TrueState-SNA: Uma Solução Baseada em SSoT para Automação de Redes Heterogêneas

Jerônimo Menezes, Diego Kreutz, Rodrigo Brandão Mansilha

[1] Laboratório de Estudos Avançados em Computação (LEA)
Programa de Pós-Graduação em Engenharia de Software (PPGES)
Universidade Federal do Pampa (UNIPAMPA)

jscmenezes@gmail.com, diegokreutz@unipampa.edu.br, mansilha@unipampa.edu.br

Resumo. *A TrueState-SNA, baseada em uma arquitetura de camadas e software de código aberto, emerge como uma solução inovadora para o gerenciamento de redes heterogêneas, combinando flexibilidade e eficiência. Ao integrar Netbox, Net2d e Ansible, a solução centraliza a gestão de configurações em uma Single Source of Truth (SSoT), automatizando a transição de estados desejados para estados operacionais. As principais contribuições da TrueState-SNA incluem: uma arquitetura aberta e adaptável, suporte a múltiplos fabricantes e tecnologias, redução do risco de lock-in e a promoção de uma gestão de rede consistente e confiável.*

1. Introdução

O número cada vez maior de dispositivos, modelos, fabricantes e protocolos, tem desafiado as corporações a gerenciar estas infraestruturas cada vez mais heterogêneas. O gerenciamento integrado de componentes de rede e serviços heterogêneos, uma tarefa complexa, é um tema recorrente na literatura [Li and Sandrasegaran, 2005, Lampropoulos et al., 2005, Landman et al., 2010]. Apesar de sua longa história, o gerenciamento integrado de redes e tecnologias continua sendo amplamente discutido em diferentes cenários e casos de uso, como firewalls, redes *5G/6G*, redes celulares, redes de satélites e redes integradas espaço-ar-solo [Liu et al., 2018, Miao et al., 2016, Wang et al., 2021, Shahjee and Ware, 2022].

Recentemente, o gerenciamento integrado de redes passou a incorporar novos conceitos, como *Intent-Based Networking* (IBN), no qual linguagens de representação e intenções desempenham um papel central no processo de gerenciamento [Leivadeas and Falkner, 2022, Zeydan and Turk, 2020, Wei et al., 2020, Riftadi and Kuipers, 2019]. As redes IBN também impulsionaram pesquisas e desenvolvimentos voltados para *Autonomous Networking* (AN) e *Zero Touch Networking* (ZTN) ou *Zero Touch Network and Service Management* (ZSM), com o uso crescente de Inteligência Artificial (IA) [Hamadanian et al., 2023, Gallego-Madrid et al., 2022, Zhang et al., 2022, Siddiqi et al., 2022, Benzaid and Taleb, 2020]. Entre as principais premissas estão a capacidade da IA de descobrir padrões ocultos, extrair *insights* automaticamente, adaptar-se a padrões temporais e espaciais da dinâmica da rede, reduzir a necessidade de intervenção humana e personalizar tarefas de gerenciamento para diferentes configurações [Zhang et al., 2022].

A implementação da *Source of Truth* (SoT) ou *Single Source of Truth* (SSoT) surge como um pilar essencial para o gerenciamento eficaz e autônomo de redes modernas, IBN, ZTN e ZSM, eliminando inconsistências e desafios na orquestração de ambientes heterogêneos [Linder et al., 2024, Karvinen, 2023, Errea et al., 2023, Lingga et al., 2022, Mulyana and Fakih, 2022a, Clemm et al., 2022]. Ao centralizar informações críticas, as SSoTs simplificam a gestão de grandes redes, asseguram a tradução precisa de intenções em políticas e facilitam a automação de configurações, promovendo consistência e confiabilidade [Khan et al., 2022, Lingga et al., 2022]. A adoção de SSoTs, exemplificada pelo uso do *Infrahub* no gerenciamento de VLANs, possibilita a unificação de dados e a validação de configurações, otimizando a eficiência e a precisão na gestão de infraestruturas complexas [Linder et al., 2024].

Neste cenário de crescente demanda por automação, é crucial desenvolver soluções baseadas em uma arquitetura que viabilize a automação de mudanças nos *estados operacionais* dos dispositivos a partir da declaração dos *estados desejados* presentes na SSoT. A automação de rede deve ser feita de forma consistente, exigindo um acordo sobre a fonte de informações válidas, que inclua dados sobre o estado real da rede (por exemplo, dispositivos, cabeamento, endereços IP) [Mulyana and Fakih, 2022b].

Neste trabalho, inspirados no legado histórico de estudos e conhecimentos acumulados na área de AN, ZTN, IBN e IA, propomos a TrueState-SNA como uma solução de arquitetura em camadas para o gerenciamento prático e operacional de redes heterogêneas, onde cada projeto utilizado representa uma dessas camadas. Partindo de uma fonte confiável de informações para os parâmetros de configuração de cada objeto da infraestrutura, ou seja, a SSoT (*Estado Desejado* de configuração de dispositivos e serviços), a arquitetura permite um fluxo de processamento dessas informações e a implementação automática desses parâmetros como configuração do *Estado Operacional* dos dispositivos e serviços. Ou seja, a TrueState-SNA permite a vinculação do *Estado Desejado* com o *Estado Operacional* através de uma abordagem que garante flexibilidade em suportar cenários heterogêneos e expandir funcionalidades como o gerenciamento de *VLANs*, protocolos de roteamento e políticas de segurança. Partindo de uma combinação de projetos de código aberto, como o *Netbox*[1], *Django Rest Framework*[2], *Net2d*[3] e *Ansible*[4], a TrueState-SNA visa oferecer aos administradores de rede e desenvolvedores uma arquitetura eficaz, flexível, baseada em SSoT para a automação da infraestrutura de TI.

Como contribuições do trabalho, destacamos: **(a)** uma arquitetura aberta e flexível para automação de dispositivos e serviços de rede; **(b)** a vinculação do *Estado Desejado* com o *Estado Operacional*; **(c)** a capacidade de suportar múltiplos fabricantes e tecnologias, reduzindo o risco de *lock-in* e aumentando a flexibilidade operacional; **(d)** a implementação de uma SSoT para garantir consistência e confiabilidade na gestão de configurações.

[1] `https://netboxlabs.com/oss/netbox/`
[2] `https://www.django-rest-framework.org/`
[3] `https://github.com/net2d-community/net2d`
[4] `https://docs.ansible.com/`

2. TrueState-SNA: Arquitetura e Implementação

A arquitetura TrueState-SNA é projetada para implantar o *Estado desejado* como *Estado operacional*, automatizando, dessa forma, as operações básicas de configuração dos dispositivos gerenciados. A Figura 1 ilustra a arquitetura proposta. Observe-se ao centro a pilha de camadas propostas, à esquerda, as soluções conceituais integradas e, à direita, os pacotes de software utilizados em cada camada.

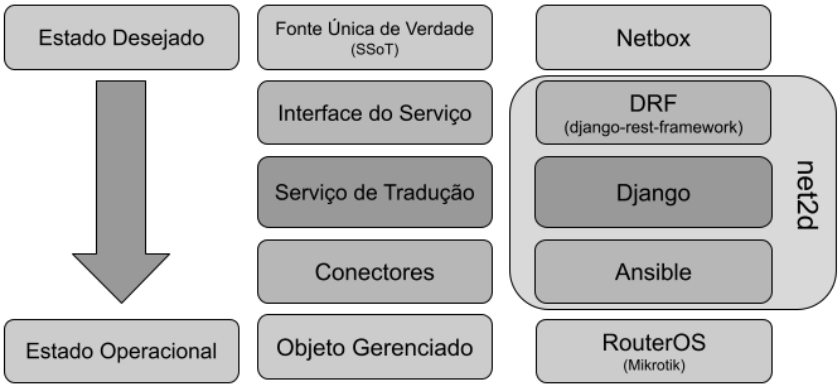


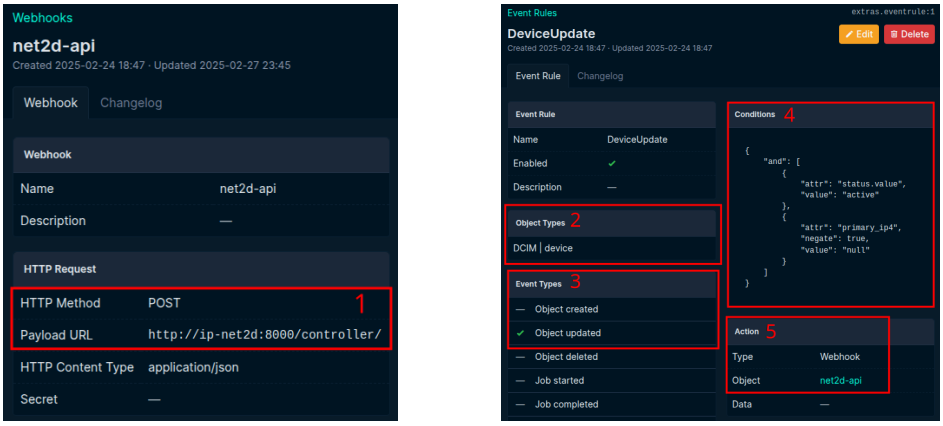
Figura 1. Visão geral da arquitetura e implementação da TrueState-SNA.

Em resumo, a TrueState-SNA consiste em um conjunto de códigos Python[5] integrados a módulos e ferramentas de código aberto já existentes. O núcleo da TrueState-SNA é implementado utilizando o *Framework Django*, combinado principalmente com o *Django REST Framework*, para a estruturação da API Web, o *Jinja*, como renderizador de templates, o Ansible, como *engine* de automação, além de outros pacotes de *software* de código aberto. Essa abordagem visa implementar uma arquitetura em camadas, focada na automação do gerenciamento de dispositivos, com ênfase na configuração dos mesmos a partir da manipulação da Fonte Única de Verdade (SSoT). A seguir, discutimos cada uma das camadas seguindo uma abordagem de cima para baixo.

Fonte Única de Verdade (SSoT). A SSoT é implementada no NetBox, que atua como um repositório centralizado para as informações de *estado desejado* de cada item de configuração. O usuário da TrueState-SNA gerencia diretamente na SSoT dados como o nome do dispositivo, interfaces de rede e endereçamentos IP. Para garantir a automação, é configurada uma *Event Rule* no NetBox, que dispara um *Webhook* com uma requisição HTTP POST para o *endpoint* "/controller/" da REST API da TrueState-SNA sempre que um objeto do tipo *Device* é atualizado.

A Figura 2 ilustra as configurações básicas para a integração do NetBox à TrueState-SNA. À esquerda, são exemplificados os parâmetros da requisição disparada pelo *Webhook*, sendo as principais (1) o método HTTP POST e a URL do Net2d. À direita, observa-se a configuração (2) dos tipos de objetos que estarão submetidos à regra, (3) em quais eventos eles serão executados, (4) em quais condições e (5) a ação que será tomada, no caso, o *Webhook*. A regra (*Event Rule*) pode ser configurada para ser executada sempre que um objeto do tipo *Device* com *status* ativo e um endereço IP associado for alterado no NetBox, por exemplo.

[5] `https://www.python.org/`



(a) *Webhook* (b) *Event Rule*

Figura 2. Configurações para integração do Netbox com o Net2d.

Interface do Serviço. Consiste em uma *API REST* que disponibiliza um *endpoint* para receber as requisições geradas pela SSoT sempre que houver alguma alteração no estado de algum objeto (por exemplo, *Device* ou *Interface*). a TrueState-SNA suporta o *deploy* completo de um dispositivo a cada vez que ele é atualizado.

A camada de Interface de Serviço é implementada utilizando o módulo *Django REST Framework* (DRF)[6], um kit de ferramentas para construção de APIs Web. Sua responsabilidade no sistema é disponibilizar o *endpoint* que será consumido pelo *Webhook* configurado no NetBox e, ao receber os dados enviados por uma requisição, disparar uma tarefa de configuração para o dispositivo alterado na SSoT. Para evitar que a comunicação com a API do NetBox seja lenta e suscetível a erros de *timeout*, foi utilizado o Celery[7], que permite executar as tarefas de *deploy* de forma assíncrona. Essa arquitetura permite que a API responda ao cliente rapidamente com um identificador, o qual pode ser usado para acompanhar a execução da tarefa. O trecho de código ilustrado na Figura 3 demonstra a utilização do DRF para implementar uma função que atende requisições POST, extrai o identificador do dispositivo e cria uma tarefa assíncrona para o *deploy* do respectivo objeto.

```
1 @api_view(['POST'])
2 def sw_deploy(request, format=None):
3     device_id = request.data['data']['id']
4     task = sw_deploy_task.delay(device_id)
5     return Response(task.id)
```

Figura 3. Criação das tarefas de *deploy* à partir das requisições recebidas pela SSoT.

Serviço de Tradução. Após a tarefa de *deploy* ser criada na *Interface do Serviço* e o seu identificador retornado ao cliente da API, o Serviço de Tradução inicia o processo de tradução do *estado desejado*, expresso na SSoT, elaborando *playbooks* contendo as tarefas de configurações necessárias para alterar o *estado operacional* do dispositivo. A Figura 4 resume o processo de tradução e configuração.

[6] https://www.django-rest-framework.org/
[7] https://docs.celeryq.dev/

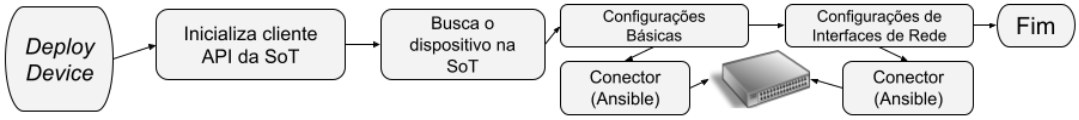


Figura 4. Etapas do processo de tradução e configuração

O *Serviço de Tradução* é implementado utilizando *templates* Jinja[8] para a renderização de *playbooks* Ansible. Um exemplo de *playbook* utilizado para a configuração de interfaces de rede de um dispositivo que executa o RouterOS[9] como sistema operacional pode ser visualizado na Figura 5.

```
---
- name: RouterOS Base
  hosts: localhost
  gather_facts: true
  module_defaults:
    group/community.routeros.api:
      hostname: {{ hostname }}
      password: admin
      username: admin
      tls: false

  tasks:
  {% block tasks %}{% endblock %}
```

(a) *Base comum para playbooks*

```
{% extends "base-playbook.yml.jinja" %}
{% block tasks %}
  - name: Configure IP Adresses
    community.routeros.api_modify:
      path: "ip address"
      handle_absent_entries: ignore
      handle_entries_content: remove_as_much_as_possible
      data:
        {% for ip in ipv4_addresses %}
        - interface: {{ interface }}
          address: {{ ip["address"] if ip is not none }}
        {% endfor %}
{% endblock %}
```

(b) *Renderização de interfaces e IPs*

Figura 5. Uso de *templates* Jinja para renderização de *playbooks* Ansible.

Os templates são gerados com base em um modelo comum, como ilustrado na Figura 5(a). Esse modelo base é enriquecido com as configurações específicas das interfaces de rede. A Figura 5(b) exibe o processo de renderização do template a partir de uma lista de endereços IP vinculados a cada interface do dispositivo na SSoT. Uma vez concluída a renderização, os playbooks resultantes são encaminhados aos conectores para serem executados.

Conectores. Os *Conectores* executam os *playbooks* gerados para cada um dos dispositivos. A TrueState-SNA está implementada para suportar dispositivos RouterOS por meio da biblioteca *librouteros*[10]. Esse módulo Python permite ao Ansible configurar os dispositivos RouterOS através de sua API. Embora a TrueState-SNA utilize o Ansible como ferramenta principal, outras soluções podem ser integradas para essa tarefa, como o *Netmiko*[11], que possibilita o envio de comandos CLI de forma programática. A escolha pelo Ansible na TrueState-SNA deve-se à sua capacidade de trabalhar com base na declaração dos estados desejados, eliminando a necessidade de preocupações com aspectos específicos da CLI de cada modelo ou fabricante de equipamento. Essa abordagem facilita a expansão do suporte a novos dispositivos no futuro, mantendo a flexibilidade e a escalabilidade da solução.

Objeto Gerenciado. Reúne dispositivos heterogêneos gerenciados através de uma SSoT. Na versão atual, os dispositivos precisam suportar Ansible.

[8] `https://jinja.palletsprojects.com/en/stable/`
[9] `https://mikrotik.com/software`
[10] `https://github.com/luqasz/librouteros`
[11] `https://github.com/ktbyers/netmiko`

3. Avaliação e Resultados

3.1. Metodologia

Avaliamos a TrueState-SNA quanto à sua capacidade de reduzir o tempo de operação e os erros humanos cometidos nas tarefas de configuração de dispositivos de rede. A Figura 6 ilustra a topologia da rede usada na avaliação. Ela é composta por 3 roteadores e 3 dispositivos clientes, cada um deles com 3 interfaces de rede. Na mesma rede de gerência, instalaram-se os serviços da TrueState-SNA. Os roteadores foram inicialmente endereçados apenas em suas interfaces de gerência, enquanto os clientes iniciaram os experimentos plenamente configurados. A topologia do experimento visa simular a configuração de roteadores para ativar o acesso às redes clientes ao *backbone* de uma organização, por exemplo. Solicitou-se aos operadores a configuração do endereçamento IPv4 e IPv6 das interfaces dos roteadores, a fim de estabelecer a comunicação ponto a ponto com os clientes e permitir conectividade com o restante da infraestrutura simulada. Utilizou-se um código Python para avaliar o tempo consumido pelos operadores na configuração dos roteadores, tanto de forma manual quanto automatizada (via TrueState-SNA).

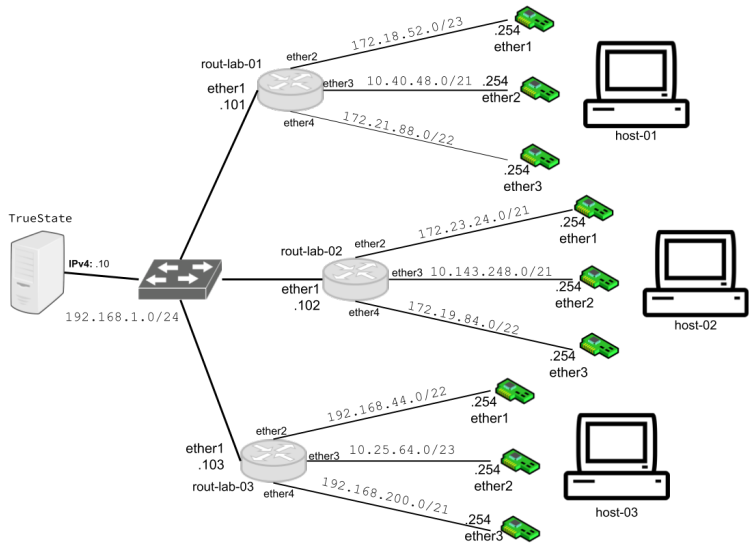


Figura 6. Topologia e endereçamentos do experimento.

Os operadores receberam como parte das instruções a topologia da Figura 6 e a Tabela 1, que apresenta os dispositivos, interfaces, endereços IPv4 e IPv6 dos enlaces ponto a ponto entre os roteadores e os dispositivos. Essa topologia está disponível como um *Appliance* do VirtualBox no repositório da TrueState-SNA.

Observe-se na Tabela 1 que os prefixos IPv4 dos enlaces ponto a ponto entre os roteadores e os dispositivos possuem máscaras de sub-rede distintas. Nesses prefixos, os roteadores utilizam o primeiro endereço IP, enquanto os clientes utilizam o último endereço, forçando a configuração correta da máscara de rede. Todos os objetos foram previamente criados na *SSoT*, porém as associações entre endereços e dispositivos não foram realizadas. Esses objetos geralmente são criados durante as fases de planejamento de uma mudança, e o objetivo do experimento é avaliar o tempo consumido nas tarefas necessárias para a implementação de uma mudança.

Tabela 1. Configuração de IPs desejada após o termino do experimento.

Roteador	Interface	IP	Rede	IP	Interface	Host
rout-lab-01	ether2	172.18.52.1/23	enlace1	172.18.53.254/23	ether1	host-01
	ether3	10.40.48.1/21	enlace2	10.40.55.254/21	ether2	
	ether4	172.21.88.1/22	enlace3	172.21.91.254/22	ether3	
rout-lab-02	ether2	172.23.24.1/21	enlace4	172.23.31.254/21	ether1	host-02
	ether3	10.143.248.1/21	enlace5	10.143.255.254/21	ether2	
	ether4	172.19.84.1/22	enlace6	172.19.87.254/22	ether3	
rout-lab-03	ether2	192.168.44.1/22	enlace7	192.168.47.254/22	ether1	host-03
	ether3	10.25.64.1/23	enlace8	10.25.65.254/23	ether2	
	ether4	192.168.200.1/21	enlace9	192.168.207.254/21	ether3	

Para configurar manualmente cada endereço IP nas interfaces dos roteadores, o operador deve se autenticar a cada um dos roteadores via SSH e executar uma sequência de comandos para atribuir os IPs da Tabela 1 às respectivas interfaces de cada dispositivo. Para avaliar o tempo consumido nas tarefas de configuração manual dos dispositivos, disparou-se o *script time_untill_ping_all.py*, que monitora, por meio de *pings*, uma lista com os 9 endereços IP dos clientes. O script armazena os tempos decorridos desde o início do experimento até que cada um dos IPs dos clientes se torne acessível a partir de um sensor configurado na rede de gerência.

Para configurar automaticamente cada endereço IP nas interfaces dos roteadores, o operador realiza diretamente na TrueState-SNA os seguintes passos: (1) acessar a página do endereço IP, (2) clicar em editar, (3) associar a interface do dispositivo e (4) salvar as alterações.

3.2. Resultados

Os resultados contendo a média após 3 repetições dos tempos (em segundos), parciais e totais, consumidos nas tarefas de configuração dos roteadores através de configuração manual e automatizada, estão expressos na Tabela 2.

Tabela 2. Tempo para os hosts tornarem-se acessíveis via configuração manual.

Dispositivo	Interface	IP	TrueState-SNA	Manual
host-01	ether1	172.18.53.254	20,04 s	60,14 s
	ether2	10.40.55.254	35,09 s	60,14 s
	ether3	172.21.91.254	80,21 s	75,20 s
host-02	ether1	172.23.31.254	95,24 s	105,27 s
	ether2	10.143.255.254	115,31 s	125,33 s
	ether3	172.19.87.254	125,34 s	145,38 s
host-03	ether1	192.168.47.254	155,40 s	175,44 s
	ether2	10.25.65.254	175,43 s	195,47 s
	ether3	192.168.207.254	190,45 s	205,49 s
Tempo total			190,46 s	205,50 s

Os resultados apresentados na Tabela 2 demonstram a redução no tempo de operação das tarefas executadas diretamente na SSoT, em relação à execução manual de comandos. A diferença total de tempo foi de aproximadamente 15 segundos para a configuração de 9 endereços IP em cada interface dos 3 dispositivos. Podemos deduzir a partir da Tabela 2 que o tempo de configuração manual para cada roteador foi de 75 segundos para o *rout-lab-01* se tornar totalmente acessível, 70 segundos para o *rout-lab-02* e 60 segundos para o hout-lab-03, uma média de aproximadamente 68 segundos por roteador. Observando os tempos consumidos pela configuração automática, foi registrada uma média de 63 segundos para cada roteador, representando uma redução de 10% no tempo.

É possível verificar, porém, que o consumo de tempo para a configuração do *rout-lab-01* ficou em torno de 80 segundos, o que representa praticamente o dobro do tempo médio da configuração do *rout-lab-02* e *rout-lab-03*, que ficaram em 40 segundos. São necessários mais testes e análises para determinar se as causas na variação dos tempos foram causadas pelo comportamento do operador ou pela aplicação, SSoT ou TrueState-SNA. Se considerarmos apenas os menores tempos de configuração manual e automática, que foram 60 segundos para o *rout-lab-03* na configuração manual e 45 segundos para o rout-lab-02 na configuração automática, a redução do consumo de tempo chega próximo a 25%.

Outro aspecto a ser observado na Tabela 2 é que, entre a configuração manual de um roteador e outro, são observados em média 30 segundos. Já os tempos médios consumidos para a ativação dos demais IPs do mesmo roteador são reduzidos para intervalos de 20 segundos. Esses resultados podem ser explicados pelo fato de o sistema operacional dos roteadores demandar apenas 1 linha de comando para configurar um endereço IP em uma interface. Porém, outros dispositivos, da Cisco e Huawei, por exemplo, demandam no mínimo duas linhas de comando para atribuir um endereço IP e, para cada novo atributo, geralmente, é necessária uma nova linha de comando. É razoável supor que a configuração automatizada pela TrueState-SNA se torne ainda mais vantajosa a partir do suporte a equipamentos de outros fabricantes que exigem mais linhas.

4. Considerações Finais

A solução TrueState-SNA simplifica a gestão de redes heterogêneas, promovendo a automação de mudanças nos estados operacionais a partir da declaração dos estados desejados na SSoT. Essa abordagem não apenas reduz o tempo de operação e os erros humanos, mas também diminui o risco de *lock-in* tecnológico, aumentando a flexibilidade operacional em ambientes heterogêneos. Além disso, a integração nativa com ferramentas como *NetBox* dispensa a necessidade de *plugins* adicionais, simplificando a implementação e reduzindo a complexidade do sistema.

As perspectivas de trabalhos futuros podem ser organizadas em três eixos principais. Primeiro, abordagens arquiteturais, como a incorporação de princípios de Zero Trust Networking (ZTN), Zero-Touch Service Management (ZSM), Software-Defined Networking (SDN) e Intent-Based Networking (IBN), além da integração de inteligência artificial para análise preditiva e otimização autônoma de redes. Segundo, análises de desempenho e segurança, englobando avaliações de escalabilidade (com orquestração de contêineres via Kubernetes), interoperabilidade entre sistemas e auditoria de segurança em todas as camadas da solução. Por fim, ampliação tecnológica, contemplando a inclusão de suporte multiforncedor, adoção de padrões emergentes (como redes 5G/6G) e implementação de protocolos avançados (como VLANs 802.1Q, Spanning Tree Protocol e protocolos de roteamento OSPF/BGP),

Agradecimentos. Esta pesquisa contou com apoio parcial da CAPES (Código de Financiamento 001), da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS), por meio dos editais 02/2022, 08/2023, 09/2023 e dos termos de outorga 24/2551-0001368-7 e 24/2551-0000726-1, e da FAPESP (processos 2020/05183-0 e 2023/00816-2).

Referências

Benzaid, C. and Taleb, T. (2020). AI-driven zero touch network and service management in 5g and beyond: Challenges and research directions. *IEEE Network*, 34(2):186–194.

Cisco (2024). 2024 global networking trends. Accessed: 2025-02-25.

Clemm, A., Ciavaglia, L., Granville, L. Z., and Tantsura, J. (2022). Rfc 9315: intent-based networking-concepts and definitions.

Errea, J., Quang, H. T., Verchere, D., Thieu, H. T., Mazzini, A., Abnaou, L., Pesic, J., Curtol, M., Imadi, A. E., Ksentini, A., and Zeghlache, D. (2023). Open disaggregated optical network control with network management as code. In *Optical Fiber Communication Conference (OFC) 2023*, page M3Z.8. Optica Publishing Group.

Gallego-Madrid, J., Sanchez-Iborra, R., Ruiz, P. M., and Skarmeta, A. F. (2022). Machine learning-based zero-touch network and service management: A survey. *Digital Communications and Networks*, 8(2):105–123.

Hamadanian, P., Arzani, B., Fouladi, S., Kakarla, S. K. R., Fonseca, R., Billor, D., Cheema, A., Nkposong, E., and Chandra, R. (2023). A holistic view of ai-driven network incident management. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, pages 180–188.

Karvinen, T. (2023). *Configuration management of distributed systems over unreliable and hostile networks*. PhD thesis, University of Westminster.

Khan, T. A., Akbar, W., Muhammad, A., and Song, W.-C. (2022). Proactive intent policy activation: An ml-assisted resource forecasting approach. In *2022 23rd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–6.

Lampropoulos, G., Passas, N., Merakos, L., and Kaloxylos, A. (2005). Handover management architectures in integrated wlan/cellular networks. *IEEE Communications Surveys & Tutorials*, 7(4):30–44.

Landman, R., Hoogendoorn, S., Westerman, M., Hoogendoorn-Lanser, S., and Van Kooten, J. (2010). Design and implementation of integrated network management in the netherlands. In *TRB 89th Annual Meeting Compendium of Papers DVD*.

Leivadeas, A. and Falkner, M. (2022). A survey on intent-based networking. *IEEE Communications Surveys & Tutorials*, 25(1):625–655.

Li, M. and Sandrasegaran, K. (2005). Network management challenges for next generation networks. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*, pages 6 pp.–598.

Linder, S., Lisetska, P., and Stutz, R. (2024). *Network Configuration Automation with Infrahub and Nornir*. PhD thesis, OST Ostschweizer Fachhochschule.

Lingga, P., Kim, J. J., and Jeong, J. P. (2022). Intent-based network management in 6g core networks. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 760–762.

Liu, J., Shi, Y., Fadlullah, Z. M., and Kato, N. (2018). Space-air-ground integrated network: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2714–2741.

Miao, Y., Cheng, Z., Li, W., Ma, H., Liu, X., and Cui, Z. (2016). Software defined integrated satellite-terrestrial network: A survey. In *International conference on space information network*, pages 16–25. Springer.

Mulyana, E. and Fakih, G. (2022a). Network automation with a single source of truth in a heterogeneous environment. *International Journal on Electrical Engineering & Informatics*, 14(1).

Mulyana, E. and Fakih, G. (2022b). Network automation with a single source of truth in a heterogeneous environment. *International Journal of Electrical Engineering and Informatics*, 14(1):6.

Riftadi, M. and Kuipers, F. (2019). P4i/o: Intent-based networking with p4. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 438–443. IEEE.

Shahjee, D. and Ware, N. (2022). Integrated network and security operation center: A systematic analysis. *IEEE Access*, 10:27881–27898.

Siddiqi, S. J., Naeem, F., Khan, S., Khan, K. S., and Tariq, M. (2022). Towards ai-enabled traffic management in multipath tcp: A survey. *Computer Communications*, 181:412–427.

Walther, D. and Jovicic, D. (2023). Ssot based network service deployment. Other thesis, OST Ostschweizer Fachhochschule. Bachelor Thesis.

Wang, Z., Zhang, F., Yu, Q., and Qin, T. (2021). Blockchain-envisioned unmanned aerial vehicle communications in space-air-ground integrated network: A review. *Intelligent and Converged Networks*, 2(4):277–294.

Wei, Y., Peng, M., and Liu, Y. (2020). Intent-based networks for 6g: Insights and challenges. *Digital Communications and Networks*, 6(3):270–280.

Yu, H., Rahimi, H., Janz, C., Wang, D., Li, Z., Yang, C., and Zhao, Y. Building a comprehensive intent-based networking framework: A practical approach from design concepts to implementation. 32(3):47.

Zeydan, E. and Turk, Y. (2020). Recent advances in intent-based networking: A survey. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5. IEEE.

Zhang, T., Qiu, H., Mellia, M., Li, Y., Li, H., and Xu, K. (2022). Interpreting ai for networking: Where we are and where we are going. *IEEE Communications Magazine*, 60(2):25–31.

Apêndice 1: Trabalhos relacionados

Em termos operacionais, embora existam diversas soluções para o gerenciamento centralizado de infraestruturas de rede, a maioria desses sistemas restringe-se ao suporte de dispositivos de um único fabricante. Essa limitação acarreta um elevado risco de dependência técnica, ou *lock-in*, o que reduz a flexibilidade da rede e aumenta os custos operacionais.

Nos últimos anos, os trabalhos científicos sobre automação de rede têm ganhado destaque. Tecnologias como *Software-Defined Networking* (SDN) e *Intent-Based Networking* (IBN) possuem arquiteturas amplamente discutidas na literatura acadêmica, servindo como base para uma nova fase de desenvolvimento das tecnologias de automação. Como destacado por [Mulyana and Fakih, 2022b], no passado, a automação de redes era de difícil implementação devido à natureza proprietária dos dispositivos e à demanda limitada por soluções automatizadas. No entanto, esse cenário mudou drasticamente. Atualmente, a escala e a complexidade das redes atingiram um patamar em que a automação tornou-se indispensável, tanto do ponto de vista técnico quanto dos requisitos de negócios.

O crescente nível de heterogeneidade das redes tem impulsionado esforços científicos não apenas em automação, mas também na importância de integrar uma Fonte de Verdade da Rede (SoT) em suas arquiteturas. De acordo com [Cisco, 2024],

para que a automação de rede seja verdadeiramente eficaz, é essencial dispor de uma Fonte de Verdade confiável (SoT) — um repositório centralizado que forneça dados precisos e atualizados sobre a rede, incluindo inventário de dispositivos, parâmetros de configuração e outras informações relevantes. a TrueState-SNA implementa a SoT como base para as ações sobre os dispositivos gerenciados, criando uma arquitetura capaz de alterar o *estado operacional* a partir de mudanças no *estado desejado*. Em outras palavras, o sistema atua sobre os dispositivos com base em alterações realizadas na SoT, garantindo consistência e precisão.

A necessidade de desenvolver a TrueState-SNA, uma solução de software capaz de implementar tecnologias de automação, é corroborada pelo trabalho de [Yu et al.,]. O autor ressalta a carência de implementações práticas da tecnologia IBN, afirmando que, embora o IBN tenha atraído significativa atenção da comunidade acadêmica e industrial na última década, resultando em inúmeras publicações científicas, ainda há uma lacuna em propostas de estruturas abrangentes que convertam conceitos teóricos em implementações de software completas. a TrueState-SNA não busca implementar exclusivamente IBN ou SDN, mas sim aproveitar os melhores aspectos de cada tecnologia para integrar de forma eficiente o *estado desejado* ao *estado operacional*.

Alguns trabalhos já exploram a integração da SoT em suas arquiteturas, como em [Walther and Jovicic, 2023], que discute workflows para implantação de serviços baseados no NetBox. A solução proposta no trabalho consiste em duas partes: o plugin Argos-NetBox, que estende as funcionalidades do NetBox, e o Argos-NAC, responsável por consultar os dados do NetBox e gerenciar a geração e implantação das configurações dos dispositivos. No entanto, essa abordagem trata o processo de configuração como uma atividade sob demanda, sem uma integração direta entre a SoT (*estado desejado*) e a configuração dos dispositivos (*estado operacional*).

A TrueState-SNA diferencia-se dessas abordagens ao permitir que alterações nas configurações no NetBox resultem automaticamente em mudanças na configuração dos dispositivos gerenciados. Além disso, as funcionalidades utilizadas para integrar a SSoT às demais camadas da arquitetura são nativas do NetBox, dispensando a necessidade de plugins adicionais. Essa característica simplifica a implementação e reduz a complexidade do sistema, mantendo a robustez e a confiabilidade da automação.