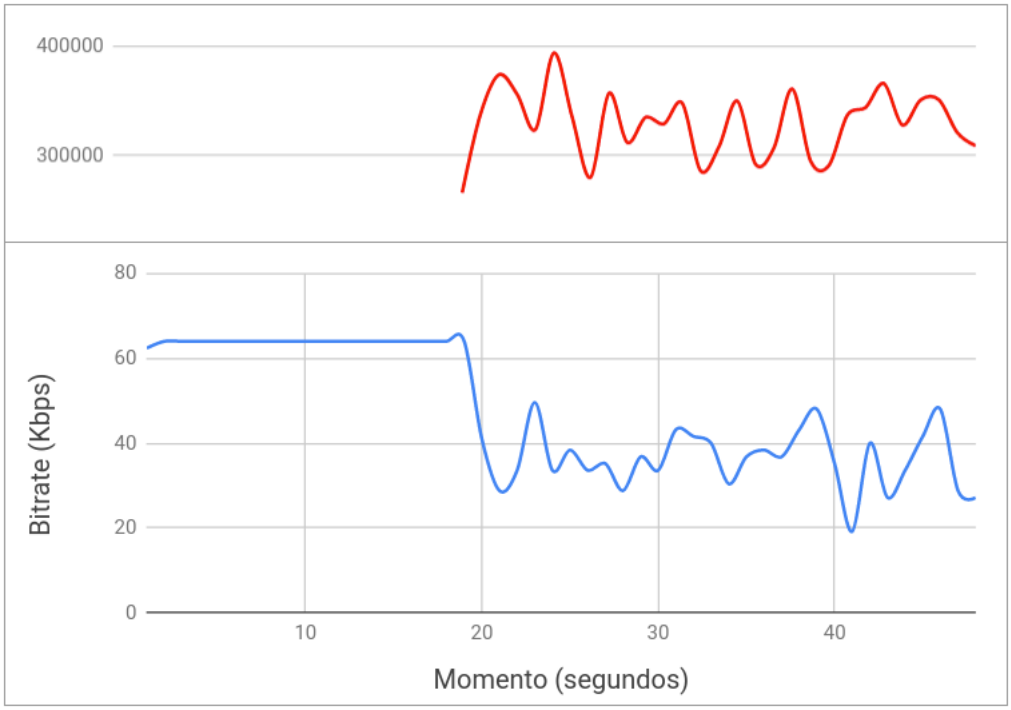


Figura 47 – Tráfego concorrente entre duas comunicações: VoIP e Torrent - Sem QoS.

Fonte: Elaborado pelo autor (2021).

A Figura 48 apresenta os resultados obtidos com o teste no qual a SeCoMP foi utilizada para aprovisionamento do serviço e que por isso foram aplicadas políticas de QoS. Neste experimento a rede é iniciada de maneira similar à executada nos experimentos anteriores. Novamente podem ser observadas duas comunicações: a primeira em azul representando o tráfego VoIP durante os primeiros 50 segundos; e a segunda em vermelho representando o tráfego de *Torrent* que é iniciado 20 segundos após o início da primeira comunicação. Os resultados demonstram que após o QoS ser implantado pela SCE a comunicação VoIP

passou a não mais ser afetada pela concorrência com outras comunicações. Isso pode ser observado pela estabilidade da linha azul (comunicação VoIP) na Figura 48 durante todo o experimento. Assim, considera-se que os resultados obtidos com este experimento atendem ao planejamento realizado na seção 7.2.3, uma vez que a comunicação com uma largura de banda mínima é garantida durante toda a comunicação VoIP.

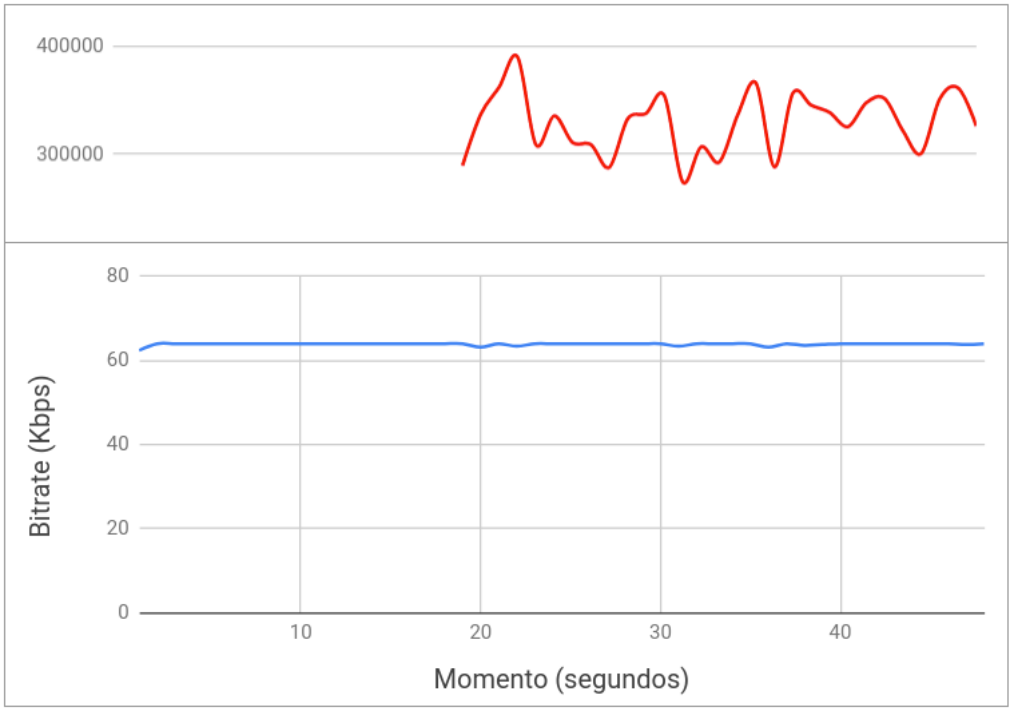


Figura 48 – Tráfego concorrente entre duas comunicações: VoIP e Torrent - Com QoS.

Fonte: Elaborado pelo autor (2021).

Foram realizadas cinco execuções em cada um dos testes, sendo as primeiras de cada teste utilizadas para a elaboração dos gráficos apresentados nas Figuras 47 e 48; e as demais execuções utilizadas para confirmar os resultados obtidos. Em todas as execuções o comportamento observado foi o mesmo: a SeCoMP se mostrou capaz de configurar o serviço de QoS de maneira automática garantindo assim uma banda mínima para o tráfego de VoIP durante toda a comunicação.

Mais que os ganhos obtidos com a utilização de QoS, estes resultados demonstram que a SONAr facilita a configuração de políticas de uso de banda através de uma abstração de alto nível de serviço. Os mesmos resultados poderiam ser obtidos ao configurar manualmente os elementos de rede com as mesmas políticas e fluxos instalados pela SCE, no entanto esta entidade provê uma solução simples que pode ser utilizada por administradores e usuários na configuração de serviços.

Capítulo 8

Conclusão

As redes de computadores são fundamentais para a sociedade e atualmente são aplicadas nas mais diversas áreas, desde entretenimento, notícias e finanças até saúde e segurança. A indisponibilidade desse serviço pode trazer inúmeros impactos negativos incluindo perdas financeiras e risco de vida, por isso as aplicações atuais definem requisitos de alto desempenho e disponibilidade que tornam as soluções de redes mais complexas. A única maneira de mitigar estes riscos e prover os requisitos definidos pelas aplicações é por meio do gerenciamento de rede, atividade está que consiste no emprego de técnicas, ferramentas e estratégias com o objetivo de manter a rede sempre em estado operacional e com um comportamento esperado.

Com o crescimento e popularização das redes verifica-se também um aumento de complexidade não só das soluções de rede no nível de aplicação, mas também do próprio gerenciamento de rede. Isto agrava-se com o fato de que nas redes de computadores atuais o gerenciamento ainda requer alto nível de conhecimento técnico dos administradores, e que por isso, oferece riscos inerentes às limitações humanas. Pode-se afirmar que há um movimento crescente que propõe a automação do gerenciamento de rede, sobretudo das redes de telecomunicações, no entanto, as redes de computadores atuais ainda estão longe de serem consideradas sistemas autônomos tais como os descritos na especificação da IBM. Para prover redes de computadores autônomas é necessário reduzir drasticamente a intervenção humana no gerenciamento através da automação de operações de coleta, análise, aprendizado, tomada de decisão e intervenção na rede.

Nesse contexto, o presente trabalho propõe uma solução para a automação do gerenciamento de rede que contempla as redes de computadores. Esta solução é apresentada nesta tese com diferentes níveis de abstração (modelo, arquitetura, *framework* e plataforma) e tem foco na automação de operações de auto-configuração. Para isso, são propostos o *Intent-Based Service Model* (IBSM) e o *Self-Organizing Network Model* (SONeM) no Capítulo 4; a *Self-Organizing Network Architecture* (SONAr) e um estudo de caso com operações da propriedade de auto-configuração no Capítulo 5; o *Self-Organizing Network*

Framework (SONAr-Framework) e a *Self-Configuration and Management Platform* (SeCoMP) no Capítulo 6.

Cada capítulo apresenta uma visão da solução proposta nesta tese, começando da abstração mais alta (modelos conceituais) até a mais baixa (*framework* de gerenciamento e plataforma de auto-configuração). Isso se deve ao processo de pesquisa e desenvolvimento adotado que define artefatos de solução de acordo com ciclos de refinamentos e com aprofundamento da abstração gradual a partir da realização de modelagem conceitual. Primeiramente são definidos dois modelos conceituais: *i) Intent-Based Service Model* (IBSM), que é um modelo para definição, representação e aprovisionamento de serviços baseados em intenções; e, *ii) Self-Organizing Network Model* (SONeM), que define um fluxo de operação básico para componentes de gerenciamento e uma forma de organização da rede em camadas. O SONeM propõe a utilização de uma camada de gerenciamento que é dividida nas subcamadas de coleta, análise e intervenção. Estes modelos se complementam na concepção de uma solução de gerenciamento, pois consideram aspectos distintos e necessários para a flexibilização e automação da rede.

De maneira a definir os princípios arquiteturais de uma rede auto-organizada, e com base nos modelos conceituais definidos na primeira etapa da solução, esta tese propõe a *Self-Organizing Network Architecture* (SONAr), uma arquitetura que define entidades de gerenciamento especificadas de acordo com os princípios de computação autônoma[1]. Essas entidades são organizadas de acordo com o modelo de referência SONeM e se dividem em: *Self-Organizing Entities* (SOEs) – responsáveis pela execução dos fluxos de cura, configuração, otimização e proteção, o que inclui a tomada de decisão e intervenção na rede; *Self-Learning Entities* (SLEs), entidades responsáveis pela análise das informações coletadas na rede sem intervenção humana através de técnicas de inteligência artificial; *Collecting Entities* (CoEs) – entidades coletoras de dados de rede; e, os componentes auxiliares que visam suportar a operação das demais entidades. A SONAr também especifica componentes para materialização dos conceitos definidos pelo IBSM, tais como o *Network Service Broker* (NSB) – responsável por prover uma interface para gerenciamento de serviços de comunicação; *Behavior Self-Learning Entity* (BSLE) – responsável pela inferência de serviços por comportamentos; e, a *Natural Language Processing Self-Learning Entity* (NSLE) – responsável por traduzir serviços descritos em linguagem natural.

Com o objetivo de prover uma implementação flexível da SONAr que permita a sua aplicação em diferentes processos de gerenciamento propõe-se o *Self-Organizing Network Framework* (SONAr-Framework), que materializa a SONAr através da abstração dos componentes de gerenciamento como microsserviços conteinerizados com *Docker*. O SONAr-Framework integra os componentes de gerenciamento por meio de um banco de dados distribuído e de um *broker* de eventos com suporte ao protocolo AMQP. Este *framework* é utilizado na implementação da SeCoMP que automatiza operações de configuração em

[1] auto-configuração, auto-cura, auto-otimização, auto-proteção, auto-aprendizado e auto-consciência

redes SDN com suporte aos protocolos *OpenFlow* e OVSDB. A SeCoMP é uma plataforma criada a partir do estudo de caso realizado neste trabalho com três operações básicas de configuração: inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços. Acredita-se que tanto a plataforma quanto o *framework* implementados por este trabalho possam ser aprimorados facilmente para suportar novas operações, propriedades e tecnologias.

Existe uma relação de dependência em cascata e composição entre os artefatos criados nesta tese. Pode-se dizer que o SeCoMP é uma plataforma de auto-configuração construída sobre o SONAr-Framework a partir do 'estudo de caso de auto-configuração através da SONAr'. Já o SONAr-Framework materializa os principais conceitos da SONAr de maneira a facilitar a implementação de uma solução de auto-gerenciamento. A SONAr por sua vez abstrai o IBSM e o SONeM em uma única solução arquitetural de rede auto-organizada flexível às necessidades dos usuários. Por fim, o IBSM é composto por: *i)* submodelo de representação de serviços (IBSRM); *ii)* submodelo de aprovisionamento de serviços (IBSPM); *iii)* definição ontológica de serviços orientados por intenções (IBSO); e, *iv)* 'levantamento de requisitos comunicacionais'. Conclui-se assim que os experimentos realizados com o SeCoMP ao comprovarem a viabilidade da solução e os ganhos qualitativos e quantitativos descritos no Capítulo 7, validam também o SONAr-Framework e a SONAr através do método de 'protótipo de prova de conceito' (*proof-of-concept prototype*) como uma solução arquitetural capaz de guiar a construção de uma rede auto-organizada. A SONAr por sua vez ao ser validada como uma solução arquitetural para abstração de redes auto-organizadas comprova também conceitos dos modelos IBSM e SONeM que são materializadas por essa arquitetura.

8.1 Análise da Hipótese e dos Problemas de Pesquisa

Considerando a hipótese descrita na seção 1.2:

> *"É possível melhorar a velocidade e flexibilidade das operações de gerenciamento através da automação da configuração dos elementos de infraestrutura."*

Considera-se que os resultados apresentados no Capítulo 7 sejam capazes de comprovar esta hipótese pois submetem a solução materializada pela SeCoMP que automatiza a configuração de dispositivos através componentes de gerenciamento especializados na inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços, à experimentos emulados inspirados em redes reais de diferentes proporções: desde as redes domésticas mais básicas até redes mais complexas de grandes ISPs. Os resultados demonstram os ganhos qualitativos referentes à flexibilização e simplificação do gerenciamento

de rede e modelagem/aprovisionamento de serviços de comunicação; e, quantitativos referentes à diminuição do tempo expendido nas operações de configuração em relação às abordagens convencionais [2]. Dessa forma, pode-se concluir que de fato ao automatizar a configuração de elementos de infraestrutura os serviços de rede e as operações de gerenciamento relacionadas à inicialização da rede e o *plug-and-play* de dispositivos são aceleradas e se tornam mais flexíveis aos requisitos das aplicações e operadores, o que se deve ao fato de se tornar possível implantar estratégias de gerenciamento por meio de Inteligência Artificial como a solução proposta neste trabalho.

Considera-se também que os problemas de pesquisa levantados na seção 1.2.1 foram endereçados o que corrobora com a afirmação realizada acima. A seguir uma breve explanação sobre como o presente trabalho responde a estes problemas:

Problema de Pesquisa 1: *Como automatizar a inicialização da infraestrutura de rede e dos componentes de controle e gerenciamento?*

Para a automação da inicialização da infraestrutura de rede e dos componentes de controle e gerenciamento a SONAr propõe o *Auto-Boot Manager* (ABM) que é responsável por aprovisionar, configurar e iniciar os componentes de controle e de gerenciamento. O ABM é responsável também por distribuir os componentes de acordo com a necessidade e a disponibilidade de servidores na infraestrutura; e por criar os *clusteres* de banco de dados (*Network Database*) e de barramentos de eventos/mensagens (*Network Event Manager*). A *Self-Configuration Entity* (SCE) também desempenha um papel fundamental para a inicialização da rede: *i)* configurar as rotas/fluxos para estabelecimento de comunicação entre os recursos de infraestrutura e os componentes de controle/gerenciamento; *ii)* configuração de recursos (e.g. *switches*, servidores e funções de rede); e, *iii)* configuração de serviços de comunicação. A tomada de decisão sobre o 'plano de configuração' utilizado pela SCE se ampara nas informações coletadas pelas *Collecting Entities* (CoEs) e analisadas pelas *Self-Learning Entities* (SLEs) que são entidades de gerenciamento propostas pela SONAr.

O SONAr-Framework (no qual a SeCoMP se baseia) define componentes específicos para a automação do gerenciamento úteis para a inicialização da rede, são eles: *Address Provider* (AP) – responsável por prover endereços para os recursos de infraestrutura; *Container Provider* (CP) – responsável por prover a abstração de contêineres; *Resource Data Provider* (RDP) – responsável por disponibilizar dados locais por meio de uma interface/-protocolo bem definida; *Neighbor Discovery Provider* (NDP) – responsável por descobrir informações sobre os vizinhos dos recursos de infraestrutura de maneira a auxiliar na descoberta da topologia; e, *Containerized Network Infrastructure Manager* (C-NIM) – responsável por interfacear a criação de contêineres para o aprovisionamento de componentes de controle/gerenciamento e funções de rede.

[2] O gerenciamento em redes de computadores TCP/IP normalmente é manual ou semi-automatizado

O fluxo de inicialização considerando os componentes SONAr foi apresentado de maneira simplificada na seção 5.1.7.3, e de maneira mais detalhada no estudo de caso com operações de configuração apresentado na seção 5.2.1. A estratégia de inicialização da infraestrutura de rede e dos componentes de controle e gerenciamento é implementada pela SeCoMP descrita na seção 6.2; é exemplificada de maneira prática no cenário descrito na seção 7.2.1 e no experimento descrito na seção 7.4.1; e tem os seus resultados analisados na seção 7.5.1.

Problema de Pesquisa 2: ***Como permitir o acoplamento e desacoplamento automático de dispositivos na infraestrutura de rede?***

O processo de automação do acoplamento e desacoplamento de dispositivos na infraestrutura de rede de maneira segura proposto por esta tese requer pelo menos três atividades: *i)* detectar quando, onde e qual dispositivo foi conectado ou desconectado; *ii)* autorizar a conexão do dispositivo quando conectado; e, *iii)* realizar configurações no novo dispositivo e nos demais dispositivos da topologia de maneira a possibilitar a comunicação no plano de controle/gerenciamento, aplicar políticas e configurações padrões, e melhorar o QoS dos serviços de comunicação ativos.

Para resolver *i* (detecção do dispositivo acoplado ou desacoplado) a SONAr define a utilização do *Control Plane Interceptor* (CPI) – componente responsável por interfacear a comunicação entre recursos da infraestrutura e componentes de controle e pela notificação dos componentes de gerenciamento quando detectadas situações específicas, como por exemplo quando o controlador recebe uma requisição DHCP via *PacketIn* do protocolo *OpenFlow*; e, *Topology Collector Entity* – entidade responsável por coletar informações de topologia de maneira ativa e por notificar os demais componentes de gerenciamento em caso de mudanças na infraestrutura. O SONAr-Framework também define a utilização do *Address Provider* (AP) – responsável por prover endereços para os recursos de infraestrutura; *Resource Data Provider* (RDP) – responsável por disponibilizar dados locais (nos recursos de infraestrutura) por meio de uma interface/protocolo bem definida; e, *Neighbor Discovery Provider* (NDP) – responsável por descobrir informações sobre os vizinhos dos recursos de infraestrutura de maneira a auxiliar na descoberta da topologia.

Para resolver *ii* (autorização do dispositivo conectado) a SONAr define a um fluxo que envolve a realização de autorização por meio de troca de certificados intermediada pela própria *Topology Collector Entity* (TCoE). E por fim, para resolver *iii* (configuração de serviços e dos fluxos de controle/gerenciamento) a SONAr propõe a utilização da *Self-Configuration Entity* (SCE) que é responsável por prover o aspecto de auto-configuração para a rede. Assim como no processo de inicialização, a SCE se ampara nas informações coletadas pelas *Collecting Entities* (CoEs) e analisadas pelas *Self-Learning Entities* (SLEs) para tomada de decisão sobre quais os enlaces e funções de rede devem ser utilizados no

estabelecimento do plano de controle/gerenciamento e no aprovisionamento de serviços de comunicação.

O fluxo de *plug-and-play* com componentes SONAr apresentado de maneira simplificada na seção 5.1.7.1, e de maneira mais detalhada no estudo de caso com operações de configuração apresentado na seção 5.2.2, é implementado pela SeCoMP que automatiza as principais operações de configuração em redes definidas por *software*. A estratégia adotada no *plug-and-play* de *switches* na topologia é exemplificada de maneira prática no cenário descrito na seção 7.2.2, no experimento descrito na seção 7.4.2 e nos resultados apresentados na seção 7.5.2.

Problema de Pesquisa 3: *Como traduzir os requisitos de aplicações e corporações em instruções de configuração da infraestrutura de rede?*

Para resolver o problema de tradução dos requisitos de aplicações em instruções de configuração a presente tese iniciou pela modelagem conceitual de 'serviço' através do *Intent-Based Service Model* (IBSM), um modelo para definição, representação e aprovisionamento de serviços de comunicação baseados em intenções. Os *Intent-Based Services* (IBSs) propostos pelo IBSM flexibilizam a utilização da rede pois possibilitam a interação entre usuários/aplicações e componentes de gerenciamento em alto nível, e assim, sem a necessidade de conhecimento técnico sobre protocolos e nem informações sobre a infraestrutura.

O IBSM propõe um conjunto de elementos conceituais responsáveis pela interpretação, aprovisionamento e validação de serviços que foram abstraídos na *Self-Organizing Network Architecture* (SONAr) por meio da *Self-Configuration Entity* (SCE) – responsável por traduzir a definição formal de serviço em instruções de configurações; *Collecting Entities* (CoEs) – utilizadas para telemetria e análise preliminar dos dados; *Self-Learning Entities* (SLE) – responsáveis por validar a configuração do serviço; e, *Network Service Broker* (NSB) – um *broker* na NBI da camada de gerenciamento para a configuração, remoção, consulta, bloqueio e desbloqueio de serviços de comunicação. Caso alguma SLE realize o diagnóstico de que o SLA foi quebrado ela dispara um evento que é interpretado pela *Self-Healing Entity* (SHE) que avalia as causas do problema e que solicita a reconfiguração do serviço, se necessário, por RPC (via NEM) para a SCE.

A SONAr define também duas entidades de análise/aprendizado para a interpretação de serviços definidos por de maneira informal: *i)* serviços descritos em linguagem natural – *Natural Language Processing Self-Learning Entity* (NSLE); e, *ii)* serviços inferidos por comportamentos – *Behavior Self-Learning Entity* (BSLE). Essas entidades flexibilizam ainda mais a solução proposta pela SONAr para a definição e configuração de serviços de comunicação, e representam de maneira direta os componentes conceituais *Intent-Based Service Natural Language Translator* (IBSNLT) e *Intent-Based Service Behavior Detector* (IBSBD) definidos pelo IBSM

O IBSM inclui uma lista básica de requisitos comunicacionais funcionais e não-funcionais obtidos por meio do processo de engenharia de requisitos. Esta lista permite a concepção dos modelos de representação e aprovisionamento propostos pelo IBSM (*Intent-Based Service Representation Model* (IBSRM) e *Intent-Based Service Provisioning Model* (IBSPM)); e, guia a construção e evolução da *Intent-Based Services Ontology* (IBSO), uma ontologia para a formulação de serviço de comunicação em rede baseados em intenções descritas ou inferidas. Considerando a maneira como os IBSs são suportados pela SeCoMP conclui-se que a extensão da lista de requisitos inclusa na IBSO e no IBSM é uma tarefa simples, o que provê flexibilidade para a solução de aprovisionamento de serviços proposta por esta tese inclusive para os requisitos a serem definidos pelas aplicações futuras.

O fluxo de aprovisionamento de serviços foi proposto de maneira simplificada na seção 5.1.7.2, e de maneira mais detalhada no estudo de caso com operações de configuração apresentado na seção 5.2.3. A estratégia adotada no aprovisionamento de serviços é exemplificada de maneira prática no cenário descrito na seção 7.2.3, no experimento descrito na seção 7.4.3 e nos resultados apresentados na seção 7.5.3.

8.2 Análise dos Objetivos do Trabalho

A presente tese define como objetivo geral a "especificação de uma plataforma de gerência para redes auto-organizáveis com foco nas operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços." na seção 1.3. Considera-se que com a implementação da SeCoMP descrito no Capítulo 6 e validado por meio dos experimentos descritos no Capítulo 7. Para a implementação da plataforma alvo da pesquisa desenvolvida e apresentada nesta tese foram definidos artefatos de solução de maneira a alcançar os objetivos específicos definidos na seção 1.3.1. A forma como cada objetivo foi alcançado é descrita abaixo:

1. **Especificar modelos conceituais de redes autônomas com serviços flexíveis aos requisitos comunicacionais atuais e futuros**: de maneira a atender a este objetivo específico o presente trabalho propõe o *Intent-Based Service Model* (IBSM), um modelo conceitual para definição, representação e aprovisionamento de serviços baseados em intenções descritas ou inferidas; e, o *Self-Organizing Network Model* (SONeM), um modelo de organização da rede que define uma camada de gerenciamento com componentes responsáveis pela coleta, análise e intervenção na rede;

2. **Descrever uma proposta de arquitetura de redes auto-organizadas e investigá-la por meio de um estudo de caso com operações de configuração**: para resolver este objetivo específico o presente trabalho propõe a *Self-Organizing Network*

Architecture (SONAr), uma arquitetura de rede auto-organizada com entidades de gerenciamento inspirada em conceitos de computação autônoma; e, realiza um estudo de caso com a SONAr para a automação das operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços;

3. **Implementar uma solução prática e flexível para a automação do aspecto de configuração no gerenciamento de rede**: este objetivo específico foi endereçado no presente trabalho pelo *Self-Organizing Network Framework* (SONAr-Framework), um *framework* de gerenciamento de rede flexível com componentes abstraídos como microsserviços conteinerizados, que permitiu a implementação da *Self-Configuration and Management Platform* (SeCoMP), uma plataforma de auto-configuração de redes definidas por *softwares*.

4. **Levantar requisitos comunicacionais funcionais e não-funcionais das aplicações, usuários e provedores de serviço**: o presente trabalho resolve este objetivo específico por meio da lista de requisitos comunicacionais apresentados na seção 4.1.1.2 que foram levantados por meio do processo de engenharia de requisitos descrito na seção 3.4.4;

5. **Formalizar de maneira ontológica uma definição de serviço comunicacional flexível**: para isso esta tese modela a *Intent-Based Services Ontology* (IBSO), uma ontologia para definição de serviços comunicacionais baseados em intenções; e,

6. **Investigar a efetividade a solução proposta por meio de experimentação com cenários que envolvam a configuração de componentes, dispositivos e serviços**: este objetivo faz parte do processo de validação do presente trabalho que se deu por meio do processo de experimentação documentado no Capítulo 7.

Como previsto na seção 1.3, ao alcançar os objetivos específicos descritos acima diversos aspectos necessários para a concepção de solução de auto-configuração são resolvidos. Conclui-se assim que o objetivo do trabalho foi alcançado.

8.3 Posicionamento do Trabalho no Estado da Arte

Ao considerar o 'Estado da Arte' estabelecido na seção 2.4, verifica-se que a grande diferença entre os trabalhos correlatos e o trabalho apresentado nesta tese está no foco em redes de computadores incluindo as redes baseadas em tecnologias legadas. Além disso, a presente tese propõe uma solução completa com múltiplos níveis de abstração (desde modelos conceituais até uma plataforma de auto-configuração), e apresenta resultados emulados com implementações e cenários práticos. Outra diferença está no escopo amplo do trabalho apresentado nesta tese que propõe uma solução abrangente de gerenciamento

e que implementa uma solução de configuração prática tanto para o *bootstrapping* da rede quanto o *plug-and-play* de dispositivos e o aprovisionamento de serviços.

Tsagkaris et al. (2015) propõe o ANM que se difere da proposta deste trabalho por ser um trabalho conceitual que não considera aspectos como aprovisionamento de serviços e de componentes de gerenciamento. Já o SELFNET (NEVES et al., 2016) é um trabalho com foco em redes 5G, enquanto o presente trabalho pretende ser aplicado também em redes de computadores convencionais. Ademais, ambos os trabalhos são contemporâneos e similares, e portanto, acredita-se em uma futura cooperação ou integração entre ambas as propostas. O SUPERFLUIDITY (BIANCHI et al., 2016) propõe uma alternativa ao NFV com funções de gerenciamento autônomo, mas que não considera aspectos de auto-configuração como os tratados nesta tese. O COGNET (XU et al., 2016) propõe o uso de componentes com capacidades cognitivas acoplados aos componentes do MANO (arquitetura NFV) que se assemelham com os propostos nesta tese. Entretanto, a solução proposta no presente trabalho se difere por poder ser aplicada a redes de computadores convencionais e por não se limitar a utilização de NFV. Abdallah et al. (2018) propõe o uso de um componente Gerenciador, enquanto no trabalho apresentado nesta tese são propostas entidades e componentes de gerenciamento organizadas em uma camada de gerenciamento específica. Ferreira et al. (2020) propõe o ArchSDN que também se baseia na ideia de um componente Gerenciador, mas com foco na distribuição do plano de controle de maneira vertical, enquanto o presente trabalho trata essa distribuição apenas de maneira horizontal.

Sharma et al. (2013) propõe um método de configuração de *switches* OpenFlow, enquanto a pesquisa apresentada nesta tese compreende de maneira completa as operações de *boostrapping* da rede e *plug-and-play* de dispositivos, e ainda aborda a operação de aprovisionamento de serviços. Patil, Gokhale e Hakiri (2015) propõe o InitSDN que é uma solução de inicialização de redes SDN que prevê a sincronização do plano de controle (assim como o presente trabalho), mas que não considera o aspecto de aprovisionamento dos componentes de controle e de gerenciamento de maneira automática. Katiyar et al. (2015) propõe um fluxo de inicialização de rede que se assemelha ao proposto nesta tese e que também suporta redes legadas. No entanto, o autor não apresenta resultados práticos e o trabalho não considera o aprovisionamento de serviços. O LearnQoS proposto por Al-Jawad (AL-JAWAD et al., 2018; AL-JAWAD, 2018) especifica componentes de gerenciamento posicionados acima da camada de Controle (assim como a solução apresentada nesta tese), mas a proposta de Al-Jawad (2018) tem foco apenas na garantia dos serviços de comunicação, enquanto o presente trabalho foca também na automação do *bootstrapping* da rede e do *plug-and-play* de dispositivos. Por fim, Yousafzai e Hong (2020) propõe o SmartSON que é um *framework* voltado para a negociação de serviços amparados pela rede, mas que desconsidera aspectos relacionados à gestão da infraestrutura e aprovisionamento dos serviços.

8.4 Contribuições do Trabalho

A solução proposta neste trabalho se divide em artefatos que representam contribuições efetivas para diversas áreas do gerenciamento de redes. As principais contribuições são listadas abaixo:

1. ❑ Artefato: ***Intent-Based Service Model*** **(IBSM)**
   ❑ Referência: Capítulo 4 – Seção 4.1
   ❑ Contribuição: Modelo conceitual para representação e aprovisionamento de serviços de comunicação flexíveis e com definições de alto nível

2. ❑ Artefato: ***Self-Organizing Network Architecture*** **(SONAr)**
   ❑ Referência: Capítulo 5 – Seção 5.1
   ❑ Contribuição: Arquitetura para concepção de redes auto-organizadas através de componentes responsáveis pela abstração das propriedades[3] de sistemas autônomos

3. ❑ Artefato: ***Self-Configuration and Management Platform*** **(SeCoMP)**
   ❑ Referência: Capítulo 6 – Seção 6.2
   ❑ Contribuição: Plataforma de auto-configuração de redes definidas por *software* que automatiza as operações de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços

Outras contribuíções menores compreendem os seguintes artefatos:

1. ❑ Artefato: ***Self-Organizing Network Model*** **(SONeM)**
   ❑ Referência: Capítulo 4 – Seção 4.2
   ❑ Contribuição: Modelo conceitual de organização da rede com componentes de gerenciamento com capacidade de execução autônoma

2. ❑ Artefato: ***Intent-Based Services Ontology*** **(IBSO)**
   ❑ Referência: Capítulo 4 – Seção 4.1.1.3
   ❑ Contribuição: Ontologia para definição de serviços de comunicação flexíveis e com definições em alto-nível

3. ❑ Artefato: ***Levantamento de Requisitos Comunicacionais***
   ❑ Referência: Capítulo 4 – Seção 4.1.1.2
   ❑ Contribuição: Documento de referência com os requisitos comunicacionais das aplicações, usuários e provedores de serviços atuais

4. ❑ Artefato: ***Estudo de Caso: Auto-Configuração através da SONAr***

[3] auto-configuração, auto-cura, auto-otimização, auto-proteção, auto-orquestração, auto-consciência e auto-aprendizado

- ❏ Referência: Capítulo 5 – Seção 5.2
- ❏ Contribuição: Documento com estudo de caso com operações de configuração automatizadas por meio da solução proposta e especificação dos processos de inicialização da rede, *plug-and-play* de dispositivos e aprovisionamento de serviços

5. ❏ Artefato: ***Self-Organizing Network Framework*** **(SONAr-Framework)**
   - ❏ Referência: Capítulo 6 – Seção 6.1
   - ❏ Contribuição: *Framework* de gerenciamento de rede baseada em componentes abstraídos como microsserviços conteinerizados

8.5 Perspectivas Futuras

Esta seção apresenta as principais perspectivas para continuação do trabalho apresentado nesta tese:

- ❏ Extensão da solução de automação de operações de configuração implementada pela SeCoMP para novos tipos de redes com diferentes tecnologias (e.g. rede metro-ethernet e 5G);
- ❏ Implementação de novas operações de gerenciamento no SONAr-Framework, não apenas voltadas ao aspecto de configuração, mas também de cura, proteção e otimização;
- ❏ Aplicação da SeCoMP em redes reais (e.g. rede doméstica e rede de um ISP), e testes comparativos com cenários práticos;
- ❏ Criação de uma *test-bed* com a SeCoMP integrada à outras soluções baseadas no SONAr-Framework compartilhada entre os desenvolvedores do MEHAR;
- ❏ Implementação das SLEs com técnicas de Inteligência Artificial, e das CoEs não contempladas neste trabalho no SONAr-Framework;
- ❏ Amadurecimento e refinamento da SONAr através da proposição e eliminação de entidades; melhoria da documentação; e, aplicação em mais estudos de caso;
- ❏ Refinamento do IBSM e da IBSO através da revisão do levantamento de requisitos comunicacionais e aprimoramento da especificação de serviço baseado em intenção.

Deve-se destacar que a evolução do presente trabalho encontra-se em curso por meio de projetos que contemplam a propriedade de auto-cura e a definição de serviços pré-configurados especializados na configuração de *eNodeB's*. Espera-se que a próxima propriedade a ser considerada seja a de auto-otimização, e para isso far-se-á necessário

implementar as *Self-Learning Entities* (especificadas pela SONAr) dentro do SONAr-Framework, assim como as *Collecting Entities* não contempladas pela versão atual do SONAr-Framework, o que requer a utilização de técnicas de inteligência artificial e protocolos de coleta de dados de forma ativa e passiva.

A aplicação da SeCoMP em cenários reais é um passo necessário para o amadurecimento da solução, e para isso será necessário também a revisar e avaliar os componentes implementados segundo os requisitos *carrier-grade*. Considera-se a aplicação da SeCoMP sobretudo no ambiente de ISPs, tanto nas redes de transmissão de dados quanto nas de telecomunicações.

Por fim, acredita-se que o IBSM possa ser utilizado não apenas pela SONAr, mas também por outras soluções arquiteturais que visem prover serviços de comunicação mais flexíveis. A definição ontológica provida pelo IBSO, as formas de representação especificadas pela IBSRM e a solução conceitual de aprovisionamento de serviços proposta pelo IBSPM também deverão ser aprimoradas em trabalhos futuros.

8.6 Contribuições em Produção Bibliográfica

Os artigos a seguir tem relação direta com o trabalho apresentado nesta tese foram submetidos e/ou publicados pelo autor desta tese como autor principal:

1. ***Bootstrapping and Plug-and-Play Operations on Software Defined Networks: A Case Study on Self-Configuration Using the SONAr Architecture*** – publicado no CLOSER 2020; e,
2. ***Automating Configuration Operations on SDN Networks through the SONAr Management Platform*** – submetido ao TNSM 2021.

Já os artigos listados abaixo foram submetidos e/ou publicados com a coautoria do autor desta tese e aplicam a SONAr em outros contextos:

1. *A Self-healing Platform for the Control and Management Planes Communication in Softwarized and Virtualized Networks* – publicado no CLOSER 2020;
2. *Self-healing in the Scope of Software-basedComputer and Mobile Networks* – será publicado no CCIS 2021;
3. *Network Self-configuration for Edge Elements using Self-Organizing Networks Architecture (SONAr)* – será publicado no CLOSER 2021;
4. *Management Slices: Enabling Self-management for SDN and NFV Communication Layers* – submetido ao COMMAG 2021;

Referências

3GPP. *Telecommunication Management; Principles and High Level Requirements*. [S.l.], 2018. Disponível em: <http://www.3gpp.org/DynaReport/32101.htm>.

3GPP. *Telecommunication management; Self-configuration of network elements; Concepts and requirements*. [S.l.], 2018. Disponível em: <http://www.3gpp.org/-DynaReport/32501.htm>.

3GPP. *Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements*. [S.l.], 2018. Disponível em: <http://www.3gpp.org/DynaReport/-32500.htm>.

3GPP. *Telecommunication management; Self-Organizing Networks (SON) Policy Network Resource Model (NRM) Integration Reference Point (IRP); Requirements*. [S.l.], 2018. Disponível em: <http://www.3gpp.org/DynaReport/32521.htm>.

3GPP. *Telecommunication management; Self-Organizing Networks (SON); Self-healing concepts and requirements*. [S.l.], 2018. Disponível em: <http://www.3gpp.org/-DynaReport/32541.htm>.

5GPPP. *View on 5G Architecture. White paper v.3.0*. 2019. Disponível em: <https://5g-ppp.eu/wp-content/uploads/2019/07/5G-PPP-5G-Architecture-White-Paper_v3.0_PublicConsultation.pdf>.

ABDALLAH, S. et al. A network management framework for SDN. In: **2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)**. [s.n.], 2018. p. 1–4. Disponível em: <https://doi.org/10.1109/NTMS.2018.8328672>.

ABDELSALAM, M. A. **Network Application Design Challenges and Solutions in SDN**. Tese (Doutorado) — Carleton University, 2018. Disponível em: <https://doi.org/10.22215/etd/2018-13338>.

AL-JAWAD, A. Policy-based QoS Management Framework for Software-Defined Networks. In: **2018 International Symposium on Networks, Computers and Communications (ISNCC)**. [s.n.], 2018. p. 1–6. ISSN: null. Disponível em: <https://doi.org/10.1109/ISNCC.2018.8530994>.

AL-JAWAD, A. et al. Learnqos: A learning approach for optimizing qos over multimedia-based sdns. In: **2018 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, BMSB 2018, Valencia, Spain, June 6-8, 2018**. IEEE, 2018. p. 1–6. Disponível em: <https://doi.org/10.1109/BMSB.2018.8436781>.

ALAPATI, S. R. Apache cassandra: An introduction. In: **Expert Apache Cassandra Administration**. Springer, 2018. p. 3–34. Disponível em: <https://doi.org/10.1007/978-1-4842-1830-3_6>.

ANTONOPOULOS, N.; GILLAM, L. **Cloud computing**. Springer, 2010. Disponível em: <https://doi.org/10.1007/978-1-84996-241-4>.

ASTELY, D. et al. LTE: The evolution of mobile broadband. **IEEE Communications Magazine**, v. 47, n. 4, p. 44–51, abr. 2009. ISSN 0163-6804. Disponível em: <https://doi.org/10.1109/MCOM.2009.4907406>.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787–2805, out. 2010. ISSN 1389-1286. Disponível em: <https://doi.org/10.1016/j.comnet.2010.05.010>.

AZAB, M.; FORTES, J. A. B. Towards proactive SDN-controller attack and failure resilience. In: **2017 International Conference on Computing, Networking and Communications (ICNC)**. [s.n.], 2017. p. 442–448. Disponível em: <https://doi.org/10.1109/ICCNC.2017.7876169>.

BAEHNI, S.; EUGSTER, P. T.; GUERRAOUI, R. Data-aware multicast. In: IEEE. **International Conference on Dependable Systems and Networks, 2004**. 2004. p. 233–242. Disponível em: <https://doi.org/10.1109/DSN.2004.1311893>.

BAINBRIDGE, L. Ironies of automation. **Automatica**, v. 19, n. 6, p. 775–779, nov. 1983. ISSN 0005-1098. Disponível em: <https://doi.org/10.1016/0005-1098(83)90046-8>.

BARAN, P. On Distributed Communications Networks. **IEEE Transactions on Communications Systems**, v. 12, n. 1, p. 1–9, mar. 1964. ISSN 0096-1965. Disponível em: <https://doi.org/10.1109/TCOM.1964.1088883>.

BARRETT, R. et al. Field Studies of Computer System Administrators: Analysis of System Management Tools and Practices. In: **Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work**. New York, NY, USA: ACM, 2004. (CSCW '04), p. 388–395. ISBN 978-1-58113-810-8. Disponível em: <https://doi.org/10.1145/1031607.1031672>.

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software architecture in practice**. [S.l.]: Addison-Wesley Professional, 2003.

BASTUG, E. et al. Toward Interconnected Virtual Reality: Opportunities, Challenges, and Enablers. **IEEE Communications Magazine**, v. 55, n. 6, p. 110–117, 2017. ISSN 0163-6804. Disponível em: <https://doi.org/10.1109/MCOM.2017.1601089>.

BATTLE, R.; BENSON, E. Bridging the semantic web and web 2.0 with representational state transfer (rest). **Journal of Web Semantics**, Elsevier, v. 6, n. 1, p. 61–69, 2008. Disponível em: <https://doi.org/10.1016/j.websem.2007.11.002>.

BECK, K. **Test-driven development: by example**. [S.l.]: Addison-Wesley Professional, 2003.

BEHRINGER, M. et al. Autonomic networking: Definitions and design goals. **Internet, RFC7575**, 2015. Disponível em: <https://doi.org/10.17487/RFC7575>.

BEN-KIKI, O.; EVANS, C.; INGERSON, B. Yaml ain't markup language (yaml™) version 1.1. **Working Draft 2008-05**, v. 11, 2009.

BENSON, T.; AKELLA, A.; MALTZ, D. A. Unraveling the Complexity of Network Management. In: **NSDI**. [S.l.: s.n.], 2009. p. 335–348.

BERDE, P. et al. Onos: towards an open, distributed sdn os. In: **Proceedings of the third workshop on Hot topics in software defined networking**. [s.n.], 2014. p. 1–6. Disponível em: <https://doi.org/10.1145/2620728.2620744>.

BERGSTRA, J.; BURGESS, M. **Handbook of Network and System Administration**. [S.l.]: Elsevier, 2011. Google-Books-ID: NUoZ7fKOITQC. ISBN 978-0-08-055358-0.

BERNS, A.; GHOSH, S. Dissecting Self-* Properties. In: **2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems**. [s.n.], 2009. p. 10–19. ISSN: 1949-3681. Disponível em: <https://doi.org/10.1109/SASO.2009.25>.

BIANCHI, G. et al. Superfluidity: a flexible functional architecture for 5g networks. **Transactions on Emerging Telecommunications Technologies**, Wiley Online Library, v. 27, n. 9, p. 1178–1186, 2016. Disponível em: <https://doi.org/10.1002/ett.3082>.

BIERMAN, A. et al. Restconf protocol. In: **IETF RFC 8040**. [s.n.], 2017. Disponível em: <https://doi.org/10.17487/RFC8040>.

BJORKLUND, M. **YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)**. IETF, 2010. (Request for Comments, 6020). Published: RFC 6020 (Proposed Standard). Disponível em: <https://doi.org/10.17487/rfc6020>.

BLACK, U. D. **Network Management Standards: SNMP, CMIP, TMN, MIBs and Object Libraries**. [S.l.]: McGraw-Hill, Inc., 1994.

BLANCO, B. et al. Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN. **Computer Standards & Interfaces**, v. 54, p. 216–228, 2017. Publisher: Elsevier. Disponível em: <https://doi.org/10.1016/j.csi.2016.12.007>.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: guia do usuário**. [S.l.]: Elsevier Brasil, 2006.

BOSCHI, S.; SANTOMAGGIO, G. **RabbitMQ cookbook**. [S.l.]: Packt Publishing Ltd, 2013.

BOTELHO, F. A. et al. **SMaRtLight: A Practical Fault-Tolerant SDN Controller**. 2014. /paper/SMaRtLight%3A-A-Practical-Fault-Tolerant-SDN-Botelho-Bessani/65f7b3697192a79eeac3ac8c4432717bace066b2. Disponível em: <https://doi.org/10.1109/EWSDN.2014.25>.

BOURAS, C.; KOLLIA, A.; PAPAZOIS, A. Sdn & nfv in 5g: Advancements and challenges. In: IEEE. **2017 20th Conference on innovations in clouds, internet and networks (ICIN)**. 2017. p. 107–111. Disponível em: <https://doi.org/10.1109/ICIN.2017.7899398>.

BOURQUE, P. et al. The guide to the software engineering body of knowledge. **IEEE software**, IEEE, v. 16, n. 6, p. 35–44, 1999. Disponível em: <https://doi.org/10.1109/52.805471>.

BOX, D. et al. **Simple object access protocol (SOAP) 1.1**. 2000.

BRADEN, R. **Requirements for Internet Hosts - Communication Layers**. IETF, 1989. (Request for Comments, 1122). Published: RFC 1122 (INTERNET STANDARD). Disponível em: <https://doi.org/10.17487/rfc1122>.

BRAY, T. et al. **Extensible markup language (XML) 1.0**. [S.l.]: W3C recommendation October, 2000.

BREWER, E. A. Towards robust distributed systems. In: **PODC**. [s.n.], 2000. v. 7. Disponível em: <https://doi.org/10.1145/343477.343502>.

BUCHANAN, B. **The cybersecurity dilemma: Hacking, trust, and fear between nations**. Oxford University Press, 2016. Disponível em: <https://doi.org/10.1093/acprof:oso/9780190665012.001.0001>.

BUTLER, M. J. Stepwise refinement of communicating systems. **Science of Computer programming**, Elsevier, v. 27, n. 2, p. 139–173, 1996. Disponível em: <https://doi.org/10.1016/0167-6423(96)81173-7>.

CANINI, M. et al. A self-organizing distributed and in-band sdn control plane. In: **2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)**. [s.n.], 2017. p. 2656–2657. ISSN 1063-6927. Disponível em: <https://doi.org/10.1109/ICDCS.2017.328>.

CELDRÁN, A. H. et al. D7. 3 report and functional testing of the selfnet framework. SELFNET Consortium, 2018.

CHANDRASEKARAN, B.; TSCHAEN, B.; BENSON, T. Isolating and Tolerating SDN Application Failures with LegoSDN. In: **Proceedings of the Symposium on SDN Research**. New York, NY, USA: ACM, 2016. (SOSR '16), p. 7:1–7:12. ISBN 978-1-4503-4211-7.

CHARAN, R. How networks reshape organizations–for results. v. 69, n. 5, p. 104–115, 1991. ISSN 0017-8012. Disponível em: <http://europepmc.org/abstract/med/10113908>.

CHETTY, S. The case study method for research in small-and medium-sized firms. **International small business journal**, Sage Publications Sage CA: Thousand Oaks, CA, v. 15, n. 1, p. 73–85, 1996. Disponível em: <https://doi.org/10.1177/0266242696151005>.

CHOWDHURY, G. G. Natural language processing. **Annual review of information science and technology**, Wiley Online Library, v. 37, n. 1, p. 51–89, 2003. Disponível em: <https://doi.org/10.1002/aris.1440370103>.

Cisco. **Cisco and OpenFlow**. [S.l.: s.n.], 2013.

CLAISE, B. **Cisco Systems NetFlow Services Export Version 9**. IETF, 2004. (Request for Comments, 3954). Published: RFC 3954 (Informational). Disponível em: <https://doi.org/10.17487/rfc3954>.

CLEMM, A. **Network Management Fundamentals**. [S.l.]: Cisco Press, 2006. ISBN 978-1-58720-137-0.

COMMISSION, E. et al. **Horizon 2020**. 2015.

CONGDON, P. Link layer discovery protocol and MIB. **V1. 0 May**, v. 20, n. 2002, p. 1–20, 2002.

COX, J. H. et al. Advancing Software-Defined Networks: A Survey. **IEEE Access**, v. 5, p. 25487–25526, 2017. Disponível em: <https://doi.org/10.1109/ACCESS.2017.2762291>.

CROCKFORD, D. **The Application/Json Media Type for JavaScript Object Notation (JSON)**. IETF, 2006. (Request for Comments, 4627). Published: RFC 4627 (Informational). Disponível em: <https://doi.org/10.17487/rfc4627>.

CUI, C. et al. Network Functions Virtualisation. **SDN and OpenFlow World Congress**, 2012.

DAHLBERG, I. A referent-oriented, analytical concept theory for interconcept. **KO KNOWLEDGE ORGANIZATION**, Nomos Verlagsgesellschaft mbH & Co. KG, v. 5, n. 3, p. 142–151, 1978. Disponível em: <https://doi.org/10.5771/0943-7444-1978-3-142>.

DANNEN, C. **Introducing Ethereum and solidity**. Springer, 2017. v. 318. Disponível em: <https://doi.org/10.1007/978-1-4842-2535-6>.

DAY, J.; ZIMMERMANN, H. The OSI reference model. **Proceedings of the IEEE**, v. 71, n. 12, p. 1334–1340, dez. 1983. ISSN 0018-9219.

DB-ENGINES. **Popularity ranking of database management systems**. 2019. Disponível em: <https://db-engines.com/en/ranking>.

de Oliveira Silva, F. **Endereçamento por título: uma forma de encaminhamento multicast para a próxima geração de redes de computadores**. Tese (PhD Thesis) — Universidade de Sao Paulo, Sao Paulo, 2013. Disponível em: <https://doi.og/10.11606/T.3.2013.tde-22092014-111409>.

DIJKMAN, R. M.; DUMAS, M.; OUYANG, C. Semantics and analysis of business process models in bpmn. **Information and Software technology**, Elsevier, v. 50, n. 12, p. 1281–1294, 2008. Disponível em: <https://doi.org/10.1016/j.infsof.2008.02.006>.

DIJKSTRA, P. D. E. W. On the Role of Scientific Thought. In: **Selected Writings on Computing: A Personal Perspective**. Springer New York, 1982, (Texts and Monographs in Computer Science). p. 60–66. ISBN 978-1-4612-5697-7 978-1-4612-5695-3. Disponível em: <https://doi.org/10.1007/978-1-4612-5695-3_12>.

DING, J. **Advances in Network Management**. 0. ed. Auerbach Publications, 2016. ISBN 978-0-429-12091-6. Disponível em: <https://doi.org/10.1201/9781420064551>.

DORE, J. Holophrases, speech acts and language universals. **Journal of child language**, Cambridge University Press, v. 2, n. 1, p. 21–40, 1975. Disponível em: <https://doi.org/10.1017/S0305000900000878>.

DUEZ, P. P.; ZULIANI, M. J.; JAMIESON, G. A. Trust by Design: Information Requirements for Appropriate Trust in Automation. In: **Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative Research**. Riverton, NJ, USA: IBM Corp., 2006. (CASCON '06). Disponível em: <https://doi.org/10.1145/1188966.1188978>.

DUTRA, J.; IBERTIS, C. M. Ser enquanto ser: Acerca da filosofia primeira de aristóteles. **Disciplinarum Scientia| Ciências Humanas**, v. 4, n. 1, p. 1–18, 2003.

EGEVANG, K.; FRANCIS, P. **The IP Network Address Translator (NAT)**. IETF, 1994. (Request for Comments, 1631). Published: RFC 1631 (Informational). Disponível em: <https://doi.org/10.17487/rfc1631>.

EINSTEIN, A. The general theory of relativity. In: **The Meaning of Relativity**. Springer, 1922. p. 54–75. Disponível em: <https://doi.org/10.1007/978-94-011-6022-3_3>.

EMERY, D.; HILLIARD, R. Every architecture description needs a framework: Expressing architecture frameworks using iso/iec 42010. In: IEEE. **2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture**. 2009. p. 31–40. Disponível em: <https://doi.org/10.1109/WICSA.2009.5290789>.

ENNS, R. **NETCONF Configuration Protocol**. IETF, 2006. (Request for Comments, 4741). Published: RFC 4741 (Proposed Standard). Disponível em: <https://doi.org/10.17487/rfc4741>.

ETSI, G. Network functions virtualisation (nfv): Architectural framework. **ETsI Gs NFV**, v. 2, n. 2, p. V1, 2013.

FAHLAND, D. et al. Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. In: HALPIN, T. et al. (Ed.). **Enterprise, Business-Process and Information Systems Modeling**. Springer Berlin Heidelberg, 2009. (Lecture Notes in Business Information Processing), p. 353–366. ISBN 978-3-642-01862-6. Disponível em: <https://doi.org/10.1007/978-3-642-01862-6_29>.

FAUCHEUR, F. L. et al. **Multi-Protocol Label Switching (MPLS) Support of Differentiated Services**. [S.l.]: IETF, 2002. (Request for Comments, 3270). Published: RFC 3270 (Proposed Standard).

FENNER, B. **IANA Considerations for IPv4 Internet Group Management Protocol (IGMP)**. IETF, 2002. (Request for Comments, 3228). Published: RFC 3228 (Best Current Practice). Disponível em: <https://doi.org/10.17487/rfc3228>.

FERNANDES, E. L.; ROTHENBERG, C. E. OpenFlow 1.3 Software Switch. **Anais do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**, 2014.

FERREIRA, C. et al. Archsdn: a reinforcement learning-based autonomous openflow controller with distributed management properties. **SN Applied Sciences**, Springer, v. 2, n. 9, p. 1–21, 2020. Disponível em: <https://doi.org/10.1007/s42452-020-03316-7>.

FETTWEIS, G. P. The tactile internet: Applications and challenges. **IEEE Vehicular Technology Magazine**, IEEE, v. 9, n. 1, p. 64–70, 2014. Disponível em: <https://doi.org/10.1109/MVT.2013.2295069>.

FIELDING, R. Representational state transfer. **Architectural Styles and the Design of Netowork-based Software Architecture**, p. 76–85, 2000.

FONSECA, P. C. d. R.; MOTA, E. S. A Survey on Fault Management in Software-Defined Networks. **IEEE Communications Surveys Tutorials**, v. 19, n. 4, p. 2284–2321, Fourthquarter 2017.

FRIEDBERG, I. et al. Combating advanced persistent threats: From network event correlation to incident detection. **Computers & Security**, Elsevier, v. 48, p. 35–57, 2015. Disponível em: <https://doi.org/10.1016/j.cose.2014.09.006>.

GANEK, A. G.; CORBI, T. A. The dawning of the autonomic computing era. **IBM systems Journal**, IBM, v. 42, n. 1, p. 5–18, 2003. Disponível em: <https://doi.org/10.1147/sj.421.0005>.

GELENBE, E. Steps toward self-aware networks. **Communications of the ACM**, v. 52, n. 7, p. 66–75, 2009. Disponível em: <https://doi.org/10.1145/1538788.1538809>.

GIL, A. C. **Métodos e técnicas de pesquisa social**. [S.l.]: 6. ed. Ediitora Atlas SA, 2008.

GUARINO, N.; OBERLE, D.; STAAB, S. What is an ontology? In: **Handbook on ontologies**. Springer, 2009. p. 1–17. Disponível em: <https://doi.org/10.1007/978-3-540-92673-3_0>.

GUPTA, A.; JHA, R. K. A survey of 5g network: Architecture and emerging technologies. **IEEE Access**, v. 3, p. 1206–1232, 2015. ISSN 2169-3536. Disponível em: <https://doi.org/10.1109/ACCESS.2015.2461602>.

HAN, B. et al. Network function virtualization: Challenges and opportunities for innovations. **IEEE Communications Magazine**, IEEE, v. 53, n. 2, p. 90–97, 2015. Disponível em: <https://doi.org/10.1109/MCOM.2015.7045396>.

HARIRI, S. et al. Quality-of-protection (QoP)-an online monitoring and self-protection mechanism. **IEEE Journal on Selected Areas in Communications**, v. 23, n. 10, p. 1983–1993, out. 2005. ISSN 0733-8716. Disponível em: <https://doi.org/10.1109/JSAC.2005.854122>.

HARPER, D. et al. Online etymology dictionary. 2001.

HEGERING, H.-G. **Integrated Management of Networked Systems: Concepts, Architectures and Their Operational Application**. [S.l.]: Morgan Kaufmann, 1999. Google-Books-ID: pssD4DE8JlsC. ISBN 978-1-55860-571-8.

HENNING, M. The rise and fall of corba. **Queue**, ACM New York, NY, USA, v. 4, n. 5, p. 28–34, 2006. Disponível em: <https://doi.org/10.1145/1142031.1142044>.

HEVNER, A.; CHATTERJEE, S. Design science research in information systems. In: **Design research in information systems**. Springer, 2010. p. 9–22. Disponível em: <https://doi.org/10.1007/978-1-4419-5653-8_2>.

HORN, P. Autonomic computing: IBM's Perspective on the State of Information Technology. 2001.

HORRIDGE, M.; BECHHOFER, S.; NOPPENS, O. Igniting the OWL 1.1 Touch Paper: The OWL API. In: **OWLED**. [S.l.]: Citeseer, 2007. v. 258, p. 6–7.

HORRIDGE, M. et al. **Protégé**. [S.l.: s.n.], 2012.

HP. **Production-Ready SDN with OpenFlow 1.3**. [S.l.: s.n.], 2013.

Huawei. **Huawei the Only Vendor to Successfully Participate In EANTC's SDN OpenFlow 1.2 Test**. [S.l.: s.n.], 2013.

IBM. **Ibm Global Services and Autonomic Computing**. 2002. Ftp://ftp.software.ibm.com/software/ tivoli/whitepapers/wp-igs-autonomic.pdf.

IRWIN, C. E. Information systems methodologies: A framework for understanding. **Journal of the Operational Research Society**, Springer, v. 43, n. 3, p. 286–286, 1992. Disponível em: <https://doi.org/10.1057/jors.1992.40>.

ISO/IEC. ISO/IEC 7498-4:1989 — Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 4: Management framework. **International Organization for Standardization, International Electrotechnical Commission**, 1989.

_____. ISO/IEC 9595:1990 — Information technology — Open Systems Interconnection — Common management information service definition. **International Organization for Standardization, International Electrotechnical Commission**, 1990.

_____. ISO/IEC 9596-1:1991 — Information technology — Open Systems Interconnection — Common management information protocol — Part 1: Specification. **International Organization for Standardization, International Electrotechnical Commission**, 1991.

ITU-T, C. M.3010 : Principles for a telecommunications management network (TMN). **M3010**, 1992.

_____. X.700 : Management framework for open systems interconnection (OSI) for CCITT applications. **X.700**, 1993.

JR, J. F. N.; CHEN, M.; PURDIN, T. D. Systems development in information systems research. **Journal of management information systems**, Taylor & Francis, v. 7, n. 3, p. 89–106, 1990. Disponível em: <https://doi.org/10.1080/07421222.1990.11517898>.

KATIYAR, R. et al. Auto-Configuration of SDN Switches in SDN/Non-SDN Hybrid Network. In: **Proceedings of the Asian Internet Engineering Conference**. New York, NY, USA: ACM, 2015. (AINTEC '15), p. 48–53. ISBN 978-1-4503-3914-8. Event-place: Bangkok, Thailand. Disponível em: <https://doi.org/10.1145/2837030.2837037>.

KELLY, M. B. The telemanagement forum's enhanced telecom operations map (etom). **Journal of Network and Systems Management**, Springer, v. 11, n. 1, p. 109–119, 2003.

KIM, H.; FEAMSTER, N. Improving Network Management with Software Defined Networking. **IEEE Communications Magazine**, p. 114, 2013. Disponível em: <https://doi.org/10.1109/MCOM.2013.6461195>.

KLEPPMANN, M. A critique of the cap theorem. **arXivLabs**, 2015. Disponível em: <https://arxiv.org/abs/1509.05393>.

KOMPELLA, K. Self-Driving Networks. **Emerging Automation Techniques for the Future Internet**, p. 21–44, 2019. Disponível em: <https://doi.org/10.4018/978-1-5225-7146-9.ch002>.

KU, B. S. OAM amp;p technologies for optical networking. In: **OFC '98. Optical Fiber Communication Conference and Exhibit. Technical Digest. Conference Edition. 1998 OSA Technical Digest Series Vol.2 (IEEE Cat. No.98CH36177)**. [s.n.], 1998. p. 175–176. Disponível em: <https://doi.org/10.1109/OFC.1998.657308>.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. [S.l.]: Pearson Addison Wesley, 2006. ISBN 978-85-88639-18-8.

LAKATOS, E. M.; MARCONI, M. A. **Fundamentos de Metodologia Científica**. 5. ed. São Paulo, SP, Brasil: Atlas, 2003. ISBN 85-224-3397-6.

LALIBERTE, B. **The Journey to Intent-Based Networking**. 2018. 14 p.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In: **Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks**. New York, NY, USA: ACM, 2010. (Hotnets-IX), p. 19:1–19:6. ISBN 978-1-4503-0409-2.

LAWYER, J. et al. **Self-learning real-time prioritization of telecommunication fraud control actions**. [S.l.]: Google Patents, 2005. US Patent 6,850,606.

LEHTINEN, S.; LONVICK, C. **The Secure Shell (SSH) Protocol Assigned Numbers**. IETF, 2006. (Request for Comments, 4250). Published: RFC 4250 (Proposed Standard). Disponível em: <https://doi.org/10.17487/rfc4250>.

LLDPD. **lldpd - Implementation of IEEE 802.1AB (LLDP)**. 2019. Disponível em: <https://vincentbernat.github.io/lldpd/>.

LÖBACH, B. Design industrial. **São Paulo: Edgard Blücher**, 2001.

LóPEZ, L. B. et al. Key technologies in the context of future networks: operational and management requirements. **Future Internet**, Multidisciplinary Digital Publishing Institute, v. 9, n. 1, p. 1, 2017. Disponível em: <https://doi.org/10.3390/fi9010001>.

MADAKAM, S. et al. Internet of Things (IoT): A literature review. **Journal of Computer and Communications**, v. 3, n. 05, p. 164, 2015. Disponível em: <https://doi.org/10.4236/jcc.2015.35021>.

MAHMOUD, Q. **Cognitive networks: towards self-aware networks**. John Wiley & Sons, 2007. Disponível em: <https://doi.org/10.1002/9780470515143>.

MAURER, N.; WOLFTHAL, M. A. **Netty in Action**. [S.l.]: Manning Publications Co., 2015.

MCKEOWN, N. et al. OpenFlow: Enabling Innovation in Campus Networks. **SIGCOMM Comput. Commun. Rev.**, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833.

MCLARNON, B. et al. Introducing automated management through iteratively increased automation and indicators. In: **12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops**. Dublin, Ireland: IEEE, 2011. p. 1116–1121. ISBN 978-1-4244-9219-0. Disponível em: <https://doi.org/10.1109/INM.2011.5990522>.

MERKEL, D. Docker: lightweight linux containers for consistent development and deployment. v. 2014, n. 239, p. 2, 2014.

MESFIN, G. et al. Towards end-user development of rest client applications on smartphones. **Computer Standards & Interfaces**, Elsevier, v. 44, p. 205–219, 2016. Disponível em: <https://doi.org/10.1016/j.csi.2015.08.004>.

METZLER, J. The Changing Role of the Network Administrator. **Retrieved March**, v. 24, p. 2011, 2011.

MICHELSON, B. M. Event-driven architecture overview. **Patricia Seybold Group**, v. 2, n. 12, p. 10–1571, 2006. Disponível em: <https://doi.org/10.1571/bda2-2-06cc>.

MILLER, F. P.; VANDOME, A. F.; MCBREWSTER, J. **Apache Maven**. [S.l.]: Alpha Press, 2010.

MILLER, M. A. **PCIM '90 Proceedings of the 18th International Intelligent Motion Conference, Oct. 1990**. 18th edition. ed. Ventura, Calif.: Intertec Communications, Inc., 1990. ISBN 978-0-931033-28-5.

MOCKAPETRIS, P. V. **DNS Encoding of Network Names and Other Types**. IETF, 1989. (Request for Comments, 1101). Published: RFC 1101. Disponível em: <https://doi.org/10.17487/rfc1101>.

MOREIRA, M. D. et al. Internet do futuro: Um novo horizonte. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC 2009**, p. 1–59, 2009.

MORESI, E. et al. Metodologia da pesquisa. **Brasília: Universidade Católica de Brasília**, v. 108, p. 24, 2003.

MOSLEY, M.; LYNCH, J. **The story of science: Power, proof and passion**. [S.l.]: Mitchell Beazley, 2010.

NAMIOT, D.; SNEPS-SNEPPE, M. On micro-services architecture. v. 2, n. 9, p. 24–27, 2014.

NASCIMENTO, F. P. do. Classificação da pesquisa. natureza, método ou abordagem metodológica, objetivos e procedimentos. **Brasília: Thesaurus**, 2016.

NEUMANN, J. C. **The Book of GNS3**. 1st. ed. San Francisco, CA, USA: No Starch Press, 2014. ISBN 1593275544, 9781593275549.

NEVES, P. et al. The selfnet approach for autonomic management in an nfv/sdn networking paradigm. **International Journal of Distributed Sensor Networks**, SAGE Publications Sage UK: London, England, v. 12, n. 2, p. 2897479, 2016. Disponível em: <https://doi.org/10.1155/2016/2897479>.

NoviFlow. **NoviSwitch 1248 High Performance OpenFlow Switch**. [S.l.: s.n.], 2013.

NOY, N. F.; MCGUINNESS, D. L. et al. **Ontology development 101: A guide to creating your first ontology**. [S.l.]: Stanford knowledge systems laboratory technical report KSL-01-05 and . . . , 2001.

OLIVÉ, A. **Conceptual modeling of information systems**. [S.l.]: Springer Science & Business Media, 2007.

ONF. **OpenFlow Configuration and Management Protocol OF-CONFIG 1.0**. Open Networking Foundation, 2014. Disponível em: <http://opennetworking.wpengine.com/wp-content/uploads/2013/02/of-config1dot0-final.pdf>.

ON.LAB. **ONOS at ONS 2014**. 2014. Http://www.slideshare.net/ON_LAB/onos-at-ons-2014.

O'SULLIVAN, T. C. **Telnet Protocol - a Proposed Document**. [S.l.]: IETF, 1971. (Request for Comments, 137). Published: RFC 137.

PATIL, P.; GOKHALE, A.; HAKIRI, A. Bootstrapping Software Defined Network for flexible and dynamic control plane management. In: **Network Softwarization (NetSoft), 2015 1st IEEE Conference on**. IEEE, 2015. p. 1–5. Disponível em: <https://doi.org/10.1109/NETSOFT.2015.7116132>.

PAUL, S.; PAN, J.; JAIN, R. Architectures for the future networks and the next generation Internet: A survey. **Comput. Commun.**, v. 34, n. 1, p. 2–42, 2011. ISSN 0140-3664. Disponível em: <https://doi.org/10.1016/j.comcom.2010.08.001>.

PEFFERS, K. et al. A design science research methodology for information systems research. **Journal of management information systems**, Taylor & Francis, v. 24, n. 3, p. 45–77, 2007. Disponível em: <https://doi.org/10.2753/MIS0742-1222240302>.

PEPPLE, K. **Deploying openstack**. [S.l.]: "O'Reilly Media, Inc.", 2011.

PEREIRA, J. H. S. **Modelo de Titulo Para a Proxima Geracao de Internet**. Tese (PhD Thesis) — Universidade de Sao Paulo, Sao Paulo, dez. 2012.

PERREY, R.; LYCETT, M. Service-oriented architecture. In: IEEE. **2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings.** [S.l.], 2003. p. 116–119.

PFAFF, B.; DAVIE, B. **The Open vSwitch Database Management Protocol**. IETF, 2013. (Request for Comments, 7047). Published: RFC 7047 (Informational). Disponível em: <https://doi.org/10.17487/rfc7047>.

PFAFF, B. et al. The design and implementation of open vswitch. In: **12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)**. [S.l.: s.n.], 2015. p. 117–130.

PHAAL, P.; PANCHEN, S.; MCKEE, N. **InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks**. IETF, 2001. (Request for Comments, 3176). Published: RFC 3176 (Informational). Disponível em: <https://doi.org/10.17487/rfc3176>.

POPKIN, R. **The history of scepticism from Erasmus to Descartes**. [S.l.]: Read Books Ltd, 2013.

POSTEL, J. **Internet Protocol**. IETF, 1981. (Request for Comments, 791). Published: RFC 791 (INTERNET STANDARD). Disponível em: <https://doi.org/10.17487/rfc0791>.

PRAS, A. et al. **Introduction to TMN**. [S.l.]: Centre for Telematics and Information Technology, 1999.

PSAROMANOLAKIS, N. et al. Software defined networking in a converged 5g fiber-wireless network. In: IEEE. **2020 European Conference on Networks and Communications (EuCNC)**. 2020. p. 225–230. Disponível em: <https://doi.org/10.1109/EuCNC48522.2020.9200957>.

QUITTEK, J. et al. **Requirements for IP Flow Information Export (IPFIX)**. IETF, 2004. (Request for Comments, 3917). Published: RFC 3917 (Informational). Disponível em: <https://doi.org/10.17487/rfc3917>.

RAMIREZ-PEREZ, C.; RAMOS, V. Sdn meets sdr in self-organizing networks: fitting the pieces of network management. **IEEE Communications Magazine**, v. 54, n. 1, p. 48–57, January 2016. ISSN 0163-6804. Disponível em: <https://doi.org/10.1109/MCOM.2016.7378425>.

RAMIRO, J.; HAMIED, K. **Self-organizing networks: self-planning, self-optimization and self-healing for GSM, UMTS and LTE**. John Wiley & Sons, 2011. Disponível em: <https://doi.org/10.1002/9781119954224>.

RAZAVI, R.; KLEIN, S.; CLAUSSEN, H. Self-optimization of capacity and coverage in lte networks using a fuzzy reinforcement learning approach. In: IEEE. **21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications**. 2010. p. 1865–1870. Disponível em: <https://doi.org/10.1109/PIMRC.2010.5671622>.

RODRIGUES, A. d. J. **Metodologia científica: completo e essencial para a vida universitária**. Edição: 1ª. São Paulo: Editora Avercamp, 2006. ISBN 978-85-89311-30-4.

SADIGHI, A. et al. Design methodologies for enabling self-awareness in autonomous systems. In: **2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)**. IEEE, 2018. p. 1532–1537. Disponível em: <https://doi.org/10.23919/DATE.2018.8342259>.

SAROIU, S. et al. An analysis of internet content delivery systems. **ACM SIGOPS Operating Systems Review**, v. 36, n. SI, p. 315–327, 2002. Publisher: ACM New York, NY, USA. Disponível em: <https://doi.org/10.1145/844128.844158>.

SATYANARAYANAN, M. The emergence of edge computing. **Computer**, IEEE, v. 50, n. 1, p. 30–39, 2017. Disponível em: <https://doi.org/10.1109/MC.2017.9>.

SAYÃO, L. F. Modelos teóricos em ciência da informação-abstração e método científico. **Ciência da informação**, SciELO Brasil, v. 30, n. 1, p. 82–91, 2001. Disponível em: <https://doi.org/10.1590/S0100-19652001000100010>.

SCHULZ, D. Intent-based automation networks: Toward a common reference model for the self-orchestration of industrial intranets. In: IEEE. **IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society**. 2016. p. 4657–4664. Disponível em: <https://doi.org/10.1109/IECON.2016.7792959>.

SHARMA, S. et al. Automatic bootstrapping of OpenFlow networks. In: **2013 19th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)**. [s.n.], 2013. p. 1–6. Disponível em: <https://doi.org/10.1109/LANMAN.2013.6528283>.

SHOJAFAR, M. et al. P5g: A bio-inspired algorithm for the superfluid management of 5g networks. In: IEEE. **GLOBECOM 2017-2017 IEEE Global Communications Conference**. 2017. p. 1–7. Disponível em: <https://doi.org/10.1109/GLOCOM.2017.8254683>.

SIEGEL, J.; D., P. **CORBA 3 fundamentals and programming**. [S.l.]: John Wiley & Sons New York, NY, USA:, 2000. v. 2.

SIFAKIS, J. Autonomous systems–an architectural characterization. In: **Models, Languages, and Tools for Concurrent and Distributed Programming**. Springer, 2019. p. 388–410. Disponível em: <https://doi.org/10.1007/978-3-030-21485-2_21>.

SIMON, H. A. The sciences of the artificial. The MIT Press, 1969.

SIMSEK, M. et al. 5g-Enabled Tactile Internet. **IEEE Journal on Selected Areas in Communications**, v. 34, n. 3, p. 460–473, mar. 2016. ISSN 0733-8716. Disponível em: <https://doi.org/10.1109/JSAC.2016.2525398>.

SMIRNOV, M. et al. Demystifying self-awareness of autonomic systems. **ICT Mobile Summit, Santander, Spain, 10-12 June 2009**, 2009.

SNMP net. **Net-SNMP**. 2019. Disponível em: <http://www.net-snmp.org/>.

SOLDANI, C. et al. D5.1: Function allocation algorithms implementation and evaluation. SUPERFLUIDITY Consortium, 2018.

SOLOMON, H. Alcatel-lucent intros new gbethernet switches. **Network World Canada**, Laurentian Technomedia Inc., v. 24, n. 17, p. N_A, 2008.

SOMMERVILLE, I. Software engineering 9th edition. **ISBN-10**, v. 137035152, p. 18, 2011.

SOURCEFORGE. **net-snmp**. 2019. Disponível em: <https://sourceforge.net/projects/net-snmp/>.

STALLINGS, W. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2**. Addison-Wesley Longman Publishing Co., Inc., 1998. Disponível em: <https://doi.org/10.1109/COMST.1998.5340405>.

STEIN, A. et al. A concept for proactive knowledge construction in self-learning autonomous systems. In: **2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS* W)**. IEEE, 2018. p. 204–213. Disponível em: <https://doi.org/10.1109/FAS-W.2018.00048>.

STERRITT, R. Towards autonomic computing: Effective event management. In: IEEE. **27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002. Proceedings.** [S.l.], 2002. p. 40–47.

STEVENSON, D. W. Network management: What it is and what it isn't. **White Paper**, v. 4, 1995.

STRASSNER, J. et al. TMF white paper on NGOSS and MDA. In: **TeleManagement Forum/Object Management Group, February**. [S.l.: s.n.], 2004.

SUBRAMANIAN, M. **Network Management: Principles and Practice**. [S.l.]: Dorling Kindersley, 2010. Google-Books-ID: VGDMIIfL6XcC. ISBN 978-81-317-2759-1.

SVOBODA, J. et al. Network monitoring approaches: An overview. **Int J Adv Comput Netw Secur**, v. 5, n. 2, p. 88–93, 2015.

SWAN, M. **Blockchain: Blueprint for a new economy**. [S.l.]: "O'Reilly Media, Inc.", 2015.

THORAT, P. et al. Optimized self-healing framework for software defined networks. In: **Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication**. [S.l.]: ACM, 2015. p. 7.

TMF, T. F. **Telecom Operations Map: A High Level View of End-to-End Service Fulfillment, Service Assurance and Billing Processes**. [S.l.: s.n.], 1998.

TSAGKARIS, K.; BANTOUNA, A.; DEMESTICHAS, P. Self-organizing maps for advanced learning in cognitive radio systems. **Computers & Electrical Engineering**, Elsevier, v. 38, n. 4, p. 862–881, 2012. Disponível em: <https://doi.org/10.1016/j.compeleceng.2012.03.008>.

Tsagkaris, K. et al. Customizable autonomic network management: Integrating autonomic network management and software-defined networking. **IEEE Vehicular Technology Magazine**, v. 10, n. 1, p. 61–68, March 2015. Disponível em: <https://doi.org/10.1109/MVT.2014.2380633>.

UNION, I. T. **Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management framework**. International Telecommunication Union, 1989. Disponível em: <http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/01/42/14258.html>.

VAZQUEZ, C. E.; SIMÕES, G. S. **Engenharia de Requisitos: software orientado ao negócio**. [S.l.]: Brasport, 2016.

VERLINDE, E. On the origin of gravity and the laws of newton. **Journal of High Energy Physics**, Springer, v. 2011, n. 4, p. 1–27, 2011. Disponível em: <https://doi.org/10.1007/JHEP04(2011)029>.

VERMESAN, O. et al. Internet of robotic things: converging sensing/actuating, hypoconnectivity, artificial intelligence and IoT Platforms. 2017.

VILCHEZ, J. M. S.; YAHIA, I. G. B.; CRESPI, N. Self-healing mechanisms for software defined networks. In: **8th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2014)**. [S.l.: s.n.], 2014.

WAND, Y. et al. Theoretical foundations for conceptual modelling in information systems development. **Decision support systems**, Elsevier, v. 15, n. 4, p. 285–304, 1995. Disponível em: <https://doi.org/10.1016/0167-9236(94)00043-6>.

WANG, Q.; LU, P. Research on application of artificial intelligence in computer network technology. **International Journal of Pattern Recognition and Artificial Intelligence**, World Scientific, v. 33, n. 05, p. 1959015, 2019. Disponível em: <https://doi.org/10.1142/S0218001419590158>.

WEBER, R. Toward a theory of artifacts: A paradigmatic base for information systems research. **Journal of Information Systems**, v. 1, n. 2, p. 3–19, 1987.

WICKBOLDT, J. A. et al. Software-defined networking: management requirements and challenges. **IEEE Communications Magazine**, v. 53, n. 1, p. 278–285, January 2015. ISSN 0163-6804. Disponível em: <https://doi.org/10.1109/MCOM.2015.7010546>.

WOJCIECHOWSKI, R.; SIERSZEń, A.; STURGULEWSKI, \. Self-configuration networks. In: **Image Processing and Communications Challenges 7**. Springer, 2016. p. 301–308. Disponível em: <https://doi.org/10.1007/978-3-319-23814-2_34>.

XU, L. et al. Cognet: A network management architecture featuring cognitive capabilities. In: IEEE. **2016 European Conference on Networks and Communications (EuCNC)**. 2016. p. 325–329. Disponível em: <https://doi.org/10.1109/EuCNC.2016.7561056>.

YASMIN, J.; TIAN, Y.; YANG, J. A first look at the deprecation of restful apis: An empirical study. In: IEEE. **2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. 2020. p. 151–161. Disponível em: <https://doi.org/10.1109/ICSME46990.2020.00024>.

YOUSAFZAI, A.; HONG, C. S. SmartSON:A Smart contract driven incentive management framework for Self-Organizing Networks. **arXiv:2008.11803 [cs]**, ago. 2020. ArXiv: 2008.11803. Disponível em: <http://arxiv.org/abs/2008.11803>.

ZACHMAN, J. A. A framework for information systems architecture. **IBM systems journal**, IBM, v. 26, n. 3, p. 276–292, 1987. Disponível em: <https://doi.org/10.1147/sj.263.0276>.

ZAIDI, Z. et al. Will SDN Be Part of 5g? **IEEE Communications Surveys & Tutorials**, v. 20, n. 4, p. 3220–3258, 2018. ISSN 1553-877X, 2373-745X. Disponível em: <https://doi.org/10.1109/COMST.2018.2836315>.

ZANDER, S.; NGUYEN, T.; ARMITAGE, G. Self-learning ip traffic classification based on statistical flow characteristics. In: SPRINGER. **International Workshop on Passive and Active Network Measurement**. 2005. p. 325–328. Disponível em: <https://doi.org/10.1007/978-3-540-31966-5_26>.

ZANELLA, L. C. H. **Metodologia da pesquisa**. [S.l.]: SEAD/UFSC, 2006.

ZHANG, J.; XIE, W.; YANG, F. An architecture for 5g mobile network based on sdn and nfv. IET, 2015. Disponível em: <http://dx.doi.org/10.1049/cp.2015.0918>.

ZHAO, D. et al. Deep reinforcement learning with experience replay based on sarsa. In: IEEE. **2016 IEEE Symposium Series on Computational Intelligence (SSCI)**. 2016. p. 1–6. Disponível em: <https://doi.org/10.1109/SSCI.2016.7849837>.

Apêndices

APÊNDICE A

Análise multidimensional do Gerenciamento de Rede

Gerenciamento de Rede é usualmente tratado sob múltiplas perspectivas, cada uma com foco em um aspecto do gerenciamento. Essas perspectivas são conhecidas na literatura como *Dimensões de Gerenciamento* e visam o tratamento do gerenciamento de maneira completa considerando todos os fatores envolvidos, desde os comerciais até os técnicos. A ideia é dividir problemas de gerenciamento em diferentes aspectos com soluções parciais e assim prover uma solução mais completa.

Dimensões permitem uma análise espacial e comportamental dos aspectos de gerenciamento, o que permite correlacioná-los e interpretá-los. Dimensões independentes (sem correlação) são chamadas de *Dimensões Ortogonais* e contemplam aspectos únicos e normalmente complementares.

Hegering (1999) propôs o uso de cinco dimensões de gerenciamento: Disciplina, Tipo de rede, Área Funcional, Estágio e Tipo de informação. Já Clemm (2006) anos mais tarde propôs o uso de seis dimensões de gerenciamento: Função, Camada, Ciclo de vida, Objeto e Organização. As dimensões propostas por esses dois autores são representadas pelas Figuras 49 e 50 respectivamente. *Clemm* cita o trabalho de *Hegering* como referência, e por isso há similaridades entre os conceitos e as dimensões propostas.

A.1 Disciplina/Assunto

A Dimensão de *Disciplina*, proposta por *Hegering*, se refere a aspectos de gerenciamento com diferentes visões (ou focos). *Clemm* definiu essa dimensão com o nome de Dimensão de *Assunto*. Abaixo são listadas as disciplinas comuns em ambas as abordagens:

- *Rede*: responsável pelo gerenciamento de recursos de rede e tratamento de serviços de comunicação;

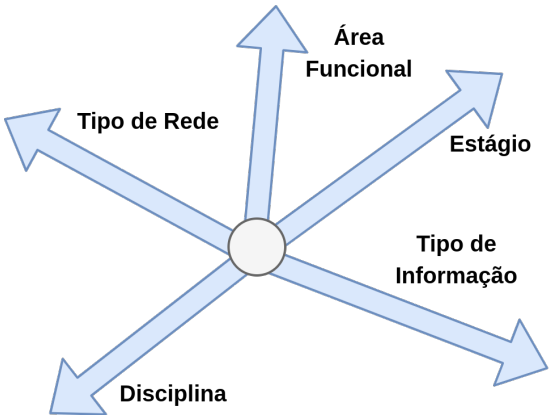


Figura 49 – Dimensões de Gerenciamento idealizadas por *Hegering*.

Fonte: Adaptado de Hegering (1999).

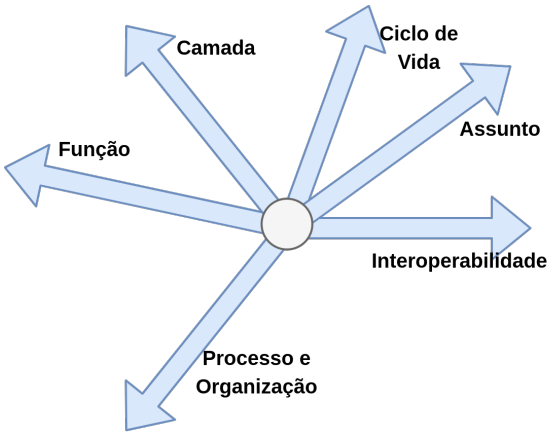


Figura 50 – Dimensões de Gerenciamento idealizadas por *Clemm*.

Fonte: Adaptado de Clemm (2006).

- *Sistema*: responsável pelo gerenciamento de sistemas finais (*hosts* e servidores);
- *Aplicação*: responsável pelo gerenciamento de aplicações rodando em sistemas finais;

Embora o gerenciamento dessas disciplinas tenham diversas similaridades, tais como o emprego de técnicas de monitoramento e correção, há também aspectos únicos de cada disciplina, como por exemplo: o Gerenciamento de Rede foca em garantir que dispositivos de rede estejam coordenados para o estabelecimento da comunicação; o Gerenciamento de Sistema tem a missão de garantir a "saúde" de servidores quanto a utilização, memória, disco e processamento; e por fim, o Gerenciamento de Aplicação tem a missão de manter as aplicações rodando a partir de atividades como atualização, licenciamento e reinicialização.

Hegering definiu três outras disciplinas que foram reorganizadas por *Clemm* em outras dimensões. É o caso das disciplinas de 'Serviço' e de 'Negócio' que foram abstraídas por *Clemm* na Dimensão de Camada, e da disciplina de 'Informação' que foi abstraída por *Clemm* na Dimensão de Interoperabilidade.

- *Informação*: responsável pelo gerenciamento da comunicação de acordo com o tipo de informação (música, vídeo, texto e etc);
- *Serviço*: responsável pelo gerenciamento do serviço em ambos os pontos de vista (do cliente e do provedor); e
- *Negócio*: responsável pelo gerenciamento estratégico do negócio, bem como pelos processos organizacionais e administrativos envolvidos.

A.2 Ciclo de Vida/Estágio

A Dimensão de *Ciclo de Vida*, proposta por *Clemm*, se refere a aspectos de gerenciamento específicos para cada estágios do ciclo de vida da rede, passando pelo planejamento e implantação até a operação e mudança. *Hegering* definiu essa dimensão com o nome de Dimensão de *Estágio*. Abaixo são listados os estágios comuns em ambas as abordagens:

- *Planejamento*: especifica um projeto de sistemas conectados que atenda aos requisitos de comunicação requeridos pelas aplicações com base em uma análise de custos e retorno;
- *Implantação*: responsável pelas atividades de campo (instalar/testar/substituir equipamentos e cabos), pela configuração de serviços e pela aplicação de políticas e técnicas de monitoramento especificadas;
- *Operação*: responsável pela continuidade do serviço através de técnicas como: monitoramento, *troubleshooting*, correção, melhoria e contabilidade; e
- *Mudança*: responsável pelo gerenciamento estratégico do negócio, bem como pelos processos organizacionais e administrativos envolvidos.

A.3 Função/Área Funcional

A Dimensão de *Função*, proposta por *Clemm*, se refere a aspectos de gerenciamento relacionados às diferentes funções necessárias para a manutenção do estado operacional da rede. *Hegering* definiu essa dimensão com o nome de Dimensão de *Área Funcional*. As funções dessa dimensão são as mesmas definidas pelo FCAPS:

- *Falha*: evitar ou corrigir falhas;
- *Configuração*: configurar a infraestrutura;
- *Contabilização*: medir a utilização dos recursos;
- *Desempenho*: avaliar e garantir níveis de serviço; e
- *Segurança*: prover aspectos como autenticação e remediação de ataques.

A.4 Camada

A Dimensão de *Camada*, proposta por *Clemm*, define camadas de responsabilidades no gerenciamento de rede de maneira similar à proposta pelo TMN:

- *Elemento de Rede*: gerenciamento local pelo próprio elemento;
- *Gerenciamento de Elemento*: gerenciamento do elemento de forma externa;
- *Gerenciamento de Rede*: gerenciamento do conjunto de elementos que compõe a rede;
- *Gerenciamento de Serviço*: gerenciamento dos serviços aprovisionados na rede; e
- *Gerenciamento de Negócio*: gerenciamento dos negócios para os quais os serviços foram aprovisionados.

A.5 Interoperabilidade

A Dimensão de *Interoperabilidade*, proposta por *Clemm*, contempla os aspectos relacionados à integração entre sistemas conectados. O gerenciamento desses sistemas é essencialmente uma solução compartilhada entre sistemas gerenciadores e dispositivos de rede. Estes sistemas precisam se comunicar e para isso precisam utilizar protocolos e linguagens específicas. A dimensão de interoperabilidade cuida do aspecto de definição de técnicas, protocolos e linguagens para a integração de sistemas conectados com três pontos de vista:

- *Comunicação*: qual o tipo de mensagens trocadas entre os sistemas?
- *Função*: a quais funções de gerenciamento essas mensagens estão relacionadas?
- *Informação*: como a informação pode ser representada?

As respostas a estas questões estabelecem padrões e normas que atuam como requisitos para o gerenciamento de rede. A escolha de tecnologias e protocolos pode estar relacionada a muitos fatores como custo, segurança e qualidade, e impactam em aspectos administrativos e de negócio.

A.6 Processo e Organização

A Dimensão de *Processo e Organização*, proposta por *Clemm*, é responsável pela tratativa de aspectos não técnicos do gerenciamento de rede relacionados à definição de processos e organização administrativa.

Os processos definidos podem variar de acordo com as dimensões das redes e criticidade dos serviços aprovisionados. Usualmente redes com aplicações críticas aplicam processos e alocam times técnicos em um sistema de plantão para corrigir erros no menor tempo possível. Por outro lado redes domésticas ou com serviços sem impactos significativos podem dispensar esses processos sem maiores prejuízos.

Normalmente provedores de acesso à Internet negociam *links* com acordos de níveis de serviço (SLA – *Service Level Agreement*) específicos. Nesse contexto, a definição de processos formais desempenha um importante papel na viabilização do negócio pois garante o ativo negociado.

A.7 Tipo de Informação

A Dimensão de *Tipo de Informação*, proposta por *Hegering*, considera aspectos de gerenciamento específicos de acordo com os tipos de informação trafegados na rede.

Os níveis de serviço estão intimamente ligados aos tipos de dados. Em geral, o tráfego de voz e vídeo em tempo real exige uma priorização da rede de maneira a melhorar a QoE dos usuários. Outros tipos de dados como arquivos via FTP, HTTP ou *torrents* não exigem os mesmos cuidados. Imagens podem eventualmente ser armazenadas em *cache* e conteúdo de texto pode ser compactado.

Um gerenciamento de rede eficiente deve ser sensível às necessidades de comunicação requeridas para cada tipo de dado, e deve provê-las de forma transparente.

A.8 Tipo de Rede

A Dimensão de *Tipo de Rede*, proposta por *Hegering*, considera aspectos de gerenciamento específicos para cada tipo de rede, sejam elas locais – corporativa ou doméstica; ou de longa distância – MAN, WAN e Internet. Diferentes tipos de redes requerem diferentes tipos de gerenciamento adequados a sua dimensão, criticidade e aplicação. Naturalmente redes corporativas exigem um gerenciamento mais completo do que redes domésticas. Da mesma forma, o gerenciamento de rede no contexto da Internet é uma tarefa muito mais complexa que o gerenciamento de LANs e MANs.

APÊNDICE B

Aspectos Conceituais sobre o Método de Pesquisa

Com a finalidade de apresentar claramente as estratégias utilizadas na concepção desta tese é oportuno definir aspectos conceituais sobre métodos de pesquisas. Deste modo, abaixo é apresentada uma definição etimológica dos termos metodologia e método:

- ***método*** vem do latim *methodus* que deriva dos termos gregos *metá* (atrás de, na direção de), *hodós* (caminho, maneira) (HARPER et al., 2001);
- ***metodologia*** vem da junção do termo *methodus* com o sufixo grego *logos* (estudo, discurso, palavra) (HARPER et al., 2001).

Rodrigues (2006) define 'metodologia' como uma disciplina que consiste em estudar, compreender e avaliar os vários métodos disponíveis para a condução de uma pesquisa cientifica; e, 'método' como um conjunto de técnicas formais e procedimentos passíveis de validação com um objetivo bem definido.

B.1 Conceituação de Ciência, *Design* e Pesquisa

O termo **ciência** vem do latim *scientia* (conhecimento) que deriva do verbo *scire* (saber) (HARPER et al., 2001) e consiste, de maneira geral, em um mecanismo para construção e organização de conhecimento (MOSLEY; LYNCH, 2010). O conceito de 'conhecimento' é definido por Wand et al. (1995) como uma "*representação cognitiva das coisas na realidade*", e dessa maneira, pode-se definir ciência como o meio pelo qual essa representação cognitiva é construída. A ciência enquanto atividade de investigação empírica começou a ser utilizada na antiguidade como uma vertente da filosofia concebida pelos filósofos pré-socráticos que questionavam senso comum da sociedade grega composta de mitos e dogmas (MOSLEY; LYNCH, 2010). Estes filósofos fundaram o 'ceticismo', movimento este caracterizado pelo questionamento de conhecimentos, fatos e opiniões que não

são suportados por evidências (POPKIN, 2013). A maneira como os pré-socráticos abordavam as questões filosóficas da época, sobretudo as relacionadas aos fenômenos naturais, começou a ser chamado "pensamento científico", e o rigor do ceticismo fundou as raízes do conceito de "método cientifico' através da observação e experimentação (POPKIN, 2013).

No século XIX, depois de acelerados avanços da ciência na idade média e no período renascentista com destaque para as 'leis do movimento' de Isaac Newton (VERLINDE, 2011) cujos princípios só seriam questionados no século XX com a teoria da relatividade geral de Albert Einstein (EINSTEIN, 1922), surgiu o conceito de "ciência moderna" que é dividida tradicionalmente nas áreas: (i) ciência natural, que lida com a natureza em um sentido amplo (e.g. física, biologia e química); (ii) ciência social, aplicada na análise de indivíduos e das sociedades (e.g. economia, sociologia e psicologia); e, (iii) ciência formal, que estuda conceitos abstratos (e,g, matemática, lógica e estatística). Estas áreas representam a 'ciência básica' que almeja a consolidação de conhecimento. Por outro lado, há também a 'ciência aplicada' que consiste na exploração do conhecimento de áreas teóricas em casos práticos com uma abordagem empírica (e.g. medicina, engenharia e computação) (HEVNER; CHATTERJEE, 2010).

A computação é essencialmente uma 'ciência aplicada' que viabiliza a concepção de elementos artificiais (e.g. redes de computadores) que não se esquadram em nenhuma grande área da ciência moderna tradicional. Para lidar com isso, Simon (1969) propôs o conceito de 'ciência artificial' que em contraposição à ciência natural lida com fenômenos criados pelo homem. Ao tratar *Tecnologia da Informação* (IT) como algo pertente ao domínio da ciência, Simon (1969) engloba o processo de criação das soluções tecnológicas, bem como os elementos de *hardware* e *software* que as compõe, como uma atividade que pode ser validada e replicada.

O método para a criação de 'coisas artificiais' é usualmente denominado de ***design*** e pode ser descrito segundo Löbach (2001) como "a concretização de uma ideia em forma de projetos ou mediante a construção e configuração resultando em um produto passível de produção em série". Hevner e Chatterjee (2010) descrevem o *design* de sistemas de informação de maneira simplificada como "o processo que lida com construção de artefatos de *software* que resolvem um problema humano". Sendo assim, embora não haja consenso alguns autores como Simon (1969) tratam *design* como uma atividade cientifica, e que portanto, deve ser respeitar os rigores do 'método cientifico' (HEVNER; CHATTERJEE, 2010).

A **pesquisa** é uma atividade inerentemente científica que permite gerar conhecimento. Lakatos e Marconi (2003) descreve pesquisa de maneira genérica como "uma atividade que contribui para a compreensão de um fenômeno". Hevner e Chatterjee (2010) descrevem a pesquisa de maneira mais precisa como "um processo através do qual tentamos alcançar de forma sistemática e com o apoio de dados a resposta a uma pergunta, a resolução de um problema ou uma maior compreensão de um fenômeno". O processo de pesquisa, usu-

almente chamado de 'metodologia de pesquisa', segundo Hevner e Chatterjee (2010) tem oito características: *i)* começa com uma questão ou problema; *ii)* requer a definição de um objetivo claro; *iii)* segue um plano de procedimentos pré-determinado; *iv)* normalmente divide o problema principal em subproblemas; *v)* é guiada pelo problema/questões de pesquisa ou hipótese; *vi)* permite a utilização de axiomas; *vii)* requer a coleta e interpretação de dados ou a criação de artefatos; e, *viii)* é naturalmente cíclica ou iterativa.

B.2 Ciência do *Design* : Pesquisa e Metodologia

Baseado nos conceitos apresentados anteriormente, os autores (HEVNER; CHATTERJEE, 2010) conceituam *Design Science Research* (DSR) como "um paradigma de pesquisa no qual um *designer* responde a perguntas relevantes aos problemas humanos por meio da criação de artefatos inovadores, contribuindo assim com novos conhecimentos para a comunidade com evidências científicas". Estes autores também propõe a utilização de sete diretrizes para guiar a aplicação da DSR:

- *Design* como um artefato – a pesquisa deve produzir um artefato viável, seja em nível conceitual ou prático;
- Relevância do problema – o objetivo da pesquisa é desenvolver uma solução para um problema real;
- Avaliação do *Design* – a utilidade, qualidade e eficácia de um artefato deve ser demonstrada de maneira rigorosa a partir de métodos de avaliação;
- Contribuições da Pesquisa – a pesquisa deve possibilitar a verificação de contribuições relevantes à resolução do problema de pesquisa;
- Rigor da Pesquisa – a pesquisa deve se amparar em rigorosos métodos de construção e avaliação dos artefatos da solução;
- *Design* como um processo de busca – a pesquisa deve se guiar pela busca de artefatos efetivos para as finalidades da pesquisa;
- Comunicação da Pesquisa – a pesquisa deve ser possível de ser apresentada (e entendida) por audiências técnicas e gerenciais.

Peffers et al. (2007) aprimora a abordagem DSR ao propor o *Design Science Research Methodology* (DSRM), uma metodologia para concepção de *design* de soluções baseada em seis atividades que são descritas abaixo na sequência de aplicação proposta pelo autor. Peffers et al. (2007) também apresenta essas atividades como um fluxo cíclico que pode ser observado na Figura 51.

1. *Identificação do Problema e Motivação* – responsável pela definição de um problema de pesquisa específico e justificativa do valor da solução. A definição do problema guia o desenvolvimento de uma solução eficaz e auxilia na análise da complexidade,

escopo e restrições da solução. A justificativa do valor motiva o pesquisador e o público-alvo a entender o problema e aceitar a solução;

2. *Objetivos da Solução* – consiste em definir objetivos de uma pesquisa a serem alcançados por uma solução a partir da definição de um problema. Os objetivos podem ser quantitativos ou qualitativos, e podem ser inferidos de maneira lógica a partir do estado da arte e especificação do problema;

3. *Design e Desenvolvimento* – consiste na criação dos artefatos que compõe a solução (e.g. modelos, métodos e instanciações). Esta atividade compreende desde os processos de modelagem, descrição de funcionalidades e organização arquitetural até as atividades de desenvolvimento dos artefatos;

4. *Demonstração* – responsável por demonstrar a eficácia dos artefatos na resolução do problema de pesquisa. Os métodos utilizados nesta atividade incluem experimentação, simulação, estudo de caso e prova conceitual;

5. *Avaliação* – responsável por demonstrar a eficácia dos artefatos na resolução do problema de pesquisa. O método utilizado nesta atividade consiste na análise de resultados qualitativos e quantitativos obtidos por meio da demonstração;

6. *Comunicação* – consiste na publicação e apresentação do trabalho para o público relevante (e.g. profissionais da área e pesquisadores) com o intuito de fomentar conhecimento e validar os artefatos desenvolvidos.

B.3 Tipos de Artefatos: Modelo, Arquitetura, *Framework* e Plataforma

O objetivo da pesquisa é gerar artefatos passíveis de serem utilizados para a resolução de um problema, sejam conceituais (e.g. modelos, métodos) ou aplicados (e.g. elementos de *software*) (WEBER, 1987). Artefatos podem ser concebidos com diferentes níveis de abstração, de acordo com os objetivos de pesquisa que eles resolvem (HEVNER; CHATTERJEE, 2010). Exemplos de artefatos que são utilizados neste trabalho são: (i) modelo – uma representação simplificada com enfoque em aspectos específicos; (ii) arquitetura - uma especificação de maneira estrutural com detalhes sobre os seus componentes e como eles se comunicam; (iii) *framework* – uma implementação de um arcabouço tecnológico com funções básicas que podem ser estendidas para aplicação em cenários específicos; e (iv) plataforma – um sistema que implementa funções de *core* sobre o qual outras aplicações podem ser executadas. Desta forma, a presente seção apresenta brevemente a etimologia e os conceitos destes artefatos.

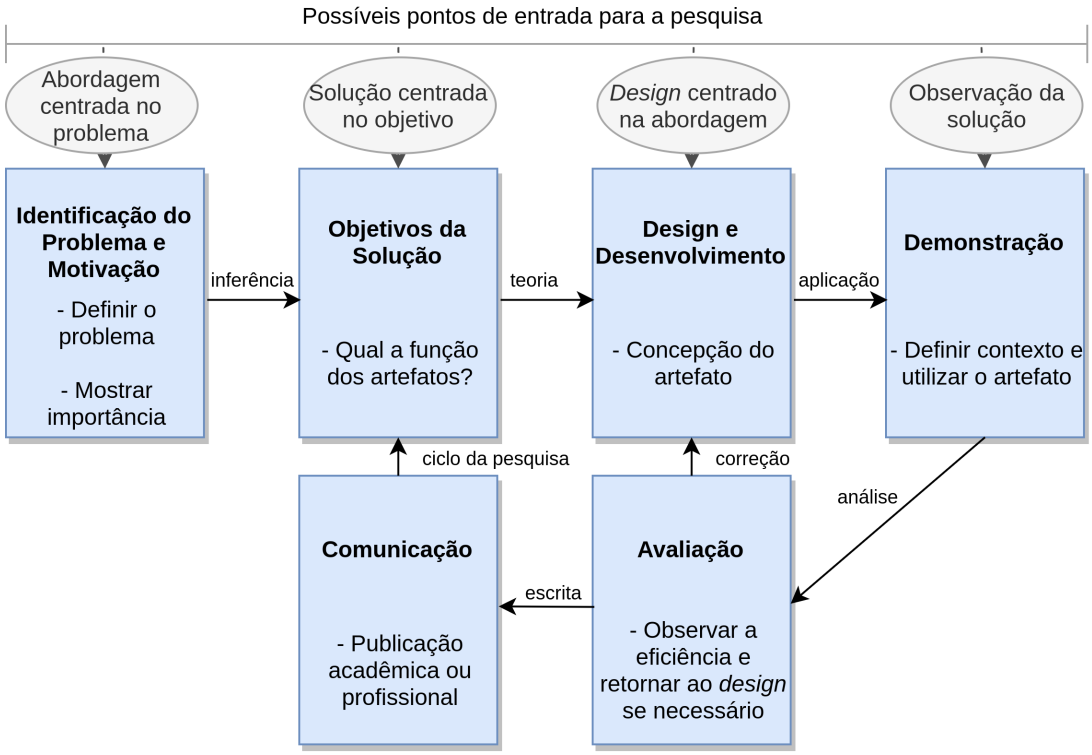


Figura 51 – Representação das etapas definidas pela *Design Science Research Methodology*.

Fonte: Adaptado de Peffers et al. (2007).

Modelo

O termo **modelo** vem do latim *modus* (modo ou maneira) (HARPER et al., 2001), e é utilizado usualmente pra se referir à uma visão simplificada que permite observar algo maior ou real com um menor número de detalhes (SAYÃO, 2001). O objetivo de um modelo é permitir a concepção de uma solução com alto nível de abstração em um ambiente controlado que permita o enfoque em aspectos específicos e relevantes ao problema da pesquisa. A construção de um modelo pode se dar por meio de um formalismo matemático, lógico ou conceitual (SAYÃO, 2001). Modelos matemáticos ou lógicos podem ser simulados, já modelos conceituais precisam ser materializados e observados.

Arquitetura

O termo **arquitetura** vem do grego *arkhé* (principal) e *tékhton* (construção) (HARPER et al., 2001) e é utilizado nos mais diversos ramos (e.g. matemática, engenharia e computação) como ato ou efeito de projetar algo passível de ser construído. A arquitetura de sistemas de informação segundo Zachman (1987) considera "toda a estrutura do sistema, desde o planejamento estratégico até o armazenamento". Mais especificamente no universo das redes o termo arquitetura usualmente se refere à organização das funções de

tratamento de dados e controle em camadas com protocolos de comunicação (KUROSE; ROSS, 2006). Considerando a aplicação do conceito de arquitetura em soluções que envolvam sistemas modulares, como é o caso das redes de computadores e de telecomunicações, podemos descrever arquitetura como o *design* de uma solução que inclui detalhes sobre seus componentes e estratégias de integração.

Framework

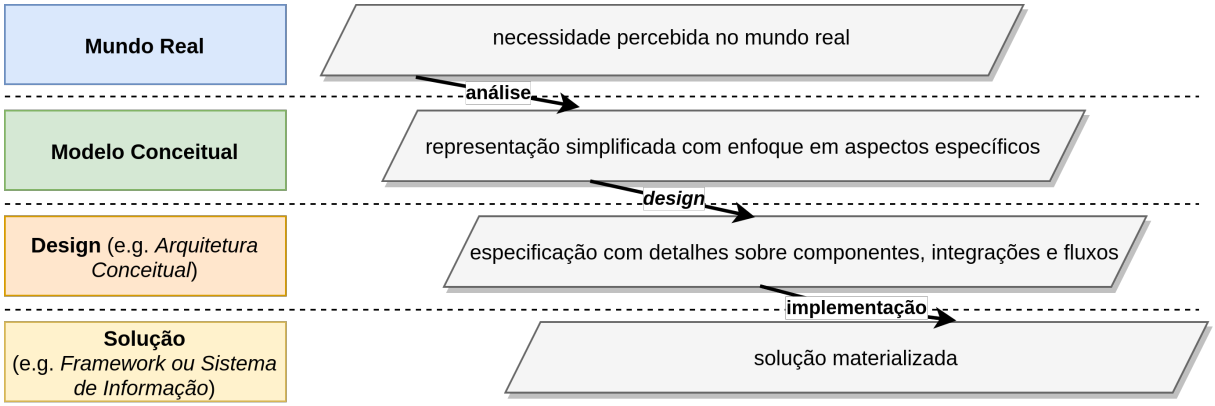
O termo ***framework*** tem origem na língua inglesa e é descrita no dicionário como uma estrutura de trabalho ou apoio (HARPER et al., 2001). No contexto dos sistemas de informação um *framework* é um arcabouço tecnológico que implementa partes genéricas de uma solução arquitetural com o objetivo de simplificar a implementação de ideias mais específicas. Desta maneira, uma solução concebida como um *framework* precisa ser guiada por padrões de projeto que garantam princípios de coesão e desacoplamento com o objetivo de flexibilizar o seu uso e extensão. A concepção de um *framework* a partir de uma arquitetura é um passo fundamental de evolução da solução (EMERY; HILLIARD, 2009), e uma possível forma de validação reside na experimentação e análise de resultados.

Plataforma

O termo **plataforma** tem origem na língua francesa do termo *platte fourme* que em tradução livre significa "em forma plana" e é descrita no dicionário como uma estrutura de apoio horizontal (HARPER et al., 2001). No contexto dos sistemas computacionais o termo 'plataforma' é usualmente utilizado para se referir à um conjunto de soluções de *software* sobre o qual pode-se implantar novos componentes e/ou utilizar de novas maneiras. Por exemplo, este termo é utilizado para se referir às soluções de *hardware*/*software* utilizadas nos *consoles* de jogos virtuais nas quais empresas desenvolvedoras destes jogos trabalham.

B.4 Modelagem Conceitual

A 'modelagem', ou seja, o processo pelo qual são construídos modelos, representa um aspecto fundamental para o desenvolvimento de Sistemas de Informação (ISs) (IRWIN, 1992). Um 'modelo' é basicamente uma representação simplificada de algo complexo que permite o enfoque em aspectos específicos (SAYÃO, 2001) (seção B.3). Wand et al. (1995) define 'modelos conceituais' como representações abstratas de um sistema sem detalhes sobre o aspecto de implementação. 'Modelos Conceituais' são especialmente úteis para fornecer uma abstração de alto nível a partir de uma necessidade do mundo real. A Figura 52 apresenta a visão de Wand et al. (1995) sobre o papel do 'modelo conceitual' no processo de *design* e *desenvolvimento* de um IS.



Inspirada em Wand et al. (1995)

Figura 52 – Representação do papel da Modelagem Conceitual no desenvolvimento de uma Solução de TI.

Como pode ser observado na Figura 52, Wand et al. (1995) propõe a modelagem conceitual de uma necessidade percebida no "mundo real" como primeira etapa do processo de desenvolvimento de um IS. Esses modelos capturam detalhes fundamentais sobre o domínio do problema que permitem o *design* de uma solução prática. O *design* de um IS usualmente inclui informações sobre os seus componentes, fluxos de trabalho e técnicas de integração. Um método amplamente utilizado pela Engenharia de *Software* com essa finalidade é a especificação por meio de UML (BOOCH; RUMBAUGH; JACOBSON, 2006) que permite a captura de diferentes aspectos do domínio do problema, tais como 'casos de uso', 'fluxos de trabalho' e 'modelo de dados'. Olivé (2007) descreve o conceito de 'arquitetura conceitual' como resultado do *design* de um sistema de maneira conceitual e estrutural sem maiores detalhes sobre a implementação. Por fim, o último passo do desenvolvimento de um IS segundo Wand et al. (1995) é a implementação que é o processo pelo qual o *design* é materializado. O presente trabalho materializa a solução proposta em duas etapas: *i)* implementação de um *framework* de auto-gerenciamento; e, *ii)* extensão/aplicação deste *framework* na automação de operações de configuração por meio de uma plataforma. A abstração da solução como um *framework* facilita a sua extensão para outras necessidades sem a necessidade de repetição dos processos de modelagem conceitual e *design*.

Wand et al. (1995) propõe a construção de modelos conceituais por meio de: *i)* ontologia, *ii)* teoria do conceito (*concept theory*) ou *iii)* linguística. A 'Ontologia' é uma abordagem filosófica pra representação do ser bem como de sua natureza que se define por meio de um conjunto de características, restrições e composições (GUARINO; OBERLE; STAAB, 2009). A 'Teoria do Conceito' é um modelo analítico de conceituação que permite a observação da natureza e estrutura dos conceitos de maneira a formalizá-los e classificá-los (DAHLBERG, 1978). E a 'Linguística', que é o estudo sobre a representação de informação com foco na comunicação. Mais especificamente Wand et al. (1995) cita a

utilização de uma área específica da linguística pra representação de modelos conceituais conhecida como 'teoria do ato de fala' (*speech act teory*) que é essencialmente um modelo declarativo que se baseia no uso de sentenças que capturam a intenção do falante e são escolhidas de acordo a causar um efeito pré-determinado nos ouvintes (DORE, 1975).

B.5 Classificação da Pesquisa

Em geral, a pesquisa pode ser classificada quanto à natureza, às abordagens, aos objetivos e aos procedimentos (NASCIMENTO, 2016). Abaixo, estes conceitos são apresentados brevemente.

- **Natureza**: tipo de resultado esperado com a pesquisa;
  - *básica*: objetiva gerar conhecimento;
  - *aplicada*: objetiva resolver um problema específico;
- **Abordagem**: os resultados podem ser traduzidos de maneira objetiva ou subjetiva;
  - *qualitativa*: análise de resultados com dados não quantificáveis mas que influenciam de maneira perceptível o objeto de pesquisa;
  - *quantitativa*: análise objetiva de resultados com dados quantificáveis e por isso mais facilmente comparáveis.
- **Objetivo**: o que se espera alcançar com a pesquisa;
  - *exploratória*: objetiva proporcionar informações sobre um problema – e.g. pesquisas bibliográficas e estudos de caso;
  - *descritiva*: objetiva caracterizar um fenômeno – e.g. levantamento;
  - *explicativa*: objetiva justificar um fenômeno – e.g. experimento;
- **Procedimento**: estratégia e técnicas utilizadas no desenvolvimento da pesquisa;
  - *levantamento bibliográfico*: é realizada por meio de pesquisa com material científico publicado (e.g. livros e artigos);
  - *aplicação experimental*: consiste na exploração da pesquisa em cenários relevantes com objetivo de gerar resultados de acordo com a abordagem definida;
  - *investigação documental*: se baseia na análise de material documental bruto (e.g. jornais, revistas e documentos);
  - *trabalho de campo*: consiste em atividades realizadas no local onde o fenômeno alvo da pesquisa ocorre (e.g. empresa de telecomunicações);

- *levantamento de informações*: consiste no levantamento de informações por meio de perguntas dirigidas ao conjunto de indivíduos relevante à pesquisa (e.g. censo);
- *análise ex-post-facto*: consiste na observação de um fenômeno ou experimento com o intuito de entendê-lo com base nos fatos resultantes em uma espécie de análise reversa de causa e efeito (e.g. análise causa-raiz de problemas de disponibilidade da rede);
- *estudo de caso*: consiste na aplicação da pesquisa (mesmo que conceitualmente) em um caso prático bem definido (e.g. automação da configuração de fluxos em *datacenters*);
- *prototipação*: envolve a concepção de uma versão preliminar do resultado pretendido com a pesquisa para testar aspectos específico e adiantar problemas de desenvolvimento (e.g. protótipo de componente de descoberta de topologia);
- *especificação formal*: consiste na representação da solução proposta com métodos formais de acordo com a área de aplicação (e.g. especificação UML).

B.6 Métodos de Avaliação da Pesquisa

Na visão de Hevner e Chatterjee (2010) os possíveis métodos avaliação dos artefatos produzidos por uma pesquisa podem ser classificados como:

❑ *Observacionais*

- *Estudo de Caso*: estudo do artefato com profundidade no contexto de sua aplicação;
- *Estudo de Campo*: monitoramento o uso do artefato em múltiplas aplicações;

❑ *Analíticos*

- *Análise Estática*: análise do artefato sobre suas características invariáveis (e.g. complexidade);
- *Análise Dinâmica*: análise do artefato sobre suas características dinâmicas (e.g. desempenho);
- *Análise Arquitetural*: análise das características do artefato em relação a definição arquitetural;
- *Otimização*: análise sobre características que tornam o artefato uma solução ótima;

❑ *Experimentais*

- *Experimento Controlado*: estudo do artefato em ambiente controlado e análise qualitativa (e.g. usabilidade);

- *Simulação*: execução do artefato com dados artificiais;

❑ *Testagem*

- *Funcional (Caixa Preta)*: executa o artefato por meio de suas interfaces para identificar falhas;
- *Estrutural (Caixa Branca)*: executa um teste de cobertura seguindo as linhas de execução do artefato;

❑ *Descritivos*

- *Argumentativo*: uso de informações consolidadas para convencimento via argumentação;
- *Cenários*: construção de cenários detalhados para demonstração da utilidade do artefato;

APÊNDICE C

Descrição Ontológica de Serviços Baseados em Intenções - *Intent-Based Services Ontology* (IBSO)

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://ufu.br/mehar/ibsrm"
     xml:base="http://ufu.br/mehar/ibsrm"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://ufu.br/mehar/ibsrm/"/>

    <!-- Object Properties -->
    <owl:ObjectProperty rdf:about="http://ufu.br/mehar/ibsrm/
       filter/restriction"/>
    <owl:ObjectProperty rdf:about="http://ufu.br/mehar/ibsrm/
       policy/requirement"/>
    <owl:ObjectProperty rdf:about="http://ufu.br/mehar/ibsrm/
       restriction/comparator"/>
    <owl:ObjectProperty rdf:about="http://ufu.br/mehar/ibsrm/
       service/filter"/>
    <owl:ObjectProperty rdf:about="http://ufu.br/mehar/ibsrm/
       service/function"/>
    <owl:ObjectProperty rdf:about="http://ufu.br/mehar/ibsrm/
       service/policy"/>
    <owl:ObjectProperty rdf:about="http://ufu.br/mehar/ibsrm/
       service/superService"/>

    <!-- Data properties -->
  <owl:DatatypeProperty rdf:about="http://ufu.br/mehar/ibsrm/
     policy/configuration"/>
    <owl:DatatypeProperty rdf:about="http://ufu.br/mehar/
       ibsrm/restriction/key"/>
    <owl:DatatypeProperty rdf:about="http://ufu.br/mehar/
```

```
        ibsrm/restriction/value"/>
24    <owl:DatatypeProperty rdf:about="http://ufu.br/mehar/ibsrm/
        service/modifier"/>
25    <owl:DatatypeProperty rdf:about="http://ufu.br/mehar/ibsrm/
        services/statement"/>
26
27      <!-- Classes -->
28      <owl:Class rdf:about="http://ufu.br/mehar/ibsrm/#
          Comparator"/>
29      <owl:Class rdf:about="http://ufu.br/mehar/ibsrm/#Filter"
          />
30      <owl:Class rdf:about="http://ufu.br/mehar/ibsrm/#Function
          "/>
31      <owl:Class rdf:about="http://ufu.br/mehar/ibsrm/#Policy"
          />
32      <owl:Class rdf:about="http://ufu.br/mehar/ibsrm/#
          Requirement"/>
33      <owl:Class rdf:about="http://ufu.br/mehar/ibsrm/#
          Restriction"/>
34      <owl:Class rdf:about="http://ufu.br/mehar/ibsrm/#Service"
          />
35
36      <!-- Comparator Individuals -->
37      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/equals">
38          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
              Comparator"/>
39      </owl:NamedIndividual>
40      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/equals_or_greater_than">
41          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
              Comparator"/>
42      </owl:NamedIndividual>
43      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/equals_or_lesser_than">
44          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
              Comparator"/>
45      </owl:NamedIndividual>
46      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/greater_than">
47          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
              Comparator"/>
48      </owl:NamedIndividual>
49      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/in">
50          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
              Comparator"/>
51      </owl:NamedIndividual>
52      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/lesser_than">
53          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
              Comparator"/>
54      </owl:NamedIndividual>
55      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/not_equals">
56          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
              Comparator"/>
57      </owl:NamedIndividual>
58      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
          /comparators/not_in">
```

```
59          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Comparator"/>
60      </owl:NamedIndividual>
61
62    <!-- Function Individuals -->
63      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /functions/anycast_interdomain">
64          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Function"/>
65      </owl:NamedIndividual>
66      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /functions/anycast_intradomain">
67          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Function"/>
68      </owl:NamedIndividual>
69      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /functions/broadcast">
70          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Function"/>
71      </owl:NamedIndividual>
72      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /functions/multicast_interdomain">
73          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Function"/>
74      </owl:NamedIndividual>
75      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /functions/multicast_intradomain">
76          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Function"/>
77      </owl:NamedIndividual>
78      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /functions/unicast_interdomain">
79          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Function"/>
80      </owl:NamedIndividual>
81      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /functions/unicast_intradomain">
82          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Function"/>
83      </owl:NamedIndividual>
84
85    <!-- Requirement Individuals -->
86      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/cheap_channel">
87          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
88      </owl:NamedIndividual>
89      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/cost_limited">
90          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
91      </owl:NamedIndividual>
92      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/dynamic_bandwidth">
93          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
94      </owl:NamedIndividual>
95      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/error_free_channel">
96          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
```

```
            Requirement"/>
97      </owl:NamedIndividual>
98      <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/fast_channel">
99          <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
100     </owl:NamedIndividual>
101     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/guaranteed_bandwidth">
102         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
103     </owl:NamedIndividual>
104     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/limited_bandwidth">
105         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
106     </owl:NamedIndividual>
107     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/optimal_channel">
108         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
109     </owl:NamedIndividual>
110     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/restricted_by_applications">
111         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
112     </owl:NamedIndividual>
113     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/restricted_by_content">
114         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
115     </owl:NamedIndividual>
116     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/restricted_by_devices">
117         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
118     </owl:NamedIndividual>
119     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/restricted_by_tecnology">
120         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
121     </owl:NamedIndividual>
122     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/restricted_by_territory">
123         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
124     </owl:NamedIndividual>
125     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/safe_channel">
126         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
127     </owl:NamedIndividual>
128     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/time_limited">
129         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
               Requirement"/>
130     </owl:NamedIndividual>
131     <owl:NamedIndividual rdf:about="http://ufu.br/mehar/ibsrm
           /requirements/usage_limited">
```

```
132         <rdf:type rdf:resource="http://ufu.br/mehar/ibsrm/#
                Requirement"/>
133     </owl:NamedIndividual>
134
135     <!--  General axioms -->
136     <owl:Restriction>
137         <owl:onProperty rdf:resource="http://ufu.br/mehar/
                ibsrm/filter/restriction"/>
138         <owl:allValuesFrom rdf:resource="http://ufu.br/mehar/
                ibsrm/#Restriction"/>
139         <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                ibsrm/#Filter"/>
140     </owl:Restriction>
141     <owl:Restriction>
142         <owl:onProperty rdf:resource="http://ufu.br/mehar/
                ibsrm/policy/requirement"/>
143         <owl:allValuesFrom rdf:resource="http://ufu.br/mehar/
                ibsrm/#Requirement"/>
144         <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                ibsrm/#Policy"/>
145     </owl:Restriction>
146     <owl:Restriction>
147         <owl:onProperty rdf:resource="http://ufu.br/mehar/
                ibsrm/restriction/comparator"/>
148         <owl:allValuesFrom rdf:resource="http://ufu.br/mehar/
                ibsrm/#Comparator"/>
149         <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                ibsrm/#Restriction"/>
150     </owl:Restriction>
151     <owl:Restriction>
152         <owl:onProperty rdf:resource="http://ufu.br/mehar/
                ibsrm/service/filter"/>
153         <owl:allValuesFrom rdf:resource="http://ufu.br/mehar/
                ibsrm/#Filter"/>
154         <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                ibsrm/#Service"/>
155     </owl:Restriction>
156     <owl:Restriction>
157         <owl:onProperty rdf:resource="http://ufu.br/mehar/
                ibsrm/service/function"/>
158         <owl:allValuesFrom rdf:resource="http://ufu.br/mehar/
                ibsrm/#Function"/>
159         <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                ibsrm/#Service"/>
160     </owl:Restriction>
161     <owl:Restriction>
162         <owl:onProperty rdf:resource="http://ufu.br/mehar/
                ibsrm/service/policy"/>
163         <owl:allValuesFrom rdf:resource="http://ufu.br/mehar/
                ibsrm/#Policy"/>
164         <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                ibsrm/#Service"/>
165     </owl:Restriction>
166     <owl:Restriction>
167         <owl:onProperty rdf:resource="http://ufu.br/mehar/
                ibsrm/service/superService"/>
168         <owl:allValuesFrom rdf:resource="http://ufu.br/mehar/
                ibsrm/#Service"/>
169         <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                ibsrm/#Service"/>
```

```
170         </owl:Restriction>
171         <owl:Restriction>
172             <owl:onProperty rdf:resource="http://ufu.br/mehar/
                    ibsrm/policy/configuration"/>
173             <owl:allValuesFrom rdf:resource="http://www.w3.org
                    /2001/XMLSchema#string"/>
174             <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                    ibsrm/#Policy"/>
175         </owl:Restriction>
176         <owl:Restriction>
177             <owl:onProperty rdf:resource="http://ufu.br/mehar/
                    ibsrm/restriction/key"/>
178             <owl:allValuesFrom rdf:resource="http://www.w3.org
                    /2001/XMLSchema#string"/>
179             <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                    ibsrm/#Restriction"/>
180         </owl:Restriction>
181         <owl:Restriction>
182             <owl:onProperty rdf:resource="http://ufu.br/mehar/
                    ibsrm/restriction/value"/>
183             <owl:allValuesFrom rdf:resource="http://www.w3.org
                    /2001/XMLSchema#string"/>
184             <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                    ibsrm/#Restriction"/>
185         </owl:Restriction>
186       <owl:Restriction>
187             <owl:onProperty rdf:resource="http://ufu.br/mehar/
                    ibsrm/service/modifier"/>
188             <owl:allValuesFrom rdf:resource="http://www.w3.org
                    /2001/XMLSchema#string"/>
189             <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                    ibsrm/#Service"/>
190         </owl:Restriction>
191         <owl:Restriction>
192             <owl:onProperty rdf:resource="http://ufu.br/mehar/
                    ibsrm/services/statement"/>
193             <owl:allValuesFrom rdf:resource="http://www.w3.org
                    /2001/XMLSchema#string"/>
194             <rdfs:subClassOf rdf:resource="http://ufu.br/mehar/
                    ibsrm/#Service"/>
195         </owl:Restriction>
196 </rdf:RDF>
197
198 <!-- Generated by the OWL API (version 5.1.17.2020-11-07T15
       :03:35Z) https://github.com/owlcs/owlapi/ -->
```