UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

SAMUEL HENRIQUE DALMAS

GERENCIAMENTO DE REDES EM UM AMBIENTE DE COMPUTAÇÃO EM NÉVOA

CAXIAS DO SUL

2022

SAMUEL HENRIQUE DALMAS

GERENCIAMENTO DE REDES EM UM AMBIENTE DE COMPUTAÇÃO EM NÉVOA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Dra. Maria de Fátima Webber Do Prado Lima

CAXIAS DO SUL

2022

SAMUEL HENRIQUE DALMAS

GERENCIAMENTO DE REDES EM UM AMBIENTE DE COMPUTAÇÃO EM NÉVOA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 27/06/2022

BANCA EXAMINADORA

Prof. Dra. Maria de Fátima Webber Do Prado Lima
Universidade de Caxias do Sul - UCS

Prof. Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

Prof. Dr. André Luis Martinotto
Universidade de Caxias do Sul - UCS

AGRADECIMENTOS

Agradeço primeiramente à minha família, que sempre me apoiou na minha trajetória como estudante.

Agradeço especialmente à minha tia-avó Maria Cândida Vieira Ramos que proporcionou a oportunidade de eu cursar o ensino superior.

Agradeço à Profa. Dra. Maria de Fátima Webber do Prado Lima pela dedição e suporte na orientação desse trabalho.

Agradeço também a todos professores e colegas que tive a oportunidade de conviver durante os anos de graduação.

“We can only see a short distance ahead, but we can see plenty there that needs to be done”

Alan Turing

RESUMO

Este trabalho possui como objetivo analisar e avaliar softwares de monitoramento e gerenciamento de redes em uma arquitetura de computação em Névoa com o intuito de identificar qual mais se adequa aos requisitos desse ambiente computacional. Para embasar esta avaliação, foram estudados conceitos de computação em Névoa, softwares de monitoramento e normas ISO/IEC de avaliação de *software*. Para gerar o entendimento dos requisitos a serem monitorados e gerenciados, foi realizada uma pesquisa na forma de Revisão Sistemática, na qual foram levantados diversos dos requisitos necessários para gerenciar o controle de *Quality of Service* (QoS) nos ambientes de Névoa, além das ferramentas a serem testadas com base no trabalho de diversos autores. Foi utilizada a base da CAPES com as palavras chave: *"Fog Computing"*, *"Requirements"*, *"QoS"* e *"Management"*. Os requisitos elencados foram divididos em categorias com base em semelhança, por fim, foram selecionadas as categorias com maior quantidade de menções que se adequam ao cenário disponível para teste. Esses requisitos elencados foram separados entre agentes, rede e servidor. A seleção das ferramentas foi baseada na quantidade de repetições encontradas na revisão bibliográfica, compatibilidade com sistema operacional Linux e possuir código livre ou com versão de teste disponível, além de contar com documentação para auxiliar no seu uso. Dessa maneira, foram selecionadas as ferramentas: Ganglia, Nagios e Zabbix. A partir desses estudos, foram definidos métricas, critérios e casos de testes para a avaliação. O ambiente de teste contou com um servidor local de Névoa, dois nodos de Névoa, um ponto de acesso e um servidor de Nuvem. O servidor da Névoa foi virtualizado através da ferramenta VirtualBox, para o servidor de Nuvem, foi utilizada uma máquina virtual no Google Cloud, enquanto que para os agentes, não foi utilizada virtualização. As ferramentas para teste foram instaladas no servidor local. Para simular um ambiente de Névoa e simular o tráfego de um ambiente característico, com alto fluxo de dados, foi desenvolvida uma aplicação em Python dividida em três módulos, o módulo agente coleta dados de uso de CPU e envia para o servidor de Névoa que por sua vez, calcula a média aritmética de cada máquina agente durante o período de um minuto e após, envia essas médias ao servidor de Nuvem, que armazena as informações em um banco de dados. Após realizados os testes, foi conduzida a avaliação das ferramentas através da norma ISO/IEC 25010 que define as características de qualidade para um produto de software. Foi escolhida a característica de funcionalidade com o intuito de avaliar se as ferramentas se adequam à arquitetura de Névoa. Dentro dessa característica, foram selecionadas duas subcaracterísticas: adequação e interoperabilidade. Para avaliar esses itens, foram adotadas métricas externas, que preveem executar as ferramentas e coletar dados no ambiente de execução. Os resultados alcançados permitiram constatar que nenhuma das ferramentas é adequada a um ambiente de Névoa. A ferramenta que obteve melhor resultado foi o Zabbix, seguida pelo Nagios e pelo Ganglia, que obteve a última colocação.

Palavras-chave: Computação em Névoa. Computação em Nuvem. Softwares de Monitora-

mento. Softwares de Gerenciamento.

ABSTRACT

This work aims to analyze and evaluate network monitoring and management software in a Fog computing architecture in order to identify which one best suits the requirements of this computing environment. To support this evaluation, concepts of Fog computing, monitoring software and ISO/IEC standards for software evaluation were studied. To generate an understanding of the requirements to be monitored and managed, a survey was carried out in the form of a Systematic Review, in which several of the requirements necessary to manage the control of QoS in Fog environments were raised, as well as the tools to be tested, based on the work of several authors. The CAPES database was used with the keywords: "Fog Computing", "Requirements", "QoS" and "Management". The listed requirements were divided into categories based on similarity, finally, the categories with the highest number of mentions that fit the scenario available for testing were selected. These listed requirements were separated between agents, network and server. The selection of the tools was based on the number of repetitions found in the literature review, compatibility with the Linux operating system and having free code or a trial version available, in addition to having documentation to assist in its use. In this way, the tools were selected: Ganglia, Nagios and Zabbix. Based on these studies, metrics, criteria and test cases were defined for the evaluation. The test environment had a local Fog server, two Fog nodes, an access point and a Cloud server. The Fog server was virtualized through the VirtualBox tool, for the Cloud server, a virtual machine was used on Google Cloud, while for the agents, virtualization was not used. Testing tools were installed on the local server. To simulate a Fog environment and simulate the traffic of a characteristic environment, with high data flow, a Python application was developed divided into three modules, the agent module collects CPU usage data and sends it to the Fog server, which by in turn, calculates the arithmetic average of each agent machine during a period of one minute and then sends these averages to the Cloud server, which stores the information in a database. After the tests were carried out, the tools were evaluated using the ISO/IEC 25010 standard that defines the quality characteristics for a software product. The feature of functionality was chosen in order to assess whether the tools are suitable for the Fog architecture. Within this characteristic, two sub-characteristics were selected: adequacy and interoperability. To evaluate these items, external metrics were adopted, which provide to run the tools and collect data in the execution environment. The results achieved showed that none of the tools is suitable for a Fog environment. The tool that got the best result was Zabbix, followed by Nagios and Ganglia, which got the last place.

Keywords: Fog Computing. Cloud Computing. Monitoring Software. Management Software.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CDN *Content Delivery Network*

HAN *Home-Area Network*

ICN *Information Centric Network*

IaaS *Infrastructure as a Service*

IoT *Internet of Things*

Kbps *Quilobit por segundo*

Mbps *Megabit por segundo*

MIB *Management Information Base*

MonALISA *Monitoring Agents in a Large Integrated Services Architecture*

OID *Object Identifier*

PaaS *Platform as a Service*

PCMONS *Private Cloud Monitoring System*

QoE *Quality of Experience*

QoS *Quality of Service*

RFC *Request for Comment*

Saas *Software as a Service*

SLA *Service Level Agreement*

SLO *Service Level Objectives*

SNMP *Simple Network Management Protocol*

SDN *Software Defined Network*

UDP *User Datagram Protocol*

VoIP *Voice Over IP*

WSN *Wireless Sensor Network*

SUMÁRIO

1 INTRODUÇÃO

A evolução e popularização de tecnologias de telecomunicações como redes celular, redes sem fio e ótica culminou na maior disponibilidade de acesso à Internet nas várias regiões do país, inclusive em locais distantes, como o meio rural (CONSTANTINO, 2021). O cenário atual da tecnologia, com gradativo aumento de usuários conectados à Rede Mundial de Computadores, faz aumentar a dependência de sua utilização no trabalho, na educação, no lazer, etc.. Além disso, o crescimento do número de dispositivos inteligentes que trocam informações entre si e enviam dados para servidores externos, para análise e processamento, está sugerindo uma redesignação e adequação da estrutura de acesso à Nuvem (BARRETO, 2020).

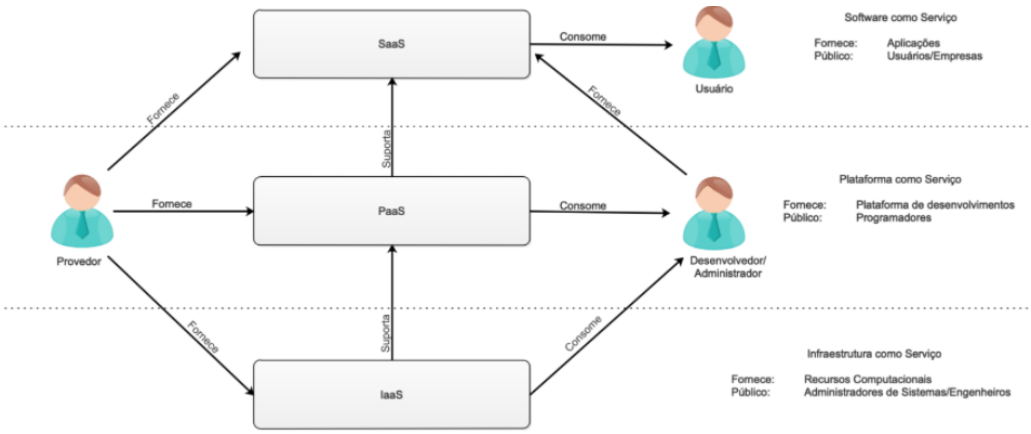
Os dados gerados em IoT são contínuos, sendo geralmente do tipo numérico provindo de sensores, gerando assim um fluxo de informações que possui tráfego constante e geralmente de alta velocidade. A diversidade dos dispositivos também é uma característica a ser considerada pois pode haver divergências nas implementações dos protocolos de comunicação de cada fabricante, resultando em dificuldade de comunicação entre os aparelhos da rede (ALENCAR, 2021).

A maior parte dos dados que são gerados por dispositivos IoT são transmitidos para a Nuvem para processamento e análise. Atualmente a Internet não está preparada para suportar esse aumento de demanda já que não é escalonável nem eficiente o suficiente. Isso reflete diretamente na arquitetura da Computação em Nuvem que necessita baixa latência, mobilidade e alto fluxo de dados, sendo difícil de manter métricas de QoS. Dessa forma, o acesso direto à Nuvem torna-se inviável para aplicações que necessitem desses requisitos (BARRETO, 2020).

Nos últimos anos, muitas soluções de software migraram para a Nuvem de forma a gerar diversas facilidades perceptíveis no cotidiano dos usuários e desenvolvedores. O rápido acesso à informação, a facilidade de gerenciamento de *backups* e a centralização de recursos são algumas das diversas vantagens da sua utilização (P.SAHARAN; KUMAR, 2015).

Dois modelos de funcionamento, de Serviço e de *Deploy*, são possíveis para a arquitetura de Nuvem. Cada um possui suas vantagens, desvantagens e aplicabilidade (P.SAHARAN; KUMAR, 2015). Os modelos de serviço contemplam diversas modalidades dentre as quais: *Software as a Service* (Saas), *Platform as a Service* (PaaS) e *Infrastructure as a Service* (IaaS), como mostra a Figura 1. Já os Modelos de *Deploy* são identificados de acordo com a infraestrutura em uso, dividindo-se em: Nuvem Privada, Nuvem Comunitária, Nuvem Pública e Nuvem Híbrida (Figura 2) (P.SAHARAN; KUMAR, 2015).

Figura 1 – Modelos de Serviço da Computação em Nuvem



Fonte: (CONSTANTINO, 2021).

Figura 2 – Modelos de *Deploy* da Computação em Nuvem

Fonte: (CONSTANTINO, 2021).

Algumas limitações são inerentes ao modelo, como segurança e privacidade; aspectos técnicos como velocidade de conexão; segregação dos dados; localização, sendo comum a operadora estar em países diferentes dos usuários; entre outros. Dessa forma, aplicações que exigem alta disponibilidade, mobilidade e baixo atraso de comunicação entre usuário e servidor saem prejudicadas.

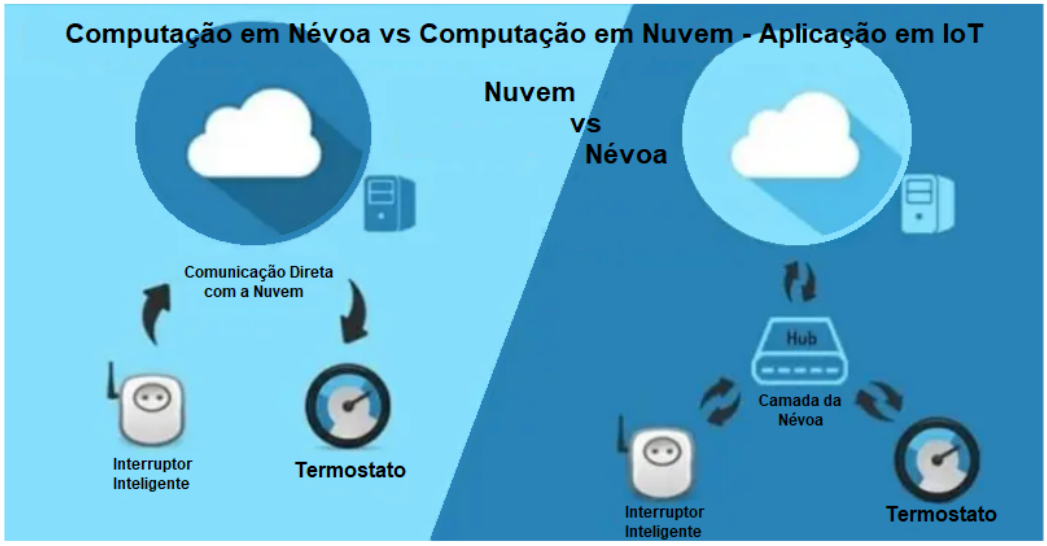
A implantação de recursos na borda da infraestrutura da rede [1] como processamento, armazenamento, etc. surge para superar esses limites impostos pela arquitetura de Nuvem. A utilização da borda confere novas possibilidades de uso já que pode servir como um elo de ligação entre dispositivos e a Nuvem oferecendo alto desempenho.

Diante dessas constatações, a Computação em Névoa surge para driblar esses empecilhos, fornecendo uma estrutura descentralizada, com recursos flexíveis e adaptáveis às diversas necessidades dos usuários (SILVA, 2021). Diferentemente de recursos em Nuvem que se dispõem na rede principal, são propostos *gateways* que oferecem recursos na borda da rede como armazenamento e processamento (CONSTANTINO, 2021).

[1] Interface entre a rede local e a Internet (KUROSE; ROSS, 2013).

Dessa maneira, torna-se possível trabalhar os dados localmente, antes de serem enviados para a Nuvem, o que contribui para a diminuição da latência e da quantidade de informações trafegadas (ALENCAR, 2021). A carga nos servidores em nuvem é diminuída, tornando possível balancear processamento e armazenamento em estrutura local e remota (PRAKASH *et al.*, 2017). A Figura 3 mostra a diferença entre as modalidades de Nuvem e Névoa.

Figura 3 – Comparação entre Computação em Névoa e Computação em Nuvem



Fonte: Adaptado de https://www.fossguru.com/fog-computing-vs-cloud-computing-iot/ Acesso em: 18 agosto 2021.

Outros aspectos de comparação entre Nuvem e Névoa são evidenciados por (PRAKASH *et al.*, 2017) (Quadro 1), o qual ressalta algumas vantagens diretas entre a adoção de uma arquitetura ou de outra. Ao optar pela Névoa, ficam claras algumas características principais como a diminuição de saltos, o suporte para localização e mobilidade bem como a maior segurança.

Quadro 1 – Diferenças das Arquiteturas de Nuvem e de Névoa

Característica	**Computação em Nuvem**	**Computação em Névoa**
Localização Nodos Servidores	Na Internet	Na borda da rede
Distância entre Cliente e Servidor	Múltiplos saltos	Único ou múltiplos saltos
Latência	Alta	Baixa
Jitter	Alto	Muito baixo
Segurança	Menos seguro	Mais seguro
Detecção de Localização	Não	Sim
Vulnerabilidade	Alta probabilidade	Baixa probabilidade
Distribuição Geográfica	Centralizada	Densa e distribuída
Número de Nodos Servidores	Poucos	Diversos
Interações em Tempo Real	Suportado	Suportado
Tipo de Conectividade Final	*Leased Line*	Sem fio
Mobilidade	Pouco suportado	Suportado

Fonte: (PRAKASH *et al.*, 2017).

A adoção da Computação em Névoa apresenta vantagens como manter os dados próximos aos usuários, eliminando o atraso de comunicação com servidores distantes, fornecendo suporte para mobilidade e gerando economia de armazenamento de informações. O uso desse novo paradigma é vantajoso em diversos segmentos por suportar aplicações de IoT e de tempo real, mantendo a baixa latência, proporcionando assim, estender os serviços de Nuvem de forma a atuar como um intermediador entre esses serviços e os usuários finais.

Alguns aspectos merecem atenção especial. Em relação à segurança, como mencionado no Quadro 1, a Névoa fornece um melhor aproveitamento em relação à Nuvem. Para isso, é necessário implementar métodos de encriptação como o *Home-Area Network (HAN)*, proteção contra máquinas intrusas e sistemas de detecção de intrusão em todas as camadas da plataforma, tendo em vista que trata-se de uma arquitetura não centralizada. Um dos problemas que pode ocorrer facilmente, caso a configuração não seja feita de forma adequada, é a inclusão de dispositivos com IPs falsos para acessar dados dos servidores. Outro fator a ser abordado é relacionado ao uso de energia. Como esses ambientes usam diversas máquinas em sua estrutura, elaborar um plano energético torna-se fundamental para evitar despesas desnecessárias por má gestão dos recursos (PRAKASH *et al.*, 2017).

A utilização da Computação em Névoa para suportar *Big Data*, por exemplo, mostra-se eficaz sendo possível diminuir tráfego de rede, *delay* e velocidade de processamento já que grandes volumes de dados são trabalhados diariamente (PRAKASH *et al.*, 2017).

Alencar (2021) e Prakash *et al.* (2017) também afirmam que em situações com grande número de sensores é possível otimizar o processamento realizando parte de forma local, sendo

assim, possível diminuir o tempo para predizer informações a partir desses dados. Isso é essencial para ambientes críticos como sistemas de segurança para máquinas industriais, carros que possuem funções autônomas, meio agrícola e da saúde.

Atualmente, estão surgindo novas soluções comerciais que aplicam esse novo conceito de computação de Névoa. A CISCO possui a plataforma CISCO IOx (RAD; SHAREEF, 2017), que visa justamente preencher a lacuna de processamento de tempo real para dispositivos de IoT já que proporciona processamento local diminuindo a carga da rede e dos servidores da Nuvem.

Gerenciar ambientes de IoT já se tornou uma realidade para diversos profissionais dado o crescimento do mercado de dispositivos inteligentes. Gerir redes para comportar as novas necessidades demandadas vem sendo um fator de suma importância para manter a manutenção e qualidade de serviços providos. Os aspectos gerenciais relacionados estão tornando-se mais custosos e complexos proporcionalmente ao crescimento e à heterogeneidade apresentados. Dessa forma, fica evidente o aumento de custos e a probabilidade de erros humanos no gerenciamento desses ambientes (TIBúRCIO; SANTOS; FERNANDES, 2017).

Os novos requisitos de gerência de redes ficam claros quando considerados sistemas de IoT, que pelas características de tráfego de dados mencionadas, mostra-se como um cenário desafiador que muitas vezes foge dos procedimentos convencionais adotados por profissionais da área. Assim, torna-se necessário uma análise de ferramentas e tecnologias viáveis que atuem sobre essa nova realidade e que sejam capazes de satisfazer suas necessidades específicas.

O gerenciamento de redes provém de meados da década de 80, quando surgiram os primeiros sistemas voltados à redes distribuídas. Atualmente, devido à grande complexidade e extensão das redes, são necessárias ferramentas que permitam monitorar seu funcionamento de forma a garantir *QoS* além de aspectos técnicos como tratar dados em tempo real, distribuir processamento e gerenciar a segurança de forma descentralizada.

Algumas ferramentas estão consolidadas nesse segmento, como o Nagios, o Zabbix, o SolarWinds, o Datadog, etc. os quais vêm sendo amplamente utilizados por profissionais da área por vários anos. São ferramentas robustas, de ampla aplicabilidade e confiabilidade para as necessidades comuns de mercado (KEARY, 2021).

Segundo Abreha (2020), essas ferramentas possuem uma estrutura hierárquica e centralizada que acabam não sendo flexíveis o suficiente. Além disso, há peculiaridades que dificultam seu uso em ambientes de Névoa como a difícil configuração, a falta de descoberta automática de dispositivos que, se existentes, costumam ser lentas e viáveis apenas para estruturas fixas de hardware ou que sofram poucas mudanças. A adaptabilidade somente para monitorar recursos que não sofrem altas taxas de atualizações é outro empecilho. Dessa maneira, é possível verificar que as ferramentas usuais não se adequam a esses ambientes em sua totalidade.

1.1 OBJETIVOS

O objetivo geral do trabalho é analisar se os softwares de gerenciamento de redes mais utilizados no mercado suportam os requisitos necessários para gerenciar redes de Computação em Névoa.

Para cumprir o objetivo proposto, três objetivos específicos foram designados:

1. Definir métricas para gerenciamento de ambientes computacionais com Arquitetura de Névoa;
2. Analisar ferramentas de monitoramento e gerenciamento já presentes no mercado e verificar se são viáveis para redes em Névoa;
3. Definir, implantar e testar um ambiente que aplique essas soluções.

1.2 ESTRUTURA DO TRABALHO

Este trabalho é composto por 6 capítulos. O Capítulo 1 apresenta a motivação do trabalho através da contextualização de aplicações de IoT e de uma comparação entre computação em Nuvem e Névoa.

No Capítulo 2, é realizado um estudo teórico sobre computação em Névoa, buscando aprofundar a arquitetura computacional, os componentes e seu funcionamento. Além disso, foi apresentada sua utilização atual no mercado, necessidades, vantagens e desvantagens de seu uso. Nesse capítulo, também foram levantados os requisitos de computação em Névoa que devem ser contemplados no gerenciamento. Baseado nesses requisitos, uma pesquisa sobre as ferramentas de monitoramento de redes foi realizada.

No Capítulo 3, foram elencados os critérios utilizados para selecionar as ferramentas de gerenciamento para teste. Dessa forma, as ferramentas foram definidas de forma a garantir que possam ser mantidos testes compatíveis e fidedignos nas etapas posteriores. O ambiente de testes das ferramentas também foi definido bem como, foram elencados os casos de testes utilizados para validar as ferramentas além de ser explicado o método de avaliação com base em normas ISO/IEC.

O Capítulo 4 relata a execução dos casos de testes em cada uma das ferramentas selecionadas. Esses testes foram realizados em um ambiente de Névoa simulado e adequado para verificar a abrangência dos requisitos.

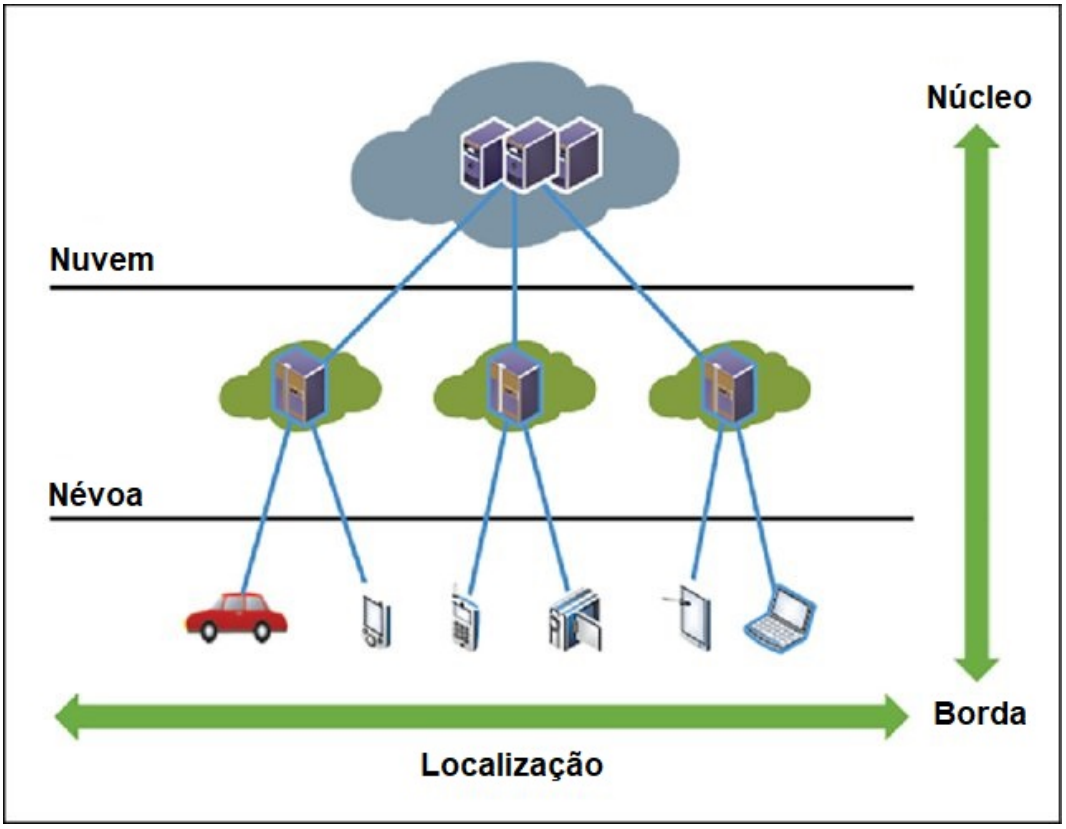
O Capítulo 5 apresenta os resultados obtidos com a realização dos testes bem como detalha a análise desses resultados com base nas normas adotadas. Ao final, é realizada uma classificação de acordo com o método de avaliação definido.

Por fim, o Capítulo 6 faz o encerramento do estudo realizado através de um panorama geral dos resultados obtidos durante seu desenvolvimento, além de apresentar sugestões de trabalhos futuros.

2 COMPUTAÇÃO EM NÉVOA

A Computação em Névoa está sendo considerada uma evolução da Computação em Nuvem. Essa tecnologia tem sido denominada “Nuvem próxima ao chão” pois está localizada entre os servidores de Nuvem e os usuários finais, conforme ilustra a Figura 4 (P.SAHARAN; KUMAR, 2015).

Figura 4 – Estruturação da Computação em Névoa



Fonte: Adaptado de P.Saharan & Kumar (2015).

P.Saharan & Kumar (2015) mencionam que Computação em Névoa deve ser entendida como uma extensão do modelo de Nuvem com algumas novas funcionalidades para contemplar limitações existentes e não um substituto para o modelo anterior, sendo mais performático e viável para alguns ambientes como o de Internet das Coisas. Até o momento, serve como um suporte para sistemas em Nuvem.

Bonomi *et al.* (2012) define essa arquitetura como “plataforma virtualizada que fornece recursos de computação, armazenamento e comunicação entre os dispositivos finais e os tradicionais *Data Centers* em Nuvem, normalmente localizados, mas não exclusivamente, na borda da rede”.

Para aprofundar o estudo da Computação em Névoa, este capítulo foi dividido em se-

ções. A Seção 2.1 descreve a arquitetura e os componentes de sistemas em Névoa. A Seção 2.2 explora o cenário atual da sua utilização. A Seção 2.3 apresenta a taxonomia da Névoa de acordo com a arquitetura, a infraestrutura e os algoritmos em uso. A Seção 2.4 menciona os requisitos considerados para o gerenciamento de redes nesses ambientes. Já a Seção 2.5 apresenta as considerações finais sobre o assunto.

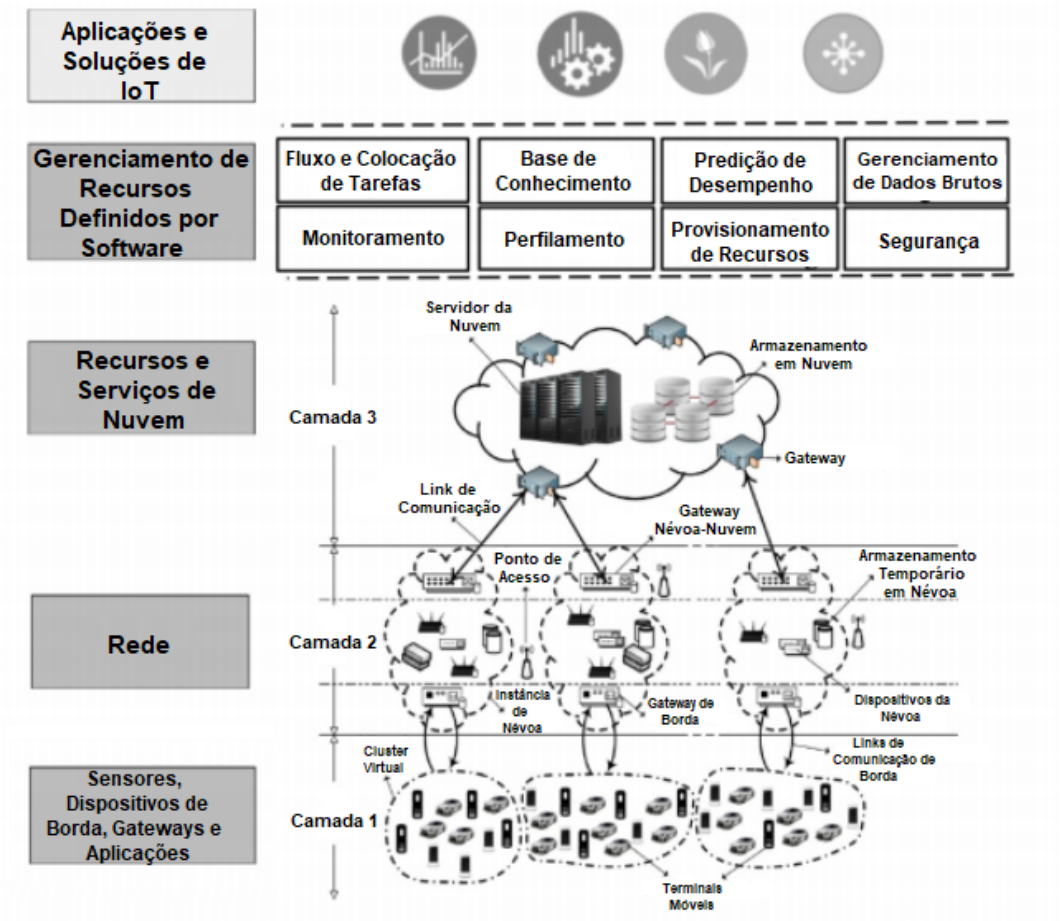
2.1 ARQUITETURA E COMPONENTES

Kashani, Rahmani & Navimipour (2020) explicam que a arquitetura de Névoa é formada por nodos físicos ou virtuais. Dentre os físicos, destacam-se servidores, roteadores, *gateways* e *switches* enquanto que os virtuais são compostos de *switches* virtualizados e máquinas virtuais. No geral, esses nodos comunicam-se com dispositivos finais oferecendo recursos computacionais, gerenciamento de dados ou serviços de comunicação entre dispositivos finais e a Névoa ou Nuvem. Além disso, esses nodos podem operar de maneira centralizada ou descentralizada. Para executar tarefas de Névoa, precisam ser capazes de executar pelo menos uma das funcionalidades:

1. Capacidade de operar independentemente e com tomada de decisões a nível de nodo local ou a nível de *cluster*;
2. Capacidade de desenvolvimento em ambientes diversos e estendidos;
3. Capacidade de suportar estruturas hierárquicas de diferentes camadas com o intuito de possibilitar diversas funcionalidades;
4. Capacidade de gerenciamento e orquestração para operação automática;
5. Capacidade programável em diferentes níveis por usuários, operadores de rede ou provedores de equipamentos.

Coutinho, Carneiro & Greve (2016), Constantino (2021) e Sarkar, Chatterjee & Misra (2015) descrevem a arquitetura da Computação em Névoa em 5 camadas (Figura 5). Na mais baixa, identificada como camada 1 ficam localizados dispositivos locais inteligentes como, sensores, *gateways*, veículos, celulares, dispositivos de borda e aplicações que podem ser instaladas nos dispositivos finais para aprimorar e ampliar sua funcionalidade.

Figura 5 – Componentes da Computação em Névoa



Fonte: Adaptado de Sarkar, Chatterjee & Misra (2015).

A camada 2 é utilizada para comunicar dispositivos da camada anterior, servindo como mediadora entre os dispositivos da camada inferior e a camada mais alta. Além disso, pode prover recursos de rede virtualizados de forma a ser possível armazenar e realizar processamento de dados provindos da camada abaixo para posterior encaminhamento para a Nuvem. A principal adição na arquitetura de Névoa para uma arquitetura comum é o uso de roteadores e *gateways* inteligentes na borda da rede que conectam dispositivos locais e as soluções de Nuvem. Essa é a característica fundamental para reduzir o tráfego entre dispositivos locais e os servidores externos. Além disso, por usar recursos em borda da rede, a latência, a disponibilidade e o desempenho da rede são aperfeiçoados (CONSTANTINO, 2021).

A camada 3 é o próprio sistema de Computação em Nuvem, envolvendo todos serviços e recursos disponíveis para o usuário final, podendo também apresentar tecnologias para aperfeiçoar o uso de recursos como *Software Defined Network (SDN)*, permitindo o suporte ao processamento de dados provenientes de IoT (*data center*, armazenamento, *gateway* de Computação em Nuvem, etc.).

A camada 4 refere-se aos softwares utilizados para gerenciar recursos que coordenam a

infraestrutura e que oferecem *QoS*. Coutinho, Carneiro & Greve (2016) salientam a importância dessa camada para o benefício das aplicações de forma a reduzir a carga de trabalho dos servidores de Nuvem, transferindo as tarefas para os nós da arquitetura de Névoa, melhorando assim a performance e o *delay*. Para isso, os autores definem uma série de serviços de referência para alcançar esses objetivos:

1. Localização e Fluxo de Tarefas: armazena informação acerca dos estados de recursos de Nuvem e Névoa utilizando o Serviço de Monitoramento. Sendo assim, pode definir a distribuição de tarefas;
2. Base de Dados de Conhecimento: guarda informações de histórico das aplicações e recursos;
3. Previsão de Desempenho: possui a responsabilidade de consultar Bases de Conhecimento para fazer uma estimativa do desempenho de recursos disponíveis;
4. Gerenciamento de Dados: oferece informações a outros serviços através de consulta a Banco de Dados;
5. Monitoramento: possui a função de controlar o desempenho das aplicações, fornecendo informações associadas;
6. Gerenciador de Perfis: coordena perfis para os recursos através de informações provenientes da Base de Conhecimento e Monitoramento de Serviços;
7. Provisionamento de Recursos: realiza aquisição de recursos na Névoa, Nuvem e rede para hospedar aplicações. A decisão sobre os recursos é baseada sobre requisitos de latência;
8. Segurança: oferece serviços de autenticação, criptografia, etc. para todos os elementos da Névoa.

Na camada mais superior, são dispostas as aplicações que fazem uso da arquitetura de Computação em Névoa para oferecer serviços inteligentes e inovadores para o usuário final.

Coutinho, Carneiro & Greve (2016) e Bonomi *et al.* (2012) reiteram alguns requisitos para a arquitetura de Névoa:

1. Reconhecimento de Localidade e Baixa Latência: envolve pontos de acesso sofisticados na borda de rede com capacidade de prestar serviços para o usuário;
2. Distribuição Geográfica: relacionado às implantações distribuídas possibilitando por exemplo, fornecer serviços de *streaming* para veículos em movimento;
3. Suporte a Redes de Sensores em Larga Escala: proporciona controle e oferta de recursos de processamento, armazenamento e comunicação máquina-a-máquina;

4. Grande Número de Nós: ocorre devido à distribuição geográfica, implementando serviços executados pelos nós da Névoa como parte de uma aplicação de Nuvem distribuída;
5. Suporte à Computação Móvel: permite comunicação direta com dispositivos associados;
6. Interações em Tempo Real: necessário disponibilizar e gerenciar recursos locais para cumprir requisitos de tempo;
7. Predominância do Acesso Sem Fio: engloba dispositivos de IoT, permitindo uso de protocolos como Bluetooth, RFID, ZigBee, Wi-Fi, LTE, etc.;
8. Heterogeneidade: envolve gerenciamento de variados dispositivos de fabricantes, protocolos e linguagens de programação;
9. Interoperabilidade e Federação: os componentes de Névoa necessitam interagir entre diferentes domínios de rede;
10. Análise de Dados em Tempo Real: habilidade para processamento de dados provindos de *Big Data* os quais necessitam ocorrer em tempo real.

2.2 CENÁRIOS DE USO

Existem diversos usos em ambientes que podem tirar vantagem dessa arquitetura. De acordo com Coutinho, Carneiro & Greve (2016), algumas características são comuns entre esses ambientes como necessidade de análise em tempo real, podendo assim gerar ações sobre os resultados envolvendo comunicação Máquina-a-Máquina ou Homem-a-Máquina.

Coutinho, Carneiro & Greve (2016) descrevem os cenários mais comuns da utilização de Computação em Névoa. Esses cenários estão resumidos a seguir.

Wireless Sensor Networks (WSNs) podem ser espalhados em grandes áreas e possuem características de baixo uso de energia e de banda para comunicação, pouco processamento e memória. Geralmente, encaminham dados unidirecionalmente para *gateways* e possuem a capacidade de monitorar precipitação, umidade, temperatura, etc.. Essas redes possuem algumas limitações no que tange a funcionalidades mais complexas, além de rastreamentos e detecções. Para isso, são utilizados atuadores que realizam funções como movimentos de abertura e fechamento. Com a adição desses aparelhos, os dados passam a serem trafegados bidirecionalmente, o que compromete a estabilidade dos sistemas causando maior latência e *jitter*. A Arquitetura em Névoa pode contribuir amplamente para esse cenário já que possui soluções que são implementadas próximas aos utilizadores (no caso os sensores) e é capaz de lidar com o alto fluxo de dados gerados por esses.

A análise de dados é outro cenário onde a Computação em Névoa é adequada. A análise é iniciada nos próprios *gateways* que recebem os dados dos dispositivos de IoT. De forma geral,

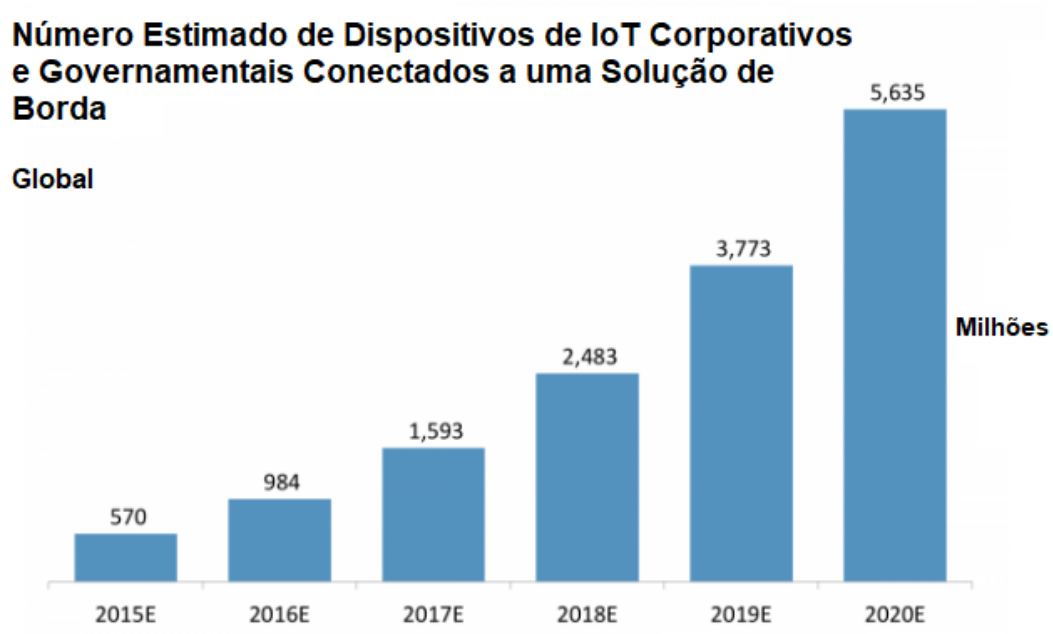
é obtida vantagem do uso da arquitetura no que tange à baixa latência, o que permite processar os dados em tempo real.

Na função de *cache* de dados, servidores na borda de rede permitem criar sistemas de *cache* para reduzir acesso e aprimorar velocidade de *download* de arquivos. Para isso, podem ser utilizadas redes *Content Delivery Networks (CDNs)* ou *Information Centric Networks (ICNs)*. Sistemas de previsão do uso de dados podem ser implementados em conjunto, os quais preveem o aumento da demanda e adicionam conteúdos específicos em *cache*.

Na área da saúde, a gerência de dados exige que ações imediatas sejam tomadas a partir de análises prévias, de forma que a vida dos pacientes possa depender disso. Sendo assim, os dispositivos de *IoT* auxiliam nas decisões e no monitoramento dos pacientes. Os autores também citam que a adoção da Névoa se faz crucial no que diz respeito à privacidade dos dados médicos, sendo assim, a sua utilização como camada intermediária para promover privacidade e flexibilidade é ideal.

Conforme a Figura 6, dados globais da utilização de arquiteturas nesse modelo indicam que o uso da tecnologia em Nuvem e Névoa é uma tendência para o mercado e vem crescendo vertiginosamente nos últimos anos.

Figura 6 – Adoção de *IoT* Integradas à Solução em Névoa na Saúde

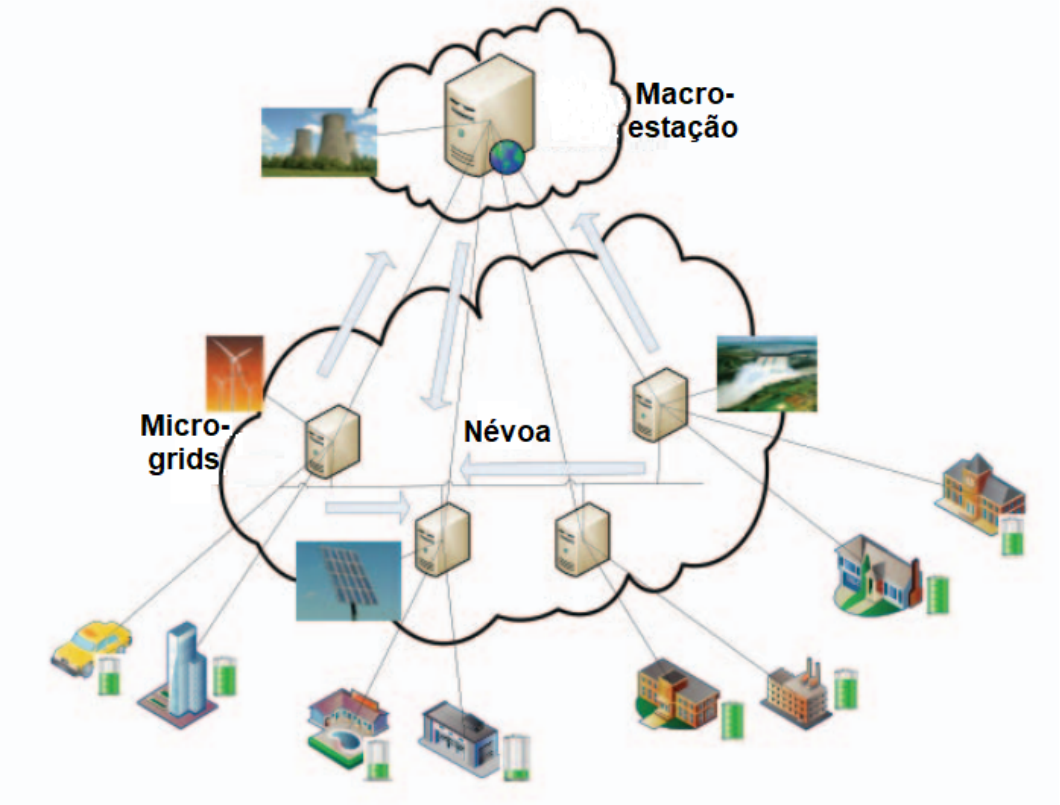


Fonte: Adaptado de https://healthitanalytics.com/features/how-fog-computing-may-power-the-healthcare-internet-of-things Acesso em: 22 agosto 2021.

No contexto de segurança, sistemas em IoT usualmente exigem altos níveis de privacidade e integridade. Dados que trafegam longas distâncias são mais vulneráveis a ataques. Por isso, a arquitetura em questão é viável já que garante menos saltos entre clientes e servidores. Além disso, considerando a característica de distribuição dos nós de Névoa, ataques de negação de serviço tornam-se menos viáveis já que envolvem um ataque em massa para todos os dispositivos próximos do cliente, exigindo mais recursos por parte de quem ataca.

Redes inteligentes de energia consistem em dispositivos como medidores de energia interconectados através de uma rede de dados. Dessa maneira, é possível balancear a carga de energia permitindo, portanto, controlar o consumo e alternar entre os tipos fornecidos para garantir maior disponibilidade dos recursos e menor preço para os usuários. A arquitetura dessas redes é exemplificada na Figura 7.

Figura 7 – Computação em Névoa para Redes Inteligentes de Energia



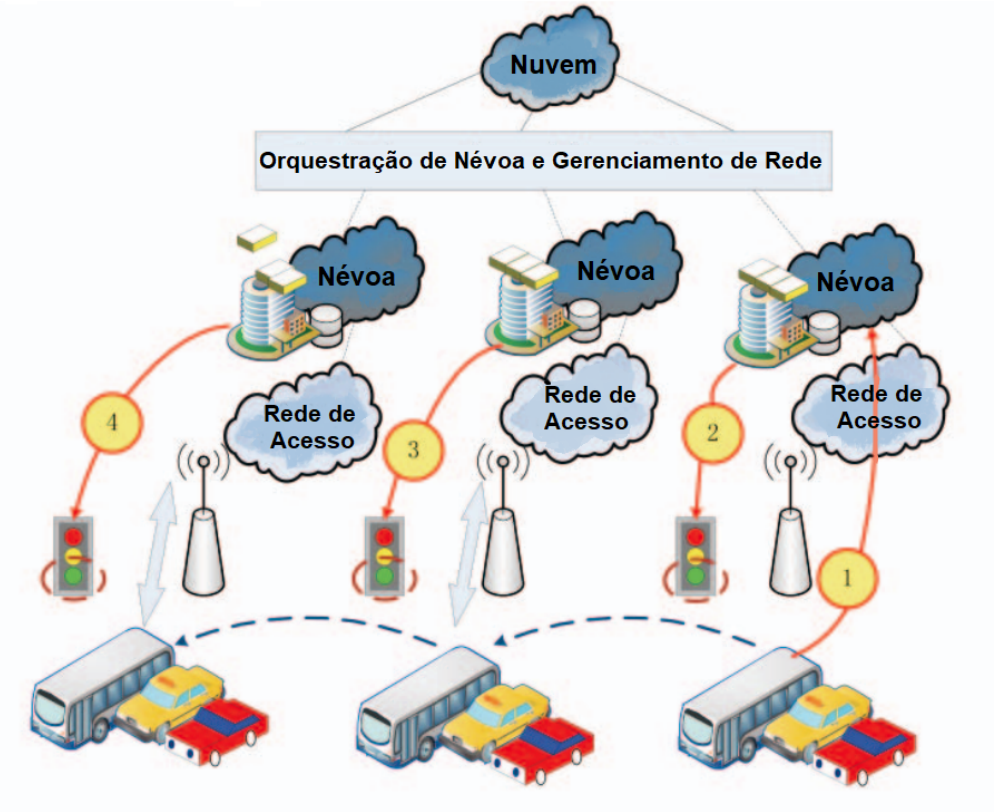
Fonte: Adaptado de Coutinho, Carneiro & Greve (2016).

Algoritmos para gerenciamento dessas redes estão sendo implementados em arquitetura de Nuvem, gerando alto consumo de banda. Por isso, soluções em Névoa devem ser eficazes através do uso de equipamentos denominados macro-*grids* e micro-*grids* (equivalentes aos dispositivos de Névoa os quais proporcionam diminuir a sobrecarga de comunicação).

No que tange às aplicabilidades no trânsito, há redes que proporcionam integração entre veículos, semáforos, postes inteligentes de iluminação, sensores, etc.. Como exemplo, tem-se sinaleiras que priorizam a passagem de ambulâncias para que estas se desloquem mais rapidamente. Outro uso são semáforos inteligentes, que analisam o trajeto dos veículos, possibilitando assim diminuir a emissão de poluentes.

Os autores explicitam que a comunicação em redes veiculares é feita de forma descentralizada, como no caso de redes *ad hoc*. Já para a aplicação de redes em Névoa, é interessante o uso de SDNs, proporcionando contornar problemas das redes *ad hoc* que em sua maioria, apresentam alta taxa de perda e colisão de pacotes. Num *framework* descrito por eles, dispositivos como semáforos atuam como elementos de Névoa, assumindo a função de roteadores no contexto de SDNs. A arquitetura do modelo é exemplificada na Figura 8.

Figura 8 – Computação em Névoa para Redes de Veículos Conectados



Fonte: Adaptado de Coutinho, Carneiro & Greve (2016).

O contexto de casas, edifícios e cidades inteligentes leva conceitos de IoT para a casa dos usuários finais e é gradativamente mais comum já que diversos fabricantes estão ofertando novos dispositivos inteligentes voltados para o mercado pessoal. As aplicabilidades são as mais variadas como monitoramento de saúde, controle de temperatura, de umidade, das luzes, de água, de energia, etc..

Guevara, Torres & Fonseca (2020) separam as aplicações que usam a estrutura de Névoa em Classes de Serviço, de acordo com os tipos de requisições e seus requisitos de *QoS*: missão crítica, tempo real, interativa, conversacional, *streaming*, *CPU-Bound* e melhor esforço.

A classe de missão crítica consiste em um curto período de tempo entre a geração de evento e a tomada de ação. No geral, é aplicada em casos em que uma falha possa causar risco para pessoas e para o ambiente. Isso inclui a área médica, drones, controles industriais, transações financeiras, caixas eletrônicos, etc..

A classe de tempo real envolve a necessidade de alta velocidade de resposta já que os dados são processados no mesmo tempo em que são gerados. Como aspectos gerais, também é necessário que o volume de dados trafegados seja o menor possível, bem como suportar determinado nível de perda de dados. Como exemplos, pode-se citar jogos *online*, realidade virtual e realidade aumentada.

A classe interativa possui o tempo de requisição do usuário e o tempo de resposta menor do que poucos segundos. Algumas aplicações consistem em televisão interativa, navegação na Internet, acesso a servidores, requisição de informações de base de dados, etc..

A classe conversacional inclui serviços de *Voice Over IP (VoIP)*. Envolve portanto, características de ser sensitivo a *delay* mas tolerante a perdas, com atrasos de até 400 ms considerados aceitáveis.

A classe de *streaming* envolve o *download* de arquivos de fluxo contínuo (vídeos, etc.). Neste contexto, é permitido demorar até 10 segundos para iniciar a reprodução. Devido à necessidade de promover interatividade e fluxo contínuo, o fator mais importante é a capacidade de transmissão de dados em largura de banda, necessitando que a capacidade de transmitir seja maior que a de geração dos dados.

Já a classe *CPU-Bound*, envolve alto processamento de dados, o que pode demandar até meses para obter um resultado. Reconhecimento facial, renderização de animações e processamento de fala são exemplos da categoria.

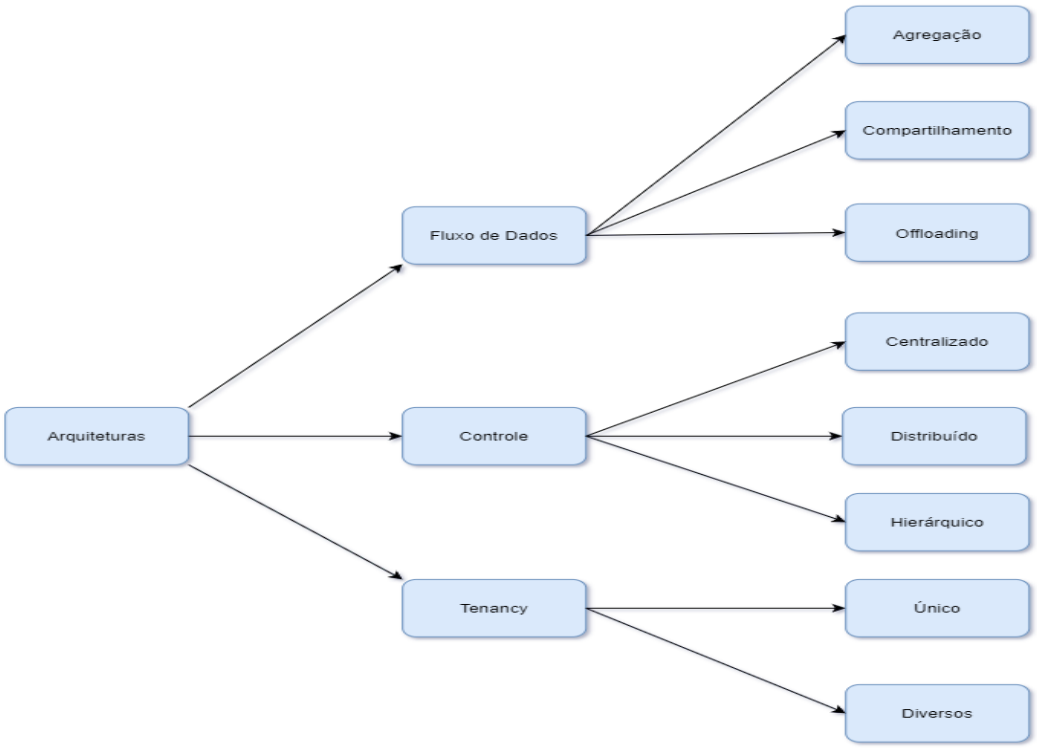
Por fim, a classe de melhor esforço é dedicada a aplicações diversas na Internet em que altos *delays* não são desejáveis mas em contrapartida não são uma ameaça de forma que o mais importante seja a transmissão completa e íntegra dos dados. Alguns exemplos são chats, *SMS*, *FTP*, etc..

2.3 TAXONOMIA

Hong & Varghese (2019) definiram uma taxonomia que categorizou a Computação em Névoa de acordo com o tipo de arquitetura, infraestrutura e algoritmos.

Na categoria arquitetura, os autores classificaram o gerenciamento de recursos com base no fluxo de dados, controle e arrendamento (Figura 9).

Figura 9 – Arquiteturas de Computação em Névoa



Fonte: Adaptado de Hong & Varghese (2019).

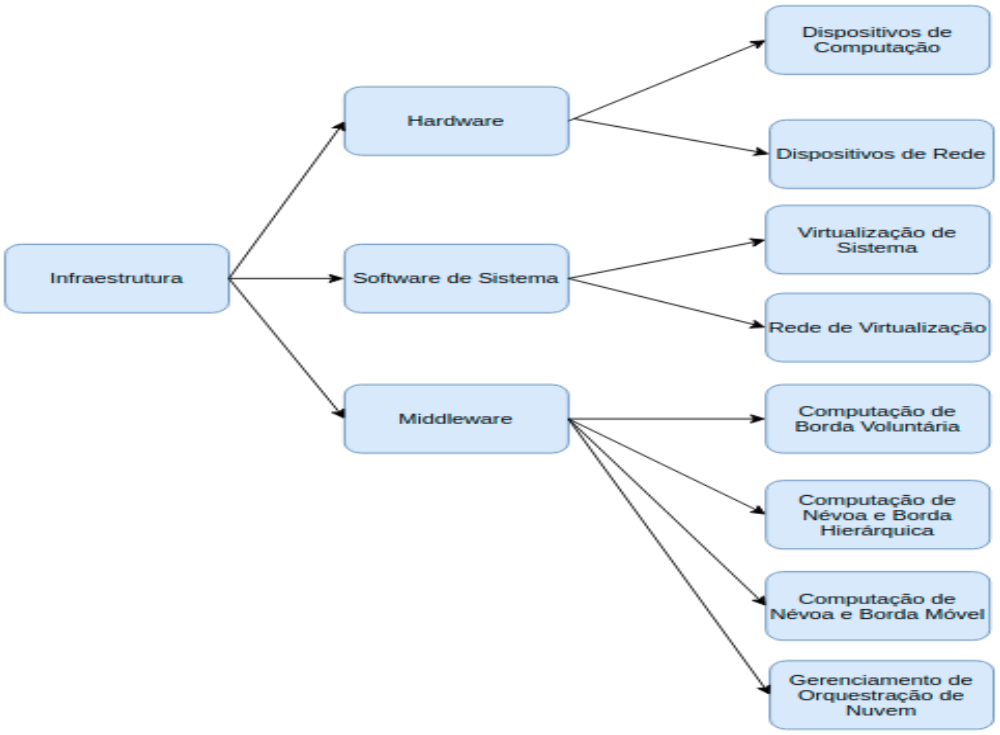
O fluxo de dados define a direção do movimento de cargas de trabalho, ou seja, se os dados são transferidos de dispositivos de usuários para nodos de borda ou de servidores de Nuvem para servidores de borda. O fluxo de dados pode ocorrer de forma agregada, compartilhada ou em *offloading*. O modo agregado possui como objetivo reduzir gargalo de comunicação já que permite evitar tráfego desnecessário além da borda de rede. No modo compartilhado, o processamento é dividido entre pares de forma a satisfazer a carga de trabalho em dispositivos móveis sem transferir para a Nuvem. Já no modelo de *offloading*, o servidor, a aplicação e os dados são movidos para a borda da rede.

A arquitetura de controle é outra maneira de classificação através do gerencimento de recursos em Névoa ou Nuvem. Suas divisões principais tangem a modalidade centralizada e a distribuída. A centralizada refere-se ao uso de um único controlador que gerencia os recursos computacionais. Por sua vez, a distribuída envolve a decisão de forma conjunta por vários nodos. Por fim, a arquitetura *Tenancy* refere-se ao compartilhamento de hardware entre entidades

com o intuito de otimizar a utilização de recursos e também para alcançar uma maior eficiência energética.

A infraestrutura para a Computação em Névoa (Figura 10) é dividida em *Hardware*, *Software* de Sistema e *Middleware*, sendo resumida nos parágrafos subsequentes.

Figura 10 – Infraestruturas de Computação em Névoa



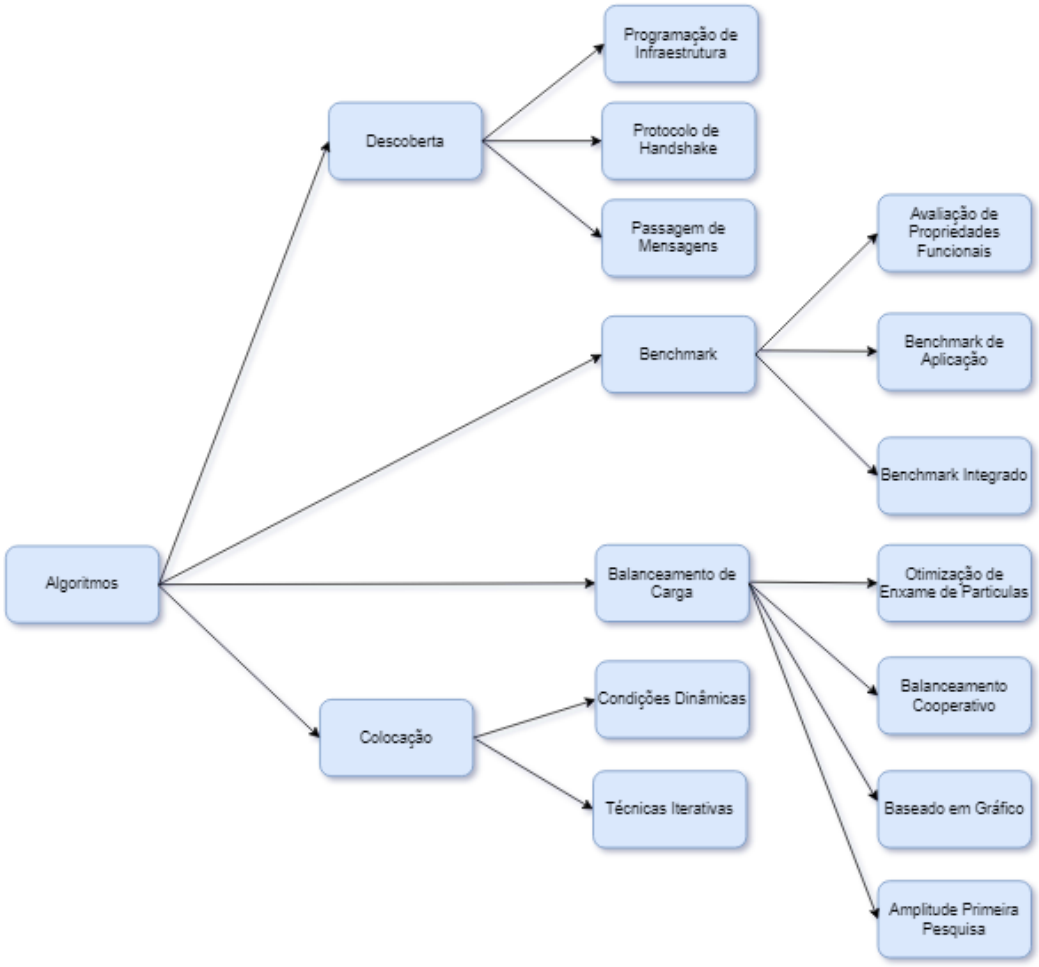
Fonte: Adaptado de Hong & Varghese (2019).

O Hardware para aplicações em Névoa são baseados em dispositivos como *Gateways*, *Access Points* de *Wi-Fi*, carros e até drones. Outras máquinas como *laptops* e *smartphones* podem também ser utilizados. As infraestruturas do tipo Software de Sistema são utilizadas para gerenciar recursos e distribuí-los para as aplicações de Névoa. Exemplos são sistemas operacionais e softwares para virtualização. Já os *Middlewares* executam nos sistemas operacionais e provêm serviços não oferecidos pelos Softwares de Sistema. Além disso, coordenam nodos de computação distribuídos além de realizar a implantação de máquinas virtuais ou containers.

Alguns tipos de algoritmos (Descoberta, Benchmarking, Balanceamento de Carga e Colocação) são utilizados para facilitar o uso de ambientes de Névoa (Figura 11). Os algoritmos de Descoberta são utilizados para identificar recursos de Névoa de forma a permitir que aplicações possam ser direcionadas a eles. Algoritmos para Benchmarking são utilizados para verificar o desempenho de memória, *CPU*, rede, etc.. Métricas importantes de cada dispositivo necessitam ser obtidas usando metodologias padronizadas. Algumas ferramentas para esse propósito são o

LINPACK[1] e o *NAS Parallel Benchmarks*[2]. Já os de balanceamento de carga implementam algoritmos para distribuir tarefas. São de suma importância para garantir uso eficiente dos recursos computacionais. Por fim, os algoritmos de colocação atuam na necessidade de alocar tarefas em recursos de Névoa apropriados. Para isso, necessitam considerar e avaliar os recursos de cada camada e alterações no ambiente de trabalho.

Figura 11 – Algoritmos para Computação em Névoa

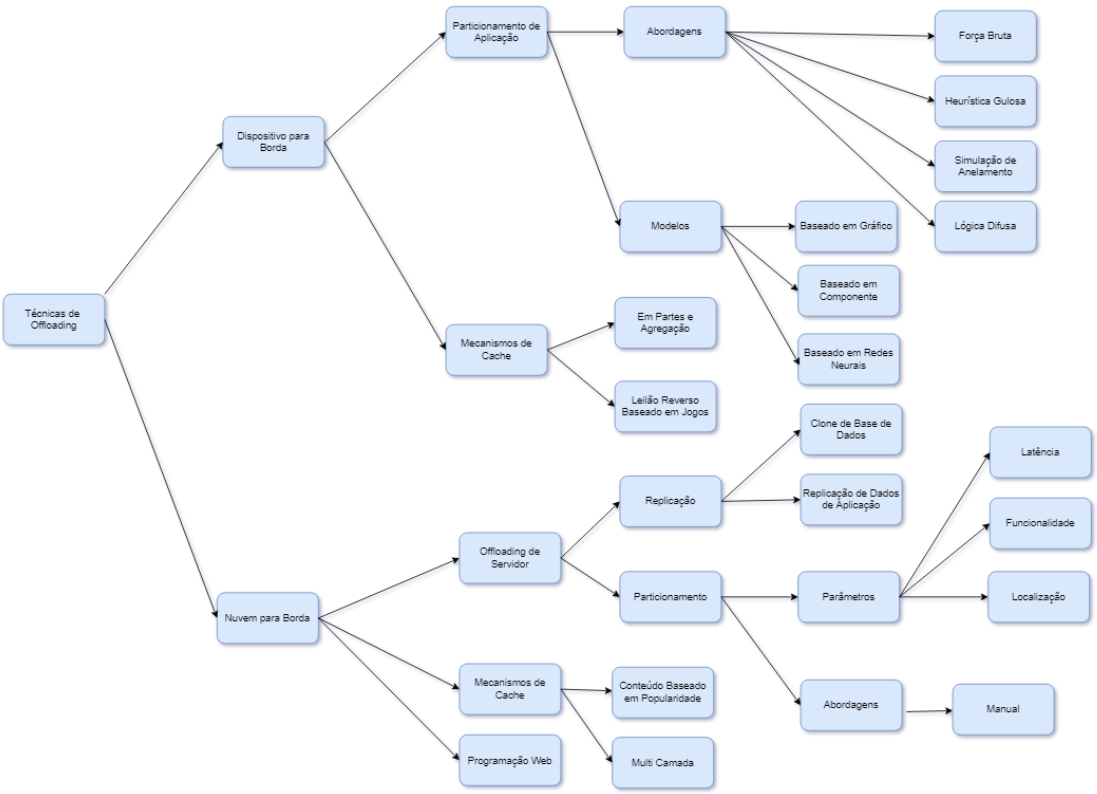


Fonte: Adaptado de Hong & Varghese (2019).

Técnicas de *offloading* (Figura 12) também são apresentadas por Hong & Varghese (2019). Consistem em mover uma aplicação e seus dados para a borda de rede, sendo divididas entre dispositivo de usuário para borda e de Nuvem para borda.

[1] https://www.top500.org/project/linpack/
[2] https://www.nas.nasa.gov/software/npb.html

Figura 12 – Técnicas de *Offloading*



Fonte: Adaptado de Hong & Varghese (2019).

O primeiro grupo de técnicas (dispositivo de usuário para borda) faz uso de nodos de borda geralmente a um salto de distância e são divididos entre particionamento de aplicações e mecanismos de *Cache* que consiste em um *cache* global na borda de rede que funciona como uma memória compartilhada para dispositivos que apresentam interações entre si. Já o segundo grupo de técnicas (de Nuvem para borda) é oposta à anterior e traz o processamento para mais perto do usuário final. Essa técnica é dividida em três:

1. *Server Offloading*: servidor que roda na nuvem é trazido para a borda através de replicação ou particionamento;
2. Mecanismos de *cache*: pretendem diminuir o uso de rede;
3. Programação WEB: execução de *scripts* em borda de rede para atender solicitações de usuários.

2.4 REQUISITOS PARA GERENCIAMENTO

Monitorar redes e sistemas é um ato crucial para identificar mudanças e tomar ações de melhoria de performance, além de proporcionar serviços como *QoS* e *Quality of Experience*

(QoE) (ABREHA, 2020). Além disso, o autor destaca que é importante para garantir o controle de hardware e software entre provedores e consumidores.

Para a realização deste trabalho, foi adotado o processo de revisão sistemática, tendo como objetivo, reunir materiais semelhantes de diversos autores e realizar uma análise estatística. A primeira etapa foi definir um conjunto de palavras chave para realizar a pesquisa bibliográfica. As palavras chave escolhidas foram *"Fog Computing"*, *"Requirements"*, *"QoS"* e *"Management"*.

A busca dos materiais foi iniciada pela base da CAPES, o que permitiu obter 286 artigos relacionados. Após, foi realizada a leitura dos resumos. Assim, 269 artigos foram excluídos já que não corresponderam ao contexto do projeto. A análise então compreendeu 17 artigos. Após uma análise aprofundada desses trabalhos, verificou-se que os principais elementos para serem analisados são: a capacidade de gerenciamento proporcionada pelas ferramentas em ambiente de Névoa, o controle de QoS e a métrica de avaliação de resultados em ambiente real.

Essas informações foram extraídas dos textos, tabulados e apresentados no Quadro 2 ao Quadro 15.

Quadro 2 – Artigo 01

Título	Resource Management in Fog/Edge Computing: A Survey
Autor	Cheol-Ho Hong e Blesson Varghese
Ano Publicação	2019
Objetivo	Identificar e classificar as arquiteturas, a infraestrutura e os algoritmos para gerenciamento de recursos na Computação em Névoa
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Auto descoberta de dispositivos 2. Desempenho 3. Balanceamento de carga 4. Colocação
Ferramentas de Gerenciamento Mencionadas	1. EyeQ framework 2. Nebula Monitor 3. Cloudy software 4. Foggy

Fonte: Autoria própria (2022).

Quadro 3 – Artigo 02

Título	Self-Adaptive Monitoring in Fog Computing by Leveraging Machine Learning
Autor	Haftay Gebreslasie Abreha
Ano Publicação	2020
Objetivo	Analisar estruturas de Computação em Névoa, as soluções de monitoramento atualmente disponíveis, os requerimentos e desafios além de propor uma solução de monitoramento
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Escalabilidade e heterogeneidade 2. Adaptabilidade e dinamismo 3. Robustez 4. Não-intrusividade 5. Interoperabilidade e federação 6. Facilidade de migração 7. Elasticidade 8. Autonomicidade
Ferramentas de Gerenciamento Mencionadas	1. Ganglia 2. Nagios 3. MonALISA 4. Zabbix 5. PCMONS 6. OpenNebula 7. DARGOS 8. Lattice 9. JCatascopia

Fonte: Autoria própria (2022).

Quadro 4 – Artigo 03

Título	On the classification of fog computing applications: A machine learning perspective
Autor	Judy C. Guevara, Ricardo da S. Torres e Nelson L.S. da Fonseca
Ano Publicação	2020
Objetivo	Mapeamento dos requisitos de QoS e classificação de acordo com classes de serviços. Propõe a implementação de um sistema de *Machine Learning* para classificar as aplicações de Névoa em função dos requerimentos de QoS
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	Agendamento de tarefas, alocação e federação. Requisitos de QoS: 1. Largura de banda 2. Sensitividade a *delay* e perda 3. Confiabilidade 4. Disponibilidade 5. Segurança 6. Locação dos dados 7. Mobilidade 8. Escalabilidade
Ferramentas de Gerenciamento Mencionadas	1. LoM2Hi 2. CASViD

Fonte: Autoria própria (2022).

Quadro 5 – Artigo 04

Título	Influence of Monitoring: Fog and Edge Computing
Autor	Vivek Kumar Prasad, Madhuri Bhavsar e Sudeep Tanwar
Ano Publicação	2019
Objetivo	Discussão de técnicas de monitoramento para computação de Névoa e de Borda e suas vantagens. Demonstração da necessidade de monitoramento em contexto exemplificado na área da saúde
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Gerenciar grande número de dispositivos 2. Dinamicidade 3. Adaptabilidade 4. Confiabilidade 5. Robustez 6. Interoperabilidade 7. Não-intrusividade 8. Escalabilidade 9. Suporte para migração
Ferramentas de Gerenciamento Mencionadas	1. Tower 4Clouds 2. Jcatascopia 3. Lattice 4. DARGOS 5. PCMONS 6. OpenNebula 7. Nagios 8. Zabbix 9. Ganglia 10. Zenoss

Fonte: Autoria própria (2022).

Quadro 6 – Artigo 05

Título	An Efficient Resource Monitoring Service for Fog Computing Environments
Autor	Sudheer Kumar Battula, Saurabh Garg, James Montgomery e Byeong Kang
Ano Publicação	2020
Objetivo	Investigação de técnicas de monitoramento e análise da viabilidade para ambientes de Névoa
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Robustez 2. Prover informações acuradas do ambiente 3. Garantia de QoS e acordo de nível de serviço 4. Gerenciamento de falhas 5. Custo benefício
Ferramentas de Gerenciamento Mencionadas	Não há menções

Fonte: Autoria própria (2022).

Quadro 7 – Artigo 06

Título	IoT/cloud-enabled smart services: A review on QoS requirements in fog environment and a proposed approach based on priority classification technique
Autor	Amel Ksentini, Maha Jebalia e Sami Tabbaner
Ano Publicação	2019
Objetivo	Análise dos requerimentos de QoS em ambientes de Névoa para identificação de métricas. Proposta de soluções para melhorias com base na técnica de Classificação de Prioridade
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Largura de Banda 2. Latência 3. Confiabilidade 4. Uso de localização 5. Processamento em tempo real 6. *Cache* 7. Mobilidade 8. Conectividade 9. Consumo de energia 10. Custo 11. Escalabilidade 12. Avaliabilidade
Ferramentas de Gerenciamento Mencionadas	Framework Proposto por Aazam *et al.* (2016b)

Fonte: Autoria própria (2022).

Quadro 8 – Artigo 07

Título	Quality of service-aware approaches in fog computing
Autor	Mostafa Haghi Kashani, Amir Masoud Rahman e Nima Jafari Navimipour
Ano Publicação	2019
Objetivo	Investigação de técnicas para garantir QoS para Névoa, dividido em três categorias: *service/resource management*, *communication management* e *application management*. Apresenta vantagens, desvantagens, ferramentas e modos de avaliação para QoS
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Taxa de transferência 2. Prazo final 3. Tempo de resposta 4. Utilização de recursos 5. Custo 6. Tempo de execução 7. Consumo de energia 8. Confiabilidade 9. Avaliabilidade 10. Escalabilidade 11. Segurança
Ferramentas de Gerenciamento Mencionadas	Frameworks, ferramentas e arquiteturas propostas por: 1. Cardellini *et al.* (2015b) 2. Shojafar, Cordeschi & Baccarelli (2019) 3. Ni *et al.* (2017) 4. Wang *et al.* (2020) 5. Gu *et al.* (2017) 6. Liu *et al.* (2018) 7. Yang *et al.* (2018b) 8. Xiao & Krunz (2018) 9. Assila *et al.* (2018) 10. Pham *et al.* (2017) 11. Bitam, Zeadally & Mellouk (2018) 12. Kafhali & Salah (2017) 13. Skarlat *et al.* (2017a) 14. He *et al.* (2018) 15. Jutila (2016)

Fonte: Autoria própria (2022).

Quadro 9 – Artigo 08

Título	A comprehensive study on managing strategies in the fog environments
Autor	Aiqun Wang, Panpan Yan e Khaldoun Batiha
Ano Publicação	2019
Objetivo	Pesquisa sobre o estado da arte em gerenciamento de Névoa e classificação em categorias. Apresenta sugestões para futuros estudos envolvendo o tema
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Gerenciamento 2. Acessibilidade 3. Segurança 4. Privacidade 5. Dinamicidade 6. Tempo de execução 7. Custo 8. Flexibilidade 9. Escalabilidade 10. *Performance* 11. Complexidade 12. Eficiência 13. Avaliabilidade 14. Tempo real 15. Latência
Ferramentas de Gerenciamento Mencionadas	Não há menções

Fonte: Autoria própria (2022).

Quadro 10 – Artigo 09

Título	Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions
Autor	Mir Salim Ul Islam, Ashok Kumar, Yu-Chen Hu
Ano Publicação	2021
Objetivo	Análise sistemática sobre *context-aware scheduling* em Névoa. Classificação do contexto em categorias: de usuário, de aplicação, de ambiente, de rede e de dispositivo
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Latência 2. Custo 3. Uso de energia 4. Escalabilidade
Ferramentas de Gerenciamento Mencionadas	Frameworks, ferramentas, arquiteturas e algoritmos propostos por: 1. Aazam & Huh (2015) 2. Aazam *et al.* (2016a) 3. Naha *et al.* (2020) 4. Dsouza, Ahn & Taguinod (2014) 5. Cirani *et al.* (2015) 6. Cardellini *et al.* (2015b) 7. Gazis *et al.* (2015) 8. Mahmud *et al.* (2016) 9. Minh *et al.* (2017) 10. Sharif, Mercelis & Hellinckx (2018) 11. Mononen, Aref & Mattila (2018)

Fonte: Autoria própria (2022).

Quadro 11 – Artigo 10

Título	Workload aware autonomic resource management scheme using grey wolf optimization in cloud environment
Autor	Bhupesh Kumar Dewangan, Amit Agarwal, Tanupriya Choudhury e Ashutosh Pasricha
Ano Publicação	2021
Objetivo	Propõe framework WARMS para o ambiente de Nuvem focando em reduzir: violação de acordo de nível de serviço, custo, gasto energético e tempo além de prover QoS aprimorado
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Avaliabilidade 2. Confiabilidade 3. Utilização de recursos 4. Consumo de energia 5. Tempo de execução 6. Taxa de violação de acordo de nível de serviço 7. Custo de acordo de nível de serviço
Ferramentas de Gerenciamento Mencionadas	1. Framework WARMS Frameworks, ferramentas, arquiteturas e algoritmos propostos por: 1. Suresh & Varatharajan (2019) 2. Ghobaei-Arani, Jabbehdari & Pourmina (2018) 3. Marinescu *et al.* (2017) 4. Sotiriadis, Bessis & Buyya (2018) 5. Zuo *et al.* (2017) 6. Zahoor *et al.* (2018) 7. Xiong *et al.* (2018) 8. Gai *et al.* (2017)

Fonte: Autoria própria (2022).

Quadro 12 – Artigo 11

Título	Deep Learning at the Mobile Edge: Opportunities for 5G Networks
Autor	Miranda McClellan, Cristina Cervelló-Pastor e Sebastià Sallent
Ano Publicação	2020
Objetivo	Discute o estado da arte em *machine learning* para computação de borda em redes móveis, além dos avanços necessários em automação, alocação de recursos adaptativas, segurança e eficiência energética para redes 5G
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Baixa latência 2. Alta disponibilidade 3. Escalabilidade 4. Automação 5. Tempo real 6. Segurança 7. Eficiência energética 8. Largura de banda 9. Taxa de erros
Ferramentas de Gerenciamento Mencionadas	ETSI MANO framework

Fonte: Autoria própria (2022).

Quadro 13 – Artigo 12

Título	COSCO: Container Orchestration Using Co-Simulation and Gradient Based Optimization for Fog Computing Environments
Autor	Shreshth Tuli, Shivananda R. Poojara, Satish N. Srirama, Giuliano Casale e Nicholas R. Jennings
Ano Publicação	2021
Objetivo	Propõe estratégia de otimização baseada em gradiente utilizando *back-propagation* para alcançar políticas de agendamento otimizadas para ambientes voláteis. Criação da política GOBI* para otimizar parâmetros de QoS
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Tempo de resposta 2. Consumo de energia 3. Violação de objetivo de nível de serviço 4. Baixo tempo de agendamento
Ferramentas de Gerenciamento Mencionadas	Frameworks: 1. COSCO Framework Políticas de agendamento: 1. GOBI 2. GOBI* 3. A3C 4. GA 5. DQLCM 6. POND 7. LR-MMT 8. MAD-MC

Fonte: Autoria própria (2022).

Quadro 14 – Artigo 13

Título	Internet of Things offloading: Ongoing issues, opportunities, and future challenges
Autor	Arash Heidari, Mohammad Ali Jabraeil Jamali, Nima Jafari Navimipour e Shahin Akbarpour
Ano Publicação	2020
Objetivo	Apresentação de uma taxonomia baseada em *offloading* e mecanismos de decisão para garantir QoS. Comparação de métodos de *offloading* existentes
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Baixa latência 2. Balanceamento de carga 3. Segurança 4. Acessibilidade 5. Consumo de banda 6. Confiabilidade
Ferramentas de Gerenciamento Mencionadas	Métodos de *offloading* propostos por: 1. Yu *et al.* (2018) 2. Wang *et al.* (2019) 3. Bhattacharya & De (2017) 4. Xu *et al.* (2018) 5. Wu (2018) 6. Zhou *et al.* (2018) 7. Shukla & Munir (2016) 8. Happ & Wolisz (2017) 9. Yang *et al.* (2018a) 10. Flores *et al.* (2017) 11. Duan *et al.* (2017) 12. Dai *et al.* (2018) 13. Elazhary & Sabbeh (2018) 14. Samie *et al.* (2016) 15. Al-Zihad *et al.* (2017) 16. Alam *et al.* (2019) 17. Lyu *et al.* (2018) 18. Yousefpour *et al.* (2018) 19. Kim (2015) 20. Min *et al.* (2019) 21. Guan *et al.* (2017) 22. Park & Kim (2015) 23. Chen *et al.* (2017), Saqib & Hamid (2016)

Fonte: Autoria própria (2022).

Quadro 15 – Artigo 14

Título	An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing
Autor	Mostafa Ghobaei-Arani, Alireza Souri, Fatemeh Safara e Monire Norouzi
Ano Publicação	2019
Objetivo	Propõe um método de agendamento utilizando algoritmos do tipo MFO para atingir um estado ótimo do gerenciamento de tarefas para satisfazer requisitos de QoS
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Baixa latência 2. Custo 3. Tempo 4. Avaliabilidade 5. Taxa de transferência de dados
Ferramentas de Gerenciamento Mencionadas	Tipos de algoritomos de agendamento: 1. *Bees Life Algorithm* 2. *New Genetic Algorithm* 3. *Particle Swarm Optimization* 4. MFO (base para agendador de tarefas TS-MFO) Algoritmos e ferramentas dos autores: 1. Cheng *et al.* (2017) 2. Bitam, Zeadally & Mellouk (2018) 3. A.M & Venkatesan (2019) 4. Verma, Bhardwaj & Yadav (2016) 5. Panwar *et al.* (2019) 6. Chen, Walters & Crago (2017) 7. Su *et al.* (2015)

Fonte: Autoria própria (2022).

Quadro 16 – Artigo 15

Título	How to place your apps in the fog: State of the art and open challenges
Autor	Antonio Brogi, Stefano Forti, Carlos Guerrero e Isaac Lera
Ano Publicação	2019
Objetivo	Revisão de soluções propostas para a colocação de aplicações em Névoa tendo como base a análise de algoritmos e modelos
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Latência 2. Largura de banda 3. Avaliabilidade 4. Confiabilidade 5. Gasto energético 6. Carga de trabalho 7. Tempo de execução
Ferramentas de Gerenciamento Mencionadas	Frameworks, ferramentas e algoritmos de colocação e migração: 1. Gupta *et al.* (2016) 2. Mahmud *et al.* (2019) 3. Guerrero, Lera & Juiz (2019) 4. Ottenwälder *et al.* (2013) 5. Brogi & Forti (2017) 6. Wang, Zafer & Leung (2017) 7. Hong, Tsai & Hsu (2016) 8. Taneja & Davy (2017) 9. Velasquez *et al.* (2016) 10. Filiposka, Mishev & Gilly (2018) Políticas de gerenciamento: 1. Mahmud *et al.* (2019) 2. Deng *et al.* (2015) Gerenciar e estimar garantia de QoS, consumo e gerenciamento de recursos: 1. Brogi & Forti (2017) 2. Chowdhury, Rahman & Boutaba (2012) 3. Yang *et al.* (2015) 4. Gu *et al.* (2017) 5. Zeng *et al.* (2016) 6. Souza *et al.* (2016) 7. Barceló *et al.* (2016) 8. Skarlat *et al.* (2017b), Cardellini *et al.* (2015a) 9. Venticinque & Amato (2019), Zhang *et al.* (2017)

Fonte: Autoria própria (2022).

Quadro 17 – Artigo 16

Título	Task scheduling approaches in fog computing: A systematic review
Autor	Mohammad Reza Alizadeh, Vahid Khajehvand, Amir Masoud Rahmani e Ebrahim Akbari
Ano Publicação	2020
Objetivo	Análise de algoritmos de agendamento propostos por diversos pesquisadores para o ambiente de Névoa e Nuvem. Apresenta vantagens e desvantagens, ferramentas e questões relacionadas aos métodos de agendamento
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Avaliabilidade 2. Tempo de resposta 3. Consumo energético 4. Confiabilidade 5. Segurança 6. Taxa de transferência de dados 7. Atraso 8. Custo e utilização de recursos 9. Tempo de execução 10. Acordo de nível de serviço 11. Largura de banda 12. Desempenho
Ferramentas de Gerenciamento Mencionadas	Algoritmos, estratégias e arquiteturas de agendamento propostas por: 1. Kaid & Othman (2013), Hoang & Dang (2017) 2. Isard *et al.* (2009), Choudhari, Moh & Moh (2018) 3. Rasheed *et al.* (2019), Kabirzadeh, Rahbari & Nickray (2017) 4. Rahbari, Kabirzadeh & Nickray (2017), Yin, Luo & Luo (2018) 5. Liu *et al.* (2017), Rahbari & Nickray (2017) 6. Wang, Guo & Yu (2018), Bitam, Zeadally & Mellouk (2018) 7. Deng *et al.* (2016), Oueis, Strinati & Barbarossa (2015) 8. Intharawijitr, Iida & Koga (2016), Pham & Huh (2016) 9. Bittencourt *et al.* (2017), Li *et al.* (2019) 10. Wang & Li (2019), Nguyen *et al.* (2019) 11. Jamil *et al.* (2019), Zhao *et al.* (2018), Abdelmoneem, Benslimane & Shaaban (2020) 12. Wan *et al.* (2018), Yang *et al.* (2018b), Ahmad, Patra & Barik (2020) 13. Jie *et al.* (2018), Ningning *et al.* (2016), Sun, Lin & Xu (2018) 14. Cardellini *et al.* (2015b), Zeng *et al.* (2016), Gazori, Rahbari & Nickray (2020) 15. Verma, Bhardwaj & Yadav (2016), Sharma & Saini (2019) 16. Gad-Elrab & Noaman (2020), Mastoi *et al.* (2020)

Fonte: Autoria própria (2022).

Quadro 18 – Artigo 17

Título	A Review of Cognitive Radio Smart Grid Communication Infrastructure Systems
Autor	Daisy Nkele Molokomme, Chabalala S. Chabalala e Pitshou N. Bokoro
Ano Publicação	2020
Objetivo	Investigar tecnologias para comunicação confiável para ambientes 5G em redes do tipo SG. Combina recursos de Nuvem e Névoa para prover melhorias de latência e segurança
Requisitos: O que precisa ser gerenciado a nível de recursos e de parâmetros de QoS	1. Latência 2. Largura de banda 3. Congestionamento de dados 4. Eficiência energética 5. Segurança 6. Confiabilidade
Ferramentas de Gerenciamento Mencionadas	Propõe arquitetura híbrida baseada em Nuvem e Névoa em ambiente de 5G e CRN's para aprimorar confiabilidade e eficiência energética

Fonte: Autoria própria (2022).

Com base nos artigos analisados, percebe-se que várias ferramentas diferentes e diversos requisitos são citados nos trabalhos realizados pelos autores. Para a escolha dos requisitos, devido à grande quantidade, a escolha possui como base a divisão em categorias com base em semelhança, para posterior análise de maneira agrupada. Em relação às ferramentas, não é observada grande quantidade de menções de uma mesma ferramenta além de várias delas já serem utilizadas para gerenciamento e monitoramento de redes domésticas e empresariais.

2.5 CONSIDERAÇÕES FINAIS

Assim como mencionado por Coutinho, Carneiro & Greve (2016), e com base na pesquisa realizada, constata-se que a Computação em Névoa é formada por conjunto de tecnologias diferentes os quais são integrados para satisfazer novas necessidades dos usuários que, devido à natureza de seu uso, requerem agilidade das redes para lidar com a grande quantidade de dados gerados diariamente, além da privacidade dos mesmos.

Levar o processamento para mais perto do usuário final, contribui para lidar com o volume das informações já que requisições que eram atendidas por servidores muitas vezes distantes e sobrecarregados, são servidas de forma mais ágil e consistente. Dessa maneira, o requisito de segurança também torna-se mais gerenciável já que grande parte dos dados acabam ficando nas redes internas das corporações.

Novas maneiras de cooperação e competição entre os provedores de Internet devem ser possíveis através da adoção da Névoa. Além disso, as organizações as quais fizerem seu uso de forma correta, terão um aumento na agilidade dos negócios, atingindo novos níveis de serviços e de segurança.

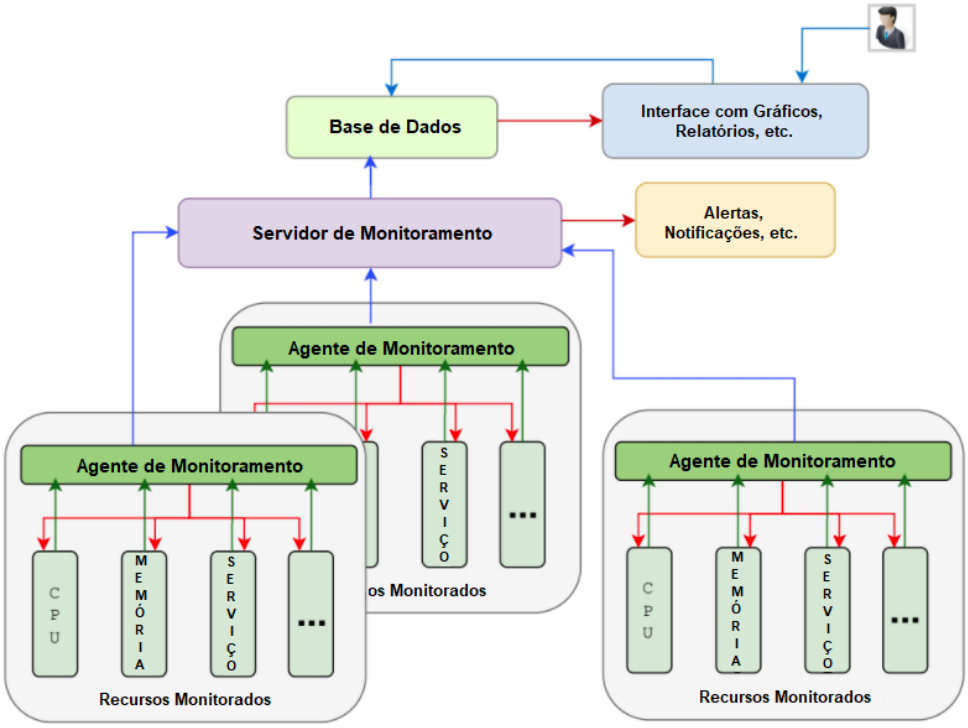
A construção dos quadros da Seção 2.4 permitiu analisar o trabalho de diversos autores no que tange à ferramentas, arquiteturas e algoritmos de agendamento, monitoramento e gerenciamento. Em relação à garantia de *QoS*, diversos parâmetros foram elencados, dentre eles, confiabilidade, custo, escalabilidade e segurança foram os mais mencionados.

Diante disso, as próximas seções irão explorar e exemplificar o uso de ferramentas de monitoramento de rede para ambientes de Névoa, com o intuito de encontrar uma solução comercial que se adeque às necessidades da arquitetura.

3 PROPOSTA DE SOLUÇÃO

Estruturas de monitoramento, no geral, utilizam modelo cliente-servidor com agentes em cada recurso a ser monitorado. Esses agentes possuem a função de disponibilizar métricas dos componentes e encaminhar os resultados para um servidor. Esse servidor deve armazenar os dados em banco de dados, analisá-los e enviar notificações ou alertas para o sistema de monitoramento. Além disso, pode gerar gráficos, relatórios, etc. Essa estrutura é ilustrada na Figura 13 (ABREHA, 2020).

Figura 13 – Estrutura Genérica de Monitoramento



Fonte: Adaptado de Abreha (2020).

A arquitetura atual de gerenciamento é descrita através de *Request for Comments (RFCs)* e possui quatro componentes básicos:

1. Gerentes: possuem o papel de coletar informações de agentes através de protocolo de gerenciamento;
2. Agentes: respondem às solicitações dos gerentes informando a ocorrência de eventos;
3. *Simple Network Management Protocol (SNMP)*: protocolo para definir um padrão de comunicação entre gerentes e agentes;

4. *Management Information Base (MIB)*: base de dados para armazenamento de informações de gerenciamento.

A arquitetura é formada por estações de gerenciamento de rede e elementos de rede. As estações controlam e monitoram os elementos por aplicações de gerenciamento. Já os elementos da rede (*gateways*, *hosts*, etc.) possuem agentes que respondem às solicitações das estações. Esses agentes coletam informações do sistema onde estão instalados e encaminham a informação ao gerente quando requisitados ou quando um evento específico acontece.

Para a troca de mensagens em uma arquitetura de monitoramento é adotado o protocolo SNMP o qual utiliza o protocolo de transporte *User Datagram Protocol (UDP)*. Esse protocolo suporta as operações:

1. *Get*: recupera uma informação específica de gerência;
2. *Get-Next*: recupera informações de gerência de objetos (tabelas, etc.);
3. *Set*: manipula informações de gerência;
4. *Trap*: reporta eventos extraordinários.

A relação entre gerentes e agentes utiliza o conceito de comunidade de forma com que quando um agente aceita comandos de um gerente ou reporta *traps* a um gerente, é dito que pertencem à mesma comunidade.

Esse capítulo apresenta como a estrutura de gerenciamento de redes em Névoa foi montada para o teste das ferramentas. A Seção 3.1 apresenta o ambiente de teste utilizado. A Seção 3.2 descreve a seleção das ferramentas de gerenciamento. A Seção 3.3 apresenta os critérios analisados nos testes. A Seção 3.4 especifica como as ferramentas foram avaliadas com base em normas ISO/IEC. A Seção 3.5 descreve os casos de testes. Por fim, a Seção 3.6 realiza as considerações finais do capítulo.

3.1 AMBIENTE DE TESTE

Essa seção visa descrever o cenário que foi utilizado para a realização dos testes juntamente com o hardware e software em uso. Para a implementação de um ambiente de Névoa real, é necessária a adoção de dispositivos específicos como roteadores, *gateways* e *switches* inteligentes os quais provêm serviços essenciais para esse tipo de ambiente computacional.

Devido à indisponibilidade desses equipamentos, para simular um ambiente de Névoa, o cenário de teste foi composto de dois computadores com sistema operacional Linux Ubuntu conectados a um ponto de acesso Wi-Fi que por sua vez, se comunica com um servidor na mesma rede local. Esses computadores funcionam como dispositivos locais inteligentes já que

são capazes de disparar dados ao servidor em alta velocidade, atuando portanto na camada 1 da arquitetura de Névoa. Já o servidor possui a função de executar a aplicação servidora a qual trabalha sobre os dados recebidos dos nodos além de hospedar as ferramentas de gerenciamento (Figura 14). Sendo assim, é capaz de atuar como um dispositivo de Névoa já que processa dados em tempo real e realiza a filtragem e o controle do envio desses dados para o servidor externo (Nuvem). O Quadro 19 ao Quadro 22 descrevem os principais componentes de hardware e software das máquinas utilizadas como nodos e servidores (da Névoa e da Nuvem).

Quadro 19 – Componentes de Hardware e Software do Servidor de Névoa (Máquina Virtual Virtual Box)

Componente	**Descrição**
CPU	AMD FX8300 4 núcleos físicos (2 núcleos dedicados para a máquina virtual) 3.3 GHz
RAM	16 GB (8 GB dedicados para a máquina virtual)
Armazenamento	240 GB SSD (15 GB dedicados para a máquina virtual)
Sistema Operacional	Linux Ubuntu 20.04.4 LTS
Kernel Linux	5.13.0-40-generic

Fonte: Autoria própria (2022).

Quadro 20 – Componentes de Hardware e Software do Servidor de Nuvem (Google Cloud)

Componente	**Descrição**
CPU	Intel Xeon (modelo não especificado) 2 núcleos físicos 2.20 GHz
RAM	8 GB
Armazenamento	10 GB SSD
Sistema Operacional	Linux Debian 10 buster
Kernel Linux	x86_64 Linux 4.19.0-20-cloud-amd64

Fonte: Autoria própria (2022).

Quadro 21 – Componentes de Hardware e Software do Nodo 1

Componente	Descrição
CPU	Intel Pentium P6200 2 núcleos físicos 2.13 GHz
RAM	4 GB
Armazenamento	480 GB SSD
Sistema Operacional	Linux Ubuntu 20.04.4 LTS
Kernel Linux	5.13.0-39-generic

Fonte: Autoria própria (2022).

Quadro 22 – Componentes de Hardware e Software do Nodo 2

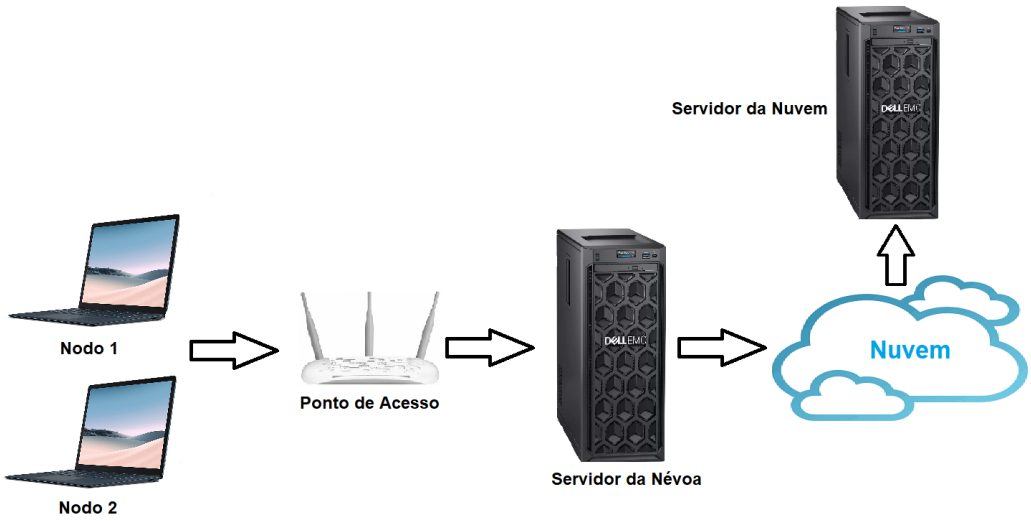
Componente	Descrição
CPU	Intel Core i3-6100U 2 núcleos físicos 2.30 GHz
RAM	8 GB
Armazenamento	1 TB
Sistema Operacional	Linux Ubuntu 20.04.4 LTS
Kernel Linux	5.13.0-41-generic

Fonte: Autoria própria (2022).

Como ponto de acesso, foi utilizado uma Raspberry Pi 3 modelo B+ com sistema operacional OpenWrt [1]. Esse sistema operacional é baseado em Linux, o que o torna compatível com diversos dispositivos de rede do mercado e oferece várias funcionalidades de gerenciamento e controle. Como exemplo, cita-se a possibilidade de utilização do protocolo SNMP bem como a instalação de agentes para diversas das aplicações de monitoramento existentes.

[1] https://openwrt.org/

Figura 14 – Arquitetura de Testes



Fonte: Autoria própria (2022).

Com intuito de gerar um fluxo de dados característico de arquiteturas de Névoa na rede, foi desenvolvida uma aplicação dividida em três componentes. O primeiro é executado nos computadores locais (nodos), coletando informação do uso atual de CPU e realizando posterior envio para o servidor local da Névoa. Não é adicionado *delay* entre o envio de pacotes de forma que tendo em vista que há poucos nodos na rede, esses devem encaminhar dados para o servidor na maior velocidade possível com o intuito de simular um tráfego de vários dispositivos simultâneos. A Figura 15 mostra sua execução a qual necessita que sejam informados como parâmetro: o IP do servidor da Névoa bem como a porta para que sejam enviados os dados obtidos.

O segundo componente localiza-se no servidor da Névoa e possui a função de receber os dados dos nodos, gerar uma média do uso de CPU por minuto e após, encaminhar para a Nuvem. A Figura 16 detalha sua execução, que exige que sejam informados como parâmetro: porta para receber dados dos nodos, IP do servidor da Nuvem e porta do servidor da Nuvem para encaminhamento das médias calculadas. Durante a sua execução, são exibidos *log's* que informam sobre o envio dos dados já processados para a Nuvem. A informação enviada é dividida em dois campos separados por ponto e vírgula: o IP do nodo na rede local da Névoa juntamente com a média de uso de CPU a cada um minuto.

Por fim, o módulo que é executado na Nuvem recebe os dados provenientes do servidor da Névoa e armazena-os em um banco de dados MySQL. A Figura 17 mostra sua execução, a qual exige que seja informada como parâmetro a porta para receber os dados provenientes da Névoa. Durante a execução da aplicação na Nuvem, são exibidas informações acerca dos valores recebidos (IP do nodo na Névoa e seu respectivo valor de uso médio de CPU calculado pelo servidor da Névoa).

Os três componentes da aplicação foram desenvolvidas em Python 3 utilizando sockets UDP. Para isso, foi utilizada a biblioteca *socket* da própria linguagem. Os dados trafegados são encapsulados a nível de camada de transporte em formato do tipo *bytes*.

Figura 15 – Aplicação Cliente

```
zabbix-agent@zabbixagent-VirtualBox:~/Downloads$ python3 client.py 192.168.15.6 8888
IP Servidor: 192.168.15.6 Porta Servidor: 8888
Cliente em execucao
```

Fonte: Autoria própria (2022).

Figura 16 – Aplicação Servidor Névoa

```
server@server-VirtualBox:~/Downloads$ python3 server.py 8888 192.168.15.14 9000
IP Cloud: 192.168.15.14 Porta Cloud: 9000
Servidor escutando na porta 8888
Enviando para nuvem dados nodo 1: 192.168.15.13;0.32
Enviando para nuvem dados nodo 1: 192.168.15.13;1.21
Enviando para nuvem dados nodo 1: 192.168.15.13;0.52
Enviando para nuvem dados nodo 1: 192.168.15.13;0.89
Enviando para nuvem dados nodo 1: 192.168.15.13;0.26
Enviando para nuvem dados nodo 1: 192.168.15.13;0.2
Enviando para nuvem dados nodo 1: 192.168.15.13;0.09
Enviando para nuvem dados nodo 1: 192.168.15.13;0.32
Enviando para nuvem dados nodo 1: 192.168.15.13;0.43
Enviando para nuvem dados nodo 1: 192.168.15.13;0.09
Enviando para nuvem dados nodo 1: 192.168.15.13;0.14
Enviando para nuvem dados nodo 1: 192.168.15.13;0.29
Enviando para nuvem dados nodo 1: 192.168.15.13;0.23
Enviando para nuvem dados nodo 1: 192.168.15.13;0.11
Enviando para nuvem dados nodo 1: 192.168.15.13;0.37
Enviando para nuvem dados nodo 1: 192.168.15.13;0.29
Enviando para nuvem dados nodo 1: 192.168.15.13;0.2
Enviando para nuvem dados nodo 1: 192.168.15.13;0.37
Enviando para nuvem dados nodo 1: 192.168.15.13;0.14
Enviando para nuvem dados nodo 1: 192.168.15.13;0.34
```

Fonte: Autoria própria (2022).

Figura 17 – Aplicação Servidor Nuvem

```
samuel@samuel-Aspire-4739Z:~/Downloads$ python3 cloud.py 9000
Cloud escutando na porta 9000
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.32
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 1.21
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.52
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.89
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.26
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.2
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.09
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.32
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.43
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.09
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.14
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.29
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.23
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.11
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.37
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.29
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.2
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.37
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.14
Valor recebido: IP nodo=192.168.15.13 Uso medio de CPU= 0.34
```

Fonte: Autoria própria (2022).

3.2 SELEÇÃO DAS FERRAMENTAS DE GERENCIAMENTO

Para realizar a seleção das ferramentas de gerenciamento a serem testadas no ambiente proposto, tomou-se por base as ferramentas indicadas nos artigos analisados na Seção 2.4. A Tabela 1 mostra o nome de todas as ferramentas de gerenciamento citadas e o número de artigos em que as mesmas foram encontradas. Pode-se verificar que nenhuma dessas ferramentas teve um grande destaque no número de citações. Porém as ferramentas Ganglia, Nagios, Zabbix, PCMONS, OpenNebula, DARGOS, Lattice e JCatascopia foram citadas duas vezes. Algumas dessas, são ferramentas de gerenciamento conhecidas como o Nagios e o Zabbix.

Tabela 1 – Quantidade de Menções das Ferramentas

Ferramenta	Quantidade de Menções
EyeQ Framework	1
Nebula Monitor	1
Cloudy	1
Foggy	1
Ganglia	2
Nagios	2
Monalisa	1
Zabbix	2
PCMONS	2
OpenNebula	2
DARGOS	2
Lattice	2
JCatascopia	2
Lom2hi	1
CASVID	1
Tower 4Clouds	1
Zenoss	1
WARMS Framework	1
ETSI MANO Framework	1
COSCO Framework	1

Fonte: Autoria própria (2021).

Abreha (2020) investigou diversas das ferramentas de monitoramento com o intuito de verificar requisitos de escalabilidade, robustez, não-intrusividade, interoperabilidade, facilidade de migração, amostragem adaptativa e filtragem adaptativa. As ferramentas analisadas por ele (Quadro 23) convergem com a revisão sistemática da Seção 2.4.

Quadro 23 – Características das Ferramentas de Monitoramento

Ferramenta	Escalabilidade	Robustez	Não-Intrusividade	Interoperabilidade	Facilidade de Migração	Amostragem Adaptativa	Filtragem Adaptativa
Ganglia	Sim	Sim	Limitado	Sim	Sim	Não	Não
Zabbix	Sim	Não	Sim	Sim	Não	Não	Não
MonALISA	Sim	Sim	Não	Sim	Limitado	Não	Não
Nagios	Não	Não	Limitado	Sim	Não	Não	Não
PCMONS	Não	Não	Limitado	Sim	Não	Não	Não
OpenNebula	Sim	Sim	Sim	Não	Limitado	Não	Não
DARGOS	Sim	Não	Sim	Não	Não	Não	Não
Lattice	Sim	Sim	Sim	Sim	Sim	Não	Não
JCatascopia	Sim	Sim	Limitado	Sim	Sim	Limitado	Limitado

Fonte: (ABREHA, 2020).

Os artigos de Abreha (2020) e Prasad, Bhavsar & Tanwar (2019) descreveram as principais características das ferramentas apresentadas no Quadro 23. A ferramenta Ganglia é hierárquica, distribuída e escalável para monitorar sistemas computacionais de alto desempenho como *Grids*. Para sua utilização, os nodos necessitam ter conhecimento de todos os estados do *cluster*. O seu desenvolvimento foi destinado para sistemas fixos ou que sofressem pouca mudança de infraestrutura o que é contra o ambiente de Névoa. Além disso, não possui a capacidade de auto descoberta e auto configuração.

O Nagios é uma ferramenta de código aberto para monitoramento de redes, servidores e aplicações. Possui a característica de oferecer monitoramento constante para identificar problemas de aplicações, serviços ou processos e tomar uma ação o quanto antes. Não possui uma base de dados mas possui extensões para exportar os dados gerados o que contribui para limitar sua flexibilidade e escalabilidade. Além disso, possui difícil configuração e não apresenta descoberta automática de dispositivos não sendo viável para sistemas que exijam alta taxa de atualização.

Monitoring Agents in a Large Integrated Services Architecture (MonALISA) é uma ferramenta robusta, escalável e distribuída para monitoramento de sistemas. Possui a funcionalidade de integração com outras ferramentas como o Ganglia utilizando o protocolo SNMP. Seu principal uso é relacionado a ambientes fixos e que sofrem poucas mudanças.

O Zabbix é uma ferramenta de código aberto para monitoramento de sistemas computacionais distribuídos. Possui uma estrutura hierárquica o que contribui para sua escalabilidade. Seus agentes são instalados localmente em cada recurso a ser monitorado de forma a coletar dados e enviar para os servidores. Permite o envio de alertas para o administrador de forma a garantir o correto funcionamento dos sistemas. Em sua estrutura, possui um servidor centralizado com outros colocados hierarquicamente para permitir a capacidade de federação. Dessa maneira, a flexibilidade é comprometida. Além disso, a auto descoberta de dispositivos é lenta contribuindo para que o seu uso seja adequado para ambientes com poucas mudanças estruturais.

Private Cloud Monitoring System (PCMONS) é uma ferramenta de código aberto modular e extensível. De forma geral, permite coletar informações relevantes para monitoramento, sendo compatível com o Nagios. Seu uso adequa-se somente a ambientes com baixas taxas de atualização.

OpenNebula consiste em um conjunto de ferramentas para gerenciamento de arquiteturas em Nuvem. Seu funcionamento baseia-se na coleta de informações do ambiente como *status* dos nodos e de máquinas virtuais e no envio para um servidor. Há dois modos de operação: modo *pull* ou modo *push*. No primeiro, o OpenNebula periodicamente requisita dados aos dispositivos enquanto que no segundo, cada dispositivo encaminha dados relativos ao monitoramento via protocolo UDP. Trata-se de um sistema leve, de fácil instalação e configuração. Sua desvantagem consiste em possuir atualizações dos dados de monitoramento em períodos

estáticos e periódicos, o que colabora por não atender ambientes dinâmicos.

DARGOS é uma ferramenta distribuída de monitoramento que utiliza um sistema híbrido de *push/pull* para disseminação de dados. Suporta ambientes virtuais e físicos na arquitetura de Nuvem com baixo uso de recursos. Além disso, é flexível e extensível para a adição de novas métricas. Dessa forma, pode ser integrado a um sistema real de maneira simplificada.

A ferramenta Lattice possui código aberto e preza por um monitoramento não intrusivo, permitindo a integração com outros sistemas de gerenciamento e monitoramento. Foi proposto para monitorar serviços e recursos em ambientes virtualizados e dinâmicos. A coleta de dados consiste no uso de *multicast* ou no protocolo UDP. Algumas desvantagens são: não possuir ferramenta para visualização e geração de alertas automatizados além de não possuir uma biblioteca de recursos para monitoramento de forma com que os desenvolvedores necessitem criar seus próprios *scripts* de conexão.

JCatascopia é outra ferramenta de código aberto capaz de oferecer provisionamento de recursos de maneira automatizada e dinâmica oferecendo interoperabilidade e escalabilidade. Fornece o uso de filtros, permitindo aos usuários adicionar métricas possibilitando reduzir o tráfego de dados e o espaço necessário para armazenamento. Baseia-se em três componentes: agentes de Monitoramento, *probes* e servidor de monitoramento. Os agentes de monitoramento são instâncias presentes em dispositivos físicos ou virtuais que coletam informações dos nodos e realizam o agrupamento por métricas para posterior envio ao servidor. *Probes* coletam métricas gerenciados por agentes de monitoramento, sendo responsável portanto por coletar dados de baixo nível além de informações de desempenho. O servidor de monitoramento processa os dados recebidos performando agregações das várias instâncias conectadas, podendo ser implementado em ambiente físico ou virtual. A identificação automática da adição ou remoção de instâncias e o monitoramento dinâmico são algumas das vantagens dessa ferramenta enquanto que o alto consumo de recursos é uma desvantagem.

Para a escolha das ferramentas a serem avaliadas, foram considerados os seguintes critérios:

1. Quantidade de repetições que a solução de gerenciamento possui nos artigos analisados (pelo menos 2);
2. O software gerente da solução de gerenciamento, instalado no servidor, deve ser compatível com o sistema operacional Linux;
3. O software agente da solução de gerenciamento deve ser compatível com o sistema operacional Linux;
4. Disponibilidade: Solução de gerenciamento deve ser de código livre ou com versão de teste disponível;

5. Documentação: para que o software seja utilizado da maneira correta, faz-se necessário que uma documentação detalhada esteja disponível para auxiliar no seu uso.

O Quadro 24 aponta a compatibilidade das ferramentas de acordo com os critérios elencados e com o ambiente de teste utilizado. A ferramenta DARGOS não foi incluída devido à falta de informações disponíveis.

Quadro 24 – Compatibilidade das Ferramentas

Ferramenta	Compatibilidade com o Sistema Operacional	Disponibilidade	Documentação
Ganglia	X	X	X
Nagios	X	X	X
Zabbix	X	X	X
PCMONS	X	X	
OpenNebula	X	X	
Lattice	X	X	
JCatascopia	X	X	

Fonte: Autoria própria (2022).

Com base no Quadro 24, as ferramentas Ganglia, Nagios e Zabbix em suas versões mais recentes foram escolhidas para realizar os testes tendo em vista sua compatibilidade com o sistema operacional do servidor e dos agentes, disponibilidade para *download* em versão de testes ou por possuir código livre além de ser as ferramentas com maior documentação disponível.

Essas ferramentas também satisfazem os requisitos mínimos de hardware (Quadro 25) já que o servidor e os hosts utilizados possuem uma configuração de hardware suficiente para a execução do servidor e dos agentes. O armazenamento e a memória RAM necessárias variam conforme a quantidade de agentes monitorados bem como a quantidade de itens em cada um. Para a ferramenta Ganglia, não foram encontradas informações de requisitos mínimos em nenhuma fonte oficial.

Quadro 25 – Requisitos Mínimos das Ferramentas

Ferramenta	Requisito	Descrição
Nagios	Processador	1 GHz
	Armazenamento	8 GB
	Memória RAM	1 GB
Zabbix	Processador	Dois núcleos
	Armazenamento	10 GB
	Memória RAM	2 GB

Fonte: Autoria própria (2022).

Em relação ao suporte para o sistema operacional a ser instalado, o Ganglia suporta nativamente distribuições Linux, macOS e Solaris (MASSIE, 2012), já o Nagios XI é compatível com Linux (CentOS, Redhat Enterprise Linux, Oracle, Ubuntu e Debian) e Windows (NAGIOS…, 2022), enquanto que o Zabbix 6.0 LTS na versão gerente pode ser instalado somente

em distribuições Linux e os agentes em Windows, Linux, macOS, AIX, FreeBSD, OpenBSD e Solaris (ZABBIX..., 2022). O Nagios XI, em seu site oficial não apresenta diferenças de compatibilidade entre sistemas oferecidos por seus agentes e gerente.

3.3 CRITÉRIOS ANALISADOS

Diversos são os tipos de aplicações que tiram vantagem de um ambiente de Névoa, como as que necessitam de baixa latência, armazenamento em *cache*, segurança, robustez e disponibilidade. Nesse contexto, pode-se elencar o tratamento de dados de sensores, gerência de dados na área da saúde, redes inteligentes de energia, redes veiculares, IoT, realidade aumentada, etc.

Para uma rede suportar a arquitetura de Névoa, necessita atender a diversos aspectos técnicos como: baixa latência de comunicação entre os dispositivos, confiabilidade, segurança, largura de banda, etc.. Dessa maneira, para uma ferramenta de gerenciamento ser considerada apta para a utilização nesse meio, é necessário que seja compatível com esses requerimentos.

Para elencar os requisitos mais importantes que as ferramentas de gerenciamento devem suportar, os requisitos de gerenciamento de recursos e QoS da Seção 2.4 foram divididos em categorias tendo como base a semelhança entre esses requisitos. A Tabela 2 apresenta a quantidade de itens em cada categoria.

Tabela 2 – Quantidade de Menções das Categorias

Categoria	Quantidade de Menções
Auto Descoberta	1
Desempenho	38
Balanceamento	2
Placement	3
Escalabilidade e Heterogeneidade	10
Adaptabilidade e Dinamismo	5
Robustez	29
Não-intrusividade	2
Interoperabilidade	2
Migração	2
Autonomicidade	2
Segurança	8
Mobilidade	2
Aspectos Gerenciais	24
Cache	1
Conectividade	1

Fonte: Autoria própria (2021).

A categoria desempenho engloba requisitos de largura de banda, sensitividade a *delay* e perda, latência, necessidade de tempo real, taxa de transferência, utilização de recursos, carga de trabalho e requisitos de tempo. A categoria de balanceamento envolve o balanceamento de carga. *Placement* engloba características relacionadas à colocação dos ambientes e uso de localização. A categoria escalabilidade e heterogeneidade envolve também aspectos de elasticidade. Em adaptabilidade e dinamismo são incluídos também requisitos de flexibilidade. Em robustez, são inclusos requerimentos de confiabilidade, disponibilidade, avaliabilidade, gerenciamento de falhas, prazo, *Service Level Agreement* (SLA), *Service Level Objectives* (SLO), taxa de erros, coleta de dados e congestionamento de dados na rede. Em aspectos gerenciais, também são abrangidos aspectos de custos, manutenção, consumo energético, acessibilidade, complexidade, uso de recursos e eficiência. O Quadro 26 ilustra os critérios presentes em cada categoria.

Quadro 26 – Requisitos Inclusos em Cada Categoria

(continua)

Categoria	**Requisitos**
Auto Descoberta	Auto descoberta de dispositivos

Quadro 26 – Requisitos Inclusos em Cada Categoria

(continua)

Categoria	**Requisitos**
Desempenho	Largura de banda Consumo de banda Sensitividade a *delay* e perda Latência (atraso) Processamento em tempo real Taxa de transferência de dados Tempo de resposta Utilização de recursos Tempo de execução Baixo tempo de agendamento Carga de trabalho
Balanceamento	Balanceamento de carga
Placement	Locação dos dados Uso de localização
Escalabilidade e Heterogeneidade	Suporte a escalabilidade e heterogeneidade Elasticidade Gerenciamento de grande número de dispositivos
Adaptabilidade e Dinamismo	Suporte a adaptabilidade e dinamismo Flexibilidade
Robustez	Confiabilidade Disponibilidade e avaliabilidade Prover informações acuradas do ambiente Gerenciamento de falhas Prazo final (tempo para completar uma requisição) Taxa de violação de SLA Taxa de erros Violação de SLO Congestionamento de dados na rede
Não-intrusividade	Permitir não-intrusividade
Interoperabilidade	Facilidade em interoperabilidade Federação
Migração	Suporte para migração
Autonomicidade	Suporte para autonomicidade Automação

Quadro 26 – Requisitos Inclusos em Cada Categoria

(conclusão)

Categoria	**Requisitos**
Segurança	Suporte para segurança Privacidade
Mobilidade	Suporte para mobilidade
Aspectos Gerenciais	Custo benefício Consumo de energia Custo Gerenciamento Acessibilidade Complexidade Eficiência Utilização de recursos Custo de SLA Eficiência energética
Cache	Mecanismos de *cache*
Conectividade	Permitir conectividade

Fonte: Autoria própria (2021).

Das categorias elencadas, algumas não são possíveis de serem testadas no ambiente proposto devido às razões seguintes:

1. Auto Descoberta: não são adicionados novos dispositivos em tempo de execução;
2. Balanceamento: devido ao baixo número de dispositivos na Névoa, o balanceamento torna-se dispensável;
3. *Placement*: uso de apenas um servidor de Névoa, dados são concentrados nesse servidor;
4. Escalabilidade e Heterogeneidade: não são adicionados novos dispositivos em tempo de execução;
5. Adaptabilidade e Dinamismo: arquitetura proposta para os testes segue uma estrutura fixa;
6. Não-intrusividade: devido à aplicação de coleta de dados dos nodos de Névoa não exigir alto poder computacional, o uso das ferramentas de monitoramento não devem interferir consideravelmente no processamento do servidor;
7. Migração: ambiente de testes não sofre migração;

8. Autonomicidade: não serão delegadas tarefas às ferramentas com requisitos de autonomicidade;
9. Segurança: não são trafegados dados criptografados no ambiente;
10. Mobilidade: ambiente de testes não possui requisitos de mobilidade;
11. Aspectos Gerenciais: aspectos de custos, consumo energético, eficiência, etc. não necessitam ser monitorados devido à baixa complexidade da arquitetura proposta;
12. *Cache*: não é necessário uso de *cache* já que trata-se de dados provindos de sensores;
13. Conectividade: ambiente de testes não possui requisitos de conectividade.

Dessa maneira, a categoria de desempenho, robustez e interoperabilidade foram elencadas. Testar requisitos da categoria de desempenho é importante pois engloba aspectos de uso de banda, *delay*, perda, etc. que são itens essenciais em um ambiente de Névoa. Já os testes para categoria robustez são necessários por estarem relacionados à disponibilidade, avaliabilidade, acuracidade, etc. o que também são essenciais dentro do contexto de Névoa. A interoperabilidade torna-se necessária visando a assertividade da troca de informações entre as ferramentas e os demais softwares.

Como o intuito desse trabalho é validar o gerenciamento do ambiente em Névoa, os softwares de gerenciamento devem ser capazes de gerenciar as máquinas agentes, a comunicação entre essas máquinas e o servidor local (rede) e o servidor local. Dessa maneira, o Quadro 27 apresenta a relação entre os requisitos elencados para o teste do gerenciamento e o componente em questão (agentes, rede ou servidor).

Quadro 27 – Componentes da Névoa para Avaliação de Requisitos

Categoria	**Requisito**	**Agentes**	**Rede**	**Servidor**
Desempenho	Largura de banda		X	
Desempenho	Consumo de banda		X	
Desempenho	Latência		X	
Desempenho	Taxa de transferência de dados	X	X	X
Desempenho	Utilização de recursos	X	X	X
Desempenho	Carga de trabalho	X	X	X
Robustez	Disponibilidade e avaliabilidade	X	X	X
Robustez	Congestionamento de dados na rede		X	
Interoperabilidade	Facilidade em interoperabilidade	X	X	X

Fonte: Autoria própria (2022).

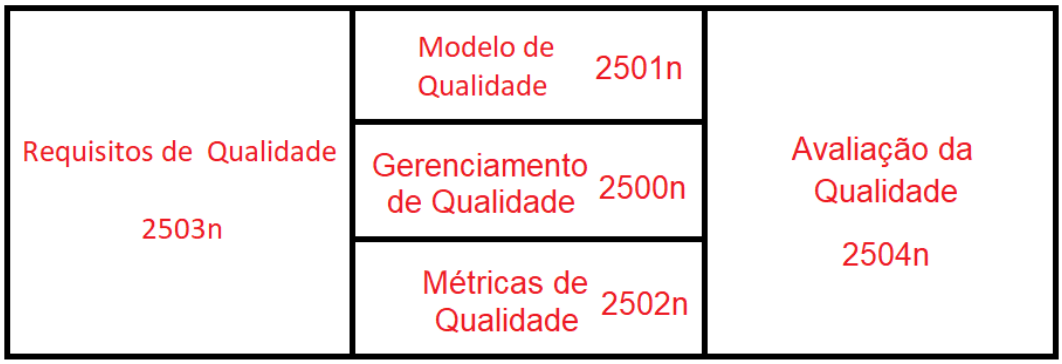
3.4 AVALIAÇÃO DAS SOLUÇÕES DE GERENCIAMENTO

A série de normas 25000 SQuaRE visa normatizar a área de Engenharia de Software de forma a estabelecer padrões ao que tange a avaliação de software e definição de critérios para requisitos de qualidade, auxiliando desenvolvedores e consumidores de produtos de software (ISO, 2005). A distribuição das normas segue a divisão:

1. Divisão da Gestão da Qualidade (ISO/IEC 2500n);
2. Divisão de Modelo de Qualidade (ISO/IEC 2501n);
3. Divisão de Medição de Qualidade (ISO/IEC 2502n);
4. Divisão de Requisitos de Qualidade (ISO/IEC 2503n);
5. Divisão de Avaliação da Qualidade (ISO/IEC 2504n).

Essa norma é uma evolução das normas ISO/IEC 9126 e ISO/IEC 14598, surgindo para suprir a falta de clareza existente. No geral, a norma SQuaRE é mais organizada e unificada, visando a especificação de requisitos e a avaliação da qualidade de software, podendo ser utilizada por gerentes de produtos e desenvolvedores para auxiliar o processo de desenvolvimento (FILHO, 2011).

Figura 18 – Arquitetura da Norma ISO/IEC 25000



Fonte: Adaptado de Filho (2011)

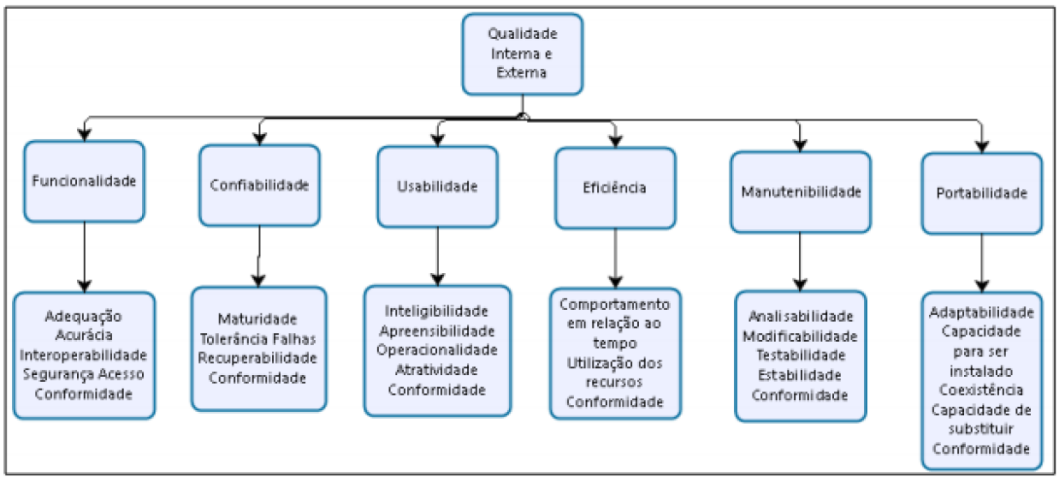
A Figura 18 mostra a arquitetura da série ISO/IEC 25000 de forma simplificada, sendo dividida em:

1. Gerenciamento de Qualidade: possui documentos dedicados a todos usuários possíveis. Contém definição dos termos utilizados;
2. Modelo de Qualidade: relacionado à antiga ISO/IEC 9126. Contém definições dos contextos de qualidade;

3. Métricas de Qualidade: relacionado ao processo de medição;
4. Requisitos de Qualidade: estabelecimento de objetivos de qualidade para um produto;
5. Avaliação da Qualidade: definição de procedimentos e metodologias para as avaliações práticas, tendo como objetivo a média de qualidade pretendida.

Como o objetivo desse trabalho é averiguar a adequação dos softwares de gerenciamento no suporte da computação em Névoa, a norma ISO/IEC 25010 define as características de qualidade para um produto de software, sendo composta de sub características (Figura 19) (ISO/IEC 25010, 2011). Para realizar a avaliação das ferramentas, foram utilizadas as medidas e fórmulas por essa definidas.

Figura 19 – ISO/IEC 25010



Fonte: (ISO/IEC 25010, 2011)

A característica de funcionalidade foi escolhida com o intuito de avaliar se as ferramentas se adequam à arquitetura de Névoa. Dessa maneira, são avaliadas as subcaracterísticas de adequação, e interoperabilidade.

A subcaracterística adequação foi selecionada com o intuito de avaliar se as ferramentas estão aptas a realizar as tarefas especificadas. A interoperabilidade foi escolhida para verficar a frequência de falhas na disposição dos dados através das ferramentas. A conformidade exerce a função de averiguar a funcionalidade com padrões e convenções aplicáveis, sendo assim, não foi selecionada.

A subcaracterística acurácia não foi selecionada já que o objetivo do trabalho não é avaliar se as informações exibidas são ou não são verídicas. A segurança de acesso também não foi selecionada pois esse trabalho não analisa aspectos de segurança em Névoa.

Após definir os critérios, torna-se necessário definir as métricas para mensurar esses critérios. Métricas internas baseiam-se em medições considerando as característica internas, sem executar o software avaliado. Sendo assim, para esse trabalho, não são adequadas com os objetivos estabelecidos.

Métricas externas adequam-se aos objetivos propostos já que baseiam-se na execução do software e na coleta de resultados no seu ambiente de execução, dessa maneira, foram adotadas para esse trabalho.

Para realizar a avaliação das ferramentas, foram utilizadas fórmulas e medidas definidas na ISO/IEC 25010 (2011) presentes no Quadro 31 e no Quadro 32. Sendo assim, foram elencadas as seguintes funcionalidades a serem testadas:

1. Avaliação de requisitos de desempenho: averiguar se as ferramentas contemplam o monitoramento e gestão de requisitos de desempenho (largura de banda, consumo de banda, latência, taxa de transferência de dados, utilização de recursos e carga de trabalho);
2. Avaliação de requisitos de robustez: averiguar se as ferramentas contemplam o monitoramento e gestão de requisitos de robustez (disponibilidade, avaliabilidade e congestionamento de dados na rede);
3. Avaliação de requisitos de interoperabilidade: averiguar se as ferramentas contemplam o monitoramento e gestão de requisitos de interoperabilidade (facilidade em interoperabilidade).

A ISO/IEC 25010 (2011) utiliza a fórmula X = 1 - (A / B) para cálculo da funcionalidade em que A é o número de itens não conformes e B é o número total de itens testados. Esse trabalho avalia 2 itens de funcionalidade. Se a ferramenta não implementa a funcionalidade, recebe 0 ponto, se implementa parcialmente, recebe 0,5 ponto e se atende completamente, recebe 1 ponto.

Com o intuito de mensurar as funcionalidades das ferramentas, a subcaracterística de adequação possui peso maior que as demais. Sendo assim, é atribuído peso de 90% enquanto que a subcaracterística de interoperabilidade possui peso de 10%.

3.5 CASOS DE TESTES

Após a seleção das ferramentas, definição dos critérios e métricas de avaliação, é necessário definir os casos de testes utilizados para padronizar os testes e a avaliação das ferramentas. Os casos de testes têm como base as métricas e funcionalidades definidas no Capítulo 3. A relação de cada caso de teste é descrita no Quadro 28 ao Quadro 30.

Quadro 28 – Caso de Teste 01

Caso de Teste	Gerenciamento efetuado nos agentes
Resumo	Ferramenta deve monitorar os requisitos: taxa de transferência de dados, utilização de recursos, carga de trabalho, disponibilidade e avaliabilidade, facilidade em interoperabilidade. Atende as subcaracterísticas de adequação e interoperabilidade.
Pré-condições	Aplicação de coleta de dados em execução e software de monitoramento parametrizado
Ação	1. Aplicação sendo executada nos agentes; 2. Software de gerenciamento monitora os agentes; 3. Software de gerenciamento exibe no *dashboard* e em relatórios os requisitos em avaliação.
Resultados Esperados	Ferramenta de gerenciamento atende aos requisitos para monitoramento

Fonte: Autoria própria (2021).

Quadro 29 – Caso de Teste 02

Caso de Teste	Gerenciamento efetuado na rede
Resumo	Ferramenta deve monitorar os requisitos: largura de banda, consumo de banda, latência, taxa de transferência de dados, utilização de recursos, carga de trabalho, disponibilidade e avaliabilidade, congestionamento de dados na rede e facilidade em interoperabilidade. Atende as subcaracterísticas de adequação e interoperabilidade.
Pré-condições	Rede em operação na arquitetura de Névoa proposta
Ação	1. Aplicação sendo executadas nos agentes e no servidor; 2. Software de gerenciamento monitora o ambiente de Névoa; 3. Software de gerenciamento exibe no *dashboard* e em relatórios os requisitos em avaliação.
Resultados Esperados	Ferramenta de gerenciamento atende aos requisitos para monitoramento de rede

Fonte: Autoria própria (2021).

Quadro 30 – Caso de Teste 03

Caso de Teste	Gerenciamento efetuado no servidor
Resumo	Ferramenta deve monitorar os requisitos: taxa de transferência de dados, utilização de recursos, carga de trabalho, disponibilidade e avaliabilidade, facilidade em interoperabilidade. Atende as subcaracterísticas de adequação e interoperabilidade.
Pré-condições	Servidor em operação na arquitetura de Névoa proposta
Ação	1. Aplicação sendo executadas nos agentes e no servidor; 2. Software de gerenciamento monitora o ambiente de Névoa; 3. Software de gerenciamento exibe no *dashboard* e em relatórios os requisitos em avaliação.
Resultados Esperados	Ferramenta de gerenciamento atende aos requisitos para monitoramento do servidor

Fonte: Autoria própria (2021).

3.6 CONSIDERAÇÕES FINAIS

O capítulo explica a proposta de solução do trabalho, constituindo a base para desenvolver a avaliação realizada posteriormente. É proposta uma análise para as ferramentas de gerenciamento de redes em um ambiente de computação em Névoa com o intuito de averiguar quais dessas ferramentas possuem maior adequação com os requisitos para esse tipo de ambiente computacional.

Na Seção 3.2 são selecionadas as ferramentas a serem avaliadas. Diversas foram elencadas com base no processo de Revisão Sistemática apresentada na Seção 2.4. Após, foram selecionadas 3 delas (Ganglia, Nagios e Zabbix) considerando as que se adequam a um maior número de requisitos.

Após, foram selecionados os critérios na Seção 3.3 e com isso, foram definidas as métricas para mensurar esses critérios. Para esse trabalho, foram selecionadas métricas externas, que contemplam medições do software considerando a sua execução e os resultados no ambiente de testes.

Por fim, os casos de teste especificados na Seção 3.5 foram definidos para estabelecer um padrão para a avaliação das ferramentas especificando como são alcançados os resultados dos testes.

4 TESTE DAS FERRAMENTAS

Conforme a proposta de solução (Capítulo 3), as ferramentas selecionadas Ganglia, Nagios e Zabbix foram testadas de acordo com os casos de testes definidos na Seção 3.5, utilizando o cenário descrito na Seção 3.1. Os casos de testes definidos utilizam os requisitos citados no Quadro 27:

1. Largura de banda: expressa a máxima capacidade de transmissão e recebimento de dados em um rede considerando um certo intervalo de tempo[1]. No caso de redes sem fio, essa medida não é facilmente obtida, tendo em vista que dependerá de diversos fatores como obstáculos entre transmissor e receptor, umidade, interferências, etc.;

2. Consumo de banda: diz repeito ao percentual de uso em relação à capacidade total dessa rede (largura de banda)[2];

3. Latência: é uma medida do atraso de uma extremidade de uma rede, link, ou de um dispositivo a outra extremidade[3]. É utilizada como uma medida de desempenho de redes;

4. Taxa de transferência de dados: refere-se à medida de informações trafegadas na placa de rede por segundo;

5. Utilização de recursos: Com base nos tipos de recursos mais utilizados em cada dispositivo, o requisito de utilização de recursos contempla:

    a) Agentes e Rede: avaliados aspectos de uso de CPU e de memória RAM. O uso de disco foi descartado tendo em vista que esses dispositivos não realizam leitura e escrita de dados em um volume suficiente para seu monitoramento ser considerado essencial;

    b) Servidor: avaliados aspectos de uso de CPU, memória RAM e de uso de disco (espaço utilizado e taxas de leitura e escrita).

6. Carga de trabalho: expressa a média do número de processos em execução ou ininterruptos em um intervalo de tempo, sendo assim, trata-se de uma média tendo como base o números de núcleos do processador[4]. Como exemplo, considerando uma máquina com dois núcleos, sua média de carga de trabalho ideal seria um valor entre 0 e 2, indicando baixo uso de recursos de processamento até uma carga total de trabalho. Valores maiores que 2 nesse caso, indicam uma sobrecarga do sistema;

[1] https://its.ucsc.edu/security/bandwidth.html
[2] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1045.5391&rep=rep1&type=pdf
[3] https://www.gta.ufrj.br/grad/99_1/rodrigo/ger_redes.htm#3.1.5)%20LAT%C3%8ANCIA
[4] http://ibg.colorado.edu/ lessem/psyc5112/usail/man/linux/top.1.html

7. Disponibilidade e avaliabilidade: a disponibilidade é definida como a probabilidade de um sistema estar operando de forma correta e encontrar-se disponível para realizar suas tarefas (AMARAL, 2014). Já a avaliabilidade pode ser entendida como a média de tempo em porcentagem em que a rede está executando suas funções conforme é esperado [5];

8. Congestionamento de dados: ocorre quando há demanda de utilização maior do que a estrutura de rede pode comportar;

9. Facilidade em interoperabilidade: refere-se à capacidade de integração com outras ferramentas de monitoramento ou com softwares que permitam consumir os dados coletados. Para esse trabalho, é considerada a capacidade da ferramenta notificar sobre alertas ocorridos com as máquinas monitoradas via e-mail de maneira nativa (padrão na instalação da ferramenta e documentado pelos desenvolvedores).

4.1 GANGLIA

A instalação da ferramenta foi realizada conforme os passos definidos por Massie (2012) o qual também especifica como configurar o software agente instalado nos nodos, bem como o gerente instalado no servidor. O software disponibiliza gráficos separados em grupos em que cada um desses representa uma métrica diferente (memória, disco, etc.). Os gráficos podem ser filtrados em períodos de: 1 hora, 2 horas, 4 horas, último dia, última semana, último mês e último ano. Foram utilizados os tempos padrão de obtenção de dados da ferramenta.

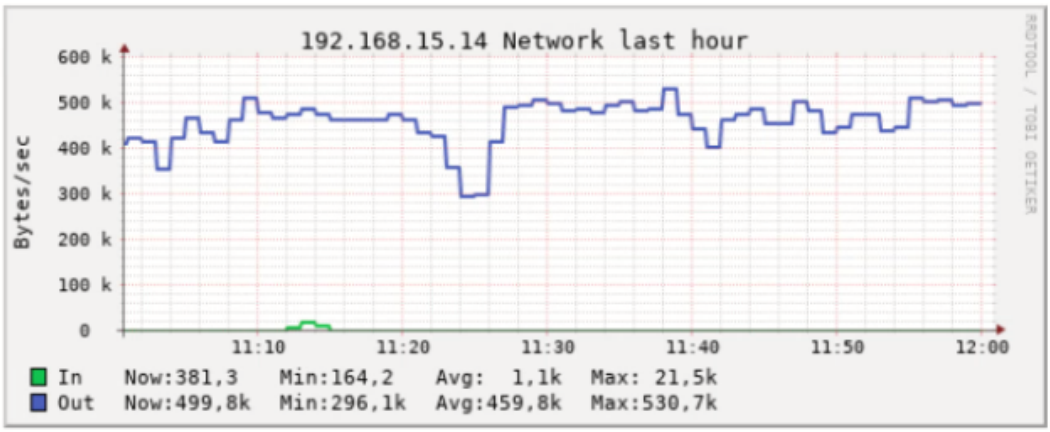
4.1.1 Caso de Teste 1: Gerenciamento Efetuado nos Agentes

O primeiro caso de teste (Quadro 28) consiste em averiguar se a ferramenta é capaz de monitorar requisitos de taxa de transferência de dados, utilização de recursos, carga de trabalho, disponibilidade e avaliabilidade além de facilidade em interoperabilidade.

O Ganglia mostra gráficos referentes à taxa de transferência de dados do dispositivo agente. Esses gráficos referem-se ao número de bytes enviados e recebidos por segundo pela interface do agente. No gráfico da Figura 20, o eixo x mostra a hora de recebimento e envio dos pacotes e o eixo y expressa a quantidade de bytes recebidos ou enviados por segundo pelo dispositivo. Além disso, são exibidas a transferência atual, mínima, máxima, e média.

[5] https://www.sciencedirect.com/topics/engineering/network-availability

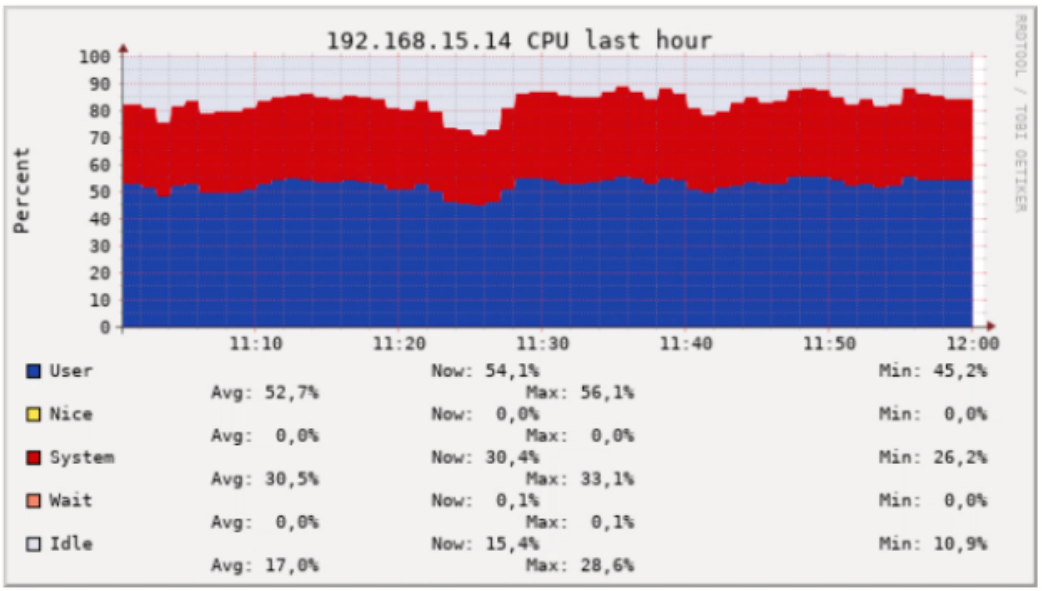
Figura 20 – Taxa de Transferência de Dados em Agentes (Ganglia)



Fonte: Autoria própria (2022).

A utilização dos recursos engloba uso de CPU e memória. A CPU é avaliada pelo gráfico da Figura 21. No eixo x, é informado o tempo em hora e minuto e no eixo y, são informados o percentual de utilização separado em tempo em espera, utilização em nível de usuário e sistema, relacionado à entrada e saída de dados além de tempo de execução com prioridade *nice* (tipo de prioridade do sistema operacional Linux calculada em nível de usuário[6]). A memória também é possível de ser monitorada. A Figura 22 ilustra o monitoramento o qual exibe no eixo x, o tempo em hora e minuto e no eixo y, na unidade *gigabytes*, a quantidade de memória em uso, compartilhada, em *cache*, em *buffer*, em *swap*, e a quantidade total da máquina. Para cada uma dessas categorias, também são exibidos o valor de média, mínima e máxima obtidos.

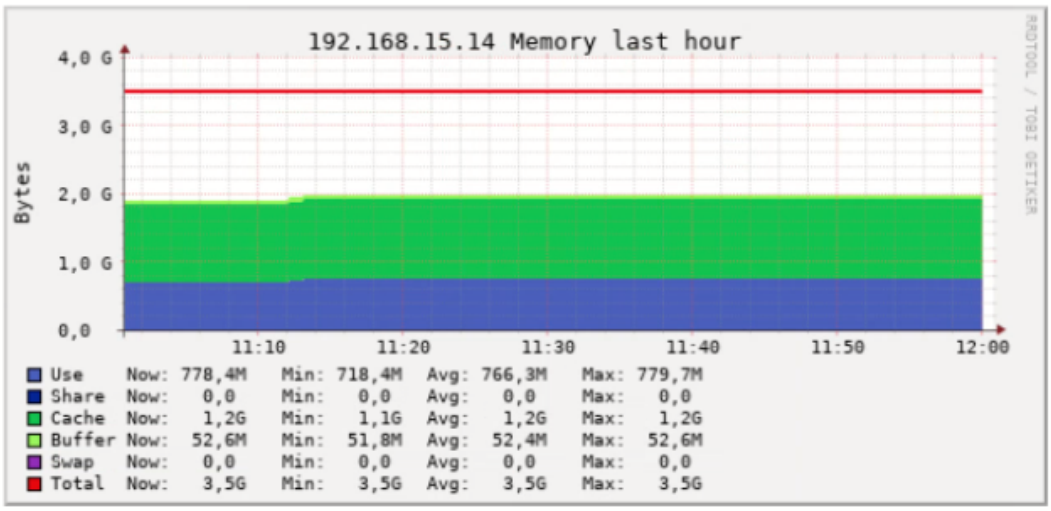
Figura 21 – Uso de CPU em Agentes (Ganglia)



Fonte: Autoria própria (2022).

[6] https://man7.org/linux/man-pages/man2/nice.2.html

Figura 22 – Uso de Memória RAM em Agentes (Ganglia)



Fonte: Autoria própria (2022).

A carga de trabalho pode ser monitorada a partir de valores relativos à média de carga para 1 minuto, 5 minutos e 15 minutos. A Figura 23 ilustra os dados obtidos em que no eixo x é exibido o tempo em hora e minuto e no eixo y, o valor da carga. Além disso, são informados os valores de média e máxima.

Figura 23 – Carga de Trabalho em Agentes (Ganglia)

Fonte: Autoria própria (2022).

Para testar a disponibilidade e avaliabilidade, tendo em vista que de forma nativa a ferrameta não oferece recursos para seu monitoramento, foram bucados nos repositórios oficiais[7] *plugins* que contemplem esses requisitos, de forma a ampliar as capacidades do Ganglia. Foram procurados os termos "*evaluability*" e "*availability*", o que não retornou resultados apropriados. Dessa maneira, constata-se que a ferramenta não possui capacidade de monitorar esses dois requisitos.

Por fim, a facilidade em interoperabilidade não foi contemplada, visto que a ferramenta não possui integração com e-mail de forma nativa.

Tendo como base os requisitos avaliados, foi possível acompanhar a taxa de transferência de dados, uso de recursos e carga de trabalho. A disponibilidade e avaliabilidade bem como a facilidade em interoperabilidade não foram averiguadas devido à falta de recursos do Ganglia.

4.1.2 Caso de Teste 2: Gerenciamento Efetuado na Rede

O segundo caso de teste (Quadro 29) tem como objetivo averiguar a capacidade de monitoramento para largura de banda, consumo de banda, latência, taxa de transferência de dados, utilização de recursos, carga de trabalho, disponibilidade e avaliabilidade, congestionamento de dados na rede e facilidade em interoperabilidade.

Tendo em vista que a ferramenta em análise não possui integração com o protocolo SNMP documentada para o usuário final nem um agente oficial para o sistema operacional OpenWrt, acaba não sendo possível realizar o monitoramento da rede.

4.1.3 Caso de Teste 3: Gerenciamento Efetuado no Servidor

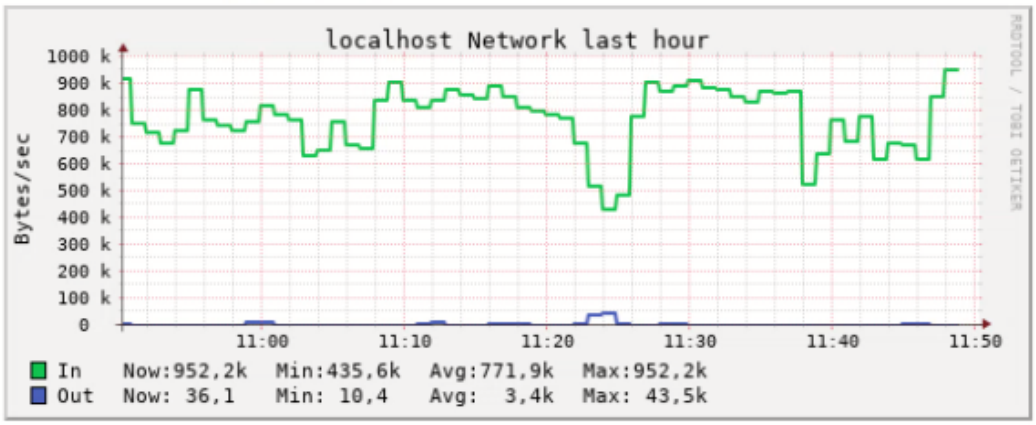
O terceiro caso de teste (Quadro 30) consiste em averiguar se a ferramenta é capaz de monitorar requisitos de taxa de transferência de dados, utilização de recursos, carga de trabalho, disponibilidade e avaliabilidade além de facilidade em interoperabilidade. Sendo assim, segue os mesmos requisitos dos casos de testes dos agentes, o que permite obter as informações de monitoramento através dos mesmos grupos de métricas no Ganglia.

A ferramenta permite exibir gráficos sobre a transferência de dados no servidor. A Figura 24 mostra no eixo y, a transferência em bytes enviados e recebidos por segundo na placa de rede, enquanto que no eixo x, é exibido o tempo em hora e minuto. Além disso, são informados a transferência atual, mínima, média e máxima.

A utilização dos recursos engloba uso de disco, CPU e memória. A Figura 25 exibe os gráficos ralacionados a disco. No eixo x, é exibida a hora e minuto enquanto que no eixo y, são exibidas informações diferentes para cada gráfico: espaço disponível (em *gigabytes*), total

[7] https://github.com/ganglia

Figura 24 – Taxa de Transferência de Dados no Servidor (Ganglia)



Fonte: Autoria própria (2022).

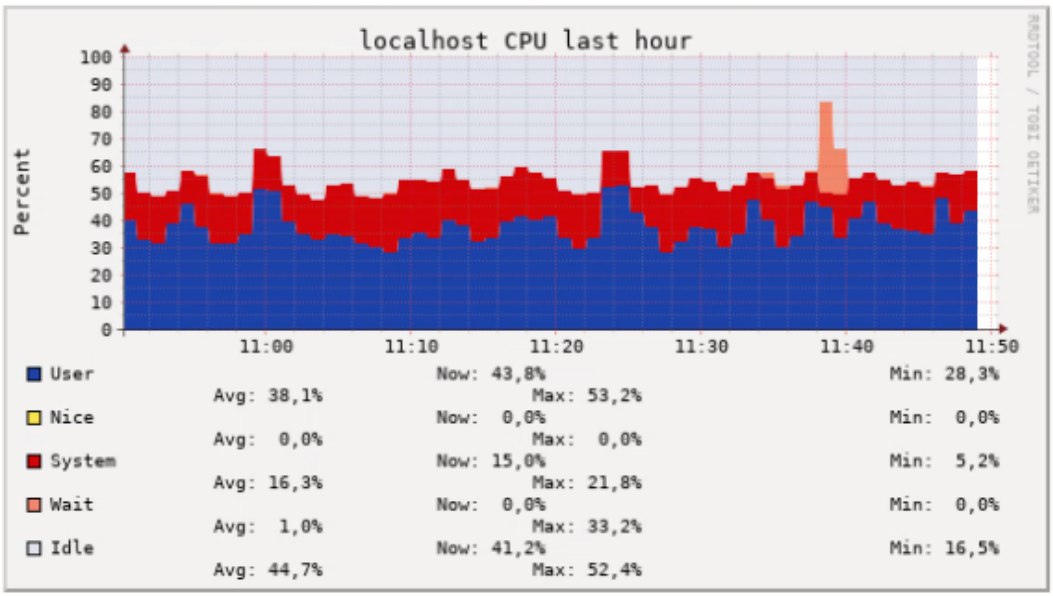
de espaço (em *gigabytes*) e máximo de espaço utilizado (em porcentagem). Além disso, há informações de média e máxima para cada gráfico.

Figura 25 – Utilização de Recursos de Disco no Servidor (Ganglia)

Fonte: Autoria própria (2022).

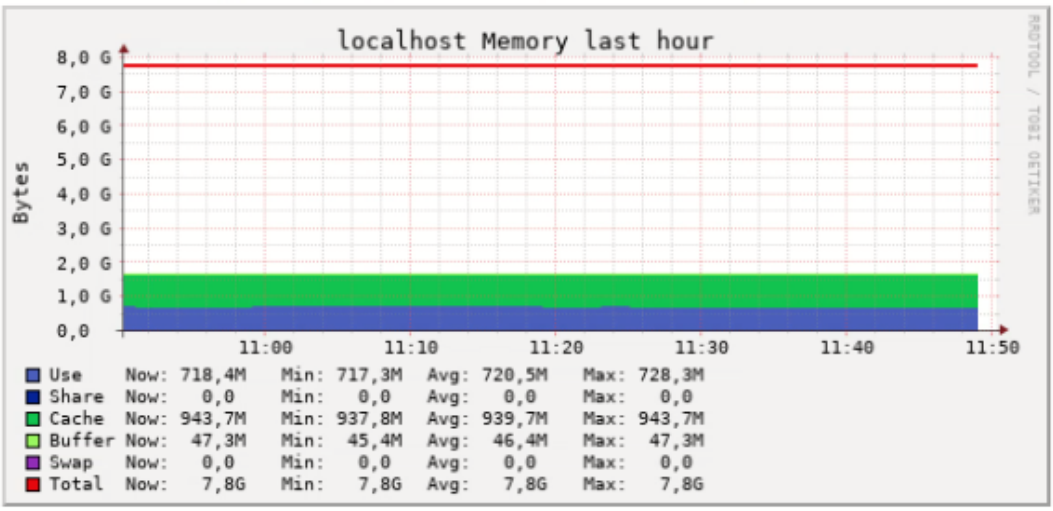
A ferramenta permite acompanhar a utilização de CPU no servidor. A Figura 26 exibe o gráfico obtido. No eixo x, é informada a hora e minuto enquanto que no y, são informados o percentual de utilização separado em tempo em espera, utilização em nível de usuário e sistema, relacionado à entrada e saída de dados além de tempo de execução em nível de usuário com prioridade *nice*. Já para o uso de memória, no eixo x, há o tempo em hora e minuto e no eixo y, na unidade *gigabytes*, a quantidade de memória em uso, compartilhada, em *cache*, em *buffer*, em *swap*, e a quantidade total da máquina servidora (Figura 27). Também são informados o uso corrente, mínimo, máximo e médio para CPU e memória.

Figura 26 – Uso de CPU no Servidor (Ganglia)



Fonte: Autoria própria (2022).

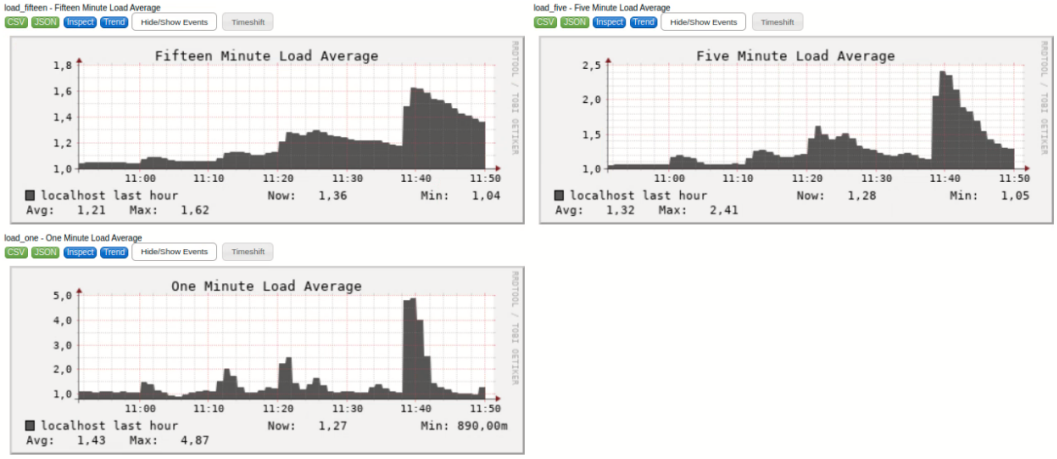
Figura 27 – Uso de Memória RAM no Servidor (Ganglia)



Fonte: Autoria própria (2022).

As informações acerca da carga de trabalho podem ser obtidas para 1 minuto, 5 minutos e 15 minutos juntamente com a média e a máxima. A Figura 28 exibe os gráficos relacionados. No eixo x, há informação do tempo de obtenção do dado enquanto que no y, há o valor da carga.

Figura 28 – Carga de Trabalho no Servidor (Ganglia)



Fonte: Autoria própria (2022).

A disponibilidade e avaliabilidade não foi contemplada pois da mesma forma com que para os agentes, a ferramenta não possui essa capacidade de forma nativa assim como não possui *plugins* para estender a capacidade do Ganglia de maneira a contemplar esses requisitos.

Assim como para os agentes, a facilidade em interoperabilidade também não foi contemplada visto que a ferramenta não possui integração com e-mail por padrão.

Em resumo, foi possível acompanhar em sua totalidade: taxa de transferência de dados e carga de trabalho. De forma parcial, foi possível acompanhar o uso de recursos tendo em vista que não foram obtidos dados de taxa de leitura e escrita de disco. Já a disponibilidade

e avaliabilidade assim como a facilidade em interoperabilidade não puderam ser monitoradas devido à falta de recursos da ferramenta.

4.2 NAGIOS

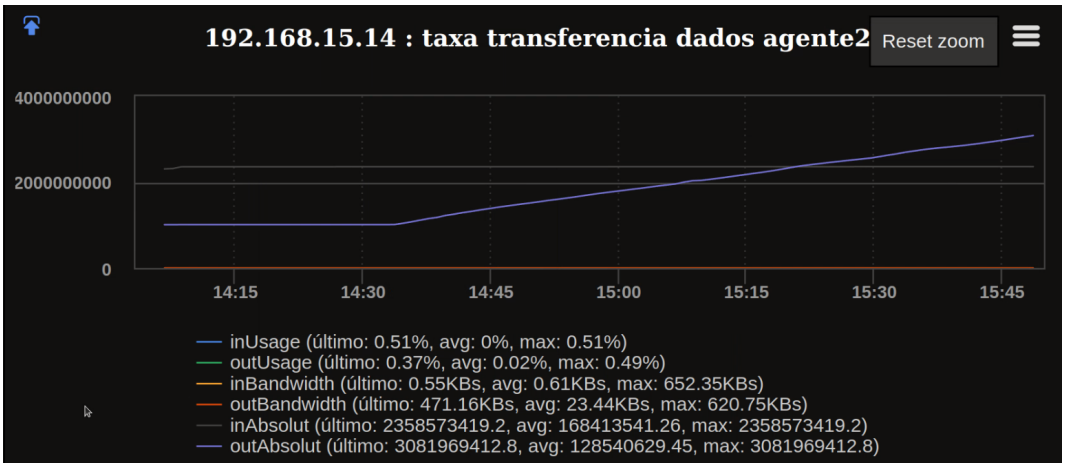
Para realizar a instalação da ferramenta no servidor foi utilizada a documentação oficial [8], para a instalação nos agentes e dispositivo de rede (ponto de acesso), foi utilizado o monitoramento via SNMP o qual também é especificado na documentação oficial [9]. O intervalo de obtenção de dados foi configurado para 1 minuto.

4.2.1 Caso de Teste 1: Gerenciamento Efetuado nos Agentes

Para o primeiro caso de teste, descrito no Quadro 28, foi possível obter as informações através do uso do template "Linux SNMP" o qual já possui diversas das métricas a serem monitoradas por padrão. Para as que não estão integradas inicialmente, pode-se utilizar *plugins* dedicados para a coleta de dados específicos.

A taxa de transferência de dados foi monitorada através do *plugin* "check_iftraffic3" [10] que permite obter a transferência de dados na placa de rede dos agentes. A Figura 29 exibe o tráfego durante o período de tempo dos testes. O eixo x expressa o tempo enquanto o y disponibiliza diversas informações: uso da interface de rede, largura de banda em uso (da interface somente) além de dados absolutos de entrada e saída. Para cada um desses, há informações do último dado coletado, média e máxima.

Figura 29 – Taxa de Transferência de Dados em Agentes (Nagios)

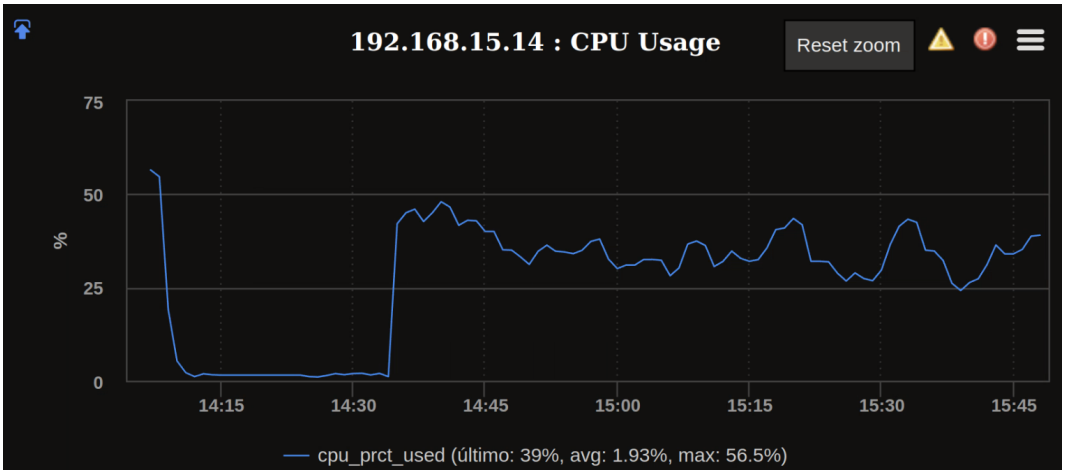


Fonte: Autoria própria (2022).

[8] https://assets.nagios.com/downloads/nagiosxi/docs/Installing-Nagios-XI-Manually-on-Linux.pdf
[9] https://assets.nagios.com/downloads/nagiosxi/docs/Monitoring-Linux-Using-SNMP.pdf
[10] https://exchange.nagios.org/directory/Plugins/Network-Connections%2C-Stats-and-Bandwidth/check_iftraffic3/details

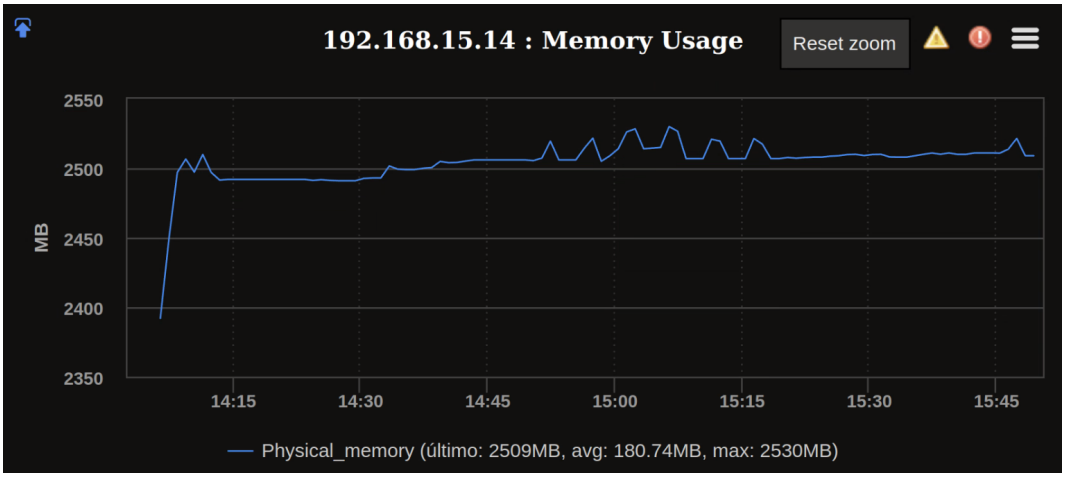
A utilização dos recursos engloba uso de CPU e memória. Para a CPU, os dados foram obtidos através do menu Serviços -> Uso de CPU (Figura 30). Para a memória, foi utilizado o menu Serviços -> Uso de Memória (Figura 31). As duas imagens condensam o uso do recurso (eixo y) ao longo do tempo (eixo x) além de exibir o último dado coletado, média e máxima.

Figura 30 – Uso de CPU em Agentes (Nagios)



Fonte: Autoria própria (2022).

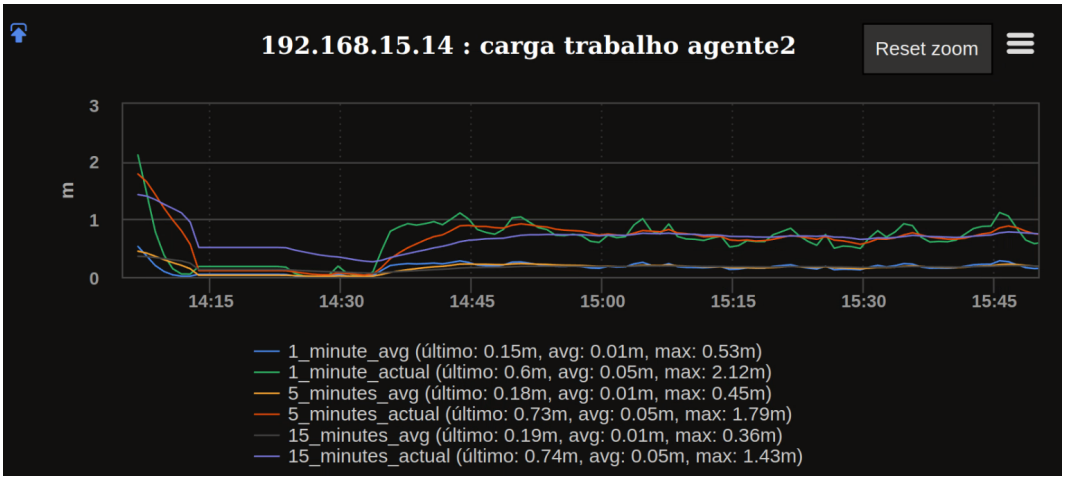
Figura 31 – Uso de Memória RAM em Agentes (Nagios)



Fonte: Autoria própria (2022).

Para a carga de trabalho, foi utilizado o *plugin* “snmp_remote_load”[11] que permite verificar as médias de carga de trabalho para 1 minuto, 5 minutos e 15 minutos. A Figura 32 exibe o gráfico gerado pelo *plugin*. No eixo y está o valor da carga enquanto que o x refere-se ao tempo. São disponibilizados dados médios e atuais para cada tempo (1 minuto, 5 minutos e 15 minutos).

Figura 32 – Carga de Trabalho em Agentes (Nagios)



Fonte: Autoria própria (2022).

Tendo em vista que a ferramenta não monitora os requisitos de disponibilidade e avaliabilidade por padrão, foi realizada uma busca no site oficial de extensões do Nagios[12]. Foi procurado por *plugins* através de uma pesquisa avançada utilizando os termos “availability” e “evaluability”. Tendo em vista que não foram retornados resultados condizentes, constata-se que a ferramenta não é capaz de averiguar esses requisitos.

Para a facilidade em interoperabilidade, é possível configurar avisos por e-mail para sinalizar sobre alertas ocorridos nos agentes. A Figura 33 exibe o alerta recebido que informa sobre o uso de *swap* em estado crítico de um agente.

[11] https://exchange.nagios.org/directory/Plugins/System-Metrics/CPU-Usage-and-Load/snmp_remote_load/details

[12] https://exchange.nagios.org/

Figura 33 – Facilidade em Interoperabilidade - E-mail Recebido Sobre Alerto do Agente (Nagios)

Fonte: Autoria própria (2022).

Com base nos testes, verifica-se que todos os requisitos exceto o de disponibilidade e avaliabilidade foram atendidos.

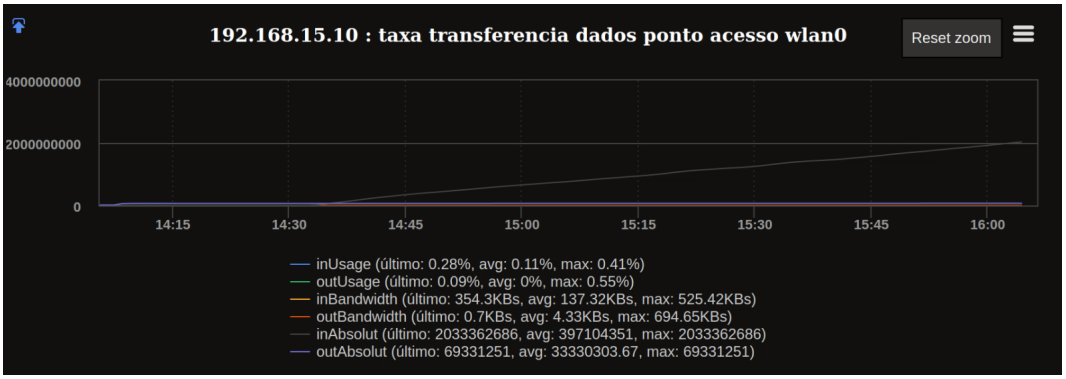
4.2.2 Caso de Teste 2: Gerenciamento Efetuado na Rede

Assim como no monitoramento efetuado nos agentes, também foi utilizado o template "Linux SNMP". Tendo em vista que o sistema operacional do ponto de acesso é baseado em Linux, pode-se utilizar os diversos *Object Identifier*s (OIDs) disponibilizados pelo sistema de forma nativa, sem ser necessário instalar um agente específico da ferramenta.

A ferramenta por padrão, não possui opção para verificar a largura de banda, consumo de banda e latência. Dessa maneira, foi realizada uma pesquisa por *plugins* relacionados no site de extensões da ferramenta. Foram utilizados os termos: "*bandwith*", "*bandwith usage*" e "*latency*". Nenhuma dessas apresentou resultado adequado, sendo assim, considera-se que esses requisitos não foram atendidos.

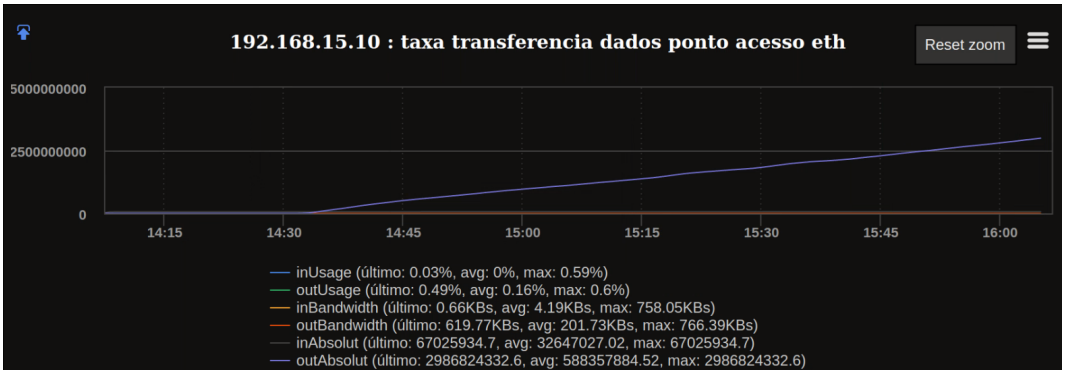
Para a taxa de transferência de dados, foi utilizado o *plugin* "check_iftraffic3" , permitindo obter a velocidade nas interfaces de rede (sem fio e cabeada) de maneira individual. A Figura 34 exibe a taxa de transferência em bits/s (eixo y) na interface sem fio em relação ao tempo (eixo x) enquanto que a Figura 35, exibe os dados relacionados à interface cabeada.

Figura 34 – Taxa de Transferência de Dados na Interface Sem Fio do Ponto de Acesso (Nagios)



Fonte: Autoria própria (2022).

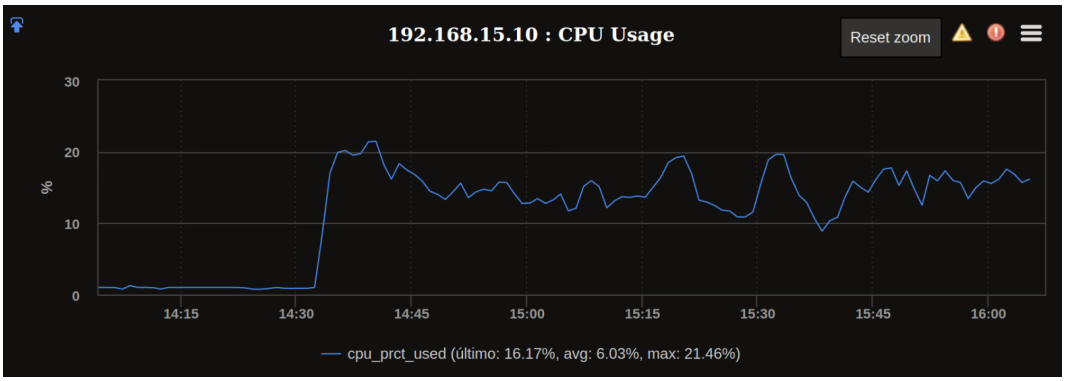
Figura 35 – Taxa de Transferência de Dados na Interface Cabeada do Ponto de Acesso (Nagios)



Fonte: Autoria própria (2022).

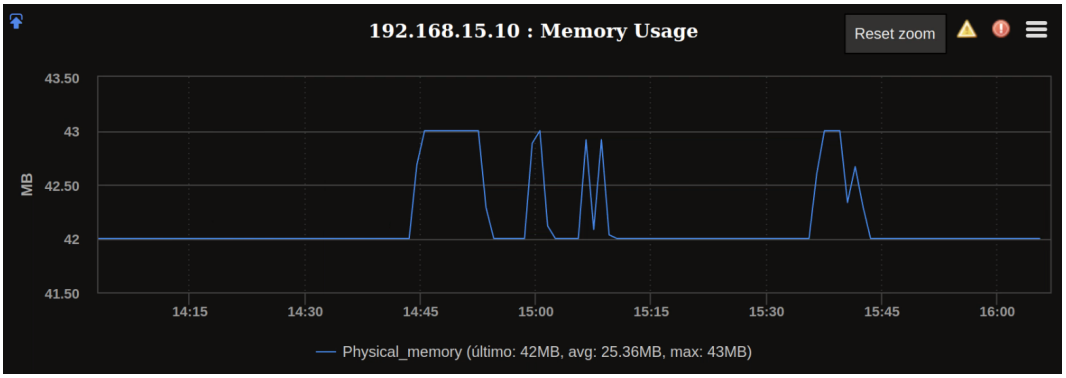
A utilização de recursos contempla CPU e memória. Para a CPU, foi possível obter a informação através do menu Serviços -> Uso de CPU. A Figura 36 detalha o uso de CPU (eixo y) em relação ao tempo (eixo x). Já para a memória, foi utilizado o menu Serviços -> Uso de memória. A Figura 37 relata o uso de memória RAM (eixo y) em relação ao tempo (eixo x).

Figura 36 – Uso de CPU do Ponto de Acesso (Nagios)



Fonte: Autoria própria (2022).

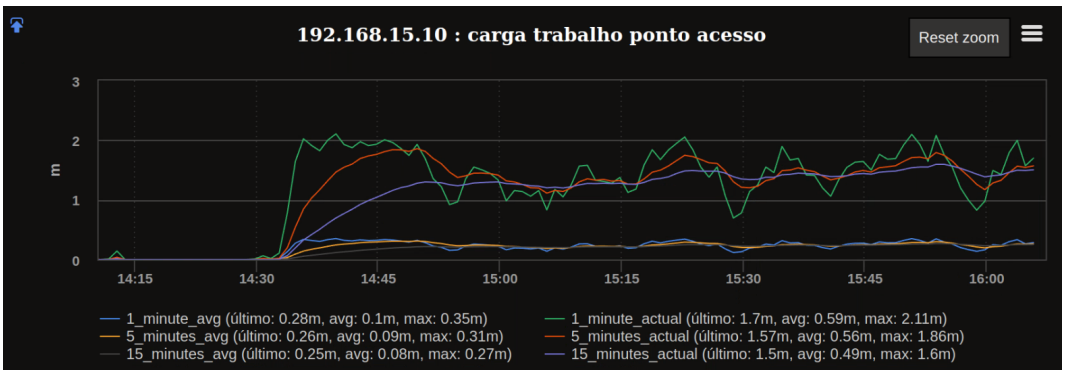
Figura 37 – Uso de Memória RAM do Ponto de Acesso (Nagios)



Fonte: Autoria própria (2022).

Para a carga de trabalho, foi utilizado o *plugin* "snmp_remote_load". A Figura 38 exibe o gráfico gerado, que detalha a carga de trabalho para 1 minuto, 5 minutos e 15 minutos.

Figura 38 – Carga de Trabalho do Ponto de Acesso (Nagios)



Fonte: Autoria própria (2022).

A disponibilidade e avaliabilidade não foi possível de ser monitorada pois a ferramenta não possui essa funcionalidade por padrão nem *plugins* para comportar esses dois requisitos.

Por padrão, a ferramenta não possui a funcionalidade de monitorar congestionamento de dados na rede. Dessa forma, foi realizada uma busca pelo termo "*network congestion*" no site de extensões. Tendo em vista que não foi exibido resultado, considera-se como requisito não atendido.

Para a facilidade em interoperabilidade, foi configurado o envio de avisos por e-mail para sinalizar sobre alertas ocorridos no ponto de acesso. A Figura 39 exibe o alerta de indisponibilidade do ponto de acesso que foi desligado propositalmente para avaliar esse requisito.

Figura 39 – Facilidade em Interoperabilidade - E-mail Recebido Sobre Alerta do Ponto de Acesso (Nagios)

Fonte: Autoria própria (2022).

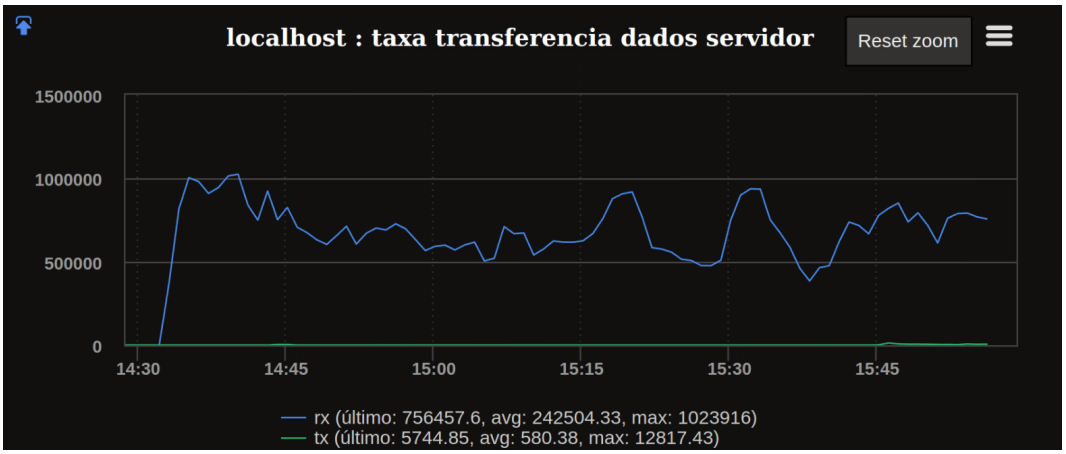
Pelos testes realizados, alguns requisitos não foram atendidos: largura de banda, consumo de banda, latência, disponibilidade e avaliabilidade e congestionamento de dados na rede.

4.2.3 Caso de Teste 3: Gerenciamento Efetuado no Servidor

Para o terceiro caso de teste, especificado no Quadro 30, as informações foram obtidas com o uso do template "linux-server" que engloba as métricas básicas a serem monitoradas em um servidor. Também foi necessário utilizar *plugins* para contemplar dados específicos.

A taxa de transferência de dados foi acompanhada através do *plugin* "check_eth"[13] o qual permite obter informações relativas ao uso das interfaces de rede em tempo real para servidores Linux. A Figura 40 exibe o gráfico gerado, no eixo x é exibido o tempo enquanto que no y, a transferência de dados em bits/s.

Figura 40 – Taxa de Transferência de Dados no Servidor (Nagios)



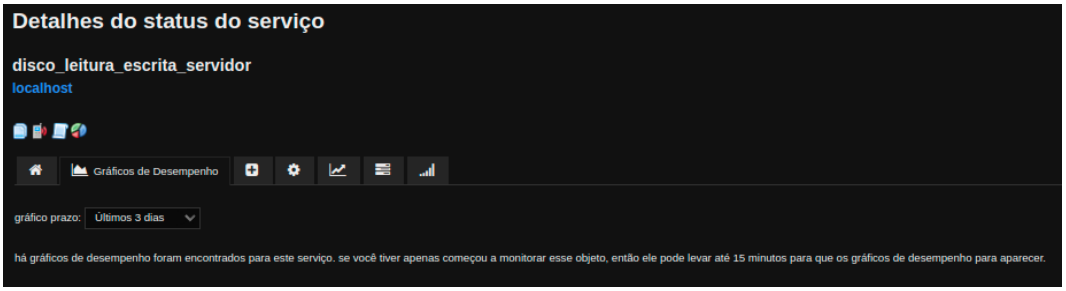
Fonte: Autoria própria (2022).

[13] https://exchange.nagios.org/directory/Plugins/Network-Connections%2C-Stats-and-Bandwidth/check_eth/details

A utilização de recursos voltados à CPU e memória foi monitorada de forma nativa, sem auxílio de *plugins*, já para averiguar o uso de disco (taxa de leitura e escrita), foi necessário utilizar o *plugin* check_iostst v1.1.0 [14] o qual coletou dados mas não gerou gráfico, conforme a Figura 41. O uso do espaço de disco em MB em relação ao tempo é obtido através do serviço "Partição Raíz" (Figura 42).

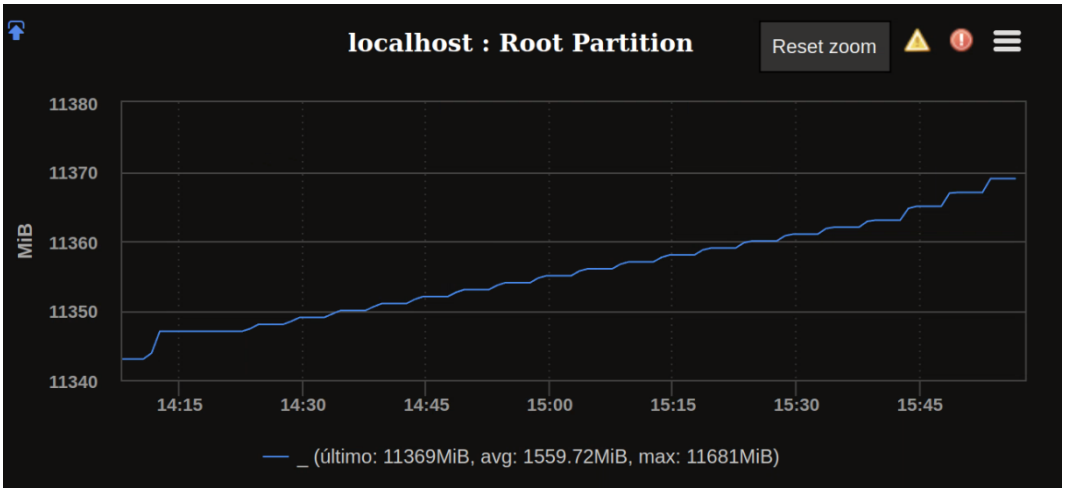
Para a CPU, observa-se o menu Painel, o qual exibe informações de uso pelo sistema, usuário, tempo livre, tempo em espera para leitura e escrita, etc. (Figura 43). A memória também é verificada pelo mesmo menu o qual possui o total disponível, a quantidade utilizada, livre, compartilhada, em *buffer* e em *cache* (Figura 44).

Figura 41 – Uso de Disco no Servidor (Nagios)



Fonte: Autoria própria (2022).

Figura 42 – Utilização de Espaço em Disco no Servidor (Nagios)



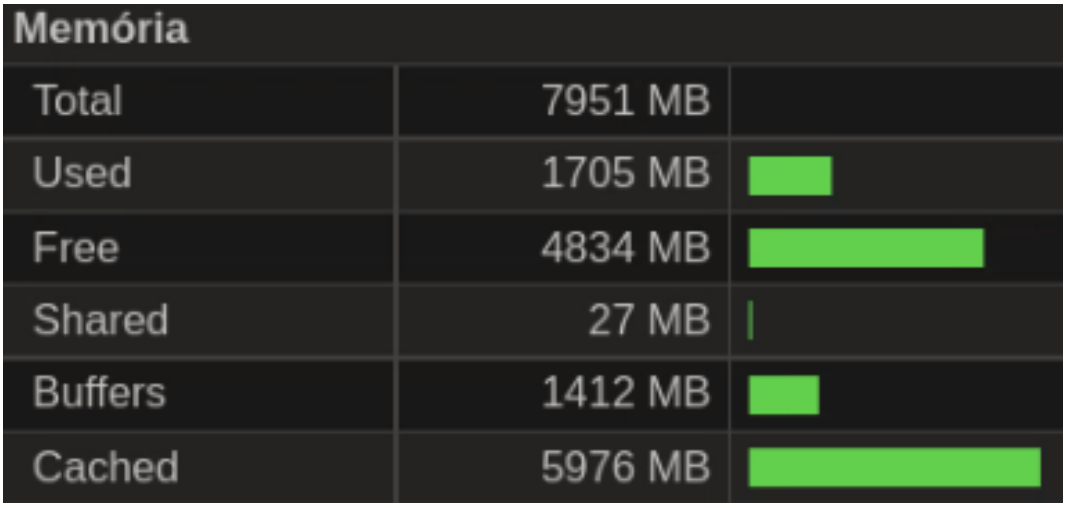
Fonte: Autoria própria (2022).

[14] https://exchange.nagios.org/directory/Plugins/Operating-Systems/Linux/check_iostat–2D-I-2FO-statistics–2D-updated-2016/details

Figura 43 – Uso de CPU no Servidor (Nagios)

Fonte: Autoria própria (2022).

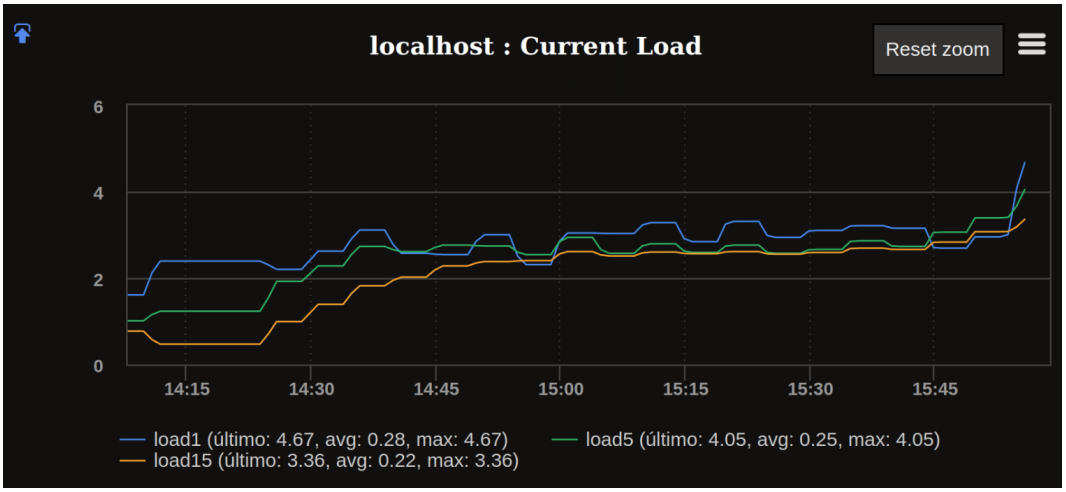
Figura 44 – Uso de Memória RAM no Servidor (Nagios)



Fonte: Autoria própria (2022).

A carga de trabalho é averiguada pelo serviço “Carga Atual”, o qual informa dados para 1 minuto, 5 minutos e 15 minutos (eixo y) ao longo do tempo (eixo x) (Figura 45).

Figura 45 – Carga de Trabalho no Servidor (Nagios)

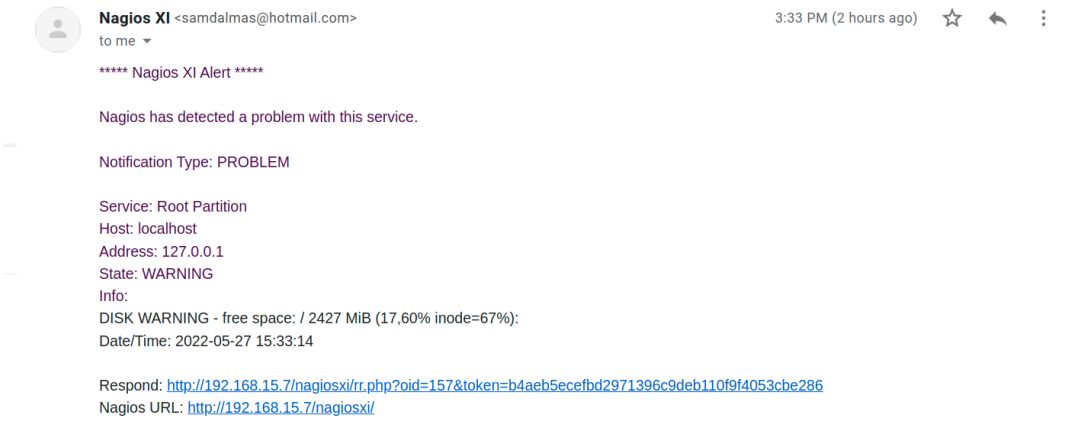


Fonte: Autoria própria (2022).

A disponibilidade e avaliabilidade para o servidor não foi possível de ser monitorada tendo em vista que a busca por *plugins* relacionados assim como para os agentes e rede, não retornou resultados adequados.

A facilidade em interoperabilidade é contemplada visto que pode-se configurar avisos através de e-mail para sinalizar sobre algum alerta ocorrido no servidor. A Figura 46 exibe um alerta de alto uso do espaço de disco.

Figura 46 – Facilidade em Interoperabilidade - E-mail Recebido Sobre Alerta do Servidor (Nagios)



Fonte: Autoria própria (2022).

De acordo com os testes realizados, verifica-se que foram atendidos em sua totalidade: taxa de transferência de dados, carga de trabalho e facilidade em interoperabilidade. O critério de utilização de recursos foi atendido de maneira parcial pois não gerou gráfico para a leitura e escrita de disco. Já o requisito de disponibilidade e avaliabilidade não foi atendido.

4.3 ZABBIX

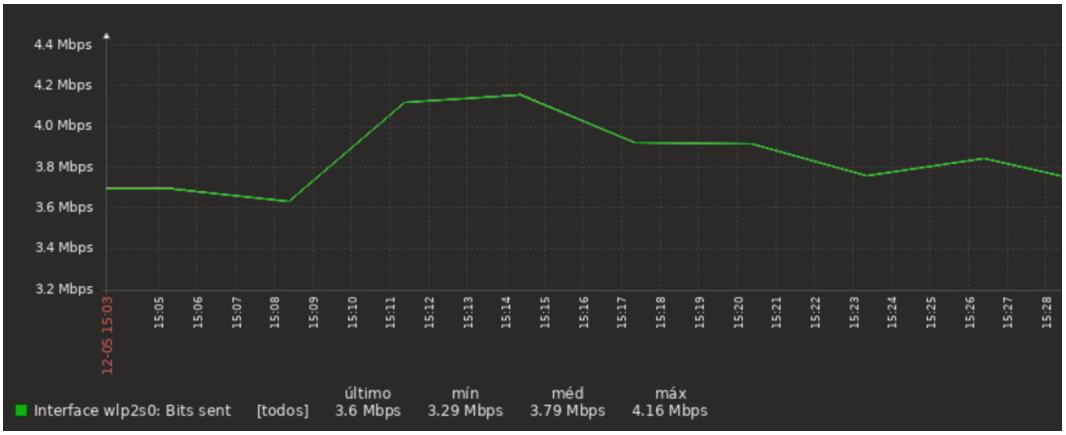
Para realizar a instalação do Zabbix no servidor e nos agentes, foram utilizadas as instruções contidas no site da ferramenta [15]. Foi possível realizar o monitoramento utilizando *templates* padrões já que esses possuem itens que englobam todos os requisitos necessários. O tempo de obtenção de dados foi mantido padrão para cada item dos *templates*.

4.3.1 Caso de Teste 1: Gerenciamento Efetuado nos Agentes

No primeiro caso de teste (Quadro 28) foi utilizado o *template "Linux by Zabbix agent"*, o qual provê um modelo de itens de monitoramento para máquinas com esse sistema operacional.

A taxa de transferência de dados foi monitorada pelos itens "Bits enviados" e "Bits recebidos" os quais informam a quantidade total de dados trafegados nas placas de rede dos agentes. A Figura 47 exibe a quantidade de dados enviados em *Megabit por segundo* (Mbps) no eixo y, enquanto que no eixo x, exibe o tempo em hora e minuto. Já a Figura 48 informa a no eixo y, a quantidade recebida em *Quilobit por segundo* (Kbps), no eixo x, também referencia a hora de coleta dos dados. Para os dois gráficos, são informados o último dado coletado, a mínima, a média e a máxima de transferência na placa de rede.

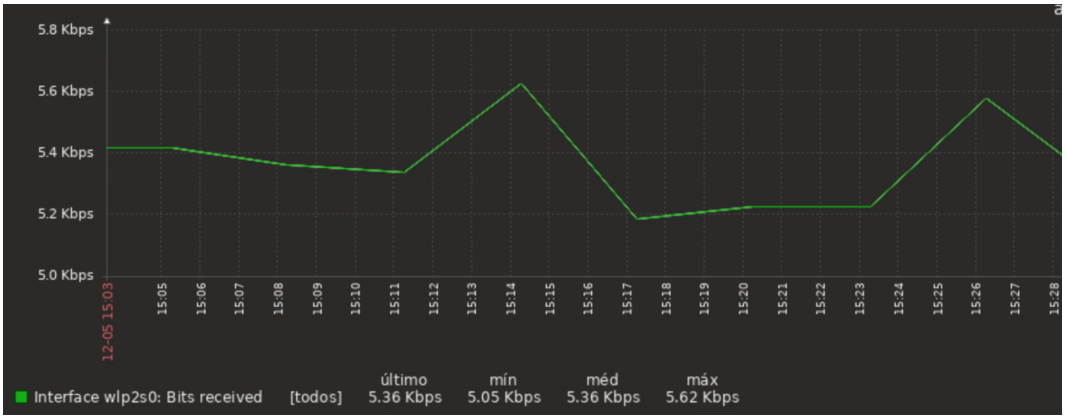
Figura 47 – Taxa de Transferência de Dados Enviados em Agentes (Zabbix)



Fonte: Autoria própria (2022).

[15] https://www.zabbix.com/download

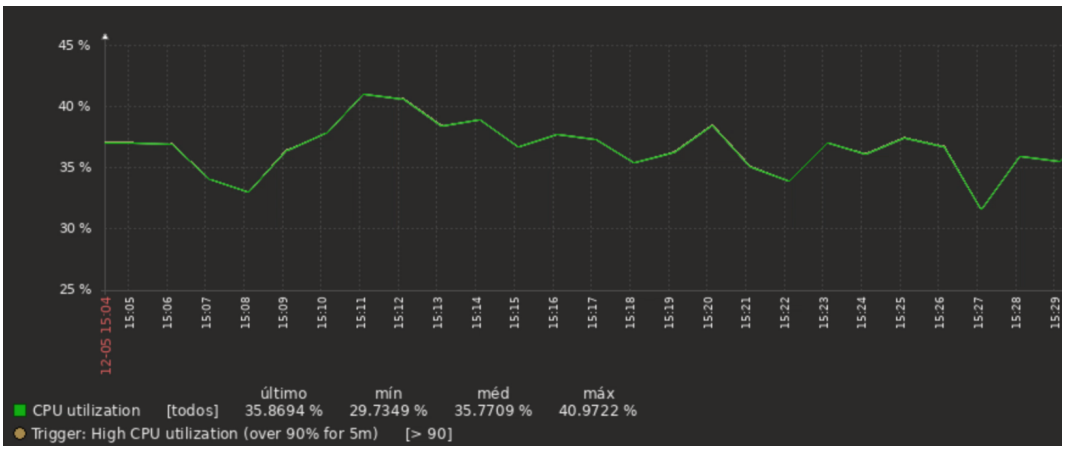
Figura 48 – Taxa de Transferência de Dados Recebidos em Agentes (Zabbix)



Fonte: Autoria própria (2022).

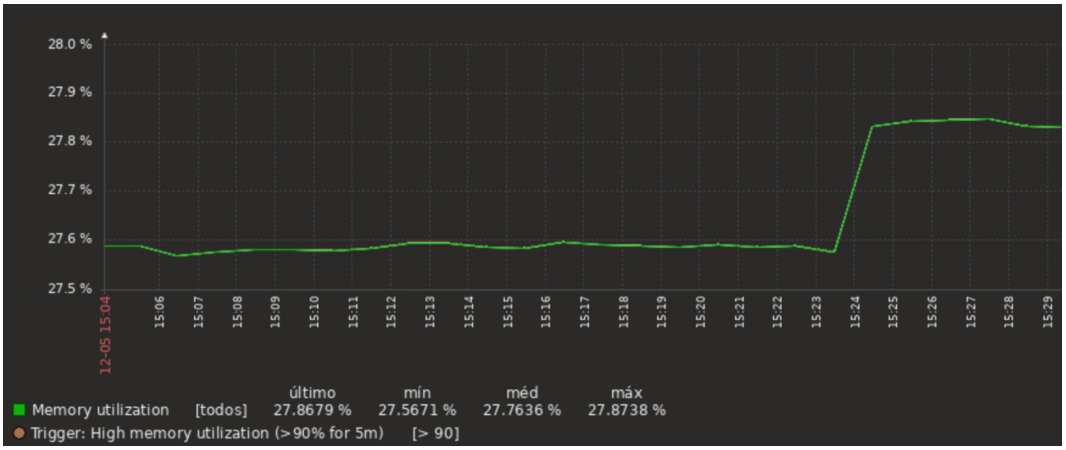
Para a utilização de recursos, foram utilizados os itens “Uso de CPU” e “Uso de memória”. A Figura 49 mostra a utilização de CPU no eixo y (em porcentagem) enquanto que no x, mostra a hora de coleta. O gráfico da Figura 50 informa a quantidade de memória RAM utilizada em porcentagem (eixo y) ao decorrer do tempo (eixo x). Também são exibidos o último dado coletado, a mínima, a média e a máxima para os dois recursos.

Figura 49 – Uso de CPU em Agentes (Zabbix)



Fonte: Autoria própria (2022).

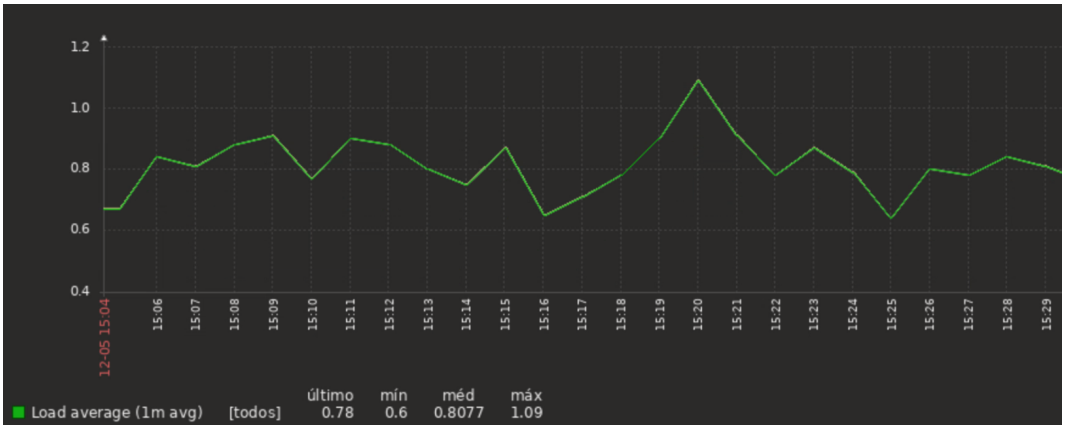
Figura 50 – Uso de Memória RAM em Agentes (Zabbix)



Fonte: Autoria própria (2022).

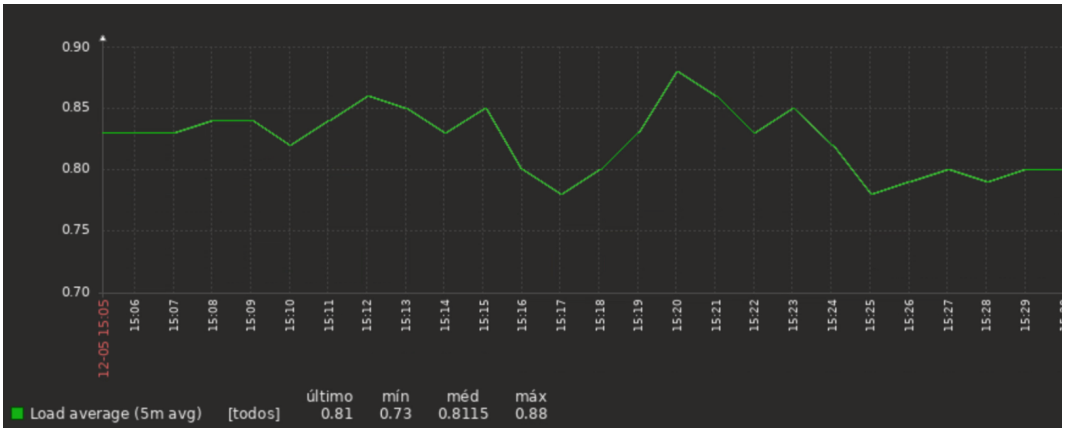
Para a carga de trabalho, foram utilizados os itens “Média de carga 1 minuto” (Figura 51), “Média de carga 5 minutos” (Figura 52) e “Média de carga 15 minutos” (Figura 53). Para esses três gráficos, no eixo y é exibida a média de carga enquanto que no eixo x, o tempo. Informações de último dado coletado, mínima, média e máxima também estão presentes.

Figura 51 – Carga de Trabalho de 1 Minuto em Agentes (Zabbix)



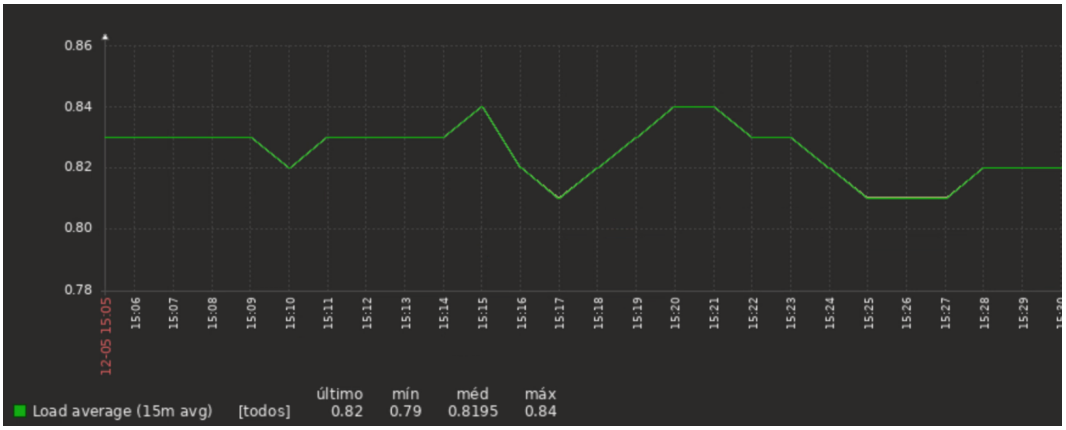
Fonte: Autoria própria (2022).

Figura 52 – Carga de Trabalho de 5 Minutos em Agentes (Zabbix)



Fonte: Autoria própria (2022).

Figura 53 – Carga de Trabalho de 15 Minutos em Agentes (Zabbix)

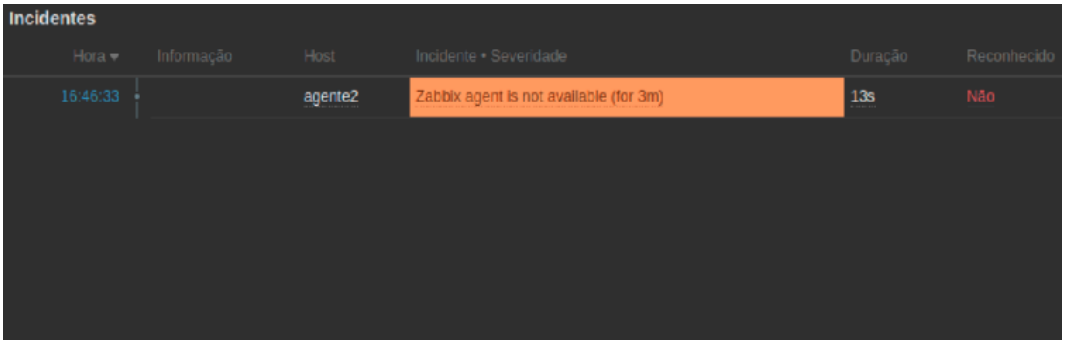


Fonte: Autoria própria (2022).

A disponibilidade e avaliabilidade não foi possível de ser monitorada pois não há itens no *template* que provenham as informações relacionadas.

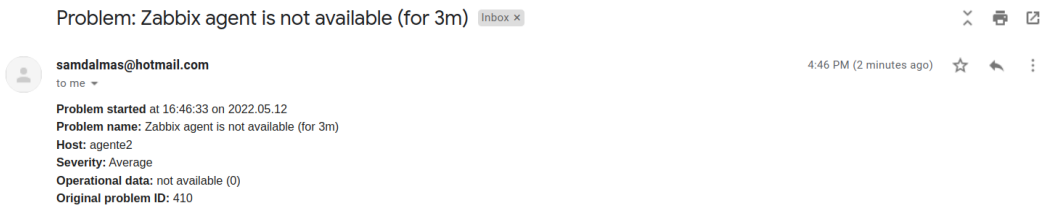
Já a facilidade em interoperabilidade foi testada a partir do envio de e-mails para informar sobre alertas ocorridos nos agentes. Para isso, foi finalizado o processo agente em uma das máquinas com o intuito de gerar um alerta que informa sobre a indisponibilidade a partir de três minutos (Figura 54). Como consequência, um e-mail é enviado para o administrador do Zabbix (Figura 55).

Figura 54 – Facilidade em Interoperabilidade - Alerta Gerado no Agente (Zabbix)



Fonte: Autoria própria (2022).

Figura 55 – Facilidade em Interoperabilidade - E-mail Recebido Sobre Indisponibilidade do Agente (Zabbix)



Fonte: Autoria própria (2022).

Com base nos testes realizados para os agentes do Zabbix, todos os requisitos foram contemplados exceto a disponibilidade e avaliabilidade, utilizando o *template* padrão para agentes Linux.

4.3.2 Caso de Teste 2: Gerenciamento Efetuado na Rede

Para o caso de teste do Quadro 29, foi utilizado o *template "Linux SNMP"* que disponibiliza os itens necessários para ser realizado o monitoramento do dispositivo de rede (ponto de acesso).

Para a largura de banda, a ferramenta informa somente o valor relacionado à interface de rede cabeada do ponto de acesso e não em relação à rede como um todo. Dessa forma, constata-se que não atende o requisito em questão.

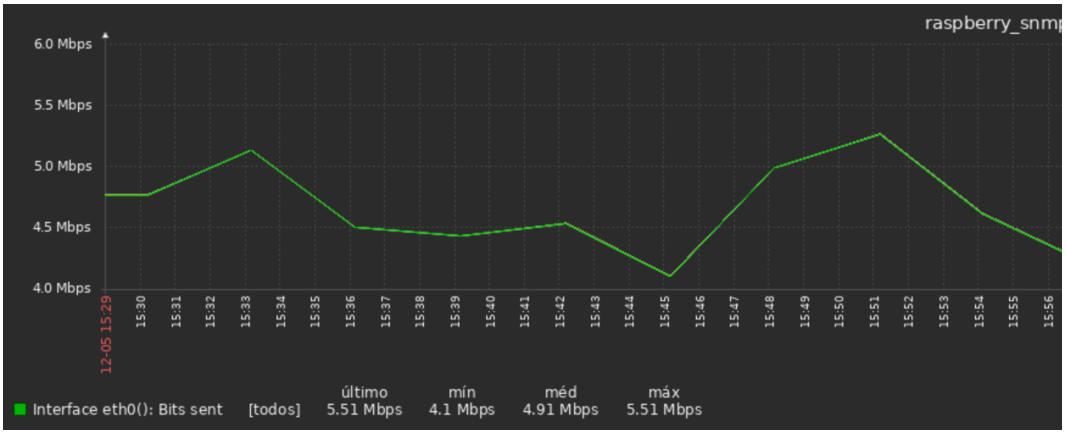
O consumo de banda também não é possível de ser monitorado tendo em vista que é dependente do requisito de largura de banda, o qual a ferramenta não suporta.

Para o requisito de latência, o monitoramento não foi realizado pois não há itens no *template* que englobem o seu funcionamento.

A taxa de transferência de dados para a interface cabeada (eth0) e sem fio (wlan0) foi averiguada a partir dos itens "Interface eth0: Bits enviados", "Interface eth0: Bits recebidos",

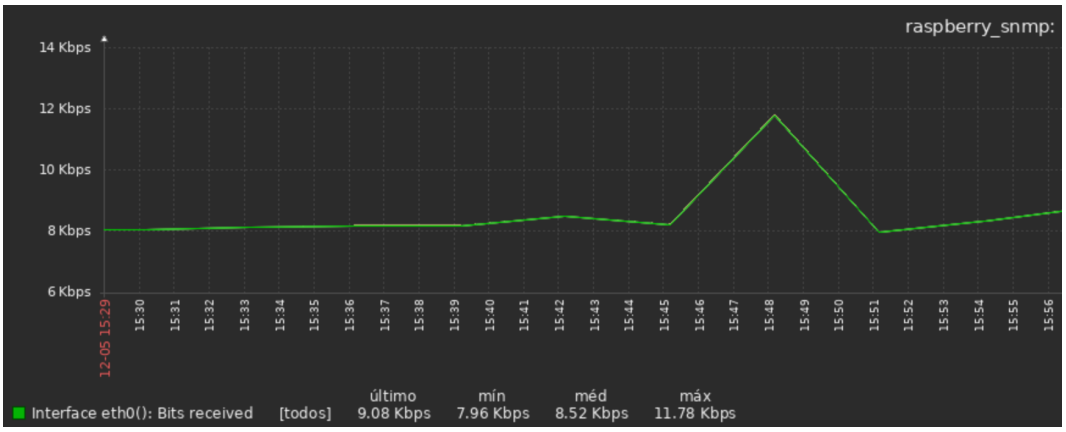
“Interface wlan0: Bits enviados” e “Interface wlan0: Bits recebidos”, permitindo consultar as informações relativas às interfaces de maneira individual. Dessa forma, quatro gráficos foram gerados: a Figura 56 expressa a quantidade de dados enviados, em Mbps em relação ao tempo na interface cabeada. A Figura 57 informa a quantidade de dados recebidos (em Kbps) em relação ao tempo na interface cabeada. A Figura 58 relata a quantidade de dados enviados (em Kbps) em relação ao tempo na interface sem fio. Por fim, a Figura 59 exibe a quantidade de dados recebidos (em Mbps) em relação ao tempo na interface sem fio.

Figura 56 – Taxa de Transferência de Dados Enviados na Interface Cabeada do Ponto de Acesso (Zabbix)



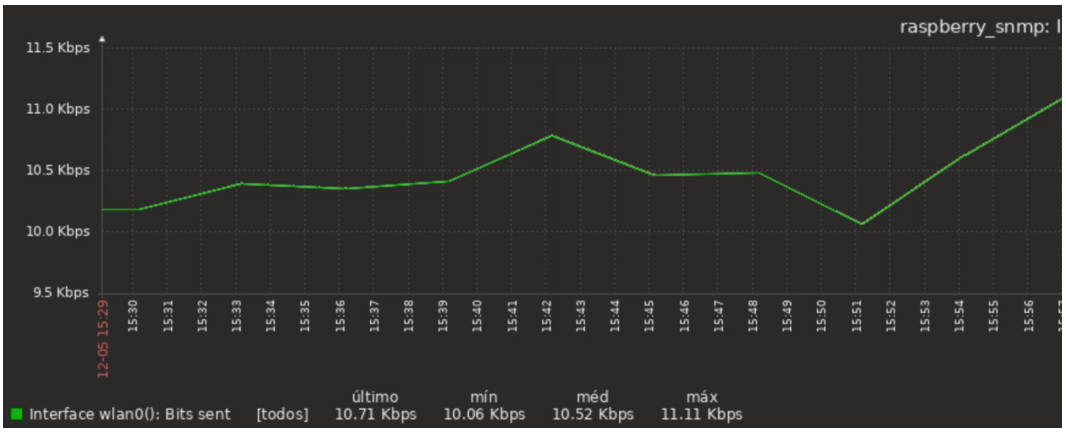
Fonte: Autoria própria (2022).

Figura 57 – Taxa de Transferência de Dados Recebidos na Interface Cabeada do Ponto de Acesso (Zabbix)



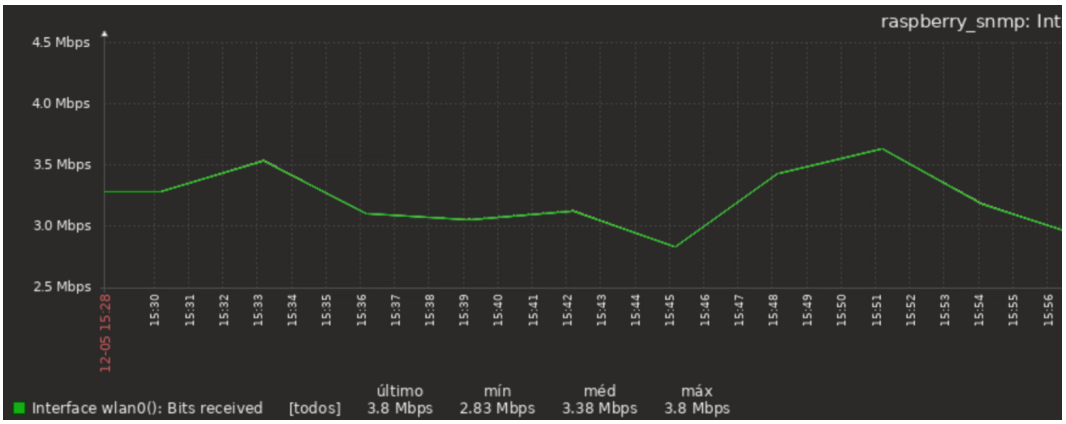
Fonte: Autoria própria (2022).

Figura 58 – Taxa de Transferência de Dados Enviados na Interface Sem Fio do Ponto de Acesso (Zabbix)



Fonte: Autoria própria (2022).

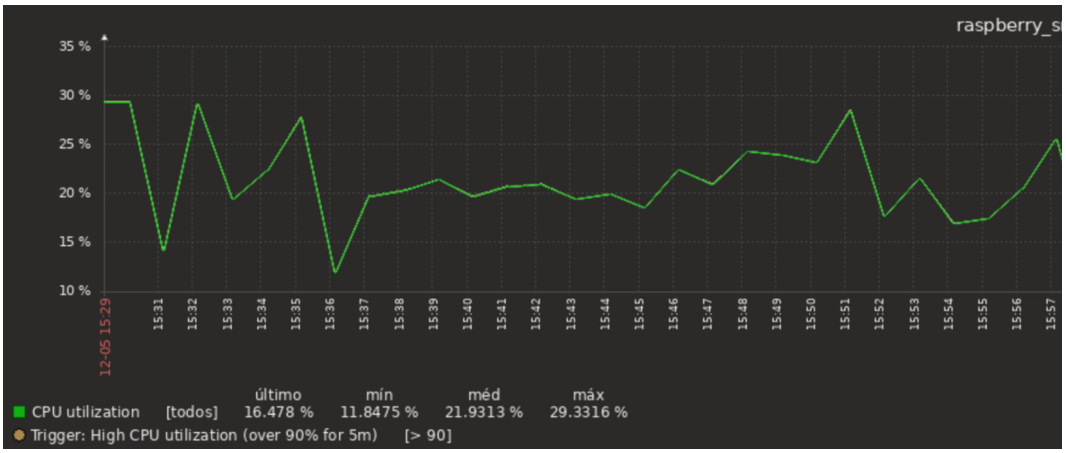
Figura 59 – Taxa de Transferência de Dados Recebidos na Interface Sem Fio do Ponto de Acesso (Zabbix)



Fonte: Autoria própria (2022).

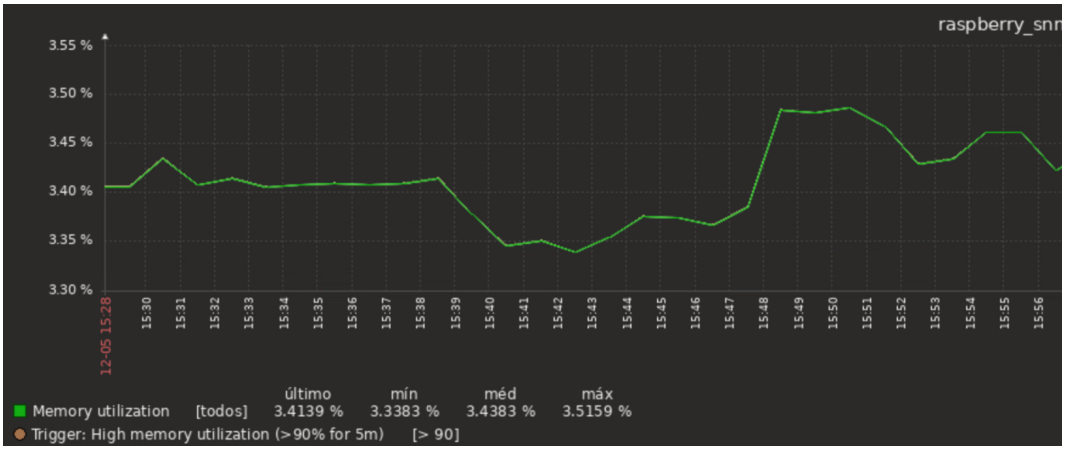
A utilização de recursos foi monitorada pelos itens “Uso de CPU” e “Uso de memória” os quais provêm informações acerca do hardware do ponto de acesso. A Figura 60 expressa o uso de CPU do ponto de acesso em porcentagem (eixo y) em relação ao tempo (eixo x). Já a Figura 61 informa a utilização de memória RAM no ponto de acesso (eixo y) ao decorrer do tempo (eixo x).

Figura 60 – Uso de CPU no Ponto de Acesso (Zabbix)



Fonte: Autoria própria (2022).

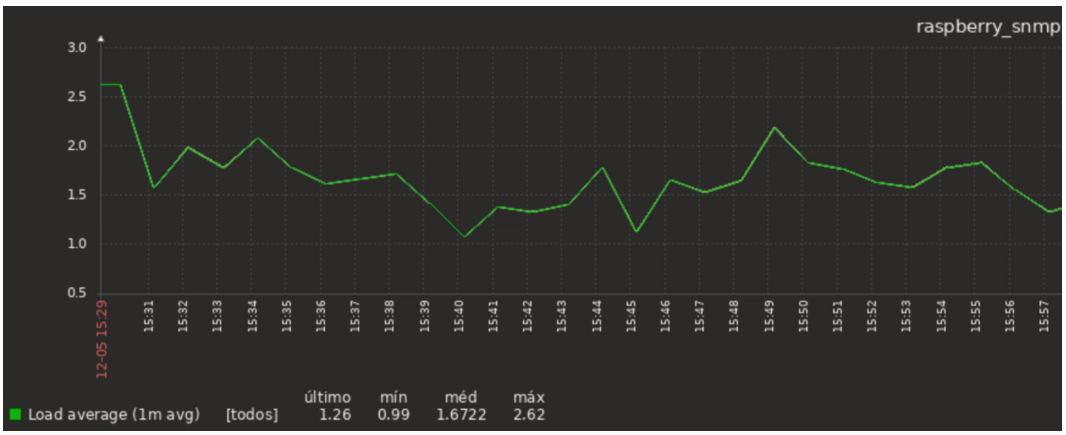
Figura 61 – Uso de RAM no Ponto de Acesso (Zabbix)



Fonte: Autoria própria (2022).

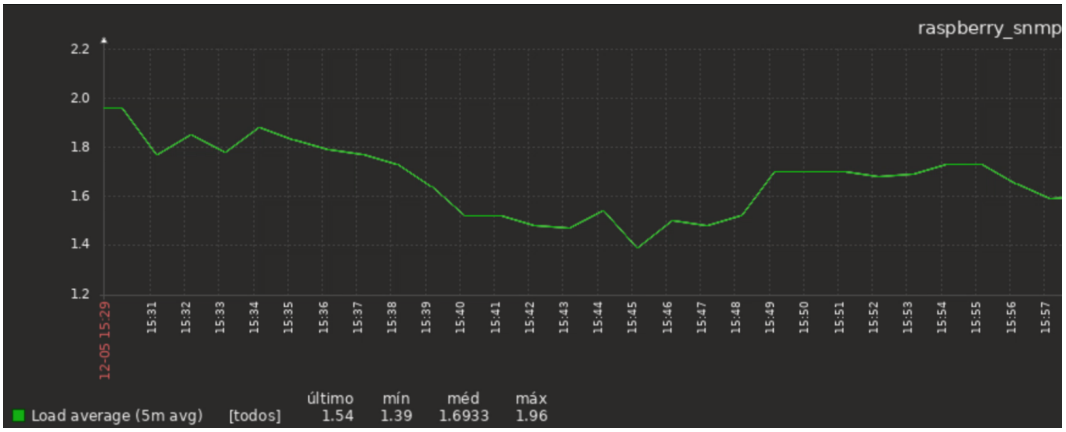
Para a carga de trabalho, foram utilizados os itens “Média de carga 1 minuto” (Figura 62), “Média de carga 5 minutos” (Figura 63) e “Média de carga 15 minutos” (Figura 64). Esses três gráficos informam no eixo y, a média de carga enquanto que no eixo x, é expressado o tempo de coleta do dado.

Figura 62 – Carga de Trabalho de 1 Minuto no Ponto de Acesso (Zabbix)



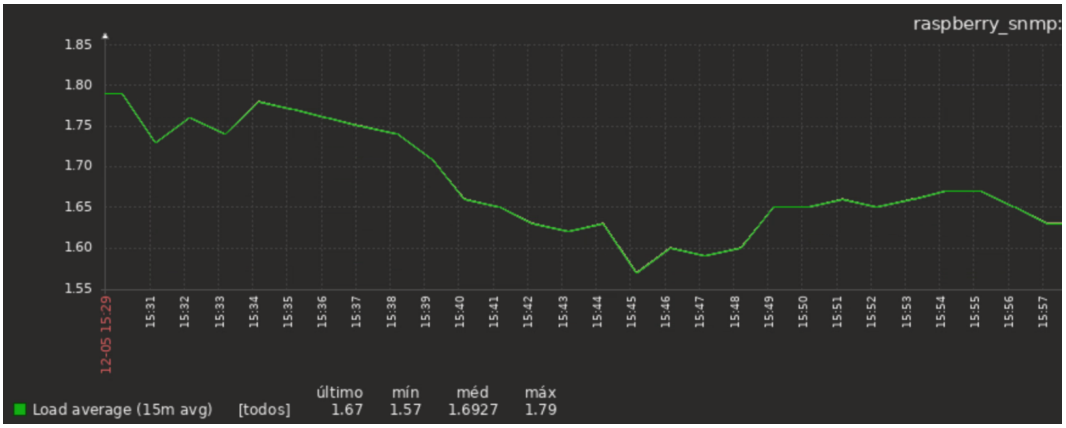
Fonte: Autoria própria (2022).

Figura 63 – Carga de Trabalho de 5 Minutos no Ponto de Acesso (Zabbix)



Fonte: Autoria própria (2022).

Figura 64 – Carga de Trabalho de 15 Minutos no Ponto de Acesso (Zabbix)



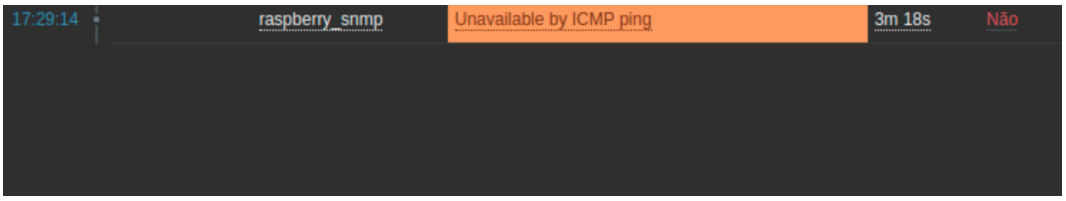
Fonte: Autoria própria (2022).

O requisito de disponibilidade e avaliabilidade não foi monitorado, pois assim como para os agentes, não há item relacionado no *template*.

Para mensurar o congestionamento de dados na rede, os indicativos que podem expressar sua ocorrência de forma parcial através da detecção de sobrecarga na comunicação de dados são: a quantidade de erros e pacotes descartados. Os itens relacionados no Zabbix são: “Pacotes de entrada descartados”, “Pacotes de entrada com erros”, “Pacotes de saída descartados” e “Pacotes de saída com erros” de cada interface de rede. Tendo em vista que tratam-se apenas de indicativos, ou seja, não é um fator determinante para mensurar o congestionamento, esse requisito é considerado como não atendido.

Para a facilidade em interoperabilidade, foi configurado o envio de avisos por e-mail para sinalizar sobre alertas ocorridos no equipamento de rede. Com o intuito de testar esse requisito, foi desligado o ponto de acesso, o que gerou um alerta no Zabbix (Figura 65). Como consequência desse alerta, foi encaminhado um e-mail para o administrador da ferramenta (Figura 66).

Figura 65 – Facilidade em Interoperabilidade - Alerta Gerado Sobre o Ponto de Acesso (Zabbix)



Fonte: Autoria própria (2022).

Figura 66 – Facilidade em Interoperabilidade - E-mail Recebido Sobre Indisponibilidade do Ponto de Acesso (Zabbix)

Fonte: Autoria própria (2022).

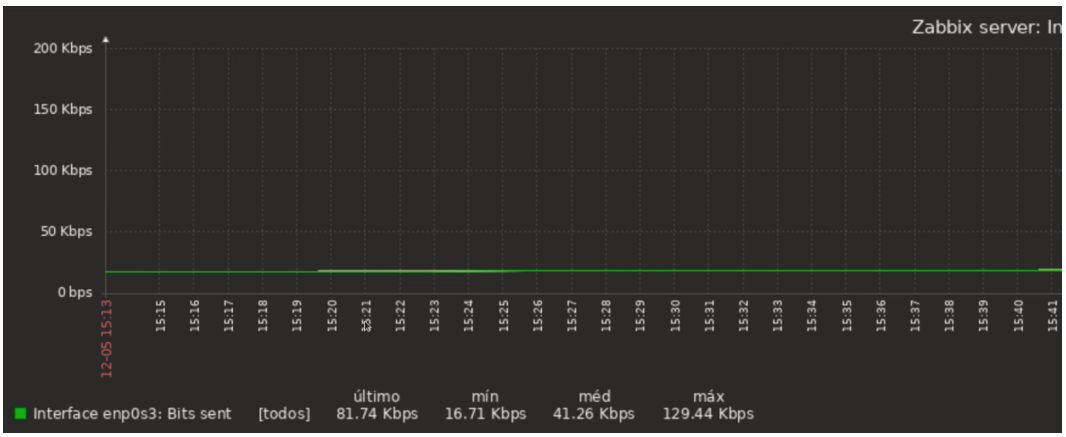
De acordo com os testes realizados, foi possível constatar que a maioria dos requisitos não foram atendidos: largura de banda, consumo de banda, latência, disponibilidade e avaliabilidade e congestionamento de dados na rede. Os demais, foram contemplados em sua totalidade.

4.3.3 Caso de Teste 3: Gerenciamento Efetuado no Servidor

No caso de teste do Quadro 30, foram utilizados os *templates "Linux by Zabbix agent"* e *"Zabbix server health"* que disponibilizam os itens necessários para monitorar servidores Linux. Tendo em vista que o servidor utiliza o mesmo sistema operacional dos agentes, os dados, em sua grande maioria, podem ser obtidos através dos mesmos itens disponibilizados pela ferramenta.

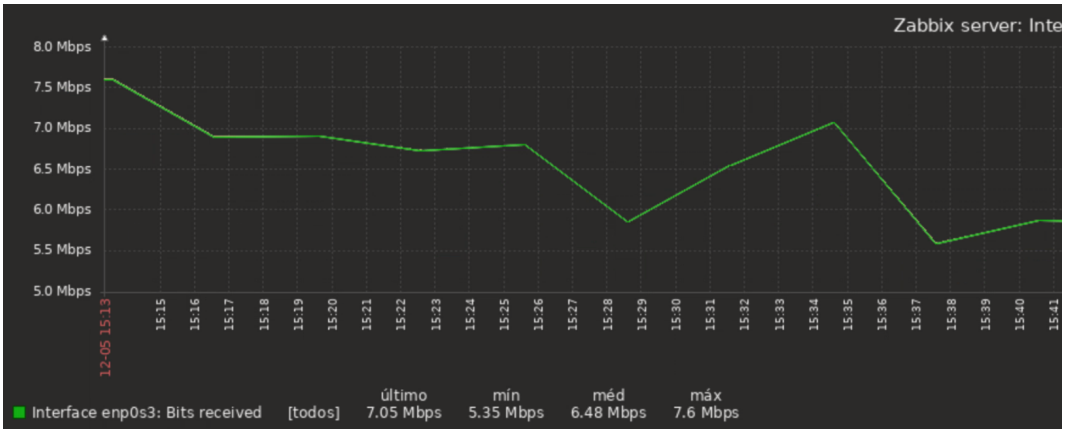
Para a taxa de transferência de dados, foram utilizados os itens "Bits enviados" e "Bits recebidos" que permitem acompanhar o tráfego na placa de rede do servidor. A Figura 67 exibe a quantidade de dados enviados em Kbps na placa de rede (eixo y) ao decorrer do tempo, enquanto que a Figura 68 detalha a taxa de transferência de recebimento em Mbps (eixo y) ao decorrer do tempo.

Figura 67 – Taxa de Transferência de Dados Enviados no Servidor (Zabbix)



Fonte: Autoria própria (2022).

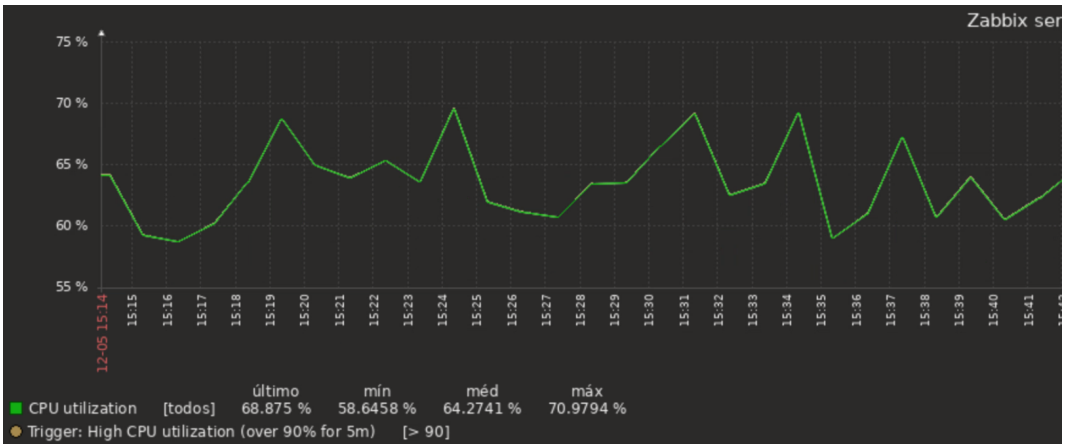
Figura 68 – Taxa de Transferência de Dados Recebidos no Servidor (Zabbix)



Fonte: Autoria própria (2022).

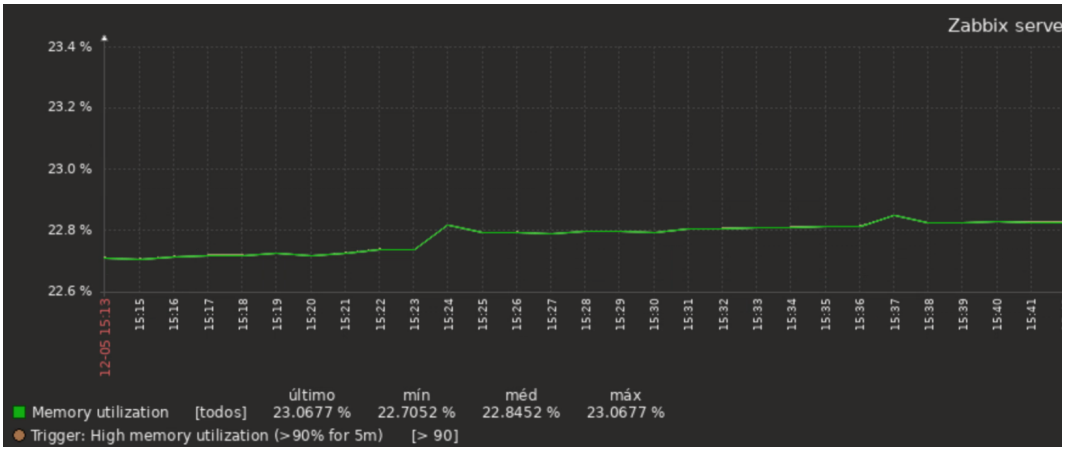
Na utilização de recursos, para avaliar o uso de CPU, memória e disco, foram utilizados os itens “Uso de CPU”, “Uso de memória”, “sda: Uso de disco” e “/: Espaço utilizado”. A Figura 69 exibe o uso de CPU em porcentagem ao decorrer do tempo. A Figura 70 exibe o uso de memória em porcentagem ao decorrer do tempo. A Figura 71 detalha o uso de disco em porcentagem ao longo do tempo, por fim a Figura 72 referencia a quantidade de espaço utilizado em disco (em *gigabytes*) em relação ao tempo.

Figura 69 – Uso de CPU no Servidor (Zabbix)



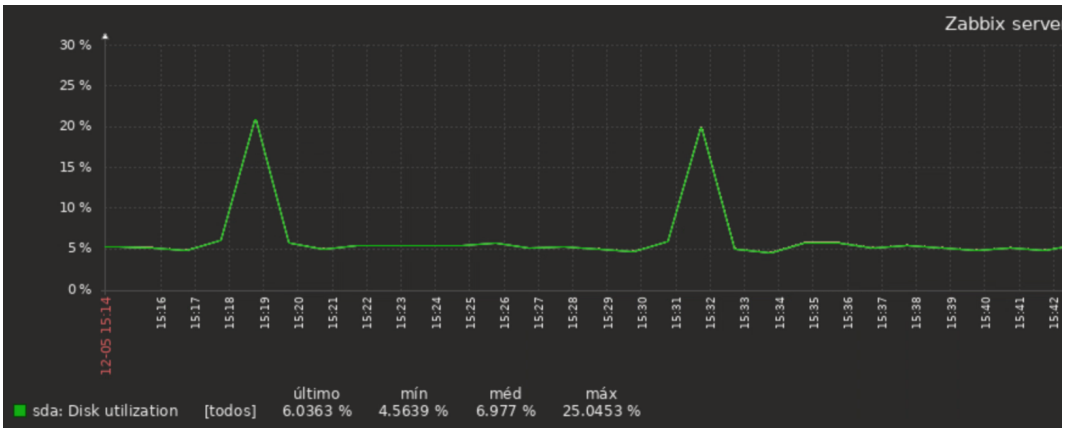
Fonte: Autoria própria (2022).

Figura 70 – Uso de Memória RAM no Servidor (Zabbix)



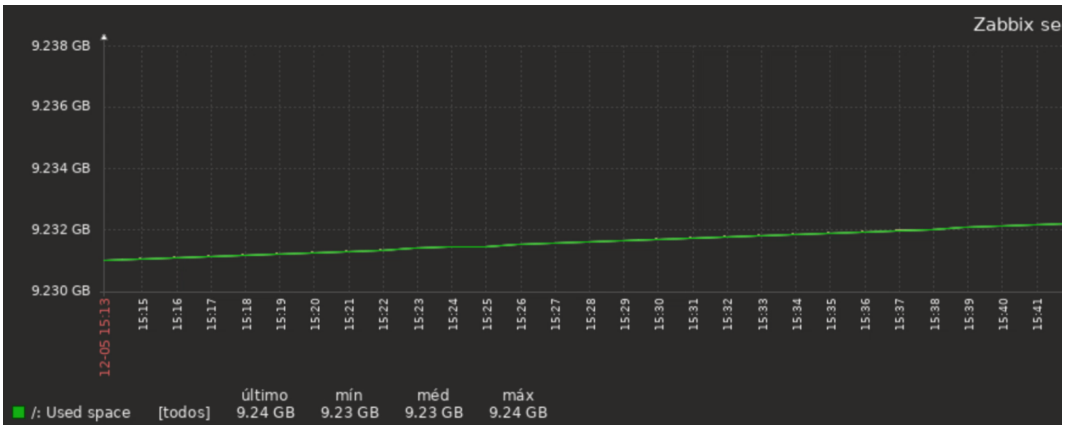
Fonte: Autoria própria (2022).

Figura 71 – Uso de Disco no Servidor (Zabbix)



Fonte: Autoria própria (2022).

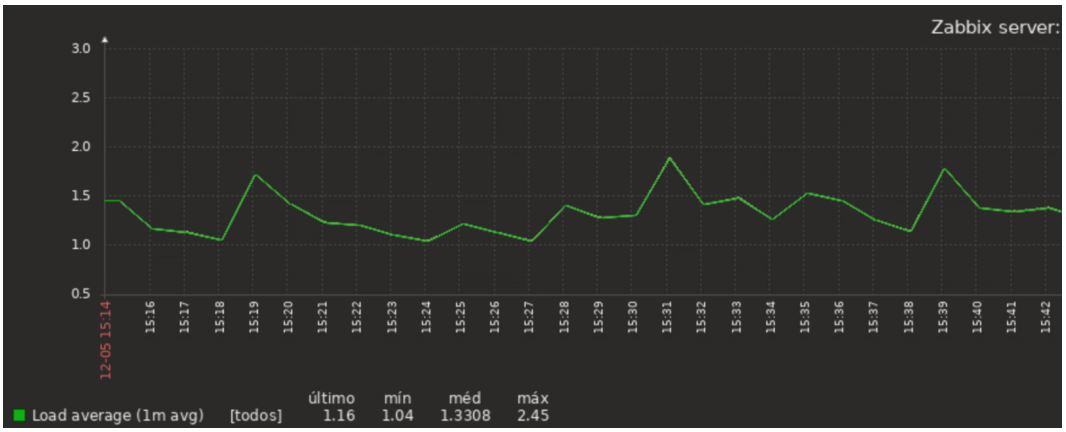
Figura 72 – Utilização de Espaço em Disco no Servidor (Zabbix)



Fonte: Autoria própria (2022).

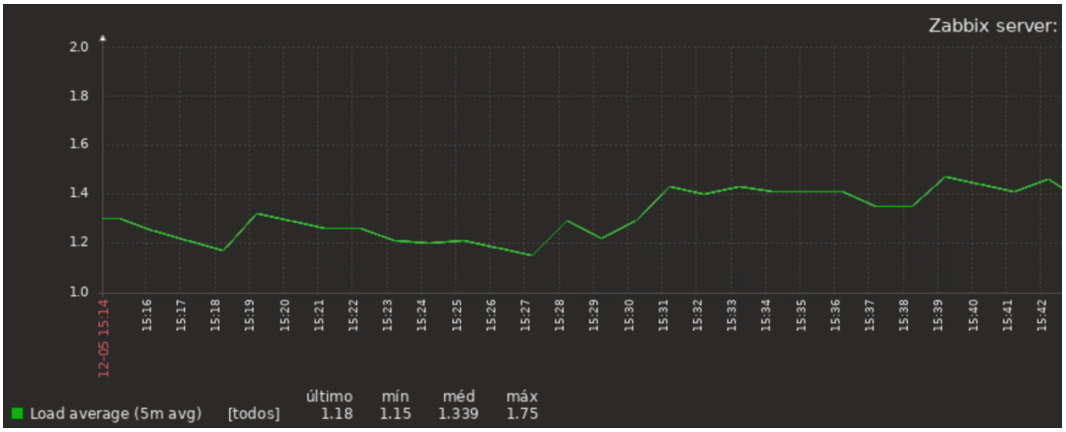
A carga de trabalho foi testada com base nos itens “Média de carga 1 minuto” (Figura 73), “Média de carga 5 minutos” (Figura 74) e “Média de carga 15 minutos” (Figura 75). Para os três gráficos, no eixo y, são exibidas a média de carga ao longo do tempo.

Figura 73 – Carga de Trabalho de 1 Minuto no Servidor (Zabbix)



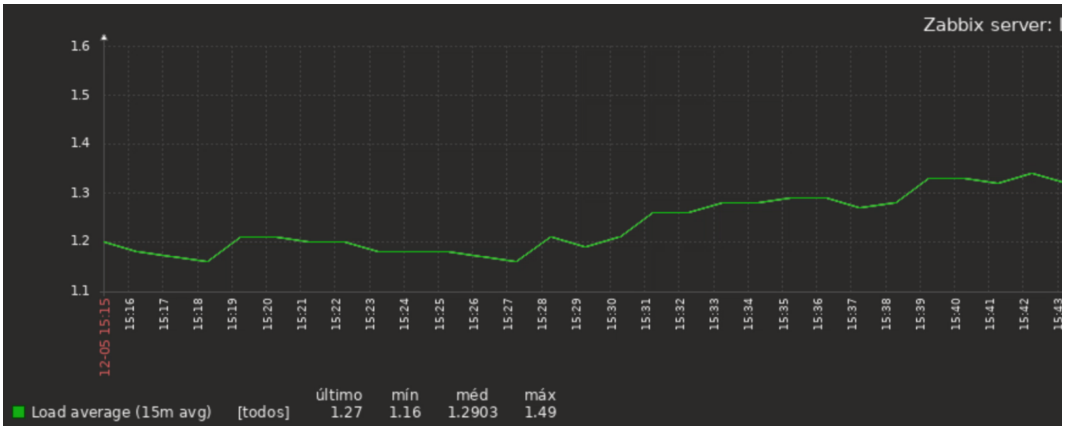
Fonte: Autoria própria (2022).

Figura 74 – Carga de Trabalho de 5 Minutos no Servidor (Zabbix)



Fonte: Autoria própria (2022).

Figura 75 – Carga de Trabalho de 15 Minutos no Servidor (Zabbix)

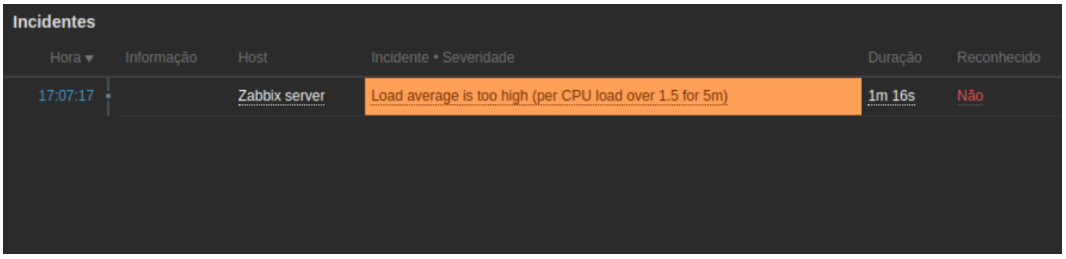


Fonte: Autoria própria (2022).

O requisito de disponibilidade e avaliabilidade não foi monitorado, pois assim como para os agentes e para a rede, não há item relacionado no *template*.

A facilidade em interoperabilidade foi testada a partir do envio de e-mails para informar sobre alertas ocorridos no servidor. A Figura 76 exibe o alerta de alto uso de CPU durante período de 5 minutos no servidor. Para aumentar a carga de uso, foi utilizado a ferramenta stress-ng [16]. Com isso, foi encaminhado um e-mail para o gerente da ferramenta informando sobre o ocorrido (Figura 77).

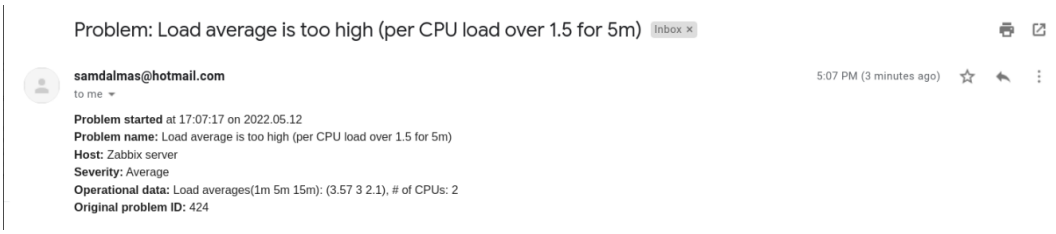
Figura 76 – Facilidade em Interoperabilidade - Alerta Gerado no Servidor (Zabbix)



Fonte: Autoria própria (2022).

[16] https://wiki.ubuntu.com/Kernel/Reference/stress-ng

Figura 77 – Facilidade em Interoperabilidade - E-mail Recebido Sobre Alta Carga de Trabalho no Servidor (Zabbix)



Fonte: Autoria própria (2022).

De acordo com os resultados obtidos nos testes, todos os requisitos exceto disponibilidade e avaliabilidade foram cumpridos para o servidor com a utilização dos *templates "Linux by Zabbix agent"* e *"Zabbix server health"*.

4.4 CONSIDERAÇÕES FINAIS

O capítulo aplica os casos de teste (Quadro 28 ao Quadro 30) nas ferramentas Ganglia, Nagios e Zabbix. Em relação ao uso dessas ferramentas, o Ganglia conta com pouca documentação e manuais disponíveis de forma a limitar sua utilização. Outro fator limitante é que essa ferramenta é voltada para *clusters* computacionais, o que difere do ambiente explorado nesse trabalho. O Nagios possui uma documentação abrangente, com diversos tutoriais disponíveis, além de contar com um site que disponibiliza diversos *plugins* para ampliar as suas funcionalidades. O Zabbix também possui boa documentação e conta com diversas funções que são instaladas junto com a ferramenta, o que facilita a criação de um ambiente de monitoramento.

Percebe-se que, de forma geral, diversos dos requisitos necessários ao monitoramento de um arquitetura de Névoa não foram atendidos pelas ferramentas, devido a fatores como falta de recursos, documentação e pelo propósito de uso dessas ferramentas estarem voltados a ambientes computacionais diferentes do explorado nesse trabalho.

5 AVALIAÇÃO DAS FERRAMENTAS

Para realizar a avaliação das ferramentas, foi utilizada a proposta de solução apresentada no Capítulo 3. Duas subcaracterísticas de funcionalidade foram escolhidas. A subcaracterística adequação permite averiguar se os *softwares* são aptos a realizar tarefas especificadas. Neste trabalho, essas tarefas referem-se aos requisitos de monitoramento para os agentes, rede e servidor. A subcaracterística interoperabilidade abrange aspectos de troca de dados com outros *softwares*. Foram escolhidas integrações com ferramentas que realizam notificações de alertas (e-mail, SMS, WhatsApp, Telegram e Slack). Foi atribuído peso de 90% para a adequação e 10% para a interoperabilidade.

Para avaliar as subcaracterísticas, foram adotadas métricas externas (Quadro 31 e Quadro 32). Esse tipo de métrica prevê executar a ferramenta e coletar dados no ambiente de execução. Para a adequação, foi selecionada a métrica de adequação funcional enquanto que para a interoperabilidade, foi selecionada a métrica interoperabilidade disponível.

Quadro 31 – Métrica Adequação Funcional

Característica	Funcionalidade
Subcaracterística	Adequação
Métrica	Adequação Funcional
Propósito da métrica	Quão adequadas são as funções avaliadas?
Método de aplicação	Número de funções que são adequadas para executar as tarefas especificadas em comparação com o número de funções avaliadas
Medida e Fórmula	X = 1-(A/B). A = Número de funções nas quais os problemas são detectados na avaliação. B = Número de funções avaliadas
Interpretação	0 <= X <= 1, quanto mais próximo de 1, mais adequado
Tipo de escala	Absoluta
Tipo de medida	Quantitativa

Fonte: Autoria própria (2022).

Quadro 32 – Métrica Interoperabilidade Disponível

Característica	Funcionalidade
Subcaracterística	Interoperabilidade
Métrica	Interoperabilidade disponível
Propósito da métrica	Com que frequência o usuário encontra restrições ou falhas inesperadas ao trocar dados entre o software avaliado e demais softwares?
Método de aplicação	Use o software avaliado em conjunto com outro software que permita a troca de dados entre eles
Medida e Fórmula	X = 1-(A/B). A = Número de softwares que apresentaram falhas ao trocar dados com o software avaliado. B = Número de softwares disponíveis
Interpretação	0 <= X <= 1, quanto mais próximo de 1, melhor
Tipo de escala	Absoluta
Tipo de medida	Quantitativa

Fonte: Autoria própria (2022).

5.1 ADEQUAÇÃO

Essa avaliação possui o propósito de verificar o quão adequadas são as funções avaliadas. Para isso, são relacionadas as funções consideradas como adequadas em relação ao número total de funções. Nenhuma das ferramentas atendeu a todos os requisitos.

Analisando os requisitos necessários para os agentes, verificou-se que a taxa de transferência de dados, utilização de recursos e carga de trabalho foram atendidos pelos três softwares. Os requisitos de disponibilidade e avaliabilidade não foram atendidos por nenhum (Quadro 33).

Quadro 33 – Requisitos de Agentes Atendidos pelas Ferramentas

Requisito	Ganglia	Nagios	Zabbix
Taxa de transferência de dados	**X**	**X**	**X**
Utilização de recursos	**X**	**X**	**X**
Carga de trabalho	**X**	**X**	**X**
Disponibilidade e avaliabilidade			

Fonte: Autoria própria (2022).

Para o servidor, a taxa de transferência de dados e a carga de trabalho foram atendidas pelas três ferramentas. A utilização de recursos foi atendida totalmente pelo Zabbix enquanto que o Ganglia e o Nagios atenderam de forma parcial tendo em vista que não contemplaram o monitoramento de todos os recursos (Quadro 34).

Por fim, para a rede, o Ganglia não apresentou resultados pois não foi possível instalar o *software* agente no ponto de acesso, tendo em vista a distribuição Linux em uso (OpenWrt). O Nagios e o Zabbix apresentaram o mesmo resultado, sendo possível monitorar taxa de

Quadro 34 – Requisitos do Servidor Atendidos pelas Ferramentas

Requisito	Ganglia	Nagios	Zabbix
Taxa de transferência de dados	**X**	**X**	**X**
Utilização de recursos	**P**	**P**	**X**
Carga de trabalho	**X**	**X**	**X**
Disponibilidade e avaliabilidade			

P= Atende parcialmente
Fonte: Autoria própria (2022).

transferência de dados, utilização de recursos e carga de trabalho. Os demais itens não foram contemplados (Quadro 35).

Quadro 35 – Requisitos da Rede Atendidos pelas Ferramentas

Requisito	Ganglia	Nagios	Zabbix
Largura de banda			
Consumo de banda			
Latência da rede			
Taxa de transferência de dados		**X**	**X**
Utilização de recursos		**X**	**X**
Carga de trabalho		**X**	**X**
Disponibilidade e avaliabilidade			
Congestionamento de dados na rede			

Fonte: Autoria própria (2022).

Para calcular a adequação, foi utilizada a fórmula X = 1-A/B em que A é o número de funcionalidades não atendidas enquanto B é o número total de funções avaliadas. Quanto mais próximo a 1, mais adequada é a ferramenta.

Ao total, 16 requisitos foram avaliados na subcaracterística de adequação: 4 requisitos de agente, 4 requisitos de gerente e 8 requisitos para monitoramento da rede (Quadro 33 ao Quadro 35). O cálculo para cada ferramenta é apresentada no Quadro 36, o que permite verificar que a ferramenta Zabbix obteve o melhor resultado.

Quadro 36 – Avaliação da Subcaracterística de Adequação

Característica	Subcaracterística	Métrica	Ferramenta	Interpretação	Resultado
Funcionalidade	Adequação	Adequação Funcional	Ganglia	X = 1 - (10,5/16)	0,3438
Funcionalidade	Adequação	Adequação Funcional	Nagios	X = 1 - (7,5/16)	0,5313
Funcionalidade	Adequação	Adequação Funcional	Zabbix	X = 1 - (7/16)	0,5625

Fonte: Autoria própria (2022).

5.2 INTEROPERABILIDADE

A interoperabilidade possui o intuito de analisar se a ferramenta possui capacidade de interagir com outros sistemas. Para esse trabalho, avaliou-se esse critério como a capacidade de alertar sobre eventos de alerta ocorridos com os equipamentos monitorados. Sendo assim, foram elencadas as ferramentas mais utilizadas para o recebimento de alertas: e-mail, SMS, Whatsapp, Telegram e Slack. Nos testes realizados, foi verificado somente o envio de e-mails, as demais compatibilidades foram investigadas a partir de uma pesquisa com o intuito de verificar a possibilidade de integração.

Para o Ganglia, não foram encontradas fontes para evidenciar as compatibilidades de integração com cada ferramenta de notificação. O Nagios pode ser integrado com e-mail (assim como verificado através dos testes), SMS[1], WhatsApp[2], Telegram[3] e Slack[4]. O Zabbix possui funcionalidade de integrar com e-mail (verificado através dos testes), SMS[5], WhatsApp[6], Telegram[7] e Slack[8]. O Quadro 37 resume os dados encontrados através da pesquisa.

Para calcular, utiliza-se a fórmula X = 1-A/B em que A expressa as tarefas que apresentam falhas ou incapacidade na troca de dados enquanto B é o número total de tarefas avaliadas.

1 https://exchange.nagios.org/directory/Addons/Notifications/SMS
2 https://www.unixmen.com/send-nagios-alert-notification-using-whatsapp/)
3 https://exchange.nagios.org/directory/Addons/Notifications/Telegram-Notifications/details
4 https://slack.com/apps/A0F81R747-nagios?tab=more_info
5 https://www.zabbix.com/integrations/sms
6 https://sudonull.com/post/109670-Receive-notifications-from-Zabbix-on-WhatsApp
7 https://www.zabbix.com/integrations/telegram
8 https://www.zabbix.com/integrations/slack

Quadro 37 – Compatibilidade das Ferramentas com Recursos de Interoperabilidade

Ferramenta de Integração	Ganglia	Nagios	Zabbix
E-mail		X	X
SMS		X	X
WhatsApp		X	X
Telegram		X	X
Slack		X	X

Fonte: Autoria própria (2022).

Quanto mais próximo a 1, mais interoperável é a ferramenta. No total, são 5 itens para cada ferramenta, o Quadro 38 apresenta os resultados obtidos, que permite observar que o Nagios e o Zabbix apresentaram o mesmo resultado, enquanto que o Ganglia obteve o pior resultado.

Quadro 38 – Avaliação da Subcaracterística de Interoperabilidade

Característica	Subcaracterística	Métrica	Ferramenta	Interpretação	Resultado
Funcionalidade	Interoperabilidade	Interoperabilidade Disponível	Ganglia	X = 1 - (5/5)	0
Funcionalidade	Interoperabilidade	Interoperabilidade Disponível	Nagios	X = 1 - (0/5)	1
Funcionalidade	Interoperabilidade	Interoperabilidade Disponível	Zabbix	X = 1 - (0/5)	1

Fonte: Autoria própria (2022).

5.3 CONSIDERAÇÕES FINAIS

Em cada seção são apresentados os resultados para cada subcaracterística. De forma quantitativa, em relação à adequação, o Ganglia completou 5,5 dos 16 requisitos, o Nagios completou 8,5 requisitos enquanto que o Zabbix contemplou 9 requisitos. Para a interoperabilidade, o Ganglia não contemplou integração com nenhuma ferramenta elencada, enquanto o Nagios e o Zabbix atenderam a todas.

Aplicando o peso de 90% para a adequação e 10% para a interoperabilidade, os resultados alcançados são detalhados no Quadro 39. Esse resultado permite constatar que a melhor ferramenta analisada foi o Zabbix, seguida pelo Nagios e por último pelo Ganglia.

Quadro 39 – Resultados da Avaliação

Software Avaliado	**Interpretação**	**Resultado**
Ganglia	0,9*0,3438 + 0,1*0	X = 0,30942
Nagios	0,9*0,5313 + 0,1*1	X = 0,57817
Zabbix	0,9*0,5625 + 0,1*1	X = 0,60625

Fonte: Autoria própria (2022).

6 CONCLUSÕES

As aplicações atuais demandam grande tráfego de dados, baixa latência, alta disponibilidade, entre outros requisitos. A arquitetura de Névoa apresenta-se como um conjunto de tecnologias integradas que levam o processamento e armazenamento para mais próximo do usuário final. Dessa forma, diversos cenários podem tirar vantagem da sua utilização, como a área de IoT, área da saúde, gestão de tráfego de trânsito, etc..

O gerenciamento de redes é essencial para tomar ações de melhoria de desempenho, controle de dispositivos, controle de segurança, dimensionamento de recursos, etc.. Em termos de gerenciamento, a Computação em Névoa, por tratar-se de um paradigma novo na área de computação, não possui ainda ferramentas amplamente difundidas que dão suporte para o controle dessas redes.

Dessa maneira, o objetivo desse trabalho foi analisar se os softwares de gerenciamento de redes suportam os requisitos necessários para gerenciar redes de Computação em Névoa. Para atingir esse objetivo, foi realizada uma pesquisa na forma de revisão sistemática em que foram elencadas ferramentas e requisitos de gerenciamento para uma arquitetura em Névoa. Conseguintemente, foram elencados os requisitos para teste, selecionada a norma ISO/IEC a ser utilizada, definidas as métricas e os casos de teste.

A fase de definição dos critérios e métricas de avaliação foi de grande importância na realização desse trabalho, tendo em vista que os métodos definidos conduziram a fase prática, garantindo a acuracidade dos testes.

Na fase de pesquisa dos artigos, esperava-se encontrar novas ferramentas de gerenciamento que pudessem ser utilizadas na Névoa mas com o decorrer da análise, foi verificado que os autores mencionavam ferramentas já existentes e analisavam seu uso no ambiente de Névoa. Diversas são as razões pelas quais essas ferramentas são consideradas inadequadas por eles. O Ganglia, por exemplo, apesar de ser escalável, hierárquico e distribuído, foi desenvolvido para sistemas que sofrem poucas mudanças de infraestrutura, não possui capacidade de auto descoberta e auto configuração. O Nagios possui difícil configuração e não apresenta descoberta automática de dispositivos. A ferramenta MonALISA também é dedicada para ambientes com poucas mudanças de infraestrutura. O Zabbix é pouco flexível, já que possui um servidor centralizado além de seu recurso de auto descoberta de dispositivos ser lento. O PCMONS é adequado somente para ambientes com baixas taxas de atualização. O OpenNebula possui atualização dos dados de monitoramento em períodos estáticos, sendo pouco viável para ambientes dinâmicos. A ferramenta Lattice não possui visualização e geração de alertas automatizados bem como não possui uma biblioteca de recursos para monitoramento. O JCatascopia possui alto consumo de recursos. Visto que nenhuma das ferramentas citadas pelos autores foram consideradas como

aptas ao cenário de Névoa, a análise foi concentrada na adequação das ferramentas aos critérios: quantidade de ao menos duas repetições nos artigos, compatibilidade com o sistema operacional Linux, código livre ou versão de testes disponível para instalação além de contar com documentação. Sendo assim, foram selecionadas as ferramentas: Ganglia, Nagios e Zabbix.

Após a realização dos testes no ambiente desenvolvido, constatou-se que a ferramenta que obteve melhor resultado, considerando os 16 requisitos e a interoperabilidade, foi o Zabbix. Em relação aos agentes e servidor, somente não foi compatível com a disponibilidade e avaliabilidade. Em contrapartida, para a rede, só monitorou taxa de transferência de dados, utilização de recursos e carga de trabalho. O Nagios, que obteve o segundo lugar, em relação aos agentes, só não foi capaz de monitorar a disponibilidade e avaliabilidade. Para o servidor, contemplou totalmente a taxa de transferência de dados e carga de trabalho. De maneira parcial, atendeu e a utilização de recursos, tendo em vista que não apresentou gráfico para taxa de leitura e escrita de disco. Para a rede, contemplou os mesmos requisitos que o Zabbix. O Ganglia, que obteve a última colocação, para os agentes, monitorou os mesmos requisitos que as demais ferramentas. Para o servidor, apresentou o mesmo resultado que o Nagios e para a rede, não se adequou a nenhum requisito tendo em vista a incompatibilidade de seu *software* agente com o ponto de acesso. Em relação à interoperabilidade, o Nagios e o Zabbix apresentaram-se como interoperáveis com todas as ferramentas de integração (e-mail, SMS, WhatsApp, Telegram e Slack) enquanto que o Ganglia não foi compatível com nenhuma delas.

A avaliação explicitou, portanto, que nenhum desses *softwares* estão aptos a monitorar um ambiente de Névoa, tendo em vista que diversos dos requisitos necessários não foram atendidos. Entre o Nagios e o Zabbix, houve pouca diferença apontada pela avaliação (Quadro 39). Em relação ao Ganglia, houve uma grande discrepância para o primeiro e segundo colocados. De maneira geral, isso deve-se à questão de nenhum requisito de rede ter sido monitorado, além da limitação encontrada para a interoperabilidade.

A realização do trabalho proporcionou ampliar os conhecimentos em redes de computadores, principalmente ao que tange o gerenciamento e monitoramento de redes e seus protocolos. Para o aprofundamento no uso das ferramentas avaliadas, foi necessário bastante estudo, tendo como fonte as documentações, além de cursos realizados de forma *online*. O conhecimento sobre o sistema operacional Linux também foi aprimorado, tendo em vista que diversas instalações e configurações tiveram que ser realizadas nos equipamentos utilizados. Encontrar um ponto de acesso com suporte ao protocolo SNMP foi um desafio, tendo em vista que é mais comum no ambiente corporativo e possui alto custo para compra. Sendo assim, foi estudado o sistema operacional OpenWrt o qual foi utilizado em uma Raspberry Pi configurada como ponto de acesso, o que permitiu testar os requisitos de rede.

Evidentemente, até a realização desse trabalho, o mercado não conta com soluções de monitoramento de ambiente computacionais em Névoa, o que colabora para dificuldades de controle, monitoramento e garantia de *QoS*. Dessa forma, reitera-se que o objetivo do trabalho

foi atingido, pois foram definidas métricas para gerenciamento, analisadas ferramentas já presentes no mercado além de ter sido definido, implantado e utilizado um ambiente para simular a arquitetura em questão.

Esse trabalho contribuiu diretamente para auxiliar sobre a adesão de ferramentas para esse propósito, tendo como base uma revisão sistemática de trabalhos relacionados e a adoção de normas ISO/IEC. Por fim, é sugerida como continuidade ou complementação desse trabalho, o estudo da viabilidade de adicionar o suporte para ambientes em Névoa nos *softwares* testados. Isso pode ser feito entrando em contato com os desenvolvedores ou até mesmo através da comunidade. O fato de possuirem código livre amplia essa possibilidade, pois permite que desenvolvedores criem iniciativas de desenvolvimento de funções específicas através de comunidades de cooperação ao redor do mundo. Dessa forma, tornam-se capazes de criar aplicações que, se desenvolvidas de maneira não cooperativa, apresentaria alto custo de implementação além de alto tempo para ir ao mercado. O Nagios, por exemplo, possui uma grande comunidade que desenvolve extensões e disponibiliza gratuitamente para o uso, utilizando alguma licença de *software* livre.

REFERÊNCIAS

AAZAM, M.; HUH, E.-N. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In: **2015 IEEE 29th International Conference on Advanced Information Networking and Applications**. [S.l.: s.n.], 2015. p. 687–694.

AAZAM, M. *et al.* Mefore: Qoe based resource estimation at fog to enhance qos in iot. In: **2016 23rd International Conference on Telecommunications (ICT)**. [S.l.: s.n.], 2016. p. 1–5.

____. Pre-fog: Iot trace based probabilistic resource estimation at fog. In: **2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)**. [S.l.: s.n.], 2016. p. 12–17.

ABDELMONEEM, R. M.; BENSLIMANE, A.; SHAABAN, E. Mobility-aware task scheduling in cloud-fog iot-based healthcare architectures. **Computer Networks**, v. 179, p. 107348, 2020. ISSN 1389-1286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1389128619313106>.

ABREHA, H. G. Self-adaptive monitoring in fog computing by leveraging machine learning. 04 2020.

AHMAD, M. A.; PATRA, S. S.; BARIK, R. K. Energy-efficient resource scheduling in fog computing using sdn framework. In: DAS, H. *et al.* (Ed.). **Progress in Computing, Analytics and Networking**. Singapore: Springer Singapore, 2020. p. 567–578. ISBN 978-981-15-2414-1.

AL-ZIHAD, M. *et al.* Bandwidth allocation and computation offloading for service specific iot edge devices. In: **2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)**. [S.l.: s.n.], 2017. p. 516–519.

ALAM, M. G. R. *et al.* Autonomic computation offloading in mobile edge for iot applications. **Future Generation Computer Systems**, v. 90, p. 149–157, 2019. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X18303996>.

ALENCAR, B. de M. Aprendizado de máquina para redução do tráfego de dados e da latência na névoa das coisas. Jun 2021. Disponível em: <https://repositorio.ufba.br/ri/bitstream/ri/33634/1/Dissertao-Brenno-Mello-correcoes.pdf>.

A.M, S. K.; VENKATESAN, D. Task scheduling in a cloud computing environment using hgpso algorithm. **Cluster Computing**, v. 22, 01 2019.

AMARAL, M. C. do. Reason - avaliação de confiabilidade e disponibilidade em redes de computadores sustentáveis. Nov 2014. Disponível em: <https://www.teses.usp.br/teses/disponiveis/3/3141/tde-04112014-104303/pt-br.php>.

ASSILA, B. *et al.* Achieving low-energy consumption in fog computing environment: A matching game approach. In: **2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON)**. [S.l.: s.n.], 2018. p. 213–218.

BARCELó, M. *et al.* Iot-cloud service optimization in next generation smart environments. **IEEE Journal on Selected Areas in Communications**, PP, p. 1–1, 10 2016.

BARRETO, B. N. Uma arquitetura de fog computing virtualizada para prover gerenciamento de recursos, qos e sla em um ambiente inteligente. Aug 2020. Disponível em: <https://ri.ufs.br/bitstream/riufs/14130/2/BRUNO_NUNES_BARRETO.pdf>.

BHATTACHARYA, A.; DE, P. A survey of adaptation techniques in computation offloading. **Journal of Network and Computer Applications**, v. 78, p. 97–115, 2017. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804516302570>.

BITAM, S.; ZEADALLY, S.; MELLOUK, A. Fog computing job scheduling optimization based on bees swarm. **Enterprise Information Systems**, Taylor & Francis, v. 12, n. 4, p. 373–397, 2018. Disponível em: <https://doi.org/10.1080/17517575.2017.1304579>.

BITTENCOURT, L. F. *et al.* Mobility-aware application scheduling in fog computing. **IEEE Cloud Computing**, v. 4, n. 2, p. 26–35, 2017.

BONOMI, F. *et al.* Fog computing and its role in the internet of things. In: **Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing**. New York, NY, USA: Association for Computing Machinery, 2012. (MCC '12), p. 13–16. ISBN 9781450315197. Disponível em: <https://doi.org/10.1145/2342509.2342513>.

BROGI, A.; FORTI, S. Qos-aware deployment of iot applications through the fog. **IEEE Internet of Things Journal**, PP, p. 1–1, 05 2017.

CARDELLINI, V. *et al.* Distributed qos-aware scheduling in storm. In: **Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems**. New York, NY, USA: Association for Computing Machinery, 2015. (DEBS '15), p. 344–347. ISBN 9781450332866. Disponível em: <https://doi.org/10.1145/2675743.2776766>.

____. On qos-aware scheduling of data stream applications over fog computing infrastructures. In: **2015 IEEE Symposium on Computers and Communication (ISCC)**. [S.l.: s.n.], 2015. p. 271–276.

CHEN, S. *et al.* Framework for adaptive computation offloading in iot applications. In: **Proceedings of the 9th Asia-Pacific Symposium on Internetware**. New York, NY, USA: Association for Computing Machinery, 2017. (Internetware'17). ISBN 9781450353137. Disponível em: <https://doi.org/10.1145/3131704.3131717>.

CHEN, Y.-A.; WALTERS, J. P.; CRAGO, S. P. Load balancing for minimizing deadline misses and total runtime for connected car systems in fog computing. In: **2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)**. [S.l.: s.n.], 2017. p. 683–690.

CHENG, S. *et al.* Cloud service resource allocation with particle swarm optimization algorithm. In: . [S.l.: s.n.], 2017. p. 523–532. ISBN 978-981-10-7178-2.

CHOUDHARI, T.; MOH, M.; MOH, T.-S. Prioritized task scheduling in fog computing. In: **Proceedings of the ACMSE 2018 Conference**. New York, NY, USA: Association for Computing Machinery, 2018. (ACMSE '18). ISBN 9781450356961. Disponível em: <https://doi.org/10.1145/3190645.3190699>.

CHOWDHURY, M.; RAHMAN, M. R.; BOUTABA, R. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. **Networking, IEEE/ACM Transactions on**, v. 20, p. 206 – 219, 03 2012.

CIRANI, S. *et al.* The iot hub: a fog node for seamless management of heterogeneous connected smart objects. In: **2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)**. [S.l.: s.n.], 2015. p. 1–6.

CONSTANTINO, E. H. Rede digital virtual e computação em névoa para cidades inteligentes sustentáveis. Mar 2021. Disponível em: <http://repositorio.unicamp.br/bitstream/REPOSIP/357731/1/Constantino_EdelsonHenrique_M.pdf>.

COUTINHO, A. A.; CARNEIRO, E.; GREVE, F. Computação em névoa: Conceitos, aplicações e desafios. In: ____. [S.l.: s.n.], 2016. p. 266–315. ISBN 2177-4978.

DAI, S. *et al.* Hybrid quantum-behaved particle swarm optimization for mobile-edge computation offloading in internet of things. In: ____. [S.l.: s.n.], 2018. p. 350–364. ISBN 978-981-10-8889-6.

DENG, R. *et al.* Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In: . [S.l.: s.n.], 2015. p. 3909–3914.

____. Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption. **IEEE Internet of Things Journal**, p. 1–1, 01 2016.

DSOUZA, C.; AHN, G.-J.; TAGUINOD, M. Policy-driven security management for fog computing: Preliminary framework and a case study. In: **Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)**. [S.l.: s.n.], 2014. p. 16–23.

DUAN, Z. *et al.* Iot-based cost saving offloading system for cellular networks. **Tsinghua Science and Technology**, v. 22, n. 4, p. 379–388, 2017.

ELAZHARY, H. H.; SABBEH, S. F. The w5 framework for computation offloading in the internet of things. **IEEE Access**, v. 6, p. 23883–23895, 2018.

FILHO, N. F. D. Maqsaas - método para avaliação da qualidade em produtos saas. 2011.

FILIPOSKA, S.; MISHEV, A.; GILLY, K. Community-based allocation and migration strategies for fog computing. In: . [S.l.: s.n.], 2018. p. 1–6.

FLORES, H. *et al.* Large-scale offloading in the internet of things. In: **2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)**. [S.l.: s.n.], 2017. p. 479–484.

GAD-ELRAB, A. A.; NOAMAN, A. Y. A two-tier bipartite graph task allocation approach based on fuzzy clustering in cloud–fog environment. **Future Generation Computer Systems**, v. 103, p. 79–90, 2020. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X19305175>.

GAI, K. *et al.* Resource management in sustainable cyber-physical systems using heterogeneous cloud computing. **IEEE Transactions on Sustainable Computing**, PP, p. 1–1, 07 2017.

GAZIS, V. *et al.* Components of fog computing in an industrial internet of things context. In: **2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)**. [S.l.: s.n.], 2015. p. 1–6.

GAZORI, P.; RAHBARI, D.; NICKRAY, M. Saving time and cost on the scheduling of fog-based iot applications using deep reinforcement learning approach. **Future Generation Computer Systems**, v. 110, p. 1098–1115, 2020. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X19308702>.

GHOBAEI-ARANI, M.; JABBEHDARI, S.; POURMINA, M. A. An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach. **Future Generation Computer Systems**, v. 78, p. 191–210, 2018. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X17302327>.

GU, L. *et al.* Cost efficient resource management in fog computing supported medical cyber-physical system. **IEEE Transactions on Emerging Topics in Computing**, IEEE Computer Society, Los Alamitos, CA, USA, v. 5, n. 01, p. 108–119, jan 2017. ISSN 2168-6750.

GUAN, M. *et al.* Joint optimization for computation offloading and resource allocation in internet of things. In: **2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)**. [S.l.: s.n.], 2017. p. 1–5.

GUERRERO, C.; LERA, I.; JUIZ, C. A lightweight decentralized service placement policy for performance optimization in fog computing. **Journal of Ambient Intelligence and Humanized Computing**, v. 10, p. 1–18, 06 2019.

GUEVARA, J.; TORRES, R.; FONSECA, N. On the classification of fog computing applications: A machine learning perspective. **Journal of Network and Computer Applications**, v. 159, p. 102596, 03 2020.

GUPTA, H. *et al.* ifogsim: A toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments. **Software: Practice and Experience**, v. 47, 06 2016.

HAPP, D.; WOLISZ, A. Towards gateway to cloud offloading in iot publish/subscribe systems. In: . [S.l.: s.n.], 2017. p. 101–106.

HE, J. *et al.* Multitier fog computing with large-scale iot data analytics for smart cities. **IEEE Internet of Things Journal**, v. 5, n. 2, p. 677–686, 2018.

HOANG, D.; DANG, D. Fbrc: Optimization of task scheduling in fog-based region and cloud. In: . [S.l.: s.n.], 2017. p. 1109–1114.

HONG, C.-H.; VARGHESE, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. **ACM Computing Surveys**, v. 52, p. 1–37, 09 2019.

HONG, H.-J.; TSAI, P.-H.; HSU, C.-H. Dynamic module deployment in a fog computing platform. In: . [S.l.: s.n.], 2016. p. 1–6.

INTHARAWIJITR, K.; IIDA, K.; KOGA, H. Analysis of fog model considering computing and communication latency in 5g cellular networks. In: . [S.l.: s.n.], 2016. p. 1–4.

ISARD, M. *et al.* Quincy: Fair scheduling for distributed computing clusters. In: **Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles**. New York, NY, USA: Association for Computing Machinery, 2009. (SOSP '09), p. 261–276. ISBN 9781605587523. Disponível em: <https://doi.org/10.1145/1629575.1629601>.

ISO. **ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)**. 2005.

ISO/IEC 25010. **ISO/IEC 25010:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models**. 2011.

JAMIL, B. *et al.* A job scheduling algorithm for delay and performance optimization in fog computing. **Concurrency and Computation Practice and Experience**, p. 1–17, 11 2019.

JIE, Y. *et al.* Online task scheduling for edge computing based on repeated stackelberg game. **Journal of Parallel and Distributed Computing**, v. 122, p. 159–172, 2018. ISSN 0743-7315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0743731518305409>.

JUTILA, M. An adaptive edge router enabling internet of things. **IEEE Internet of Things Journal**, v. 3, n. 6, p. 1061–1069, 2016.

KABIRZADEH, S.; RAHBARI, D.; NICKRAY, M. A hyper heuristic algorithm for scheduling of fog networks. In: **Proceedings of the 21st Conference of Open Innovations Association FRUCT**. Helsinki, Uusimaa, FIN: FRUCT Oy, 2017. (FRUCT'21), p. 148–155. Disponível em: <https://doi.org/10.23919/FRUCT.2017.8250177>.

KAFHALI, S. E.; SALAH, K. Efficient and dynamic scaling of fog nodes for iot devices. **The Journal of Supercomputing**, v. 73, p. 5261–5284, 12 2017.

KAID, M.; OTHMAN, M. Cost-based multi-qos job scheduling using divisible load theory in cloud computing. **Procedia Computer Science**, v. 18, p. 928–935, 12 2013.

KASHANI, M.; RAHMANI, A.; NAVIMIPOUR, N. Quality of service-aware approaches in fog computing. **International Journal of Communication Systems**, v. 33, p. e4340, 02 2020.

KEARY, T. **10 Best Network Monitoring Tools and Software for 2021 (Free and Paid)**. 2021. Disponível em: <https://www.comparitech.com/net-admin/network-monitoring-tools/>.

KIM, M. Nested game-based computation offloading scheme for mobile cloud iot systems. **EURASIP Journal on Wireless Communications and Networking**, v. 2015, 10 2015.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a internet: Uma abordagem top-down**. [S.l.]: Pearson, 2013.

LI, G. *et al.* Methods of resource scheduling based on optimized fuzzy clustering in fog computing. **Sensors**, v. 19, p. 2122, 05 2019.

LIU, L. *et al.* Multiobjective optimization for computation offloading in fog computing. **IEEE Internet of Things Journal**, v. 5, n. 1, p. 283–294, 2018.

LIU, Q. *et al.* Task scheduling in fog enabled internet of things for smart cities. **2017 IEEE 17th International Conference on Communication Technology (ICCT)**, p. 975–980, 2017.

LYU, X. *et al.* Selective offloading in mobile edge computing for the green internet of things. **IEEE Network**, v. 32, n. 1, p. 54–60, 2018.

MAHMUD, M. R. *et al.* Maximizing quality of experience through context-aware mobile application scheduling in cloudlet infrastructure. **Software: Practice and Experience**, v. 46, n. 11, p. 1525–1545, 2016. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2392>.

MAHMUD, R. *et al.* Quality of experience (qoe)-aware placement of applications in fog computing environments. **Journal of Parallel and Distributed Computing**, v. 132, p. 190–203, 2019. ISSN 0743-7315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0743731518301771>.

MARINESCU, D. C. *et al.* An approach for scaling cloud resource management. **Cluster Computing**, Kluwer Academic Publishers, USA, v. 20, n. 1, p. 909–924, mar. 2017. ISSN 1386-7857. Disponível em: <https://doi.org/10.1007/s10586-016-0700-8>.

MASSIE, M. **Monitoring with ganglia**. [S.l.]: O'Reilly, 2012.

MASTOI, Q.-u.-a. *et al.* A novel cost-efficient framework for critical heartbeat task scheduling using the internet of medical things in a fog cloud system. **Sensors**, v. 20, n. 2, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/2/441>.

MIN, M. *et al.* Learning-based computation offloading for iot devices with energy harvesting. **IEEE Transactions on Vehicular Technology**, v. 68, n. 2, p. 1930–1941, 2019.

MINH, Q. T. *et al.* Toward service placement on fog computing landscape. In: **2017 4th NAFOSTED Conference on Information and Computer Science**. [S.l.: s.n.], 2017. p. 291–296.

MONONEN, T.; AREF, M. M.; MATTILA, J. Filtering scheme for context-aware fog computing in cyber-physical systems. In: **2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)**. [S.l.: s.n.], 2018. p. 1–7.

NAGIOS XI - Download. 2022. Disponível em: <https://www.nagios.com/downloads/nagios-xi/>.

NAHA, R. K. *et al.* Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. **Future Generation Computer Systems**, v. 104, p. 131–141, 2020. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X19319016>.

NGUYEN, B. M. *et al.* Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment. **Applied Sciences**, v. 9, n. 9, 2019. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/9/9/1730>.

NI, L. *et al.* Resource allocation strategy in fog computing based on priced timed petri nets. **IEEE Internet of Things Journal**, v. 4, n. 5, p. 1216–1228, 2017.

NINGNING, S. *et al.* Fog computing dynamic load balancing mechanism based on graph repartitioning. **China Communications**, v. 13, n. 3, p. 156–164, 2016.

OTTENWäLDER, B. *et al.* Migcep: Operator migration for mobility driven distributed complex event processing. In: **Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems**. New York, NY, USA: Association for Computing Machinery, 2013. (DEBS '13), p. 183–194. ISBN 9781450317580. Disponível em: <https://doi.org/10.1145/2488222.2488265>.

OUEIS, J.; STRINATI, E.; BARBAROSSA, S. The fog balancing: Load distribution for small cell cloud computing. **IEEE Vehicular Technology Conference**, v. 2015, 07 2015.

PANWAR, N. *et al.* Topsis–pso inspired non-preemptive tasks scheduling algorithm in cloud environment. **Cluster Computing**, v. 22, 12 2019.

PARK, Y.; KIM, M. Game-based data offloading scheme for iot system traffic congestion problems. **EURASIP Journal on Wireless Communications and Networking**, v. 2015, p. 192, 07 2015.

PHAM, X.-Q.; HUH, E.-N. Towards task scheduling in a cloud-fog computing system. In: **2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)**. [S.l.: s.n.], 2016. p. 1–4.

PHAM, X.-Q. *et al.* A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. **International Journal of Distributed Sensor Networks**, v. 13, p. 155014771774207, 11 2017.

PRAKASH, P. *et al.* Fog computing: Issues, challenges and future directions. **International Journal of Electrical and Computer Engineering**, v. 7, p. 3669–3673, 12 2017.

PRASAD, V.; BHAVSAR, M.; TANWAR, S. Influence of montoring: Fog and edge computing. **Scalable Computing**, v. 20, p. 365–376, 05 2019.

P.SAHARAN, K.; KUMAR, A. Fog in comparison to cloud: A survey. **International Journal of Computer Applications**, v. 122, p. 10–12, 07 2015.

RAD, B. B.; SHAREEF, A. Fog computing: A short review of concept and applications. v. 17, p. 68–74, 11 2017.

RAHBARI, D.; KABIRZADEH, S.; NICKRAY, M. A security aware scheduling in fog computing by hyper heuristic algorithm. In: . [S.l.: s.n.], 2017. p. 87–92.

RAHBARI, D.; NICKRAY, M. Scheduling of fog networks with optimized knapsack by symbiotic organisms search. In: **Proceedings of the 21st Conference of Open Innovations Association FRUCT**. Helsinki, Uusimaa, FIN: FRUCT Oy, 2017. (FRUCT'21), p. 278–283. Disponível em: <https://doi.org/10.23919/FRUCT.2017.8250193>.

RASHEED, S. *et al.* A cloud-fog based smart grid model using max-min scheduling algorithm for efficient resource allocation: The 21st international conference on network-based information systems (nbis-2018). In: ____. [S.l.: s.n.], 2019. p. 273–285. ISBN 978-3-319-98529-9.

SAMIE, F. *et al.* Computation offloading and resource allocation for low-power iot edge devices. In: **2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)**. [S.l.: s.n.], 2016. p. 7–12.

SAQIB, M. T.; HAMID, M. A. Fogr: A highly reliable and intelligent computation offloading on the internet of things. In: **2016 IEEE Region 10 Conference (TENCON)**. [S.l.: s.n.], 2016. p. 1039–1042.

SARKAR, S.; CHATTERJEE, S.; MISRA, S. Assessment of the suitability of fog computing in the context of internet of things. **IEEE Transactions on Cloud Computing**, PP, p. 1–1, 10 2015.

SHARIF, M.; MERCELIS, S.; HELLINCKX, P. Context-aware optimization of distributed resources in internet of things using key performance indicators. In: XHAFA, F.; CABALLÉ SANTIAND BAROLLI, L. (Ed.). **Advances on P2P, Parallel, Grid, Cloud and Internet Computing**. Cham: Springer International Publishing, 2018. p. 733–742. ISBN 978-3-319-69835-9.

SHARMA, S.; SAINI, H. A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. **Sustainable Computing: Informatics and Systems**, v. 24, p. 100355, 2019. ISSN 2210-5379. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2210537919300885>.

SHOJAFAR, M.; CORDESCHI, N.; BACCARELLI, E. Energy-efficient adaptive resource management for real-time vehicular cloud services. **IEEE Transactions on Cloud Computing**, v. 7, n. 1, p. 196–209, 2019.

SHUKLA, R. M.; MUNIR, A. A computation offloading scheme leveraging parameter tuning for real-time iot devices. In: **2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)**. [S.l.: s.n.], 2016. p. 208–209.

SILVA, F. dos S. Algoritmo para posicionamento de serviços multimídia em ambientes hierárquicos nuvem-névoa. Mar 2021. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/357549?mode=full>.

SKARLAT, O. *et al.* Optimized iot service placement in the fog. **Service Oriented Computing and Applications**, v. 11, p. 427–443, 12 2017.

____. Towards qos-aware fog service placement. In: . [S.l.: s.n.], 2017.

SOTIRIADIS, S.; BESSIS, N.; BUYYA, R. Self managed virtual machine scheduling in cloud systems. **Information Sciences**, v. 433-434, p. 381–400, 2018. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025517308277>.

SOUZA, V. *et al.* Handling service allocation in combined fog-cloud scenarios. In: . [S.l.: s.n.], 2016.

SU, J. *et al.* Steiner tree based optimal resource caching scheme in fog computing. **China Communications**, v. 12, n. 8, p. 161–168, 2015.

SUN, Y.; LIN, F.; XU, H. Multi-objective optimization of resource scheduling in fog computing using an improved nsga-ii. **Wireless Personal Communications**, v. 102, 09 2018.

SURESH, A.; VARATHARAJAN, R. Competent resource provisioning and distribution techniques for cloud computing environment. **Cluster Computing**, v. 22, 09 2019.

TANEJA, M.; DAVY, A. Resource aware placement of iot application modules in fog-cloud computing paradigm. In: . [S.l.: s.n.], 2017. p. 1222–1228.

TIBúRCIO, P.; SANTOS, M.; FERNANDES, S. Cidades inteligentes: Uma arquitetura de gerenciamento autonômica no contexto de iot. In: **Anais do IV Workshop Pré-IETF**. Porto Alegre, RS, Brasil: SBC, 2017. ISSN 2595-6388. Disponível em: <https://sol.sbc.org.br/index.php/wpietf/article/view/3607>.

VELASQUEZ, K. *et al.* Service placement for latency reduction in the internet of things. **Annals of Telecommunications**, v. 72, 06 2016.

VENTICINQUE, S.; AMATO, A. A methodology for deployment of iot application in fog. **Journal of Ambient Intelligence and Humanized Computing**, v. 10, 05 2019.

VERMA, M.; BHARDWAJ, N.; YADAV, A. Real time efficient scheduling algorithm for load balancing in fog computing environment. **International Journal of Information Technology and Computer Science**, v. 8, p. 1–10, 04 2016.

WAN, J. *et al.* Fog computing for energy-aware load balancing and scheduling in smart factory. **IEEE Transactions on Industrial Informatics**, v. 14, n. 10, p. 4548–4556, 2018.

WANG, J.; LI, D. Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. **Sensors**, v. 19, n. 5, 2019. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/19/5/1023>.

WANG, N. *et al.* Enorm: A framework for edge node resource management. **IEEE Transactions on Services Computing**, v. 13, n. 6, p. 1086–1099, 2020.

WANG, S.; ZAFER, M.; LEUNG, K. Online placement of multi-component applications in edge computing environments. **IEEE Access**, v. 5, p. 2514–2533, 02 2017.

WANG, T. *et al.* A comprehensive survey on mobile data offloading in heterogeneous network. **Wirel. Netw.**, Springer-Verlag, Berlin, Heidelberg, v. 25, n. 2, p. 573–584, fev. 2019. ISSN 1022-0038. Disponível em: <https://doi.org/10.1007/s11276-017-1576-0>.

WANG, Y.; GUO, C.; YU, J. Immune scheduling network based method for task scheduling in decentralized fog computing. **Wireless Communications and Mobile Computing**, v. 2018, p. 1–8, 09 2018.

WU, H. Multi-objective decision-making for mobile cloud offloading: A survey. **IEEE Access**, v. 6, p. 3962–3976, 2018.

XIAO, Y.; KRUNZ, M. Distributed optimization for energy-efficient fog computing in the tactile internet. **IEEE Journal on Selected Areas in Communications**, v. 36, n. 11, p. 2390–2400, 2018.

XIONG, Z. *et al.* Cloud/fog computing resource management and pricing for blockchain networks. **IEEE Internet of Things Journal**, PP, p. 1–1, 09 2018.

XU, D. *et al.* A survey of opportunistic offloading. **IEEE Communications Surveys Tutorials**, v. 20, n. 3, p. 2198–2236, 2018.

YANG, L. *et al.* Cost aware service placement and load dispatching in mobile cloud systems. **IEEE Transactions on Computers**, v. 65, p. 1–1, 01 2015.

YANG, X. *et al.* Small-cell assisted secure traffic offloading for narrowband internet of thing (nb-iot) systems. **IEEE Internet of Things Journal**, v. 5, n. 3, p. 1516–1526, 2018.

YANG, Y. *et al.* Debts: Delay energy balanced task scheduling in homogeneous fog networks. **IEEE Internet of Things Journal**, v. 5, n. 3, p. 2094–2106, 2018.

YIN, L.; LUO, J.; LUO, H. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacture. **IEEE Transactions on Industrial Informatics**, PP, p. 1–1, 06 2018.

YOUSEFPOUR, A. *et al.* On reducing iot service delay via fog offloading. **IEEE Internet of Things Journal**, v. 5, n. 2, p. 998–1010, 2018.

YU, W. *et al.* A survey on the edge computing for the internet of things. **IEEE Access**, v. 6, p. 6900–6919, 2018.

ZABBIX Download and install. 2022. Disponível em: <https://www.zabbix.com/download>.

ZAHOOR, S. *et al.* Cloud–fog–based smart grid model for efficient resource management. **Sustainability**, v. 10, n. 6, 2018. ISSN 2071-1050. Disponível em: <https://www.mdpi.com/2071-1050/10/6/2079>.

ZENG, D. *et al.* Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. **IEEE Transactions on Computers**, v. 65, p. 3702–3712, 2016.

ZHANG, H. *et al.* A hierarchical game framework for resource management in fog computing. **IEEE Communications Magazine**, v. 55, p. 52–57, 08 2017.

ZHAO, S. *et al.* Femos: Fog-enabled multitier operations scheduling in dynamic wireless networks. **IEEE Internet of Things Journal**, v. 5, n. 2, p. 1169–1183, 2018.

ZHOU, H. *et al.* A survey on mobile data offloading technologies. **IEEE Access**, v. 6, p. 5101–5111, 2018.

ZUO, L. *et al.* Dynamically weighted load evaluation method based on self-adaptive threshold in cloud computing. **Mobile Networks and Applications**, v. 22, 02 2017.