Revista Brasileira de Computação Aplicada, Novembro, 2024

DOI: 10.5335/rbca.v16i3.15696
Vol. 16, Nº 3, pp. 63–73
Homepage: seer.upf.br/index.php/rbca/index

RBCA
ISSN 2176-6649

ARTIGO ORIGINAL

Prevenção de ataques de injeção de regras de fluxo em SDNs utilizando Blockchains

Preventing flow rules injection attacks in SDNs using Blockchains

Ramon Amorim dos Santos Delgado Ama[1] and Valério Rosset [2]

[1]Universidade Federal de São Paulo - UNIFESP, [2]Instituto tecnológico de Aeronáutica - ITA

ramon.a.ama@unifesp.br; *rosset@ita.br;

Recebido: 25/03/2024. **Revisado:** 12/11/2024. **Aceito:** 30/11/2024.

Resumo

Devido a característica de programabilidade das Redes Definidas por Software (SDNs) tanto o software de controle como as aplicações de rede estão sujeitos a ataques, como por exemplo a injeção de regras de fluxo falsas. Desse modo, existe uma necessidade de se investigar soluções alternativas que permitam a verificação de integridade, autenticidade, corretude e acordo sobre o conjunto de regras de fluxo a serem implantadas nos comutadores. Nesse contexto, a distribuição segura de regras de fluxo pode ser implementada por meio de Registros Distribuídos (DLs), mais especificamente utilizando a tecnologia de *Blockchains* (BCs). Esta tecnologia permite a manutenção de um registro de informações de forma distribuída e colaborativa entre os participantes, sendo estes os responsáveis pela integridade e validação dos dados inseridos. Tendo isso em vista esse artigo apresenta uma proposta de um modelo integrado utilizando BC para suporte a segurança para a distribuição e estabelecimento de regras de fluxo em SDNs. O modelo proposto foi validado em ambiente emulado. Os resultados mostraram que o modelo proposto é eficaz contra ataques baseados em técnicas de injeção de pacotes.

Palavras-Chave: SDN, Blockchain, Segurança, Análise de Desempenho.

Abstract

Due to the programmability feature of Software-Defined Networks (SDNs), the control software and network applications are susceptible to attacks, such as the injection of false flow rules. Thus, there is a need to investigate alternative solutions that enable the verification of integrity, authenticity, correctness, and agreement on the set of flow rules to be deployed on switches. In this context, the secure distribution of flow rules can be implemented through Distributed Ledgers (DLs), specifically utilizing Blockchain (BC) technology. This technology allows for the maintenance of a distributed and collaborative information registry among participants, who are responsible for the integrity and validation of the inserted data. With this in mind, this article presents an integrated model using BC to support security in the distribution and establishment of flow rules in SDNs. The proposed model was validated through experimental performance analysis and security assessment in an emulated environment. Results show that the proposed model is effective against attacks based on packet injection techniques.

Keywords: SDN; Blockchain; Security; Performace Analysis.

1 Introdução

A Internet das coisas (*Internet of Things*) é caracterizada por uma heterogeneidade de dispositivos embarcados equi-

pados com sensores/atuadores interconectados a outros recursos físicos ou virtuais que, por sua vez, são utilizados para controle, processamento e análise de dados (Gubbi et al., 2013). O desenvolvimento de soluções para esses ambientes deve considerar aspectos relacionados a confiabilidade, desempenho, segurança e privacidade de dados e, principalmente, a escalabilidade para tratar o grande volume de dados produzido pelos dispositivos de IoT. Nesse cenário o modelo de computação em nuvem (*Cloud Computing*) torna-se imprescindível e paradigmas de computação distribuída em Névoa (*Fog Computing*) a nas bordas (*Edge Computing*), são utilizados para reduzir significativamente a quantidade de dados transferidos para a infraestrutura central da Nuvem.

Uma das principais tecnologias associadas ao suporte a computação nas bordas são as Redes Definidas por Software (SDNs, *Software-Defined Networks*) (Turner et al., 2023). O plano de controle das SDNs oferece interfaces para implementação de políticas, algoritmos de roteamento entre outras funções (e.g. balanceamento de carga, virtualização, suporte a mobilidade etc) para determinar conjunto de regras das tabelas de fluxo dos comutadores de rede. O plano de controle, apesar de ser logicamente centralizado, pode ser implementado como um sistema fisicamente distribuído por razões de escalabilidade e confiabilidade. Nesse caso, APIs East-Westbound são utilizadas para permitir que vários controladores SDN se comuniquem e troquem informações de rede referentes a um mesmo domínio ou diferentes domínios (Rosa e Rothenberg, 2018).

Entretanto, a flexibilidade e falta de padronização do plano de controle das SDNs favorecem o surgimento de ameaças de segurança (Shu et al., 2016; Bannour et al., 2018). Desse modo, tanto o software de controle quanto as aplicações de rede estão sujeitos a ataques anteriormente inviáveis nas redes tradicionais, como por exemplo a injeção de regras de fluxo falsas e de *scripts* ou códigos maliciosos nas aplicações de rede (Abed et al., 2023). Portanto, existe a necessidade de se investigar soluções de segurança alternativas que permitam a verificação de consistência, autenticidade, corretude e acordo sobre o conjunto de regras de fluxo a serem instaladas nos comutadores. Nesse contexto, a distribuição segura de informações de controle de rede (e.g. informações de topologia e estado de enlaces) pode ser implementada por meio de armazenamento de dados em Registros Distribuídos (DLs, *Distributed Ledgers*) utilizando tecnologias como Blockchains (BCs). Tendo isso em vista esse artigo apresenta uma proposta de modelo para compartilhamento de regras de fluxo entre Controladores SDN baseado em Blockchain, para suporte a segurança de serviços de comunicação destinados a aplicações IoT. O modelo foi implementado em ambiante emulado e validado por meio de avaliação experimental segurança e de desempenho.

O restante deste artigo está organizado da seguinte forma: A Seção 2 descreve os trabalhos relacionados. Na Seção 3 apresentamos uma visão geral do modelo de ameaças à segurança, seguida de detalhes sobre o modelo proposto e a metodologia de execução. A Seção 4 se aprofunda na avaliação da proposta, apresentando os métodos de validação, experimentos realizados e os resultados obtidos. Finalmente, a Seção 5 resume os resultados obtidos e aponta direções de trabalho futuro.

2 Trabalhos Relacionados

Em Rosa e Rothenberg (2018), é proposto um modelo de integração entre SDN e BC em cenários de múltiplos domínios, utilizando uma implementação de SDN com controladores Ryu e protocolo OpenFlow v1.3. Os testes demonstraram resultados favoráveis, com baixa latência mesmo com o uso de Smart Contracts do Ethereum. Em Jiasi et al. (2019), foi sugerida a inserção de uma camada intermediária de comunicação baseada em BC para segurança e confiabilidade, armazenando fluxos e eventos nos blocos do BC. No contexto de redes IoT, Tselios et al. (2017) propôs o uso de BC como substituto da entidade central de verificação, descentralizando a validação de dados provenientes de dispositivos autorizados para que qualquer nó participante possa identificar falhas ou fluxos indevidos na rede.

Em Steichen et al. (2017), é apresentado o ChainGuard, um mecanismo de firewall baseado em SDN para proteger nós participantes de redes blockchain permissionadas. Esse mecanismo utiliza um controlador com protocolo OpenFlow conectado a um comutador para avaliar o fluxo de dados encaminhados aos nós da rede blockchain, mitigando ataques de negação de serviço (DoS e DDoS) e armazenando informações em listas brancas e negras. Em Kataoka et al. (2018), é proposta a Trust List, uma integração entre blockchain e SDN para mitigar ataques DDoS e garantir comunicação segura em dispositivos IoT, especialmente dispositivos de borda. A implementação envolve controladores SDN conectados a nós da rede blockchain, armazenando regras de fluxo como validação e confiança dos nós, propagando-as por meio do blockchain usando Smart Contracts da rede Ethereum. Já em Basnet e Shakya (2017), é proposto um sistema de troca segura de informações em SDN usando blockchain privado e criptografia SHA256. Nesse sistema, um arquivo é cifrado e submetido como transação no blockchain por meio de Smart Contracts do Ethereum, garantindo que apenas o host com a chave privada correta possa decifrar o arquivo, enquanto todos os participantes podem visualizar o conteúdo do blockchain.

Em Xue et al. (2019), é proposto um sistema que permite que múltiplos dispositivos heterogêneos participem de uma mesma SDN e padronizem a maneira como cada um deles pode transmitir dados por meio do blockchain. Duas abordagens são apresentadas: uma envolvendo blockchain consórcio com criptografia de chave pública e Smart Contracts, e outra usando topologia de controladores SDN distribuídos, onde a comunicação entre nós e controladores é feita por meio de Smart Contracts. Em El Houda et al. (2019), é proposto um framework chamado *Cochain-SC* para mitigação de ataques DDoS em SDNs, com abordagens inter e intra-domínio utilizando SDN e blockchain para controle de fluxos e validação. O estudo avalia a resiliência da rede contra ataques DoS, demonstrando uma notável resistência. Além disso, Abbasi e Khan (2017) apresenta o VeidBlock, uma técnica de verificação de identidade segura em SDNs, utilizando blockchain para autenticação e estabelecimento de regras da rede. Em Moin et al. (2019) é apresentada uma análise aprofundada sobre a aplicação da tecnologia blockchain em IoT, mencionando estratégias existentes, como *VeidBlock* e *ChainGuard*, e propondo um framework generalizável para implantação de block-

chain em dispositivos IoT, visando melhorar a segurança da rede.

No contexto do tema abordado neste artigo, Boukria et al. (2019) propõe um sistema para verificar a injeção de regras de fluxo falsas em redes definidas por software (SDN). O sistema utiliza blockchain para criar redundância nas regras de fluxo geradas pelo controlador, sendo que um nó do blockchain possui conhecimento da topologia para verificar se a regra gerada é consistente com o cenário real.

O grande diferencial do modelo proposto neste artigo está na prevenção da injeção de regras de fluxo falsas em redes SDNs, focando no plano de controle, de forma que tal solução não impacte negativamente no desempenho da rede e possa ser adaptada para diferentes contextos de SDNs.

3 Proposta

3.1 Modelo de Ameaças

Consideramos três possíveis pontos de ataque para injeção de regras de fluxo falsas em SDN, abrangendo um plano de controle distribuído com uma topologia plana. Na Fig. 1 indica os pontos em que um atacante pode explorar uma vulnerabilidade específica conforme descrito a seguir.

- **Ataque ao Controlador**: Um atacante pode personificar um comutador e reportar alterações de enlaces falsos, induzindo o controlador a criar novas regras de fluxo que direcionam o tráfego para um destino incorreto. Isso pode levar ao roubo de dados, ataques DoS ou descarte de dados ("black hole").
- **Ataque Man-in-the-Middle (MiTM)**: O atacante se coloca entre a comunicação do controlador e do comutador, alterando as regras de fluxo enviadas ou os parâmetros do comutador.
- **Ataque ao Comutador**: O atacante pode injetar regras falsas diretamente no comutador, personificando o controlador ou enviando pacotes de fluxo inválidos. Isso pode levar ao roubo de dados, concentração de fluxos em um único ponto da rede ou outros tipos de ataque.

3.2 Modelo Proposto

De acordo com o modelo de ameaças é perceptível que a existência de um mecanismo de validação das regras, antes da instalação nos comutadores, poderia evitar tais ataques, mitigando estes pontos de vulnerabilidade.

Desse modo, a utilização de uma rede de registros distribuídos (DL) para armazenamento das informações de rede é importante para constituir um banco de dados confiável e distribuído. O modelo também é resiliente a falhas de controladores uma vez que um controlador de backup tem acesso as informações consolidadas na DL ao assumir o controle sobre os comutadores geridos pelo controlador defeituoso.

Uma visão geral da proposta de implementação de BC em uma SDN é apresentada na Fig. 1.

3.3 Modelo do Sistema

O controlador é o responsável pelo gerenciamento da rede e controle das regras implementadas nos comutadores, bem como a descoberta de *hosts* e criação da NIB. Entretanto existem duas formas de se elaborar as regras a serem implementadas: proativamente e reativamente.Na controle reativo o controlador cria as regras de fluxo de acordo com a demanda gerada pelos comutadores. Ao receber pacotes de PACKET_IN, criados pelos enviados quando não há uma regra aplicável a um fluxo gerado por algum *host*. O controlador então determina a regra de fluxo para o encaminhamento dos pacotes e a instala no comutador. No controle proativo o controlador instala previamente uma série de regras de fluxo nos comutadores.

Nesse estudo foi considerada uma SDN com topologia plana composta por três controladores. Cada comutador da rede deverá estar conectado a múltiplos controladores e poderá ter *N hosts* conectados a ele. Os comutadores só exercem a função de repasse de pacotes. Os *hosts* são impedidos de comunicar-se diretamente com os controladores. As regras de fluxo e para o repasse de pacotes são processadas nos controladores, utilizando o controle proativo para a criação de regras de fluxo.

Considera-se que a NIB é uma informação que não se altera com frequência em curtos espaços de tempo, sendo que nesse contexto, a utilização de uma abordagem de controle proativo é mais interessante. A atualização da NIB é realizada pela Aplicação de Gestão de rede periodicamente, nesse processo, ela implanta novas regras de fluxo na SDN e atualiza da NIB na rede BC de forma automatizada.

Todos os elementos da rede possuem identificadores únicos e a comunicação entre controladores e comutadores é realizada por meio do protocolo OpenFlow. Também é assumido que na inicialização da rede todas as entidades são verificadas.

O modelo proposto considera um tipo privado de rede Blockchain (BC) para a implementação do serviço de registros distribuídos. A rede BC utiliza a mineração de um único bloco vazio, de forma a cada transação recebida com informações da NIB da rede ser inserida em um novo bloco, mantendo-se uma base de dados de estados de rede em ordem cronológica.

O modelo foi validado por meio de uma análise de desempenho e de segurança descritos nas próximas seções.

3.4 Metodologia de Execução

A execução do modelo é dividido em duas etapas, chamadas de pré-configuração e operação da rede, em que são definidas as funções dos respectivos componentes da rede: Controladores, comutadores e a rede de registros distribuídos (BC).

3.4.1 Etapa de Pré-configuração

Durante a fase de pré-configuração são realizadas trocas de mensagens de descoberta de rede entre controladores e comutadores via protocolo OpenFlow. Um esquema da etapa de pré-configuração é ilustrado na Fig. 2.Para cada *host* conectado a um comutador, é gerado um PACKET_IN (via OpenFlow) pelo comutador que é encaminhado ao controlador. O controlador reúne todas os dados recebidos em

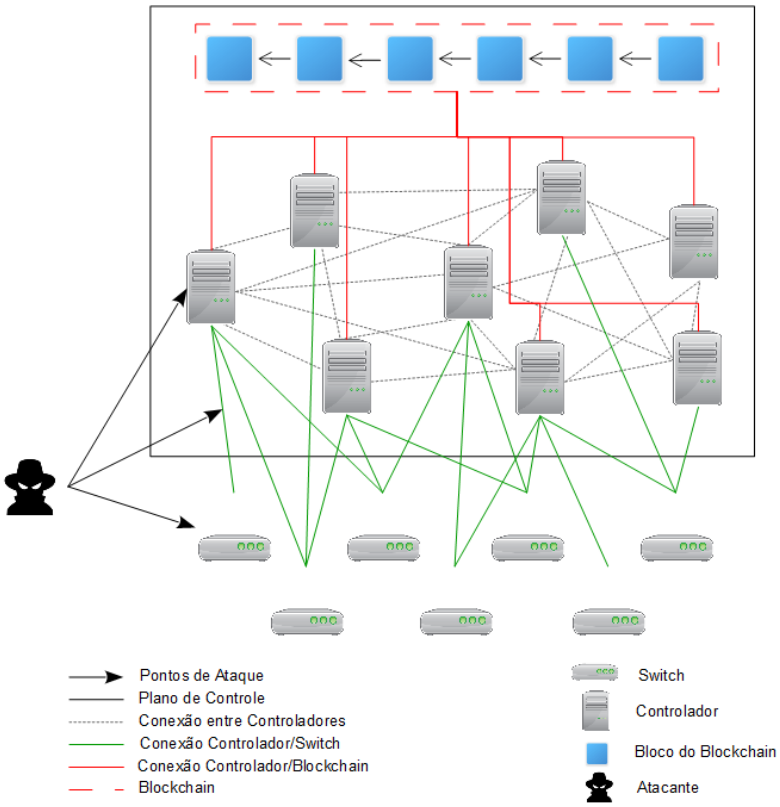

Figura 1: Modelo proposto de Integração de DLs em uma SDN para comunicação segura de regras e estado da rede entre os controladores.

uma NIB em formato JSON que é registrada no BC e armazena os endereços MAC dos *hosts* em uma *white list*. O controlador comunica-se com a Aplicação de Gestão através de protocolos REST, em canal seguro e com autenticação, para enviar informações sobre estado da rede e dispositivos. A Aplicação de Gestão processa as NIBs recebidas pelos controladores. Então uma NIB global, contendo uma lista de dispositivos confiáveis, é criada e submetida na rede BC (Multichain) em forma de transação.

A Aplicação de Gestão utiliza a NIB global mais recente da rede BC, através da busca pela última transação do último bloco, para geração das regras de fluxo. Mais especificamente, são utilizadas *Intents* para geração das regras de fluxo. *Intents* são diretivas baseadas em políticas. O núcleo do ONOS traduz as *Intents*, por meio de compilação, em regras de fluxo instaláveis. Desse modo, as *Intents* são dinâmicas e podem ser recompiladas automaticamente no caso de gargalos na rede ou queda de enlaces, sendo a escolha para um controle proativo no modelo proposto. Cada *Intent* determina os endereços dos extremos de uma comunicação, por meio do endereço MAC dos adaptadores de rede dos hosts. São criadas *Intents* host a host e submetidas para todos os controladores através de comunicação REST. Que, por sua vez, as processam e as instalam nos comutadores para o encaminhamento correto dos pacotes.

3.4.2 *Etapa de Operação de Rede*

Na etapa de operação da rede, *hosts* passam a gerar os fluxos de pacotes. Os comutadores, por sua vez, encaminham os pacotes através da implementação das regras instaladas pelos controladores. A atualização da NIB no BC ocorre periodicamente ou por alguma alteração do estado da rede, e.g. queda de conexão ou novo host conectado. Um esquema dessa etapa operação pode ser visualizada em mais detalhes através do fluxograma da Fig. 3. Desse modo, os controladores ONOS notificam a Aplicação de Gestão sobre as mudanças na rede realizando a atualização da NIB. Entretanto, a NIB atualizada só é submetida ao BC se não houver alterações.

4 Experimentos

4.1 Ferramentas Utilizadas

O modelo proposto para foi implementado por meio da integração das ferramentas: mininet, ONOS e multichain.

O controlador ONOS (*Open Network Operating System*, {www.opennetworking.org/onos/}), foi escolhido devido a boa escalabilidade e desempenho. O Plano de Controle foi configurado por meio de um *cluster* de controladores ONOS interconectados entre si. Cada controlador é instanciado em um *container* e possui um identificador único, podendo ser nomeado com um alias durante sua

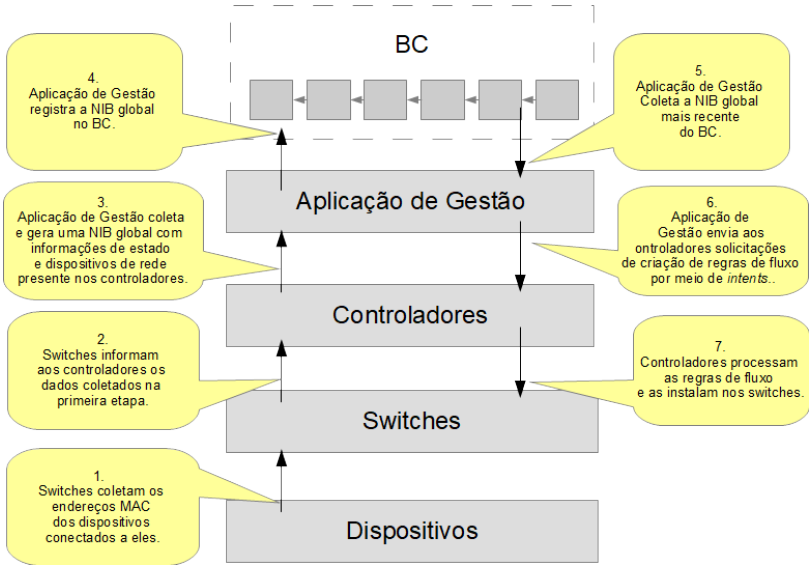

Figura 2: Esquema da fase de pré-configuração.

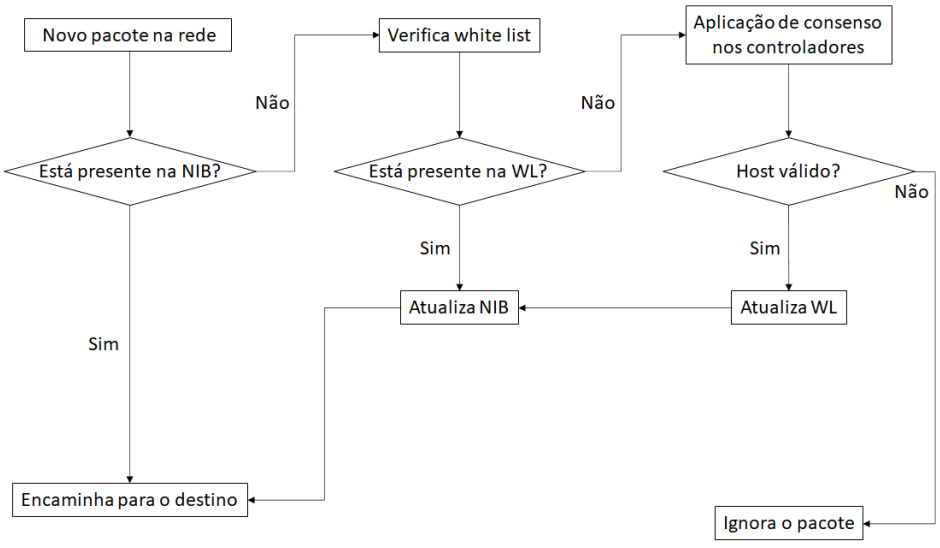

Figura 3: Etapa de operação de rede.

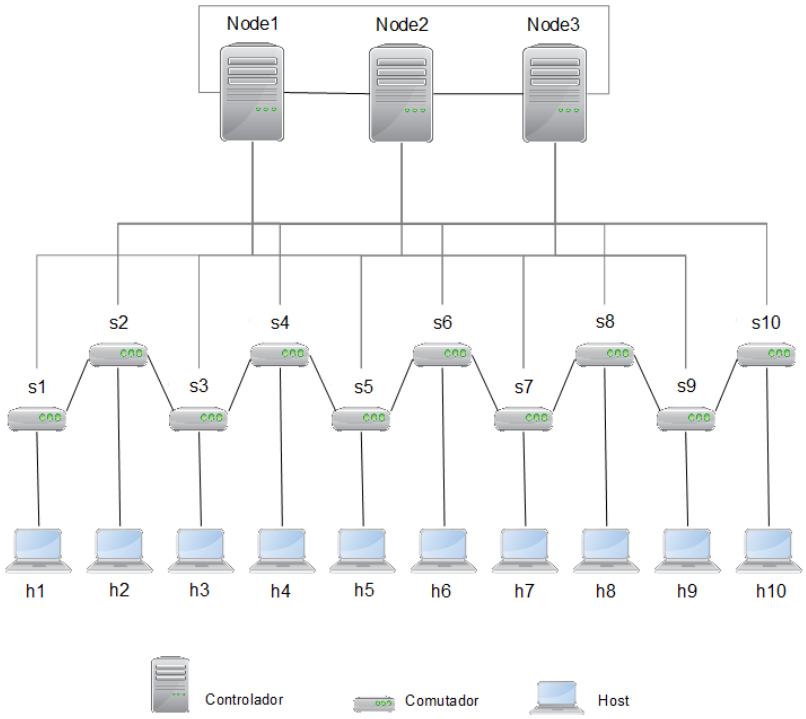

Figura 4: Topologia Linear.

instanciação e possuindo um endereço IP de acesso e comunicação entre os demais elementos da rede. O plano de dados foi configurado por meio do *mininet* em que hosts ficam isolados dos controladores.

A ferramenta MultiChain (*www.multichain.com*) foi utilizada para criação de uma rede Blockchain privada (BC) em que as informações de NIB observadas por cada controlador são armazenadas. Por fim, a Aplicação de Gestão foi implementada em linguagem Python 3.4+. Ela comunica-se com os controladores ONOS via REST e com o Multichain via REST.

4.2 Cenários para Análise de Desempenho

Foram consideradas três topologias distintas: uma com poucos elementos de rede e conexões (topologia Linear), outra com médio número de elementos de rede e conexões (topologia *Spine Leaf*) e por fim uma topologia com grande número de conexões e elementos de rede (topologia *Complete*). Além disso, topologias utilizando o modelo proposto considerando baixo, médio e alto tráfego de dados na rede, e então comparado os resultados com as mesmas topologias e tráfegos, porém seguindo um modelo tradicional de encaminhamento de pacotes *Reactive Forwarding*.

A seguir serão discutidos os detalhes de cada topologia.

- **Topologia Linear**: Essa topologia é formado por comutadores e *hosts* interconectados linearmente conforme a representação da Fig. 4.A topologia apresenta uma baixa conectividade entre os elementos de rede, posibilitando gargalo em pontos centrais. Embora exista essa linearidade de conexão, todos os comutadores possuem conexão com os controladores.
- **Topologia Spine Leaf**: Essa topologia é formada por duas camadas de comutadores, uma central e uma de borda. Na Fig. 5 os comutadores centrais ({sa, sb}) interconectados aos comutadores de borda ({s1, s2, s3, s4})
- **Topologia Complete**: Essa topologia apresenta um modelo completamente conectado, em que todos os comutadores possuem conexão entre si. A visão geral desta topologia pode ser visualizado na Fig. 6.

4.3 Resultados da Análise de Desempenho

A análise de desempenho visa comparar latência e vazão da rede, considerando cada topologia descrita, no controle reativo tradicional (*Reactive Forwarding*) e no controle proativo baseado em *Intents* combinado ao BC, chamado de modelo (*Proactive Intents*). Além disso, avaliou-se sua estabilidade e velocidade de recuperação em caso de quedas de link ou mudanças de rota. Ainda, visando avaliar a robustez do modelo em diferentes situações de utilização da rede, foram considerados três cenários fluxo de dados durante os experimentos: baixo (sem concorrência), médio (60% de nós transmitindo concorrentemente) e alto (100% de nós transmitindo concorrentemente). A vazão foi medida utilizando a ferramenta *iperf* (com 20 repetições de cada experimento), conforme os parâmetros descritos na Tabela 1.

4.3.1 *Resultados da Vazão*

A Fig. 7 apresenta os resultados referentes a vazão observada nos experimentos independente das topologias utilizadas.

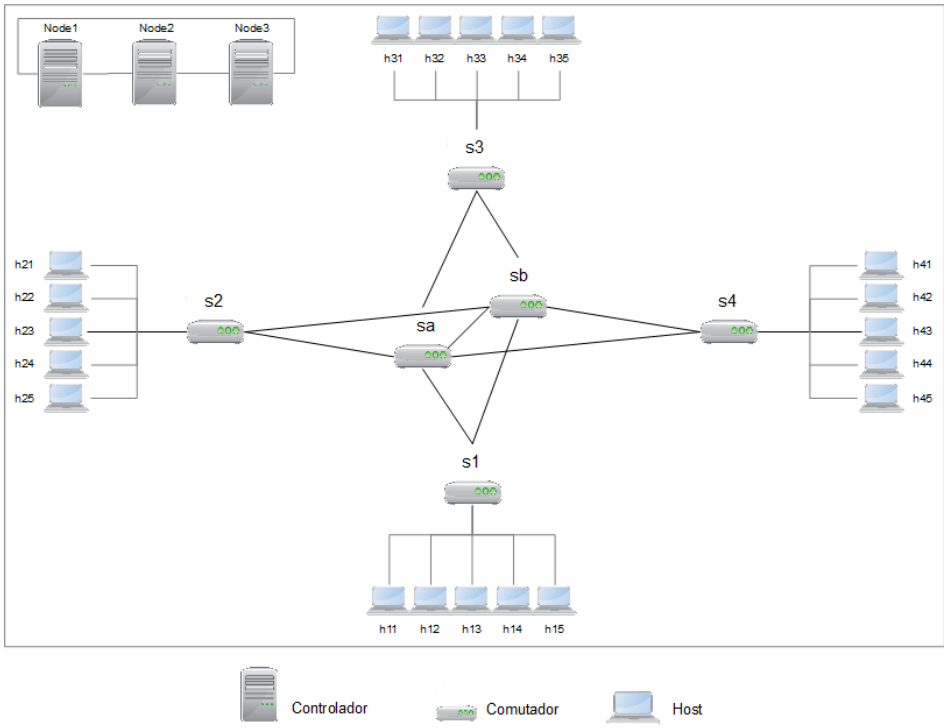

Figura 5: Topologia Spine Leaf.

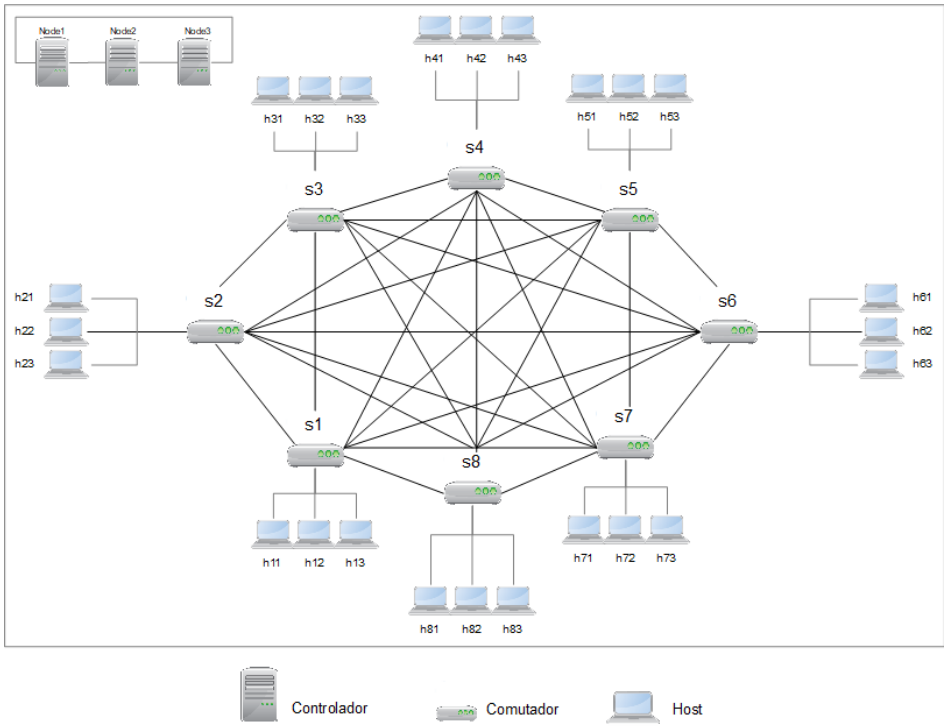

Figura 6: Topologia Complete.

Tabela 1: Tabela de Parâmetros

Parâmetros	Valores		
Topologia	Linear	Spine Leaf	Complete
Número de Hosts	10	20	24
Número de Comutadores	10	6	8
Número de Controladores	3	3	3
Taxa de Transmissão dos enlaces (gbps)	100		
Tipo de Enlace	Gigabit Ethernet		
Protocolo de Transporte	TCP		
Tamanho dos pacotes (bytes)	1500		
Tempo de Transmissão (s)	60		
Tráfego Concorrente	alto high,medium e low		

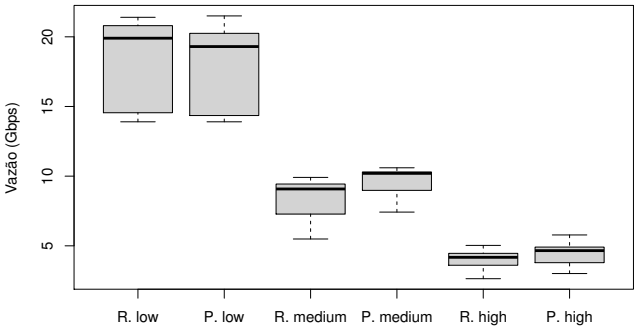

Figura 7: Vazão em diferentes cenários de tráfego da rede.

De modo geral, é esperado que a vazão na comunicação entre pares diminua com o aumento do tráfego concorrente. Nesse contexto, o modelo proposto apresentou desempenho comparativamente superior nos cenários em que o tráfego concorrente é mais intenso. Os resultados individuais observados nas diferentes topologias são discutidos na próximas subseções.

4.3.1.1 Topologia Linear. A Fig. 8 apresenta os resultados da vazão observada nos experimentos com topologia linear, considerando os três volumes de tráfego concorrente.

Para o cenário com baixo tráfego concorrente (low traffic) é possível perceber que o controle reativo apresentou um resultado de vazão superior ao proativo. Entretanto os resultados do controle reativo apresentam uma maior variação comparados ao proativo que, apesar de apresentar uma vazão ligeiramente inferior, foi mais estável.

No cenário de trafego médio (Medium Traffic) o modelo proposto mantém uma menor variação nos resultados obtidos, porém ocorre uma inversão em que o controle proposto passa a apresentar, em média, vazão superior de 8.6%.

No cenário de alto tráfego concorrente, modelo proposto apresenta uma vazão, em média, de 13.3% superior ao reativo.

4.3.1.2 Topologia Spine Leaf. Os resultados obtidos são apresentados na Fig. 9.

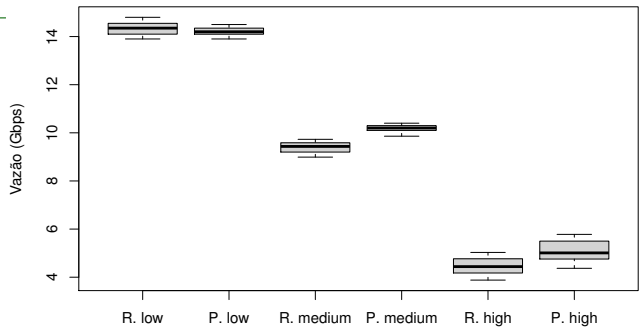

Figura 8: Comparação entre a vazão da topologia Linear utilizando *reactive forwarding* e *proactive Intents* em diferentes cenários de tráfego da rede.

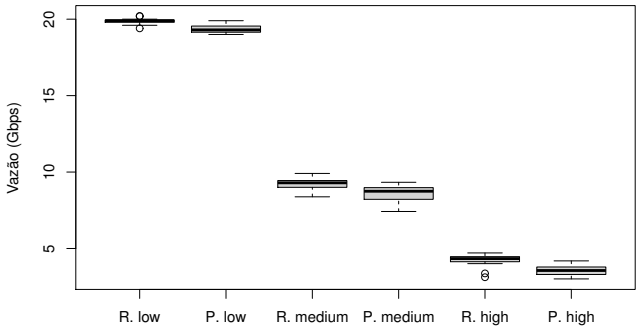

Figura 9: Comparação entre a vazão da topologia *Spine Leaf* utilizando *reactive forwarding* e *proactive intents* em diferentes cenários de tráfego da rede.

Nesse cenário o controle *Reactive Forwarding* apresentou melhor desempenho, em todas as intensidades de tráfego, se comparado ao modelo proativo proposto. Essa condição se justifica, porque o menor número de saltos necessários para a comunicação entre os pares favorece o controle reativo, quando comparada com a topologia linear.

4.3.1.3 Topologia Complete. A topologia complete consiste em um cenário de maior número de enlaces e *hosts* com múltiplas conexões, aumentando a complexidade do encaminhamento de pacotes. A Fig. 10 apresenta os resultados observados sobre a vazão nesse cenário.

Nesse cenário o modelo *Reactive Forwarding* apresentou desempenho superior com tráfego reduzido. Entretanto, considerando tráfegos concorrentes médio e alto foi observado uma superioridade significativa do modelo proativo proposto.

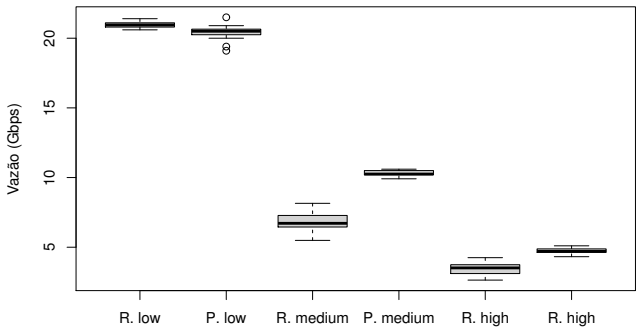

Figura 10: Comparação entre a vazão da topologia *Complete* utilizando *reactive forwarding* e *proactive intents* em diferentes cenários de tráfego da rede.

4.3.2 *Latência*

Além da vazão, foram realizados experimentos avaliar o atraso de comunicação observado entre pares de *hosts*, considerando uma situação normal de operação e uma com falha de enlace . As Fig. 11 e Fig. 12 apresentam os resultados obtidos comparando as abordagens *Reactive Forwarding* e *Proactive Intents* nesses cenários.

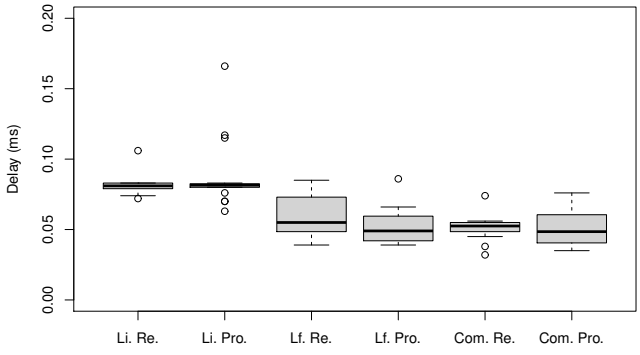

Figura 11: Comparação entre os atrasos na comunicação utilizando *Reactive Forwarding* e *Proactive Intents* com comunicação estável.

No cenário em que não há falhas de enlace, é possível observar que o controle *proactive intents* se mantém mais estável, entretanto com um atraso ligeiramente menor que o modelo *reactive forwarding* nas topologias Spine Leaf e Complete. Ao considerar uma instabilidade na rede, como uma queda de conexão momentânea (15 segundos), podemos observar que o modelo *proactive intents* também possui um atraso menor em sua recuperação comparado ao modelo *reactive forwarding*. Considerando a topologia

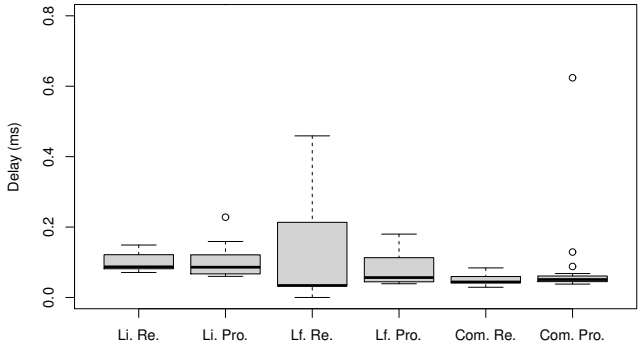

Figura 12: Comparação entre os atrasos na comunicação *Reactive Forwarding* e *Proactive Intents* com falha nos enlaces.

Spine Leaf, no cenário com falhas, ambos os controles apresentaram uma variação maior nos atrasos observados.

4.4 Análise Experimental de Segurança

A análise de segurança visou verificar experimentalmente se o modelo proposto foi capaz de mitigar ataques de injeção de regras de fluxo descritos no modelo de ameaças. Consideraram-se as seguintes condições para o manutenção de segurança do modelo: *i)* os comutadores estão conectados a múltiplos controladores e *ii)* mais da metade dos elementos rede não pode estar comprometida.

Foram considerados cenários de ataques a elementos de rede em que: *i.* o atacante envia regras de fluxo falsas para implantação de regras diretamente nos comutadores e *ii.* o atacante envia pacotes com informações falsas de enlace para que um controlador gere regras de fluxo equivocadas e as implemente nos comutadores.

Para cada cenário foi escolhido um *host* como atacante da rede e então efetuado os testes de segurança propostos. Considerando os cenários *i e ii*, em que o atacante se conecta a um comutador legítimo, foi avaliado se o *host* fraudulento é capaz de enviar pacotes ao comutador com sucesso, ter seus pacotes recebidos por outros hosts, gerar e injetar pacotes do tipo PACKET_IN e submeter pacotes OpenFlow ao comutador, cujos resultados são resumidos na Tabela 2.

Foi observado que um *host* fraudulento não é capaz de estabelecer comunicação direta com o controlador, tornando esta forma de ataque impossível nos casos da rede configurada nos modos *Reactive Forwarding* e no *Proactive Intents*. Entretanto, considerando a criação de pacotes PACKET_IN de forma indireta, foi observado que, ao receber pacotes, o comutador realiza esse envio ao controlador automaticamente no modo *Reactive Forwarding*. Sendo assim é possível enviar informações falsas ao *switch* para que o mesmo encaminhe um PACKET_IN errado ao controlador e gere regras de fluxo falsas na rede. Por outro lado, o

Tabela 2: Comparação da segurança utilizando *Reactive Forwarding* e *Proactive Intents*.

Modelo	Tipos de ataques			
	Envio	Recebimento	PACKET_IN	OpenFlow
Reactive Forwarding	✓	✓	✓	✓/✗
Proactive Intents	✗	✗	✗	✗

encaminhamento de pacotes PACKET_IN de forma automática pelos comutadores é bloqueada no modelo *Proactive Intents*. Como esperado, no modelo proposto todos os pacotes provenientes de *hosts* não validados na NIB do BC são automaticamente descartados, desse modo não ocorreram as trocas de mensagens via PACKET_IN entre comutador e controlador.

De maneira semelhante, pacotes injetados do protocolo OpenFlow foram descartados automaticamente no modo *Proactive Intents*, pois o modelo proativo proposto não prevê alterações de regras dinamicamente, indicando que o modelo é resistente a ataques de injeção de pacotes OpenFlow. Por outro, o modo *Reactive Forwarding* se mostrou vulnerável, visto que a injeção de pacotes OpenFlow obteve sucesso. Entretanto, essa vulnerabilidade pode ser evitada em redes com *Reactive Forwarding* através da validação de portas de controle e autenticação dos controladores.

5 Conclusões

Este artigo propôs um modelo integrado utilizando Blockchain para garantir a segurança na distribuição e estabelecimento de regras de fluxo em Redes Definidas por Software (SDNs). O modelo *Proactice Intents* foi validado em ambiente emulado e os resultados demonstraram desempenho adequado e resistência a ataques de injeção de pacotes maliciosos por meio da integração do Blockchain para a validação do conjunto de regras de fluxo a serem implantadas nos comutadores.

A injeção de regras de fluxo falsas é uma vulnerabilidade comum em redes SDN, e a prevenção desse tipo de ataque é um desafio ainda não completamente solucionado. Embora algumas pesquisas tenham abordado a detecção desse tipo de ataque, a prevenção completa ainda é um obstáculo. O modelo proposto neste artigo apresenta uma nova abordagem para a prevenção da injeção de regras de fluxo falsas, focando no plano de controle da rede SDN. Essa abordagem visa garantir que a solução não comprometa o desempenho da rede e possa ser adaptada a diferentes cenários. O diferencial do modelo apresentado nesse artigo está na utilização de técnicas de blockchain para fortalecer a segurança da rede SDN, oferecendo uma camada adicional de proteção contra ataques e garantindo a integridade das regras de fluxo.

A utilização de Blockchain no contexto de SDNs apresenta diversos benefícios, como a imutabilidade das regras de fluxo, a transparência das transações, a descentralização da rede e a confiabilidade do sistema. Embora o modelo tenha sido validado em ambiente emulado, pesquisas futuras serão necessárias para avaliar seu desempenho em ambientes reais e em larga escala. Além disso, outros aspectos de segurança, como a proteção contra ataques direcionados ao próprio Blockchain, também devem ser considerados.

Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo apoio financeiro (Processo: 19/23382-2).

Referências

Abbasi, A. G. e Khan, Z. (2017). Veidblock: Verifiable identity using blockchain and ledger in a software defined network, *Companion Proceedings of The10th International Conference on Utility and Cloud Computing*, UCC '17 Companion, Association for Computing Machinery, New York, NY, USA, p. 173–179. https://doi.org/10.1145/3147234.3148088.

Abed, S., Jaffal, R. e Mohd, B. J. (2023). A review on blockchain and iot integration from energy, security and hardware perspectives, *Wireless Personal Communications* **129**(3): 2079–2122. https://doi.org/10.1007/s11277-023-10226-5.

Bannour, F., Souihi, S. e Mellouk, A. (2018). Distributed sdn control: Survey, taxonomy, and challenges, *IEEE Communications Surveys & Tutorials* **20**(1): 333–354. "https://doi.org/10.1109/COMST.2017.2782482.

Basnet, S. R. e Shakya, S. (2017). Bss: Blockchain security over software defined network, *2017 International Conference on Computing, Communication and Automation (ICCCA)*, IEEE, pp. 720–725. https://doi.org/10.1109/CCAA.2017.8229910.

Boukria, S., Guerroumi, M. e Romdhani, I. (2019). Bcfr: Blockchain-based controller against false flow rule injection in sdn, *2019 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, pp. 1034–1039. https://doi.org/10.1109/ISCC47284.2019.8969780.

El Houda, Z. A., Hafid, A. S. e Khoukhi, L. (2019). Cochain-sc: An intra-and inter-domain ddos mitigation scheme based on blockchain using sdn and smart contract, *IEEE Access* **7**: 98893–98907. https://doi.org/10.1109/ACCESS.2019.2930715.

Gubbi, J., Buyya, R., Marusic, S. e Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions, *Future generation computer systems* **29**(7): 1645–1660. https://doi.org/10.1016/j.future.2013.01.010.

Jiasi, W., Jian, W., Jia-Nan, L. e Yue, Z. (2019). Secure software-defined networking based on blockchain, *arXiv preprint arXiv:1906.04342* . https://doi.org/10.48550/arXiv.1906.04342.

Kataoka, K., Gangwar, S. e Podili, P. (2018). Trust list: Internet-wide and distributed iot traffic management using blockchain and sdn, *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, IEEE, pp. 296–301. https://doi.org/10.1109/WF-IoT.2018.8355139.

Moin, S., Karim, A., Safdar, Z., Safdar, K., Ahmed, E. e Imran, M. (2019). Securing iots in distributed blockchain: Analysis, requirements and open issues, *Future Generation Computer Systems* **100**: 325–343. https://doi.org/10.1016/j.future.2019.05.023.

Rosa, R. V. e Rothenberg, C. E. (2018). Blockchain-based decentralized applications meet multi-administrative domain networking, *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, ACM, pp. 114–116. https://doi.org/10.1145/3234200.3234217.

Shu, Z., Wan, J., Li, D., Lin, J., Vasilakos, A. V. e Imran, M. (2016). Security in software-defined networking: Threats and countermeasures, *Mobile Networks and Applications* pp. 1–13. https://doi.org/10.1007/s11036-016-0676-x.

Steichen, M., Hommes, S. e State, R. (2017). Chainguard - a firewall for blockchain applications using sdn with openflow, *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, IEEE, pp. 1–8. https://doi.org/10.1109/IPTCOMM.2017.8169748.

Tselios, C., Politis, I. e Kotsopoulos, S. (2017). Enhancing sdn security for iot-related deployments through blockchain, *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, pp. 303–308. https://doi.org/10.1109/NFV-SDN.2017.8169860.

Turner, S. W., Karakus, M., Guler, E. e Uludag, S. (2023). A promising integration of sdn and blockchain for iot networks: A survey, *IEEE Access* . https://doi.org/10.1109/ACCESS.2023.3260777.

Xue, C., Xu, N. e Bo, Y. (2019). Research on key technologies of software-defined network based on blockchain, *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE, pp. 239–2394. https://doi.org/10.1109/SOSE.2019.00041.