UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CARLOS ANDRÉ RODRIGUES DA SILVA

Aplicação de Comando e Controle Ciente do Estado da Rede Através de Suporte SDN

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre
2024

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Silva, Carlos André Rodrigues da

Aplicação de Comando e Controle Ciente do Estado da Rede Através de Suporte SDN / Carlos André Rodrigues da Silva. – Porto Alegre: PPGC da UFRGS, 2024.

59 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2024. Orientador: Edison Pignaton de Freitas.

1. C2, *DTN*, *ICN*, *ONOS*, Mininet, *SDN*. I. Freitas, Edison Pignaton de. II. Título.

AGRADECIMENTOS

Agradeço a Deus em primeiro lugar, e em segundo a minha família por ter me dado todo o suporte durante o período de estudo. Quero agradecer também a meu orientador Professor Edison Pignaton pelo tempo dedicado e os ensinamentos passados. Da mesma forma agradeço também aos Professores Júlio do Anjos e Juliano Wickboldt e aos amigos Márcio e Lauro.

RESUMO

As aplicações de Comando e Controle (C2) apresentam requisitos de desempenho específicos que variam conforme a criticidade da situação. Em um contexto de operação militar, os parâmetros da rede estão em constante mudança devido às condições desafiadoras do ambiente. Portanto, satisfazer os requisitos da aplicação é uma tarefa complexa. Este estudo propõe uma estrutura para superar esse desafio, utilizando uma abordagem de reconhecimento do estado da rede. Nesta abordagem, as aplicações de C2 adaptam seu da rede para atender aos seus requisitos, utilizando a orquestração de Rede Definida por Software (SDN) em conjunto com mecanismos de Rede Centrada em Informação (ICN). Os resultados obtidos fornecem evidências convincentes do sucesso da abordagem proposta em atender aos requisitos necessários de adequação do uso da rede de forma a vialibizar o cumprimento da missão.

Palavras-chave: C2, *DTN*, *ICN*, *ONOS*, Mininet, *SDN*.

ABSTRACT

Command and Control (C2) applications present specific performance requirements that vary depending on the criticality of the situation. In a military operation context, network parameters are constantly changing due to challenging environmental conditions. Therefore, satisfying application requirements is a complex task. This study proposes a framework to overcome this challenge using a network state recognition approach. In this approach, C2 applications adapt the network to meet their requirements, using Software Defined Networking (SDN) orchestration in conjunction with Information Centric Networking (ICN) mechanisms. The obtained results provide convincing evidence of the success of the proposed approach in meeting the requirements necessary for the successful execution of the mission.

Keywords: C2, ICN, SDN.

LISTA DE ABREVIATURAS E SIGLAS

ACL	Lista de controle de acesso
API	Application programming interface
ARP	Address Resolution Protocol
CCSDS	Comitê Consultivo para Sistema de Dados Espaciais
CLI	Command-line interface
C2	Comando e Controle
CS	Content Store
CSRF	Cross Site Request Forgery
DPDK	Data Plane Development Kit
DTN	Delay tolerant networking
FIB	Forwarding Information Base
GIS	Geographical Information System
GPS	Global Positioning System
GUI	Graphical user interface
HTB	Hierarchical Token Bucket
ICMP	Internet Control Message Protocol
ICN	Information-centric networking
IoBT	Internet of Battlefield Things
IP	Internet protocol
IPv4	Internet protocol Versão 4
IBN	Intent-based networking
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
LAN	Local Area Network

LAT	Latitude
LLDP	Link Layer Discovery Protocol
LONG	Longitude
MPLS	Multiprotocol Label Switching
NEC	Network Enterprise Center
NLSR	Named-data Link State Routing Protocol
NDN	Named data networking
NFD	NDN Forwarding Daemon
ONOS	Open Network Operating System
OODA	Observe, Oriente, Decida e Aja
ORM	Object Relational Mapping
OVS	Open Virtual Switch
OVSDB	Open Virtual Switch Database Management
OTAN	Organização do Tratado do Atlântico Norte
PIT	Pending Interest Table
QoS	Quality of Service
REST	Representational state transfer
RSPAN	Remote SPAN
SDN	Software Defined Network
SDQ	Software-Defined Queuing
SDR	Software-Defined Radio
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VBTP	Viatura blindada de transporte de pessoal
VLAN	Virtual LAN

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O século XXI trouxe um aumento na complexidade das operações militares, e algumas razões são responsáveis por este cenário: a velocidade com que os eventos acontecem e a necessidade de informações precisas e oportunas (STOCCHERO et al., 2023a). A velocidade com que os eventos se desenrolam no campo de batalha é influenciada por vários fatores, incluindo a natureza do conflito, as tecnologias empregadas e as estratégias adotadas pelos comandantes. Ter informações precisas e oportunas no campo de batalha aumenta a consciência situacional dos comandantes e é consequentemente essencial para o planejamento estratégico, a tomada das melhores decisões e o sucesso geral da missão (LEE; BAEK JEANY SON, 2023).

Em paralelo com esses fatores, foram projetados sistemas para auxiliar no gerenciamento e na tomada de decisão dos comandantes, conhecidos como Sistemas de Comando e Controle (C2) (ALBERTS et al., 2014). Eles se destacam pela sua agilidade e alto desempenho. O C2 se tornou uma ferramenta indispensável para apoiar os comandantes no processo de tomada de decisão (ZHOU et al., 2020).

A velocidade e a priorização da transmissão de dados são fatores cruciais para alcançar a agilidade na tomada de decisão. Mas como é possível atingir esses objetivos quando há competição pelos recursos de rede entre vários nós usando simultaneamente a rede e a largura de banda é insuficiente para atender a todas as necessidades? É possível otimizar o desempenho de uma rede C2 utilizando *SDN*, por meio de configurações de *QoS* que mapeiem as necessidade de alto nível de uma aplicação C2? Há benefício em se utilizar a integração de *SDN* com *ICN* para se atender às necessidades de adequação da rede para aplicações C2?

A Agilidade de Comando e Controle aplicada no contexto das operações militares pode ser definida como uma estratégia para alcançar oportunidades relevantes mais rápido do que o inimigo. Seguindo as características necessárias para obter uma Agilidade de C2, este trabalho implementa um *framework* chamado C2-APP, que se integra com tecnologias como: *Open Network Operating System (ONOS), OpenFlow, Open vSwitch e MiniNDN* para simular uma rede no campo de batalha com conectividade a dispositivos que fazem parte do equipamento de soldados ou presentes em veículos blindados ou em drones e dispositivos sensores.

Os principais objetivos deste trabalho são:

- Propor um *framework* que permita o ajuste de parâmetros de rede baseados em re-

quisitos de aplicações de Comando e Controle conforme as necessidades da missão;

- Instrumentar a reconfiguração de uma rede *SDN* por meio de mudanças nas configurações de políticas de rede de acordo com a mudança situacional do campo de batalha;
- Estabelecer integração eficiente do sistema C2-APP com dispositivos da Internet das Coisas do Campo de Batalha (*IoBT*) utilizando uma arquitetura *REST API*;
- Utilizar *QoS* (Qualidade de Serviço) para implementar priorizações de determinadas rotas no campo de batalha que mapeiem os critérios definidos nas políticas de atendimento dos requisitos de alto nível das aplicações C2.

Este trabalho está organizado em 7 capítulos. Além desta introdução, o conteúdo é apresentado da seguinte forma:

No Capítulo 2 é apresentada a revisão bibliográfica, com a apresentação das tecnologias, abordagens e ferramentas utilizadas para a realização deste trabalho.

O Capítulo 3 aborda os trabalhos relacionados, onde é feita a comparação e a diferença entre eles e este trabalho.

No Capítulo 4 é detalhado o projeto da aplicação ciente do estado da rede. O projeto mostra a estrutura do trabalho, a apresentação do aplicativo C2-APP, os seus módulos e a forma de integração com a rede, e a dimensão da orquestração, onde mostra o controlador *ONOS* e suas possibilidades de aplicação.

No Capítulo 5 é demonstrada a metodologia dos experimentos realizados detalhando as configurações utilizadas.

No Capítulo 6 são apresentados os resultados e discussões sobre os resultados alcançados.

O Capítulo 7 conclui o trabalho, destacando as principais contribuições e limitações be como apresentando sugestões de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Esse capítulo apresenta as principais tecnologias e conceitos que foram utilizadas como base deste trabalho.

2.1 Comando e controle

O chamado ciclo da Tomada de decisão de Boyd, consistia nos verbos: observe, oriente, decida e aja *(OODA)*. O criador do processo foi o coronel da Força Aérea John Boyd na observação de pilotos em combate. O seu objetivo era ajudar pilotos de caça a tomarem decisões adequadas de forma rápida. Mais adiante, esse padrão foi aceito como modelo de processo de negócio para o Comando e Controle (C2). O *Loop OODA* pregava que qualquer indivíduo ou organização que seja bem-sucedido e complete o ciclo *OODA* mais rápido que seu oponente vencerá o conflito (ZHAO et al., 2023).

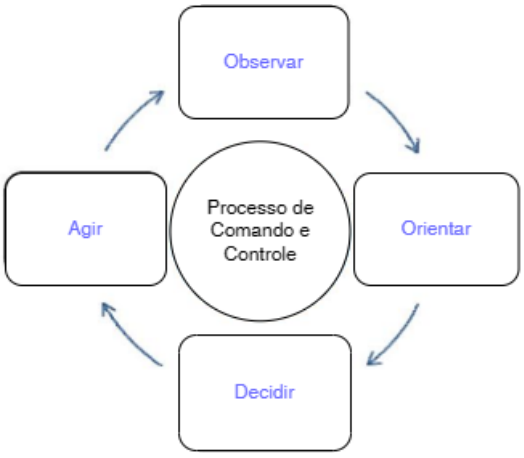

Figura 2.1 – *Loop OODA* Simples para Processo C2 (RéVAY; LíšKA, 2017).

- Observar: é o processo de coletar informações sobre o ambiente e interagir com ele.
- Orientar: após observar o ambiente, a etapa de orientação propõe possíveis soluções. É feita a avaliação de dados e fatos, além de experimentos e testes.
- Decidir: é o processo de selecionar entre uma série de hipóteses sobre a situação ambiental e escolher a solução mais adequada ao contexto.
- Agir: é a aplicação da solução escolhida.

Segundo (RéVAY; LíšKA, 2017), dois fatores são cruciais para a compreensão do C2 no século XXI: a natureza da missão e a transformação contínua das instituições militares do século passado. Atualmente, a natureza da missão é severamente imprevisível

e incerta, e as ameaças não são mais convencionais, mas sim de outros níveis, cada vez mais tecnológicas e mortais. Nesse contexto, o uso da tecnologia criou outro patamar nos conflitos. O emprego do C2 não se restringe apenas a operações de combate, mas também se expandiu para operações de apoio em catástrofes e conflitos internos. A Organização do Tratado do Atlântico Norte (*OTAN*) classificou o *Network Enterprise Center (NEC)* como uma alta prioridade para seus países membros. Em termos gerais, o *NEC* é a capacidade de obter e compartilhar a maior quantidade de informações relevantes em um só local, visando conscientização e compreensão. Isso é, um grande Centro de Controle com informações de seus países membros.

Outros importantes conceitos empregados no C2 são os domínios de combate. A Figura 2.2 mostra os quatro domínios de combate e como eles estão sobrepostos e ligados entre si.

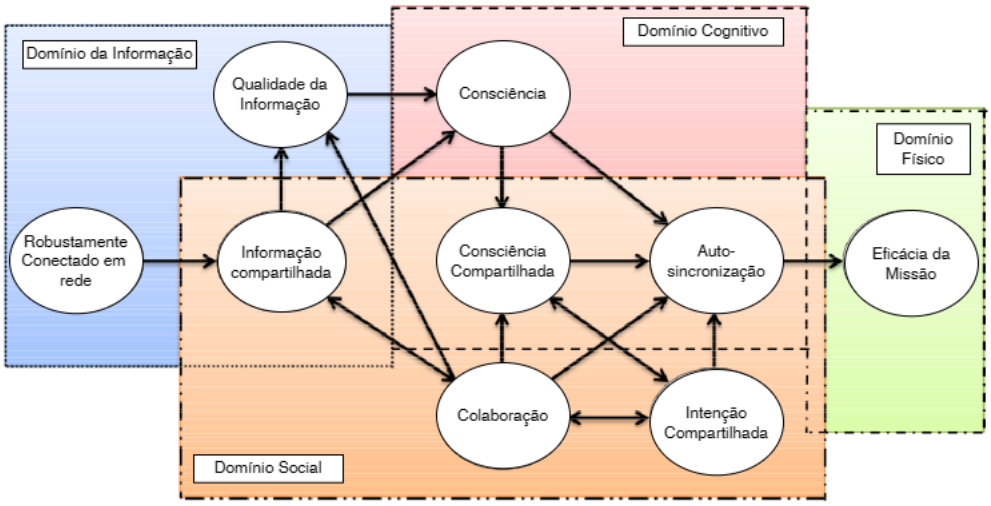

Figura 2.2 – Domínios de combate (RéVAY; LíšKA, 2017).

- Domínio físico: é o domínio que pertencente a uma guerra convencional. Ele utiliza os ambientes terrestres, aéreos, marítimos e espaciais. É o local onde se realizam operações militares, os dispositivos se conectam e onde se mede o poder de combate.
- Domínio informativo: é o domínio onde é criada, manipulada e transformada a informação. Ele deve facilitar a transferência de informações entre os combatentes. Por questões de segurança, deve ser defendido e protegido da ofensiva do inimigo.
- Domínio cognitivo: é aquele domínio que está presente na mente dos participantes. A maioria das vitórias passam por esse domínio. Para obter êxito, geralmente sempre apresenta as características de liderança, moral, coesão, nível de treinamento e experiência.

- Domínio social: é o domínio obtido pelo comportamento humano. Ele reflete as atitudes e valores de cada indivíduo e inclui também a disciplina e a comunicação.

Para (LEAL, 2020), as operações militares modernas mudaram o conceito das guerras, com o emprego da tecnologia no campo de batalha amplificou-se a importância do Comando e Controle (C2) e com isso, as necessidades de redes eficientes para o sistema. Um sistema C2 tem a função de gerenciar todos os dados referentes a uma missão, a qual está sendo empregado. Para conseguir isso necessita receber as informações do máximo de dispositivos presentes no combate.

2.2 *IoBT*

Segundo (KOTT; SWAMI; WEST, 2016), na última década surgiu o conceito da Internet das Coisas (*IoT*), que trouxe a possibilidade da conexão de dispositivos, objetos e pessoas numa mesma rede. Durante esse período desenvolveu-se uma indústria em proveito da criação de aplicativos, protocolos e pesquisas para facilitar a vida do ser humano. A integração da *IoT* com a Inteligência Artificial *(IA)* possibilitou a automação de necessidades da vida, como, por exemplo, a identificação da falta de produtos em uma geladeira. Como a geladeira é um objeto que faz parte de uma rede *IoT*, ela é configurado para se conectar a internet e comprar os produtos que estão faltando, facilitando assim a vida do homem.

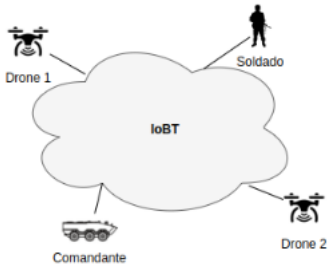

Figura 2.3 – *IoBT*

Baseada na Internet das Coisas, foi criada a Internet das Coisas do Campo de Batalha (*IoBT*). Ela surgiu devido ao desenvolvimento tecnológico de dispositivos móveis e comunicação sem fio aliados à necessidade crescente de conectar esses diversos dispositivos no campo de batalha. O desenvolvimento da *IoBT* possibilitou a conexão de diversos dispositivos presentes em navios, aviões, veículos blindados, soldados e em drones, sensores e tantos outros. O objetivo da *IoBT* é conectar e compartilhar informações produzidas pelos dispositivos no campo de batalha, para obter vantagens no cumprimento

da missão (PAPAKOSTAS et al., 2021). A Figura 2.3 mostra a interação de elementos presentes no Campo de Batalha com a rede representada pela nuvem *IoBT*.

De acordo com (YU et al., 2022), a Internet das Coisas de Batalha (*IoBT*) proporcionou avanços significativos, especialmente no que diz respeito à coordenação de ataques e à desminagem do campo de batalha. A coordenação de ataques foi aprimorada pela capacidade de gerenciar de forma mais eficaz as forças militares e seus dispositivos conectados. Isso resultou em operações mais inteligentes e eficientes. Além disso, a *IoBT* permitiu que a desminagem de uma área de campo de batalha fosse realizada por veículos não tripulados ou drones, reduzindo assim o risco de perda de vidas humanas. Esses avanços demonstram o potencial transformador da *IoBT* no campo de batalha.

Segundo (STOCCHERO et al., 2023b), as redes de Internet das Coisas de Batalha (*IoBT*) apresentam características distintas em relação às redes de Internet das Coisas (*IoT*) tradicionais, devido aos desafios extremos que enfrentam no campo de batalha. Estes desafios incluem a heterogeneidade dos dispositivos utilizados, o ambiente hostil em que operam e as constantes ameaças cibernéticas.

2.3 *Software Defined Network (SDN)*

A concepção do *SDN* foi introduzida em junho de 2009 na Universidade de Stanford, nos Estados Unidos, como resultado do trabalho realizado com a tecnologia *OpenFlow* em 2008 (PALIWAL; SHRIMANKAR; TEMBHURNE, 2018).

O surgimento do *SDN* foi motivado pela necessidade de automatização e otimização das redes, visando aprimorar as aplicações de serviços e armazenamentos em bancos de dados. Além disso, o *SDN* acompanhou as mudanças tecnológicas de provedores de serviços e operadoras nos últimos anos.

O *SDN* é uma arquitetura de rede moderna que simplifica a manipulação dos dispositivos e a programabilidade da rede. Isso permite o gerenciamento de todo o sistema, melhorando o desempenho, a confiabilidade e a qualidade do serviço. O *SDN* trouxe uma flexibilidade e programabilidade sem precedentes em redes de computadores, facilitando assim o seu gerenciamento.

Atualmente, o *SDN* é uma tecnologia em evidência que abre novas possibilidades de projetos e técnicas de rede, além de enriquecer pesquisas e experimentações. Em comparação com as redes tradicionais, o *SDN* pode oferecer diversas vantagens, como a redução de custos, a melhoria do desempenho da rede, o controle em tempo real e o

gerenciamento centralizado.

A principal característica da Rede Definida por Software é a separação do plano de controle do plano de dados. No plano de controle, o controlador é o centro da *SDN*. Ele unifica e centraliza o controle de todos os dispositivos de rede, como switches, roteadores, balanceadores de cargas, etc. No plano de dados, o tráfego é encaminhado segundo as decisões do plano de controle. O plano de dados é representado pelos dispositivos físicos, como switches, hosts, sensores e qualquer outro tipo de dispositivo que faça parte da rede *SDN* (HASAN et al., 2020).

Conforme a Figura 2.4, o *SDN* possui a separação entre o plano de dados e o plano de controle. Esta separação proporcionou características bem diferentes das redes tradicionais, tornando-as uma opção relevante para implementações. As principais características das Redes Definidas por Software são:

- Gerenciamento e programação de maneira centralizada: foi possível com a separação das funções de encaminhamento e de controle de rede;
- Redução de custos: as principais soluções *SDN*, como o controlador ONOS, não necessitam da aquisição de soluções de hardware proprietárias para sua utilização;
- Flexibilidade e escalabilidade: com a virtualização da infraestrutura de rede, é possível ampliar e contrair recursos de rede conforme e quando necessário;
- Gerenciamento simplificado: a *SDN* torna as políticas de rede mais fáceis de serem implementadas, removidas e modificadas.

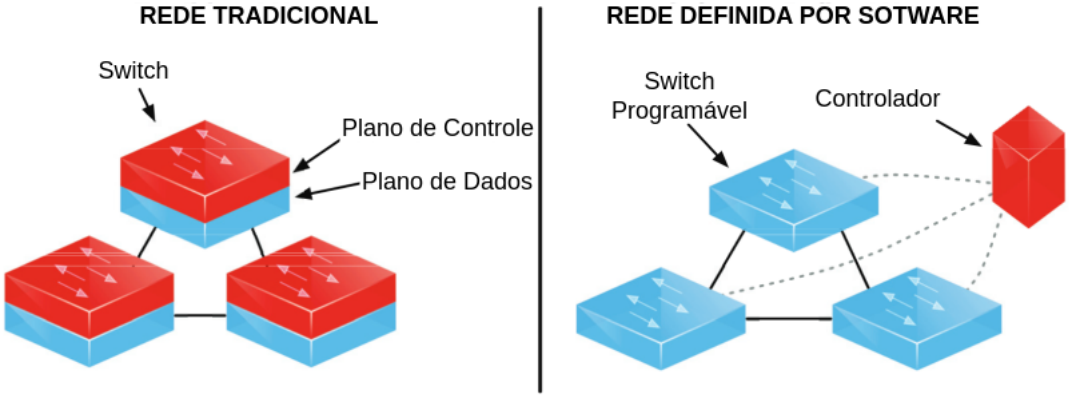

Figura 2.4 – Comparativo de Arquiteturas (JHA, 2017).

2.4 *OpenFlow*

O *OpenFlow* é um protocolo de comunicação amplamente adotado que estabelece uma conexão entre o Plano de Dados (composto por switches ou roteadores) e o Plano de Controle (controlador *SDN*). Introduzido em 2008, o *OpenFlow* é reconhecido como um dos protocolos fundamentais no cenário das Redes Definidas por Software (*SDN*).

O *OpenFlow* desempenha as seguintes funções essenciais:

- Permitir que os controladores de rede determinem as rotas de pacotes através dos switches ou roteadores na rede;
- Oferecer a possibilidade de gerenciar o tráfego usando Lista de Controle de Acesso (*ACL*) e protocolos de roteamento;
- Proporcionar a compatibilidade para o gerenciamento remoto de switches de diferentes fornecedores. Esta característica é particularmente útil, pois cada fornecedor geralmente possui suas próprias interfaces proprietárias e linguagens de *scripts*;
- Facilitar a implementação de Redes Definidas por Software (*SDN*) ao permitir a adição, modificação e remoção de regras e ações em pacotes de dados.

O protocolo *OpenFlow* foi lançado inicialmente na versão 1.1 em fevereiro de 2011, e a versão mais recente, 1.5.1, foi disponibilizada em março de 2015. As principais características suportadas pelas diferentes versões do *OpenFlow* incluem:

- v1.1.x: Suporte para *MPLS, VLAN, multipath,* tabelas de grupo, tabelas *multiflow* e filas;
- v1.2.x: Introdução de *In-match*, *Packet-In*, cabeçalhos extensíveis, correspondência de fluxo flexível e IPV6;
- v1.3.x: Adição de tunelamento, tabelas de medidores e campos adicionais nas tabelas de fluxo;
- v1.4.x: Inclusão de gerenciamento de portas ópticas, tabelas sincronizadas, despejo e monitoramento de fluxo de tabelas (prioridade, tempos limite, *cookie*, sinalizadores);
- v1.5.x: Implementação de tabelas de saída, estatísticas de entrada de fluxo, pipeline com reconhecimento de tipo de pacote.

2.5 *ICN*

As redes *NDN* são baseadas no paradigma das redes centradas na informação (*ICN*). Elas seguem dois conceitos fundamentais do *ICN*: o cache universal e o acesso ao conteúdo pelo nome. O sucesso das redes táticas dependem da capacidade de comunicações e troca de informações entre o comando e os nós da rede. C2-App tem a possibilidade de fazer consultas as informações produzidas pelos nós da rede. Estes nós podem ser equipamentos conduzidos por soldados, drones, sensores ou outros equipamentos pré-definidos. Eles podem possuir a capacidade de produzir vídeos, imagens, áudios e documentos a qualquer momento no campo de batalha. Um nó *NDN* quando recebe um pacote de interesse do consumidor, realiza uma busca em seu Content Store (*CS*) pelo conteúdo solicitado; se a solicitação corresponder aos dados disponíveis no *CS*, os dados interessados serão encaminhados ao consumidor; caso contrário, o nome do conteúdo, juntamente com sua interface de entrada associada (informações do nó vizinho do qual o pacote de interesse foi recebido), é registrado no Pending Interest Table (*PIT*). O *PIT* também realiza uma pesquisa para a entrada do nome do conteúdo. Se o nome do conteúdo já existir no *PIT*, indicará que existem outros consumidores esperando o conteúdo, então a interface de entrada desse novo interesse será inserida no conteúdo do *PIT*. As informações de nomes presentes no *PIT* ficarão ativas até o interesse ser atendido ou descartado por expiração de tempo, preempção ou por definição na estratégia. Os pacotes de interesse seguem sendo encaminhados e se repetem de nó em nó até que o conteúdo procurado seja encontrado. Quando o conteúdo é localizado, o pacote de dados (com a solicitação do consumidor) retorna pelo caminho inverso até alcançar os consumidores. O *CS* dos nós *NDN* ao longo do caminho pode decidir armazenar em cache o conteúdo, dependendo de sua estratégia de controle. Toda vez que um consumidor solicita um conteúdo, ele não necessita ser atendido diretamente pelo produtor dos dados. Os outros nós intermediários podem ter armazenados em cache os dados em seu *CS* e, assim, os dados podem ser enviados diretos de um (S.; BOREGOWDA, 2022).

Este trabalho utiliza o *NDN* como forma de fazer solicitações aos nós da rede *SDN*. Uma determinada área de interesse pode ser escolhida por C2-APP diretamente no mapa da operação. Por meio de uma ferramenta de seleção é possível definir a área de interesse e quais nós deverão buscar as informações. Todos os nós da rede no campo de batalha recebem a solicitação para a produção de informações da área de interesse e o tempo para o cumprimento da demanda. Tão logo as informações solicitadas forem produzidas, elas

ficaram disponíveis para o consumidor.

Segundo (GIBSON et al., 2017), as redes IP tradicionais possuem limitação na disseminação de dados em larga escala e no atendimento aos requisitos específicos de redes táticas no campo de batalha. Sendo assim, as redes *NDN* surgem como candidatas para arquiteturas de redes mais adequadas para dinâmicas e segurança. As redes *NDN* possuem diversos benefícios em relação à rede *IP* tradicional:

- possui capacidade de entregar e compartilhar um volume maior dados;
- melhora de desempenho de arquivos de *streaming*;
- melhor tempo de resposta e distribuição de conteúdos;
- utiliza os caminhos ociosos da rede para melhorar a entrega de conteúdo.

Nas redes *NDN*, são encontrados exclusivamente dois tipos de pacotes: os de interesse e os de dados. O nó que cria e disponibiliza uma informação ou arquivo é conhecido como produtor, enquanto aquele que solicita um arquivo com um nome específico é denominado consumidor. Uma rede *NDN* possui três estruturas principais:

- *Content Store (CS)*: é uma memória temporária de tamanho variável para pacotes de dados recebidos pelo roteador, ele pode ser armazenado em *buffer* para atendimento a interesses futuros;
- *Pending Interest Table (PIT)*: é uma tabela que armazena interesses insatisfeitos, uma entrada é adicionada quando um novo pacote de interesses chega, e removido quando é satisfeito pelo pacote de dados correspondente. O pacote de interesse insatisfeito é encaminhado para o próximo nó. Uma *FIB* mantém as interfaces dos nós vizinhos;
- *Forwarding Information Base (FIB)*: é uma base de informações de encaminhamento, é uma tabela de transporte que mapeia os nós da rede;

A Figura 2.5 mostra o modelo de funcionamento de um nó *NDN*.

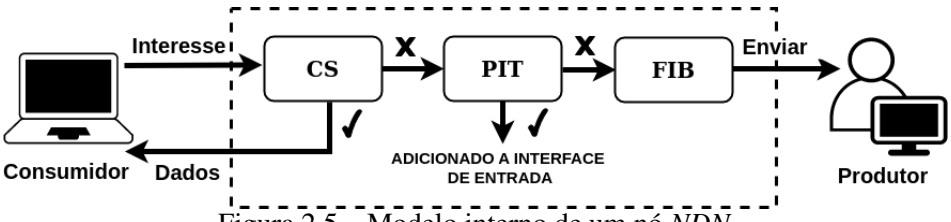

Figura 2.5 – Modelo interno de um nó *NDN*

2.6 Ferramentas utilizadas

2.6.1 *ONOS*

O *Open Network Operating System (ONOS)* é um controlador *SDN* de código aberto, para a criação de soluções de próxima geração. Ele é um sistema distribuído que possui em seu conteúdo diversos dispositivos autônomos que poderão ser ativados em sua rede de comunicação, como mostra a Figura 2.6. O *ONOS* foi projetado para atender às necessidades daqueles desenvolvedores que desejarem construir soluções de alto nível e aproveitarem a economia em relação ao hardware comercial. Possui a flexibilidade para criação e implantação de novos serviços de rede dinâmicos com interfaces programáticas simplificadas.

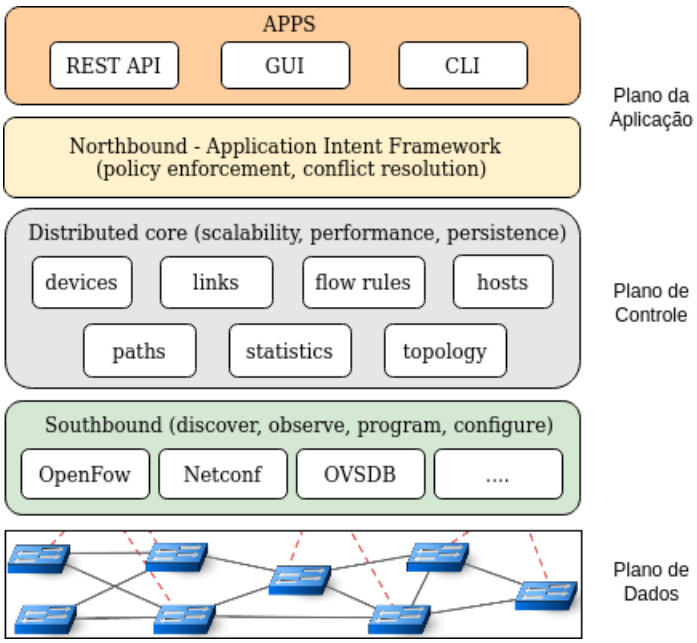

Figura 2.6 – Arquitetura *ONOS SDN* (VACHUSKA, 2022).

Os usuários finais podem criar facilmente novos aplicativos de rede sem a necessidade de alterar os sistemas de plano de dados. A visão de topologia de rede global que ele oferece expõe o status atual de toda a rede à camada de aplicação e possibilita a configuração e o controle em tempo real da rede. Em síntese, o *ONOS* é um sistema operacional de rede de arquitetura *SDN*, que visa alcançar alta escalabilidade, disponibilidade e desempenho, que são características indispensáveis das redes em produção (VACHUSKA, 2022). O controlador ONOS, como mostra a Figura 2.6, se divide em:

- *Northbound*: realiza a comunicação entre o *ONOS* e os aplicativos;

- *Southbound*: realiza a comunicação entre o controlador e os dispositivos do plano de dados;
- *Distributed core*: é onde se localiza o plano de controle;
- *APPS*: possui como principais aplicativos: *REST API, GUI E CLI*.

Na Figura 2.6, o controlador *ONOS*, possui uma arquitetura de três camadas onde a camada inferior é o plano de dados (escravo), a camada intermediária é o plano de controle (centro) e a camada superior é a camada de aplicação.

2.6.2 *Open Virtual Switch*

O *Open Virtual Switch (OVS)* é um *switch* de software multicamada de código aberto. Ele foi desenvolvido para trabalhar em ambiente virtualizado e oferece suporte a diversas tecnologias de virtualização com base no Linux. A versão 3 do *OVS* oferece suporte a diversos recursos como:

- Monitoramento: *NetFlow, sFlow, SPAN e RSPAN*;
- Segurança: *VLAN, isolation e traffic filtering;*
- *QoS: Traffic queuing e traffic shaping;*
- Controle automatizado: *OpenFlow, OVSDB e mgmt, protocol*.

O *OVS* também pode trabalhar no espaço do usuário sem a ajuda de um módulo do *kernel*. O *OVS* no espaço do usuário pode acessar dispositivos *Linux* ou *DPDK (Data Plane Development Kit)*. Possui como características principais: a flexibilidade e capacidade de integração com diversas plataformas e tecnologias, o que o tornou seu uso popular em ambientes virtuais. Como exemplos de flexibilidade e integração podem ser citados:

- Uso de tabelas de fluxo: o OVS armazena em tabelas os quadros e pacotes da rede até que o destino seja decidido. Desta forma é possível definir o gerenciamento do tráfego de rede;
- Integração: é portável para diversas plataformas de virtualização e chipsets de comutação, incluindo KVM, XenServer, Xen Cloud Platform (XCP) e os sistemas operacionais Ubuntu, Debian, Fedora, OpenSuse, FreeBSD e NetBSD entre outros.

2.6.3 *Mininet*

É uma plataforma *open source* que emula uma rede *SDN* mantendo a fidelidade de uma topologia de rede real. É um emulador de rede capaz de criar e executar uma coleção de *hosts*, *switches*, controladores e *links* virtuais. Ele possui as seguintes funcionalidades: criação de topologias de rede personalizadas, interação com programas reais do sistema operacional *linux* e programação de *switches*. Em resumo, o *Mininet* consegue simular uma rede experimental completa. As redes Mininet executam código real, incluindo aplicativos de rede *Unix/Linux* padrão. O código desenvolvido e testado no Mininet, pode ser movido para um sistema real com poucas alterações, para testes reais, avaliação de desempenho e implantação (MAMUSHIANE; SHOZI, 2021). Também possui outras características:

- Permite um desenvolvimento barato e simplificado;
- Permite testes de topologia complexos, sem utilizar uma rede física;
- Prototipagem rápida de redes definidas por software;
- Inclui uma CLI com reconhecimento de topologia para testes;
- Suporta topologias personalizadas arbitrárias e inclui um conjunto básico de topologias parametrizadas;
- Fornece uma *API Python* direta e extensível para criação e experimentação de rede;
- Vários desenvolvedores simultâneos podem trabalhar de forma independente na mesma topologia.

O *Mininet* possui como variações o *Mininet-Wifi* e o *MiniNDN*.

O *Mininet-WiFi* é um emulador para redes sem fio oriundo do *Mininet*. Ele foi criado para preencher uma lacuna importante acerca da experimentação de projetos e pesquisas em redes sem fio. Com ele é possível abordar diversos exemplos práticos, mobilidade, redes *mesh e adhoc*, balanceamento de carga, segurança, Qualidade de Serviço (*QoS*), redes veiculares, Internet das coisas, e muitos outros. O *Mininet-WiFi* possui suporte nativo a *WiFi*, com ele é possível a virtualização de estações, pontos de acesso, *hosts, switches* e controladores.

O *MiniNDN* é uma ferramenta de emulação de rede leve que permite testes, experimentação e pesquisa na plataforma *NDN* baseada em *Mininet*. O *MiniNDN* usa as bibliotecas *NDN, NFD, NLSR* e ferramentas lançadas pelo projeto *NDN* para emular uma rede *NDN* em um único sistema (TEAM, 2022).

3 TRABALHOS RELACIONADOS

A investigação de trabalhos relacionados a este tema foi conduzida em plataformas acadêmicas renomadas na área de tecnologia, como ieeeexplore.ieee.org e researchgate.net, complementada por consultas a periódicos científicos e congressos. A estratégia inicial focou na identificação de trabalhos mais recentes. Contudo, quando não foram encontrados resultados satisfatórios, a pesquisa foi progressivamente ampliada até que um estudo de relevância fosse identificado. Os trabalhos pesquisados abordavam tecnologias pertinentes a este estudo, incluindo: *SDN, ICN, DTN, MiniNet, QoS, NDN, IoBT, C2, OpenFlow e ONOS*. Um trabalho foi considerado relevante se incorporasse mais de uma dessas tecnologias em seu conteúdo.

O estudo conduzido por (ABBOU; SONG, 2021) focou na investigação de recursos tecnológicos para o fornecimento de garantia de *QoS* em redes *SDN*. Como resultado, foi desenvolvido um software denominado *SDQ (Software-Defined Queuing)*, que foi integrado diretamente ao controlador *ONOS*. O *SDQ* era composto por dois módulos principais: gerenciamento de filas e engenharia de tráfego. O módulo de gerenciamento de filas empregava a interface de linha de comando do ONOS (*CLI*) e o protocolo *Open Virtual Switch Database Management (OVSDB)* para administrar as filas nos *switches*. Por outro lado, o módulo de engenharia de tráfego operava sobre as regras de fluxo para estabelecer comunicação com os *switches*. A simulação foi executada em uma topologia de rede, construída no *Mininet*, que consistia em três *switches* e três *hosts*. Quatro categorias de filas de prioridade foram empregadas: baixa, média, alta e melhor esforço. Os experimentos conduzidos demonstraram que a latência se manteve estável, limitada e extremamente baixa para pacotes de tráfego de alta e média prioridade.

O trabalho realizado por (JHA, 2017) buscou implementar mecanismos no controlador *ONOS* para fornecer garantias de *QoS* ao usuário. Para isso, um aplicativo chamado *MyAPP* foi desenvolvido e instalado no *ONOS*. No entanto, o aplicativo não funcionou conforme o esperado após a instalação, pois não conseguiu executar seus comandos diretamente no *OVS*. Em um experimento sequente, descobriu-se que a criação de *QoS* e filas devia ser realizada diretamente no *OVS* por meio de comandos no sistema operacional. Além disso, concluiu-se que as versões do *ONOS* e do *OpenFlow* disponíveis na época do estudo não ofereciam suporte aos medidores do *OVS*. No último experimento, as filas e os medidores foram testados com sucesso no controlador Ryu. Diversas ferramentas foram utilizadas para a execução do trabalho, incluindo *ONOS, Ryu7, OpenFlow, Open*

Virtual Switch, Mininet, VirtualBox, Ubuntu 17.04, CPqD, Lagopus e Intellij IDEIA 10. A pesquisa concluiu que é essencial ter um procedimento para criar e configurar *QoS* e filas a partir do próprio plano de controle. Assim como JHA, este estudo também utilizou o método para criação de *QoS* e filas no *OVS* por meio de comando no sistema operacional. A Tabela 3.1 mostra a comparação entre trabalhos que utilizaram técnicas e soluções semelhantes.

Tabela 3.1 – Comparativo entre os trabalhos

Referência	Tecnologias	Objetivo
C2-APP	*ONOS*, *MiniNDN* e *OpenFlow*	Priorização de nós no C2
(ABBOU; SONG, 2021)	*ONOS*, *Mininet* e *OpenFlow*	*QoS* em redes *SDN*
(JHA, 2017)	*ONOS*, *Mininet* e *OpenFlow*	Fornecer Garantias *QoS*

O trabalho de (STOCCHERO et al., 2023a) propôs uma combinação de Redes Definidas por Software (*SDN*) com Redes Centradas em Informação (*ICN*) para atender aos requisitos de alto nível dos sistemas de Comando e Controle. Ele citou que os sistemas de Comando e Controle (C2) centram suas operações em rede e por esse motivo a dinâmica era fundamental. O emprego de dispositivos heterogêneos comprovou a aplicação dos conceitos de *IoBT* no C2. Este trabalho abordou uma prova sobre o conceito da arquitetura comparando as redes somente *IP* e *SDN*. Os resultados da implementação do trabalho foram definidas como significativos, pois obtiveram uma melhoria do atraso na rede, na carga da rede e na perda de pacotes em comparação às abordagens convencionais. Este trabalho também utilizou essas abordagens, visto que a ideia era aproveitar as melhores características de cada um deles. A programabilidade da rede do *SDN* e o acesso do conteúdo pelo nome e a persistência e autenticidade dos dados do *ICN*.

(ZHANG et al., 2023), abordou o emprego de *ICN-IoT*, discutiu suas problemáticas, empregos e desafios. Sobre o *IoT*, o trabalho relatou a ineficiência do suporte a mobilidade e da ineficácia em trabalhar com cache na rede. Ele propôs o emprego de *IoT* em conjunto com *ICN*, como forma de redução de dados e economia de energia em dispositivos *IoT*. Apresentou uma definição de políticas para melhoria de cache, como a redução do tráfego de *IoT* baseado em *ICN*.

O artigo de (CHOUDHARI; NITURE, 2022) foi o resultado de uma pesquisa que começou com a história dos Protocolos de Comunicação Espacial apresentados pelo CCSDS (Comitê Consultivo para Sistema de Dados Espaciais). Ele tratou da necessidade e implantação do *DTN*, testes e análises em tempo real. Descreveu as vantagens do *DTN* sobre outros protocolos no que dizia respeito ao gerenciamento de rede, segurança

e qualidade dos serviços. Ele relatou a análise de desempenho do *DTN* em comparação ao *TCP*. Identificou a necessidade de melhorias no *DTN* para futuras missões espaciais.

O trabalho conduzido por (LEE; BAEK JEANY SON, 2023) buscou melhorar o desempenho de uma rede baseada em software definido por rádio (*SDR*). Ele propôs melhorias baseadas em dois esquemas: o controle de acesso do usuário com o auxílio de impressão digital e a otimização do desempenho da rede. A otimização de desempenho da rede previa a utilização de métodos de alocação de energia, de recursos sem fio e melhoria de *QoS* da rede com experimentos. Os resultados da simulação mostram que os algoritmos propostos podiam melhorar a utilidade da rede e vários *QoS* da rede em um ambiente *IoBT*.

A Tabela 3.2 mostra a comparação entre trabalhos que utilizaram técnicas e soluções semelhantes.

Tabela 3.2 – Comparação entre as tecnologias

Referência	Tecnologias	Objetivo
C2-APP	*ICN* e *IoBT*	Uso de tecnologias heterogêneas
(STOCCHERO et al., 2023a)	*ICN* e *SDN*	Uso combinado de *ICN/SDN*
(ZHANG et al., 2023)	*ICN e IoT*	Uso em conjunto de *ICN/IoT*
(CHOUDHARI; NITURE, 2022)	*DTN*	Pesquisa e análise do *DTN*
(LEE; BAEK JEANY SON, 2023)	*IoBT*	Uso de *QoS* com *SDR*

Os trabalhos de Abbou, JHA e o presente estudo apresentam semelhanças significativas. Todos eles empregaram tecnologias similares, incluindo *Mininet, ONOS, OpenFlow e OVS*. Uma característica comum entre os dois primeiros estudos é a implementação de aplicativos que operam diretamente no controlador *ONOS*.

O objetivo dessas implementações era aprimorar o desempenho do controlador por meio da criação de *QoS* e filas, uma funcionalidade que, ainda não está disponível na versão 3.0 do *ONOS*. Devido à ausência dessa função, a maioria do processo de criação de *QoS* e filas precisa ser realizada por meio de comandos do *OVS* no sistema operacional hospedeiro. Um exemplo desse processo é a troca de fila dos nós da rede, que pode ser realizada diretamente no *ONOS* por meio da criação de um *intent*, conforme ilustrado na Figura 4.8.

Assim como o trabalho de Abbou, este estudo também desenvolveu um aplicativo. No entanto, a principal distinção reside no propósito de cada um. O primeiro tinha como objetivo facilitar a criação de Qualidade de Serviço (*QoS*) no *Open Network Operating System (ONOS)*. Por outro lado, o C2-APP foi projetado para se integrar ao ONOS com o

intuito de gerenciar uma rede de Definição de Software por Rede (*SDN*).

O trabalho de Lee, assim como este estudo atual, buscou soluções para a implementação de *QoS* (Qualidade de Serviço) em redes *IoBT (Internet of Battlefield Things)*. No entanto, as tecnologias empregadas divergiram. Enquanto o primeiro utilizou *SDR* (Rádio Definido por Software), este trabalho optou por *SDN* (Rede Definida por Software) como o núcleo da rede.

Uma característica fundamental de um bom controlador é a capacidade de alterar as prioridades dos fluxos de rede segundo a intenção do comandante da missão. Esta é uma das exigências para o funcionamento adequado conforme os critérios de Agilidade do C2. Em diferentes momentos da missão, podem ser necessárias configurações distintas para distribuir as informações críticas aos nós que precisam recebê-las ou enviá-las o mais rápido possível (STOCCHERO et al., 2023a). A maneira mais simples de alcançar a priorização em uma rede é criando Qualidade de Serviço (*QoS*) e filas em todos os nós da rede.

4 PROJETO DA APLICAÇÃO CIENTE DO ESTADO DA REDE

Este trabalho propõe a utilização de *QoS* para obter priorizações de determinados fluxos no campo de batalha. Em certos momentos do combate, algum nó da rede poderá obter uma priorização em relação aos outros nós que estão na mesma rede.

4.1 Arquitetura de rede

Esta seção é dividida em duas partes principais: o Modelo Real e o Modelo Lógico. O Modelo Real apresenta um exemplo de como uma rede *SDN* seria configurada, utilizando dispositivos e cenários do mundo real. Em contraste, o Modelo Lógico utiliza programação em emuladores e sistemas operacionais para simular o funcionamento de uma rede de forma teórica e controlada. O desenvolvimento deste trabalho foi realizado exclusivamente com o Modelo Lógico, enquanto o Modelo Real foi utilizado apenas para fins de correlação e definição do cenário.

4.1.1 Modelo Real

A Figura 4.1, mostra uma situação de como seria uma rede *SDN* no campo de batalha no mundo real. Este modelo engloba os seguintes componentes, tanto materiais quanto de pessoal:

- Viatura blindada de transporte de pessoal (*VBTP*): é a viatura pertencente ao comandante do pelotão. Essa viatura carrega em seu interior o servidor do *ONOS*, do C2-App e um *Access Point*.
- soldados (soldado1, soldado2, soldado3 e soldado4): militares pertencentes ao pelotão que prestam segurança a viatura. São dotados de dispositivos tipos *tablets* que são os nós da rede que possuem a função de *Access Point* e estação de trabalho.
- Drones (drone1 e drone2): é um nó da rede que possui a função de *Access Point* e estação de trabalho.
- Sensores: dispositivos *weireless* inseridos em posições estratégicas no campo de batalha. Eles comunicam-se entre si por meio de uma rede *ad-hoc*. As informações adquiridas pelos sensores são coletadas pelos drones do pelotão por meio de *Delay-tolerant networking (DTN)*.

- *Access Point* (AP1): é o dispositivo de entrada na rede *SDN*. Os nós que estão dentro da zona *Wi-Fi* estão conectados a rede por meio de dispositivos (soldado1, soldado2, soldado3, soldado4 e VBTP) e os drone1 e drone2).
- *NDN*: é utilizada para conectar e receber informações de interesse entre os nós da rede.
- *DTN*: é utilizada para realizar a comunicação entre os sensores e drones que entrarem ao alcance para comunicação.

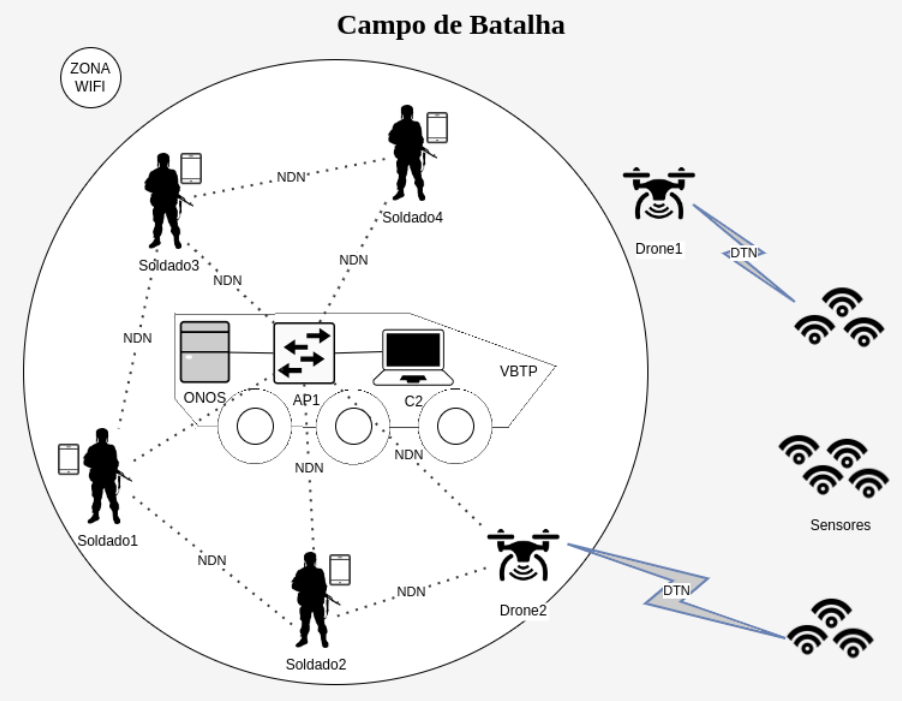

Figura 4.1 – Exemplo de Rede SDN no mundo real.

4.1.2 Modelo Lógico

Neste estudo, a configuração da rede é composta por um total de 36 nós. A escolha de 36 nós foi feita para se aproximar do número real de um pelotão, o que pode ser valioso se o modelo for aplicado em um cenário real. Os estudos mencionados no Capítulo 3, implementam topologias que variam de 3 a 10 nós, uma quantidade significativamente menor do que a utilizada neste trabalho.

A Figura 4.2 permite a identificação de 2 nós de rede. Cada par de dispositivos, que consiste em 1 *switch* e 1 *host*, é considerado como um único nó dentro da estrutura da rede. A configuração completa da rede foi estabelecida mediante um *script* em linguagem de programação *Python*, utilizando o emulador *MiniNDN*. Além disso, a rede emprega o

OVS 3.1, o *OpenFlow* 1.4 e o *ONOS* 3.0 como controlador *SDN*.

Embora a comunicação sem fio seja essencial em uma implementação real, este estudo utilizou uma rede com fio configurada para simular as características de uma rede *Wi-Fi*. Isso foi necessário devido às dificuldades encontradas ao tentar implementar o modelo de rede *Wi-Fi* em *Mesh*, disponível no emulador *Mininet-WiFi*. Este modelo permitiu a operação sem erros somente até 10 nós simultaneamente. No entanto, o escopo deste projeto exigia a utilização de um número maior de nós para representar um cenário realístico (no caso 36 nós), o que tornou inviável a utilização da rede *Wi-Fi*. Portanto, optou-se pela implementação de uma rede cabeada com parâmetros alterados para se aproximar de uma rede *Wi-Fi*.

Este aspecto não comprometeu a representação da rede como uma rede sem fio, uma vez que todos os parâmetros foram ajustados para simular o comportamento de uma rede *Wi-Fi*. Os parâmetros primordiais que foram configurados incluíram o atraso (*delay*) e a capacidade da banda de rede entre os nós.

A configuração de um atraso (*delay*) padrão de 10ms para cada salto de rede foi estabelecida para aproximar as características da rede simulada às das redes de *Wi-Fi*. No entanto, isso também pode ser visto como uma limitação, pois poderia influenciar os resultados dos testes.

Além disso, a largura de banda da rede foi estabelecida em 500 kbps. Esta configuração foi definida após uma extensa série de experimentos com larguras de banda variando de 1 a 10 Mbps. Durante estes experimentos, constatou-se que a rede não exibia níveis de congestionamento adequados para os testes propostos. Assim, determinou-se que uma largura de banda de 500 kbps seria a mais apropriada para a execução dos experimentos, uma vez que esta configuração resultou em mais congestionamentos nas rotas de rede.

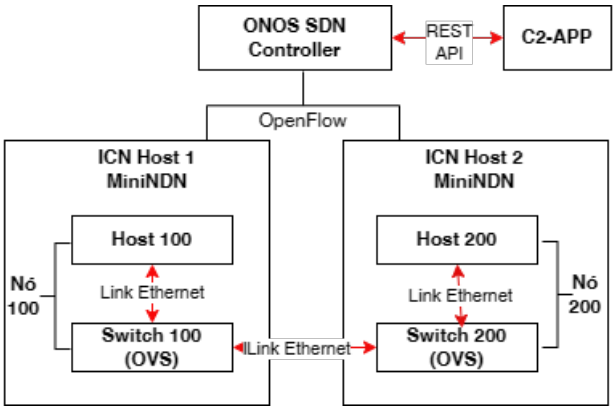

Figura 4.2 – Visão da Arquitetura de dois nós

4.2 Dimensão da Aplicação C2-App

C2-App visa gerenciar os dispositivos conectados no campo de batalha e desta forma auxiliar o comandante da tropa na tomada de decisões. As informações adquiridas pelos dispositivos espalhados pela área de combate podem ser importantes para o cumprimento da missão. Informações obtidas de sensores, dispositivos dos soldados e drones, se transmitidas de forma ágil, poderão ser determinantes para obter uma vantagem sobre o inimigo. O sistema C2-App possui 2 módulos principais: $Area\ de\ Interresse$ e $Monitoramento$. Ambos os módulos usam um Sistema de Informações Geográficas (*GIS*) e fazem parte da proposta da arquitetura de rede.

4.2.1 Desenvolvimento

Para o desenvolvimento de C2-App foi utilizado o *Django* na versão 3.2. *Django* é um *framework web full stack open source* (código aberto) baseado em *Python*. Ele foi criado para resolver os problemas mais comuns do processo de desenvolvimento de aplicações web, como, por exemplo, autenticação, rotas, *object relational mapping (ORM)* e *migrations*. A Figura 4.3 mostra o diagrama de componentes de C2-App utilizado para desenvolvê-lo e os demais dispositivos que fazem parte deste trabalho, como o *ONOS* e o *MiniNDN*.

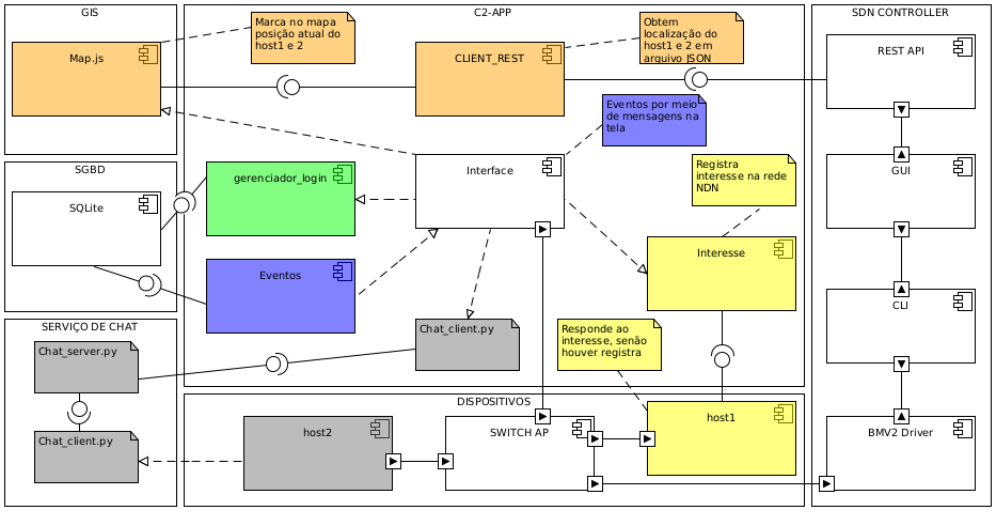

Figura 4.3 – Diagrama de Componentes.

4.2.2 Módulo de Monitoramento

Blue Force Tracking (SECURITY, 2014) é um termo utilizado pelas forças armadas americanas para fazer referência ao monitoramento e a localização das forças amigas, do pessoal e dos recursos em uma área de operação. Esse sistema combina o *GPS* com outras tecnologias e dispositivos de comunicação sem fio, visando melhorar a consciência situacional. O conhecimento da localização em tempo real do pessoal e ativos são essenciais para ajudar nas emergências e na tomada de decisões durante um incidente ou evento de maior amplitude. O rastreamento das forças amigas utiliza um receptor *GPS* para os cálculos de localização e um módulo de comunicação para a transmissão das informações. O software coleta e exibe informações atuais sobre pessoal e localização de ativos em uma interface gráfica acessível aos comandantes. Alguns exemplos de dispositivos de rastreamento que podem ser usados para fornecer localização:

- Pulseiras e relógios de pulso;
- Dispositivos portáteis e de bolso;
- Dispositivos montados em veículos;
- Computadores portáteis;
- Terminais de satélite portáteis;
- *Smartphones* e *tablets*.

Os receptores *GPS* determinam a localização, sinais de rádio de baixa potência para obter informações orbitais e de horas de pelo menos quatro satélites. O receptor utiliza esta informação para determinar sua posição na superfície da Terra com uma margem de até 20 metros de erro.

O software de rastreamento, que possui informações de localização como longitude, latitude e elevação, emprega um aplicativo de mapeamento do Sistema de Informações Geográficas (*GIS*) para monitorar o movimento de pessoal e ativos. Isso pode ser feito em tempo real ou em intervalos de tempo especificados.

Em operações militares reais são utilizadas tecnologias correspondentes ao *Blue Force Tracking*, por se tratar de uma simulação, C2-App parte da premissa que todos os nós da rede possuem suas informações de localização atualizadas e que estão disponíveis para consulta por meio da R*EST API* do controlador *ONOS*. C2-App utiliza um sistema de informação geográfica (*GIS*) a partir de *tiles* do *Google Maps e do OpenStreetMap*. *Tiles* são imagens dispostas que formam a base de um mapa. Os *tiles* utilizados neste trabalho

são de mapas rodoviários, de fotos de satélites e da topografia. A Figura 4.4, mostra a utilização do *tile* de mapa rodoviário do *Google Maps*. Além dos *tiles*, o módulo utiliza a biblioteca *Leaflet* para apresentação dos mapas. A biblioteca *Leaflet* tem a finalidade de criação de mapas interativos na linguagem *Python*. O módulo de monitoramento possui grande importância ao sistema porque nele é possível acompanhar o posicionamento da tropa amiga (cor azul) e a tropa inimiga (cor vermelha), conforme a Figura 4.4. O módulo de monitoramento solicita as informações dos nós da rede para atualizar o posicionamento da tropa no mapa. As informações de posicionamento foram configuradas mediante mudanças situacionais no campo de batalha. Como, por exemplo, ela pode estar definida inicialmente em 60 segundos e devido à mudança situacional, ela pode ser alterada para 30 segundos. C2-App possui um *Client Rest*, que solicita a *REST API* informações que podem ser de dispositivos, *hosts* e *intents*, por exemplo. O capítulo 5 traz informações detalhadas sobre a interação com o *ONOS* via *REST API*.

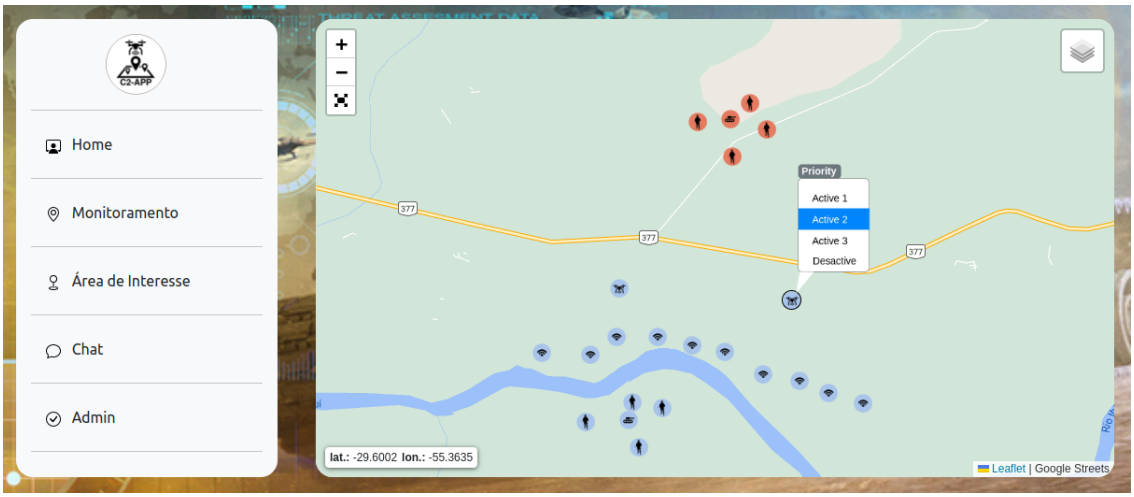

Figura 4.4 – Interface C2-APP.

4.2.3 Área de Interesse

É uma região do campo de batalha que possui interesse do comando da operação, conforme a tela do sistema C2-APP mostrado na Figura 4.5. Ao acessar o item de menu Área de Interesse é aberta uma nova janela. No lado esquerdo do mapa é habilitada uma barra de ferramentas de desenho. Por meio dela é possível fazer o mapeamento dos elementos de requisição, como, por exemplo:

- Hora da rede: 10:00 horas;
- Área 1: área de interesse número 1;

- Dispositivo responsável: drone 1 ;
- Validade da requisição: 10:30 horas;
- Área 2: área de interesse número 2;
- Dispositivos responsáveis: drone 1 e drone 2;
- Validade da requisição: 10:45 horas;
- Área 3: área de interesse número 3;
- Dispositivo responsável: drone 2;
- Validade da requisição: 11:00 horas.

A Figura 4.5 mostra a validade da requisição 1 em 30 minutos e a requisição 2 em 45 minutos. Essa configuração é apenas um modelo de exemplo e as requisições de interesse dependerão sempre da consciência situacional dos comandantes.

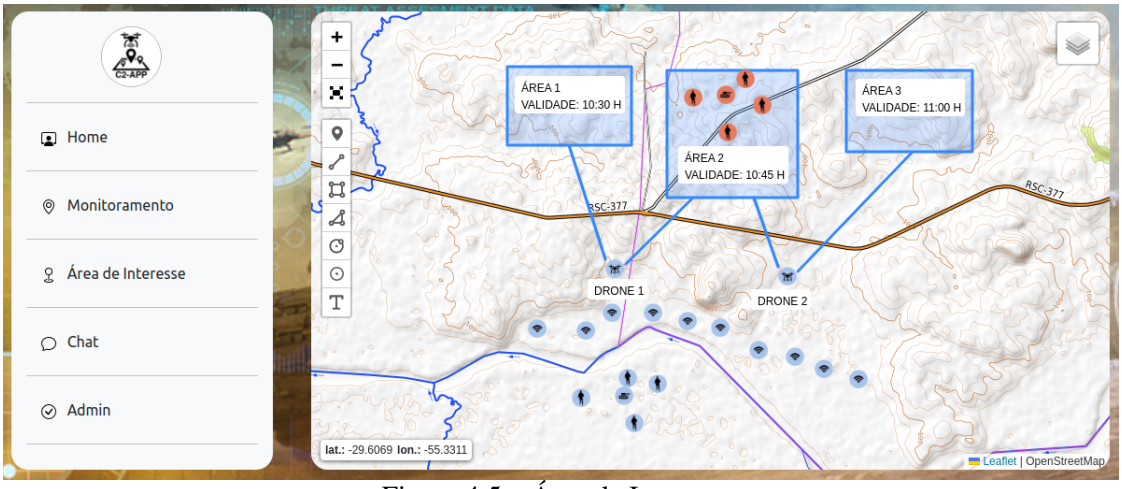

Figura 4.5 – Área de Interesse.

4.2.4 Serviço de *Chat*

O serviço de *chat* é baseado em modelo cliente-servidor. Para funcionamento do mesmo são utilizados dois *scripts* em *Python*, um para o servidor (servidor.py) e outro para o cliente (client.py). Juntamente com C2-App o servidor do *chat* é inicializado executando-se o arquivo servidor.py. O serviço de chat no cliente é inicializado após selecionar no menu o item *Chat*. Este serviço tem a finalidade de manter uma comunicação entre os soldados desembarcados e a viatura do comandante, a ser usado sempre que houver relevância de informações. A comunicação entre os elementos é feita utilizando um código de mensagens pré-definidas conforme o exemplo na Tabela 4.1. A finalidade da

utilização de códigos e mensagens pré-definidos é facilitar e melhorar a agilidade no envio de informações no campo de batalha. Na Tabela 4.1 foi definido que o termo vermelho trata de assuntos do inimigo e o termo azul trata de assuntos da tropa amiga. Durante uma troca de mensagens a única coisa que deve ser enviada é um código pré-definido. A letra X presente no código deve ser substituída por uma localização geográfica. Usando o primeiro código, por exemplo, a mensagem ficaria assim: VAL-30.06L-51.12, que significa: o Inimigo foi avistado LAT -30.06 LONG -51.12. Outra mensagem importante que também deve ser enviada é o valor (tamanho do efetivo) do inimigo. O código da mensagem ficaria assim: VIVPEL, que significa: informo que o inimigo possui valor Pelotão.

Tabela 4.1 – Exemplos de códigos e mensagens pré-definidas

Código	Assunto	Mensagem pré-definida
VALXXXXXLXXXX	vermelho	Inimigo foi avistado LAT XXXXX LONG XXXXX
VIVXXXXXXXXXX	vermelho	Info inimigo possui VALOR BTL/CIA/PEL/GC
AFLXXXXXLXXXX	azul	soldado ferido LAT XXXXX LONG XXXXX
VBLXXXXXLXXXX	vermelho	Blindado destruido LAT XXXXX LONG XXXXX
ABLXXXXXLXXXX	azul	Blindado destruido LAT XXXXX LONG XXXXX

4.2.5 Eventos e Alertas

A Figura 4.6, mostra os processos de eventos e alertas para algumas situações do campo de batalha monitoradas por C2-App.

1. cadastramento inicial: representado pelo número 1 na Figura 4.6, utiliza as funções cadAzul e atualizaPosição. O primeiro cadastra as informações da tropa amiga no banco de dados. O segundo insere as informações salvas no banco de dados no mapa. A tropa amiga e dos dispositivos usados em combate são cadastrados de forma manual no sistema.
2. cadastramento automático: representado pelo número 2 na Figura 4.6, utiliza as funções obtemPosição e atualizaAuto. A primeira consiste na atualização do posicionamento dos nós da tropa amiga. O componente *client Rest*, na Figura 4.3 se conecta a *REST API* do *ONOS* e obtém a configuração dos nós da rede no formato *JSON*. Este arquivo possui as informações atualizadas de latitude e longitude que serão utilizadas para atualização do posicionamento. A função atualizaAuto atualiza o posicionamento no mapa do sistema.

3. cadastramento do inimigo: a função cadVermelho insere no sistema a localização da tropa inimiga, pode atualizar o seu posicionamento no terreno e pode removê-las se houver baixas. A atualização é feita o mais rapidamente possível pelo primeiro nó que obter a informação. A atualização das informações são feitas de forma manual.
4. eventos do sistema: a função geraEvento inicia-se logo após a inserção do inimigo no mapa. Após isso, então surgirá um pop-up de Alerta na tela do sistema. Outros eventos de alerta que poderão surgir são os seguintes: soldado ferido, dispositivo fora da área, inimigo avistado e inimigo se deslocando, por exemplo.

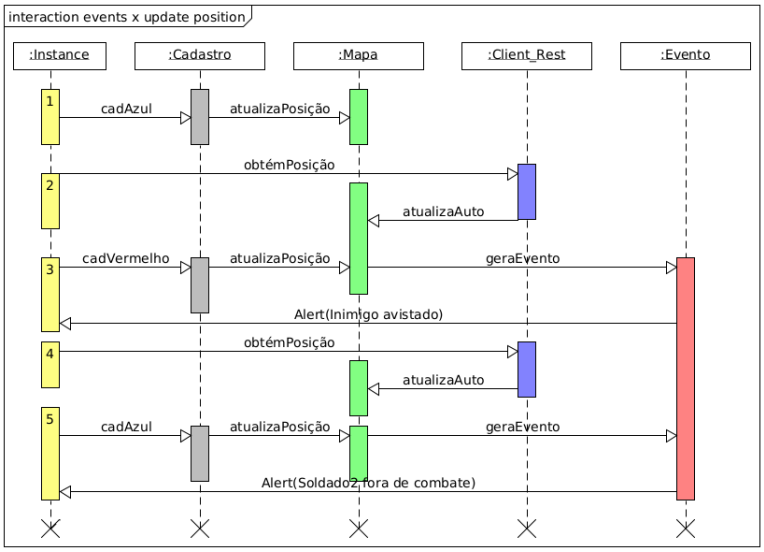

Figura 4.6 – Diagrama de sequência de eventos.

4.2.6 Reorganização dos nós da rede via *REST API*

O aplicativo C2-APP emprega os serviços do *ONOS* por meio da *REST API (Representational State Transfer)* para acessar informações de rede e definir prioridades de fluxo. Este sistema é capaz de reorganizar os nós de rede conforme necessário.

Quando há uma alteração situacional no campo de batalha, o C2-APP é notificado com alerta exibidos na tela. O tipo de alerta determina a ação sequente: o comandante da tropa pode ordenar a mudança de prioridade dos nós de rede.

No entanto, para que a reorganização dos nós de rede seja efetuada, certas condições devem ser cumpridas. Essas condições garantem que a reorganização seja realizada de maneira eficiente e eficaz, mantendo a integridade e a funcionalidade da rede. Para que a reorganização dos nós da rede seja realizada, algumas condições devem ser atendidas:

- O *QoS* e suas respectivas filas devem ser planejadas e criadas antecipadamente para todos os nós desejados;
- A criação de *QoS* e filas deve ser feita diretamente no *OVS*, uma vez que o controlador *ONOS* não possui essa função;
- Cada *QoS* criada pode ter mais de duas filas, sendo que a fila "0"será a padrão. O valor configurado na fila padrão será automaticamente atribuído a porta do nó definido. Se forem criadas mais de uma fila, devem possuir configurações diferentes da fila "0".

A Figura 4.7 mostra o comando usado para criar *QoS* e filas no nó 200 (s200-eth1). Este é um comando do *OVS* que é executado diretamente no sistema operacional onde está instalado. O comando mostra a criação de uma nova *QoS* para a porta eth1 do *switch* 200 (nó 200), considerado o "produtor" neste trabalho. O comando configura uma banda de 500 kbps para a *QoS* e também cria as filas 0, 1, 2 e 3. A fila "0"foi criada com uma banda mínima de 100 kbps e máxima de 200 kbps, a fila "1"com uma banda mínima de 200 kbps e máxima de 300 kbps, a fila "2"com uma banda mínima de 300 kbps e máxima de 400 kbps, e a fila "3"foi criada com uma banda mínima de 400 kbps. Após a execução deste comando, a porta s200-eth1 será configurada para a fila "0"e só mudará de fila quando o comando mostrado na Figura 4.8 for executado. Todos os nós que não possuem configurações de *QoS* estarão sob o serviço de melhor esforço (*Best Effort*). No serviço de melhor esforço, a rede tenta encaminhar os pacotes o mais rápido possível e não oferece garantias de *QoS*.

```
ovs-vsctl --\
 -- set port s200-eth1 qos=@newqos --\
 --id=@newqos create qos type=linux-htb other-config:max-rate=500000
  queues=0=@0,1=@1,2=@2,3=@3 --\
 --id=@0 create queue other-config:min-rate=100000 other-config:max-rate=200000 --\
 --id=@1 create queue other-config:min-rate=200000 other-config:max-rate=300000 --\
 --id=@2 create queue other-config:min-rate=300000 other-config:max-rate=400000 --\
 --id=@3 create queue other-config:min-rate=400000
```

Figura 4.7 – Comando para criar *QoS* e filas

Neste estudo, o C2-APP, através da *REST API* do *ONOS*, enviou um comando para que a porta s200-eth1, que atualmente opera na fila "0" padrão da *QoS*, passe a operar na fila "3". Este comando é executado no controlador *ONOS* por meio da criação de *intent*.

A Figura 4.8 mostra o comando para a criação de uma *Intent*, com os endereços *MAC* de origem (s200-eth1) e destino (s100-eth1). Com isso, foi configurado que o fluxo de dados da rota entre o *MAC* de origem e o *MAC* de destino obedecerá à banda configurada na fila "3" (mínimo de 400 kbps).

```
{ "type": "HostToHostIntent",
 "appId": "org.onosproject.fwd",
 "priority": 100,
 "treatment": {
       "instructions": [
         {
           "type": "QUEUE",
           "queueId": 3
         }
       ],
       "deferred": []
       },
 "one": "CA:93:C4:1E:F6:B3/None",
 "two": "AA:A3:41:2E:94:AB/None"
}
```

Figura 4.8 – Comando para criar *Intent* no *ONOS*

4.3 Dimensão da Orquestração

O controlador *ONOS* é o centro da rede baseada em software presente neste trabalho. Ele possui de forma integrada o *Intent Framework*, o qual é o serviço que permite priorizar fluxos de dados para rede *SDN*. É responsabilidade do controlador *ONOS* traduzir essas políticas em uma configuração de rede. Uma das vantagens do *ONOS* é possuir como característica a instalação de *intents*. Uma rede baseada em *Intent-based networking (IBN)* pode abstrair os mecanismos de como as redes são configuradas para focar na especificação de metas que a rede deve alcançar. Como o *ONOS* possui o seu próprio *Intent Framework*, ele ganha em funcionalidades em relação a outros controladores (UJCICH; SANDERS, 2019). As instalações de *intents* (IRFAN et al., 2019) no *ONOS* podem ser feitas por meio da *REST API* e do *CLI*. No *ONOS* é possível a instalação dos seguintes tipos de *intents*:

- *host-to-host*: instalação de um intent para comunicação direta entre duas máquinas. Exemplo usado no *ONOS CLI*: *add-host-intent <h1-ID> <h4-ID>*.
- *add-point-intent*: instalação de um intent para fazer comunicação ponto a ponto entre duas interfaces de um dispositivo. Exemplo usado no *ONOS CLI*: *add-point-intent <device/port>(1) <device/port>(1)*.
- *add-single-to-multi-intent*: instalação de um intent entre um dispositivo de entrada e múltiplos dispositivos de saída. Exemplo usado no *ONOS CLI*: *add-single-to-multi-intent <device/port>(1) <device/port>(2) <device/port>(...)*.
- *add-multi-to-single-intent*: instalação de um *intent* entre múltiplos dispositivos de

entrada e um dispositivo de saída. Exemplo usado no *ONOS CLI*: *add-multi-to-single-intent <device/port>(2) <device/port>(...) <device/port>(1).*

4.3.1 *REST API*

Nessa seção são apresentadas as principais opções disponibilizadas para acesso via *REST API* no controlador *ONOS*. A *REST API* utiliza o formato *JavaScript Object Notation (JSON)* e as chamadas que a mesma recebe são feitas via *URL*, conforme é visto na Figura 4.9.

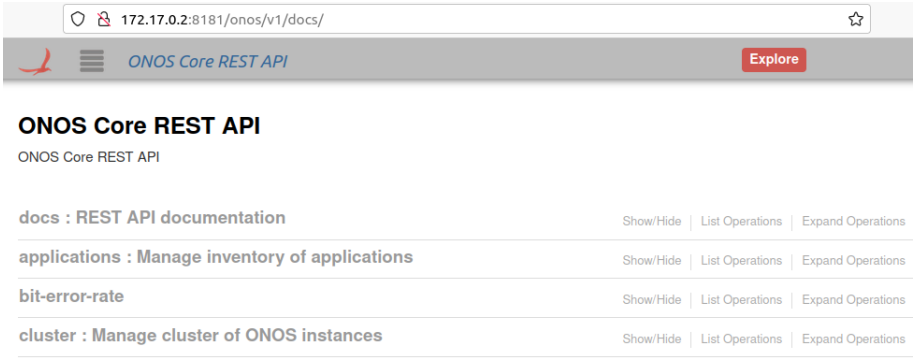

Figura 4.9 – Acesso a *REST API* via *URL*

4.3.2 Gerenciamento de *intents*

O *ONOS* reconhece as intenções como *intents* da rede. A Figura 4.10 mostra o endereço da *URL* para o acesso às opções de *intents* possíveis de serem consultadas pelo sistema C2-APP para obter (*GET*), adicionar (*POST*) e remover (*DELETE*).

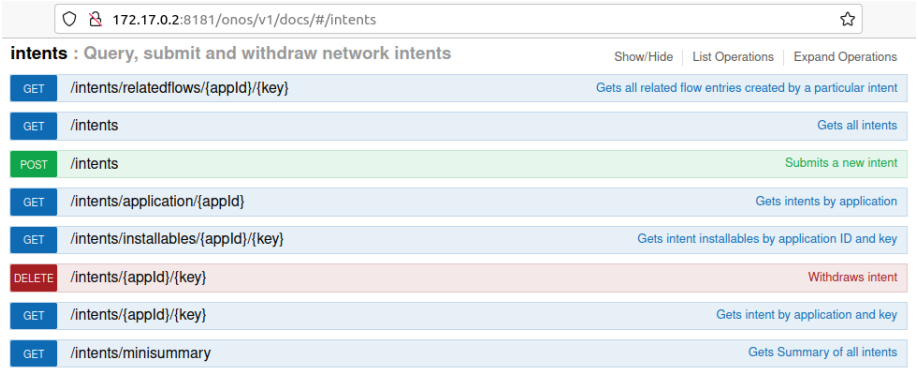

Figura 4.10 – Acesso as configurações de *intents* via *REST API*

4.3.3 Gerenciamento de *meters*

Os *meters* são usados para monitorar a taxa de entrada de fluxos. Também podem ser combinados com as filas para implementar *QoS*. O *ONOS* tem a possibilidade de criar *meters* via *REST API*, conforme a Figura 4.11, estão disponíveis pelo sistema C2-APP as opções de consultar (*GET*), adicionar (*POST*) e remover (*DELETE*).

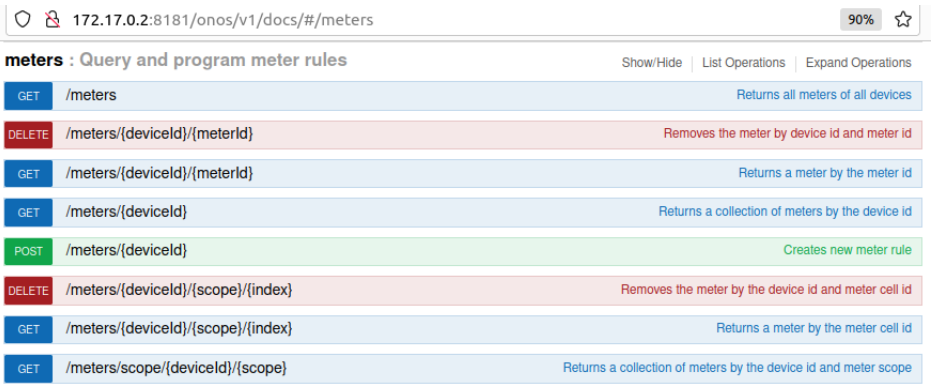

Figura 4.11 – Acesso as configurações de meters via *REST API*

4.3.4 Gerenciamento de dispositivos

O *ONOS* reconhece *switch/Access Point* como dispositivos da rede. A Figura 4.12 mostra o endereço da *URL* para o acesso às opções de dispositivos possíveis de serem consultadas pelo sistema C2-APP para obter (*GET*), adicionar (*POST*) e remover (*DELETE*).

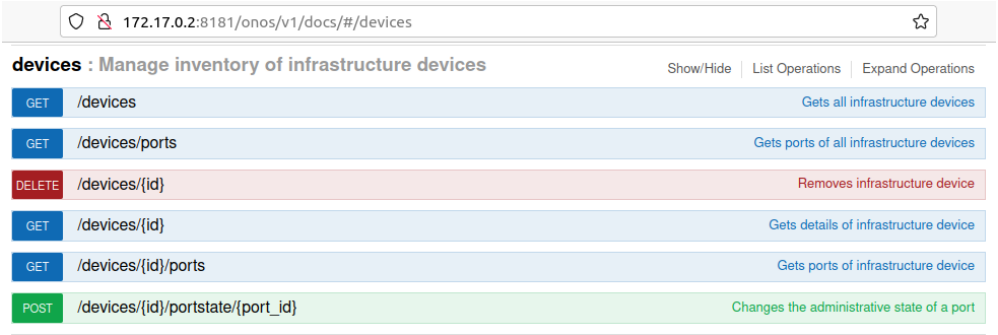

Figura 4.12 – Acesso as configurações de dispositivos via *REST API*

4.3.5 Gerenciamento de *hosts*

O *ONOS* reconhece *hosts/stations* como *hosts* da rede. A Figura 4.13 mostra o endereço da *URL* para o acesso às opções de *hosts* possíveis de serem consultadas pelo sistema C2-APP para obter (*GET*), adicionar (*POST*) e remover (*DELETE*).

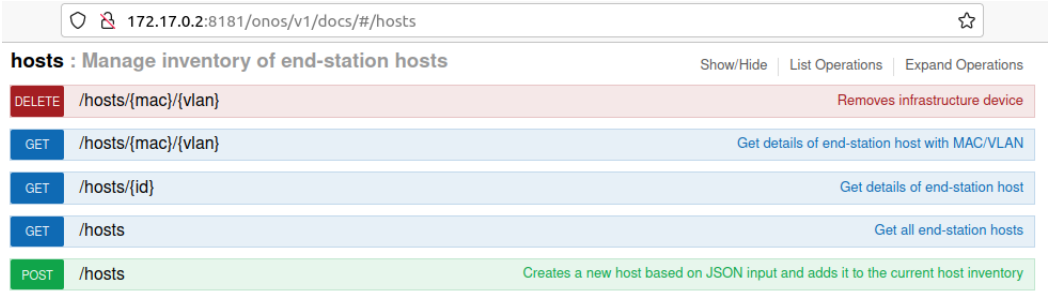

Figura 4.13 – Acesso as configurações de *hosts* via *REST API*

4.3.6 *Command-line Interface (CLI)*

Outra forma de interagir com o *ONOS* é utilizando a *CLI*. Para conseguir acesso deve ser utilizado usuário e senha pelo *SSH* para o endereço *IP* do servidor do *ONOS*. Todos os comandos aplicados na parte gráfica também podem ser utilizados na *CLI*. Exemplo de acesso a *CLI*, Figura 4.14.

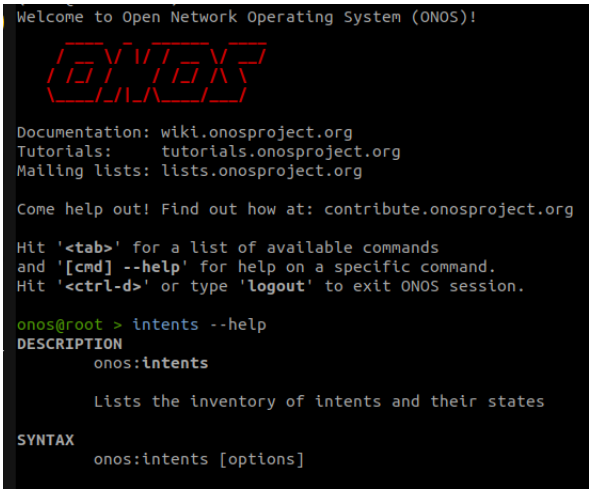

Figura 4.14 – Acesso as configurações do *CLI*

4.3.7 *Graphical user interface (GUI)*

Outra forma de interagir com o *ONOS* é utilizando a parte gráfica (*GUI*). Para conseguir o acesso deve ser utilizado usuário e senha pela *URL* para o endereço *IP* do servidor do *ONOS*. Como exemplo de acesso a *GUI*, a Figura 4.15, mostra uma rede em *Mesh* com 36 *switches* e 36 *hosts*.

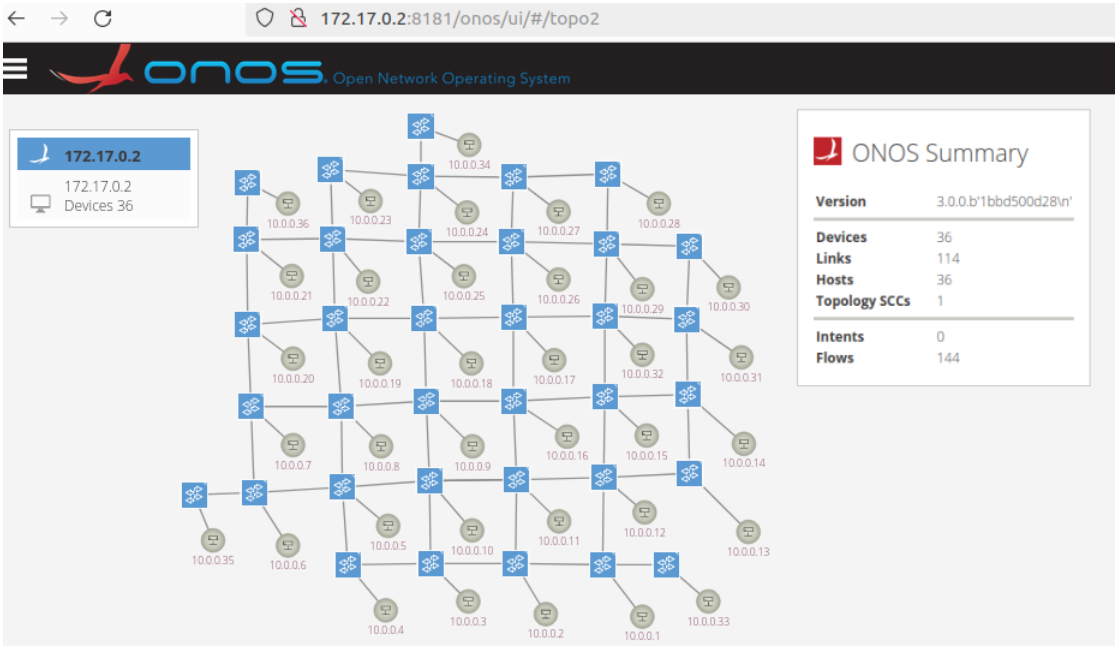

Figura 4.15 – Interface Gráfica do *ONOS (GUI)*

5 METODOLOGIA

5.1 Cenário de estudo

Embora uma rede no campo de batalha possa estar em constante movimento, este trabalho não considerou a mobilidade para realização de experimentos, mas sim instantes em que a rede não está em movimento.

A Figura 5.1 mostra dois casos de estudo para aplicação em campo de batalha, representadas pelos números 1 e 2. Em ambas as situações, os elementos representam nós de rede (comandante, drone 1, drone 2 e soldado), setas (troca de informações entre nós) e *IoBT* (representa todos os outros nós de rede não mencionados anteriormente). Os nós comandante e soldado são os nomes dos dispositivos de rede que acompanham as pessoas. Vale ressaltar que, em uma situação real, os indivíduos utilizariam diversos tipos de dispositivos eletrônicos. Isso inclui smartphones, dispositivos de bolso, dispositivos montados em veículos e outros dispositivos mencionados na seção 4.2.2.

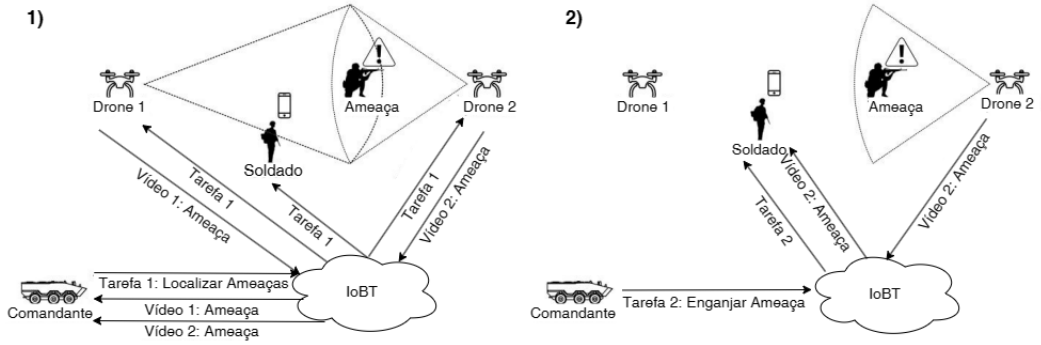

Figura 5.1 – Cenário de aplicação de priorização de nós. (STOCCHERO et al., 2023a)

Na situação 1, o nó comandante solicita informações dos drones sobre possíveis ameaças. Ele tem prioridade para receber essas informações, enquanto os outros nós recebem as informações como um serviço de melhor esforço. Esta situação é caracterizada pelo reconhecimento do campo de batalha pelos drone 1 e drone 2, que coletam informações por meio de vídeos e fotos e as enviam para o nó comandante para a tomada de decisão.

A situação 2 envolve três nós de rede: comandante, drone 2 e soldado. Neste cenário, o drone 2 identifica uma potencial ameaça e informa por meio de alerta no sistema C2-APP. Após a tomada de decisão, o nó soldado recebe a tarefa de se engajar com o inimigo, e todas as informações adquiridas pelo drone 2 são priorizadas para o soldado. Isso ocorre porque o nó soldado está em uma posição mais próxima à ameaça identificada,

em comparação aos outros nós da rede. Esta decisão destaca que, naquele momento específico, o objetivo de maior importância estava dentro do âmbito de atuação do nó soldado.

5.2 Configurações

A Tabela 5.1traz um resumo das principais configurações, protocolos e softwares utilizados neste trabalho. Já a Figura 5.2 representa a topologia de rede que mapeia estes cenários de aplicação representado na Figura 5.1. Esta topologia foi utilizada como a base para desenvolvimento deste trabalho. A implementação foi realizada utilizando a linguagem de programação *Python*. O desenvolvimento se baseou em modelos disponíveis nas pastas de exemplos dos repositórios *Mininet, Mininet-Wi-Fi e MiniNDN*, todos acessíveis no site *GitHub*.

Tabela 5.1 – Detalhes da Emulação

Sistema Operacional	Ubuntu 22.04
Softwares e protocolos	*ONOS* 3.0, *OpenVSwitch* 3.1, *OpenFlow* 1.4, *MiniNDN*
Topologia	*Mesh* com 36 nós
Banda por *link*	500 kbps
Delay por *link*	10ms
Prioridades das filas (kbps)	0 (100 - 200), 1 (200 – 300), 2 (300 – 400), 3 (≥ 400)
Melhor Esforço	nós 100, 300, 400
IoBT	nós 1 - 32
Prioridade 3	nó 200
Mecanismo das filas	*Hierarchical Token Bucket (HTB)*

Para utilizar tráfego de dados *NDN* foi necessário a criação de outra camada de rede no *MiniNDN*. Os nós em amarelo na Figura 5.2 representam esta camada de rede *NDN*, configurada para os experimentos deste trabalho. Conforme ilustrado na Figura 5.2, as rotas A e B possuem alguns nós compartilhados. Especificamente, estes nós são os de números 200, 24, 25, 18, 9 e 10. O modelo adotado neste estudo emprega o *MiniNDN* para a transmissão de dados, explorando assim as vantagens inerentes às redes *NDN*. Sendo uma das principais vantagens das redes *NDN*, conforme descrito na seção 2.5 (*ICN*) deste trabalho, a capacidade de entregar e compartilhar uma quantidade maior de dados. Esta vantagem é evidenciada quando um arquivo é solicitado por múltiplos consumidores. Os dados que passam pelos nós da rede ficam disponíveis no cache destes nós. Assim, quando um segundo consumidor solicita o mesmo arquivo, os dados são disponibilizados pelo nó mais próximo que já possua o dado em cache. Isso otimiza a entrega de dados,

aproveitando eficientemente o cache da rede.

A rota A compreendendo os nós 200, 24, 25, 18, 9, 10, 3, 2, 1 e 100 e representa a sequência dos nós que encaminham os dados para o nó 100 (comandante). A rota B compreendendo os nós 200, 24, 25, 18, 9, 10, 5, 6, 7 e 300 e representa a sequência que encaminha dados para o nó 300 (soldado).

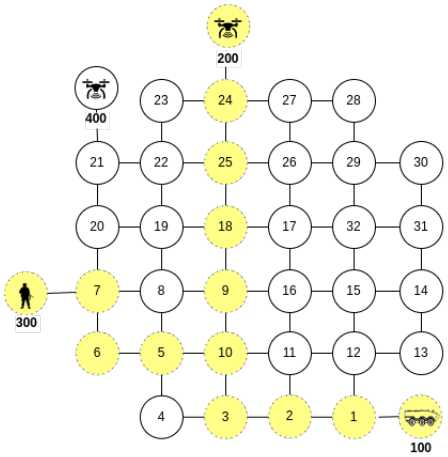

Figura 5.2 – Topologia de rede.

Para simular um congestionamento de rede e estabelecer condições realistas para o cenário em questão, foi levado em consideração que as condições de operação em um campo de batalha são frequentemente incertas. Tendo isso em consideração, foram estabelecidas três categorias distintas de tráfego de Ping (*ICMP*). As quantidades de pings e de nós alvos foram definidos de forma empírica. O comando utilizado para gerar o congestionamento na rede foi “ping -f -w 100 hx”, onde “x” representa o nó alvo. As categorias foram organizadas em ordem decrescente de fluxo em:

- Tráfego I: formado pelo envio de 47 pings foram enviados dos nós 4, 11, 13, 21, 23, 27, 29 e 31, tendo como alvo os outros 27 nós: 1, 2, 3, 5, 6, 9, 10, 18, 24, 25, 100, 200, 300, 400, 8, 19, 20, 22, 12, 14, 15, 16, 17, 26, 28, 30 e 32;
- Tráfego II: formado pelo envio de 34 pings foram enviados dos nós 4, 11, 13, 21, 23, 27, 29 e 31, tendo como alvo os outros 26 nós: 1, 2, 3, 5, 6, 8, 9, 10, 14, 15, 16, 17, 18, 19, 20, 22, 24, 25, 26, 28, 30, 32, 100, 200, 300 e 400;
- Tráfego III: formado pelo envio de 22 pings foram enviados dos nós 4, 11, 13, 21, 23, 27, 29 e 31, tendo como alvo os outros 14 nós: 8, 12, 14, 15, 16, 17, 19, 20, 22, 26, 28, 30, 32 e 400;

No contexto da avaliação do *ICN*, foi enviado um arquivo de vídeo do tipo .mp4, de tamanho 2.402.234 *bytes* através do *MiniNDN*, nas rotas A e B. Durante a execução

do experimento, o tráfego *ICMP* e o tráfego *ICN* foram executados simultaneamente, permitindo-nos testar a eficácia da abordagem proposta em cenários de sobrecarga de rede. Esta metodologia proporcionou uma avaliação efetiva do desempenho da rede sob as condições estipuladas.

5.3 Coleta de dados

O software Wireshark foi empregado para a análise do tráfego de rede, executado simultaneamente ao experimento. Durante a inspeção do log de tráfego de rede coletado pelo *Wireshark*, foram identificados os registros de cinco protocolos distintos: *LLDP, ARP, UDP, ICMP e IPv4*. Os protocolos *LLDP* e *ARP* foram gerados automaticamente pela rede. O tráfego *ICMP*, por outro lado, foi produzido como resultado da execução de comandos de ping, utilizados para gerar o tráfego na rede. O tráfego de rede originado via MiniNDN, especificamente pelos protocolos *IPv4* e *UDP*, foi examinado por meio da dissecação do *log* do tráfego de rede fornecido pelo *Wireshark*. Foram realizados uma média de 20 experimentos para cada situação proposta, totalizando mais de 320 testes. No valor total também foram considerados os testes que não foram aproveitados. Cada resultado a ser discutido no Capítulo 6 foi derivado da média de 20 experimentos conduzidos. Quaisquer valores anômalos que se desviaram significativamente da média, seja para mais ou para menos, foram desconsiderados.

5.4 Métricas

Para determinar o *Throughput*, foi levado em consideração o tamanho do arquivo enviado, que é de 2.402.234 *bytes*. Este valor foi então multiplicado pelo número de saltos em cada rota. Tanto a rota A quanto a rota B possuem 9 saltos. Portanto, para que 100% dos dados sejam recebidos, cada rota deve receber no mínimo 21.638.106 bytes. Este valor corresponde a nove vezes o tamanho do arquivo originalmente enviado. Esta métrica nos permite avaliar a eficiência da transmissão de dados em cada rota.

A taxa de pacotes perdidos foi calculada utilizando a proporção entre o número de pacotes perdidos e o número total de pacotes recebidos. Esta métrica fornece uma indicação precisa da qualidade e confiabilidade da transmissão de dados na rede.

6 RESULTADOS E DISCUSSÕES

Foi conduzida uma série de experimentos para avaliar o desempenho da transmissão de dados em redes *ICN*, utilizando a plataforma *MiniNDN* para a transmissão. As métricas utilizadas para a comparação incluíram: *Throughput, Jitter*, Latência e a taxa de perda de pacotes.

Após a transmissão de vários arquivos de diferentes tamanhos através do MiniNDN, determinou-se que o tempo ideal para a execução do experimento era de 100 segundos. Este período de 100 segundos foi dividido em duas fases: a fase de igualdade e a fase de comparação.

Na fase de igualdade, que vai do segundo "0"ao segundo "50", as rotas A e B são tratadas sem qualquer prioridade. Por outro lado, na fase de comparação, que se estende do segundo "51"ao segundo "100", uma das rotas, seja A ou B, recebe prioridade.

Em todas as visualizações gráficas deste capítulo, a rota A foi considerada como a rota prioritária. A rota B, sendo similar à rota A, não necessitou de experimentos adicionais. A situação "2"da Figura 5.2 foi utilizada como modelo para os testes.

6.1 *Throughput* e taxa de perda de pacotes

Nesta seção, foi realizada uma análise do *Throughput* e da taxa de perda de pacotes, para quantificar a quantidade de dados transferidos durante um período de 100 segundos. Esta análise foi realizada para duas rotas distintas: a rota A, que conecta o produtor (nó 200) ao consumidor (nó 100), e a rota B, que conecta o mesmo produtor (nó 200) a um consumidor diferente (nó 300). Esta avaliação nos permitiu comparar o desempenho de transmissão de dados entre diferentes rotas na rede.

6.1.1 Com tráfego tipo I

Durante o experimento, a rota B recebeu 14.125.474 bytes, enquanto a rota A, que foi priorizada, recebeu 15.409.608 bytes. Estes resultados indicam a ocorrência de perda de pacotes. Mais detalhes podem ser encontrados na Tabela 6.1.

A definição do tráfego do tipo I é detalhada no Capítulo 5, intitulado “Metodologia”, deste trabalho. A Figura 6.1 apresenta os resultados obtidos ao executar as rotas

A e B sob a condição de tráfego do tipo I durante um período de 100 segundos. O eixo y representa a quantidade de dados transmitidos, enquanto o eixo x indica o tempo de transmissão em segundos.

Tabela 6.1 – Perda de pacotes I

Rota	Pacotes Perdidos	Pacotes Recebidos	Taxa de perda	Prioridade
A	4072	10471	28%	x
B	4834	9385	33%	

A Figura 6.1 apresenta os resultados de *Throughput* para as rotas A e B. No intervalo de tempo entre o segundo 0 e o segundo 50, ambas as rotas são tratadas de maneira igual, sem priorização. No entanto, a partir do segundo 51, a rota A recebe priorização em relação à rota B. Como consequência, o *Throughput* para a rota A supera o da rota B até o término do experimento. Ao comparar a rota A (com priorização) e a rota B (sem priorização), observa-se que a primeira recebeu 1.284.134 *bytes* a mais do que a segunda, resultando em um ganho de 6%.

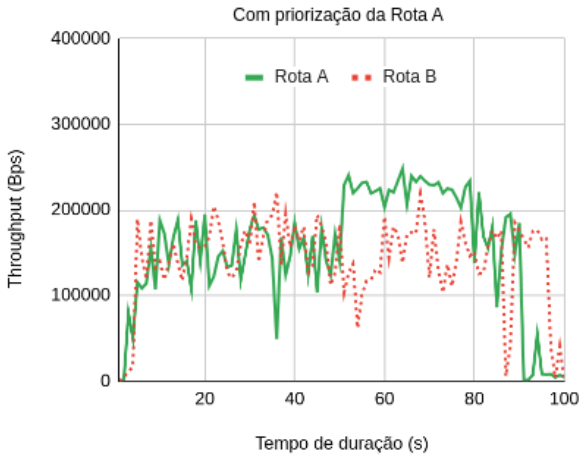

Figura 6.1 – Comparativo entre rotas com tráfego tipo I

6.1.2 Com tráfego tipo II

Uma descrição do tráfego do tipo II está disponível no capítulo 5 Metodologia deste trabalho. A Figura 6.2 mostra o *Throughput* das rotas A e B com este tipo de tráfego em um período de 100 segundos. O eixo y representa a quantidade de dados transferidos e o eixo x o tempo de transmissão em segundos.

A rota B transferiu 16.693.743 *bytes* e a rota A, que foi priorizada, transferiu

18.191.899 *bytes*. Isso indica que ocorreu perda de pacotes, como pode ser observado na Tabela 6.2.

Tabela 6.2 – Perda de pacotes II

Rota	Pacotes Perdidos	Pacotes Recebidos	Taxa de perda	Prioridade
A	2453	13898	15%	x
B	3548	12576	22%	

A Figura 6.2 apresenta os resultados de *Throughput* para as rotas A e B. No intervalo de tempo entre o segundo 0 e o segundo 50, ambas as rotas são tratadas de maneira igual, sem priorização. No entanto, a partir do segundo 51, a rota A recebe priorização em relação à rota B. Como consequência, o *Throughput* para a rota A supera o da rota B até o término do experimento. Ao comparar a rota A (com priorização) e a rota B (sem priorização), observa-se que a primeira transferiu 1.498.156 bytes a mais do que a segunda, resultando em um ganho de 7%.

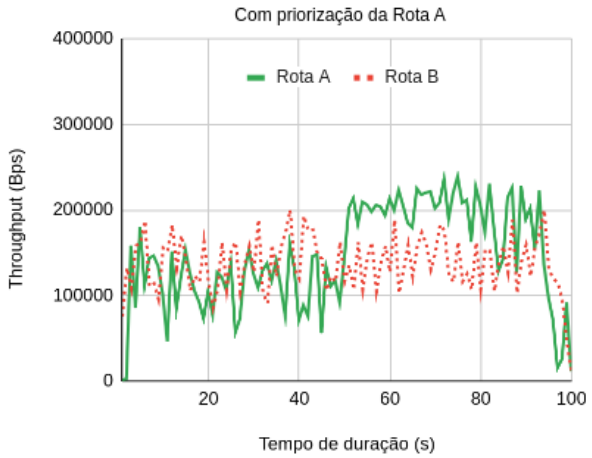

Figura 6.2 – Comparativo entre as rotas com tráfego tipo II

6.1.3 Com tráfego tipo III

A definição do tráfego do tipo III pode ser encontrada no capítulo 5 Metodologia, deste trabalho. A Figura 6.3 mostra o desempenho de *Throughput* para as rotas A e B, sob o cenário de tráfego do tipo III, descrito no capítulo 5. O experimento durou 100 segundos, e os eixos x e y representam, respectivamente, o tempo de transmissão e a quantidade de dados transferidos.

A rota B processou um total de 20.332.122 *bytes*, enquanto a rota A, que foi priorizada, processou 21.402.234 *bytes*. Isso evidencia que não houve perda de pacotes

na rota que recebeu priorização, conforme ilustrado na Tabela 6.3. Notavelmente, a rota A registrou uma perda de pacotes de 0%. Isso pode ser atribuído ao fato de que o tráfego do tipo III concentra um volume menor de tráfego *ICMP*, resultando em uma interferência reduzida na rede.

Tabela 6.3 – Perda de pacotes III

Rota	Pacotes Perdidos	Pacotes Recebidos	Taxa de perda	Prioridade
A	0	18023	0%	x
B	1254	16662	7%	

A Figura 6.3 mostra os resultados de *Throughput* para as rotas A e B. Do segundo 0 ao segundo 50, as rotas comparados não têm priorização. A partir do segundo 51, a rota A é priorizada sobre a rota B. Como resultado, o *Throughput* para a rota A é maior do que para a rota B até o final do experimento. No comparativo entre as rotas, a primeira rota recebeu 1.070.112 *bytes* a mais do que a segunda rota, o que dá um ganho de 5%.

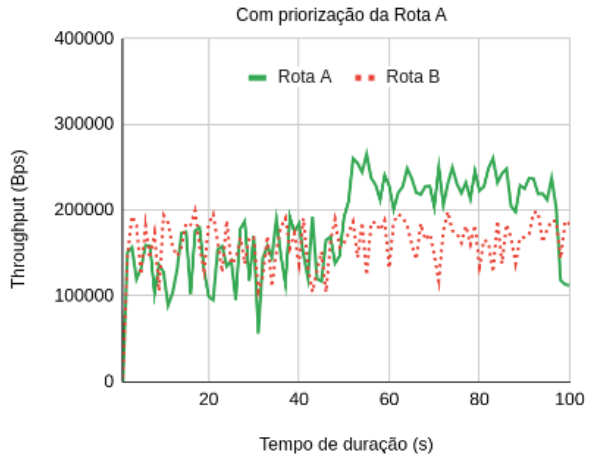

Figura 6.3 – Comparativo entre as rotas com tráfego tipo III

6.1.4 Discussão sobre os resultados das métricas Throughtput e perda de pacotes

O *Throughput* é uma métrica crucial para avaliar a eficiência de uma rede, pois indica a quantidade de dados transferidos com sucesso.

A metodologia empregada neste estudo, embora seja eficaz na obtenção de uma variedade de métricas, impõe limitações na precisão dos resultados do *Throughput*. Isso se deve à criação intencional de três tipos distintos de tráfego na rede.

Os resultados obtidos com a utilização dos tráfegos I, II e III demonstram que o *Throughput* diminui à medida que o congestionamento da rede aumenta. Ao variar os trá-

fegos do mais pesado (I) para o mais leve (III), foram utilizados diferentes intensidades de congestionamento. Estes resultados mostraram ganhos de 6%, 7% e 5%, respectivamente, na rota A (priorizada).

A taxa de pacotes perdidos em uma rede pode ser influenciada pelo tamanho dos pacotes. Pacotes de maior tamanho têm o potencial de aumentar a taxa de perdas, além de elevar a probabilidade de congestionamento e atraso na rede. A rota A (priorizada) obteve taxa de perdas 28%, 15% e 0% como respectivamente resultados de experimentos com essa métrica.

O hardware e o software da rede também podem influenciar os resultados das métricas testadas. Este estudo utilizou *hardware* e *software* com configurações modestas em comparação com equipamentos reais, como servidores, roteadores e *switches*. Portanto, é provável que equipamentos reais possam obter resultados mais expressivos.

6.2 *Jitter*

Neste experimento, foi empregado o *Iperf3*, uma ferramenta de medição de desempenho de rede, para avaliar o *Jitter*. No gráfico resultante, o eixo y representa a média do *Jitter* em milissegundos (ms), enquanto o eixo x ilustra o tempo de transmissão em segundos. Avaliamos especificamente a rota A sob duas condições distintas: com priorização de tráfego e sem priorização. Além disso, foram considerados três diferentes tipos de tráfego durante a análise.

6.2.1 Com tráfego tipo I

A Figura 6.4 ilustra uma comparação do *Jitter* na rota A, considerando dois cenários: com e sem priorização. No experimento sem priorização, representado por uma linha pontilhada na cor vermelha, o tempo total consumido foi de 2.211.929 ms. Por outro lado, no experimento com priorização, indicado pela cor verde, o tempo total consumido foi de 1.739.532 ms. Neste contexto, o teste com priorização resultou em um *Jitter* significativamente menor.

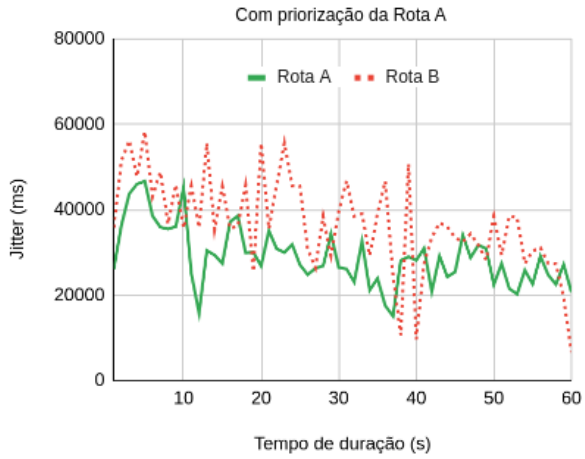

Figura 6.4 – Comparativo com tráfego I.

6.2.2 Com tráfego tipo II

A Figura 6.5 exibe uma comparação do *Jitter* na rota A, em dois cenários distintos: com e sem priorização. No experimento sem priorização, representado por uma linha pontilhada de cor vermelha, o tempo total consumido foi de 1.887.782 ms. Em contraste, no experimento com priorização, indicado pela cor verde, o tempo total consumido foi de 1.365.714 ms. Neste contexto, o teste com priorização resultou em um *Jitter* consideravelmente menor.

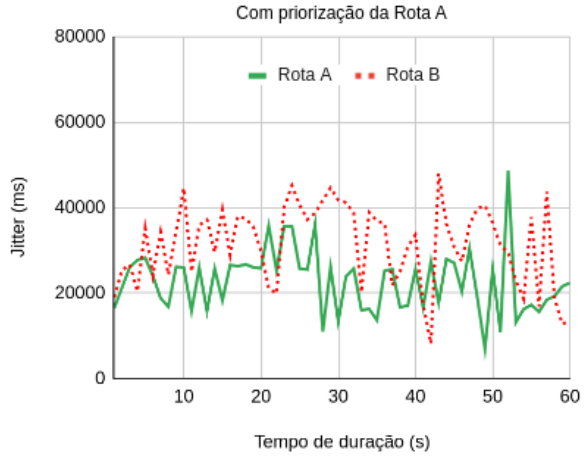

Figura 6.5 – Comparativo com tráfego II.

6.2.3 Com tráfego tipo III

A Figura 6.6 exibe uma comparação do *Jitter* na rota A, sob duas condições distintas: com e sem priorização. No experimento sem priorização, representado por uma

linha pontilhada de cor vermelha, o tempo total consumido foi de 1.947.720 ms. Em contrapartida, no experimento com priorização, indicado pela cor verde, o tempo total consumido foi de 1.286.096 ms. Neste contexto, o teste com priorização resultou em um *Jitter* significativamente menor.

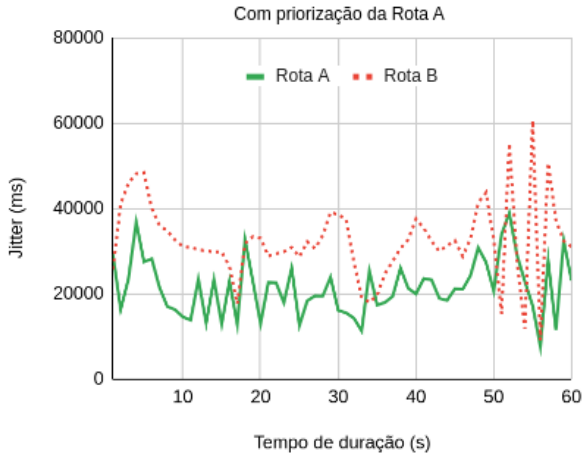

Figura 6.6 – Comparativo com tráfego III.

6.2.4 Discussão sobre os resultados da métrica Jitter

A métrica de *Jitter* é fundamental para a avaliação da qualidade de uma rede, sendo de particular importância para serviços que operam em tempo real, como as transmissões de vídeo e voz. Nos experimentos realizados neste estudo, empregamos arquivos do tipo .mp4. Os Sistemas de Comando e Controle (C2), que necessitam de agilidade e alto desempenho, são exemplos de serviços em tempo real que podem transmitir arquivos de vídeo e voz. A metodologia utilizada neste trabalho, apesar de ser eficaz na obtenção de uma variedade de métricas, impõe limitações na precisão dos resultados do *Jitter*. Isso ocorreu devido à criação intencional de três tipos distintos de tráfego na rede.

Os resultados obtidos nos três testes realizados com a métrica *Jitter* apontam para a necessidade de otimização da rede em diversos aspectos, como congestionamento, roteamento e latência. Além disso, é importante considerar que o uso simultâneo de vários dispositivos em uma rede *Wi-Fi* pode resultar em *Jitter*.

Apesar dos índices de *Jitter* observados nos experimentos serem altos, constatou-se que a rota A, que foi priorizada, apresentou um desempenho superior, com um *Jitter* menor em todas as ocasiões. Isso destaca a importância de estratégias de priorização de tráfego para aprimorar o desempenho da rede.

6.3 Latência

A Figura 6.7 mostra os resultados de atraso para as rotas A e B. Do segundo 0 ao segundo 50, os rotas comparados não têm priorização. A partir do segundo 51, a rota A é priorizada sobre a rota B. Como resultado, o atraso para a rota A é menor que para a rota B até o final do experimento.

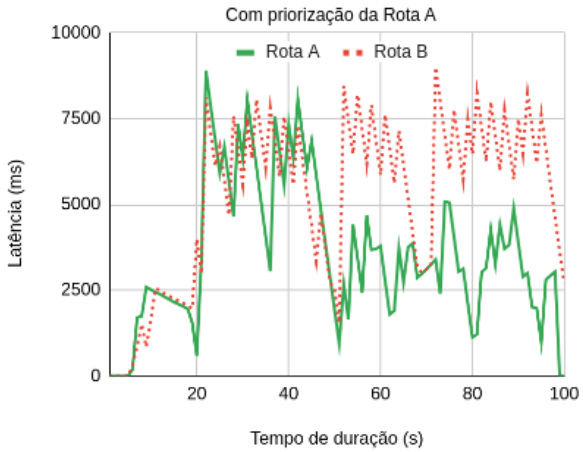

Figura 6.7 – Comparação para a Métrica de Latência.

6.3.1 Discussão sobre os resultados da métrica Latência

A Latência representa o tempo necessário para um pacote de dados viajar de um ponto a outro em uma rede. Este tempo pode ser influenciado pela distância física entre os pontos de origem e destino.

Neste estudo, a latência foi avaliada utilizando o tráfego II, caracterizado por um nível intermediário de congestionamento de rede.

É possível verificar no gráfico A Figura 6.7, que até os 20 segundos de execução as rotas A e B permanecem abaixo de 2500ms e entre o segundo "20"e o segundo "50"eles ficam entre 5000ms e 7500ms. Após os 50 segundos, quando a rota A é priorizada, permanece dentro do padrão que tinha até os 50 segundos, já a rota A após os 50 segundos diminuiu a latência ficando entre 2500ms a 5000ms. Os testes realizados demonstraram que a rota A, que foi priorizada, apresentou uma latência menor. Este resultado linha a relevância de estratégias de priorização de tráfego para otimizar o desempenho da rede.

7 CONCLUSÃO

Este estudo propôs um modelo de aplicação de comando e controle consciente da rede, capaz de impor ajustes à rede para atender aos objetivos da aplicação. Isso foi viabilizado por meio de uma rede reconfigurável, apoiada pelos paradigmas *SDN* e *ICN*. O aplicativo C2-APP instanciou o modelo proposto, se mostrando uma ferramenta eficiente para reconfigurar e modificar as políticas de rede, conforme evidenciado pelos resultados obtidos.

A simulação de três categorias de tráfego na rede confirmou que o desempenho da rede tende a diminuir à medida que o tráfego aumenta, mesmo com a implementação de priorizações. No entanto, a rota com prioridade superou a rota sem prioridade em todas as métricas de rede avaliadas, indicando que a aplicação de priorizações pode ser uma estratégia eficaz em redes críticas.

A agilidade é um atributo essencial em sistemas C2, que exigem velocidade e priorização de dados. Esses fatores são influenciados pelo cenário de aplicação, mudanças situacionais e fatores humanos. Embora a rede possa ser configurada para atender a padrões específicos de priorização de dados, a largura de banda disponível pode ser insuficiente para atender a todos os nós simultaneamente devido a essas variáveis.

Este estudo demonstrou que é possível melhorar o desempenho de uma rede *SDN* através do uso de *QoS*. A implementação de *QoS* provou ser fundamental para alcançar o objetivo de ajustar a rede para atender aos requisitos de desempenho desejados em função de alteração na situação da missão. Os resultados obtidos fornecem evidências do sucesso da abordagem proposta, com a redução da perda de pacotes e do atraso na rota priorizada através dos experimentos realizados. Embora os resultados não tenham atingido valores de melhoria percentualmente tão significativos, a tendência de melhoria foi confirmada. Entretando, a obtenção de resultados mais significativos está condicionada às características e a escala da rede. Desta forma, em outros cenários é possível alcançar resultados superiores. Além disso, a utilização da *REST API* do *ONOS* para integração com o sistema C2-APP provou ser eficaz, resultando em uma arquitetura inovadora e funcional. A integração de SDN com ICN apresentou benefícios, como a facilidade de alterações nas políticas de rede através da programabilidade do *SDN* e a redução do tempo de resposta e distribuição de conteúdo através dos mecanismos providos pelo paradigma *ICN*.

Apesar dos avanços estabelecidos neste trabalho, ele apresenta algumas limitações, incluindo: a utilização de *hardware* com capacidades inferiores em comparação

com dispositivos reais; configurações de rede ajustadas especificamente para este projeto (como largura de banda e *delay*); e a falta de um estudo ligado às questões sobre a mobilidade dos nós da rede. É crucial enfatizar que os dispositivos reais possuem configurações de *hardware* e *software* otimizadas para suas aplicações específicas, o que pode levar a um aumento considerável no desempenho da rede. Na execução deste trabalho, foi empregado um modelo de rede estático. No entanto, em cenários reais, a mobilidade é um fator essencial para o sucesso das missões. Assim, é importante notar que os resultados obtidos neste estudo podem apresentar variações quando aplicados a sistemas reais, devido às diferenças nas configurações e parâmetros de rede adotados.

Esta dissertação apresenta contribuições significativas, incluindo a adoção conjunta das tecnologias de *SDN* e *ICN*. Além disso, propôs-se uma arquitetura baseada em *REST API* para reconfiguração da rede. A proposta para a implementação conjunta de *ICN* e *SDN* foi projetada para ser aplicável em cenários reais, aproveitando efetivamente as vantagens inerentes a essas tecnologias. A integração eficaz do sistema C2-APP com a *IoBT* foi alcançada através da adoção da arquitetura baseada em *REST API*. O uso dessas tecnologias altamente configuráveis permitiu a reconfiguração da rede *SDN* através de modificações nas políticas de rede. Isso demonstra a flexibilidade e adaptabilidade da abordagem proposta.

As direções de trabalhos futuros apontam para a implantação da abordagem proposta em dispositivos do mundo real, lidando com desafios de implementação prática. Existe ainda a possibilidade da comparação objetiva com trabalhos relacionados. Além disso, a migração do controlador *SDN* (sincronização de controladores distribuídos) em casos de particionamento de rede (quando houver perda de conexão com a rede) ainda é uma questão em aberto que precisa ser abordada.

REFERÊNCIAS

ABBOU, T.; SONG. A software-defined queuing framework for qos provisioning in 5g and beyond mobile systems. **IEEE Network**, v. 35, n. 2, p. 168–173, 2021.

ALBERTS, D. et al. **Command and Control (C2) Agility**. 2014.

CHOUDHARI, C.; NITURE. Disruption tolerant network (dtn) for space communication: An overview. In: **2022 IEEE 7th International conference for Convergence in Technology (I2CT)**. [S.l.: s.n.], 2022. p. 1–5.

GIBSON, C. et al. Opportunities and challenges for named data networking to increase the agility of military coalitions. In: **2017 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computed, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)**. [S.l.: s.n.], 2017. p. 1–6.

HASAN, M. et al. Sdn mininet emulator benchmarking and result analysis. In: **2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)**. [S.l.: s.n.], 2020. p. 355–360.

IRFAN, T. et al. Onos intent path forwarding using dijkstra algorithm. In: **2019 International Conference on Electrical Engineering and Informatics (ICEEI)**. [S.l.: s.n.], 2019. p. 549–554.

JHA, P. **End-to-end Quality-of-Service in Software Defined Networking**. 2017. Disponível em: <https://publications.scss.tcd.ie/theses/diss/2017/TCD-SCSS-DISSERTATION-2017-040.pdf>.

KOTT, A.; SWAMI, A.; WEST, B. J. The internet of battle things. **Computer**, v. 49, n. 12, p. 70–75, 2016.

LEAL, G. M. **Abordagem baseada em redes centradas à informação e redes definidas por software para suporte de aplicações militares críticas**. 2020. Disponível em: <http://hdl.handle.net/10183/216997>.

LEE, C.-E.; BAEK JEANY SON, Y.-G. H. J. The journal of supercomputing. In: **Deep AI military staff: cooperative battlefield situation awareness for commander's decision making.** [S.l.: s.n.], 2023. v. 79.

MAMUSHIANE, L.; SHOZI, T. A qos-based evaluation of sdn controllers: Onos and opendaylight. In: **2021 IST-Africa Conference (IST-Africa)**. [S.l.: s.n.], 2021. p. 1–10.

PALIWAL, M.; SHRIMANKAR, D.; TEMBHURNE, O. Controllers in sdn: A review report. **IEEE Access**, v. 6, p. 36256–36270, 2018.

PAPAKOSTAS, D. et al. Backbones for internet of battlefield things. In: **2021 16th Annual Conference on Wireless On-demand Network Systems and Services Conference (WONS)**. [S.l.: s.n.], 2021. p. 1–8.

RéVAY, M.; LíšKA, M. Ooda loop in command amp; control systems. In: **2017 Communication and Information Technologies (KIT)**. [S.l.: s.n.], 2017. p. 1–4.

S., S. H.; BOREGOWDA, U. Quality-of-service-linked privileged content-caching mechanism for named data networks. **Future Internet**, v. 14, n. 5, 2022. ISSN 1999-5903. Disponível em: <https://www.mdpi.com/1999-5903/14/5/157>.

SECURITY, U. D. of H. **GPS Blue Force Tracking Systems**. 2014. Disponível em: <https://www.dhs.gov/sites/default/files/publications/GPS-Blue-Force-AppN_0414-508_0.pdf>.

STOCCHERO, J. et al. Combining information centric and software defined networking to support command and control agility in military mobile networks. **Peer-to-Peer Networking and Applications**, v. 16, p. 1–20, 01 2023.

STOCCHERO, J. M. et al. Secure command and control for internet of battle things using novel network paradigms. **IEEE Communications Magazine**, v. 61, n. 5, p. 166–172, 2023.

TEAM, M.-N. **Mini-NDN: A Mininet-based NDN emulator**. 2022. Disponível em: <https://minindn.memphis.edu/>.

UJCICH, B. E.; SANDERS, W. H. Data protection intents for software-defined networking. In: **2019 IEEE Conference on Network Softwarization (NetSoft)**. [S.l.: s.n.], 2019. p. 271–275.

VACHUSKA, T. **Open Network Operating System**. 2022. Disponível em: <https://opennetworking.org/onos/>.

YU, C. et al. Leveraging energy, latency, and robustness for routing path selection in internet of battlefield things. **IEEE Internet of Things Journal**, v. 9, n. 14, p. 12601–12613, 2022.

ZHANG, Z. et al. In-network caching for icn-based iot (icn-iot): A comprehensive survey. **IEEE Internet of Things Journal**, v. 10, n. 16, p. 14595–14620, 2023.

ZHAO, Y. et al. Service-oriented intelligent ooda loop. In: **2023 26th ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)**. [S.l.: s.n.], 2023. p. 182–187.

ZHOU, W. et al. A reconciliation model of agile c2 organization based on converged networks. In: **2020 6th International Conference on Big Data and Information Analytics (BigDIA)**. [S.l.: s.n.], 2020. p. 58–65.