UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Orquestração de Serviços *SDN* com Controladores Heterogêneos em Domínio Administrativo Único

Dissertação de Mestrado

Marco Aurélio Cruz Fonseca

São Cristóvão – Sergipe

2020

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marco Aurélio Cruz Fonseca

Orquestração de Serviços *SDN* com Controladores Heterogêneos em Domínio Administrativo Único

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Prof. Dr. Ricardo José Paiva de Britto Salgueiro

São Cristóvão – Sergipe

2020

Marco Aurélio Cruz Fonseca
Orquestração de Serviços *SDN* com Controladores Heterogêneos em Domínio Administrativo Único/ Marco Aurélio Cruz Fonseca. – São Cristóvão – Sergipe, 2020-
83 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Ricardo José Paiva de Britto Salgueiro

Dissertação de Mestrado – UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO, 2020.

1. SDN. 2. Orquestração 3. Teoria do Perigo 4. Anonimização
CDU 02:141:005.7

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: Marco Aurélio Cruz Fonseca

Em 31 dias do mês de agosto do ano de dois mil e vinte, com início às 15h00min, realizou-se na Sala virtual xxxxxxxxxxx. A Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Marco Aurélio Cruz Fonseca**, que desenvolveu o trabalho intitulado: "*Orquestração de serviços SDN com controladores heterogêneos em domínio administrativo único*", sob a orientação do Prof. Dr. **Ricardo José Paiva de Britto Salgueiro**. A Sessão foi presidida pelo Prof. Dr. **Ricardo José Paiva de Britto Salgueiro** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Rubens de Souza Matos Júnior** (PROCC/UFS), Profª Dra. **Edilayne Meneses Salgueiro** (DComp/UFS) e, em seguida, a Profª. Dra. **Fernanda Maria Ribeiro de Alencar** (UFPE - Universidade Federal de Pernambuco). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) Aprovado "*(aprovado/reprovado)*". Atendidas as exigências da Instrução Normativa 01/2017/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), Resolução nº 25/2014/CONEPE e da Portaria nº 413 de 27 de maio de 2020 (Banca por videoconferência) que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos*", 31 de agosto de 2020.
"participação à distância por videoconferência"

Prof. Dr.Ricardo José Paiva de Britto Salgueiro
(PROCC/UFS)
Presidente

Prof. Dr. Rubens de Souza Matos Júnior
(PROCC/UFS)
Examinador Interno

Profº. Dra. Edilayne Meneses Salgueiro
(Dcomp-UFS)
Examinador Externo

Profº. Dra. Fernanda Maria Ribeiro de Alencar
(UFPE)
Examinador Externo

Marco Aurélio Cruz Fonseca
Candidato

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE SERGIPE
GABINETE DO REITOR

DECLARAÇÃO DE PARTICIPAÇÃO REMOTA EM BANCA EXAMINADORA

Declaro que no dia 31/08/2020, às 15 horas participei, de forma remota com os demais membros deste ato público, da banca examinadora de defesa da dissertação de mestrado, do discente Marco Aurélio Cruz Fonseca do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe – UFS.

Considerando o trabalho avaliado, as arguições de todos os membros da banca e as respostas dadas pelo(a) discente(a), formalizo para fins de registro, minha decisão de que o(a) discente está _____Aprovado_____(Aprovado(a) ou Reprovado(a)).

Atenciosamente,

_____
Prof. Dr. Ricardo José Paiva de Britto Salgueiro
Presidente da banca
Orientador

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE SERGIPE
GABINETE DO REITOR

DECLARAÇÃO DE PARTICIPAÇÃO REMOTA EM BANCA EXAMINADORA

Declaro que no dia 31/08/2020, às 15 horas participei, de forma remota com os demais membros deste ato público, da banca examinadora de defesa da dissertação de mestrado, do discente Marco Aurélio Cruz Fonseca do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe – UFS.

Considerando o trabalho avaliado, as arguições de todos os membros da banca e as respostas dadas pelo(a) discente(a), formalizo para fins de registro, minha decisão de que o(a) discente está _Aprovado_(Aprovado(a) ou Reprovado(a)).

Atenciosamente,

Prof. Dr. Rubens de Souza Matos Júnior
Avaliador interno - UFS

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE SERGIPE
GABINETE DO REITOR

DECLARAÇÃO DE PARTICIPAÇÃO REMOTA EM BANCA EXAMINADORA

Declaro que no dia 31/08/2020, às 15 horas participei, de forma remota com os demais membros deste ato público, da banca examinadora de defesa da dissertação de mestrado, do discente Marco Aurélio Cruz Fonseca do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe – UFS.

Considerando o trabalho avaliado, as arguições de todos os membros da banca e as respostas dadas pelo(a) discente(a), formalizo para fins de registro, minha decisão de que o(a) discente está _Aprovado_(Aprovado(a) ou Reprovado(a)).

Atenciosamente,

Profª Dra. Edilayne Meneses Salgueiro
Avaliador externo - DComp/UFS

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE SERGIPE
GABINETE DO REITOR

DECLARAÇÃO DE PARTICIPAÇÃO REMOTA EM BANCA EXAMINADORA

Declaro que no dia 31/08/2020, às 15 horas participei, de forma remota com os demais membros deste ato público, da banca examinadora de defesa da dissertação de mestrado, do discente Marco Aurélio Cruz Fonseca do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe – UFS.

Considerando o trabalho avaliado, as arguições de todos os membros da banca e as respostas dadas pelo(a) discente(a), formalizo para fins de registro, minha decisão de que o(a) discente está _Aprovado_(Aprovado(a) ou Reprovado(a)).

Atenciosamente,

Fernanda Alencar

Profª Dra. Fernanda Maria Ribeiro de Alencar
Avaliador externo - UFPE

Dedico este trabalho primeiramente a Deus, que dá sentido à minha existência; à toda minha família, principalmente à minha esposa Camila e à minha filha Aurora, que me deram todo suporte e apoio para que meu empenho fosse bem sucedido, compreendendo que minhas ausências durante essa jornada foram difíceis, porém necessárias; aos meus amigos da equipe do STI, cuja torcida para esta conquista sempre esteve presente; aos meus amigos do ELAN, que compartilharam comigo os desafios da pesquisa, fracassos e sucessos ao longo do caminho; e aos professores, que cumpriram papel além da orientação acadêmica, contribuindo para meu crescimento como pesquisador, profissional e ser humano.

Agradecimentos

Agradeço ao meu orientador, Prof. Dr. Ricardo José Paiva de Britto Salgueiro, e à Prof[a]. Dr[a]. Edilayne Meneses Salgueiro pela companhia e orientação durante toda a pesquisa, me guiando pelo caminho científico e pela acolhida no laboratório. Agradeço também à Carolina Santana Louzada, cuja parceria foi essencial para superarmos os obstáculos e encontrarmos as soluções para que o experimento fosse bem sucedido. Não poderia deixar de agradecer aos amigos do ELAN (Filipe, Itauan, Jonathan, Lucas, Túlio e Wesley), graças ao convívio com vocês minhas forças para perseverar nesta empreitada eram sempre renovadas. Finalmente aos amigos do STI (Alexandre, Dilton, Eric, Fernandes, Karina, Kleber, Marcos, Rodrigus e Tiago), fica registrado meu muito obrigado!

Valha-me Nossa Senhora, Mãe de Deus de Nazaré!
A vaca mansa dá leite, a braba dá quando quer.
A mansa dá sossegada, a braba levanta o pé.
Já fui barco, fui navio, mas hoje sou escaler.
Já fui menino, fui homem, só me falta ser mulher.
Valha-me Nossa Senhora, Mãe de Deus de Nazaré!
(O Auto da Compadecida)

Resumo

As novas tecnologias desenvolvidas para os desafios crescentes nas redes de computadores, tais como o SDN, desempenham um papel fundamental na inovação da infraestrutura atual, mas mantêm problemas na sua gestão. Se por um lado SDN introduz facilidades ao gerenciamento e configuração das redes com a separação dos planos de controle e dados, por outro lado, a diversidade de soluções implementadas através de controladores heterogêneos em um mesmo domínio vem promovendo incompatibilidades na integração dos serviços. Diante desse cenário, é aqui proposta a adoção de uma arquitetura para a orquestração de serviços em SDN que permite a convivência de controladores heterogêneos em um domínio administrativo único. A solução é demonstrada a partir da orquestração dos serviços de detecção de perigo e de anonimização de pacotes IP em controladores heterogêneos. Este trabalho orquestra os serviços de segurança MAdPE-K / SDN e BomIP, alcançando um tratamento de ameaças mais robusto do que aqueles oferecidos separadamente. Desenvolvido para o controlador Ryu, o MAdPE-K / SDN monitora os sinais da rede e reage de forma bioinspirada, de acordo com a teoria do perigo, derrubando os fluxos percebidos como ameaças. O BomIP, por outro lado, isola os fluxos da rede através da anonimização dos endereços implementados em um controlador RunOS, de forma que qualquer perigo seja impedido de atingir o restante da rede. O serviço de orquestração foi implementado em SDN com um único domínio administrativo, provando que é possível integrar diferentes controladores trabalhando juntos sem subdividir o plano de dados. Os casos de uso testados demonstraram o equilíbrio do plano de controle com referência ao nível de ameaça, onde o orquestrador coordenou o tratamento de segurança mais adequado entre os controladores.

Palavras-chave: SDN, Orquestração, Teoria do Perigo, Anonimização, Segurança de Redes, Gerência de Redes, Redes Autonômicas.

Abstract

The new technologies developed for the growing challenges in computer networks, such as SDN, play a fundamental role in the innovation of the current infrastructure, but maintain problems in their management. On the one hand, SDN introduces facilities for network management and configuration with the separation of control and data plans, on the other hand, the diversity of solutions implemented through heterogeneous controllers in the same domain has been promoting incompatibilities in the integration of services. In view of this scenario, it is proposed here to adopt an architecture for the SDN services orchestration that allows the coexistence of heterogeneous controllers in a single administrative domain. The solution is demonstrated by the orchestration of the danger detection and anonymization services of IP packets in heterogeneous controllers. This work orchestrates the MAdPE-K / SDN and BomIP security services, achieving a more robust threat treatment than those offered separately. Developed for the Ryu controller, MAdPE-K / SDN monitors network signals and reacts in a bio-inspired manner, according to the theory of danger, dropping the flows perceived as threats. BomIP, on the other hand, isolates network flows by anonymizing the addresses implemented in a RunOS controller, so that any danger is prevented from reaching the rest of the network. The orchestration service was implemented in SDN with a single administrative domain, proving that it is possible to integrate different controllers working together without subdividing the data plan. The tested use cases demonstrated the balance of the control plan with reference to the threat level, where the orchestrator coordinated the most appropriate security treatment among the controllers.

Keywords: SDN, Orchestration, Theory of Danger, Anonymization, Network Secutiry, Network Management, Autonomic Networks.

Lista de ilustrações

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

SDN	Software-Defined Networking
OF	OpenFlow
ABNO	Application-based Network Operations
NFV	Network Function Virtualization
VNF	Virtualized Network Function
OPEX	Operational Expenditures
CAPEX	Capital Expenditures
COTS	Common Off-The-Shelf
VM	Virtual Machine
TE	Traffic Engineering
NMS	Network Management System
GMPLS	Generalized Multiprotocol Label Switching
ONF	Open Networking Foundation
QoS	Quality of Service
QoE	Quality of Experience
SDO	Standard Developing Organization
KPI	Key Performance Indicators
REST	Representational State Transfer
JSON	JavaScript Object Notation
SON	Self-Organizing Network
DoS	Denial Of Service
TLS/SSL	Transport Layer Security/Secure Sockets Layer
DC	Dendritic Cell
ADC	Artificial Dendritic Cell

Sumário

1 Introdução

1.1 Contextualização

As redes de computadores são ambientes dinâmicos e complexos, onde o gerenciamento e manutenção continuam sendo desafios para os seus administradores. O grande número e a diversidade de equipamentos, com eventos ocorrendo de maneira simultânea, são o cenário corriqueiro de trabalho destes profissionais. Aos administradores de redes cabe a responsabilidade por traduzir as políticas definidas em alto grau de abstração através de inúmeros elementos distribuídos pelo ambiente, via configurações de baixo nível. Esta tarefa extremamente custosa envolve a lida com diferentes padrões proprietários fechados, dos diversos fabricantes que compõe a infraestrutura administrada. Até hoje em dia poucos ou nenhum mecanismo de automatização dessas atividades são providos, o que é essencial em se tratando de um ambiente cujas condições mudam a todo momento. Porém, a falta de integração e interfaces quando se trata de equipamentos fabricados com configurações proprietárias e fechadas torna extremamente difícil a introdução de novos protocolos e soluções que permitam avançar na necessária automação. É preciso otimizar os esforços dos administradores, já que as demandas por novas funcionalidades e políticas de uso continuam a crescer. Essas atividades mecânicas e repetitivas, que tomam o tempo das tarefas mais complexas e que realmente demandam intervenção humana, devem passar a ser resolvidas de forma automática para tornar a administração das redes eficiente. (HYOJOON; FEAMSTER, 2013).

Todas as questões citadas ganham proporções e complexidades muito maiores quando vamos tratar da Internet. De acordo com Tayyaba et al. (2017), redes tradicionais não são capazes de lidar com o número de dispositivos conectados (na casa de bilhões) e de manipulação de dados gigantes (beirando já *exabytes*) que se conjectura para o cenário da Internet em 2020. Somado a isso a alta pressão para diminuir os custos operacionais (do inglês, *Operational Expeditures* (OPEX)), tornar mais sustentáveis as despesas de capital (do inglês, *Capital Expeditures* (CA-

PEX)), melhorar a experiência do usuário, dentre outras necessidades. Todas essas exigências têm pressionado a mudança do atual ambiente gerenciado de modo semi-automático para um ambiente inteligente e autônomo, onde tecnologias como Redes Definidas por Software (do inglês, *Software-Defined Network* (SDN)) terão papel crucial (NEVES et al., 2016). Apesar de ser uma tecnologia explicada com maior profundidade mais adiante (Seção 3.2), para entendimento desta contextualização podemos dizer que SDN atende a essas exigências ao propor um novo paradigma de redes onde a divisão da infraestrutura em dois planos (de controle e de dados) e a inserção do *software* controlador passam a ser o foco do trabalho com as novas redes. Com esse novo paradigma, torna-se possível a otimização do esforço do administrador e automatização dos processos de configuração dos dispositivos.

Apesar da importância para as atuais necessidades dos serviços oferecidos pela Internet, a heterogeneidade das infraestruturas de redes existentes torna inviável ou limitada a implantação de novas tecnologias (CASELLAS et al., 2015). O trabalho dos gestores de redes em manter e inovar suas infraestruturas tem crescido expressivamente, visto que as infraestruturas planas ou hierarquizadas de gerenciamento têm tornado impraticável o controle de um ecossistema cada vez maior de dados e equipamentos conectados, sendo um complexo e crescente desafio para a engenharia de tráfego (do inglês, *Traffic Engineering* (TE)) abstrair e distribuir o controle. Isso sem falarmos que a alta demanda popular por serviços como *streaming* de vídeo em alta definição já desafia os limites do *status quo* das redes, e serviços críticos, como aplicações de telemedicina, estão simplesmente fora de cogitação, caso não se garanta a devida QoS (do inglês, *Quality of Service*) (KOTRONIS et al., 2016).

Em Neves et al. (2016) é explorada a integração das tecnologias de ponta para redes, além da já citada SDN. Em conjunto com a Inteligência Artificial, Virtualização de Função de Rede (do inglês, *Network Function Virtualization* (NFV)), Rede Auto-Organizável (do inglês, *Self-Organizing Network* (SON)) e Computação em Nuvem (do inglês, *Cloud Computing*) é descrito um sistema de gerenciamento de redes escalável, extensível e inteligente. Em resposta às exigências atuais do mercado, tais como conectividade 5G, é preciso proteger as redes dos ataques, curar suas falhas, otimizar seus recursos e configurar seus dispositivos automaticamente. Todavia, ao mesmo tempo que somente através das tecnologias de ponta a automatização torna-se possível, como dito em Rotsos et al. (2017a), cada fabricante desenvolve protocolos e mecanismos próprios para prover esses serviços autônomos e automatizados. Assim sendo, interoperabilidade e padronização são cruciais para o sucesso da integração e convívio dessas tecnologias.

Aqui vale um aprofundamento do termo “automação” (e seus equivalentes). Neste trabalho ele diz respeito a alguma tarefa da rede que pôde ser delegada do humano para solução de *software* e/ou *hardware*, de modo a se obter uma maior eficiência na sua execução e um melhor aproveitamento do trabalho do administrador. A intenção com isso é deixar de dedicar esforço naquilo que é repetitivo e manual nas tarefas rotineiras de gerenciamento da rede para

dedicar às demandas complexas e que realmente dependam do intelecto humano para solução. De acordo com conceitos herdados de redes autonômicas, a depender da tarefa da rede onde a automação ocorre, diz-se que essa rede tem como característica:

- auto-cura: quando tem capacidade de recuperação de gargalos ou falhas;
- auto-proteção: quando reage a ataques e incidentes de segurança;
- auto-otimização: quando adapta-se às condições e cargas do momento;
- auto-configuração: quando conecta de modo *plug-and-play* equipamentos e/ou serviços à infraestrutura;

1.2 Problemática e Hipótese

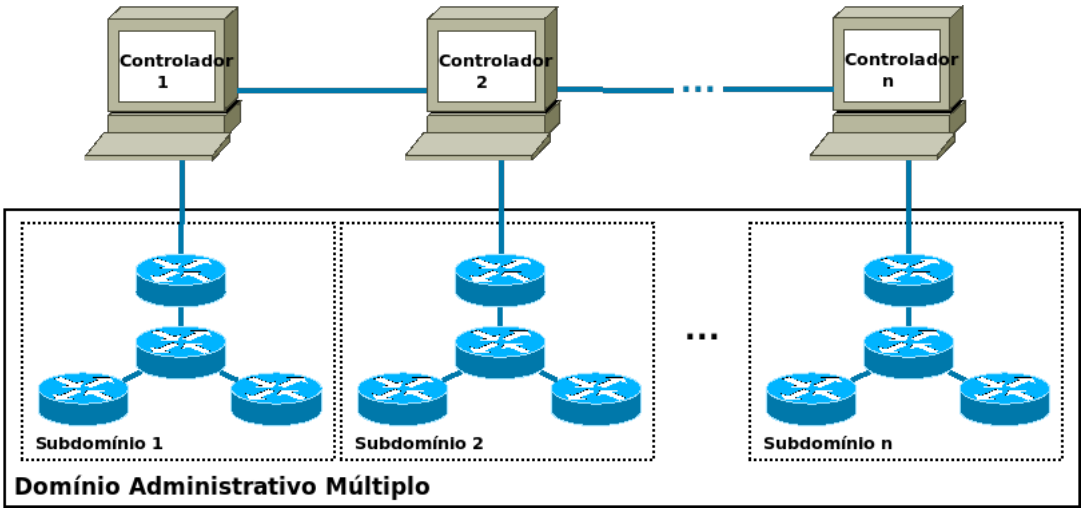
Nesta seção serão discutidos o Problema de Pesquisa que enfrentamos e a Hipótese que levantamos a respeito dele.

1.2.1 Problemática

Apesar de cruciais para o avanço do futuro cenário de redes, as novas tecnologias têm seus serviços chancelados por Organizações de Desenvolvimento de Padronização diferentes (do inglês, *Standard Developing Organization* (SDO)), e em assim sendo elas não contam ainda com arquitetura combinada (NEVES et al., 2016). E como não poderia deixar de acontecer, ainda não foram definidas padronizações próprias de maneira completa, nem os parâmetros de integração dos serviços entre si, o que torna quaisquer estudos (experimentos, arquiteturas, ferramentas, metodologias, etc) de união entre elas de suma importância para sobrevivência e avanço destes esforços. Será necessário, ao trabalhar com mais de um deles simultaneamente no mesmo ambiente, coordenar, sincronizar, balancear, automatizar, entre outros requisitos de conjunto. O que torna possível atender a esses requisitos de conjunto é denominado orquestração, realizado através de um elemento chamado orquestrador.

Frate, Marczuk e Verdi (2019) afirmam que para conseguir escalabilidade, as redes atuais têm sido divididas. Quando isso acontece com as SDN, trabalha-se com múltiplos subdomínios de redes, cada um isoladamente controlado, como vemos na Figura 1.

Figura 1 – SDN de Domínio Administrativo Múltiplo

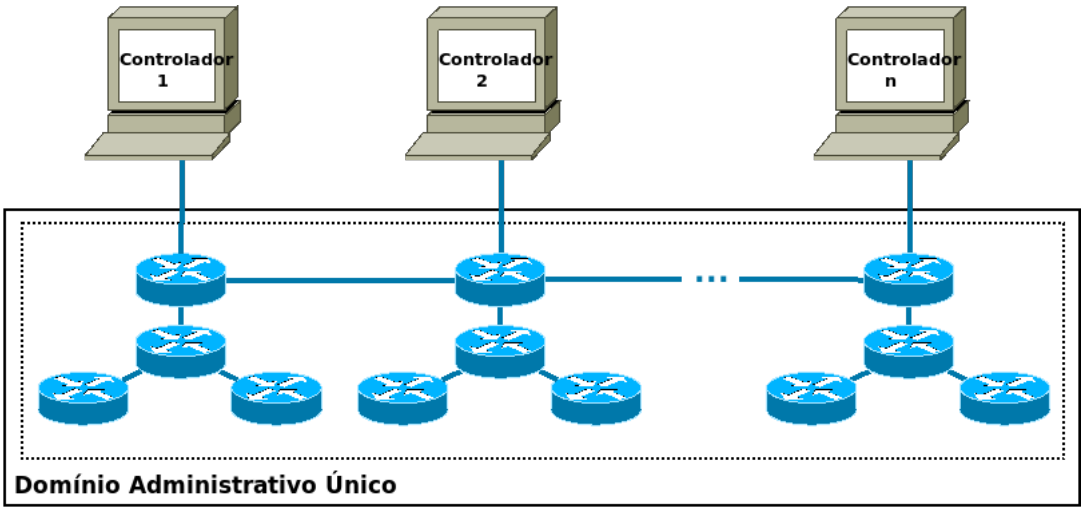


Fonte: Autoria Própria

Na Figura 1 podemos ver que é necessário alocar um controlador para cada divisão, o que exige um interfaceamento por fora dos subdomínios para comunicação entre os controladores, além de tornar cada dispositivo da rede gerenciável por somente um dos controladores por vez.

Porém , se a SDN ganha escalabilidade sem divisão da rede, acontece o que vemos na Figura 2.

Figura 2 – SDN de Domínio Administrativo Único



Fonte: Autoria Própria

Na Figura 2 podemos ver que todos os dispositivos da rede são passíveis de serem controlados por quaisquer controladores , e se trabalha a topologia da rede como um domínio administrativo único (com planos de dados e controle disputado entre vários controladores). Aqui a comunicação entre controladores não demanda conexões além das já existentes entre os dispositivos da própria rede.

Uma vez resolvida a escalabilidade, é criado um outro desafio no sentido de tolerar falha de um controlador: em princípio cada subdomínio está comunicável com o seu respectivo controlador somente. Em caso de falha de um dos controladores, como seus *switchs* e *hosts* seriam realocados para outro controlador disponível? Lyu et al. (2018) lembram que muitas abordagens com múltiplos controladores são estáticas, usando uma associação de mão única entre controlador e elementos do seu subdomínio que permanece inalterada até interferência física na rede.

Um outro ponto em aberto ainda na literatura trata da falta de esclarecimento de como será tratado um mesmo plano de controle gerenciado por controladores diferentes. Caso dois controladores necessitem orientar fluxos coincidentes, como determinar quem ganha a disputa no caso de não haver conciliação possível? Como dito em Frate (2017), alguns controladores são distribuídos, ou seja, suas instâncias iguais são replicadas pela rede e o interfaceamento entre elas é especificado de forma a compartilhar o controle da rede. Na forma distribuída este controle pode ser compartilhado de duas formas:

- Controle Plano: nessa forma a rede é particionada e cada instância controla um subconjunto dela;
- Controle Hierárquico: nessa forma os controladores distribuídos de forma hierárquica, ficando os de nível mais baixo em contato direto com os dispositivos da rede e controlados pelos controladores de nível mais alto. Dessa forma, no topo da hierarquia de controladores é possível ter uma visão global do controle da rede;

Em ambas as formas a comunicação definida entre os controladores distribuídos permite soluções quanto a tolerância a falhas e a problemas de escalabilidade. Isto sem falar que tendo a interface entre suas instâncias iguais por padrão definidas, podemos considerá-los como um caso "básico" de orquestração entre controladores.

Porém, a maioria dos controladores é desenvolvida de forma a centralizar o controle, atuando individualmente para fornecer serviços específicos, como *backup* e segurança por exemplo. Assim sendo, eles são desenvolvidos sem terem como prioridade resolver a questão de viabilizar a coexistência coletiva de controladores heterogêneos, projetados por iniciativas independentes para atuarem sozinhos no plano de controle. Apesar deste controle múltiplo numa mesma SDN ser uma necessidade cada vez mais frequente, a orquestração deste tipo de controladores é o caso realmente mais complexo, que permanece sem solução geral e que é o problema atacado em nosso trabalho.

1.2.2 Hipótese

Através da integração e coordenação dos elementos em comum ou equivalentes entre os controladores SDN de uma mesma rede de domínio administrativo único, é possível construir

uma arquitetura cujo orquestrador utilize os serviços SDN implantados de modo a alcançar pelo menos um dos graus de autonomia da rede, a saber, auto-cura, auto-proteção, auto-otimização ou auto-configuração.

1.3 Objetivos

1.3.1 Objetivo Geral

Implantar Orquestração em *software* para viabilizar a convivência numa SDN entre múltiplos controladores num mesmo plano de controle , cujos serviços operem sobre o mesmo plano de dados, ou seja, num mesmo domínio administrativo único.

1.3.2 Objetivos Específicos

- Promover a orquestração de serviços para autonomia, em ao menos um grau, em SDN de domínio administrativo único com múltiplos controladores;
- construir um *software* orquestrador capaz de lidar com ao menos dois controladores SDN distintos;
- resolver problemas decorrentes da integração entre controladores orquestrados.

1.4 Organização do Documento

No próximo capítulo (Mapeamento Sistemático) é apresentada a técnica de levantamento de dados utilizada na realização deste trabalho. No capítulo 3 (Fundamentação Teórica) são abordados os conceitos que embasam a pesquisa desenvolvida. No capítulo 4 (Arquitetura de Orquestração) é apresentada a solução elaborada para o problema de pesquisa levantado. No capítulo 5 (Análises e Resultados) são explorados os recursos utilizados e experimentos realizados para validação da solução proposta. No capítulo 6 (Conclusão) são apontadas as considerações finais, principais contribuições, publicação submetida e possibilidades de continuação para a pesquisa aqui relatada.

2
Mapeamento Sistemático

Neste capítulo é apresentado o mapeamento sistemático sobre os estudos que se debruçaram sobre o tema de pesquisa Orquestração. O processo de mapeamento sistemático da literatura foi escolhido por ser mais adequado ao rigor acadêmico e científico deste trabalho do que uma simples revisão bibliográfica. Ao invés de uma busca geralmente não estruturada por trabalhos relacionados, que seria o caso da revisão bibliográfica, foi seguido o protocolo do mapeamento sistemático. Nele, uma busca bem definida é realizada, de forma que outros pesquisadores, ao trabalharem com o mesmo tema de pesquisa e seguindo este mesmo processo, devem ser capazes de encontrar os mesmos resultados.

2.1 Introdução

O Mapeamento Sistemático da Literatura consiste no processo onde um protocolo de passos objetivos de pesquisa resulta no mesmo conjunto de referências obtidas independente de quem seja o pesquisador que o execute. São fases deste processo:

- Planejamento: fase na qual são formuladas as questões de pesquisa, identificados e avaliados criticamente os estudos;
- Execução: onde são selecionados os trabalhos para estudos primários e deles extraídos seus dados;
- Análise: onde os resultados são sintetizados e interpretados;

2.2 Planejamento

2.2.1 Questões de Pesquisa

Para se alcançar o tema dessa pesquisa, num escopo viável, foi necessário uma definição e refinamento do mesmo. Ao deparar com o termo Orquestração, percebeu-se que é usado como "coringa" nas mais diferentes tecnologias de modo a denotar algum caráter autônomo (livre de interferência humana), seja no gerenciamento, na otimização, na recuperação, na configuração, e em outros aspectos de recursos de redes. Assim sendo, dada relevância da tecnologia base para a Orquestração, disponibilidade de materiais, referências e recursos, optou-se por restringir nos estudos de Orquestração baseados em controladores SDN. E para aprofundamento deste tema foram definidas as questões da Tabela 1.

Tabela 1 – Questões de pesquisa

Código da Questão	**Questão de Pesquisa**
QP1	Quais controladores SDN foram trabalhados nas orquestrações estudadas?
QP2	Quais elementos dos controladores foram trabalhados que fizeram possível a orquestração?
QP3	Foi definida alguma arquitetura ou modelo para aplicar a orquestração?

2.2.2 Estratégias de Buscas

Com as questões de pesquisa definidas e contatos iniciais com trabalhos sobre o tema, foram identificados termos sinônimos e palavras-chaves relevantes, e definidas as *strings* de busca a serem usadas nas bases de busca. Dada a relevância da língua inglesa para a área de redes de computadores, e da utilização de bases de busca nacionais (que abarcam trabalhos em Português e em Inglês) foram preparadas as *strings* da Tabela 2 para incluir tanto bases brasileiras quanto bases estrangeiras.

Tabela 2 – Strings de busca

Idioma(s)	***string***
Inglês	(Orchestration AND ("sdn"OR "software-defined network"OR "software defined network"OR "software-defined networking"OR "software defined networking"OR "software-defined networks"OR "software defined networks"))
Inglês e Português	((Orchestration OR Orquestração) AND ("sdn"OR "software-defined network"OR "software defined network"OR "software-defined networking"OR "software defined networking"OR "software-defined networks"OR "software defined networks"OR "redes definidas por software"))

2.2.3 Bases de Busca

Foram selecionadas bases importantes (Tabela 3) para a área da computação de modo geral, de modo a garantir a obtenção de trabalhos que sejam atuais, bem referenciados e relevantes para o tema pesquisado. Todas as bases forneciam portais *web* através dos quais as buscas

aplicaram as *strings* anteriormente definidas. Por questões de limitações particulares a cada um, refinamentos diferentes foram necessários na busca, mas de modo unânime não houve restrição de data dado que não influenciaria significativamente nos resultados deste tema tão recente.

Tabela 3 – Bases de busca

Fonte de Busca	**URL**
ACM Digital Library	https://dl.acm.org/
BDBComp	http://www.lbd.dcc.ufmg.br/bdbcomp/
CTD CAPES	http://catalogodeteses.capes.gov.br/catalogo-teses/#!/
Elsevier	http://www.sciencedirect.com/
IEEE Xplorer	http://ieeexplore.ieee.org
Periódicos CAPES	http://www-periodicos-capes-gov-br.ez20.periodicos.capes.gov.br/
Scopus	https://www.scopus.com/
Springer Link	https://link.springer.com/

2.2.4 Critérios de Inclusão e Exclusão

Os critérios definidos nas Tabelas 4 e 5 serviram para delimitação dos resultados obtidos classificados para aprofundamento nas etapas posteriores, além de determinarem um processo confiável e replicável de obtenção de estudos primários que atenderam ao escopo permitido dentro da pesquisa realizada.

Tabela 4 – Critérios de Inclusão

Código do Critério	**Critério de Inclusão**
CI1	Está completo e disponível na WEB
CI2	Publicado em algum veículo (*journal*, simpósio, conferência) da base citada
CI3	Apresenta prática em algum nível de orquestração (protótipo, arquitetura, controlador, experimento, modelo, etc.)

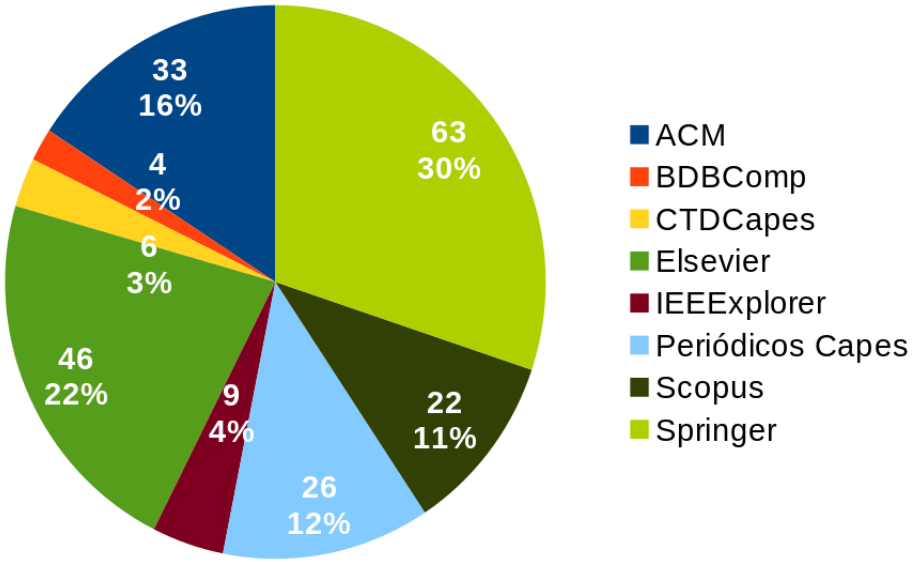
Tabela 5 – Critérios de Exclusão

Código do Critério	**Critério de Exclusão**
CE1	Duplicado
CE2	Tema ou foco do estudo não é orquestração via SDN
CE3	Trabalho resumido e *survey*

2.3 Execução

Esta fase focou na seleção dos estudos primários e extração de dados, foram em seguida apresentados os resultados obtidos referentes às bases de buscas e à filtragem dos critérios de inclusão e exclusão. Na Figura 3, percebem-se as quantidades de trabalhos encontrados nas buscas das bases, assim como sua respectiva porcentagem em relação ao total (ACM retornou 33 trabalhos (16%), BDBComp retornou 4 trabalhos (2%), CDTCapes retornou 6 trabalhos (3%), Elsevier retornou 46 trabalhos (22%), IEEExplorer retornou 9 trabalhos (4%), Periódicos Capes retornou 26 trabalhos (12%), Scopus retornou 22 trabalhos (11%) e Springer retornou 63 trabalhos (30%)).

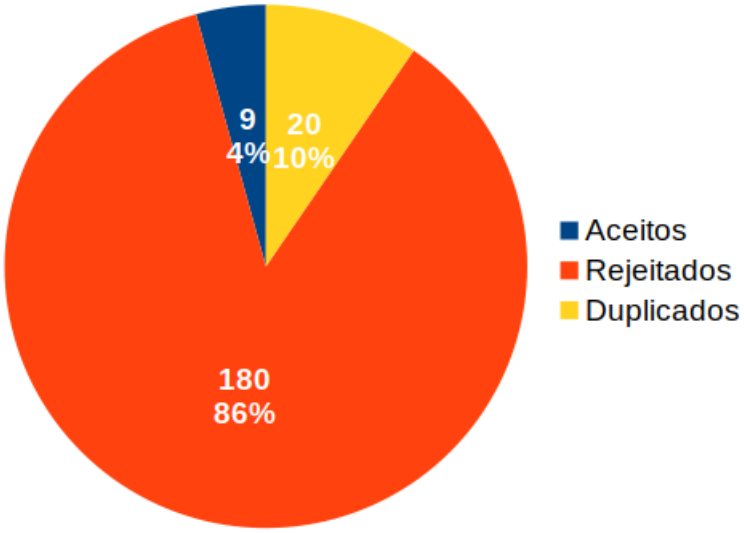
Figura 3 – Trabalhos encontrados por base de busca



Fonte: elaboração própria

Na Figura 4, é apresentada a quantidade final de estudos que passaram por todas as etapas de seleção (Figura 5) e deles foram extraídos os dados para fase de Análise.

Figura 4 – Trabalhos aceitos, rejeitados e duplicados

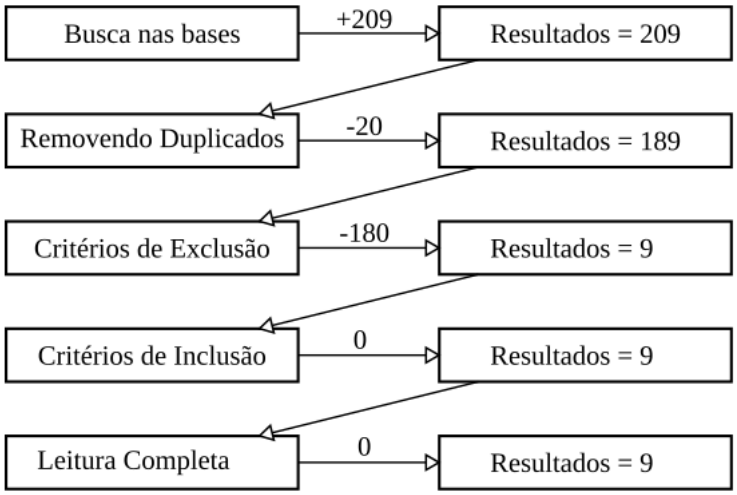


Fonte: elaboração própria

2.4 Análise

A Tabela 6 sintetiza os estudos primários, ou seja, aqueles que passaram por todas as etapas anteriores (de inclusão e exclusão) e que melhor responderam às questões de pesquisa.

Figura 5 – Etapas e seleção dos artigos



Fonte: elaboração própria

Tabela 6 – Estudos primários selecionados

#	Ano	Título	Referência	Fonte
EP01	2015	Control and orchestration of multidomain optical networks with gmpls as inter-sdn controller communication [invited].	(CASELLAS et al., 2015)	Periódicos CAPES
EP02	2016	Control exchange points: Providing qos-enabled end-to-end services via sdn-based inter-domain routing orchestration.	(KOTRONIS et al., 2016)	Periódicos CAPES
EP03	2016	The SELFNET approach for autonomic management in an NFV/SDN Networking Paradigm.	(NEVES et al., 2016)	Scopus
EP04	2017	Automatic monitoring management for 5g mobile networks.	(CELDRáN et al., 2017)	Elsevier
EP05	2017	Sdn orchestration architectures and their integration with cloud computing applications.	(MAYORAL et al., 2017)	Elsevier
EP06	2017	Software defined network (sdn) based internet of things (iot): A road ahead.	(TAYYABA et al., 2017)	ACM Digital Library
EP07	2018	A novel self-organizing network solution towards crypto-ransomware mitigation.	(MONGE; VIDAL; VILLALBA, 2018)	ACM Digital Library
EP08	2018	Multi-timescale decentralized online orchestration of software-defined networks.	(LYU et al., 2018)	IEEE Xplore
EP09	2019	Orchflow: An architecture for orchestration of multiple controllers in openflow networks.	(FRATE; MARCZUK; VERDI, 2019)	Springer Link

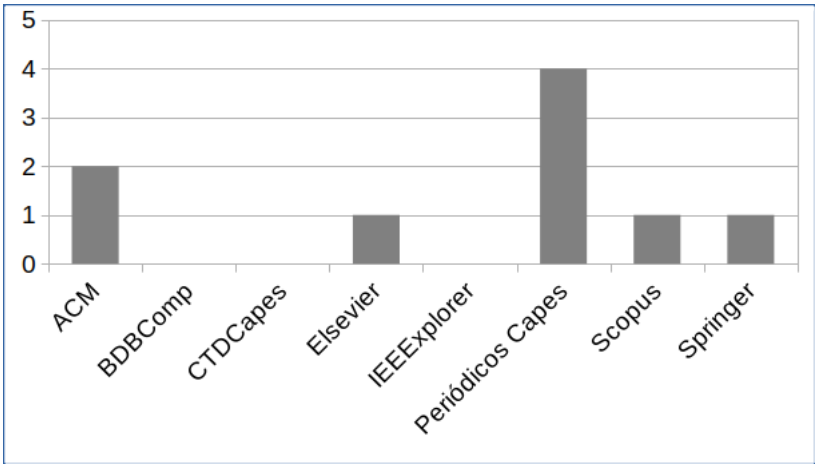
2.5 Resultados da Análise

2.5.1 Bases dos Trabalhos

Para fase de extração de dados inicialmente foi levantada a contribuição de cada base de busca para os estudos primários selecionados. Como podemos ver na Figura 6, a maioria foi obtida no Periódicos Capes (com 4 trabalhos), seguida pela ACM (com 2 trabalhos) e finalizando

com Elsevier, Scopus e Springer (1 trabalho cada). Evidenciamos a importância de incluir bases nacionais na pesquisa pois, como será visto posteriormente, foram delas que vieram a maioria dos trabalhos relacionados com maior completude de respostas às questões de pesquisa listadas na Tabela 1.

Figura 6 – Quantidade de trabalhos selecionados por base de busca



Fonte: elaboração própria

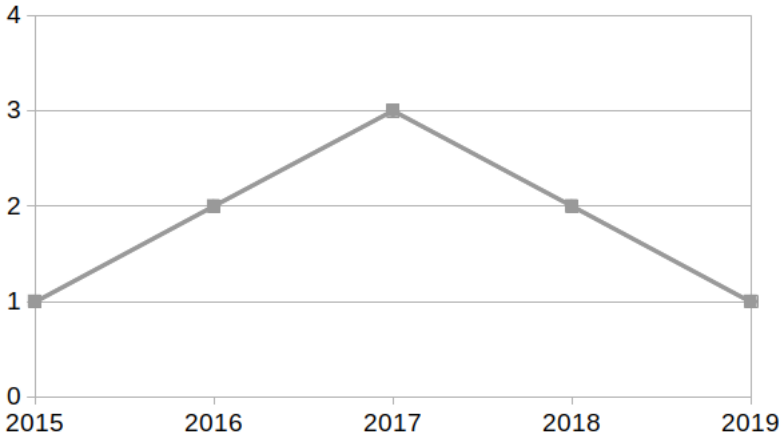
2.5.2 Anos das Publicações

Continuamos analisando os estudos primários selecionados, agora sob a ótica do ano de publicação dos trabalhos. Acreditamos que, apesar de bastante em voga no momento, o viés de Orquestração que estamos pesquisando (via controladores SDN), foi foco de trabalhos junto com outras tecnologias, assim resultando em resultados "poucos"se olharmos isoladamente, mas no contexto geral de Orquestração, um somatório que revela interesse crescente. Segundo mostrado no gráfico da Figura 7, o trabalho mais antigo encontrado foi em 2015 (1 trabalho), seguido por publicações em todos os anos seguintes até o atual (2 em 2016, 3 em 2017, 2 em 2018 e 1 em 2019), o que entendemos demonstrar um interesse oscilante (sobre a abordagem que estamos pesquisando), porém constante, mediante não haver nenhum ano no qual pelo menos 1 trabalho publicado envolvendo diretamente Orquestração e SDN juntos.

2.5.3 Países das Publicações

Como observado na Figura 8, não foi surpresa deparar com uma maioria de publicações tendo como país constantemente representante na área de tecnologia os Estados Unidos (4 trabalhos). Todo restante foi publicado uniformemente em veículos que abrangeram o continente americano (Canadá, 1 trabalho) e europeu (Bélgica, Holanda, Reino Unido e Alemanha, 1 trabalho cada), valendo destacar que não foi possível concluir uma predominância ocidental de interesse nesta pesquisa. Esta inferência se torna fraca ao se observar nos trabalhos selecionados

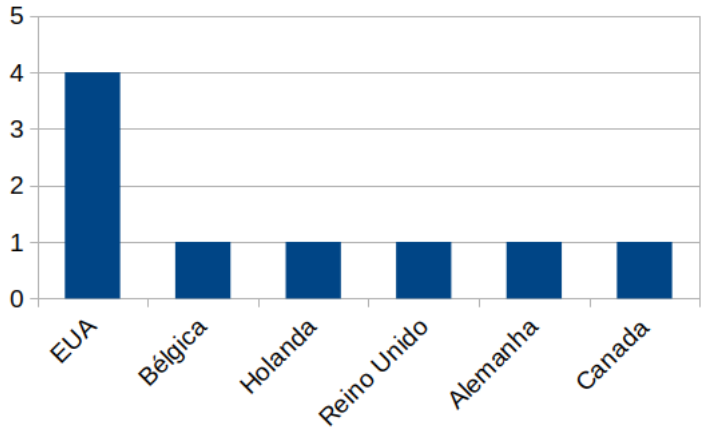
Figura 7 – Quantidade de trabalhos selecionados por ano de publicação



Fonte: elaboração própria

uma grande maioria de trabalhos conjuntos de pesquisadores cujas nacionalidades (ou vínculos) incluem tanto os países de origem da publicação do estudo quanto outras em que foram coautores completam os esforços entre países de pesquisa acadêmica.

Figura 8 – Quantidade de trabalhos selecionados por país



Fonte: elaboração própria

2.6 Respostas às Questões de Pesquisa

Sendo determinantes para nossa pesquisa, as respostas que conseguimos dos estudos selecionados sobre as questões de pesquisa tentaram levantar o estado da arte sobre os trabalhos publicados envolvendo diretamente as tecnologias de Orquestração e SDN. Como demonstrado na Tabela 7, acreditamos que o ineditismo do viés que seguimos, e também pelas lacunas naturalmente existentes em se tratando de tecnologias ainda em desenvolvimento, foram necessários

adotar critérios de classificação das repostas de modo a contemplar a completude com a qual as questões de pesquisa foram satisfeitas.

Tabela 7 – Questões de pesquisa respondidas por trabalho

#	QP1	QP2	QP3
EP01	P	T	T
EP02	P	P	N
EP03	P	P	T
EP04	N	N	T
EP05	T	T	T
EP06	N	N	P
EP07	N	N	T
EP08	N	N	T
EP09	T	T	T

Completude de respostas: totalmente(T), parcialmente(P) ou, não respondida(N)

2.6.1 Questão de Pesquisa QP1

Esta questão de pesquisa 1 busca responder, dentre os controladores SDN existentes atualmente, quais os que foram usados para implementar orquestração. Segue abaixo a lista por estudo primário dos controladores SDN encontrados:

- (MAYORAL et al., 2017) : OpenDaylight;
- (FRATE; MARCZUK; VERDI, 2019) : Ryu e Floodlight;

Os outros trabalhos que responderam parcialmente (P) a esta questão permitem inferir de forma indireta (porém não conclusiva) quais os controladores orquestrados, através das linguagens e protocolos citados, dentre outras pistas.

2.6.2 Questão de Pesquisa QP2

Esta questão de pesquisa 2 busca responder, dentre os controladores SDN trabalhados nos estudos selecionados, quais ferramentas foram utilizados neles para tornar possível a orquestração. Segue abaixo a lista por estudo primário das ferramentas SDN dos controladores encontradas:

- (CASELLAS et al., 2015) : Interfaces E/W e protocolo OSPF-TE;
- (MAYORAL et al., 2017) protocolo REST;
- (FRATE; MARCZUK; VERDI, 2019) : componente OLA e *midleware* OrchFlow;

Do mesmo modo que na questão anterior, são citadas ferramentas potenciais (protocolos de rede, componentes de *software*, para implementação de orquestração. Porém, não é possível ter a certeza do que realmente foi utilizado para este fim.

2.6.3 Questão de Pesquisa QP3

Esta questão de pesquisa 3 busca responder, agora sob a ótica de desenvolvimento de solução, quais artefatos teóricos (nos quais focamos em modelos e arquiteturas) deram embasamento para a construção do produto de *software* e ou das conclusões obtidas. Segue abaixo a lista por estudo primário dos artefatos teóricos encontrados:

- (CASELLAS et al., 2015): Arquitetura de Controle e Orquestração Híbrida ;
- (NEVES et al., 2016): Arquitetura SELFNET;
- (CELDRáN et al., 2017): Componente Orquestrador dentro da Arquitetura de Gerenciamento dos Recursos da Rede ;
- (MAYORAL et al., 2017) : Arquiteturas SC-Arch e ABNO;
- (MONGE; VIDAL; VILLALBA, 2018) : *Framework* para mitigação de *Crypto-ransomware* com camada de Orquestração de Controladores SDN;
- (LYU et al., 2018): Modelo matemático e algoritmo para otimização de controladores SDN distribuídos;
- (FRATE; MARCZUK; VERDI, 2019) : Arquitetura OrchFlow;

Para esta questão específica, o que encontramos em Tayyaba et al. (2017) são trabalhos onde a orquestração é encontrada em implementações de IoT baseadas em soluções de SDN. Com o conflito indireto (mas que não consideramos completo) que esse fato causou com o Critério de Exclusão CE2, entendemos que a resposta foi parcialmente atendida.

3
Fundamentação Teórica

A seguir, serão abordados as principais teorias e conhecimentos que deram embasamento ao trabalho aqui realizado, necessários para o entendimento tanto da Orquestração propriamente dita quanto dos serviços SDN que serão integrados pelo orquestrador. Também são apresentados os trabalhos relacionados ao que foi aqui desenvolvido e finalizado com um comparativo que explicita nossa contribuição para a pesquisa deste tema em relação aos trabalhos encontrados.

3.1 Definição de Orquestração

Percebeu-se que na atual literatura de redes, o termo “orquestração” tem sido usado quando se trabalha com diferentes tecnologias e com diferentes abordagens sem ter sido completamente definido por nenhuma das iniciativas (grupo de pesquisa, consórcio de instituições, fundações, etc) que dele fazem uso. Apesar de se identificarem pontos em comum nos estudos que envolvem este tema, cada um o define à sua maneira de modo a contextualizar o trabalho desenvolvido. Para entendimento do que iremos encontrar ao longo desta dissertação, evidenciamos algumas das definições que mais se relacionam com o que pretendemos fazer para por fim gerar uma versão nossa que dê lastro à nossa proposta.

Segundo Casellas et al. (2015), orquestração sucintamente definida, é o controle coordenado de sistemas heterogêneos normalmente envolvendo múltiplas interfaces e seus fluxos de redes, integrados de forma distribuída ou centralizada.

Em Rotsos et al. (2017b) encontramos que, para dar vazão aos novos paradigmas de redes em desenvolvimento, é necessário um novo sistema de controle e gerenciamento, capaz de orquestrar as diferentes tecnologias e recursos disponíveis nas infraestruturas de redes modernas. Estes sistemas são responsáveis por convergir a heterogeneidade de controle e gerenciamento entre as tecnologias, num esforço de sintetizar as interfaces orientadas a serviços, e permitir a automação na implantação e entrega destes. O esforço de desenvolver ferramentas para orquestrar

tem acelerado, mas, como normalmente cada fabricante desenvolve protocolos e mecanismos proprietários, a integração permanece desafio em aberto.

Já Mayoral et al. (2017) define como orquestração de redes a coordenação e automação do estabelecimento e entrega de múltiplas conexões entre redes independentes para o provisionamento de serviços de conectividade E2E (*end-to-end*) através de domínios de redes heterogêneos (que podem inclusive ser compostos por diferentes tecnologias).

Segundo Neves et al. (2016), a orquestração de serviços de monitoramento é tarefa essencial para conduzir à auto-configuração, auto-cura, e auto-otimização dos processos em redes auto-organizáveis (do inglês, *Self-Organizing* Networks (SONs)). É afirmado que as futuras redes móveis devem orquestrar os serviços de monitoramento de rede considerando não somente as informações relativas aos dados contidas nos fluxos, mas também as relativas ao controle.

Integração, apesar de ser basilar para orquestração, parece não ter sido foco durante o desenvolvimento das inovações em redes de computadores, como podemos observar em Frate, Marczuk e Verdi (2019) ao enfatizar que em SDN o protocolo *OpenFlow* não especifica como a comunicação entre controladores deverá ser realizada, permanecendo este um desafio para estudo, e que será abordado nesta pesquisa.

Diante do que foi exposto, aqui destacamos as seguintes ideias como denominador comum para chegarmos à nossa própria definição:

- auto-cura;
- auto-proteção;
- auto-otimização;
- auto-configuração;
- integração coordenada de sistemas diferentes que operem sobre os mesmos elementos (ou num mesmo ambiente compartilhado).

Assim sendo, tomemos como nossa definição de orquestração o esforço teórico e prático em computação, através de modelagem e ferramentas, de integração de sistemas de redes que alcance automação(cura, proteção, otimização e configuração). Isso é obtido através do somatório coordenado das novas tecnologias, superando suas implementações em diferentes ambientes proprietários e a falta de padronização entre elas, de modo a otimizar o tempo e esforço dos administradores de redes.

A seguir, como prometido na sessão 1.1 do capítulo 1, vamos nos aprofundar na tecnologia de maior relevância para nosso trabalho.

3.2 SDN

Diante dos desafios das redes atuais a *Open Network Foundation* (ONF) apresentou o paradigma de Redes Definidas por Software (do inglês, *Software Defined Networks*).

No atual cenário com dispositivos de redes diferentes operados por protocolos proprietários, todo gerenciamento e monitoramento da infraestrutura fica refém de uma topologia configurada manualmente em linguagens de diferentes padrões fechados, o que torna a implantação de políticas e solução de problemas tarefas altamente custosas e de baixo grau de abstração por parte dos administradores da rede. Toda lógica de operação é particionada e distribuída por inúmeros *switchs* que individualmente processarão parte do funcionamento abstrato da rede, o que descentraliza sua visão geral, sobrando ao analista manter atualizada a documentação que reflita abstratamente a operacionalização concreta dos fluxos de dados.

Com intuito de centralizar a operação da rede num ponto único, com interface uniformizada de ativos da rede que independam de fabricante, e tornem o trabalho do gestor mais otimizado, SDN separa em dois planos o que antes era misturado numa mesma infraestrutura:

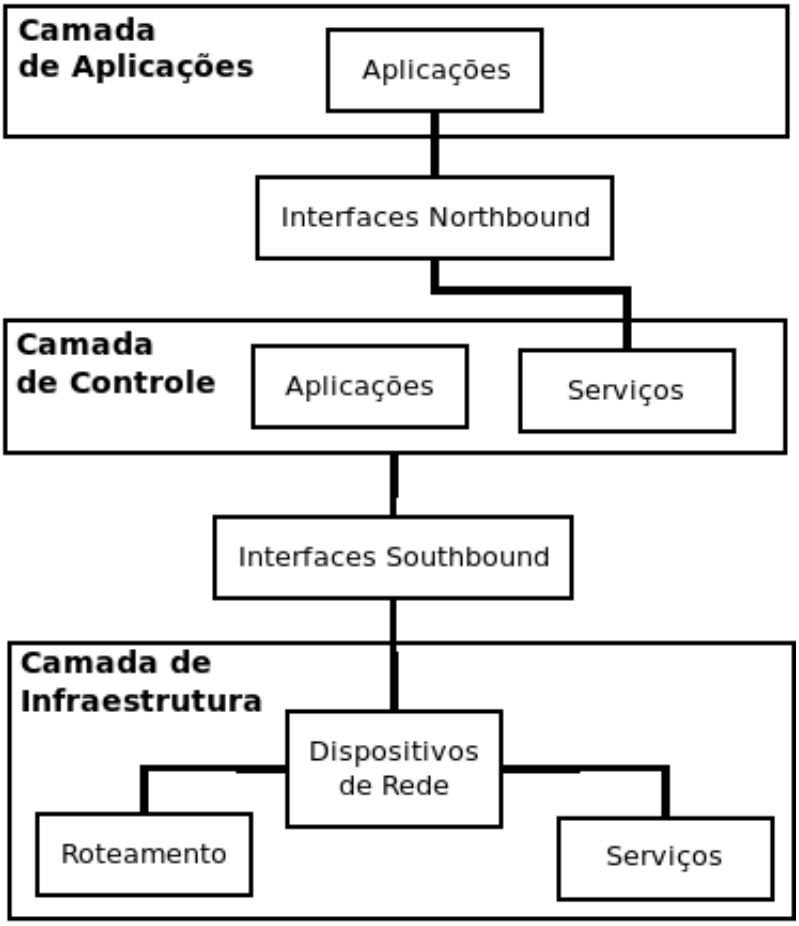
- Plano de Controle: toda rede estará aqui representada de forma abstrata e padronizada, livre das limitações impostas por configurações enrijecidas, onde o que será foco será o mapa lógico da rede, cujas mudanças de topologia e políticas serão realizadas via *software* mediante as demandas das aplicações e serviços dos usuários, que não terão que lidar (ou até estarem inviabilizadas) por um mar de diferentes protocolos e arquitetura descentralizada;
- Plano de Dados: aqui encontram-se os elementos de rede que somente executarão o roteamento de pacotes segundo o que for definido pelo plano de controle, sendo aqui a intenção jogar para um nível mais alto (com maior grau de inteligência e abstração) todo gerenciamento e operação que ficariam presas às limitações de alcance de visão e de processamento que existiam até então;

3.2.1 Arquitetura

Resumidamente, SDN's são redes cuja arquitetura permite a programação centralizada de sua topologia e de seus elementos, de modo a separar nos planos de dados e de controle as funções de rotear e processar os eventos de rede de maneira abstrata. A Figura 9 apresenta esta arquitetura, cujos elementos são:

- Na Camada de Aplicação residem os *softwares* dos usuários da rede (aplicações), cuja interação com tais elementos desta estava impedida ou limitada no cenário anterior a SDN já descrito, mas que agora por meio de interfaces chamadas *Northbound Interfaces* (NBI)

Figura 9 – Arquitetura SDN



Fonte: (AKHUNZADA et al., 2015)

podem implementar APIs (*Applications Programming Interfaces*) que operem uma visão abstrata e centralizada da rede existente e gerenciada pela camada seguinte.

- Na Camada de Controle é trabalhado o plano de controle, que centraliza toda topologia e fluxo de dados da rede independentemente de configuração manual em equipamento de protocolo fechado e que mantém atualizada toda visão da rede, pois interage via *software* diretamente com a infraestrutura física, ficando livre de falhas de documentação de topologia e configuração de políticas.

- A última é a Camada de Infraestrutura, que é a rede propriamente dita, formada pelos dispositivos que a partir deste paradigma ficarão exclusivamente dedicados às funções elementares de transmissão de dados. Para tal, o roteamento de pacotes e gerenciamento de fluxos será informado pela Camada de Controle por interfaces conhecidas como *Southbound Interfaces* (SBI), que implementarão protocolos de padrão aberto (segundo orientação da ONF) para que alcancemos cada vez mais uma padronização de ferramentas, permitindo uma maior independência em relação a fornecedores de *hardware* de dispositivos de rede.

3.2.1.1 Controlador

O *software* responsável por tornar programável a rede, permitindo a automação de políticas, o gerenciamento de serviços e intermediando a comunicação entre aplicações de alto nível de abstração com dispositivos de redes com baixo grau de inteligência é chamado de Controlador. Em sendo um programa, é possível definir aplicações de redes no controlador que são codificados via linguagens de programação.

É possível encontrar controladores que trabalhem de forma distribuída ou centralizada, oferecendo recursos de backup e segurança. Na literatura pesquisada a rede controlada também é referenciada como domínio administrativo, e pode estar subdividida (delegando um controlador para cada divisão) ou não. Este último caso chamamos de domínio administrativo único, e pode ter 1 ou mais controladores. Com 1 controlador atende-se ao padrão (1 controlador para 1 domínio administrativo único) , mas existindo mais de 1 (vários controladores para 1 domínio administrativo único) ainda não há solução estabelecida, sendo o desafio em aberto explorado por este trabalho.

3.2.1.2 OpenFlow

Também proposto pela ONF, *OpenFlow* consiste em um protocolo de padrão aberto que permite a comunicação entre controladores e equipamentos de redes. Por parte do plano de controle este suporte é amplamente oferecido devido à incorporação natural de componentes de linguagens de programação, utilizadas para codificar os *softwares* controladores. Já por parte do plano de dados, este suporte tem sido cada vez mais oferecido por parte dos fabricantes de dispositivos de redes (citamos aqui Arista Networks, Big Switch Networks, Cisco, Extreme Networks, HP, IBM, Infinera, Juniper e NEC), de modo a tornar a implantação de SDN nas redes atuais um processso gradual e de manutenção do legado existente previamente nas infraestruturas já em produção.

3.2.1.3 Interfaces E/W

Não é definida junto com o paradigma pela ONF, mas é recorrente na literatura a noção de Interfaces *East* / *West* dentro de SDN. Do mesmo modo que a arquitetura contempla as interfaces NBI entre Camada de Aplicações e Camada de Controle e as interfaces SBI entre a Camada de Controle e Camada de Infraestrutura, quando é necessário se trabalhar com mais de um controlador, surge a necessidade de definir-se como será a interface deles entre si, dentro do plano de controle SDN. Como não há padronização definida, cada estudo tem implementado este interfaceamento de maneira diferente, por meio de protocolos variados que podem incluir os utilizados para NBI, SBI ou ainda outros protocolos de redes adaptados à esta necessidade.

3.2.2 Desafios

Os benefícios das redes SDN ainda contrastam com diversos desafios ocasionados pela sua própria arquitetura. Jammal et al. (2014) em seu trabalho cita estes desafios, e para vários deles levantamos como orquestração tem uma influência significativa:

- Confiabilidade: Em redes convencionais, quando há falha de algum dos dispositivos da rede o fluxo é desviado para outros dispositivos de forma a manter o fluxo. Entretanto, nas redes SDN, a falha no controle lógico se torna um grande problema para o quesito confiabilidade, principalmente se esse controle está de fato centralizado em somente um controlador. E mesmo que na literatura existam trabalhos com SDN's de mais de um controlador, a falta de chancela da ONF para estes cenários torna a orquestração ainda mais crucial, já que não se tem definido como um controlador pode perceber a falha de outro e assumir o controle da parte da rede atingida por esta falha.

- Escalabilidade: O desacoplamento dos planos de dados e de controle gera uma característica de interdependência na evolução das partes, contudo, também gera problemas na escalabilidade da rede. À medida que novos dispositivos físicos são inseridos e novos fluxos de rede são criados, o controlador pode passar a receber muito mais requisições do que pode processar, causando um "gargalo"na rede. Como no desafio anterior, também não está definido pela SDO desta tecnologia meios de realizar o balanceamento de fluxos entre controladores.

- Performance: A performance de uma rede SDN é medida por duas métricas: números de fluxos por segundo e tempo de configuração de fluxo. Para o caso do tempo de configuração de fluxo ainda existem os modos proativo e reativo, sendo que o último é normalmente utilizado pelo controlador SDN. Resumidamente, no modo reativo o tempo de configuração de fluxo é medido a partir do momento que o pacote chega no *switch* e não há regras na tabela que especifiquem a ação sobre o mesmo. Dessa forma, o controlador toma alguma decisão sobre o pacote independente da tabela de fluxo. O tempo que o controlador leva para processar o pacote, tomar uma decisão e atualizar a tabela pode levar a problemas relacionados à latência e vazão. O intermédio de um orquestrador poderia ajudar no sentido de favorecer a carga de fluxos para controladores de maior performance, de modo a favorecer o tempo de resposta médio no plano de dados e otimizar a atuação entre os serviços SDN no plano de controle a depender da demanda do serviço mais requisitado no momento.

- Posicionamento do controlador: A questão sobre a localização do controlador influencia em cada aspecto do plano de controle, seja nas latências dos fluxos até confiabilidade da rede e performance. O problema se resume em como posicionar um dado número de controladores numa certa rede física tal que suas funcionalidades sejam otimizadas

para um objetivo específico. A adição do orquestrador na rede pode ajudar no sentido de tornar este problema indiferente, diante da garantia do provimento dos serviços destes controladores.

- Limitações da CPU: As limitações da CPU no *switch* afetam a banda entre este e o controlador. Esse quesito também afeta diretamente a questão da escalabilidade.

- Uso de interfaces de baixo nível entre controlador e dispositivos de rede: Apesar de SDN simplificar o gerenciamento de redes através das interfaces simplificadas para determinar políticas de rede de alto nível, o *framework* das camadas inferiores precisam traduzir essas políticas para configurações de baixo nível no *switch*. Tal característica exige que os desenvolvedores implementem métodos que coordenem múltiplos eventos assíncronos até para tarefas simples, além do conhecimento detalhado que se deve ter do hardware e do módulo de software.

- Segurança: A separação do plano de dados e controle tornou o gerenciamento de redes mais simplificado e dinâmico, contudo a segurança não é um quesito embutido nas características da arquitetura, pois o ato de concentrar a lógica da rede em um *software* controlador, torna a rede suscetível a ataque de hackers, podendo comprometer toda a rede. Além disso, as tecnologias atuais de segurança foram pensadas especificamente para redes convencionais e justamente por ser um paradigma emergente, as redes SDN ainda não estão totalmente padronizadas, gerando ainda mais desafios para o quesito segurança. E quando se trata de um orquestrador numa SDN de vários controladores, centralizando todo o plano de controle num elemento único central, os potenciais perigos são ainda maiores.

Os serviços SDN usados nos experimentos desta dissertação operam sobre a área de segurança de redes. Assim sendo, para entendimento mais completo dos mesmos, a seguir nos aprofundaremos nos temas de segurança de redes que baseiam teoricamente os seus funcionamentos.

3.3 Segurança de Rede

Gerenciar redes de computadores em um cenário com tantos usuários, grande volume de dados e diversidade tecnológica gerou novas vulnerabilidades nos sistemas, tornando-se um desafio para os pesquisadores da segurança de informação, independente da arquitetura implantada. Com referência no padrão ITU (2016) para serviços de segurança de rede e de acordo com Kurose e Ross (2010), uma comunicação para ser segura deve possuir as seguintes propriedades:

- Confidencialidade: Indica que somente o remetente e o destinatário devem ter conhecimento do conteúdo da mensagem transmitida.

- Autenticação de ponto final: A comunicação deve ocorrer de forma que a identidade do remetente ou destinatário possa ser confirmada ou autenticada pela outra parte envolvida na comunicação.
- Integridade de mensagem: Deve-se garantir que a mensagem seja entregue sem alterações externas, mantendo a integridade da mesma até que o remetente possa ler a mensagem.
- Segurança Operacional: Redes com acesso à Internet pública, principalmente redes corporativas, tornam-se vulneráveis para atacantes que a utilizam como forma de acesso à rede privada. Para este caso é importante que exista um sistema de detecção de atividade suspeita, sendo crucial para a segurança da informação.

Por sua definição uma ameaça é uma potencial violação da segurança, que ocorre quando existe uma circunstância, ação ou evento que possa gerar dano devido a uma vulnerabilidade, sendo intencional ou não (SHIREY, 2000).

3.3.1 Segurança em SDN

Com o desenvolvimento das redes SDN e das tecnologias envolvidas, o estudo da segurança nessas redes tornou-se uma das principais preocupações, principalmente quanto a sua aplicação em grandes infraestruturas de rede. Isso se deu porque o paradigma SDN se foca nas características operacionais da separação da camada de controle e de dados, deixando o quesito segurança como uma preocupação externa. Os desafios enfrentados quando se trata de SDN é que suas vulnerabilidades se encontram justamente nas suas principais características: programação de rede por *software* e centralização do controle lógico. Kreutz et al. (2014) descreve em sua pesquisa sete ameaças potenciais nas redes SDN:

1. Ataques por vulnerabilidades nos *switches* OpenFlow: No nível do plano de dados, esse tipo de ataque a um *switch* OpenFlow pode causar lentidão no fluxo, perda de pacotes, clonagem ou desvio de tráfego, além de ser possível injetar tráfego ou requisições falsas para sobrecarregar o controlador e *switches* vizinhos.
2. Fluxos de tráfego falsos: O atacante pode utilizar elementos físicos da rede para gerar ataques de DoS (do inglês, *Denial Of Service*) contra os *switches* OpenFlow e contra os controladores.
3. Ataques através do canal de comunicação do plano de controle e plano de dados: Esse ataque é feito direto no canal de comunicação entre o *switches* e controladores para gerar ataques DoS ou para roubo de dados. A falha nesse caso se encontra no protocolo TLS/SSL, que por si só não garante confiabilidade no canal.
4. Ataques por vulnerabilidades no controlador: Possíveis ataques por meio do controlador podem comprometer toda a rede justamente devido à centralização lógica do controle.

Utilizar um mecanismo de detecção nesse caso pode não funcionar porque o próprio controlador pode mascarar seu comportamento por não existir um padrão fixo de eventos que o torne suspeito.

5. Ausência de mecanismos de confiabilidade de comunicação entre controladores e aplicações: Análoga à ameaça 3, a preocupação se encontra na segurança do canal entre o controlador e as aplicações.

6. Ataques por vulnerabilidades nos *hosts*: A falha em um *host* pode levar o atacante a acessar o controlador. Nas redes convencionais o princípio de tomar o controle é o mesmo, porém em uma rede SDN a visão global torna esse tipo de ataque ainda mais perigoso.

7. Falta de recursos confiáveis para análise forense e remediação dos problemas: A detecção e remediação de uma ameaça não são garantidos se há falta de informações confiáveis dos elementos da rede.

Segundo Shirey (2000), um serviço de segurança é um processo ou serviço de comunicação que fornece uma proteção específica para os recursos de uma rede. Entre esses serviços podemos citar serviços de autenticação, disponibilidade, auditoria, anonimização, entre outros. Para o cenário da arquitetura SDN, considerando as ameaças potenciais citadas e as comparando com as ameaças de redes convencionais, algumas soluções já foram propostas e desenvolvidas. Entre elas podemos citar o uso de módulos firewall, serviços para controle de acesso, monitoramento e auditoria, proteção de privacidade, serviços para detecção de intrusos e melhoramento das políticas da rede SDN. (ALSMADI; XU, 2015)

3.3.2 Anonimização

Segundo Boschi e Trammell (2011), anonimização é a modificação dos dados em tráfego na rede de forma a proteger a identidade dos usuários finais. Para que isso ocorra é necessário que se remova a habilidade de identificar a conexão entre dois sistemas finais enquanto se protege a integridade dos dados. A anonimização é normalmente classificada de acordo com duas propriedades: recuperabilidade e contagem. Todas as técnicas de anonimização devem mapear identificadores ou valores em um espaço à parte, de acordo com alguma função específica. Caso essa função seja invertível, ou seja, caso seja possível recuperar o identificador ou valor real sem utilizar outra informações adicionais, então a técnica de anonimização utilizada é dita recuperável. Já a propriedade de contagem diz respeito à dimensão do espaço anonimizado e denota como a contagem de valores/identificadores únicos é preservado por uma função de anonimização.

O tráfego de dados consiste em uma sequência de pacotes com origem e destino específicos. Esse fluxo é definido através de 5 campos: endereço IP de origem, endereço IP de destino, número da porta de origem, número da porta de destino e tipo de protocolo, sendo que

ainda existem outros campos adicionais que identificam os pacotes e o fluxo a que pertencem. O processo de anonimização consiste justamente em proteger ou tornar anônimo dados que identifiquem unicamente os sistemas finais de origem ou destino (FARAH; TRAJKOVIć, 2013). Tendo em vista as características anteriores, a anonimização pode ser alcançada de forma geral por 4 modos: (BREKNE; ÅRNES; ØSLEBØ, 2006)

1. Remoção de dados: Implica na remoção de dados irreversível. Pode ser implementado substituindo os dados com uma constante;
2. Randomização: Implica na substituição dos dados por uma informação aleatória.
3. Generalização: Implica na substituição de dados por outros dados gerais. Caso esses outros dados identifiquem unicamente um usuário a anonimização pode falhar.
4. Truncamento: Tipo de generalização em que um valor fixo de bits menos significativos são deletados, enquanto o restante se mantém original.

Atualmente já existem diversos algoritmos de anonimização que utilizam de forma geral um dos modos citados com algumas alterações. De acordo com Bomfim et al. (2017) a maioria dos serviços de anonimização para redes SDN utiliza a técnica de deslocamento aleatório, contudo, a remoção ou substituição aleatória dos endereços IP inutiliza o pacote no sentido de não ser possível uma análise do fluxo. Também foi constatado em seu mapeamento que os endereços IP são os dados mais frequentemente anonimizados.

3.3.3 Redes Autonômicas e Teoria do Perigo

Lidar com a complexidade de sistemas e infraestruturas, bem como suas abstrações, sempre foi o maior desafio para a área da computação. De acordo com Horn (2001), o desenvolvimento de sistemas cada vez mais poderosos e complexos tem o propósito de contribuir para a evolução humana em seus negócios e necessidades, portanto, a automação dos processos sempre fez parte do progresso. Entretanto, a crescente complexidade de Infraestrutura de TI, aliada à Internet como conhecemos hoje, chegou em um patamar que ameaça estagnar os benefícios que a tecnologia pode oferecer. Essa complexidade também chegou aos administradores e usuários comuns gerando desafios também no quesito gerenciamento dessas tecnologias. Nesse contexto, o autor introduz a ideia de que a computação autonômica é último recurso possível para o progresso.

A computação autonômica envolve o conceito de auto gerenciamento do sistema a partir dos objetivos do administradores. O termo “autonômico” tem origem da biologia, através do conceito do sistema nervoso autonômico. Essa analogia se dá justamente porque o corpo humano possui um conjunto de sistemas altamente complexos que são interdependentes e se auto gerenciam de acordo com fatores externos e internos. De acordo com Behringer et al. (2015) existem algumas definições específicas e importantes para a área de redes autonômicas:

- Intenção: Política abstrata de alto nível usada para operar a rede. É definida e fornecida por uma entidade centralizada.
- Domínio Autonômico: Um conjunto de nós autonômicos que instanciam a mesma intenção.
- Função Autonômica: Um funcionalidade ou função que não requer configuração e pode derivar todas as informações necessárias através do auto-conhecimento, descoberta ou intenção.
- Agente de serviço autonômico: Agente implementado em um nó autonômico que também implementa uma função autonômica.
- Nó Autonômico: Um nó da rede que possui exclusivamente funções autonômicas e não requer nenhuma configuração. Pode ser exemplificado pelo roteador, switch, PCs, entre outros.
- Rede Autonômica: Rede que contém exclusivamente nós autonômicos. Pode conter um ou vários domínios autonômicos.

Além dessas definições, Kephart e Chess (2003) também compara conceitos entre a computação tradicional e autonômica.

Seguindo as analogias da área da Biologia para redes autonômicas, os mecanismos de defesa do corpo vem sendo muito pesquisados para que seus conceitos sejam aplicados a sistemas computacionais. A Teoria do Perigo, proposta por Matzinger (1994), tem como base a ideia de que uma resposta imune é gerada como reação a uma situação de perigo, devido ao comportamento das células imunes do corpo, chamadas de Células Dendríticas(do inglês *Dendritic Cells* (DC)).

A abordagem da Teoria do Perigo contrapõe a visão anterior da Seleção Negativa, que afirma que a resposta imune tem origem nos estímulos que o ambiente reconhece como dano. Ela propõe que a morte das células ocorre de maneira controlada, um comportamento conhecido apoptose. Quando esse fenômeno acontece, moléculas imunossupressoras são liberadas no corpo, indicando que a situação não é de perigo. Já numa situação real de perigo, quando ocorre a morte não-natural de células, o corpo libera sinais de perigo para que o sistema imunológica possa agir de acordo com a concentração das mesmas (MATZINGER, 1994).

Mais especificamente, as DC's têm como responsabilidade capturar, revelar e transformar antígenos nas células T, que são responsáveis pelo reconhecimento e combate de corpos estranhos, bem como também recebem outros sinais vindos da vizinhança. Dos sinais existentes 4 são citados como principais (GREENSMITH; AICKELIN, 2007):

- Sinais PAMP (do inglês *Pathogen Associated Molecular Pattern* e traduzido como Padrão Molecular Associado a Patógenos): são indicadores de uma situação anormal, portanto

quando são recebidos as DC's produzem as chamadas moléculas coestimulatórias (CSM) para representar a indicação de um elemento estranho no corpo.

- Sinais Seguros (do inglês *Safe Signs* (SS)): são indicadores de um situação normal, no qual as células morrem naturalmente, portanto, não há necessidade de uma reação imune.
- Sinais de Perigo (do inglês *Danger Signs* (DS)): são indicadores de uma situação anormal para o caso de morte não natural de uma célula. São menos confiáveis do que os sinais PAMP.
- Sinais de Inflamação (do inglês *Inflammation Signs* (IS)): sinais que possuem efeito de amplificação para os outros sinais citados.

Considerando a concentração desses sinais, o estado de maturação das DC's podem ser classificados como Imaturo, Semi maduro ou Maduro. O estado imaturo é o estado inicial de uma DC, em que não há concentração de sinais recebidos. O estado semi maduro corresponde ao estado em que a DC recebe maior concentração de sinais seguros. Já o estado maduro ocorre quando as DC's possuem uma maior concentração de sinais PAMP ou de perigo em relação aos sinais seguros.

A partir desse mecanismo de defesa, Greensmith e Aickelin (2008) desenvolveram o Algoritmo Determinístico das Células Dendríticas, projetado para lidar com sistemas que necessitem de detecção de ameaças. Analogamente ao comportamento das DC's do corpo humano, o algoritmo utiliza a ideia de que sinais específicos do sistema podem ser coletados por uma Célula Dendrítica Artificial(em inglês *Artificial Dendritic Cell* (ADC)), de forma que esta possa rotular tais sinais como normais ou anormais e, dependendo da concentração destas nas ADC's, elas podem atingir um dos estados de maturação mencionados.

No processamento computacional, os sinais de saída são calculados a partir de pesos associados a cada tipo de sinal de entrada, conforme consta em Greensmith, Aickelin e Twycross (2009). Com base da intensidade desses sinais, na fase do processamento é ainda definido o chamado MCAV (*Mature Context Antigen Value*). Ele é um índice de anomalia de um antígeno, calculado a partir da média ponderada das células maturadas em adição ao total de células migradas. Para uma situação possivelmente normal, seu valor é definido como 0 e para uma situação possivelmente anormal, o valor é 1. (GREENSMITH; AICKELIN, 2007)

3.4 Trabalhos Relacionados

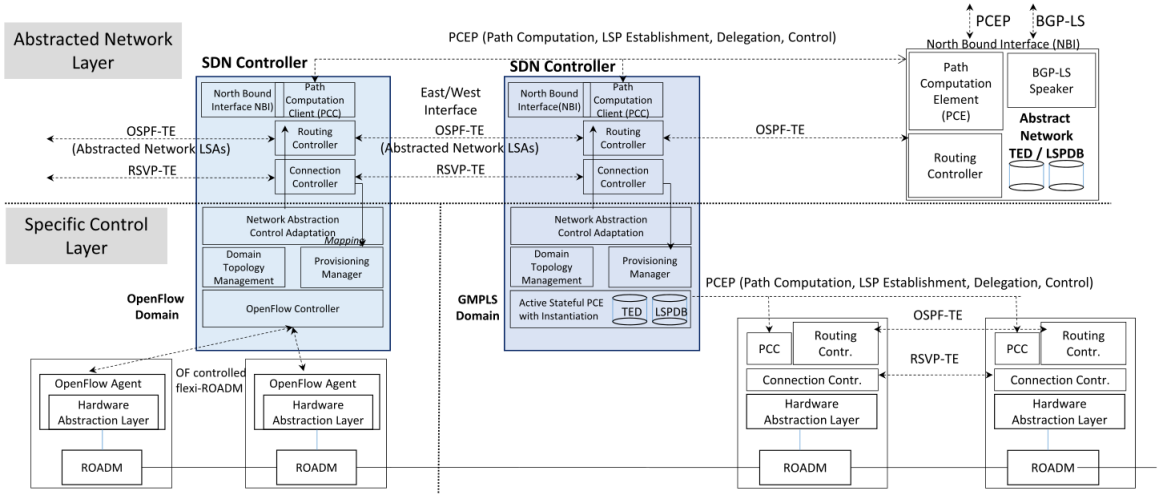
Dado o caráter de inovação do paradigma SDN e da escassez de trabalhos encontrados que se alinhem ao tema aqui abordado, conforme discutido no capítulo 2, explanaremos a seguir os trabalhos que mais se relacionam (e que mais tem a contribuir) com nosso trabalho, ao mesmo tempo em que destacamos, para cada um deles, quanto nossa solução difere.

3.4.1 NMS

Em Casellas et al. (2015), é apresentada uma arquitetura de duas camadas que implementa um Sistema de Gerenciamento de Rede (do inglês *Network Management System* (NMS)) para orquestrar múltiplos controladores SDN em redes óticas e seus respectivos subdomínios de rede, atendendo às restrições tecnológicas dos vários fabricantes existentes neles. Para isto implementações de *Generalized Multiprotocol Label Switching* e *OpenFlow* foram feitas nos controladores, e na camada superior (de abstração) trabalham com representações dos subdomínios para operar em cima de um único domínio abstrato. Através das interfaces E/W (usando protocolo OSPF (*Open Shortest Path First*) com extensões para engenharia de tráfego) existe comunicação de modo a criar e coordenar os fluxos inter-subdomínios (na camada inferior, de controle específico). Para validação deste trabalho foi utilizado um *testbed* experimental com foco na latência do provisionamento de serviços e na sobrecarga do plano de controle.

Nota-se que neste referido trabalho, além de não serem apresentados quais controladores foram usados no experimento, não é tratada nenhuma estratégia de recuperação do controle de um subdomínio em caso perda de comunicação deste com o seu respectivo controlador. Esta recuperação seria possível devido a existência no trabalho de outros controladores candidatos na arquitetura a assumir este plano de controle parcialmente perdido. A arquitetura deste trabalho é representada pela Figura 10.

Figura 10 – Arquitetura de orquestração e controle híbrido



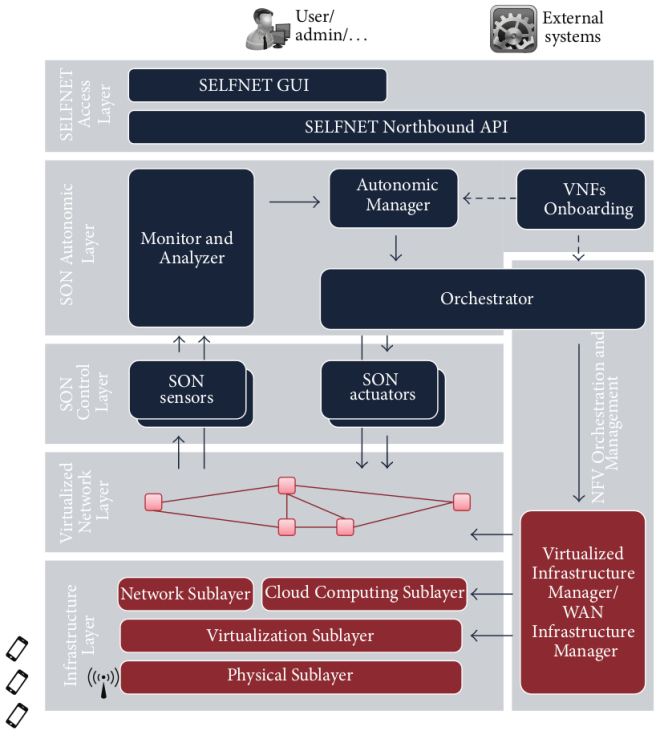
Fonte: (CASELLAS et al., 2015)

3.4.2 SELFNET

Já em Neves et al. (2016), o projeto SELFNET oferece uma arquitetura multicamadas (Figura 11), com intuito de integrar as novas tecnologias que darão substrato ao desenvolvimento e ao atendimento dos desafios impostos pelos parâmetros de *Quality of Service* (QoS), *Quality of Experience* (QoE) e *Key Performance Indicators*(KPI) definidos para as redes 5G.

As tecnologias SDN, NFV, *Self-Organizing Network* (SON), *Cloud Computing* e Inteligência Artificial são abordadas e integradas ao SELFNET em algum nível, o que torna orquestração seguramente uma necessidade vital para seu crescimento e continuidade. Além disso, este referido trabalho pretendeu contribuir ao desenvolvimento de uma solução escalável, extensível e autogerenciável para arquiteturas futuras, onde existam autonomamente monitoramento, manutenção, provisionamento de ferramentas e serviços de redes.

Figura 11 – Arquitetura Geral do Projeto SELFNET



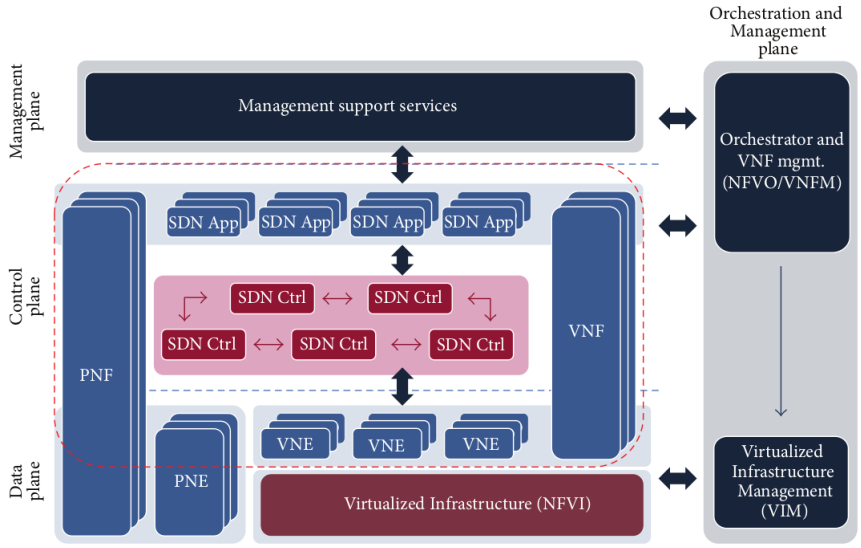
Fonte: (NEVES et al., 2016)

Dentre as camadas do SELFNET, destacamos:

- *Infrastructure Layer*: aqui estão os recursos que tornam possíveis a instanciação das VNFs, tais como rede e *storage*;
- *Virtualized Network Layer*: representa a topologia das VNFs;
- *NFV Orchestration and Management Layer*: orquestração das VNFs e controladores SDN para gerenciar os dados e redes dos *datacenters* interconectados, cuja arquitetura combinada encontra-se na Figura 12 ;

Apesar do mérito de ter apresentado uma arquitetura que combine as mais novas tecnologias de redes, permanece sem resposta a questão de como na prática será realizada a comunicação

Figura 12 – Arquitetura combinada NFV e SDN



Fonte: (NEVES et al., 2016)

entre controladores SDN de modo a favorecer a orquestração planejada , além de não ter havido validação real, nem teste experimental (com implementação de controladores diferentes, coordenação de domínios de redes, etc) da mesma.

3.4.3 SC-ARCH e ABNO

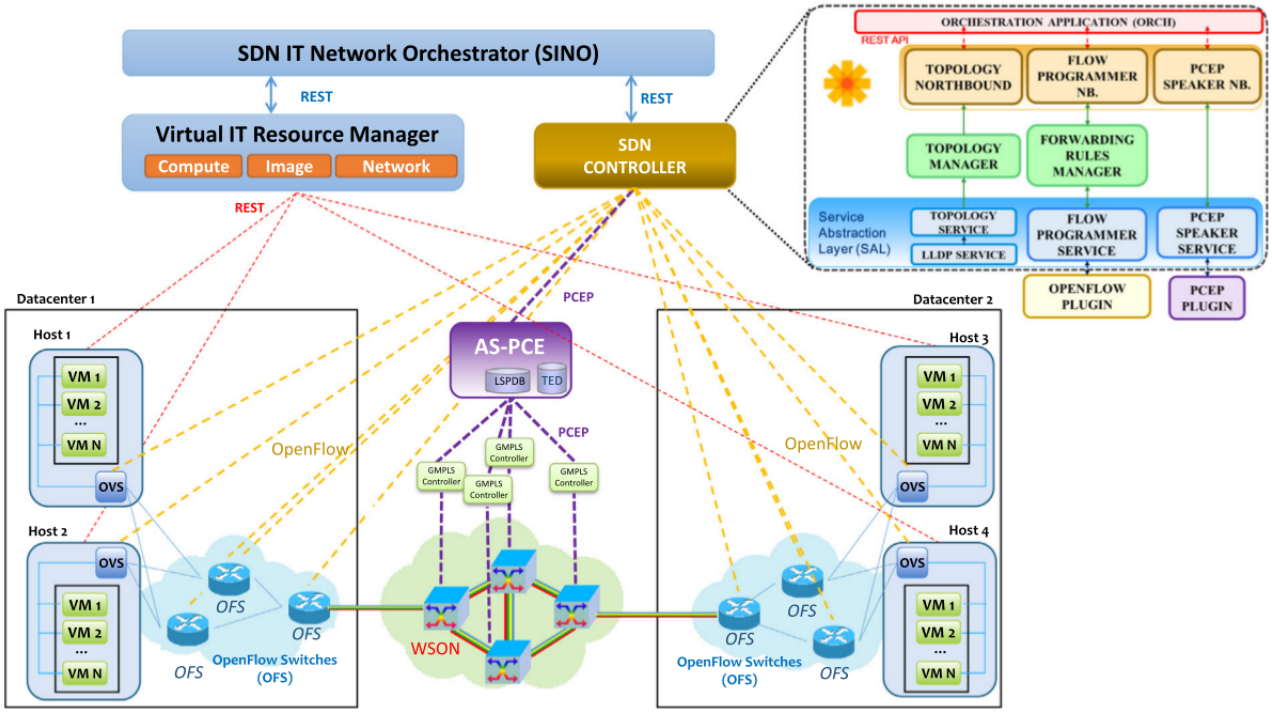
Duas arquiteturas são propostas em Mayoral et al. (2017), mas apesar de ambas trabalharem com orquestração em algum nível, em ambas a abordagem é para um controlador SDN único, trabalhando múltiplos domínios de rede. Para tornar possível a orquestração via SDN, as duas arquiteturas precisaram atender os seguintes requisitos:

- Descoberta da Topologia: os diferentes domínios podem trabalhar a sua topologia via diferentes protocolos ou interfaces, em assim sendo a abrangência total da rede orquestrada necessita ser possível implementar no controlador SDN da solução;
- Processamento de Rotas: a depender do domínio(s) da rede em questão, o controlador SDN deverá ter que calcular com elementos de rede do mesmo domínio ou de domínios de redes distintos (onde a orquestração se fará necessária);

Na primeira arquitetura (Figura 13), chamada de *Single Controller Architecture* (SC-Arch) o controlador OpenDaylight implementa uma aplicação orquestradora (ORCH) para lidar com a coordenação de fluxos que abranjam domínios de redes diferentes.

Na segunda arquitetura (Figura 14), chamada de *ABNO Architecture*(ABNO), um módulo externo ao controlador OpenDaylight realiza através de *Application Based Network Operations*

Figura 13 – Arquitetura SC-Arch

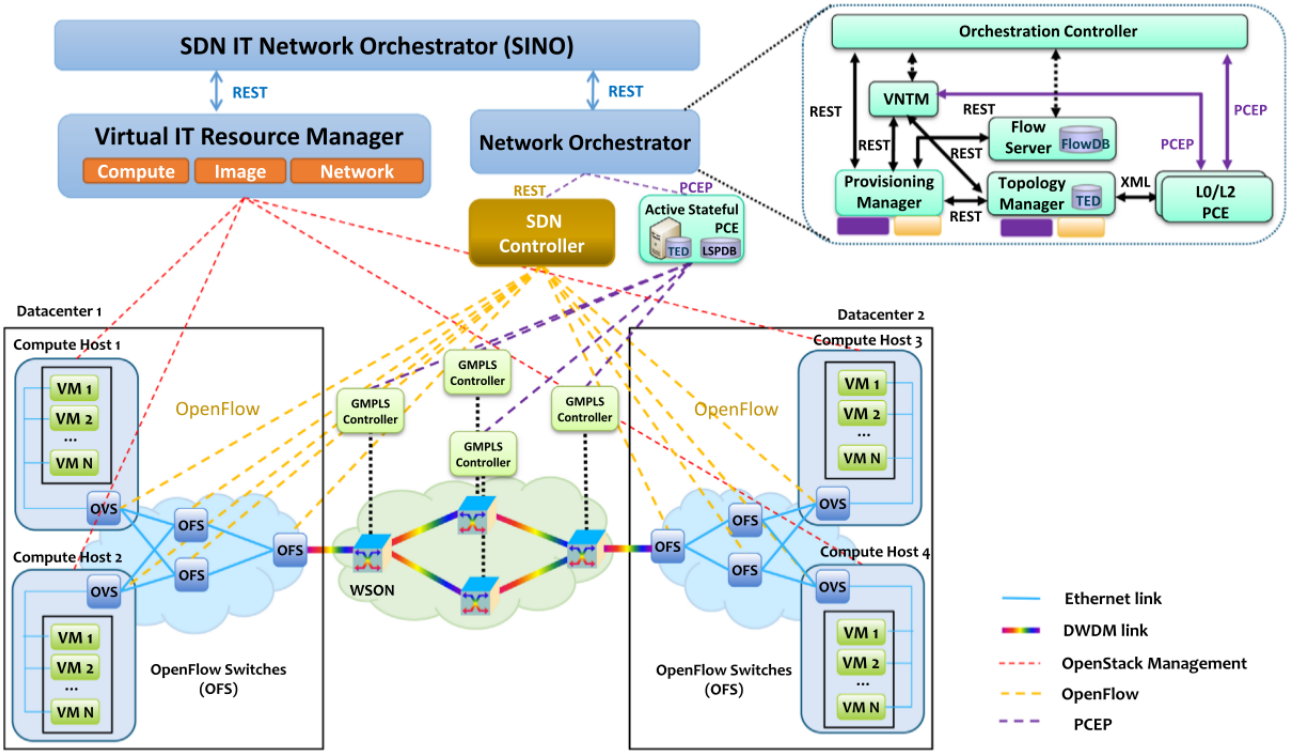


Fonte: (MAYORAL et al., 2017)

a orquestração da rede, através do processamento dos fluxos dos vários domínios de redes controlados.

Fica esclarecido neste referido trabalho a possibilidade de se trabalhar orquestração dentro do controlador OpenDaylight ou via módulo externo que com ele se comunique por interface *NBI* implementada por interface REST (do inglês, *Representational State Transfer*). Porém, repete-se o ponto de falha nas duas arquiteturas, onde um único controlador fica responsável pelo plano de controle total dos domínios das redes, assim como ficou para uma futura continuidade do mesmo trabalhar com controladores SDN além do OpenDaylight e cenários com múltiplos controladores.

Figura 14 – Arquitetura ABNO



Fonte: (MAYORAL et al., 2017)

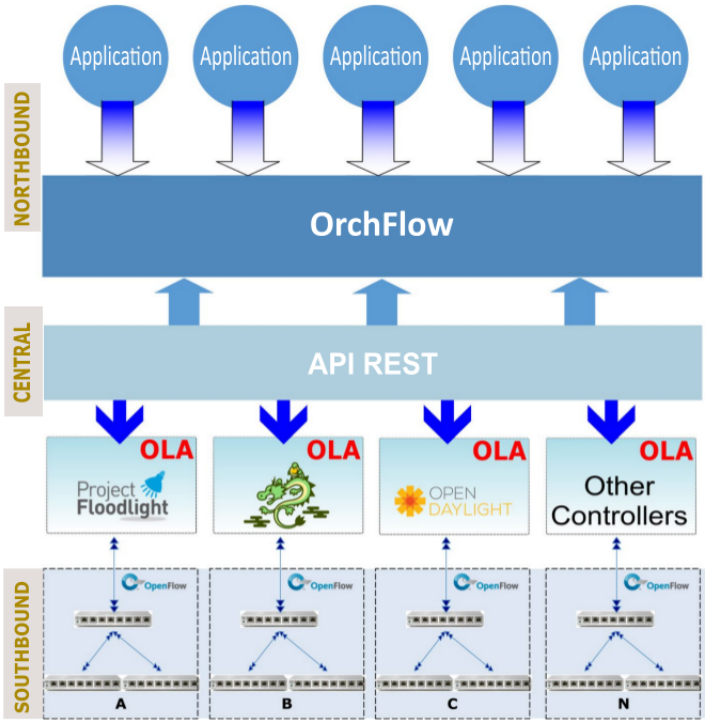
3.4.4 OrchFlow

Por último falaremos sobre *OrchFlow*, arquitetura de orquestração via controladores SDN iniciada em Frate (2017) cuja referência mais recente foi encontrada em Frate, Marczuk e Verdi (2019), e é sobre esta última que explanaremos a seguir. Segundo os autores *OrchFlow* é um *middleware*, um programa de orquestração para redes SDN baseadas em protocolo *OpenFlow*. A Figura 15 apresenta sua arquitetura. Foi desenvolvido em linguagem *Java*, utiliza um módulo interno a cada controlador SDN, chamado de *OrchFlow Local Agent*(OLA), responsável por armazenar as regras OF assim que um evento de chegada de pacote ocorrer. Além disso, também são usados *Neo4j*, um banco de dados orientado a grafos que guarda todos os detalhes dos enlaces e dos *host* de borda, e REST, para obtenção da topologia das redes. Para operação são trabalhados 3 modos:

- Proativo: todas as requisições de estabelecimento de serviço que gerem nova rota tem este novo fluxo repassado imediatamente a todos os *switchs* envolvidos;
- Reativo: novos fluxos são armazenados inicialmente no controlador (no módulo OLA), e somente sob demanda repassados aos *switchs*;
- Híbrido: inicialmente age como modo Reativo (armazenando novos fluxos no OLA), mas após necessidade de integração com subdomínios adjacentes, repassa para todos *switchs*

envolvidos a partir daí atuando como modo Proativo;

Figura 15 – Arquitetura OrchFlow



Fonte: (FRATE; MARCZUK; VERDI, 2019)

Para avaliação da arquitetura *OrchFlow* foram avaliados 5 subdomínios de redes a serem orquestrados , 2 implementações de controladores SDN (Ryu e Floodlight) e, como serviço de rede que gerasse fluxo inter-subdomínios, foi usada a ferramenta *Iperf*.

De todos os trabalhos relacionados, Frate, Marczuk e Verdi (2019) foi o que mais forneceu subsídios teórico e prático para este trabalho, com o porém de novamente não ter resolvida a questão de controlar um mesmo domínio de rede SDN com mais de controlador, de modo a repassar (em caso de falha) ou compartilhar (para balanceamento de processamento ou integração de serviços) o plano de dados deste mesmo domínio.

3.5 Comparativo

Ao final reunimos na Tabela 8 as ferramentas SDN e a abordagem de orquestração de cada um dos trabalhos relacionados, e comparamos com nosso trabalho realizado:

Tabela 8 – Aspectos considerados por cada trabalho relacionado e por nossa solução

Trabalho	**Ferramentas SDN**	**Orquestração abordada**
(CASELLAS et al., 2015)	OF modificado para redes *grid* óticas, mas controladores não citados	Abstração única de subdomínios específicos em arquitetura hierarquizada de orquestração
(NEVES et al., 2016)	não apresentadas	Arquitetura combinada entre controladores SDN e VNFs
(MAYORAL et al., 2017)	OpenDaylight e REST	Controlador SDN único
(FRATE; MARCZUK; VERDI, 2019)	Ryu e Floodlight	OrchFlow
Arquitetura de Orquestração desta dissertação	**Controladores Ryu e RunOS, Simulador Mininet**	**Arquitetura de controladores SDN heterogêneos em único domínio administrativo de rede**

Os trabalhos relacionados, encontrados durante a Revisão Sistemática, trazem cada qual sua contribuição para a pesquisa de Orquestração, mas a esta solução se diferencia no sentido de avançar em relação a eles ao trabalhar simultaneamente com:

- Controladores heterogêneos: uma SDN que trabalhe com um mesmo controlador instanciado de maneira repetida, ou que utilize um controlador que é projetado de forma distribuída, tem uma orquestração facilitada ao lidar com controladores implementados na mesma linguagem de programação e que utilizem as mesmas interfaces NBI e SBI. Aqui é encarado o desafio mais difícil de orquestrar controladores heterogêneos por se tratar de um problema mais amplo e que traria maior contribuição à pesquisa;
- Domínio administrativo único: particionar uma SDN (e por consequência os planos de dados e controle) pode não ser sempre possível, além de contornar o desafio da orquestração ao controlar de modo individual cada subdomínio da rede. Elencar como domínio único uma abstração que representa o somatório das redes particionadas não representa com justiça o cenário de uma orquestração real, pois os controladores envolvidos operariam de modo isolado, ficando seus serviços SDN sem a possibilidade de integração que o orquestrador proveria;
- Solução experimentada e testada: adicionado à proposição de solução de orquestração (à qual alguns trabalhos se limitam), esta arquitetura foi totalmente implementada em laboratório e testada segundo parâmetros validados pela literatura;
- Serviços Orquestrados: os serviços SDN orquestrados por esta arquitetura e seus respectivos controladores foram explicitamente apresentados e explorados, de modo a permitir a outros pesquisadores avaliar nossa solução e ter embasamento para trabalhar propostas alternativas à esta;

A Tabela 9 sumariza as diferenças entre as soluções apresentadas nessa dissertação e os demais já comentados.

Tabela 9 – Diferenças de cada trabalho relacionadas com a nossa solução

Trabalho	**Controladores Heterogêneos**	**Domínio Único**	**Realizou Experimentos**	**Orquestrou Serviços**
(CASELLAS et al., 2015)	Não mencionado	Não	Sim	Sim
(NEVES et al., 2016)	Sim	Não	Não	Não
(MAYORAL et al., 2017)	Sim	Não	Sim	Sim
(FRATE; MARCZUK; VERDI, 2019)	Sim	Não	Sim	Não
Arquitetura de Orquestração desta dissertação	Sim	Sim	Sim	Sim

4

Arquitetura de Orquestração

Neste capítulo serão descritos os principais esforços empregados neste trabalho. Iniciando com a proposta de Arquitetura de Orquestração ORCHDOMAIN, que soluciona os problemas até o momento abordados ao se trabalhar com o tema da pesquisa. Em seguida são descritos os serviços SDN que foram trabalhados no experimento que validou a arquitetura aqui desenvolvida, MAdPE-K/SDN e BomIP. Finalizamos com ORCHSEC, que foi a instanciação de ORCHDOMAIN para o experimento de orquestração dos serviços SDN citados, projetado para testar a implementação, gerar dados, analisar resultados e validar o trabalho desenvolvido.

4.1 Arquitetura ORCHDOMAIN

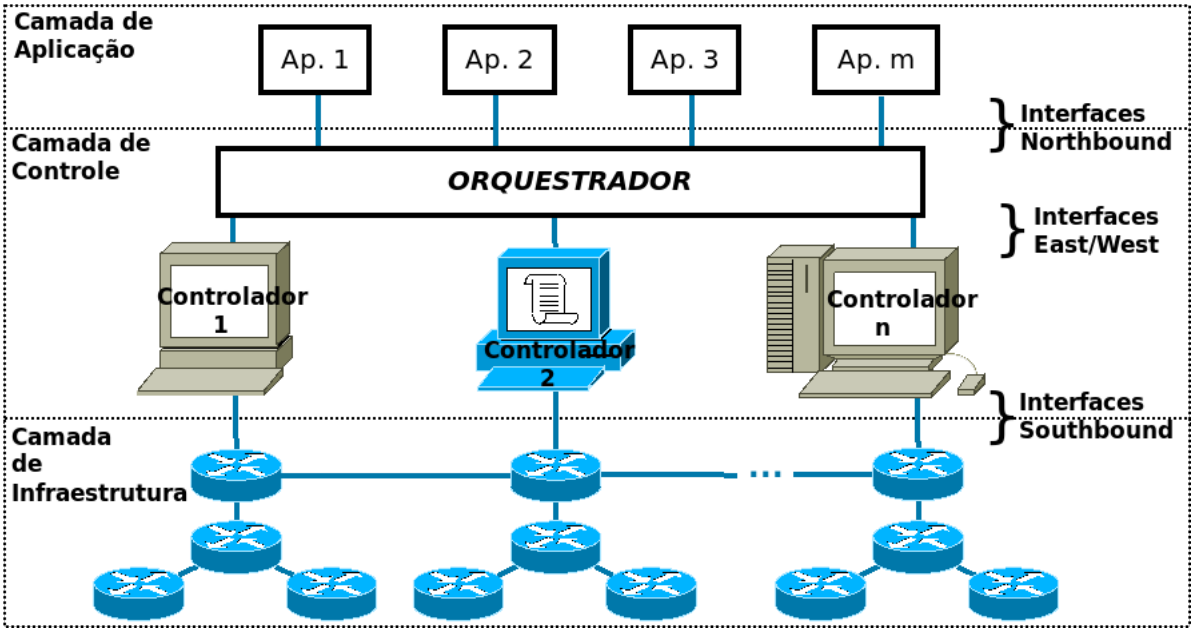
Diante dos desafios para SDN e para atender às demandas de orquestração para serviços de redes (como os serviços de segurança) é proposta uma arquitetura denominada ORCHDOMAIN, representada na Figura 16.

Tomada por base a arquitetura SDN chancelada pela ONF, os pontos principais de ORCHDOMAIN que permitem dentro do novo paradigma orquestrar serviços são :

1. Orquestrador: adicionado à camada de controle, é responsável por implementar a comunicação entre todos os controladores envolvidos (com suas linguagens de programação e interfaces, NBI, E/W e SBI específicas a cada um). É ele quem passará a interfacear as requisições que as aplicações dos usuários farão para os serviços SDN disponibilizados pelos controladores, observando as questões de balanceamento, concorrência, simultaneidade, *deadlocks* e tudo que envolve a disputa pelo mesmo plano de controle e de dados;

2. Domínio administrativo único: para uma visão total da topologia sem necessidade de maiores intervenções e real enfrentamento do cenário de orquestração, aqui será possível para todos os controladores visualizar toda topologia de rede de modo concreto, sem depender

Figura 16 – Arquitetura ORCHDOMAIN



Fonte: Autoria Própria

do orquestrador com ponto central para intermediar a comunicação entre dispositivos de subdomínios separados. Desse modo não há necessidade de processamento adicional para conseguir uma visão geral de toda rede, que por padrão era unificada e autocontida na camada de infraestrutura;

3. Controladores heterogêneos integrados: com serviços SDN sendo concorrentes, balanceados, hierárquicos, paralelizados, e/ou quaisquer outro modos de operação que o orquestrador permita, é possível trabalhar com múltiplos controladores heterogêneos simultaneamente na camada de controle.

   Assim dos controladores distribuídos

   Apesar desta ser uma solução direta no sentido de implantar de modo mais imediato funcionalidades como tolerância a falhas (com substituição facilmente configurável em se tratando da mesma linguagem de programação para todos os controladores), ou mesmo balanceamento de fluxos (caso a linguagem em questão aborde paradigma concorrente ou paralelo), o mesmo pode ser conseguido ao trabalharmos com controladores diferentes. Obviamente essa integração não será solução pronta, como no caso anteriormente citado (e quando o projeto do controlador é pensado para executar de forma distribuída), mas o interfaceamento E/W realizado pelo orquestrador pode incorporar as várias linguagens envolvidas, com o adicional de trazer as vantagens de seus paradigmas de programação (paralelo, concorrente, objeto, funcional, etc) para nossa arquitetura ;

Para ORCHDOMAIN cobrir todas as possibilidades existentes, não restringimos as

interfaces SBI para nenhum protocolo de interface entre controladores e dispositivos de rede, mas vale ressaltar que, pelo observado na literatura e pela crescente incorporação por parte dos fabricantes do paradigma SDN, muito provavelmente os controladores heterogêneos envolvidos terão todos suporte para o protocolo Openflow. Além de não excluir a possibilidade de adicionar protocolos outros de interfaceamento com a infraestrutura, que podem existir, a depender dos controladores envolvidos na orquestração, esse denominador comum no suporte cada vez mais popularizado ao Openflow traz para nossa arquitetura uma aplicabilidade mais próxima aos cenários reais das redes, que é uma das conquistas do nosso trabalho.

4.2 Serviços de Segurança *SDN*

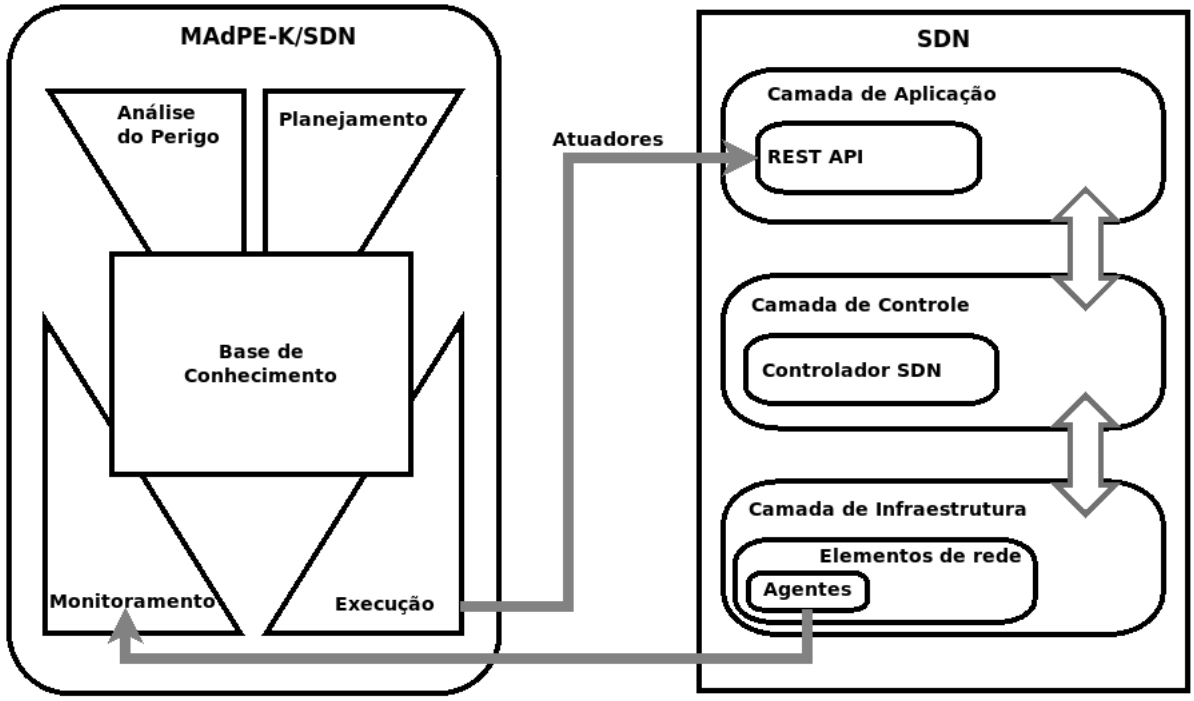
Um ponto importante para nossa pesquisa seria encontrar trabalhos com serviços SDN cuja orquestração fizesse sentido, para dar lastro à arquitetura que estamos propondo. Outro fator crucial seria um entendimento profundo destes (através de documentação farta, código implementado e se possível contato direto com os desenvolvedores) para implantação dentro do ambiente disponível, e assim tornar possível posteriormente fazer os experimentos com Orquestração que serão realizados para validação da pesquisa. Assim sendo, foram selecionados os trabalhos de Bomfim (2017) e Menezes (2018), cujos autores forneceram documentação, implementações e disponibilidade em colaborar. Para fins de esclarecimento, a seguir são brevemente descritos cada um destes trabalhos base para nossa proposta.

4.2.1 MAdPE-K/SDN

Desenvolvido por Menezes (2018) e ilustrado pela Figura 17, o serviço MAdPE-K (em Controlador Ryu) trabalha o aspecto de segurança em redes SDN através da abordagem bioinspirada no sistema imunológico humano (mais especificamente aplicando a teoria de detecção de perigo) e de redes autonômicas. Através de agentes nos *hosts* fazendo o papel de células dendríticas, sinais são gerados diante de anomalias reconhecidas na rede, de modo a disparar uma resposta autonômica via controlador Ryu, que bloquearia o fluxo via comando OF no *switch* de modo a eliminar a fonte de perigo antes que a mesma se alastrasse pelo restante da rede. Da documentação deste trabalho sabemos que, além do controlador Ryu (programado por linguagem *python*) foram usadas as ferramentas GNS3 para emulação da rede, hipervisor VirtualBox para virtualização de ambientes, sistema operacional Ubuntu 16.04 LTS para máquinas virtuais, container Docker para instanciação do OpenVSwitch 2.5, pacotes JSON (*JavaScript Object Notation*) e API REST usados no cenário de teste.

O serviço MAdPE-K/SDN, foi inspirado no trabalho de Oliveira, Salgueiro e Moreno (2013) a partir dos resultados obtidos quanto à predição de ataques e no modelo de gerência autonômico com base na Teoria do Perigo. O fluxo do serviço pode ser dividido em 4 fases:

Figura 17 – Arquitetura MAdPE-K



Fonte: (MENEZES, 2018)

- Monitoramento: compreende a coleta dos dados dos dispositivos da rede a partir de entidades chamadas de agentes. Tais dados podem ser obtidos a partir de arquivos de *logs*, alertas ou outros dados fornecidos pelo sistema operacional;
- Análise do perigo: os agentes, após coletarem os dados, os repassam para o gerente (todos na camada de infraestrutura deste serviço), que faz o devido processamento dos sinais gerados pelo algoritmo de células dendríticas, responsável pela detecção antecipada do perigo, cujo pseudocódigo está mostrado no Código 1. A referência para determinar o nível de perigo é dada pelo valor resultante do **MCAV**;
- Planejamento: após análise, por meio dos gerente as ações são planejadas para mitigar os ataques com base nas políticas definidas pela gestão da rede. No caso do trabalho estudado, a ação consiste no bloqueio do fluxo analisado;
- Execução: O gerente, após devido planejamento, envia suas instruções para o controlador SDN por meio da interface *Northbound* via API, que as executa;

Código 1 – Pseudocódigo do Algoritmo Determinístico de Células Dendríticas

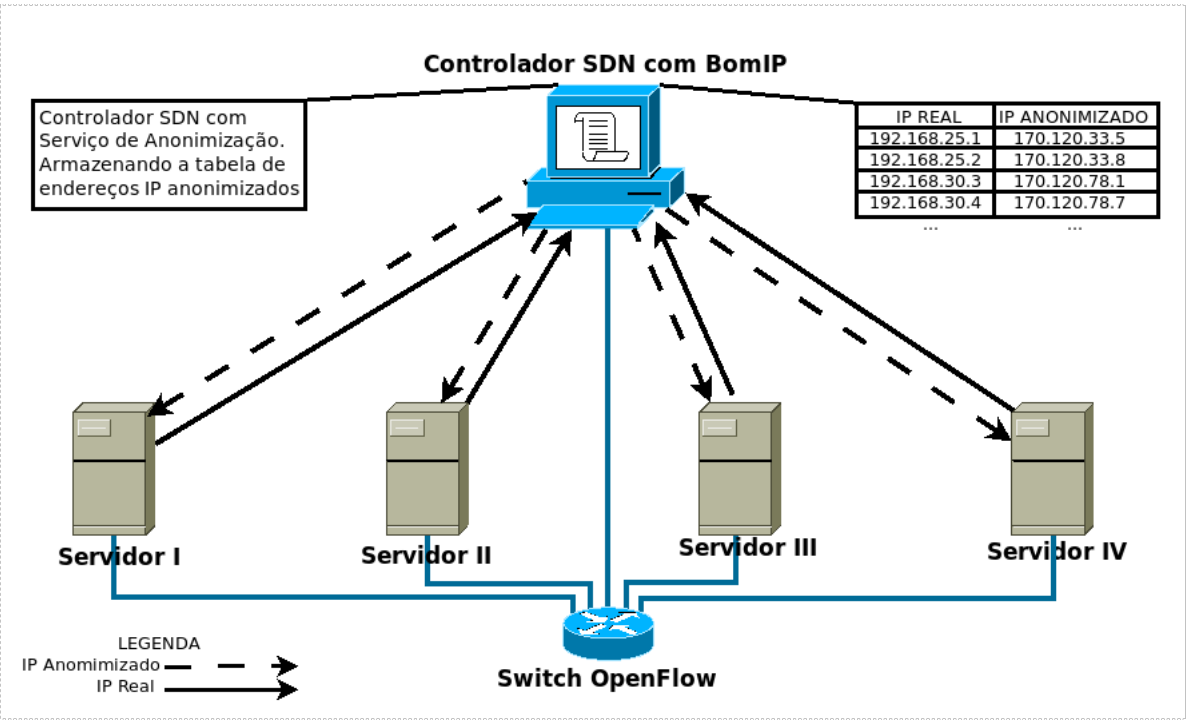
```
input: antigens and signals
output: types of antigens and Ka

defines sizePopulation of DCs;
initialize DCs;
        while there is input data
        choose input
                        case antigen
                                antigenCounter++;
                                cell_Index = agCounter %= sizePopulation;
                                DC-> antigen of index cell_Index++;
                                updates the profile of the DC antigen;
                        end
                        case sign
                        calculate csm and k;
                        for each DC
                                DC.time -= csm;
                                DC.k += k;
                                if DC.time <= 0 then
                                        store antigen, DC.k;
                                        renew DC;
                                        update MCAV;
                                end
                        end
                end
         end
 end
 for each type of antigen
        calculate Ka anomaly metrics;
 end
```

A arquitetura deste serviço, se relaciona com as 3 camadas da arquitetura SDN, sendo que a etapa de monitoramento, análise e planejamento é feita no nível da infraestrutura, enquanto a execução é feita pela camada de controle a partir do uso da API Rest, na camada de aplicação. A base de conhecimento, apesar de não ser uma etapa, é uma estrutura embutida no serviço, que auxilia a troca de informações entre todas as fases, sendo responsável pelo autoconhecimento do sistema.

4.2.2 BomIP

Desenvolvido por Bomfim (2017) e apresentado na Figura 18, o serviço BomIP (em Controlador RunOS) trabalha o aspecto de segurança em redes SDN através da anonimização dos endereços IP's de um fluxo detectado como ataque. O controlador RunOS ficaria responsável por gerenciar os endereços reais e anonimizados, de modo a isolar do restante da rede o fluxo suspeito, e para conter e realizar o tratamento de mitigação da ameaça. No controlador RunOS foi programada em linguagem C++ todos os componentes de anonimização, e os experimentos utilizaram *traces* de ataques de redes publicamente disponibilizados para geração de dados, cuja análise permitiu avalizar o desempenho do algoritmo de anonimização implementado no controlador RunOS.

Figura 18 – Funcionamento do Serviço BomIP

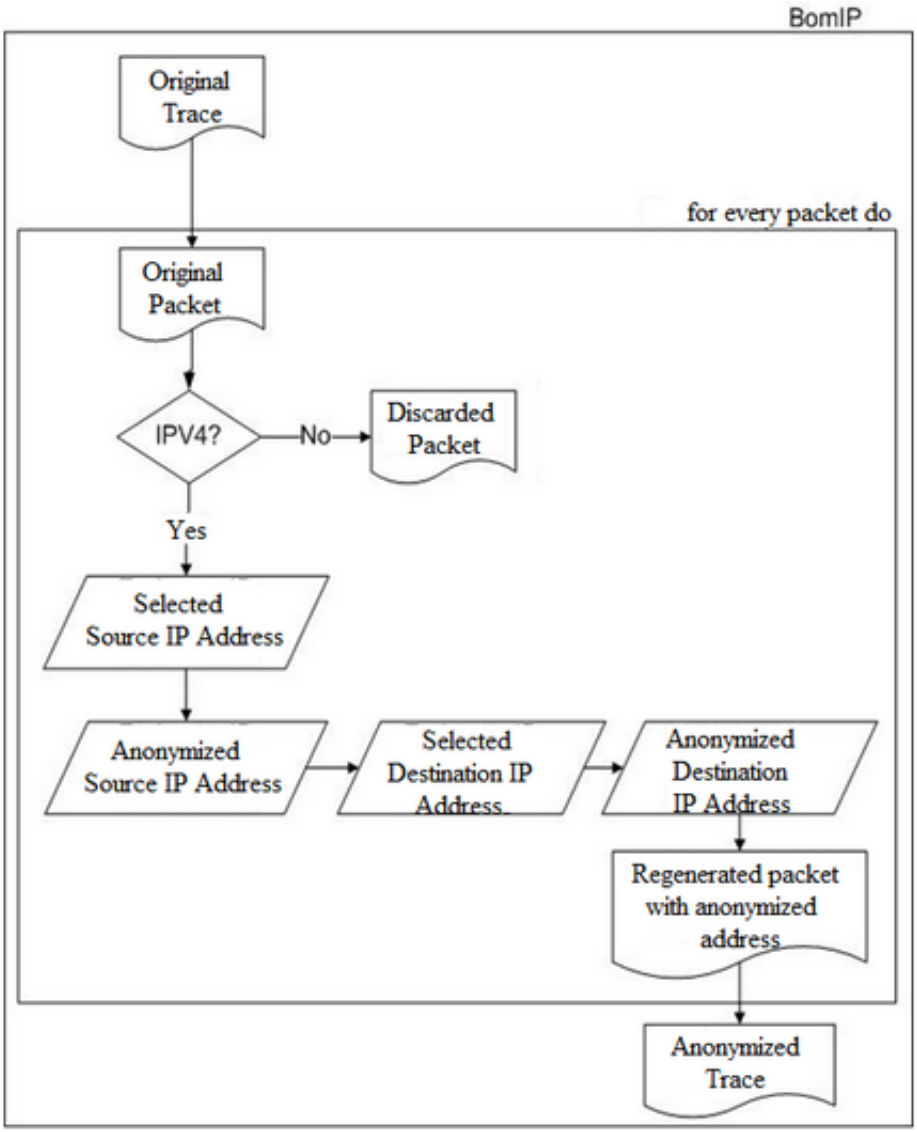


Fonte: (BOMFIM et al., 2017)

O processo de anonimização está demonstrado na Figura 19 e segundo consta em Bomfim et al. (2017), para cada IP (de origem e de destino do fluxo trabalhado) os 4 octetos são processados separadamente. Para cada octeto com valor$x$, é buscado o valor correspondente anonimizado y no vetor de anonimização na posição x. Caso na posição em questão o valor anonimizado já tenha sido gerado, o processo continua com o octeto seguinte. Caso na posição x o flag *-1* apareça, o valor anonimizado y é gerado randomicamente, verificado para que não esteja repetido ao longo do vetor de anonimização para finalmente ser guardado. Desse modo, para cada valor de octeto x, o vetor de anonimização vai guardando um único valor anônimo correspondente y ao longo de todo processo de anonimização, para todos os IP's (sejam de origem ou destino) envolvidos no fluxo. Na Figura 20 está ilustrado um exemplo de um vetor de anonimização ao

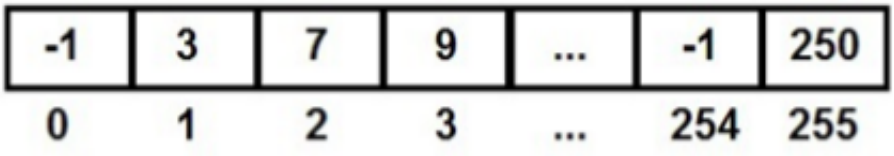
longo do processo do Serviço BomIP, com os octetos 1, 2, 3 e 255 correspondendo aos valores anonimizados 3, 7, 9 e 250. Já para os octetos 0 e 254 o *flag -1* ainda está presente, o que significa que neste exemplo ainda não foram anonimizados IP's com estes octetos.

Figura 19 – Processo de Anonimização do Serviço BomIP



Fonte: (BOMFIM et al., 2017)

Figura 20 – Vetor de Anonimização do Serviço BomIP



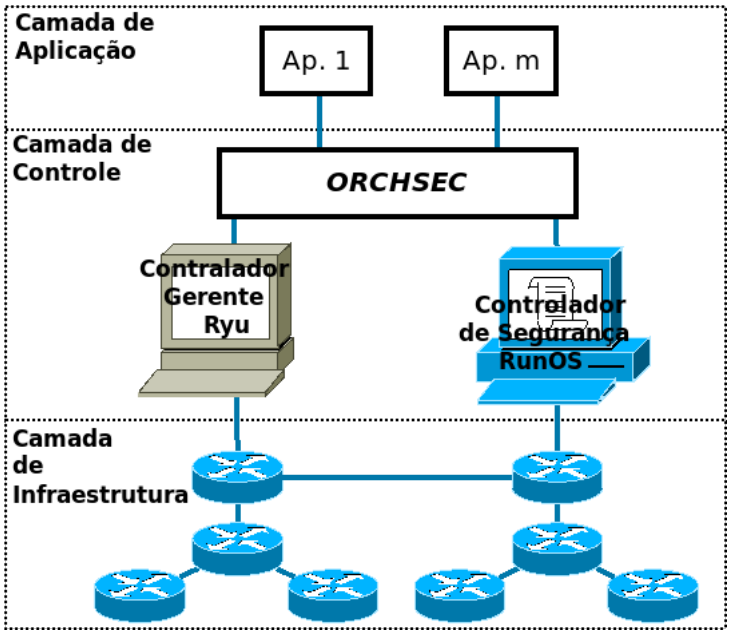
Fonte: (BOMFIM et al., 2017)

4.3 ORCHSEC

Atendendo às características específicas de cada serviço foi realizada uma instanciação de ORCHDOMAIN para os tratamentos de segurança servidos, cujo orquestrador foi chamado ORCHSEC, como ilustrado na Figura 21, a qual exibe sua arquitetura de 3 camadas. Pela figura podemos observar que todo processo de orquestração é executado na camada de controle, onde estão presentes o orquestrador ORCHSEC, o Controlador Gerente Ryu e o Controlador de Segurança RunOS. A camada de controle oferece serviço a camada de aplicação comandando a camada de infraestrutura, onde estão presentes os dispositivos de rede.

A implementação efetuada neste trabalho levou em consideração como cada serviço de segurança foi implementado no respectivo trabalho original, e tratou o Ryu como controlador gerente e o RunOS como controlador de segurança. Desse modo a orquestração entre os serviços pôde aqui ser implementada com mínimo de intervenção no código correspondente. A análise dos ataques baseada no Algoritmo de Células Dendríticas determina o nível da ameaça calculada, e uma ação de anonimização do fluxo será realizada (para isolar os IP's do atacante e da vítima), ou o próprio bloqueio do fluxo (no caso isolamento não garante segurança suficiente).

Figura 21 – Arquitetura de Orquestração para os Controladores Ryu e RunOS, e orquestrador ORCHSEC



Fonte: Autoria Própria

O serviço apresentado por Menezes (2018) envolve uma arquitetura específica de funcionamento sem a necessidade de modificação no controlador Ryu, que responderia de acordo com a análise do valor do MCAV. Já o serviço de anonimização BomIP fornecido por Bomfim (2017) não possui a necessidade de uma arquitetura específica para funcionar ou de informações extras dos dispositivos além dos IP's dos*hosts*, sendo o serviço embutido no próprio controlador

RunOS. Tendo em vista as características específicas de cada serviço surgiu o principal desafio para esse trabalho, pois a criação do orquestrador envolvia como utilizar os serviços de segurança já existentes de forma cooperativa sem necessidade de fazer grandes modificações nos próprios serviços envolvidos.

Após uma análise de ambas as implementações, o ORCHSEC atuará com referência na previsão dos ataques obtida através do Algoritmo das Células Dendríticas. Para tanto, ações tanto a nível de comunicação com o plano de controle, quanto a nível de do plano de dados foram necessárias para orquestração. Dessa forma, mediante o nível da ameaça calculado, o orquestrador decidirá por uma ação de anonimização do fluxo (para isolar os IP's do atacante e da vítima), ou o próprio bloqueio do fluxo (no caso do isolamento não ser garantia de segurança suficiente).

Em termos de código, ORCHSEC recebe como dados os valores de contexto das células dendríticas, sendo 0 para a condição onde o valor do maduro é maior que o do sinal semi maduro e 1 para a situação inversa. Como o serviço possui uma limitação relacionada a coleta de dados em tempo real, definiu-se o valor limite de 15 para coleta das amostras, tendo como referência o trabalho de Anandita et al. (2015), que projetou uma implementação simplificada de um sistema de detecção de anomalias baseada no algoritmo de células dendríticas. Após atingir limite da coleta dessas amostras, o orquestrador avalia o nível da ameaça (calculando o MCAV) e age de acordo com sua gravidade.

5
Análises e Resultados

Aqui são descritos o experimento projetado para nossa arquitetura, o ambiente em que ele foi implementado, os recursos utilizados na implementação, e as duas avaliações experimentais que fizeram o orquestrador trabalhar com ambos os serviços SDN.

5.1 Experimento

A seguir serão apresentados o fluxo de funcionamento dos testes, bem como seus resultados, demonstrando a aplicação a partir dos experimentos. A partir do *script* da topologia (demonstrado no Código 2) e dos serviços de segurança citados anteriormente foram criados dois casos para validar a arquitetura de orquestração. O experimento utilizou uma estrutura cliente-servidor para que o agente (h1) pudesse enviar os valores de contexto das células para o gerente (h2). Esse valores de contexto são obtidos como saída do algoritmo determinístico das células dendríticas, a partir da análise dos sinais coletados pelos agentes.

5.2 Ambiente de Testes

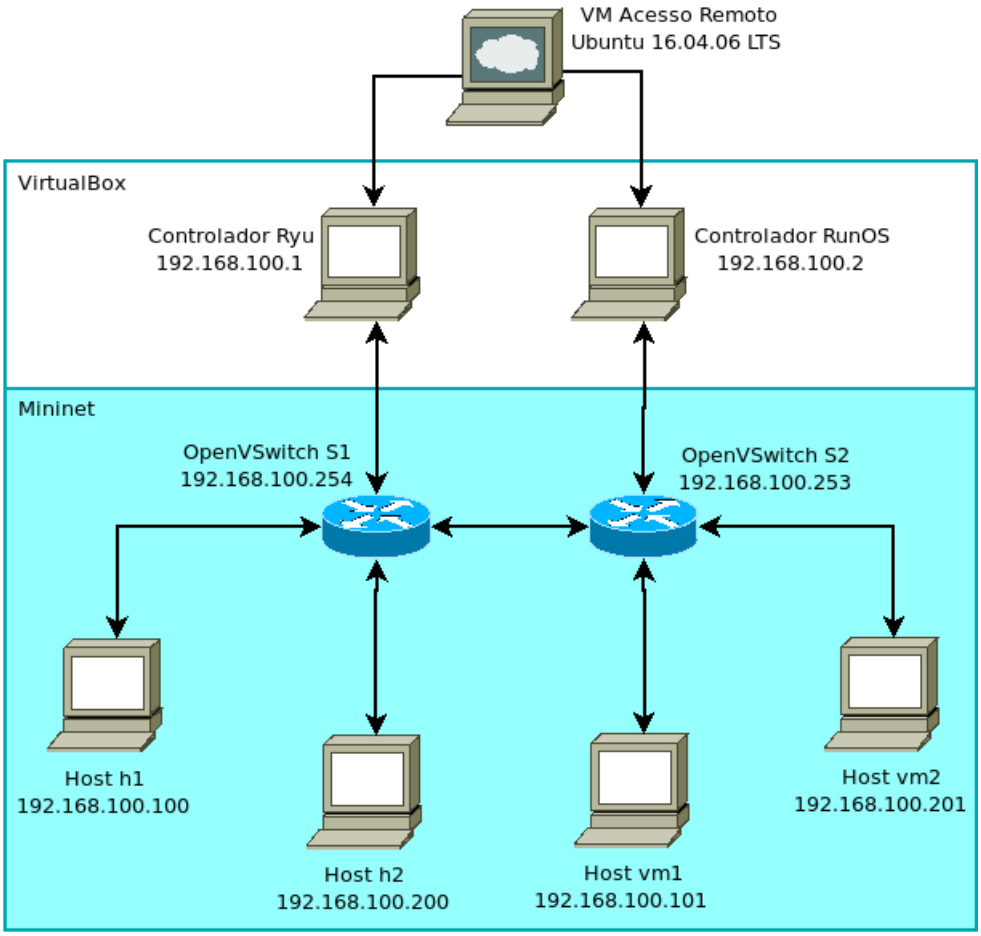
A estrutura utilizada para experimentação foi a do ELAN (*Experimental Laboratory in Computer Networks*), onde não somente esta pesquisa, mas outras em SDN são desenvolvidas, além de outras áreas de redes de computadores. Localizado no prédio do Departamento de Computação (DCOMP), o ELAN conta com uma gama de ambientes de experimentação disponíveis, dos quais para esta pesquisa destacamos a nuvem *Openstack*, implementada em um cluster de servidores DELL.

Para maior compreensão dos serviços envolvidos e dos próprios controladores, bem como para validar o mecanismo implementado, foi criado dentro do *Openstack* uma instância com software de virtualização *VirtualBox* em sua versão console e com emulador *Mininet*, que

simula com bastante precisão diversas topologias de rede, com diversos tipos de configurações a partir um *script* em linguagem *Python*.

Como mostrado na Figura 22, foram acessadas remotamente 2 máquinas virtuais criadas dentro da instância *Openstack* com a hipervisora *VirtualBox*. Cada máquina virtual foi configurada exclusivamente para uso dos controladores de forma que possam ser utilizadas pela topologia criada pelo *Mininet* de modo mais independente e próximo de uma rede real. A partir dos módulos *python* fornecidos pelo *Mininet* foi possível desenvolver, via *script* do Código 2, a topologia necessária para o experimento, sendo composta de 2 OpenVSwitch com suporte para OpenFlow 1.3 e 4 *hosts*, todos dispostos com a mesma faixa de IP. Cada controlador utilizado foi conectado a um dos *switches*, de forma a cooperarem no plano de controle dos *hosts* da rede.

Figura 22 – Topologia para experimentos



Fonte: Autoria Própria

Código 2 – *Script* Mininet da topologia do experimento

```
# -*- coding: utf-8 -*-
from mininet.net import Mininet
from mininet.node import Controller, OVSKernelSwitch, RemoteController
from mininet.log import setLogLevel, info
from mininet.cli import CLI

def twoControllersNet():

    net = Mininet( controller=RemoteController, switch=OVSKernelSwitch)
    info("***Configuring Controllers \n")
    c1 = net.addController('ryu', ip='192.168.100.1', port=5555)
    c2 = net.addController('runos', ip='192.168.100.2', port=6653)

    info("***Creating hosts\n")
    h1 = net.addHost('vm1', ip = '192.168.100.100', mac='00:00:00:00:00:01')
    h2 = net.addHost('vm2', ip = '192.168.100.200', mac='00:00:00:00:00:02')
    h3 = net.addHost('h1', ip = '192.168.100.101', mac='00:00:00:00:00:03')
    h4 = net.addHost('h2', ip = '192.168.100.201', mac='00:00:00:00:00:04')

    info("***Creating switches\n")
    s1 = net.addSwitch('s1', protocols=["OpenFlow13")]
    s2 = net.addSwitch('s1', protocols=["OpenFlow13")]

    info("***Adding links\n")
    s1.linkTo(h1)
    s1.linkTo(h2)
    s1.linkTo(s2)

    s2.linkTo(h3)
    s2.linkTo(h4)

    info("***Starting network\n")
    net.build()

    c1.start()
    c2.start()

    s1.start([c1])
    s2.start([c2])

    CLI(net)

if __name__ == '__main__':
    setLogLevel('info')
    twoControllersNet()
```

Para o serviço provido pelo RunOS utilizou-se o Sistema Operacional Ubuntu 14.04, já para o controlador Ryu foi utilizado o Ubuntu 16.04.

5.3 Recursos Utilizados

Para que houvesse experimentação do controladores de forma exploratória e também para validar o orquestrador as seguintes ferramentas e linguagens foram utilizadas ou exploradas durante o desenvolvimento deste trabalho:

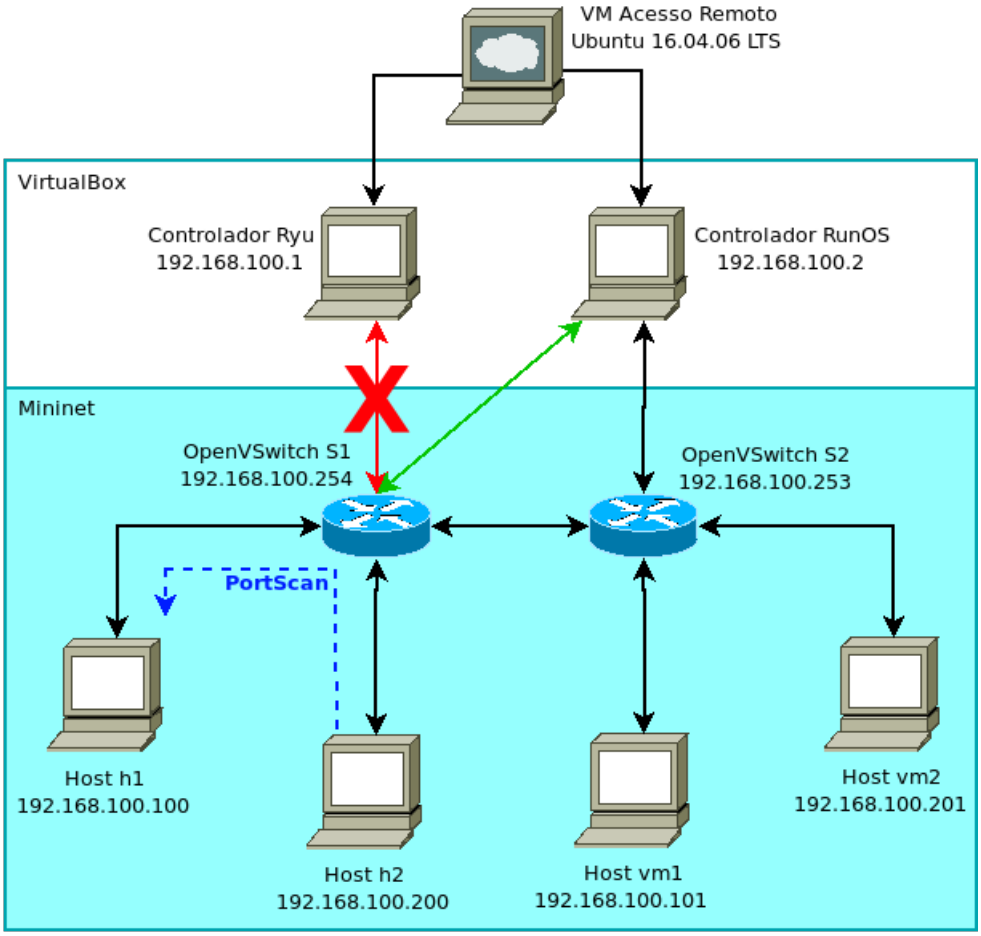
1. ambiente de implementação e execução em OpenStack;
2. sistemas operacionais Ubuntu 14.04 e 16.04 para os controladores;
3. controladores Ryu e RunOS;
4. pacote *OpenVSwitch* com suporte ao *OpenFlow* em diversas versões;
5. emulador de rede Mininet;
6. hipervisora *VirtualBox* para virtualizar sistemas operacionais dos controladores e *hosts*;
7. ferramenta *Curl* para acesso às API's dos controladores;
8. ferramenta *Ping* utilizada para testes de conectividade de rede;
9. linguagem de programação *Python* para implementação do orquestrador e para a criação da topologia com *Mininet* (além da exploração das aplicações no Ryu);
10. linguagem de programação C++ para compreensão do controlador RunOS e do anonimizador BomIP;

A seguir serão apresentados o fluxo de funcionamento dos testes, bem como seus resultados, demonstrando a aplicação a partir dos experimentos.

5.4 Avaliação Experimental com balanceamento de plano de controle

A Figura 23 ilustra a primeira avaliação experimental, onde ocorre um *PortScan* entre os *hosts* e é notificado como perigoso. Essa ameaça nesta avaliação experimental foi considerado de nível tratável e por isso o fluxo não será bloqueado, porém o orquestrador ORCHSEC solicita que o controlador RunOS se responsabilize pelos dois *switches* da topologia, para que o fluxo seja anonimizado e a ameaça não se espalhe. O Ryu continua em execução pronto para atuar, contudo a conexão com o *switch* é desabilitada.

Figura 23 – Passagem do controle do *OpenVSwitch* S1 para Controlador RunOS



Fonte: Autoria Própria

Durante esta primeira execução foram guardados *logs* gerados pelo *OpenVSwitch* para verificação da mudança de controlador. No *log* que antecede o ataque, na Figura 24, verifica-se que os *switches* estão devidamente conectados aos controladores RunOS e Ryu a partir do IP apontado (campo *is_connected* com valor *true*). Já na Figura 25, verifica-se a conexão estabelecida de ambos *switches* com o controlador RunOS após orquestração.

Figura 24 – Conectividade dos *switches* antes do balanceamento do plano de controle

```
7e7ed981-1cf5-432f-b216-663b37eca362
    Bridge "s2"
        Controller "tcp:192.168.100.2:6653"
            is_connected: true
        fail_mode: secure
        Port "s2-eth1"
            Interface "s2-eth1"
        Port "s2-eth2"
            Interface "s2-eth2"
        Port "s2"
            Interface "s2"
                type: internal
        Port "s2-eth3"
            Interface "s2-eth3"
    Bridge "s1"
        Controller "tcp:192.168.100.1:5555"
            is_connected: true
        fail_mode: secure
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth3"
            Interface "s1-eth3"
    ovs_version: "2.0.2"
```

Fonte: Autoria Própria

Figura 25 – Conectividade dos *switches* depois do balanceamento do plano de controle

```
7e7ed981-1cf5-432f-b216-663b37eca362
    Bridge "s2"
        Controller "tcp:192.168.100.2:6653"
            is_connected: true
        fail_mode: secure
        Port "s2-eth1"
            Interface "s2-eth1"
        Port "s2-eth2"
            Interface "s2-eth2"
        Port "s2"
            Interface "s2"
                type: internal
        Port "s2-eth3"
            Interface "s2-eth3"
    Bridge "s1"
        Controller "tcp:192.168.100.2:6653"
            is_connected: true
        fail_mode: secure
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth3"
            Interface "s1-eth3"
    ovs_version: "2.0.2"
```

Fonte: Autoria Própria

Para confirmar que os endereços de todos os *hosts* foram anonimizados utilizou-se a ferramenta *Curl* para fazer requisições à API REST do RunOS. O *log* mostrado pela Figura 26, retorna os 4 IP's anonimizados da rede.

Figura 26 – *Log* comprovando anonimização de todos os *hosts* pelo Controlador RunOS

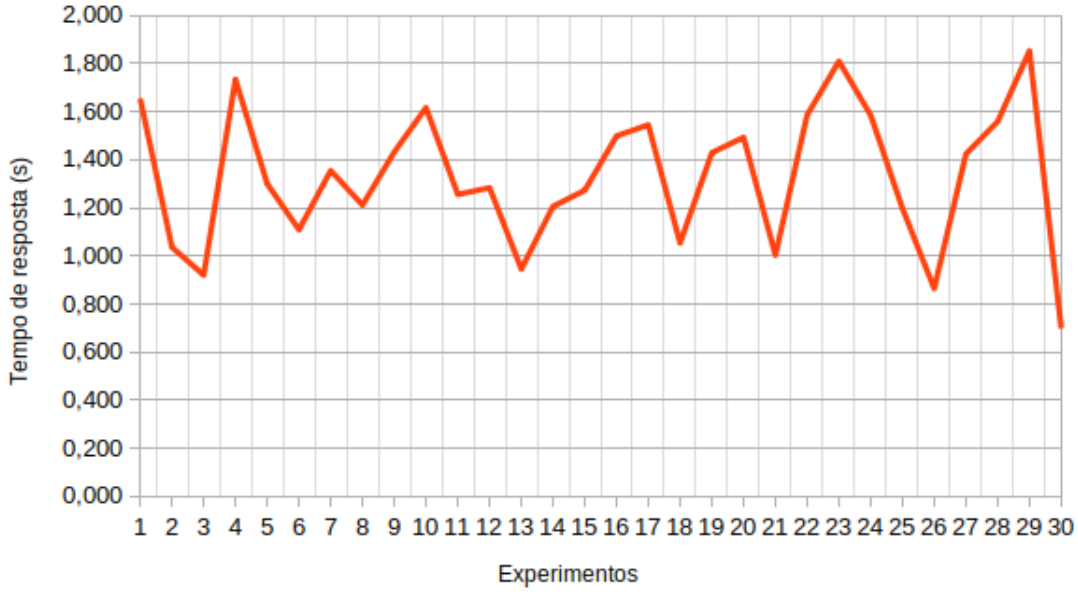

```
'{\"00:00:00:00:00:01\": {\"ID\": \"0000000000001654\", \"ip\": \"196.178.114.114\", \"mac\": \"00:0
0:00:00:00:01\", \"switch_id\": \"0000000000000002\", \"switch_port\": 1}, \"00:00:00:00:00:02\": {\
'ID\": \"0000000000001164\", \"ip\": \"196.178.114.222\", \"mac\": \"00:00:00:00:00:02\", \"switch_i
d\": \"0000000000000002\", \"switch_port\": 1}, \"00:00:00:00:00:03\": {\"ID\": \"0000000000001444\"
, \"ip\": \"196.178.114.105\", \"mac\": \"00:00:00:00:00:03\", \"switch_id\": \"0000000000000002\",
\"switch_port\": 2}, \"00:00:00:00:00:04\": {\"ID\": \"0000000000001634\", \"ip\": \"196.178.114.203
\", \"mac\": \"00:00:00:00:00:04\", \"switch_id\": \"0000000000000002\", \"switch_port\": 3}, \"7a:6
```


Fonte: Autoria Própria

Para verificar o tempo de resposta desta avaliação experimental utilizou-se um módulo de *Python* chamado *Timeit*. Esse módulo é muito utilizado devido à sua precisão temporal e pelo fato de desabilitar o coletor de lixo da memória para não interferir nos resultados. Após a obtenção dos resultados para 30 testes, foi possível gerar o gráfico ilustrado na Figura 27.

A partir dos valores obtidos calculou-se que o tempo médio de resposta do mecanismo

Figura 27 – Tempo de reação do orquestrador da análise até balanceamento do plano de controle



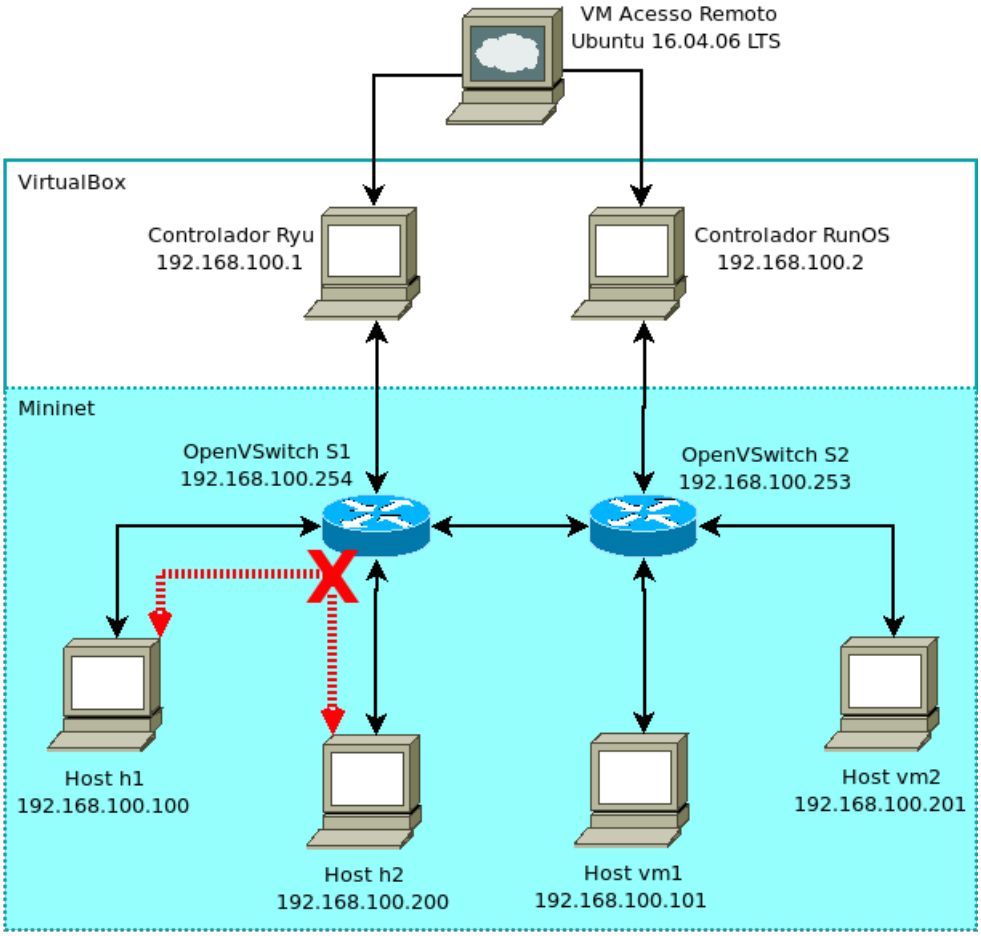
Fonte: Autoria Própria

é de 1,331s para um intervalo de confiança de 95% entre [1,226 : 1,435]. Apesar das variações obtidas, o tempo de reação é considerado rápido para uma situação de iminência de ataque demonstrando a viabilidade de usar o orquestrador ORCHSEC para cooperação entre controladores.

5.5 Avaliação Experimental com fluxo bloqueado

Durante esta segunda execução, representada na Figura 28, o *PortScan* entre os *hosts* foi considerado de nível intratável e por isso o fluxo será bloqueado. Foram guardados *logs* gerados pelo Ryu para verificação da ordem de bloqueio de fluxo e gerados pelos *OpenVSwitchs* para verificação da manutenção da conectividade da topologia após orquestração.

Figura 28 – Bloqueio do fluxo entre os *hosts* h1 e h2



Fonte: Autoria Própria

No *log* que antecede o ataque, na Figura 29, verifica-se que os *switches* estão devidamente conectados aos controladores RunOS e Ryu a partir do IP apontado (campo *is_connected* com valor *true*).

Figura 29 – Conectividade dos *switches* antes de bloqueio do fluxo

```
7e7ed981-1cf5-432f-b216-663b37eca362
    Bridge "s2"
        Controller "tcp:192.168.100.2:6653"
            is_connected: true
        fail_mode: secure
        Port "s2-eth1"
            Interface "s2-eth1"
        Port "s2-eth2"
            Interface "s2-eth2"
        Port "s2"
            Interface "s2"
                type: internal
        Port "s2-eth3"
            Interface "s2-eth3"
    Bridge "s1"
        Controller "tcp:192.168.100.1:5555"
            is_connected: true
        fail_mode: secure
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth3"
            Interface "s1-eth3"
    ovs_version: "2.0.2"
```

Fonte: Autoria Própria

A mesma situação percebe-se na Figura 30, comprovando que após orquestração a topologia foi mantida.

Figura 30 – Conectividade dos *switches* depois de bloqueio do fluxo

```
7e7ed981-1cf5-432f-b216-663b37eca362
    Bridge "s2"
        Controller "tcp:192.168.100.2:6653"
            is_connected: true
        fail_mode: secure
        Port "s2-eth1"
            Interface "s2-eth1"
        Port "s2-eth2"
            Interface "s2-eth2"
        Port "s2"
            Interface "s2"
                type: internal
        Port "s2-eth3"
            Interface "s2-eth3"
    Bridge "s1"
        Controller "tcp:192.168.100.1:5555"
            is_connected: true
        fail_mode: secure
        Port "s1-eth2"
            Interface "s1-eth2"
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth3"
            Interface "s1-eth3"
    ovs_version: "2.0.2"
```

Fonte: Autoria Própria

A confirmação do bloqueio do fluxo consta na Figura 31, mostrando o *log* da execução do controlador Ryu no momento em que é enviada ordem de bloqueio de fluxo.

Figura 31 – *Log* comprovando o bloqueio do fluxo

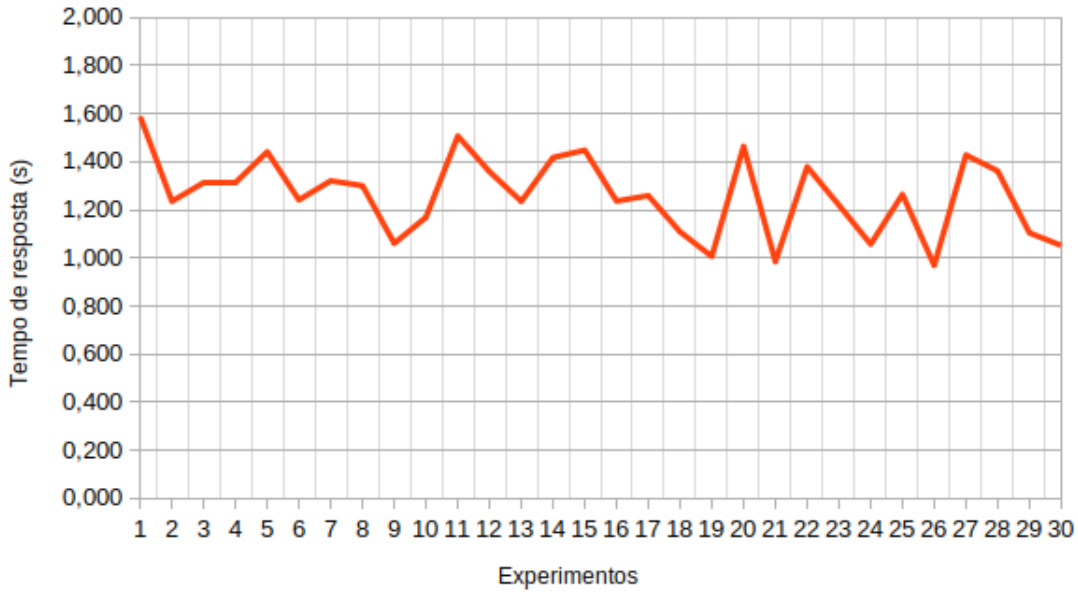
```
EVENT ofp_event->SimpleSwitchRest13 EventOFPPacketIn
packet in 1 e6:23:6c:8d:9e:49 33:33:00:00:00:16 3
EVENT ofp_event->SimpleSwitchRest13 EventOFPPacketIn
packet in 1 1a:b4:d8:09:08:d9 33:33:00:00:00:02 3
(1859) accepted ('192.168.100.2', 37060)
192.168.100.2 - - [05/Aug/2020 23:14:441 "GET /stats/flowentry/delete HTTP/1.1" 404 278 0.002967
EVENT ofp_event->SimpleSwitchRest13 EventOFPPacketIn
packet in 1 00:00:00:00:00:01 33:33:00:00:00:02 1
EVENT ofp_event->SimpleSwitchRest13 EventOFPPacketIn
packet in 1 00:00:00:00:00:03 33:33:00:00:00:02 3
EVENT ofp_event->SimpleSwitchRest13 EventOFPPacketIn
packet in 1 00:00:00:00:00:02 33:33:00:00:00:02 2
```

Fonte: Autoria Própria

Para verificar o tempo de resposta desta avaliação experimental de uso utilizou-se o

mesmo módulo *Python* (*Timeit*), e após a obtenção dos resultados para 30 testes, foi possível gerar o gráfico ilustrado na Figura 32.

Figura 32 – Tempo de reação do orquestrador da análise até execução do bloqueio do fluxo



Fonte: Autoria Própria

A partir dos valores obtidos calculou-se que o tempo médio de resposta do mecanismo é de 1,261s para um intervalo de confiança de 95% entre [1,202 : 1,320]. Apesar das variações obtidas, o tempo de reação novamente foi considerado rápido para uma situação de iminência de ataque, novamente demonstrando a viabilidade de usar o orquestrador para cooperação entre controladores.

6
Conclusão

Para finalizar este trabalho este capítulo aborda as considerações finais que achamos importantes sobre a pesquisa aqui concluída, relata as principais contribuições da nossa proposta e lista possibilidades de trabalhos futuros para dar continuidade à exploração do tema.

6.1 Considerações Finais

A Orquestração de Serviços é um desafio em aberto essencial para o cenário das futuras redes. E ao fornecer novos serviços, evoluindo as atuais infraestruturas, as SDN trouxeram grande flexibilização para seu gerenciamento, através do controlador e devido a separação dos planos de controle e de dados. O paradigma tornou programável a lógica da rede e facilitou a implantação de soluções que antes eram dificultadas devido ao engessamento provocado pelos softwares proprietários dos dispositivos físicos. Entretanto, essas mesmas vantagens também são motivos de preocupação para uma arquitetura de rede ainda emergente, principalmente quanto às questões de segurança e escalabilidade.

A segurança de uma SDN, por não possuir uma solução inerente à sua arquitetura, deve ser provida por serviços externos ou por adaptações na arquitetura da rede. Dessa forma, para solucionar problemas relacionados à escalabilidade e segurança, surgiram as arquiteturas SDN com múltiplos controladores, podendo ser estes homogêneos (caso padrão já explorado pela literatura) ou heterogêneos (cenário desafiador foco desta pesquisa). Para ambas as possibilidades o plano de dados pode ser dividido (domínio administrativo múltiplo) ou não (onde orquestrar é vital) . Nesse cenário, este trabalho teve como objetivo criar uma arquitetura de orquestração dos serviços SDN para controladores heterogêneos trabalhando numa rede de domínio administrativo único.

Em relação aos experimentos, é importante destacar que na primeira avaliação experimental, apesar de a anonimização ter sido realizada com sucesso, o invasor ainda pode realizar

ataques de força bruta para descobrir os endereços. Entretanto, dado o número de combinações possíveis para os IPs da rede, um tempo muito maior será gasto para essa descoberta, como comprovado pelo trabalho de Bomfim (2017), o qual mostrou que o atacante demoraria 12h para invadir a rede devido à anomização dos IPs. Nesse contexto, de acordo com as políticas gerenciais da rede ainda é possível adaptar o orquestrador para outras ações de prevenção e/ou mitigação de forma cooperativa entre os controladores, como corte, redirecionamento de fluxos ou outras políticas definidas pelos administradores da rede. Já na segunda avaliação experimental, o bloqueio do fluxo foi a ação concebida no trabalho de Menezes (2018), e pode ser considerada bruta como única saída ofertada. Porém, trabalhos futuros podem expandir o tratamento de ameaça realizado pelo MAdPE-K/SDN para que, integrado ao nosso Orquestrador, possamos servir um conjunto de serviços de segurança ainda mais eficaz.

Para prova de conceito e dentro das possibilidades deste trabalho, todos os experimentos e ferramentas foram implantados em ambientes virtualizados, o que impactou consideravelmente nos tempos de execução obtidos nos testes. Porém, ficou demonstrada a viabilidade teórica da arquitetura de orquestração proposta, pelo cumprimento da hipótese. Observarmos isso no comportamento de auto-configuração (na avaliação experimental com balanceamento de plano de controle, onde o orquestrador redefiniu a configuração do *switch* trocando seu controlador) e no de auto-proteção (na avaliação experimental com fluxo bloqueado, onde o orquestrador reagiu mediante ameaça na rede).

A continuação do trabalho focará em tornar nossa solução prática, testada em ambiente físico e medida com parâmetros reais.

6.2 Contribuições

Os trabalhos de Bomfim (2017) e Menezes (2018) foram escolhidos e estudados para que seus serviços de segurança SDN pudessem ser utilizados de forma integrada. O primeiro criou um algoritmo de anonimização, chamado BomIP, que anonimiza endereços IP na rede de forma fixa, para retardar o tempo de invasão aos hosts. Já o segundo, sugeriu uma nova arquitetura de rede chamada MAdPE-K/SDN, como solução para uma rede SDN autonômica, baseada no algoritmo determinístico de células dendríticas.

Esta pesquisa explorou a orquestração dos serviços MAdPE-K/SDN e BomIP numa rede de domínio administrativo único, e demonstrou ser possível desenvolver soluções mais sofisticadas para diversos controladores em uma mesma rede, que utilizem a arquitetura de orquestração e forneçam seus serviços de segurança de forma cooperativa em um tempo hábil mediante a previsão de ataque.

O trabalho aqui realizado elaborou uma definição de Orquestração que acolhe os vários trabalhos recentemente realizados em cima deste tema; propôs uma Arquitetura que ataca a ocorrência dele numa nova tecnologia de redes (SDN); encontrou serviços implantados prontos

para explorar; construiu com ferramentas de fácil disposição na indústria e na academia um ambiente de experimentação; resgatou com novas perspectivas pesquisas em SDN anteriormente realizadas ao mesmo tempo que preparou o caminho para continuidade e expansão desta em Orquestração de serviços que foi aqui iniciada.

Assim sendo, chegamos ao final com o desafio superado de atacar um tema inovador, com hipótese testada, com resultados já promissores medidos em ambiente experimental, apontando que nossa solução não somente tem viabilidade teórica, mas está pronta para ser aplicada em ambiente real.

A pesquisa realizada gerou um artigo submetido em periódico internacional, cuja submissão em sua plataforma *online* foi confirmada conforme o Anexo A.

6.3 Trabalhos Futuros

Para este trabalho, os controladores foram dispostos na mesma rede, contudo não foi considerado uma situação onde um deles falha, ou ainda, a possibilidade de utilização de mais de um controlador gerenciando o mesmo *switch*. Dessa forma, sugere-se novos trabalhos futuros que se concentrem nas características de redundância e tolerância à falhas. Considerando a possibilidade de termos mais de uma instância de cada controlador na arquitetura do experimento, o orquestrador ORCHSEC poderia tranquilamente substituir uma instância do Ryu que falhasse (ou do RunOS) por outra, através de uma extensão simples da implementação do balanceamento do plano de controle.

Dentro do ambiente do ELAN são realizadas ainda outras pesquisas em SDN cuja integração com esta realizada seria de grande valia. Está sendo trabalhado o serviço MAdPE-K/SDN em ambiente físico, com os *switches* do fabricante *HP* disponíveis no laboratório para exploração do protocolo *OpenFlow*. E já foi explorada a implementação em circuito *NetFPGA* do serviço BomIP. A integração destas pesquisas com esta aqui realizada com certeza produzirão mais resultados e publicações, tornando a submissão aqui relatada no periódico apenas o pontapé inicial de frutos de pesquisas futuros.

Referências

AKHUNZADA, A. et al. Securing software defined networks: Taxonomy, requirements, and open issues. *IEEE Communications Magazine*, v. 54, 2015. Citado na página 38.

ALSMADI, I.; XU, D. Security of software defined networks: A survey. *Computers & Security*, v. 53, p. 79 – 108, 2015. ISSN 0167-4048. Disponível em: <http://www.sciencedirect.com/science/article/pii/S016740481500070X>. Citado na página 43.

ANANDITA, S. et al. Implementation of dendritic cell algorithm as an anomaly detection method for port scanning attack. In: . [S.l.: s.n.], 2015. p. 1–6. Citado na página 63.

BEHRINGER, M. et al. *Autonomic Networking: Definitions and Design Goals*. [S.l.], 2015. Citado na página 44.

BOMFIM, L. H. da S. *Um Serviço para Anonimização em Redes Definidas por Software*. Dissertação (Mestrado) — Universidade Federal de Sergipe, Centro de Ciências Exatas e Tecnologia, Programa de Pós-Graduação em Ciência da Computação, 2017. Citado 4 vezes nas páginas 57, 60, 62 e 77.

BOMFIM, L. H. da S. et al. A systematic mapping about anonymization services for software-defined networking. *IEEE Latin America Transactions*, v. 15, n. 6, p. 1113–1120, June 2017. ISSN 1548-0992. Citado 3 vezes nas páginas 44, 60 e 61.

BOSCHI, E.; TRAMMELL, B. *IP Flow Anonymization Support*. 2011. RFC 6235. (Request for Comments, 6235). Disponível em: <https://tools.ietf.org/html/rfc6235>. Citado na página 43.

BREKNE, T. o. n.; ÅRNES, A.; ØSLEBØ, A. Anonymization of ip traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In: *Proceedings of the 5th International Conference on Privacy Enhancing Technologies*. Berlin, Heidelberg: Springer-Verlag, 2006. (PET'05), p. 179–196. ISBN 3-540-34745-3, 978-3-540-34745-3. Disponível em: <http://dx.doi.org/10.1007/11767831_12>. Citado na página 44.

CASELLAS, R. et al. Control and orchestration of multidomain optical networks with gmpls as inter-sdn controller communication [invited]. *J. Opt. Commun. Netw.*, OSA, v. 7, n. 11, p. B46–B54, Nov 2015. Disponível em: <http://jocn.osa.org/abstract.cfm?URI=jocn-7-11-B46>. Citado 8 vezes nas páginas 21, 30, 33, 34, 35, 47, 53 e 54.

CELDRáN, A. H. et al. Automatic monitoring management for 5g mobile networks. *Procedia Computer Science*, v. 110, p. 328 – 335, 2017. ISSN 1877-0509. 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050917312802>. Citado 2 vezes nas páginas 30 e 34.

FARAH, T.; TRAJKOVIć, L. Anonym: A tool for anonymization of the internet traffic. In: *2013 IEEE International Conference on Cybernetics (CYBCO)*. [S.l.: s.n.], 2013. p. 261–266. Citado na página 44.

FRATE, M. *OrchFlow – Uma arquitetura para orquestração de redes OpenFlow com múltiplos controladores*. Dissertação (Mestrado) — Universidade Federal de São Carlos, Centro de Ciências em Gestão e Tecnologia, Programa de Pós-Graduação em Ciência da Computação, 2017. Citado 2 vezes nas páginas 24 e 51.

FRATE, M.; MARCZUK, M. K.; VERDI, F. L. Orchflow: An architecture for orchestration of multiple controllers in openflow networks. *Journal of Network and Systems Management*, v. 27, n. 3, p. 551–572, Jul 2019. ISSN 1573-7705. Disponível em: <https://doi.org/10.1007/s10922-018-9476-x>. Citado 9 vezes nas páginas 22, 30, 33, 34, 36, 51, 52, 53 e 54.

GREENSMITH, J.; AICKELIN, U. Dendritic cells for syn scan detection. In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: Association for Computing Machinery, 2007. (GECCO '07), p. 49–56. ISBN 9781595936974. Disponível em: <https://doi.org/10.1145/1276958.1276966>. Citado 2 vezes nas páginas 45 e 46.

GREENSMITH, J.; AICKELIN, U. The deterministic dendritic cell algorithm. In: BENTLEY, P. J.; LEE, D.; JUNG, S. (Ed.). *Artificial Immune Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 291–302. ISBN 978-3-540-85072-4. Citado na página 46.

GREENSMITH, J.; AICKELIN, U.; TWYCROSS, J. Articulation and clarification of the dendritic cell algorithm. *CoRR*, abs/0910.4903, 2009. Disponível em: <http://arxiv.org/abs/0910.4903>. Citado na página 46.

HORN, P. *Autonomic Computing: IBM's Perspective on the State of Information Technology*. [S.l.], 2001. Citado na página 44.

HYOJOON, K.; FEAMSTER, N. Openflow: Enabling innovation in campus networks. *IEEE Communications Magazine*, Georgia, USA, 2013. ISSN 0163-6804. Citado na página 20.

INTERNATIONAL TELECOMMUNICATIONS UNION. *Recommendation ITU-T X.509 (2016)*: Information technology –open systems interconnection –the directory: Public-key and attribute certificate frameworks. Genebra, 2016. 254 p. Disponível em: <https://www.itu.int/rec/T-REC-X.509-201610-S>. Citado na página 41.

JAMMAL, M. et al. Software defined networking: State of the art and research challenges. *Computer Networks*, v. 72, p. 74 – 98, 2014. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128614002588>. Citado na página 40.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 36, n. 1, p. 41–50, jan. 2003. ISSN 0018-9162. Disponível em: <http://dx.doi.org/10.1109/MC.2003.1160055>. Citado na página 45.

KOTRONIS, V. et al. Control exchange points: Providing qos-enabled end-to-end services via sdn-based inter-domain routing orchestration. *CoRR*, abs/1611.02628, 2016. Disponível em: <https://www.researchgate.net/publication/264435440_Control_Exchange_Points_Providing_QoS-enabled_End-to-End_Services_via_SDN-based_Inter-domain_Routing_Orchestration>. Citado 2 vezes nas páginas 21 e 30.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, v. 103, p. 14–76, 2014. Citado na página 42.

KUROSE, J.; ROSS, K. *Computer Networking: A Top-down Approach*. Addison-Wesley, 2010. (Pearson International edition). ISBN 9780136079675. Disponível em: <https://books.google.com.br/books?id=gxLePQAACAAJ>. Citado na página 41.

LYU, X. et al. Multi-timescale decentralized online orchestration of software-defined networks. *IEEE Journal on Selected Areas in Communications*, v. 36, n. 12, p. 2716–2730, Dec 2018. ISSN 0733-8716. Citado 3 vezes nas páginas 24, 30 e 34.

MATZINGER, P. Tolerance, danger, and the extended family. *Annual Review of Immunology*, v. 12, n. 1, p. 991–1045, 1994. PMID: 8011301. Disponível em: <https://doi.org/10.1146/annurev.iy.12.040194.005015>. Citado na página 45.

MAYORAL, A. et al. Sdn orchestration architectures and their integration with cloud computing applications. *Optical Switching and Networking*, v. 26, p. 2 – 13, 2017. ISSN 1573-4277. Advances on Path Computation Element. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1573427716000047>. Citado 9 vezes nas páginas 30, 33, 34, 36, 49, 50, 51, 53 e 54.

MENEZES, P. M. *Autonomicidade em uma rede definida por software utilizando teoria do perigo*. Dissertação (Mestrado) — Universidade Federal de Sergipe, Centro de Ciências Exatas e Tecnologia, Programa de Pós-Graduação em Ciência da Computação, 2018. Citado 4 vezes nas páginas 57, 58, 62 e 77.

MONGE, M. A. S.; VIDAL, J. M.; VILLALBA, L. J. G. A novel self-organizing network solution towards crypto-ransomware mitigation. In: *Proceedings of the 13th International Conference on Availability, Reliability and Security*. New York, NY, USA: ACM, 2018. (ARES 2018), p. 48:1–48:10. ISBN 978-1-4503-6448-5. Disponível em: <http://doi.acm.org/10.1145/3230833.3233249>. Citado 2 vezes nas páginas 30 e 34.

NEVES, P. et al. The selfnet approach for autonomic management in an nfv/sdn networking paradigm. *International Journal of Distributed Sensor Networks*, v. 2016, 2016. Cited By 20. Disponível em: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84959280720&doi=10.1155%2f2016%2f2897479&partnerID=40&md5=61debad0347a9dcf17d2212cee078e58>. Citado 10 vezes nas páginas 21, 22, 30, 34, 36, 47, 48, 49, 53 e 54.

OLIVEIRA, D. D. de; SALGUEIRO, R. J. P. d. B.; MORENO, E. D. Predizendo o perigo: Alerta antecipado contra a propagação de wormsutilizandoo algoritmo das células dendríticas. SBRC - XVIII Workshop de Gerência e Operação de Redes e Serviços, 2013. Disponível em: <http://sbrc2013.unb.br/files/anais/wgrs/artigos/artigo-8.pd>. Citado na página 57.

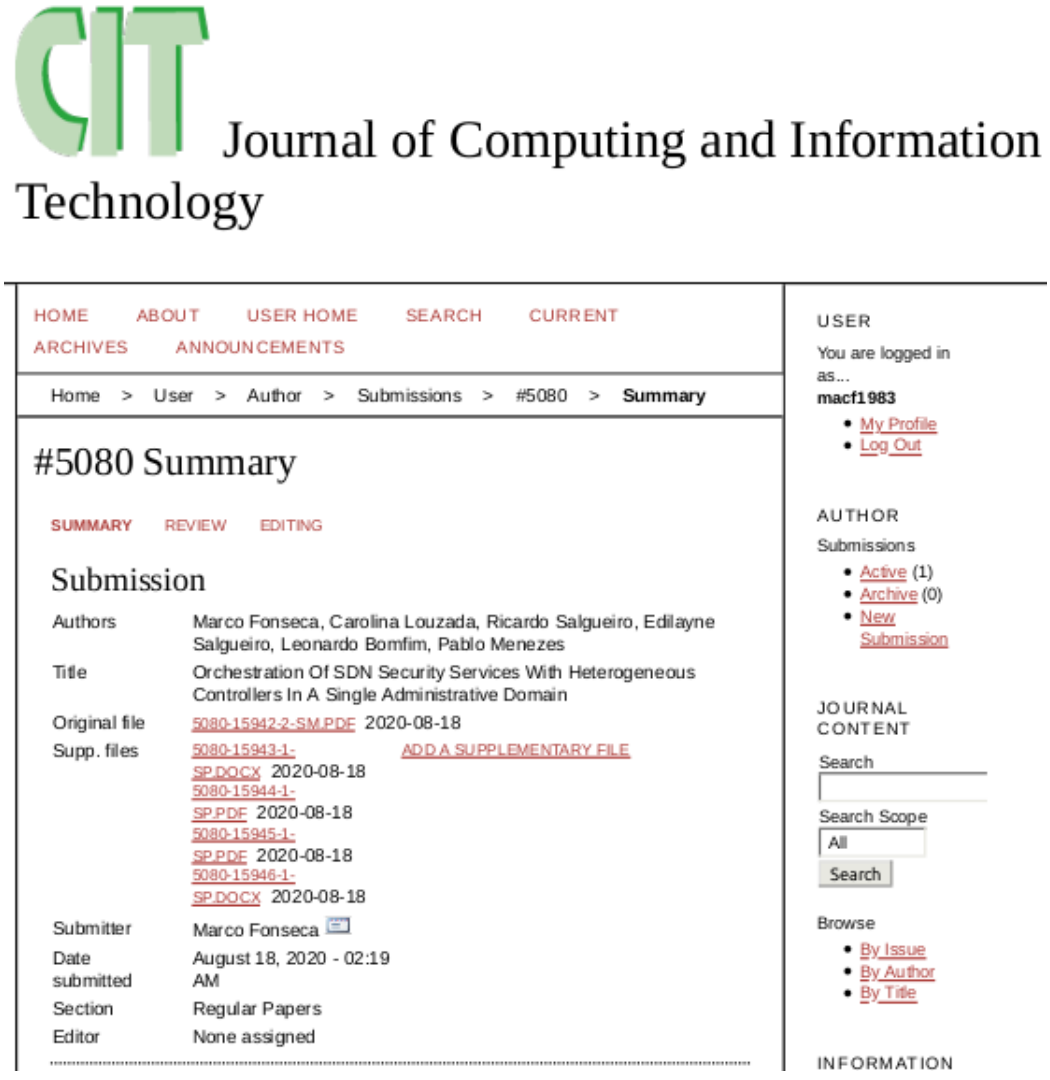
ROTSOS, C. et al. Network service orchestration standardization: A technology survey. *Computer Standards & Interfaces*, v. 54, p. 203 – 215, 2017. ISSN 0920-5489. SI: Standardization SDN&NFV. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0920548916302458>. Citado na página 21.

ROTSOS, C. et al. Network service orchestration standardization: A technology survey. *Computer Standards & Interfaces*, v. 54, p. 203 – 215, 2017. ISSN 0920-5489. SI: Standardization SDN&NFV. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0920548916302458>. Citado na página 35.

SHIREY, D. R. *Internet Security Glossary*. RFC Editor, 2000. RFC 2828. (Request for Comments, 2828). Disponível em: <https://rfc-editor.org/rfc/rfc2828.txt>. Citado 2 vezes nas páginas 42 e 43.

TAYYABA, S. K. et al. Software defined network (sdn) based internet of things (iot): A road ahead. In: *Proceedings of the International Conference on Future Networks and Distributed Systems*. New York, NY, USA: ACM, 2017. (ICFNDS '17), p. 15:1–15:8. ISBN 978-1-4503-4844-7. Disponível em: <http://doi.acm.org/10.1145/3102304.3102319>. Citado 3 vezes nas páginas 20, 30 e 34.

ANEXO A – Submissão na plataforma do periódico



Fonte: Autoria Própria

Vemos que o artigo propriamente dito está listado no campo "*Original file*" como arquivo PDF, que segundo procedimento de submissão do periódico deve ter excluídas as informações dos autores, suas filiações e quaisquer referências que permitam a identificação dos mesmos. Já no campo "*Supp. files*" constam o Artigo completo (com todas as informações dos autores, em arquivo DOCX), o Formulário *COPYRIGHT TRANSFER AGREEMENT* (assinado pelo autor, concordando com a publicação pelo periódico e reuso do material nela contido), o Formulário *CONTRIBUTIONS FORM* (assinado pelo autor e por todos os coautores, onde é descrita a contribuição original trazida pela submissão) e as Referências do artigo em formato DOCX;