UNIVERSIDADE KIMPA VITA

08 - 05 | 2025

Protecção do controlador de uma rede definida por Software

Controller protection of a Software-defined network

Adão António Bagi

Versão eletrónica
URL: https://ciencia.unikivi.ao

Data de publicação: 08-05-2025. Páginas: 01-12

Editor
Revista Científica Interdisciplinar da UNIKIVI

Referência eletrónica
Bagi, A. A. (2025). Protecção do controlador de uma rede definida por Software. Revista da UNIKIVI. 01(01), 01-12.

Universidade Kimpa Vita
Bairro Popular n°1, rua Henriques Fretas
CP 77, Cidade do Uíge
Angola

Revista Científica Interdisciplinar da UNIKIVI – Universidade Kimpa Vita,
Vol. 01, N.º 01, 2025, pp. 12, Uíge
https://ciencia.unikivi.ao/

PROTECÇÃO DO CONTROLADOR DE UMA REDE DEFINIDA POR SOFTWARE

Controller protection of a software-defined network

Adão António Bagi

Universidade Gregório Semedo, Angola

adsubagi@gmail.com | ORCID: 0009-0007-8353-4591

RESUMO

Software Defined Network (SDN), trazem um novo paradigma de redes, permitindo maior flexibilidade e automação ao separar o plano de controlo do plano de dados. Contudo, a centralização em um único controlador apresenta riscos críticos, como a vulnerabilidade a falhas que podem comprometer a continuidade dos serviços. Este estudo investiga estratégias para mitigar esses riscos, destacando a implementação de múltiplos controladores como solução para aumentar a resiliência da rede. O objectivo principal foi testar a resiliência de uma rede SDN com dois controladores POX e um balanceador *Zookeeper*, avaliando a capacidade da rede de operar mesmo diante de falhas de controladores. A pesquisa iniciou com uma revisão bibliográfica para compreensão profunda da arquitetura SDN, focando nas camadas de aplicação, controlo e infraestrutura. A metodologia incluiu a montagem de uma infraestrutura de testes utilizando o Mininet para simulação de redes e o POX como controlador, coordenados pelo *Zookeeper* para assegurar alta disponibilidade. Os resultados mostraram que a rede manteve uma disponibilidade de (95,04 %) durante os testes de falha do controlador primário ao passo que em uma arquitetura SDN centralizada sem redundância, a falha no controlador central pode resultar em indisponibilidade total da rede até que o problema seja resolvido. Esses dados confirmam a eficácia do modelo distribuído para mitigar os riscos associados à centralização, garantindo uma operação contínua mesmo em cenários de falha. A reflexão sobre os resultados destaca a importância de estruturas distribuídas em SDNs para assegurar a resiliência e continuidade de serviços. As conclusões evidenciam o impacto deste estudo para o avanço do conhecimento tecnológico, propondo soluções práticas para problemas críticos. Além disso, o estudo contribui para a sociedade ao promover redes mais seguras e confiáveis, fundamentais em um mundo cada vez mais digital e interconectado.

Palavras-chave: Redes definidas por software, resiliência de rede, controladores SDN, *Zookeeper*, Mininet.

ABSTRACT

Software Defined Network (SDN) brings a new paradigm for networks, allowing greater flexibility and automation by separating the control plane from the data plane. However, centralization in a single controller presents critical risks, such as vulnerability to failures that can compromise service

continuity. This study investigates strategies to mitigate these risks, highlighting the implementation of multiple controllers as a solution to increase network resilience. The main objective was to test the resilience of an SDN network with two POX controllers and a Zookeeper balancer, evaluating the network's ability to operate even in the face of controller failures. The research began with a literature review to gain a deep understanding of the SDN architecture, focusing on the application, control and infrastructure layers. The methodology included the assembly of a test infrastructure using Mininet for network simulation and POX as a controller, coordinated by Zookeeper to ensure high availability. The results showed that the network maintained (95.04 %) availability during the primary controller failure tests, whereas in a centralized SDN architecture without redundancy, a central controller failure can result in total network unavailability until the problem is resolved. These data confirm the effectiveness of the distributed model to mitigate the risks associated with centralization, ensuring continuous operation even in failure scenarios. The reflection on the results highlights the importance of distributed structures in SDNs to ensure resilience and service continuity. The conclusions highlight the impact of this study on the advancement of technological knowledge, proposing practical solutions to critical problems. In addition, the study contributes to society by promoting safer and more reliable networks, which are essential in an increasingly digital and interconnected world.

Keywords: Software defined networks, network resilience, SDN controllers, Zookeeper, Mininet.

1 INTRODUÇÃO

As Redes Definidas por *Software* (SDN) surgem como um paradigma arquitectónico inovador que está reformulando a maneira como as redes de computadores são gerenciadas e operadas. Segundo (McKeown et al., 2008), esse modelo oferece uma flexibilidade inigualável, automação avançada e gerenciamento simplificado, elementos essenciais em um ambiente de que exige cada vez mais agilidade e eficiência. A relevância das SDN na actualidade está ligada à crescente demanda por redes mais dinâmicas e adaptáveis. Em um cenário onde a virtualização e a nuvem se tornaram normais, elas permitem que as organizações centralizem a inteligência da rede em um controlador, separando o plano de controlo do plano de dados. Conforme descrito pela *Open Networking* Foundation (ONF, 2016), essa separação possibilita que o controlo da rede seja programável directamente enquanto encaminhamento de dados pode ser ajustado de forma dinâmica. Esta abordagem cria uma abstração de rede mais limpa e simplificada, facilitando a inovação e proporcionando maior flexibilidade operacional (McKeown et al., 2008). Contudo, apesar das vantagens, a arquitetura SDN também apresenta desafios significativos, especialmente no que se refere à dependência de um ponto centralizado de controlo. A falha do controlador pode impactar toda a rede, causando interrupções consideráveis no serviço. A segurança e a resiliência do controlador SDN são, portanto, críticas para garantir a confiabilidade e a integridade da rede. (Alharbi et al., 2017) enfatizou a necessidade de proteger esses pontos centrais para assegurar a continuidade das operações de rede. Dessa contextualização, este estudo justifica-se pela necessidade de compreender melhor as capacidades e limitações da SDN em cenários reais com o intuito de oferecer insights valiosos para a sua implementação em ambientes de produção. O objectivo do estudo é avaliar a resiliência de redes definidas por software por meio de uma arquitectura com dois controladores POX e o *Zookeeper* como mecanismo de balanceamento de carga. Para testar a solução, foi Implementado um ambiente experimental para simular o funcionamento da SDN, com o Mininet, dois controladores POX e *Zookeeper* como monitor dos controladores. Foi desligado o controlador principal supondo que sofreu um ataque ou outra acção que possa causar este estado de desligamento total para analisar o comportamento da rede e sua resiliência no contexto da infraestrutura com controlador redundante.

2 ENQUANDRAMENTO TEÓRICO

2.1 Redes Definidas por Sofwares

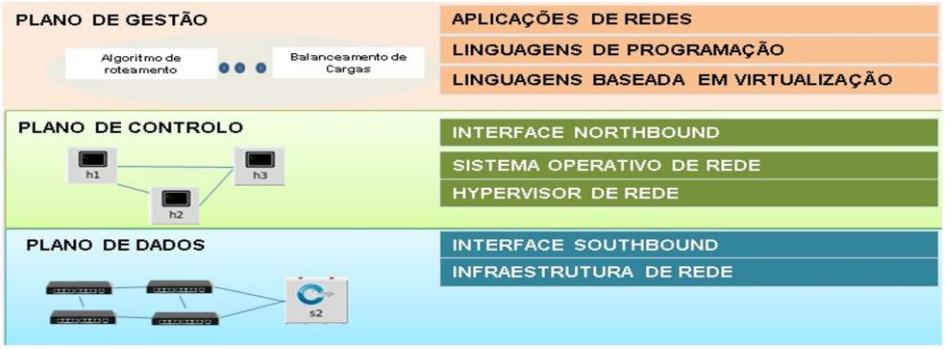
A evolução contínua das redes definidas por software (SDN) e da virtualização de funções de rede (NFV) está impulsionando inovações significativas no gerenciamento de recursos e na implantação de serviços em redes de próxima geração. Pesquisadores e desenvolvedores estão explorando activamente a aplicação conjunta dessas tecnologias para aprimorar o desempenho, a segurança

e a eficiência das redes (Nadeau & Gray, 2013). A convergência entre SDN e NFV tem sido reconhecida como uma estratégia poderosa para alcançar objectivos avançados de controlo e gerenciamento de redes. Quando aplicadas em conjunto, essas tecnologias oferecem soluções para desafios com o gerenciamento dinâmico de recursos e a orquestração inteligente de serviços (Farhady et al., 2015). Uma das principais vantagens dessa combinação é a capacidade de criar ambientes de serviço virtual dinâmicos, que podem ser rapidamente configurados e adaptados conforme as necessidades variáveis de tráfego e aplicativos (Chowdhury & Boutaba, 2010). Essa integração permite maior flexibilidade e agilidade na resposta às demandas da rede, promovendo uma adaptação eficiente às mudanças no cenário tecnológico e nas exigências dos usuários.

Entretanto, um dos desafios críticos da arquitetura SDN é a dependência de um controlador centralizado, cuja falha pode impactar toda a rede, causando interrupções consideráveis no serviço. Esse problema ressalta a importância de mecanismos de alta disponibilidade e resiliência, como a coordenação de múltiplos controladores por meio de soluções distribuídas, garante a continuidade operacional da rede mesmo diante de falhas inesperadas.

Figura 1

Infraestrutura de Redes definidas por Sotfware



Fonte: Adaptado de Kreutz et al. (2015)

Assim, para analisar o comportamento da infraestrutura proposta neste estudo, foi implementada uma topologia composta por dois controladores, sendo um activo e outro em modo de espera (backup). Com base nessa topologia, foi possível calcular a disponibilidade da rede (%), definida como a razão entre o tempo em que a rede permaneceu operacional durante os testes (horas) e o tempo total de observação (horas). De acordo com Avizienis et al. (2001), essa relação é expressa pela Equação 1:

$$\textbf{Disponibilidade} = \frac{\textbf{Tempo operacional}}{\textbf{Tempo total}} * \textbf{100} \qquad \textbf{Equação 1}$$

Adicionalmente, para mensurar o impacto de falhas e a eficácia do mecanismo de recuperação, foi avaliado o Tempo de Recuperação (RTO – *Recovery Time Objective*)**,** que representa o intervalo necessário para restaurar completamente a funcionalidade da rede após a interrupção causada pela falha do controlador primário e a posterior activação do controlador de backup. Segundo Sterbenz et al. (2010), o cálculo do tempo de recuperação (horas) é obtido por meio da diferença entre o tempo total de observação (horas) e o tempo em que a rede esteve efectivamente operacional (horas), conforme a Equação 2:

$$\textbf{Tempo de recupeação} = \textbf{Tempo total} - \textbf{Tempo operacional} \qquad \textbf{Equação 2}$$

2.1.1 Plano de dados

O plano de dados em redes SDN é composto por dispositivos físicos, como *switches* e roteadores programáveis, que formam a base da infraestrutura de rede. Esses dispositivos são responsáveis por encaminhar o tráfego de rede de acordo com as instruções recebidas do controlador SDN. Segundo Kreutz et al. (2015), as interfaces sul desempenham um papel crucial nesse plano, pois permitem a comunicação entre os dispositivos de rede físicos e o controlador SDN. Essa interacção é fundamental para que o controlador possa gerenciar e configurar a rede de maneira centralizada e dinâmica, garantindo a eficiência e a flexibilidade do sistema.

2.1.2 Plano de controlo

O plano de controlo em redes SDN actua como um intermediário entre o plano de dados e o plano de gestão, sendo responsável por gerenciar políticas de rede e tomar decisões sobre o roteamento do tráfego. Ele opera por meio de um sistema operacional de rede e hipervisores, que oferecem uma visão centralizada e abrangente da rede. As interfaces nortes desempenham um papel crucial, permitindo a comunicação do plano de controlo com as aplicações de rede e serviços localizados no plano de gestão (3GPP, 2015). Essa estrutura garante que o controlo da rede seja eficiente e coordenado, facilitando a gestão centralizada.

2.1.3 Plano de gestão

O plano de gestão em redes SDN engloba aplicações de rede e algoritmos responsáveis por definir as políticas e regras operacionais da rede. Esta camada utiliza linguagens de programação e técnicas de virtualização para implementar funcionalidades avançadas, como balanceamento de carga e algoritmos de roteamento, garantindo uma gestão mais eficiente e adaptável da infraestrutura de rede. Embora já exista um balanceador de carga no plano de gestão para distribuir o tráfego entre os dispositivos de rede, a presença do *Zookeeper* como coordenador dos controladores SDN desempenha um papel distinto na orquestração da alta disponibilidade dos controladores, garantindo que, em caso de falha do controlador primário, um *backup* assuma automaticamente, assegurando a continuidade operacional da rede. Segundo Amani et al., (2014), essa camada é crucial para a operação de redes modernas, pois possibilita a implementação de soluções personalizadas e dinâmicas, que podem ser ajustadas conforme as necessidades específicas do tráfego e das aplicações em uso.

2.2 Virtualização da rede

As redes de comunicação de próxima geração estão cada vez mais fundamentadas em infraestruturas virtualizadas, onde funções de rede são implementadas em máquinas virtuais, substituindo os equipamentos proprietários tradicionais. Essa transição busca eliminar a arquitetura baseada em caixas-pretas, cada uma com hardware especializado, em favor de uma nova estrutura composta por "caixas brancas" que executam uma variedade de softwares de rede especializados (Gkioulos et al., 2018). Nesse contexto, as redes definidas por software e a virtualização de funções de rede emergem como tecnologias essenciais para a implementação dessa transformação. Ambas oferecem benefícios significativos, como a redução de custos operacionais, melhor utilização de recursos e gerenciamento simplificado (Abbasi & Jin, 2018). A adopção dessas tecnologias está crescendo, impulsionada pela necessidade de aumentar a eficiência dos recursos de rede e reduzir os custos operacionais, objectivos cruciais na era do gerenciamento e controle de redes (Stamou et al., 2019).

A virtualização de rede permite a consolidação de hardware, reduzindo custos de aquisição, manutenção e energia associados aos dispositivos tradicionais. Além disso, melhora a escalabilidade, permitindo a adaptação dinâmica da capacidade da rede sem necessidade de investimentos antecipados em *hardware* específico (Chowdhury & Boutaba, 2010). Essa abordagem também promove a inovação contínua e a rápida introdução de novos serviços, já que a separação das funções de rede do hardware facilita actualizações e mudanças por meio de alterações no *software*, sem impactar a infraestrutura física. Isso aumenta a agilidade e adaptabilidade nas operações de rede, permitindo que as organizações respondam eficazmente às demandas em constante evolução (Chowdhury & Boutaba, 2010).

2.3 Máquina virtual

Uma máquina virtual (VM) funciona como uma camada de abstração entre o *hardware* físico e o usuário final, permitindo a execução de sistemas operacionais virtualizados em um ambiente controlado (Kreutz et al., 2015). Essas VMs, também conhecidas como servidores virtuais, compartilham recursos físicos do sistema host, como CPUs, memória, disco e E/S, criando ambientes isolados e independentes uns dos outros (Kreutz et al., 2015). O conceito de máquina real refere-se ao sistema operacional host e seus componentes físicos "bare metal", como memória, CPU, placa-mãe e interface de rede, que formam a infraestrutura subjacente (Goldberg, 1971).

As VMs são construídas sobre essa infraestrutura física, proporcionando a capacidade de executar múltiplos sistemas operacionais em uma única máquina real. Os hipervisores, ou monitores de máquina virtual (VMMs), desempenham um papel essencial nesse processo, gerenciando e controlando o acesso das VMs aos recursos físicos (Kreutz et al., 2015). Eles realizam as traduções necessárias para que o sistema operacional convidado acredite estar interagindo directamente com o hardware físico (Abbasi, 2019). Dessa forma, os hipervisores garantem que as VMs operem eficientemente, maximizando o uso dos recursos disponíveis e mantendo o isolamento entre os diferentes ambientes virtuais.

2.4 Mininet

Mininet é um emulador de rede que permite a criação de sistemas virtuais compostos por *hosts, switches*, controladores e conexões, utilizando a programação padrão do Linux. Seus switches são compatíveis com *OpenFlow*, uma tecnologia chave na arquitetura SDN, tornando-o uma ferramenta essencial para emulação de redes definidas por software. A conectividade com o sistema é facilmente estabelecida através de CLI e API, sendo que a API Python é uma das suas funcionalidades mais poderosas. Esta API oferece uma interface flexível e robusta para configuração de redes, permitindo a criação de experimentos, topologias e tipos de nós personalizados com poucas linhas de código. Isso facilita a definição de testes de regressão, execução de comandos em múltiplos nós e exibição dos resultados, como observado por Kaur et al., (2014).

Mininet é amplamente utilizado devido à sua rapidez na inicialização, suporte a topologias personalizadas e capacidade de emular redes reais. Ele é compatível com PCs, servidores e máquinas virtuais, facilitando o compartilhamento e recriação de cenários. A emulação no Mininet baseia-se na virtualização de processos, permitindo a criação de entidades dentro de um único kernel do sistema operacional que executa código real. Isso significa que um projecto desenvolvido no Mininet pode ser transferido para redes práticas com hardware real, sem grandes modificações, tornando-o uma ferramenta crucial para o desenvolvimento e teste de redes em grande escala, como destacado por Xia et al. (2015) e Kreutz et al. (2015).

2.5 Zookeeper

O *Zookeeper* é um serviço de coordenação distribuída e de código aberto, projectado para simplificar a gestão de aplicativos distribuídos. Ele fornece um conjunto de primitivas simples que ajudam na implementação de serviços de alto nível, como sincronização, manutenção de configuração, gestão de grupos e nomenclatura, tornando-se essencial em ambientes distribuídos. Seu modelo de dados é organizado em uma estrutura hierárquica de nós chamados *znodes*, que funcionam de forma semelhante a arquivos e directórios de sistemas de arquivos tradicionais. No entanto, diferentemente dos sistemas convencionais, os dados do *Zookeeper* são mantidos na memória, o que proporciona alta taxa de transferência e baixa latência. Cada *znode* pode armazenar dados e ter filhos, permitindo uma estrutura flexível para a coordenação de informações (Apache Zookeeper, 2020).

A robustez do *Zookeeper* é garantida por sua arquitetura replicada. Ele opera em um conjunto de hosts, conhecido como ensemble, onde cada servidor mantém uma cópia do estado em memória e logs de transacções em armazenamento persistente. Isso assegura que, enquanto a maioria dos servidores estiver disponível, o serviço *Zookeeper* continuará operacional. Os clientes conectam-se a um único servidor *Zookeeper*, mantendo uma conexão TCP que gerencia solicitações e respostas (Apache Zookeeper, 2020). O *Zookeeper* também suporta nós efémeros, que existem apenas enquanto a sessão que os criou está activa. Essa funcionalidade é útil para implementar comportamentos que dependem da actividade contínua de um serviço ou componente. A replicação do *Zookeeper* evita que ele se torne um ponto único de falha, enquanto sua ordenação estrita permite a implementação de primitivas de sincronização sofisticada (Apache Zookeeper, 2020).

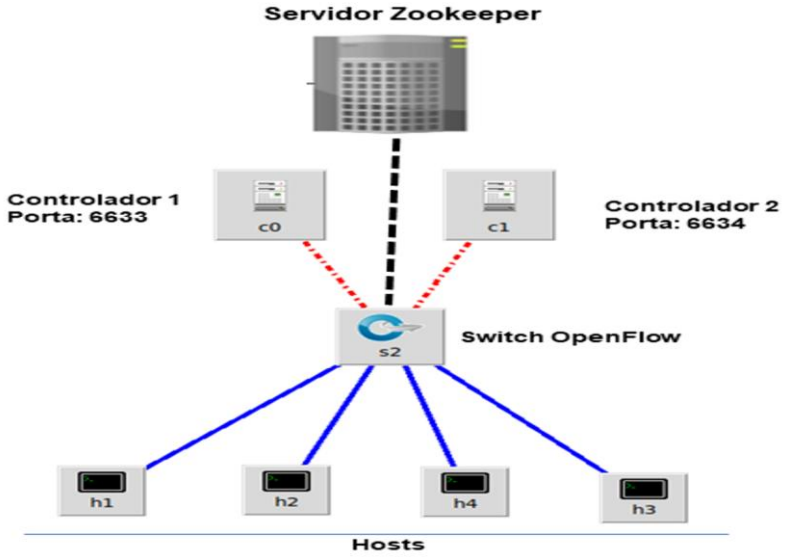
3 METODOLIGIA

A metodologia deste estudo foi desenvolvida a partir de uma revisão bibliográfica, com o objectivo de aprofundar o conhecimento sobre o funcionamento das redes definidas por software (SDN) e identificar as principais vulnerabilidades e potenciais vectores de ataque. A pesquisa focou na arquitetura das SDNs, especialmente nas suas três camadas principais: aplicação, controle e

infraestrutura. A compreensão detalhada dessas camadas foi fundamental para estabelecer uma base teórica sólida e orientar a fase prática do estudo. Para avaliar empiricamente a segurança e a tolerância a falhas em ambientes SDN, foi montada uma infraestrutura de testes que simulava um cenário operacional realista e um dos controladores foi deliberadamente desligado para simular um ataque, uma vez que o foco não está na análise do comportamento da rede diante de diferentes tipos de ataque, mas sim em assegurar a continuidade da comunicação quando o controlador activo falhar.

Figura 2

Proposta de infraestrutura SDN com redundância de controladores



Fonte: Elaborado pelo Autor, 2024

O servidor *Zookeeper* está conectado ao switch *OpenFlow* (s2) como um serviço de coordenação para o sistema, garantindo a consistência e a sincronização entre os controladores SDN. Existem dois controladores SDN conectados ao switch *OpenFlow* (s2), cada um operando em portas diferentes (Controlador 1 na porta 6633 e Controlador 2 na porta 6634). Esses controladores são responsáveis por gerenciar as regras de encaminhamento do switch, determinando como o tráfego deve ser tratado na rede. O switch *OpenFlow* (s2) é o elemento central da rede. Ele recebe instruções dos controladores sobre como encaminhar os pacotes de dados entre os hosts. A infraestrutura foi configurada com o objectivo de criar um ambiente de rede virtual que simulasse de forma eficaz uma rede SDN com redundância de controladores, podendo ser dois ou mais controladores na rede pois o propósito é garantir comunicação ininterrupta caso haja disfunção do controlador activo. Para isso, foi utilizado o Mininet, que permitiu a criação de topologias de rede virtuais compostas por *switches*, *hosts* e *links* e também desenvolvido um script na linguagem de programação Python abaixo mencionado, que ao ser executado registra e configura ambos os controladores no *Zookeeper*, permitindo que este os monitore continuamente.

```
from kazoo.client import KazooClient
from kazoo.recipe.election import Election
from kazoo.exceptions import KazooException
import time
# Configurações do Zookeeper e do controlador POX
ZOOKEEPER_HOST = '127.0.0.1:2181'
CONTROLLER_IP = '127.0.0.1'
CONTROLLER_PORT = 6633
CONTROLLER_ID = 'controller1'
while True:
    try:
        # Conectando ao Zookeeper
        zk = KazooClient(hosts=ZOOKEEPER_HOST, timeout=30.0)
```

```
        zk.start()
        # Caminho onde os controladores estarão registrados
        CONTROLLERS_PATH = '/controllers'
        LEADER_PATH = '/controllers/leader'
        # Criando o nó principal se ele não existir
        if not zk.exists(CONTROLLERS_PATH):
            zk.create(CONTROLLERS_PATH)
        # Registrando o controlador no Zookeeper
        controller_path = f'{CONTROLLERS_PATH}/{CONTROLLER_ID}'
        controller_data = f'{CONTROLLER_IP}:{CONTROLLER_PORT}'
        if not zk.exists(controller_path):
            zk.create(controller_path, controller_data.encode('utf-8'))
        else:
            zk.set(controller_path, controller_data.encode('utf-8'))
        print(f'Controlador {CONTROLLER_ID} registrado no Zookeeper em {controller_path} com IP:Port {controller_data}')
        # Iniciar a eleição de líder
        election = Election(zk, LEADER_PATH)
        def leader_function():
            print(f'{CONTROLLER_ID} foi eleito como líder.')
            while True:
                time.sleep(1)
        # Participar da eleição
        election.run(leader_function)
        # Manter o script em execução para que o controlador permaneça registrado e participe da eleição
        while True:
            time.sleep(1)
    except KazooException as e:
        print(f"Erro de conexão: {e}")
        print("Tentando reconectar...")
        time.sleep(10)  # Esperar 10 segundos antes de tentar reconectar
    finally:
        if 'zk' in locals() and zk.connected:
            zk.stop()
```

Durante a inicialização, o primeiro controlador detectado pelo *Zookeeper* assume a função de controlador primário, enquanto o segundo opera como controlador de backup. Para garantir a coordenação entre os controladores, o *Zookeeper* foi configurado para gerenciar o estado e a sincronização entre as duas instâncias. Por meio deste mesmo script, ele monitora constantemente o status dos controladores e, caso o controlador primário falhe, o *Zookeeper* detecta essa inactividade e redistribui suas funções ao controlador de *backup*. Esse processo ocorre atribuindo a nova função de primário ao controlador reserva, garantindo que ele assuma imediatamente a responsabilidade pelo gerenciamento da rede. Dessa forma, a infraestrutura mantém a continuidade operacional, evitando interrupções no serviço e assegurando maior resiliência à rede SDN.

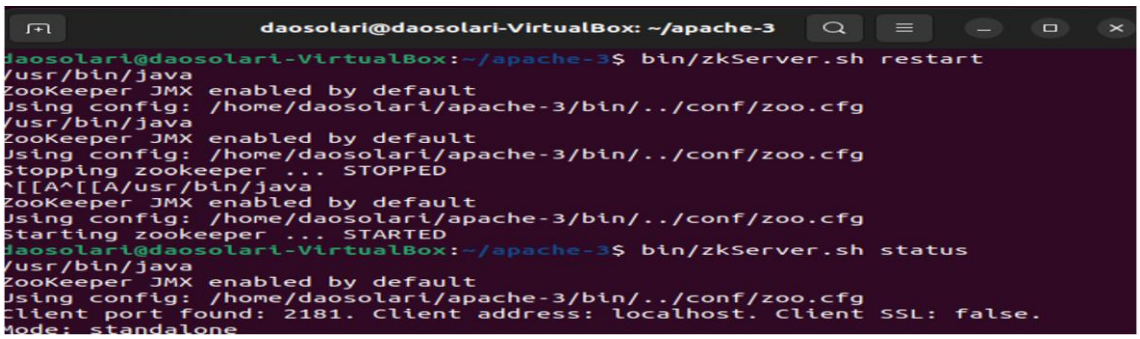
4 RESULTADOS E DISCUSSÃO

4.1 Resultados

A figura (3) mostra os comandos usados para reiniciar e verificar o status do servidor *Zookeeper*. O servidor foi iniciado e reiniciado com sucesso e está operando no modo *standalone*, escutando na porta 2181.

Figura 3

Inicialização do servidor Zookeeper

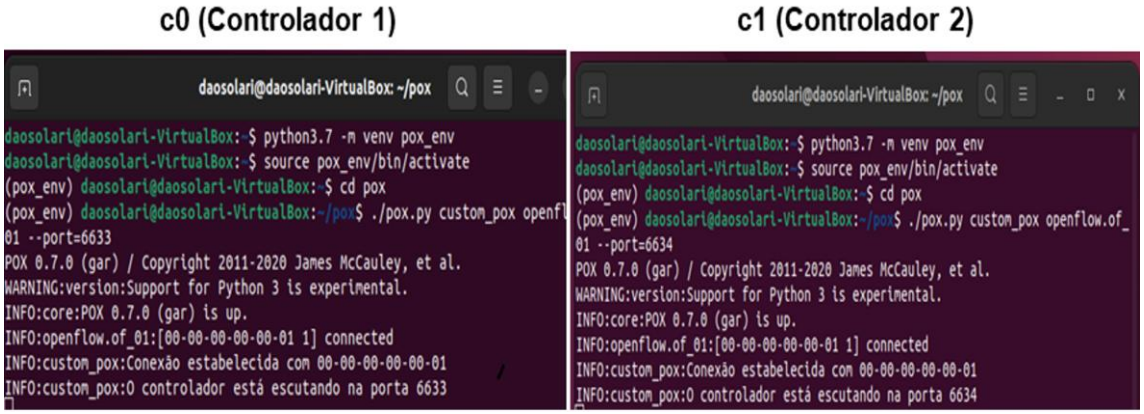


Fonte: Autor, 2024

A figura (4) apresenta os *logs* resultantes da inicialização bem-sucedida do controlador 1 (c0) e controlador 2 (c1) conectados no *switch openflow* e escutando na porta 6633 e 6634 respectivamente.

Figura 4

Inicialização dos controladores

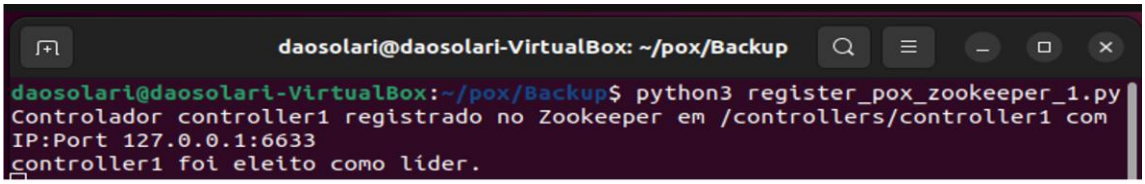


Fonte: Autor, 2024

Através da execução do *script register_pox_zookeeper_1.py* os controladores são registados e armazenados no servidor *Zookeeper* para os gerenciar, sincronizar e eleger o líder. A figura (5), mostra os detalhes do controlador1 como líder eleito.

Figura 5

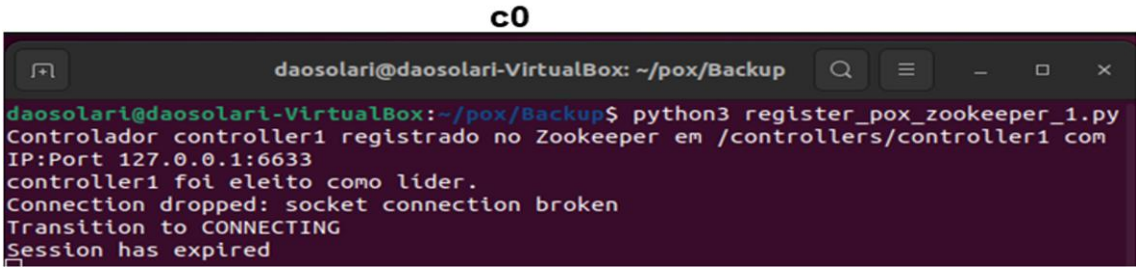
c0 como o controlador líder das redes definidas por software



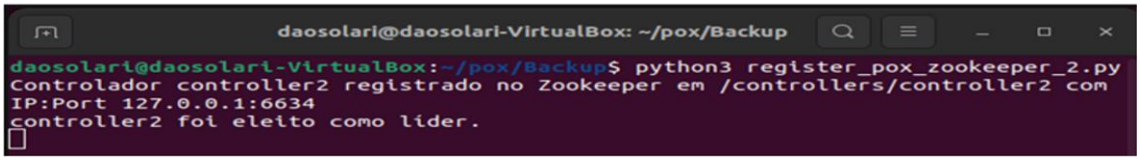
Fonte: Autor, 2024

Figura 6

Interrupção e eleição de novo Líder



c1



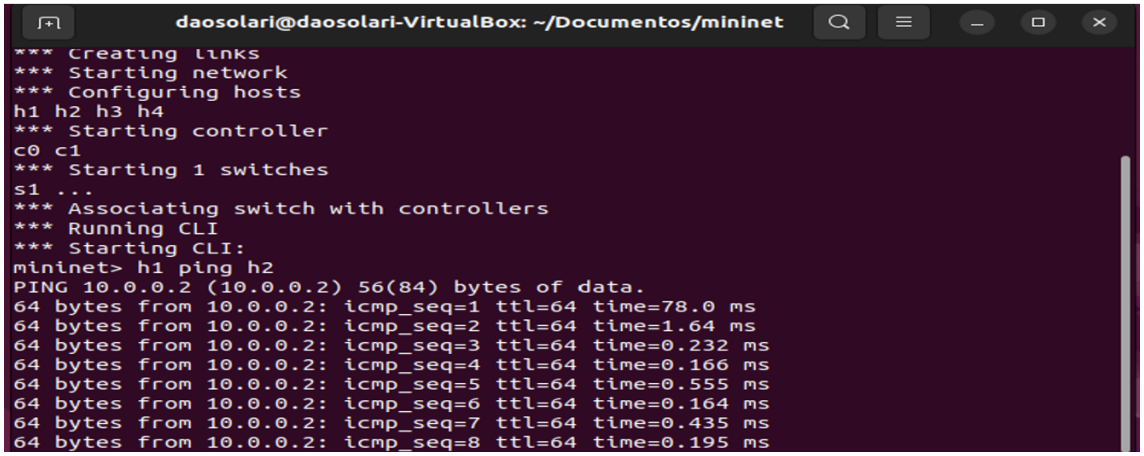
Fonte: Autor, 2024

Propositalmente foi desligado o controlador líder (c0) para simular qualquer tipo de ataque que resulta na perda total da comunicação. Como mostra a figura (6), o resultado é visível nos logs do controlador (c0) "*Session has expired* (sessão terminada)" que até então era o controlador activo. Automaticamente, o *Zookeeper* configura um novo líder, o c1 (Controller2 foi eleito com líder).

Com a eleição do novo controlador líder da rede, os links de conexão entre os dispositivos permanecem activos, a comunicação é reposta dentro do tempo estimado de recuperação. Como apresenta a figura (7), o *ping* (pacote de rede) enviado do host 10.0.0.2 usou o protocolo ICMP para um outro destino, tem uma sequência numérica ordenada, ttl=64 (tempo de vida do pacote até ser descartado) e tempo de ida e volta (RTT – Round Trip Time) alternada numa média de 0.543 milissegundo.

Figura 7

Rede funcional após a eleição do novo líder



Fonte: Autor, 2024

Com base na relação matemática proposta por Avizienis et al. (2001) (Equação 1), considerando um tempo operacional de 950,42 horas em um total de 1.000 horas de observação, a disponibilidade da rede foi calculada em 95,04 %. Paralelamente, de acordo com Sterbenz et al. (2010), o tempo de recuperação (*Recovery Time Objective* – RTO) obtido pela relação apresentada na Equação 2, foi estimado em 49,58 horas, com base na diferença entre o tempo total de observação (1 000 horas) e o tempo em que a rede permaneceu operacional (950,42 horas).

4.2 Discussão

A análise da disponibilidade da rede revelou um índice de 95,04 %, com base em um tempo operacional de 950,42 horas dentro de um intervalo total de 1 000 horas observadas. Este valor, embora elevado, indica que a rede esteve indisponível por aproximadamente 49,58 horas ao longo do período avaliado. Em termos práticos, isso representa um tempo considerável de inatividade, especialmente em ambientes que demandam alta disponibilidade, como instituições financeiras, hospitais ou serviços em nuvem. Segundo padrões de referência em Acordos de Nível de Serviço (SLAs), muitos provedores de serviços estabelecem metas mínimas de disponibilidade entre 99,0% e 99,99 %, dependendo da criticidade da aplicação (*Amazon Web Services*, n.d.; *Internacional Organization for Standardization*, 2018). Nesse contexto, a disponibilidade observada de 95,04 % evidencia uma margem de melhoria significativa. Isso sugere a necessidade de revisão da infraestrutura de rede, políticas de manutenção preventiva, ou ainda, da resiliência dos equipamentos e enlaces utilizados. Além disso, a indisponibilidade acumulada pode impactar negativamente a produtividade dos usuários finais, bem como comprometer a qualidade dos serviços oferecidos. Portanto, torna-se essencial identificar as principais causas das interrupções e

estabelecer um plano de acção voltado à mitigação desses eventos, buscando elevar o índice de disponibilidade para patamares compatíveis com boas práticas e exigências de mercado.

4.3 Recomendações

Com base nos achados deste estudo, recomenda-se: o uso de múltiplos controladores SDN. Diante dos resultados obtidos e considerando os padrões de excelência em ambientes profissionais que exigem níveis mínimos de 99 % tornam-se necessárias acções estratégicas para elevar a confidencialidade da infraestrutura analisada. As seguintes recomendações são propostas:

1. Implantação de monitoramento contínuo com utilização de ferramentas como Zabbix ou Nagios para garantir a detecção em tempo real de falhas, degradações de desempenho e anomalias, possibilitando respostas imediatas a incidentes;
2. Estabelecer uma rotina de manutenção periódica, incluindo actualizações de firmware, substituição de equipamentos obsoletos e testes de disponibilidade programados;
3. Documentar todos os eventos de indisponibilidade e realizar análises de causa raiz para reduzir reincidência e promover melhorias estruturais.

5 CONCLUSÃO

Este estudo abordou a problemática da protecção do controlador em redes definidas por software (SDN), destacando os riscos associados ao controle centralizado e propondo uma abordagem baseada na redundância de controladores com suporte do serviço de coordenação *Zookeeper*. A implementação de dois controladores POX em regime de *failover* demonstrou ser uma estratégia viável para mitigar falhas no plano de controlo, promovendo maior resiliência operacional da rede. Os testes realizados evidenciaram que, mesmo diante da falha de um dos controladores, a rede manteve-se funcional, com disponibilidade de 95,04 % ao longo de um período de 1 000 horas observadas. Apesar de satisfatória, essa taxa representa um tempo de inactividade acumulado de aproximadamente 49,58 horas, mas reforça a eficácia da estratégia de múltiplos controladores, para garantir a robustez da arquitetura SDN. Além disso, a utilização do controlador POX, embora adequada para ambientes de teste e validação conceitual, representa uma limitação quando se considera a escalabilidade e a performance em ambientes de produção. Assim, estudos futuros devem considerar a adopção de controladores SDN mais avançados, como ONOS ou *OpenDaylight*, bem como a avaliação de mecanismos mais robustos de detecção de falhas e recuperação automática, com ênfase na automação de resposta a incidentes.

6 REFERÊNCIAS

3rd Generation Partnership Project (3GPP). (2015). Technical Specification Group Services and System Aspects; Management and orchestration; Architecture framework. https://www.3gpp.org/ - Acesso 07/05/2024.

Abbasi, A.A., & Jin, H. (2018). v-Mapper: An Application-Aware Resource Consolidation Scheme for Cloud Data Centers. Future Internet, 10, 90. DOI:10.3390/fi10090090.

Amazon Web Services. (n.d.). AWS service level agreements. https://aws.amazon.com/legal/service-level-agreements. Acesso em 10/04/2025.

Alharbi, S., Hou, J., & Khan, S. U. (2017). Software-Defined Networking (SDN) and Security: A Survey. Computers & Security, 70, 324-344. doi: 10.1016/j.cose.2017.07.008.

Amani, S., Mahmoodi, T., Tatipamula, M., & Aghvami, H. (2014). Network function virtualization: State-of-the-art and research challenges. IEEE Communications Surveys & Tutorials, 16(1), 165-186. DOI:10.1109/COMST.2015.2477041.

Apache Zookeeper. (2020). Documentação do Zookeeper: 3.9. Zookeeper-Apache. Disponível em: https://zookeeper.apache.org/doc/r3.9.2/index.html. Acesso 10/04/2024.

Avizienis, A.; laprie, J. C., & Randell, B. (2001). Fundamental Concepts of Dependability. University of Newcastle upon Tyne, Computing Science, Technical Report No. CS-TR-739.

Chowdhury, N. M. M. K., & Boutaba, R. (2010). Network Virtualization: State of the Art and Research Challenges. IEEE Communications Magazine, 47(7), pp. 20-26. doi:10.1109/MCOM.2009.5237277.

Farhady, H., Lee, K., & Nakao, A. (2015). Software-Defined Networking: A Survey. Computer Networks, 81(1), pp. 79-96. DOI:/10.1016/j.comnet.2015.02.014.

Gkioulos, V.; Gunleifsen, H., & Weldehawaryat, G. K. (2018). A Systematic Literature Review on Military Software Defined Networks. Future Internet, 10, 88. DOI:10.3390/fi10090088.

Goldberg, R. P. (1971). Virtual machines—Semantics and examples. IEEE Computer Society Conference. Pp. 141-142.

Internacional Organization for Standardzation. (2018). ISO/IEC 20000-1:2018 – Information technology – Service management – part 1: service management system requirements. Geneva, Switzerland: ISO.

Kaur, K., Singh, J., & Ghumman, N. S. (2014). "Mininet as softwares defined networking testing platform," in International Conference on Communication, Computing & Systems (ICCCS), pp. 139-42.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), pp.14-76.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., & Shenker S. (2008). OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 38(2), pp. 69-74. DOI:10.1145/1355734.1355746.

Nadeau, T., & Gray, K. (2013). Final Thoughts and conclusions. SDN.O'REILLY Media, Inc.

Open Networking Foundation (ONF). (2016). SDN Architecture: Issue 1.1 (ONF TR-521). Open Networking Foundation. Disponível em https://opennetworking.org,

Stamou, A., Kakkavas, G., Tsitseklis, K., Karyotis, V., & Papavassiliou, S. (2019). Autonomic Network Management and Cross-Layer Optimization in Software Defined Radio Environments. Future Internet, 11, 37. DOI:10.3390/fi11020037.

Sterbenz, J. P.G., Hutchison, D., Çetinkaya, E.K., Jabbar, A., Rohrer, J.P., Scholler, M., & Smith, P. (2010). Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. Computer Networks, 54(8), pp. 1245-1265.

VMware (2019). " A Guide to SDN, SD-WAN, NFV, and VNF". Solution overview. USA. 6 p. Disponível em https://www.vmware.com/docs/208805aq-so-vcloud-guide-sd-wan-nfv-vfn-uslet-web.

Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2015). "A survey on software-defined networking". IEEE Communications Surveys & Tutorials, 17(1), pp. 27- 51. doi: 10.1109/COMST.2014.2330903.