UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO FERREIRA DE MENDONÇA JÚNIOR

GERENCIAMENTO DE RECURSOS COMPUTACIONAIS EM REDES *FOG* VEICULARES: cenários, limites e alocação de tarefas baseada em leilões

Recife

2021

FRANCISCO FERREIRA DE MENDONÇA JÚNIOR

GERENCIAMENTO DE RECURSOS COMPUTACIONAIS EM REDES *FOG* VEICULARES: cenários, limites e alocação de tarefas baseada em leilões

Tese de Doutorado apresentada à Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação em 28 de Julho de 2021.

Área de Concentração: Redes de Computadores e Sistemas Distribuídos

Orientador (a): Kelvin Lopes Dias

Coorientador (a): Pedro M. d'Orey

Recife

2021

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

M539g Mendonça Júnior, Francisco Ferreira de
Gerenciamento de recursos computacionais em redes *fog* veiculares: cenários, limites e alocação de tarefas baseada em leilões / Francisco Ferreira de Mendonça Júnior. – 2021.
168 f.: il., fig., tab.

Orientador: Kelvin Lopes Dias.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2021.
Inclui referências e apêndice.

1. Redes de computadores e sistemas distribuídos. 2. Internet dos veículos. 3. Computação *fog*. 4. Computação na borda. I. Dias, Kelvin Lopes (orientador). II. Título

004.6 CDD (23. ed.) UFPE - CCEN 2022 – 102

Francisco Ferreira de Mendonça Júnior

“Gerenciamento de Recursos Computacionais em Redes Fog Veiculares: Cenários, Limites e Alocação de Tarefas Baseada em Leilões”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovado em: 28/07/2021.

_____

Orientador: Prof. Dr. Kelvin Lopes Dias

BANCA EXAMINADORA

_____

Prof. Dr. José Augusto Suruagy Monteiro
Centro de Informática / UFPE

_____

Prof. Dr. Renato Mariz de Moraes
Centro de Informática / UFPE

_____

Prof. Dr. Carlos André Guimaraes Ferraz
Centro de Informática / UFPE

_____

Prof. Dr. Antônio Alfredo Ferreira Loureiro
Departamento de Ciência da Computação / UFMG

_____

Prof. Dr. Jó Ueyama
Instituto de Ciências Matemáticas e de Computação / USP

A emoção mais antiga e mais forte da humanidade é o medo, e o tipo de medo mais antigo e mais poderoso é o medo do desconhecido. (LOVECRAFT, 1987, p. 10)

AGRADECIMENTOS

Chega um ponto da vida em que o ser humano percebe que não consegue fazer absolutamente nada sozinho. Alguém pode até pensar que pregou um prego na parede sem ajuda de ninguém, mas "quem fez o prego?", "quem fez o martelo?", "quem ensinou a pregar?". Esse é um exemplo simples, mas já serve para imaginar a complexidade das relações humanas e como elas geram tudo o que somos, sabemos e fazemos.

Em primeiro lugar, agradeço à minha família. Agradeço à minha mãe e ao meu pai pelo recursos emocionais, afetivos e todos os aprendizados que me proporcionaram. Agradeço aos meus irmãos, pelas discussões filosófico-científicas que sempre tínhamos, e por dividirem suas moradias comigo, seja na minha cidade natal, seja na cidade de Recife.

Agradeço especialmente à Lívia Leite, minha namorada/noiva/esposa há tanto tempo que nem me dissocio dela. Passamos muita coisa juntos durante toda a nossa vida, mas passamos por momentos muito dolorosos ou felizes durante toda a minha trajetória na pós-graduação até aqui. Sem o apoio dela, eu não estaria onde estou.

É a ela que dedico esta tese. Só ela sabe como foi chegar até aqui.

Agradeço ao meu professor e orientador Kelvin Dias, pelas orientações, discussões e encaminhamentos que proporcionaram a realização deste trabalho. Além disso, ele foi responsável por abrir muitas portas na pesquisa e por me manter sempre atualizado com o que poderia ser mais relevante para o desenvolvimento deste trabalho. Foram vários projetos e artigos durante toda essa caminhada e o processo de orientação, acompanhamento e revisão, sempre detalhado e perspicaz, me serviu de base para minha vida profissional e pessoal. Ele tem uma contribuição importantíssima na minha formação como pesquisador, tanto nas questões metodológicas, quanto nas questões que envolvem o funcionamento real, muito intenso e nada fácil, deste mundo da academia e do ensino, sempre conectado com a indústria.

Agradeço ao meu coorientador Pedro D'Orey. Durante o meu doutorado sanduíche, ele proporcionou que eu entrasse em contato com uma cultura diferente e com principais tecnologias de redes veiculares e *Edge Computing*. Por meio dele, conheci o Zafeiris. Juntos, os dois contribuíram em vários elementos desta tese.

Também tenho que agradecer a todos que formam nosso grupo de pesquisa, em especial a Alexsanderson Santos e Edigleison Barbosa. Trabalhamos juntos por muito tempo e este tese tem muita contribuição deles, tanto em termos de discussões, quando nas implementações.

Inclusive, produzimos vários estudos, análises e resultados que, apenas de não entrarem nesta tese, foram essenciais para a evolução da metodologia desenvolvida.

Preciso agradecer também ao Bruno Roberto e à Atrícia Sabino, que me ajudaram a ter muitas ideias e sempre deixaram o ambiente no laboratório mais leve. Os mais novos, Samuel, Roseli e Henrique, além de Andson, contribuíram na formação como pesquisador, mostrando pontos de vista muito diferentes do que eu estava pensando o tempo todo. Aqui vale um agradecimento especial à Roseli. Sem a ajuda dela eu não teria conseguido me inscrever no doutorado sanduíche, que deve contribuir ainda mais para minha formação e para o desenvolvimento deste trabalho.

Por fim, agradeço aos meus professores e aos meus amigos e colega nesta caminhada, tanto do CIn, quanto das instituições onde estudei e trabalhei. Eles são parte importante do profissional que sou e do que quero me tornar.

RESUMO

Os veículos têm apresentado cada vez mais poder de processamento e demanda por aplicações, como o suporte à segurança dos usuários, direção autônoma e entretenimento. Neste contexto, observa-se um aumento da complexidade computacional necessária para integrar infraestruturas e algoritmos que dão suporte a essas aplicações. Isso acontece devido à natureza combinatorial com limite de tempo da alocação de vários tipos de recursos em redes veiculares, como canais, largura de banda, mensagens, aplicações e processamento. Assim, esta tese propõe uma metodologia de gerenciamento de recursos computacionais na Computação Fog Veicular - Vehicular Fog Computing (VFC) - através de Leilão Aberto de Primeiro Preço - One Shot Open First Price Auction (OSOFPA). A análise e a avaliação nos permitiram identificar que existem condições de rede e computação específicas que permitem o funcionamento da Computação *Fog* Veicular com suporte à baixa latência, especialmente relacionadas com aplicações completamente distribuídas, ou seja, sem suporte da nuvem centralizada. Tal modelo de gerenciamento de recursos permite a integração dos dispositivos de alta mobilidade e intermitência de conexão, no caso veículos, à cadeia de computação. De forma complementar, apresentamos um arcabouço que permite análise e investigação do funcionamento de veículos como servidores na VFC. O algoritmo proposto se adequa especialmente àquelas aplicações que possuem um funcionamento de acordo com um modelo de Requisição-Processamento-Resposta-Atuação (RPRA). Para isso, desenvolvemos e aperfeiçoamos a representação do funcionamento interno de dispositivos *Edge/Fog*, identificando relações entre capacidade e carga; simulamos e avaliamos protocolos para o gerenciamento de recursos, especialmente nos quesitos de gerenciamento de aplicações e gerenciamento de descoberta. Por fim, estendemos e propomos princípios de *design* que devem ser seguidos pelas aplicações. Esses princípios ajudam a formalizar uma definição de *Fog* e VFC, além de servir de base para a classificação de mecanismos de gerenciamento de recursos existentes e futuros. Os resultados da avaliação do leilão e de um *baseline* baseado em tratamento de resposta indicam que uma VFC totalmente distribuída consegue entregar resultados para aplicações de baixa latência, além de altas taxas de resposta em cenários com densidades menores de 33 Veículos Por Quilômetro Quadrado (VPQQ), ou com taxas de requisições menores que 10 Requisições Por Segundo (RPS). Além disso, fornecemos uma série de métricas relacionadas ao bem-estar social, que avalia o mecanismo de leilão e reflete esses mesmos cenários em termos da distribuição dos recursos computacionais.

Palavras-chaves: internet dos veículos; computação *fog*; computação na borda; requisição-processamento-resposta-atuação; gerenciamento de recursos; leilão.

ABSTRACT

Vehicles have shown increasing processing power and demand for applications supporting user safety, autonomous driving, and entertainment. Also, there is an increase in the computational complexity needed to integrate infrastructures and algorithms that support these applications due to the time-constrained combinatorial nature of allocating various types of resources in vehicular networks, such as channels, bandwidth, messages, applications, and processing. Thus, this thesis proposes a methodology for managing computational resources in Vehicle Fog Computing (VFC) through a One-Shot Open First Price Auction (OSOFPA). Analysis and evaluation allowed us to identify specific network and computing conditions that allow VFC to support low latency applications, especially those entirely distributed, without centralized cloud support. Such a resource management model allows the integration of highly mobile and intermittent connection devices, in this case, vehicles, to the computing chain. Complementarily, we present a framework that allows analysis and investigation of the operation of vehicles as servers in VFC. The proposed algorithm is especially suited to those applications that operate according to a Request-Processing-Response-Actuation (RPRA) model. Therefore, we developed and improved the representation of the internal operation of Edge/Fog devices, identifying relationships between capacity and load; we simulated and evaluated protocols for managing resources, especially in terms of application management and discovery management. Finally, we extend and propose design principles that applications must follow. These principles support a better definition of Fog and VFC and serve as a basis for classifying existing and future resource management mechanisms. The evaluation results indicate that a fully distributed VFC can deliver results for low latency applications, in addition to high response rates in scenarios with densities below 33 Vehicles per Square Kilometer (VPSK) or with request rates less than 10 Requests per Second (RPS). Finally, we provide a series of metrics related to social welfare that assess the auction mechanisms and reflect these same scenarios in terms of the distribution of computing resources.

Keywords: internet of vehicles; fog computing; edge computing; request-processing-response-actuation; resource management; auction.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

3GPP	3rd Generation Partnership Project
AA	Autonomic Application
AAC	Autonomic Application Controller
AP	Access Point
API	Application Programming Interface
AU	Application Unit
BER	Bit Error Rate
DANS	Driver-Abnormal Notification Scheme
DBT	Dynamic Binary Translation
DDNN	Distributed Deep Neural Networks
DFG	Dissolution and Formation of Groups
DRL	Deep Reinforcement Learning
DSRC	Dedicated Short Range Communications
DTN	Delay-Tolerant Netwoking
EH	Energy Harvesting
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
FISVER	Fog-Framework for Intelligent Public Safety In Vehicular Environment
FNC	Fog Node Controller
G-ECU	Gateway Electronic Control Unit
GPS	Global Positioning System
IA	Inteligência Artificial
IAAS	Infrastructure As A Service
IoT	Internet of Things
IoV	Internet of Vehicles

ITS	Intelligent Transportation Systems
LTE	Long Term Evolution
MANET	Mobile Ad Networks
MCC	Mobile Cloud Computing
MEC	Multi-Access Edge Computing
MFLOPS	Millions of Floating-Point Operations Per Second
MIPS	Millions of Instructions Per Second
ML	Machine Learning
NFV	Network Function Virtualization
O-REQ	Overhead De Requisições
OBU	On-Board Unit
OSOFPA	One Shot Open First Price Auction
OVS	Open Vswitch
PAAS	Platform As A Service
QOE	Quality of Experience
QOS	Quality of Service
R-TER	Requisições De Terceiros
RPRA	Requisição - Processamento - Resposta - Atuação
RPS	Requisições Por Segundo
RSU	Road Side Unit
RSU	Road Side Units
RTT	Round-Trip Time
SAAS	Software As A Service
SDN	Software-Defined Networking
TR	Taxa De Resposta
T-RES	Taxa De Resposta
TA	Trusted Authority

TMS	Traffic Management System
UE	User Equipment
URLLC	Ultra-Reliable Low-Latency Communications
V2I	Vehicle To Infrastructure
V2R	Vehicle To Roadside
V2V	Vehicle To Vehicle
V2X	Vehicle To Everything
VANET	Vehicular Ad Hoc Network
VANETS	Vehicular Ad Hoc Networks
VCC	Vehicular Cloud Computing
VCG	Vickrey-Clarke-Groves
VFC	Vehicular Fog Computing
VM	Virtual Machice
VPQQ	Veículos Por Quilômetro Quadrado
WAVE	Wireless Access In Vehicular Environments
WIMAX	Worldwide Interoperability for Microwave Access
WSDN	Wireless Software-Defined Networks

LISTA DE SÍMBOLOS

γ	Letra grega Gama
$\in$	Pertence
δ	Delta
θ	Teta
σ	Sigma
μ	Mi

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Na era da Internet of Things (IoT), os sensores estão espalhados pelas cidades inteligentes, fábricas da indústria 4.0 e até mesmo das casas inteligentes. Todos esses aparelhos estão conectados de forma a prover mais conforto e comodidade aos moradores. Há algum tempo, os sensores foram colocados dentro e fora dos veículos passaram a prover dados para os sistemas de tomada de decisão das cidades, especialmente no que diz respeito ao tráfego. A proliferação de sensores, além dos serviços de multimídia consumidos pelos usuários, requerem novas metodologias para lidar com a disponibilidade e distribuição dos serviços, além de um controle do tráfego dos dados gerados.

Novas tecnologias de nuvem, rede e virtualização, como Software-Defined Networking (SDN), Network Function Virtualization (NFV) e contêineres, além dos próprios avanços em termos de redes sem fio, estão criando perspectivas para o desenvolvimento de algoritmos e técnicas avançadas para o controle de tráfego dos dados de veículos e usuários. Grande parte das aplicações IoT e até mesmo as aplicações dos usuários, ainda depende fortemente de Internet e acesso à infraestruturas de nuvem.

Para que as aplicações funcionem num cenário de alta mobilidade, foram desenvolvidos algoritmos complexos de roteamento, *handover*, *caching*, *slicing*, virtualização de rede, previsão de mobilidade, previsão de uso de serviços, balanceamento de carga, entre outros (CHAHAL et al., 2017). A maior parte desses mecanismos busca reduzir a latência para o acesso a serviços de nuvem através das infraestruturas - Vehicle To Infrastructure (V2I) - ou até mesmo de *relaying* - Vehicle To Vehicle (V2V) - onde os veículos trocam dados ou servem de roteadores de dados uns para os outros. Existem estudos que apontam que a alta mobilidade e dinâmica da rede estão elevando a complexidade necessária para a distribuição desses dados (OUYANG; ZHOU; CHEN, 2018; ZHU et al., 2018b; FENG et al., 2020; BARIK et al., 2021).

Recentemente, a Cisco lançou o conceito de computação *Fog* (névoa) que pretende levar os benefícios e as aplicações de nuvem para a borda da rede (YI; LI; LI, 2015). *Fog* é uma expansão do conceito de computação *Edge* (borda), que permite a execução de serviços e algoritmos de nuvem especialmente projetados para dispositivos móveis - Mobile Cloud Computing (MCC) - ou especialmente voltados para veículos - Vehicular Cloud Computing (VCC) e Vehicular Fog Computing (VFC); ou servidores com infraestruturas de nuvem rodando na infraestrutura

de rede - Multi-Access Edge Computing (MEC); ou até mesmo dentro dos dispositivos de rede (Road Side Unit (RSU), *switches*, roteadores). Essas tecnologias ainda são pensadas, em algum momento, em termos de sua interação com a infraestrutura de nuvem centralizada, mas se posicionam cada vez mais como independentes.

Uma vez que estão sendo executados na borda em, pelo menos, uma dessas modalidades, os serviços apresentam baixa latência, ou mesmo baixa variação nessa latência. Além disso, os serviços ganham um componente de posicionamento, e podem ser implantados (*deployed*) nas proximidades do usuário. Tal implantação garante um balanceamento de carga espacial e pode, dependendo dos algoritmos e técnicas aplicados, gerar uma redução na carga total da rede, aliviando o uso das redes de acesso.

Neste trabalho, propomos FogWise, um arcabouço que, além de levar os serviços para a borda da rede, avalia a participação de veículos, que podem ter um poder de processamento elevado (SOUA; TOHME, 2018) na execução de serviços para os usuários, para outros veículos, e para o tratamento de dados de aplicações IoT, especialmente aquelas que se enquadram no modelo Requisição - Processamento - Resposta - Atuação (RPRA)(HONG et al., 2013), mas que pode ser estendido para outros modelos de funcionamento. FogWise utiliza análises da dinâmica das redes veiculares, das aplicações e das questões relacionadas à requisição, processamento e resposta dos serviços.

Através disso, propusemos uma estratégia de gerenciamento de recursos computacionais e de rede através de um Leilão Aberto de Primeira Oferta e Primeiro Preço - One Shot Open First Price Auction (OSOFPA). O leilão, como foi projetado, garante o balanceamento de carga entre os servidores da VFC - os veículos - permitindo execução de aplicações de baixa latência sob determinadas condições.

Os resultados dessa investigação indicam que existem limites para alguns tipos de aplicações de baixa latência executarem em redes VFC totalmente distribuídas, ou seja, sem interação ou dependência da Nuvem. Os principais fatores que podem determinar a possibilidade ou não da execução dessas aplicações são a densidade dos veículos, a taxa de requisições das aplicações e a quantidade de dados necessária para o processamento (tamanho da requisição). Avaliamos que aplicações de baixa latência em VFC podem funcionar com alta taxa de resposta quando as densidades estão abaixo de 33 Veículos por Quilômetro Quadrado, e com taxas de mensagens menores de 10 Mensagens por Segundo. Além desses resultados, avaliamos uma série de métricas relacionadas ao funcionamento de clientes, de servidores, do próprio leilão, e de bem-estar social, que avalia o mecanismo de leilão e reflete esses mesmos

cenários em termos da distribuição dos recursos computacionais.

1.2 PROBLEMA DE PESQUISA

Existem estudos que apontam que a alta mobilidade e dinâmica da rede estão elevando a complexidade necessária para a distribuição desses dados (OUYANG; ZHOU; CHEN, 2018; ZHU et al., 2018b; FENG et al., 2020; BARIK et al., 2021). Essa complexidade é encontrada em vários tipos de alocação necessárias para que redes veiculares funcionem de forma satisfatória.

Isso acontece devido à natureza combinatorial (ZHOU et al., 2019) da distribuição de recursos para veículos que estão em movimento. Durante o funcionamento normal de uma rede veicular, encontramos vários tipos de alocação necessárias:

- recursos de rádio e antenas (LIU et al., 2020);
- largura de banda para diferentes aplicações (Lin et al., 2018);
- tarefas de processamento (FENG et al., 2020; Peng; Ota; Dong, 2020; ZHU et al., 2018b);
- canais e mensagens (KU et al., 2014);
- dados e *cache* (BENAMAR et al., 2014; LIU et al., 2020).

Devido à elevada complexidade, a alocação geralmente se dá através de heurísticas, uma vez que as limitações de tempo geradas pela mobilidade inviabilizam as soluções ótimas (BENAMAR et al., 2014). Nesse contexto, buscamos dar uma solução com complexidade controlada para a alocação de tarefas em VFC totalmente distribuída, ou seja, sem o suporte da nuvem.

1.2.1 Questões de Pesquisa

A principal questão de pesquisa a ser respondida por esta tese está relacionada com a dinâmica das redes veiculares e o esquema RPRA, que é característico de aplicações IoT mas também de vários outros tipos de aplicações para usuários no modelo *Webservices*, como processamento de imagens e vídeos, além do processamento de *analytics* (SARKAR; MISRA, 2016):

- Como garantir a execução de aplicações de baixa latência em ambientes VFC totalmente distribuídos?

Os desdobramentos desse questionamento principal estão relacionados com a identificação de mecanismos e algoritmos necessários para que os dispositivos finais, especialmente os veículos, participem da VFC como servidores, a saber:

- Quais são as características de aplicações (tempo de execução, tipos de operações), capacidade de processamento e rede que um nó móvel (veículo) deve ter para participar da VFC em um determinado nível de mobilidade?
- Quais são os algoritmos e protocolos de comunicação necessários para garantir esse funcionamento, especialmente em baixa latência?
- Quais são as situações, ou tipos de aplicações, nas quais os veículos podem compor uma VFC completamente distribuída?
- Quais são as dificuldades inerentes à participação dos dispositivos finais, especialmente os veículos, na VFC?
- Quais são os limites de uma VFC independente da nuvem?

1.3 HIPÓTESES

Com base nesses questionamentos, foram levantadas algumas hipóteses iniciais sobre os resultados que podem ser obtidos ao final deste trabalho. A principal hipótese testada nesta tese contempla que:

- é possível projetar um algoritmo distribuído e um modelo de veículo como servidor que permita aos veículos monitorar o ambiente em termos de características da rede (espectro eletromagnético, tecnologias e protocolos); características das aplicações (tempo de execução, tipos de operações); e características dos próprios veículos (mobilidade, poder de processamento) de forma a permitir que a VFC disponibilize computação para aplicações de baixa latência, sejam elas aplicações de outros veículos de usuários ou mesmo de dispositivos IoT.

Outras hipóteses secundárias serão investigadas no decorrer do trabalho:

- existem limites, em termos da latência de execução, taxa de resposta, para que aplicações sejam executadas na VFC e esses limites estão relacionados com as características dos processamentos de veículo e da rede;

- as aplicações que mais se enquadram no modelo de computação *Fog* são aquelas que funcionam de acordo com um mecanismo RPRA;
- existem requisitos que precisam ser atendidos por todas as aplicações e por todos os dispositivos para que os veículos possam participar da cadeia de computação *Fog*.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Dadas essas questões de pesquisa, o objetivo geral da proposta FogWise é:

- Determinar o funcionamento de um mecanismo que permita a execução de aplicações de baixa latência de acordo com os requisitos e característica de uma VFC completamente distribuída.

1.4.2 Objetivos Específicos

O objetivo geral é composto por uma série de objetivos intermediários e específicos que precisam ser atingidos. Eles podem ser divididos em dois tipos de objetivos. Os primeiros são relacionados com o funcionamento interno do veículo como servidor *Fog* na VFC. O outro conjunto de objetivos envolve a dinâmica da rede veicular, e diz respeito principalmente à distribuição das aplicações para que elas possam ser executadas na VFC dentro dos requisitos esperados. Quanto ao funcionamento interno dos veículos, pretendemos:

- identificar as relações entre parâmetros de processamento e rede que indiquem os requisitos que os servidores *Fog* devem ter para participar da VFC;
- determinar pelo menos um algoritmo de execução distribuída de aplicações para que os veículos possam atuar como servidores *Fog* mantendo os requisitos das aplicações.

Já em relação à distribuição das aplicações na rede veicular, pretendemos:

- analisar, avaliar e propor pelo menos um mecanismo para o funcionamento de uma VFC completamente distribuída com suporte à disponibilidade, eficiência e balanceamento de carga.

1.5 METODOLOGIA

Pelas suas características esta pesquisa foi realizada através de uma abordagem quantitativa e tem uma natureza de pesquisa aplicada. O problema de pesquisa foi especificado e delimitado através de, pelo menos, dois processos, complementares entre si, de revisão sistemática de literatura. Depois disso, essas especificações foram descritas através de pressupostos e princípios de design que procuram descrever e até mesmo determinar o funcionamento de clientes e servidores em sistemas VFC totalmente distribuídos, além de delimitar cenários em que esse funcionamento é aplicável para atender aos princípios de design. Assim, esta tese ganha seu caráter descritivo. Seguindo esses princípios de design, a distribuição de recursos computacionais foi realizada através de um leilão aberto de primeira oferta e primeiro preço. Por fim, o funcionamento de clientes e servidores foi avaliado e, através dessa avaliação, foram determinados os cenários dando a esta pesquisa, um caráter experimental. As avaliações foram realizadas a partir de parâmetros e métricas relevantes na literatura de VFC. Os principais parâmetros escolhidos foram: densidade de veículos, taxa de clientes em relação à quantidade de servidores, tamanho das requisições, e taxa de requisições. As principais métricas coletadas são relacionadas com clientes (latência de serviço, taxa de resposta, erros de rede, enfileiramento de requisições), e com servidores (enfileiramento de requisições, sobrecarga de processamento).

1.6 ESTRUTURA DESTA TESE

Esta tese está estruturada de forma a apresentar os conceitos básicos necessários à compreensão da evolução das redes veiculares em direção à VFC no capítulo 2. Depois disso, discutimos, minunciosamente, trabalhos relacionados com FogWise, especialmente no que diz respeito a modelos de programação, modelagem, simulação e *testbed Fog*; APIs, algoritmos, técnicas de Machine Learning (ML), otimização e orquestração de recursos em *Fog*; além de características, parâmetros e métricas de interesse na seção 3. A seção 4 apresenta os detalhes do arcabouço FogWise. Em seguida, a seção 5, apresenta e discute a avaliação da proposta. Esta tese termina no capítulo 6, que contém as considerações finais, limitações e os próximos passos que podem ser realizados a partir do estado atual do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção contém os conceitos básicos para a compreensão deste trabalho. Inicialmente serão apresentados conceitos relacionados a IoT, especialmente Internet of Vehicles (IoV), descrevendo desde algumas arquiteturas possíveis até propostas de funcionamento. A seguir, serão apresentadas tecnologias basilares como computação *Cloud*, como *Edge* e *Fog*, buscando destacar suas características, diferenças, e funcionamento básico. Também serão apresentados os modelos de aplicações. Por fim, serão apresentadas algumas técnicas e algoritmos de ML que podem ser usadas nesse contexto, destacando as situações ondem podem ser aplicadas e quais são os possíveis mecanismos.

2.1 A INTERNET DOS VEÍCULOS

Por volta de 2011, relatórios da Cisco estimavam que atualmente estaríamos observando um despontar claro nas tecnologias de veículos conectados(MAI; SCHLESINGER, 2010; MAI; SCHLESINGER, 2011). De acordo com esses relatórios, mais de 300 milhões de carros conectados estariam gerando mais de 400 *PetaBytes* de dados em redes móveis por volta de 2022. Essas previsões não se concretizaram na velocidade esperada e as estimativas, um pouco menos otimistas, apontam um crescimento na quantidade e no tráfego dos veículos autônomos, integrando-os ao ecossistema IoT e de redes móveis no 5G (CERWALL et al., 2020). Atualmente, outras tendências se concretizaram e surgiram outras tecnologias que, juntas, formam o que hoje conhecemos como IoV (BARBOSA; JúNIOR; DIAS, 2020).

Na década de 90, surgiram os primeiros padrões de comunicação entre veículos, já antecipando o crescimento da frota de veículos e a possibilidade de utilizá-la com infraestrutura de comunicação, seja para serviços de utilidade pública, segurança ou outras aplicações (AL-SULTAN et al., 2014). Desde então, os veículos e as tecnologias evoluíram. Os veículos ganharam potência mecânica, além de vários tipos de sensores e recursos tecnológicos. Também possuem poder computacional equivalente aos melhores computadores e não têm restrições de potência elétrica nem de comunicação como é previsto na Internet das Coisas (AL-SULTAN et al., 2014; SOUA; TOHME, 2018). Essas características permitem que uma série de aplicações cotidianas

possam ser utilizadas dentro de veículos.

Num primeiro momento, os veículos ganharam vários sensores para monitoramento de suas funções internas. Os sensores monitoram o estado do motor (temperatura geral, nível de óleo e refrigerador, conectores), a pressão dos pneus, o estado dos sistemas mecânico, elétrico e hidráulico (freios, aceleradores); e até mesmo o conforto e segurança dos passageiros (cinto de segurança, nível de atenção, estilo de pilotagem, reflexo). Os dados dos sensores são concentrados na Unidade Concentradora de Controle Eletrônico *Gateway Electronic Control Unit (G-ECU)*. As gerações mais recentes contêm sensores externos, como radares, câmeras e *lidar* (radar baseado em laser), que ajudam em tarefas como estacionamento e detecção de obstáculos.

A evolução da engenharia de tráfego identificou a necessidade de utilizar os dados internos dos veículos para tornar o trânsito mais seguro e eficiente. Dessa forma, foi necessário criar e adaptar tecnologias de comunicação para que os veículos pudessem tanto enviar esses dados para a nuvem, quando utilizar esses dados de forma eficiente, da mesma forma que IoT.

Um exemplo disso, são aplicações colaborativas para cálculos de rotas. Num primeiro momento, as rotas foram calculadas apenas a partir da distância e do tempo de trajeto, procurando minimizar essas métricas. Com o tempo, provedores de serviços passaram a monitorar os padrões de direção dos motoristas, seja se valendo de dispositivos específicos para rotas, seja de *smartphones* nos quais os motoristas fornecem a informação explicitamente, e com isso puderam calcular rotas baseadas em preferências pessoais (atalhos), condições das pistas, paisagens, e até mesmo no fluxo de veículos dependendo do horário.

As vias também foram "aumentadas". A IoT tem uma variação específica e dedicada no cenário de Vehicular Ad Hoc Network (VANET). As vias, as esquinas e interseções são preenchidas com sensores, *beacons* e pontos de acesso de rede com os quais os veículos se comunicam para coletar informações ambientais: as RSU. Esses *beacons* coletam e difundem informações de tráfego, dos outros veículos e do ambiente para alimentar os algoritmos de decisão. Atualmente, além de ser apenas fonte de informações, as RSU, são pontos de acesso com capacidade de comunicação e computação completa, e podem prover conectividade sem fio, através do padrão 802.11p para os veículos, além de armazenar informações e até mesmo processar dados de sensores (SATOH, 2013).

Num futuro próximo, os motoristas podem não ser mais necessários. Atualmente, percebemos muitas evoluções em termos de Controle e Automação, além de Inteligência Artificial, e os veículos autônomos passam a se tornar uma realidade cada vez mais próximas (SCHOETTLE;

SIVAK, 2014). A ausência de motorista gera uma consequência imediata: além dos sensores internos, os veículos precisam monitorar o ambiente com muita precisão para tomar as decisões em grande escala (rotas) mas especialmente em pequena escala (conversões, curvas, desvios e prevenção de acidentes).

Sucessivas avaliações identificaram que utilizar apenas os sensores pode não ser tão eficiente. Assim, os veículos também trocarão informações críticas relacionadas com o tráfego e direção, necessitando, inclusive tomar decisões em menos 100 ms, tempo que envolve troca de dados completa, processamento e atuação, sobre a vida das pessoas em caso de acidentes (AL-SULTAN et al., 2014; EZE; ZHANG; LIU, 2014; LEE et al., 2014). Ciência e indústria, juntos através de entidades padronizadoras como European Telecommunications Standards Institute (ETSI), Federal Communications Commission (FCC) e 3rd Generation Partnership Project (3GPP), têm unido esforços para desenvolver e aperfeiçoar tecnologias para que veículos possam se comunicar e trocar os dados dessa gama de sensores (internos e do ambiente) para construir um sistema de transporte inteligente e até mesmo autônomo (Intelligent Transportation Systems (ITS)).

Mas, além dos sensores, os veículos também precisarão trafegar dados volumosos oriundos de navegação web, áudio, vídeo sob demanda, videoconferência e até mesmo jogos online de alta definição(CERWALL et al., 2020). Isso deriva da evolução das características dos próprios usuários, através da utilização cada vez mais massiva de *smartphones* e multimídia. Esses dois tipos de aplicações, dados críticos e dados em rajadas, funcionam de forma diferente, e possuem requisitos distintos. A princípio, elas dependem de tecnologias sem fio robustas com grande controle e tratamento de erros além de um bom suporte à mobilidade física (velocidade) dos veículos.

O padrão sem fio 802.11p, que no cenário de Vehicular Ad Hoc Networks (VANETS) é conhecido como Dedicated Short Range Communications (DSRC), apresenta suporte para essas características, especialmente através de suas características físicas e de acesso ao meio. A combinação entre DSRC e Wireless Access In Vehicular Environments (WAVE), tecnologias consideradas padrão para VANETS, pode atender aos requisitos do tráfego de dados dos sensores e dados de segurança entre os veículos, além dos dados críticos usados para tomada de decisão com baixíssima latência especialmente em um salto, porque tratam-se de dados pouco volumosos com poucos *bytes* (Kenney, 2011). Essa combinação ainda apresenta problemas para lidar com o tráfego de grandes volumes de dados, em múltiplos saltos ou em redes muito densas ou muito esparsas devido às suas características de *frames* e transmissão e podem

necessitar de mecanismos mais sofisticados de roteamento (JI et al., 2016).

Isso acontece porque o padrão WAVE se vale de mecanismos de inundação (*flooding*) no envio das mensagens. Isso pode ser benéfico, uma vez que vários veículos podem receber a informação sobre o ambiente ao mesmo tempo (notificações de acidentes, buracos, frenagens bruscas) facilitando a disseminação de informações para o trânsito inteligente. Por outro lado, o mesmo mecanismo gera uma ocupação das bandas sem fio, impedindo que várias notificações ou informações sobre essas sejam trocadas dependendo do volume de veículos.

Aplicações de navegação web e multimídia geram tráfego volumoso e podem necessitar de garantias de conexão e sessão, que não podem ser asseguradas por DSRC/WAVE. Esse tipo de aplicações precisam de protocolos de roteamento mais avançados que aqueles destinados a Mobile Ad Networks (MANET). Grande parte desses protocolos possuem restrições para funcionar em VANET e podem se limitar a regiões com alta densidade de veículos muito próximos. (AL-SULTAN et al., 2014).

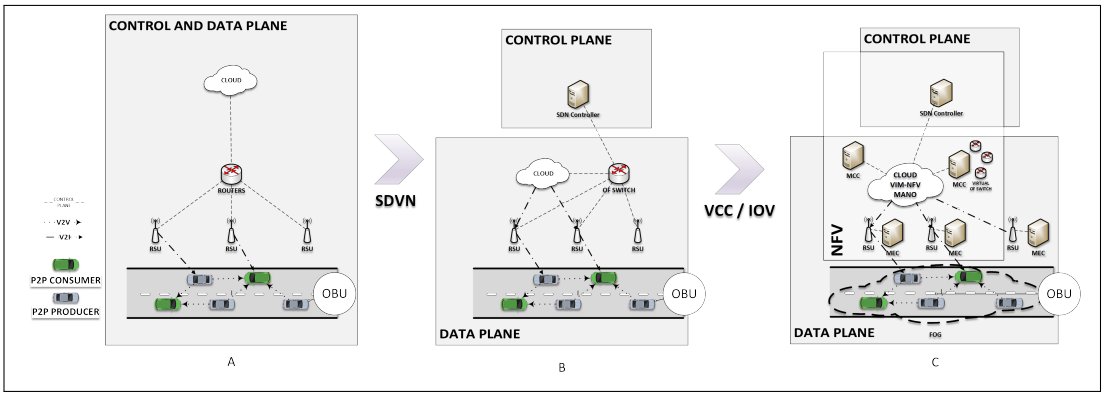
Uma alternativa seria utilizar algoritmos de roteamento baseado em características de mobilidade baseada em Global Positioning System (GPS) e preferências dos usuários. Esse tipo de algoritmo se vale especialmente da previsão de posicionamento para um planejamento antecipado da entrega de pacotes, reduzindo erros e perdas, aumentando a confiabilidade da rede e a vazão disponível para as aplicações (JI et al., 2016). Outras tecnologias como *Long Term Evolution (LTE)/Worldwide Interoperability for Microwave Access (WIMAX)* podem dar essas garantias, inclusive com acesso à aplicações de nuvem e à Internet, mas apesar de apresentar um maior alcance, vazão e até confiabilidade, o Round-Trip Time (RTT) pode se tornar elevado para aplicações de segurança das vias (AKYILDIZ et al., 2016; HUANG et al., 2012).

Baseado em todas as características apresentadas para VANETS, especialmente pelas características que podem estar próximas de IoT, com tráfego de dados de sensores, até tráfego de multimídia, que necessita de um suporte tecnológico completamente diferentes, misturando as necessidades de comunicação entre veículos e um acesso à internet, convencionou-se chamar esse conjunto de requisitos e tecnologias de Internet dos Veículos (IoV). IoV pode ser definida, então, como:

> "uma rede aberta baseada em multi-humanos, multi-máquinas, multi-veículos e um ambiente coordenado. O sistema é controlável, gerenciável, operacional e confiável. Ele sensoria, reconhece, transmite e computa as informações dinâmicas, estáticas, complexas e volumosas dos humanos, veículos, redes de comunicação, e da infraestrutura do tráfego usando tecnologias avançadas de comunicação e processamento. A informação fornecida/criada é

> utilizada para resolver os problemas da computabilidade, escalabilidade, e sustentabilidade do serviço inteligente de gerenciamento e informação em uma rede heterogênea. Finalmente, ele ajuda a efetuar/compreender a integração profunda entre humanos, veículos e ambiente".
>
> (YANG et al., 2017, p. 4)

Figura 1 – Evolução de VANETs



Fonte: Elaborada pelo autor (2021)

A Figura 1 demonstra a evolução em termos mais conceituais do que práticos, do que se esperava da consolidação de cada fase das redes veiculares até a IoV. Num momento inicial, esperava-se que veículos trocassem dados entre si e com a infraestrutura para coletar e fornecer dados prioritariamente sobre o tráfego. Neste momento, predominavam os modos de comunicação V2V e V2I. Num segundo momento, esperava-se uma interação maior com nuvem e até mesmo IoT. Veículos poderiam se comunicar diretamente com a nuvem, através das redes celulares, ou poderiam utilizar a infraestrutura das RSUs para isso. Observamos a inclusão do modelo Vehicle To Everything (V2X) onde veículos podem se comunicar com quaisquer dispositivos ao seu redor. Por fim, as previsões atuais indicam um modelo de rede mais auto-gerenciada e composta por outras tecnologias como computação *Edge*, computação *Fog* e NFV, consolidando o conceito de IoV.

2.2 MOBILIDADE E DINÂMICA DO TRÂNSITO E DAS REDES VEICULARES

As discussões sobre o trânsito, seus problemas de congestionamento e sua dinâmica não são temas novos. Com a emergência das redes veiculares, eles voltaram à tona como temas de pesquisa importantes para as novas tecnologias que surgem (HAO et al., 2017; MOURADIAN et al., 2017; BOUKERCHE; ROBSON, 2018).

Inicialmente, é preciso considerar que os veículos podem participar de cenários com altíssima conectividade, quando o trânsito está denso, e outros em que os contatos e comunicações são

raras, seja em locais pouco movimentados das cidades, ou nas rodovias (BENAMAR et al., 2014). Por muito tempo, os pesquisadores buscaram modelos de mobilidade que pudessem viabilizar a comunicação entre veículos, assim como era feito com os padrões de mobilidade dos usuários móveis, mas as tentativas se mostraram infrutíferas (ALTAYEB; MAHGOUB, 2013; ZHU et al., 2010; XIAO; KUI, 2015). Outros pesquisadores até sugerem que a dinâmica de tráfego é tamanha que requer uma complexidade computacional elevada(FENG et al., 2020; BARIK et al., 2021), e apenas modelos de mobilidade não são suficientes (ZHU et al., 2018b).

Para viabilizar as comunicações veiculares é preciso levar em consideração elementos como: características percebidas internamente e externamente pelos sensores, controle de mensagens e latência percebida, qualidade de links, quantidade de dispositivos vizinhos e previstos (OUYANG; ZHOU; CHEN, 2018; YANG et al., 2017). Sendo assim, esse tema requer uma discussão mais aprofundada pois cada um desses aspectos impacta diretamente no tipo de aplicações que podem executar dentro e fora dos veículos, além do tipo e, especialmente a quantidade de dados, que eles podem trocar.

Acreditamos então, que num sistems de transporte completamente inteligente, ou seja, na realização de um ITS totalmente autônomo e voltado para otimização constante, a mobilidade e a dinâmica serão tratadas de maneira diferente pois:

- os veículos terão conhecimento sobre suas rotas, assim como as preferências dos usuários para sair daquela rota padrão caso seja necessário, então não será preciso utilizar modelos de previsão de rotas;
- os veículos poderão trocar informações de rotas uns com os outros, além de enviar para RSU ou mesmo para a nuvem, e essas informações poderão ser utilizadas para reconfigurações das redes de comunicação no *backhaul*, no *backbone* e até mesmo para o controle das características das redes sem fio (HAQUE; ABU-GHAZALEH, 2016);
- algoritmos poderão utilizar essas informações então será possível fazer o controle fino das mobilidades e das redes;
- os próprios veículos poderão participar dessa decisão rodando esses algoritmos internamente através de mecanismos de computação distribuída, como é o caso da computação *Fog*.

Nas próximas sessões, apresentaremos o funcionamento e discutiremos os requisitos e as consequências de algumas das aplicações previstas para veículos, desde aqueles que ainda

dependem de um motorista, passando pelos momentos de transição até chegar aos veículos autônomos. Depois disso, serão apresentadas as tecnologias de suporte para essas aplicações, especialmente aquelas necessárias para que os veículos participem da cadeia de computação.

2.3 ARQUITETURA DE COMUNICAÇÃO NA INTERNET DOS VEÍCULOS

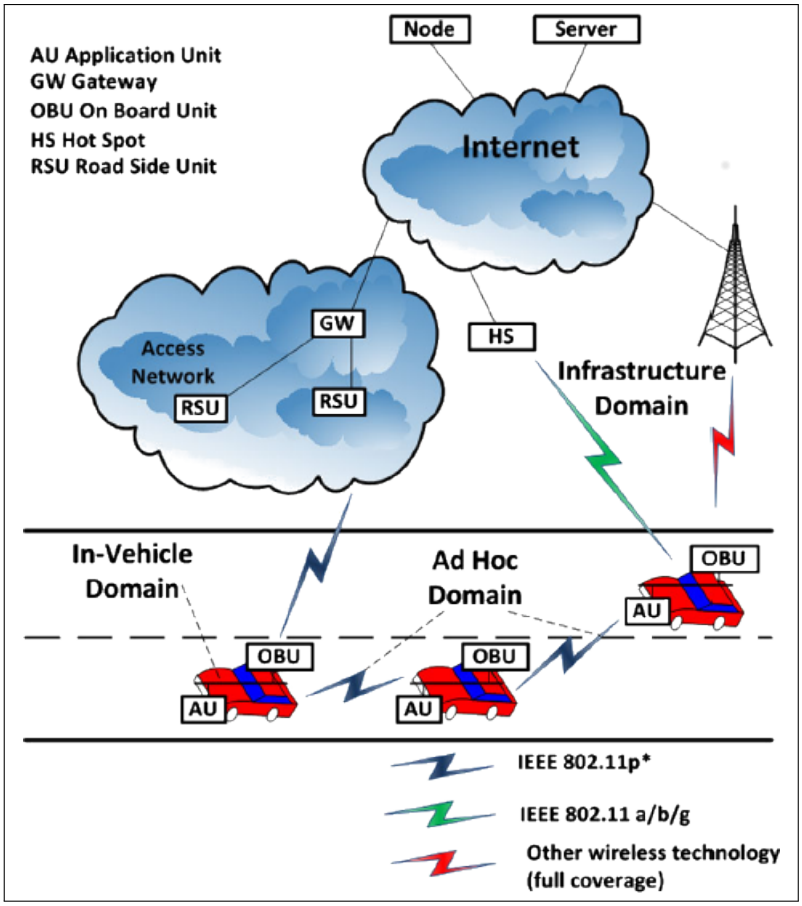
As aplicações em IoV trocam dados através de de uma arquitetura de comunicação dividida em níveis. Os principais são (AL-SULTAN et al., 2014; YANG et al., 2017) apresentados nas figuras 2 e 3:

- comunicação intraveicular
- comunicação veículo-veículo V2V
- comunicação veículo-infraestrutura V2I

A comunicação intraveicular se destina prioritariamente à coleta de dados dos sensores internos e externos do veículo na G-ECU. Ela demonstra indicativos fortes de estar evoluindo de uma comunicação cabeada, fortemente baseada em protocolos de comunicação industrial, para um esquema mais maleável, escalável e sem fio, semelhante a IoT. Para um esquema de comunicação mais robusto, em que os veículos trocam dados entre si e com a nuvem, a G-ECU é substituída pelo conjunto On-Board Unit (OBU)-Application Unit (AU). A OBU é responsável por prover comunicação, especialmente utilizando 802.11p, para a AU, que executa as aplicações dentro do veículo. O tráfego interno não se limita aos dados de sensores do veículo, podendo ser estendido a dados dos próprios usuários internos e externos, além de aplicações IoT (SOUA; TOHME, 2018).

A comunicação veículo-veículo pode ser resumida como a comunicação entre duas ou mais OBU que, num nível mais alto, pode ser a comunicação entre aplicações rodando nesses veículos. Na forma mais simples, veículos utilizam V2V para enviar notificações uns aos outros sobre situações para decisão imediata (buracos, desvios, frenagens bruscas, acidentes iminentes). Num modelo mais avançado de ITS, os veículos podem utilizar dados de outros veículos para melhorar seus próprios algoritmos internos de tomada de decisão. Esse tipo de comunicação depende muito da quantidade e da duração do contato entre os veículos que, por sua vez, depende da tecnologia utilizada (ZHU et al., 2010).

Figura 2 – Arquitetura de Comunicação IoV



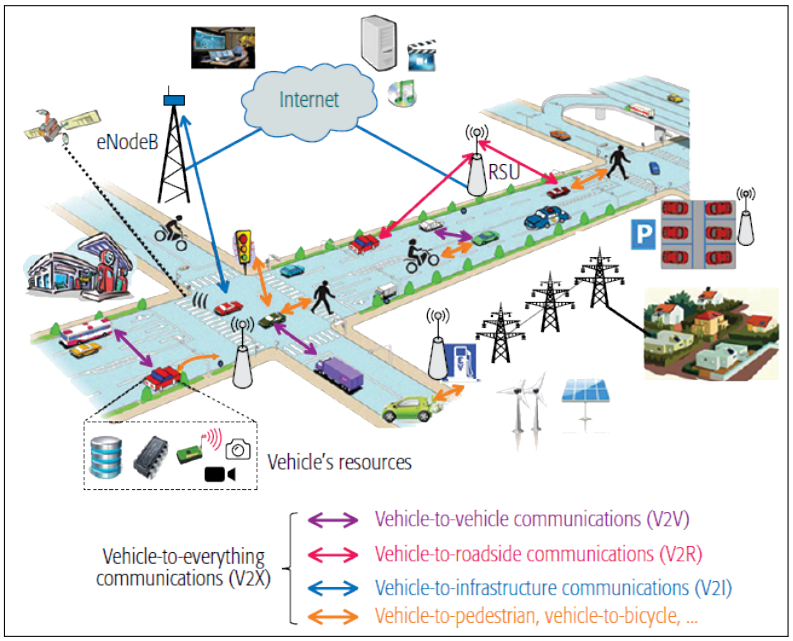
Fonte: (AL-SULTAN et al., 2014, p. 382)

Por fim, a comunicação veículo-infraestrutura tem dois modelos. No primeiro, representa a comunicação entre OBU e RSU, também conhecida como Vehicle To Roadside (V2R). No segundo, pode ser a comunicação entre o veículo e uma estrutura de rede sem fio de alta cobertura como LTE ou WIMAX, que em alguns trabalhos é reconhecida como V2I. Boa parte da literatura aponta uma tendência de que as comunicações em IoV utilizem prioritariamente o modelo OBU-RSU. A investigação sobre qual dos dois modelos se encaixa melhor para IoV ainda é um tema quente de pesquisa, e as conclusões apontam que cada um atende melhor a um tipo de aplicações e/ou situações.

Por fim, existe um consenso de que a realização completa da IoV envolve todos os tipos de comunicação possíveis. A figura 3 demonstra os tipos de comunicação apresentados, mas deixa as tecnologias futuras em aberto. Para o desenvolvimento completo da IoV os veículos podem precisar se comunicar com dispositivos IoT, *smartphones* dos usuários, bicicletas, sistemas de iluminação, e muitos outros tipos de tecnologias possíveis. Para representar esse tipo de comunicação em que o veículo troca dados com tudo o que está ao seu redor, foi cunhado o termo

V2X. Todas essas comunicações devem orquestradas de forma que os recursos estejam sempre disponíveis e, no fim, os veículos autônomos possam sempre tomar as melhores decisões.

Figura 3 – Arquitetura de Comunicação IoV



Fonte: (AMADEO; CAMPOLO; MOLINARO, 2016, p. 99)

2.4 APLICAÇÕES NA INTERNET DOS VEÍCULOS

A literatura apresentam dezenas de aplicações que podem compor um ITS. Esta seção destaca aplicações relacionadas com veículos autônomos, cooperação veicular, segurança e conforto dos usuários. São aplicações que causam impacto externo e dos quais os veículos podem participar através de trocas de dados entre si e com outros dispositivos a sua volta.

Um veículo completamente autônomo é a composição de várias aplicações. Existem 5 níveis de autonomia para veículos (YANG et al., 2017). No nível 0 (zero), o veículo precisa de total controle de um motorista. No nível 1 (um) o veículo possui apenas uma função autônoma, como assistência a estacionamento. No nível 2, o veículo tem, pelo menos, duas funções autônomas. No nível 3, ele possui muitas funções autônomas sob determinadas condições. É o caso de algumas montadoras que em 2018 já ofereciam versões de veículos com direção completamente autônoma e até mesmo um suporte ao Piloto Automático Adaptativo, no qual o veículo pode ser configurado para seguir outro que, por sua vez, pode estar sendo conduzido por um humano ou ele mesmo ser completamente autônomo. A partir do nível 3, o veículo deve ter um conjunto de regras éticas bem definidas, que serão especialmente utilizadas em

situações que envolvem a vida dos passageiros, mas também dos pedestres e dos ocupantes de outros veículos. Por fim, no nível 4, o veículo está completamente autônomo e faz tudo isso com segurança, inclusive o cálculo de rotas, controle de velocidade, e estacionamento.

Na tabela 1 (AL-SULTAN et al., 2014), classificamos as aplicações em dois tipos: aquelas voltadas para o usuário e aquelas voltadas para veículos. A grande maioria dessas aplicações é baseada apenas em notificação, ou seja, *push*. Nesse modelo, um veículo, ou um conjunto deles, coleta um dado do ambiente e dispara a notificação para os vizinhos baseada nos sensores e algoritmos internos dos veículos. As aplicações de segurança, nas duas colunas, e as aplicações da seção de Extensão de Sinalização, se encaixam nesse tipo de aplicações por notificação.

Tabela 1 – Aplicações para a Internet dos Veículos

Usuário	**Autonomia**
Condução: expansão de visibilidade, clima, vídeo e videoconferência, jogos digitais.	Condução: piloto automático, piloto automático adaptativo, cálculo autônomo de rotas, *platoon*.
Segurança: resgate de emergência, notificação para reparo ou *recall*, alerta contra roubos, furtos e sequestros.	Segurança Pública: Aproximação de Ambulâncias ou Policiais, Prevenção de Colisão, Sinalização pós colisão, Linha verde.
Serviços: notificação de serviços próximos às vias, comércio local.	Extensão de Sinalização: curva perigosa, colisão, área de velocidade reduzida, acidente, condição da via, frenagem brusca necessária, mudança de faixa, necessidade de conversão.

Fonte: Elaborada pelo autor (2021)

As aplicações das seções de condução, nas duas colunas, além da seção de serviços, podem funcionar com base num modelo semelhares a *Web* atual, ou seja, utiliza fortemente o modelo IoT baseado em nuvem. Para executá-las, os veículos dependem fortemente de seus sensores internos e de esquemas de requisição-resposta. Para as aplicações de condução, na coluna de condução autonômica, os veículos utilizam seus sensores e algoritmos internos, mas frequentemente podem fazer requisições aos veículos vizinhos, ou mesmo à nuvem para completar, validar ou verificar suas decisões passadas e futuras. A mesma descrição pode ser aplicada à seção Serviços. A seção de condução, na coluna de usuário, contém aplicações que dependem fortemente de serviços de nuvem, exceto pela aplicação de "Expansão de Visibilidade"que depende completamente da troca de dados entre veículos (V2V) de forma que veículos vizinhos, especialmente veículos no início de filas, enviam imagens ou mesmo vídeos para os outros de forma que os usuários possam ter uma visão expandida dos arredores.

2.5 COMPUTAÇÃO EM NUVEM E NA BORDA DA REDE

Desde seus primórdios, a IoT se utiliza fortemente da computação em nuvem. As aplicações rodando em dispositivos IoT enviam seus dados para a nuvem diretamente ou através de *relaying* como meio de gerar análise e visualização desses dados em larga escala. Além disso, algumas aplicações, especialmente aquelas que não dependem de dispositivos muito limitados, usam a nuvem como um meio para processar requisições e posteriormente realizar atuação. Para isso, uma infraestrutura de nuvem contém diversos mecanismos internos para lidar com demandas elásticas, que podem ser um requisito da IoT conforme a quantidade de dispositivos aumenta.

De acordo com o NIST (MELL; GRANCE et al., 2011),

> computação em nuvem é um modelo para permitir acesso ubíquo, conveniente e sob demanda através de uma rede a um conjunto compartilhado de recursos (ex. redes, servidores, armazenamento, aplicações e serviços) que podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor de serviços. (*tradução livre)*

Além disso, ela possui cinco características (autoatendimento sob demanda, amplo acesso à rede, *pooling* de recursos, elasticidade rápida, serviço medido), três tipos de serviço (software como serviço - Software As A Service (SAAS), plataforma como serviço - Platform As A Service (PAAS), infraestrutura como serviço - Infrastructure As A Service (IAAS)) e quatro modelos de implantação (nuvem privada, nuvem comunitária, nuvem pública, nuvem híbrida). No contexto de 5G, esses elementos são levados ao seu limite em meio à quantidade de serviços e aplicações necessárias para tratar todas as necessidades das redes e dos usuários (AKYILDIZ et al., 2016). A consequência mais imediata é a sobrecarga das redes de acesso à nuvem, dependendo da densidade das aplicações. A isso, se adiciona as intermitências das redes sem fio, cuja vazão e latência de rede são impactadas pela densidade dos dispositivos. Dessa forma, as aplicações são afetadas em termos de Quality of Service (QOS) e Quality of Experience (QOE) por conexões instáveis e intermitentes à nuvem.

No contexto mais recente, onde as aplicações e os usuários são móveis e muitas vezes dependem de informações localizadas, as aplicações têm sua QoS degradada devido a maior dificuldade nesse acesso e isso gera um impacto negativo na experiência do usuário. Algumas aplicações possuem características locais, como sensores de umidade, temperatura e câmeras

de segurança, mas, mesmo assim, a dependência da nuvem faz com que essas informações tenham que trafegar longas distâncias físicas, ou mesmo lógicas, em termos de roteadores, para serem consolidadas e se tornarem utilizáveis, especialmente quanto se trata de agregação de dados com outras aplicações (YI; LI; LI, 2015). Além disso, a mobilidade de usuários e aplicações pode gerar *handovers* frequentes, resultando em necessidade de muitas reconexões, ou de grandes esforços para manter sessões ativas, afetando as aplicações.

Uma vez detectados, esses problemas começaram a ser amplamente abordados e investigados na academia a na indústria. Inicialmente, foram investigados algoritmos de otimização para os problemas isoladamente seja na utilização e controle de banda da rede de acesso; nas próprias características da computação em nuvem, como *scaling*, ou mesmo na utilização eficiente dos recursos da rede sem fio, como alocação e canais e micro-células (CHÁVEZ-SANTIAGO et al., 2015). Recentemente, os esforços foram redirecionados para a própria atualização do paradigma de computação em nuvem, através dos conceitos de computação *Edge* e *Fog*.

Isso significa que, em vez de procurar soluções em termos de acesso, houve a transição em direção à extensão da própria nuvem. No caso, *Edge* e *Fog*, adicionam componentes de mobilidade e localização, o que melhora o fornecimento dos serviços contribuindo, especialmente, para a redução de latência de rede e, consequentemente, dos serviços. A computação *Edge* pode funcionar de duas formas básicas: MCC e MEC (YI; LI; LI, 2015). Elas diferem em relação ao escopo e ao tipo de serviço que prestam.

MCC estende a computação em nuvem adicionando especialmente o componente de mobilidade. Trata-se de utilizar *Cloudlets*, que são infraestruturas de nuvem com capacidades completas implantadas a pouquíssimos saltos do usuário móvel, mas que têm tamanho reduzido em relação a *DataCenters* e são especialmente talhados para receber computação ou armazenamento de dispositivos móveis. Com isso, os dispositivos realizam *offloading* de suas atividades e podem economizar recursos computacionais. O componente móvel de MCC vem da possibilidade migrar recursos, especialmente na forma de Virtual Machice (VM), entre *Cloudlets* de forma que o recurso sempre esteja logicamente próximo do usuário em termos de saltos. Isso pode se dar através de migração vertical, quando o recurso é migrado da infraestrutura de nuvem centralizada até a *cloudlet*; ou migração vertical, quando o recurso migra diretamente entre *cloudlets*.

Já MEC é uma tecnologia iniciada pela indústria, e largamente padronizada pelo ETSI. Podemos considerar que é um complemento ao MCC, mas que tem a característica especial de ser implantado diretamente na infraestrutura da rede e conter as caraterísticas específicas e

padronizadas para servir tanto a usuários quanto à própria rede. Assim como MCC, MEC prevê a migração de recursos, mas nesse caso, ele migra dentro da infraestrutura ou entre federações, além de suportar outras tecnologias além de VM para executar seus serviços. Isso é possível devido à padronização das interfaces de comunicação e gerenciamento, além do uso de outras técnicas de virtualização, como os contêineres. Alguns autores advogam que MEC deve ter capacidade de funcionar independentemente de uma conexão com a Nuvem centralizada e, por isso, tem uma capacidade maior de prover serviços relacionados com alguma atuação mais imediata. Nesse contexto, se encaixam aplicações como processamento de dados, *analytics* e análise de tráfego (de veículos ou de dados da rede).

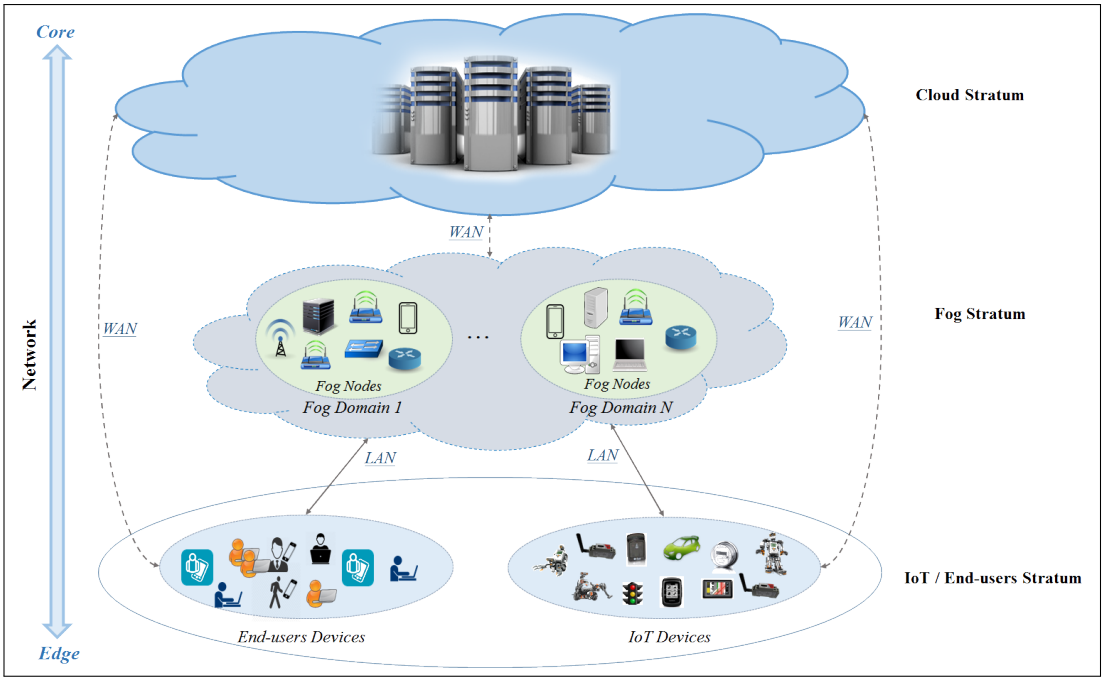
A maior parte dos trabalhos, como (MOURADIAN et al., 2017) e (YI; LI; LI, 2015) conceituam *Fog* como uma composição entre MCC e MEC. O conceito de *Fog* surgiu com a CISCO em 2012 com o IOx: uma arquitetura capaz de executar em dispositivos estrategicamente posicionados dentro da rede, e que possuem a capacidade de rodar aplicações sobre um sistema hospedeiro. Dessa forma, *Fog* se aproxima ainda mais que MEC e MCC dos usuários ou das aplicações que precisa executar.

A Figura 4 apresenta a relação entre a arquitetura em camadas com a representação de uma infraestrutura de nuvem centralizada, um estrato *Fog* e os dispositivos. A figura já contém a sobreposição entre os conceitos de MCC, MEC e *Fog*. Destacamos, na figura, as conexões e a hierarquização comumente encontradas nesse tipo de interação. Os dispositivos da borda (*edge*) da rede, que estão todos incluídos na definição de *Fog*, provêem algum tipo de serviço diretamente para os dispositivos dos usuários ou para os dispositivos IoT. A figura prevê que eles tenham algum tipo de conexão com a infraestrutura de nuvem centralizada. A Figura 4 também sugere que os dispositivos *Fog* sejam organizados em domínios, que podem ser vistos como uma interação transparante entre eles. Um detalhe que não está tão claro na figura, mas que está previsto nas definições de *Fog*, é a transparência de serviço, ou seja, o dispositivo final não deve saber se está sendo servido pela nuvem ou pela *Fog*.

(YI; LI; LI, 2015) propõe com uma definição de *Fog* como sendo:

> um cenário em que um grande número de dispositivos heterogêneos (sem fio e às vezes autônomos) onipresentes e descentralizados que se comunicam e potencialmente cooperam entre eles e com a rede para executar tarefas de armazenamento e processamento sem a intervenção de terceiros. Essas tarefas podem ser para suportar funções básicas de rede ou novos serviços e aplicativos executados em um ambiente de *sandbox*. Os usuários que alugam parte de seus dispositivos para hospedar esses serviços recebem incentivos para isso. (*tradução livre*)

Figura 4 – *Cloud*, *Edge* e *Fog*



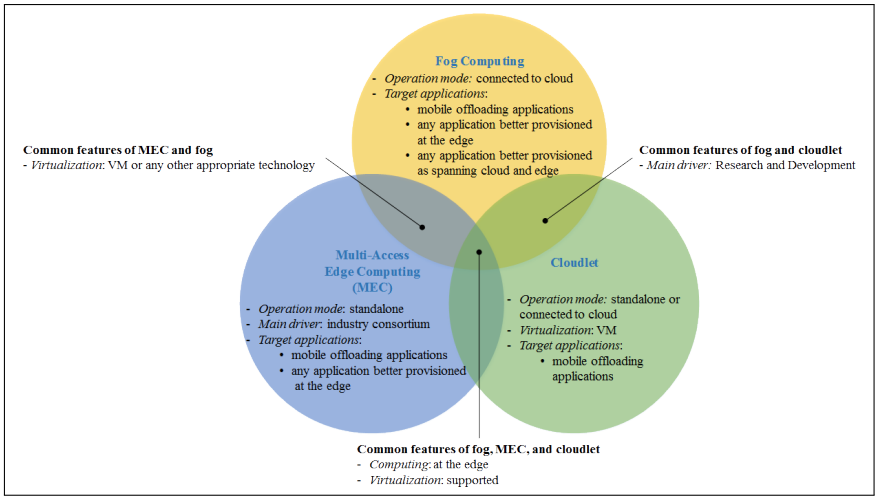
Fonte: (MOURADIAN et al., 2017, p. 6)

De acordo com a definição, *Fog* deixa de depender de uma tecnologia específica para se tornar um cenário com dispositivos heterogêneos que participam da computação. Além disso, prevê a execução de aplicações de forma cooperativa. A Figura 5, extraída de (MOURADIAN et al., 2017), apresenta quatro diferenças entre *Fog*, MCC e MEC colocando:

- MEC como sendo altamente padronizado e adotado pela indústria, enquanto MCC e *Fog* estão mais atrelados à academia e à pesquisa;
- MCC dependente de VM, enquanto *Fog* e MEC podem utilizar outras tecnologias como contêineres;
- MCC dependente da infraestrutura de nuvem centralizada para funcionar, enquanto MEC e *Fog* em modeo *standalone*;
- MCC como tendo foco em *offloading* de aplicações de usuários, MEC como provedora de aplicações de usuário e da própria rede; e *Fog* estendendo o alcance, podendo variar entre todos os níveis.

Relatórios mais recentes da Cisco também propõem uma separação de tarefas entre *Cloud* e *Fog*, especialmente no contexto de IoT (CHIOSI et al., 2012). A *Fog* fica responsável por receber e processar dados nas proximidades dos dispositivos, quase em tempo real; executar

Figura 5 – Diferenciação de tecnologias para computação na borda



Fonte: (MOURADIAN et al., 2017, p. 7)

aplicações com necessidades em tempo real e analyctics, com resposta de poucos milisegundos; prover armazenamento temporário; enviar dados sumarizados para a nuvem. Por sua vez, as infraestruturas de nuvem fica com funções específicas como receber e agregar os dados recebidos da nuvem, analisar dados comparando fontes diferentes, participar do controle dos dispositivos *Fog*, enviar regras ou padrões de comportamento e/ou processamento.

A proximidade física ou lógica entre as aplicações e a nuvem, realizada através do conceito de *Fog*, contém vários benefícios. Entre eles destacam-se:

- redução da latência para próximo de 0;
- economia de banda, especialmente nas redes de acesso, uma vez que os dados são sumarizados antes de serem enviados à nuvem, ou podem nem mesmo ser enviados;
- previsão do aumento da segurança dos dados, uma vez que ocorre a redução do tráfego pela rede evitando problemas clássicos como *tampering*, manipulação de DNS, entre outros;
- além do aumento da disponibilidade e da confiabilidade, através de elasticidade e escalabilidade na borda da rede.

2.5.1 Computação *Fog* e IoV

Propomos, então, a conexão entre esse conceito de *Fog* (YI; LI; LI, 2015) e o conceito de IoV contido em (YANG et al., 2017) através do conceito de VFC. Por definição:

> "VFC é uma arquitetura que utiliza uma multidão colaborativa de clientes usuários finais ou dispositivos de borda próximos ao usuário para realizar uma quantidade substancial de comunicação e computação. Além das características da nuvem, como fornecimento de dados, computação, armazenamento e serviços de aplicativos para os usuários finais, o VFC se diferencia de outras técnicas existentes por sua proximidade com os usuários finais, distribuição geográfica densa e suporte para mobilidade."(HOU et al., 2016)

Disso podemos extrair uma infraestrutura com altos níveis de automação em todos os aspectos de forma a poder funcionar dependente ou independentemente da Nuvem. Além disso, os dados são processados de forma transparente e os resultados são usados para agir tanto no ambiente como na própria rede. Acreditamos que a única forma de atingir a realização completa de IoV é através da consolidação da VFC.

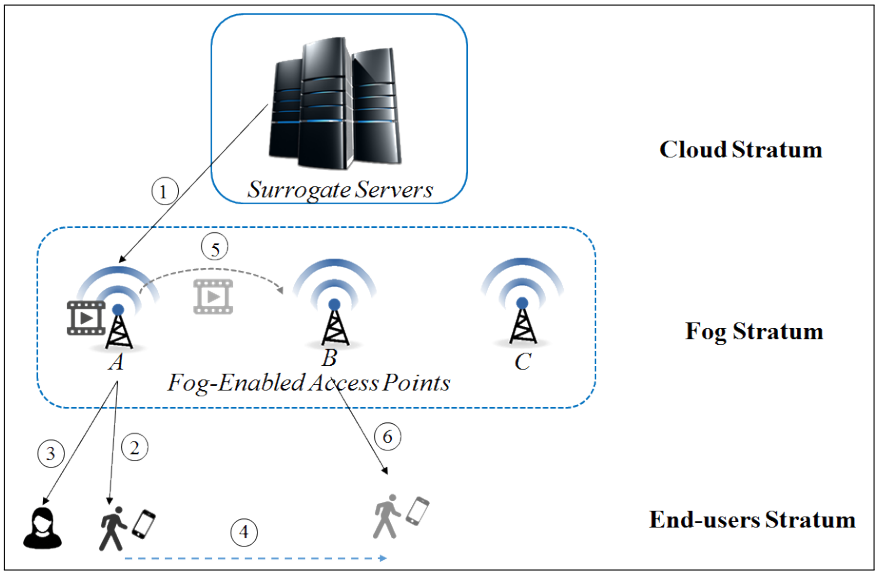
Ela surge a partir das limitações dos tecnologias formadoras dando um suporte cada vez maior a aplicações veiculares. Isso acontece devido às suas características intrínsecas de mobilidade, uma vez que os próprios servidores, os veículos, estão em movimento enquanto fornecem seus serviços. Ao mesmo tempo, tal adequação limita seu uso universal. A mesma mobilidade pode impedir conexões mais longas pois pode complexificar o *handover*.

Um dos principais objetivos da computação *Fog*, pela sua própria definição, é prover computação ubíqua. (YI; LI; LI, 2015) Para tanto, esse tipo de computação requer uma série de características:

- modelo de programação e/ou Application Programming Interface (API) bem determinados, com suporte a processamento e armazenamento persistentes ou transientes, além de agregação de dados e processamento paralelo/síncrono dependendo da aplicação;
- distribuição de aplicações e serviços, seja através de VM, virtualização de aplicação (contêiner), lojas de aplicativos ou mecanismos de compilação/interpretação entre arquiteturas de forma que qualquer dispositivo com capacidade de processamento/armazenamento possa lidar com tarefas uns dos outros;
- regulamentação e utilização eficiente de recursos de rede, desde o cabeamento (fibra ótica, eternet) ao espectro eletromagnético (LTE, Wi-fi, rádios cognitivos), especialmente dando suporte a conexões intermitentes e de baixa vazão;
- uma ou mais entidades certificadoras que garantam requisitos de segurança como: aplicações seguras, dispositivos seguros, além de dados confidenciais e íntegros.

Dentre todas as aplicações que podem se beneficiar da *Fog*, as aplicações IoV se destacam devido aos seus requisitos restritos, tanto em termos de vazão quanto de latência. Destacamos alguns exemplos relacionados com distribuição de conteúdo para os usuários, na Figura 6, e de processamento de dados de sensores, na Figura 7. No primeira caso, os dados são mantidos em *cache* próximo ao usuário. Assim, conteúdos populares podem ser deixados na borda da rede, em servidores de propósito específico, como MCC, ou em quaisquer dispositivos, como o caso de *Fog*, ou migrados de acordo com a necessidade. Nesse caso, se observa o benefício da redução de banda nas redes de acesse (ponto 1), além das características de redução da latência de serviço, e da manutenção das sessões, no caso de *handover* (pontos 2 a 6).

Figura 6 – Caso de uso: conteúdo

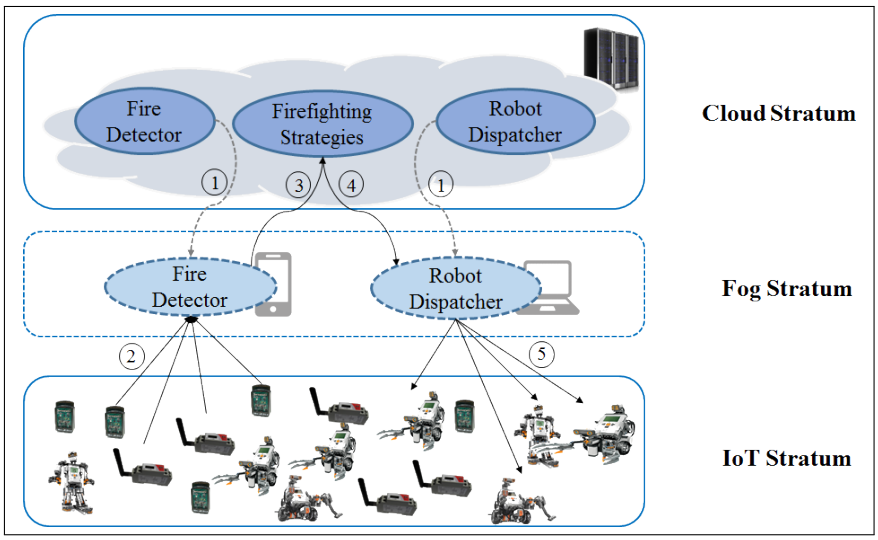


Fonte: (MOURADIAN et al., 2017, p. 8)

No caso de aplicações IoT, apresentadas na Figura 7, *Fog* provê benefícios em termos de atuação imediata. Além disso, a adoção de *Fog*, e a sua atuação próxima a dispositivos IoV, permite que os fabricantes mantenham os dispositivos tão simples, limitados e baratos quanto possível, uma vez que as necessidades de comunicação e processamento podem ser simplificadas. Na figura, também é possível observar alguns relacionamentos entre *Fog* e a Nuvem. As aplicações de Detecção de Incêndio e Controle de Robôs, possuem uma contraparte na nuvem e outra na *Fog*. Dessa forma, ela consegue tratar os dados de forma rápida, especialmente no caso de atuação imediata, mas também consegue realizar consolidação dos dados na nuvem, seja para determinas novas formas de atuação, novas formas de processamento dos dados, ou até mesmo cruzar dados com outras aplicações. No caso, as duas aplicações estão interligadas. Quando a aplicação de detecção de incêndio detecta uma ocorrência, a proximidade das

aplicações devido à *Fog* faz com que a atuação seja realizada com os robôs de forma quase instantânea.

Figura 7 – Caso de uso: IoT



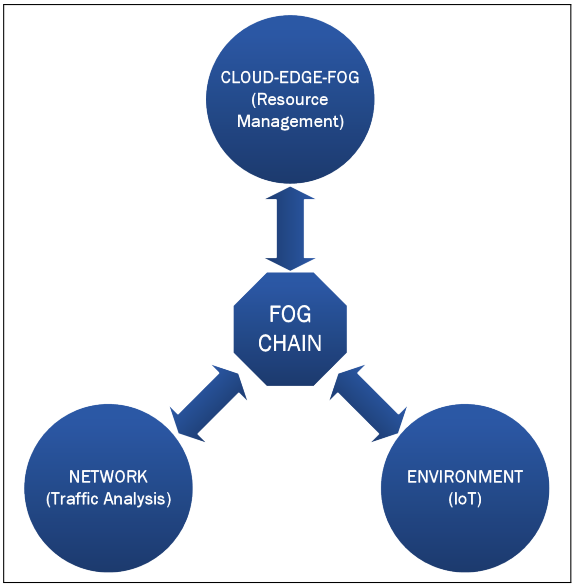
Fonte: (MOURADIAN et al., 2017, p. 9)

A essa dinâmica inerente à computação *Fog* que gera agregação entre aplicações e seus dados para gerar valor para todas elas e para a própria rede, damos o nome de cadeia de computação *Fog*, que está apresentada na Figura 8. *Fog* deve ter uma autonomia em nível tal que permita com que todo o processo de computação, além da coleta de dados, e o gerenciamento de recursos da rede, sejam transparentes, como uma extensão dos conceitos de IoV e *Fog* adotados nesta proposta de tese. Para isso, a junção e a expansão dessas definições implica na participação dos dispositivos finais, que podem ser os veículos, *laptops* ou mesmo *smartphones* nessa cadeia de computação.

2.5.2 Migração de aplicações e tarefas (tasks)

Todos os benefícios que *Fog* pode prover para as aplicações e para a rede vêm com um custo de complexidade. A mobilidade é considerada a maior geradora dessa complexidade, que gera a necessidade de controlar correta e eficientemente todos os recursos relacionados a rede e computação devido à dinâmica refletida na enorme variação desses recursos conforme usuários ou servidores se movem entre pontos de acesso. Os primeiros métodos de gerenciamento de recursos tratam de um planejamento da migração dos recursos em avanço, ou seja, com previsão do serviço a ser ofertado. (YI; LI; LI, 2015)

Figura 8 – Cadeia de computação Fog



Fonte: Elaborada pelo autor (2021)

Outra dificuldade está na descoberta e negociação de serviços e recursos. As estratégias mais simples estão relacionadas com inundação, onde os serviços utilizam *broadcast* para anunciar sua presença, ou mesmo o controle centralizado, onde uma entidade controladora, semelhante ao controlador SDN, determina quais servidores responderão à requisições específicas. Nos dois casos, tanto na migração quanto na descoberta, os autores sugerem a utilização dos parâmetros computacionais para essa tomada de decisão. O trabalho de (HONG; VARGHESE, 2019) apresenta uma análise detalhada de vários mecanismos utilizados para gerenciamento de recursos em *Fog*.

Boa parte desses mecanismos deriva de sua contraparte na computação em nuvem, garantindo que *Fog* seja uma extensão otimizada para determinadas situações. Na nuvem, os desafios de otimização de migração de serviços e banda acontecem num ambiente de *DataCenter*, dotado de recursos praticamente ilimitados e redes com altíssima vazão. Mesmo com esses recursos abundantes, a qualidade de serviço também é investigada como um desafio. Segundo (AMENDOLA; CORDESCHI; BACCARELLI, 2016), existem 4 técnicas principais de migração na nuvem: Migração para-copia - Stop-and-Copy migration (SaCM); Migração pré-cópia - Pre-copy migration (PeCM); Migração pós-cópia - Post-copy migration (PoCM); e Migração Híbrida - Hybrid migration (HyBM). Elas variam em termos da banda e do tempo em que o serviço fica desativado (*downtime*) durante o processo. A migração contém vários componentes de tempo, que envolvem: cálculo de recursos necessários, planejamento de banda, preparação do ambiente receptor, cópia (pausada ou ao vivo), redirecionamentos de fluxo de dados e migração

do estado.

Todos esses desafios são potencializados quando é preciso utilizar redes sem fio, como é o caso de IoT e IoV. No contexto de Computação *Fog*, o gerenciamento de recursos requer otimização de todos os mecanismos necessários da computação em nuvem. Em seu trabalho (HONG; VARGHESE, 2019) sumarizam vários mecanismos de controle de recursos no contexto de *Fog*. Eles classificam o gerenciamento de recursos em termos de arquiteturas, infraestruturas e algoritmos detalhando, com isso, várias nuances do funcionamento dos mecanismos da interação *Fog-Cloud*. A Tabela 2 destaca os termos e elementos relacionados à cada uma dessas classificações. As informações da tabela, assim como a análise detalhada dos trabalhos relacionados, guiam a escolha de metodologias, parâmetros e métricas de interesse para esta tese.

Tabela 2 – Classificação de Arquiteturas, Infraestruturas e Algoritmos para Gerenciamento de Recursos em *Fog*

Arquiteturas	
Fluxo de Dados	Agregação, Compartilhamento, Offloading
Controle	Centralizado, Distribuído
Locação (*Tenancy*)	Virtualização, Slicing
Infraestruturas	
Hardware	Dispositivos de Computação, Dispositivos de Rede
Software	Virtualização de Sistema, Virtualização de Rede
Middleware	Computação *Edge* Voluntária, Computação *Edge*/*Fog* Hierárquica, Computação *Edge*/*Fog* Móvel, Orquestração de Nuvem
Algoritmos	
Descoberta	Infraestrutura de Programação, Protocolo de *Handshaking*, Mensagens
Benchmarking	Avaliação de Propriedades Funcionais, Benchmarking de aplicação, *Banchmarking* integrado
Balanceamento de Carga	Otimização por enxame de partículas, balanceamento de carga cooperativo, baseado em grafos, primeira pesquisa
Placement	Dinâmico Ciente de condição, Técnicas iterativas

Fonte: Elaborada pelo autor (2021)

2.6 UMA VISÃO DAS APLICAÇÕES IOV COM SUPORTE DE NUVEM OU PROCESSAMENTO

As aplicações listadas na Tabela 1 dependem da nuvem de formas diferentes. Neste momento, destacamos como algumas delas podem funcionar interagindo com a nuvem e com seus derivados. Além disso, aplicações *Fog* podem se adequar a utilizar o poder de processamento dos veículos em vez de recorrer à *Edge* e à nuvem. Nesta subseção, descreveremos algumas aplicações que compõem ITSs e que têm suporte bem determinado tanto da nuvem, quando de *Edge* e especialmente *Fog*.

(HONG et al., 2013) apresenta duas aplicações. Uma delas realiza rastreamento de veículos com suporte à *Fog* chamada BOLO que se adequa muito bem a um modelo RPRA. Quando funciona na nuvem, as câmeras enviam *streams* de vídeo ou imagens com fotos de veículos suspeitos ou que cometeram infrações para a nuvem. Lá, algoritmos de detecção de objetos e reconhecimento facial ajudam a identificar possíveis suspeitos de crimes. Outro algoritmo pode ajudar a extrair os algarismos da foto da placa e transformá-los em texto. Numa aplicação tradicional de nuvem, não é possível controlar as câmeras em tempo real para rastrear o veículo ou mesmo ajustar seus parâmetros, como o foco, em tempo viável. Com suporte da computação *Edge* e *Fog*, veículos podem ser rastreados com muito mais velocidade, pois os algoritmos podem ser executados em Access Points (APs) ou mesmo dentro dos veículos. Podemos estender esse cenário para utilizar as próprias câmeras dos veículos (conhecidas como *dash cams*) nessa tarefa, coordenando uma possível troca de câmeras para manter o veículo suspeito sob vigilância e, possivelmente, alertar autoridades competentes para realizar interceptação.

Outra aplicação, também mencionada por (GIANG; LEUNG; LEA, 2016) e (Wang; Ning; Wang, 2018) trata-se do controle de tráfego de veículos, ou até mesmo "linha verde", que podem ter o suporte da nuvem. Nesse modelo, vários sensores espalhados pelas vias, como câmeras, radares e até mesmo *beacons* 802.11p, capturam a quantidade veículos em cada trecho e enviam para a nuvem. Lá, algoritmos podem calcular estimativas e fazer previsões sobre o fluxo de veículos acionar os semáforos para realizar uma melhor distribuição dos veículos pelas vias. Em alguns cenários possíveis, os algoritmos podem detectar que determinados semáforos que dão acesso a vias principais podem gerar alto volume de tráfego e podem aumentar o tempo que eles ficam fechados. Ao mesmo tempo, podem acionar semáforos que dão acesso a vias coletoras ou arteriais, redirecionando parte do tráfego para outras regiões. Num modelo

integrado como previsto naa definições de IoV e *Fog*, os aplicativos de navegação usados pelos motoristas ou mesmo pelos veículos autônomos podem receber essas informações e alterar seus cálculos de rota. No modelo *Edge*/*Fog*, todos esses processo podem ser otimizados uma vez que os algoritmos podem ser implantados nas RSUs ou mesmo dentro dos veículos. Assim, é possível usar dados locais para tomada de decisão mais rápida sobre ativação de semáforos e cálculos de rotas.

Um exemplo mais recente, sugere que jogos digitais, inclusive aqueles com gráficos avançados 3D sejam executados na nuvem, com possível suporte à computação *Edge* e *Fog* (Cai; Leung; Chen, 2013; KUMAR et al., 2013). Nesse paradigma, os jogos são executados na nuvem e o usuário recebe um *stream* de vídeo e envia os comandos. Em algumas implementações, é possível distribuir a execução dos jogos, especialmente aqueles que não dependem da interação com outras pessoas, para a borda da rede, reduzindo a latência para e permitindo que usuários economizem recursos computacionais de seus dispositivos. Os veículos autônomos são fortes candidatos a executar essas aplicações, tanto pelo aumento do poder de processamento, quando pela falta de necessidade de um motorista, fazendo com que os ocupantes dos veículos estejam livres para os mais variados tipos de tarefas, como os jogos digitais. Esse modelo tem vários atrativos para as empresas como (Mukherjee; Shu; Wang, 2018):

- o jogo se torna um serviço, o que cria novas modelos de negócios;
- independência de plataforma, pois os dispositivos devem ter capacidade apenas de receber *streams* de vídeo não requerendo elevado processamento 3D;
- aumento do controle de privacidade sobre os dados dos usuários, já que eles ficam contidos apenas nos datacenters e não precisam ser mantidos nos dispositivos;
- torna-se mais fácil controlar fraudes nos jogos;
- torna-se mais fácil atualizar os jogos, não sendo necessário distribuir arquivos ou mesmo mídias físicas.

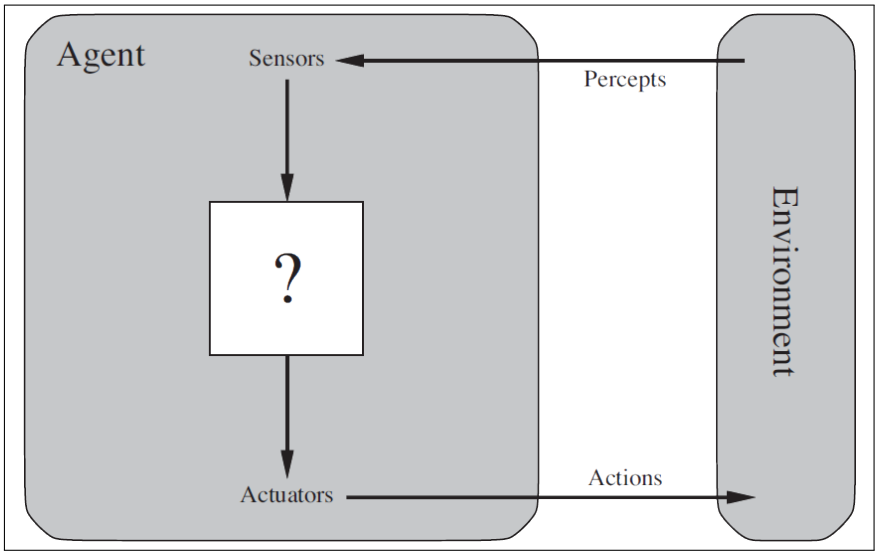
Independentemente da implementação e da proposta, qualquer implementação de aplicações que dependem da Nuvem, *Edge* e *Fog* levam a uma discussão sobre o compromisso de ganhar em desempenho e latência de rede para economizar recursos, enquanto é necessário aumentar a complexidade para instanciar as aplicações e rotear os dados entre eles e para os usuários (OUYANG; ZHOU; CHEN, 2018). Os desenvolvedores das aplicações também têm que

aprender a lidar com essa complexidade, desenvolvendo aplicações cada vez mais modulares e preparadas para auto-gerenciamento dentro da rede (Mukherjee; Shu; Wang, 2018; BARBOSA; JúNIOR; DIAS, 2020).

2.7 AGENTES INTELIGENTES

Para integrar a VFC, veículos precisam de capacidades computacionais maiores, uma vez que podem executar vários tipos de aplicações de usuários e até mesmo fornecer seu poder de processamento para a rede. Acreditamos que eles devem ter capacidades de ser Agentes Inteligentes, monitorando a rede e atuando sobre ela. Um Agente é uma entidade que está imersa num ambiente, seja ele físico ou simulado; que percebe esse ambiente através de sensores; que pode agir no ambiente através de atuadores, conforme demonstrado na Figura 9.

Figura 9 – Um agente reativo simples com seus sensores e atuadores e sua relação com o ambiente



Fonte: (RUSSELL; NORVIG, 2016, p. 35)

Um Agente difere de um sistema clássico de Inteligência Artificial (IA) porque ele não é pensado como um sistema comum que recebe entradas e saídas e tem seu objetivo definido externamente. Ele deve conter toda a lógica interna necessária para realização dos seus objetivos, sejam individuais ou coletivos. Em modelos mais elaborados, um Agente pode possuir um ou mais objetivos e ele escolhe suas ações em função dos objetivos que deseja alcançar.

Um Agente é chamado racional se ele tem condições de escolher a melhor ação possível seguindo algum princípio de racionalidade. Especialmente, dada uma sequência perceptiva, o agente escolhe as ações que melhor satisfazem seus objetivos de acordo com seus conhecimen-

tos internos. Destacamos que a racionalidade do agente dificilmente implica em onisciência porque o agente está limitado pelos seus sensores, pelos seus atuadores e pelo seu algoritmo interno.

Para determinar se um agente atingiu, ou está próximo de atingir, seus objetivos, ele utiliza uma medida de desempenho. O ideal é que essa medida leve em consideração as características do ambiente, especialmente uma sequência de estados que dependam da atuação do agente. Quando essa sequência de estados do ambiente é desejável, o agente pode decidir que teve um bom desempenho. Essa medida, então, deve levar em consideração a sequência de ações do agente realiza atuando sobre o ambiente. Não existe uma medida de desempenho que sirva para todos os agentes. Cada caso deve ser analisado e as medidas devem ser determinadas pelo projeto, levando em consideração as características do agente e, especialmente, do ambiente. Em resumo, um Agente pode ser descrito utilizando a sigla PEAS:

- P - *performance* (desempenho)
- E - *environment* (ambiente)
- A - *actuators* (atuadores)
- S - *sensors* (sensores)

Agentes inteligentes possuem uma série de propriedades relacionadas:

- autonomia;
- adaptabilidade e aprendizagem;
- comunicação e cooperação;
- personalidade;
- continuidade temporal;
- mobilidade.

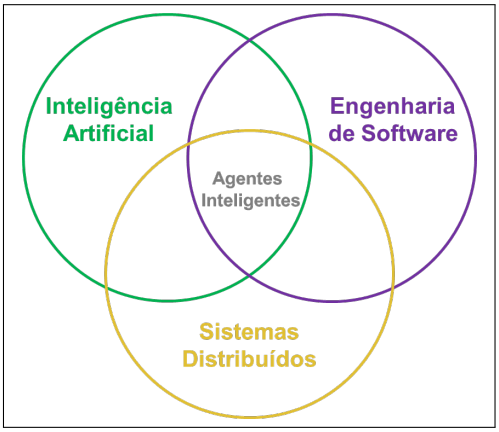
Para ter autonomia, o agente requer uma máquina de inferência e uma base de conhecimento. A adaptabilidade e a aprendizagem são a capacidade de adaptação a novas situações, para as quais não foi fornecido todo o conhecimento necessário com antecedência. Isso pode

ser feito através de implementação de algoritmos de aprendizagem e ou programação declarativa. A comunicação e a cooperação compreendem uma série de algoritmos avançados de sistemas distribuídos, seja em termos de protocolos e padrões de comunicação, cooperação e negociação; de raciocínio autônomo sobre crenças e confiabilidade; ou mesmo arquiteturas de interação social entre agentes.

A personalidade está relacionada com algoritmos de IA que modelam atitudes e emoções dos agentes. Essas técnicas são mais usadas em sistemas ligados ao entretenimento e à imitação das emoções humanas. A propriedade de continuidade temporal está ligada com conceitos de computação pois remete ao tratamento e, especialmente, à persistência dos dados e estados quando necessário. Em termos computacionais, isso pode significar uma necessidade de maior armazenamento secundário ou de maior memória primária. Por fim, espera-se que agentes tenham uma relação intrínseca com a mobilidade em termos de percepção dessa característica, mas também a um suporte à comunicação em mobilidade, especialmente redes sem fio, protocolos de segurança e suporte a mobilidade de código para mudanças e atualizações.

Juntas, essas propriedades dão a dimensão de que um sistema com um agente, e principalmente com muitos agentes, é a interseção entre 3 grandes áreas do conhecimento: Inteligência Artificial, Engenharia de Software e Sistemas Distribuídos, conforme a Figura 10.

Figura 10 – Um agente carrega elementos de Inteligência Artificial, Engenharia de Software e Sistemas Distribuídos



Fonte: Elaborada pelo autor (2021)

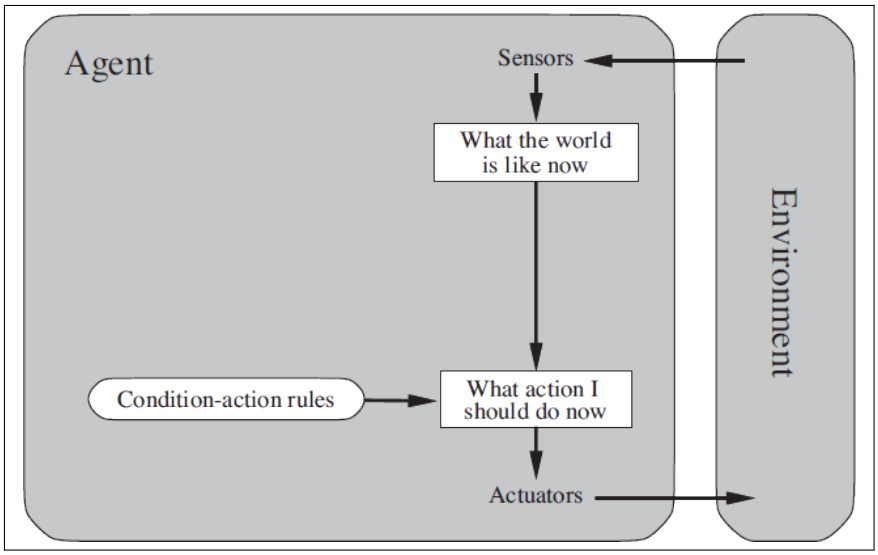
Um agente, em alguns casos, é visto como um mapeamento entre um conjunto de entradas obtidas pelos sensores, e uma ou mais atuações. Tal agente, que parece ter uma tabela interna, não pode ser considerado um agente inteligente, porque sua complexidade pode crescer muito dependendo das características do ambiente. Esse tipo de agente funciona apenas em ambiente que sejam totalmente observáveis, deterministas, episódicos, estáticos, discretos

e pouco complexos. Agentes inteligentes possuem uma arquitetura e podem ser classificados como:

- reativo simples;
- reativo com estado interno;
- cognitivo (baseado em objetivos);
- otimizador;
- adaptativo.

O agente reativo simples, apresentado na Figura 11 percebe o ambiente através de seus sensores e utiliza algum algoritmo interno, fortemente baseado num mapeamento entre entradas e saídas, para escolher uma ação que será realizada. Esse tipo de agente não armazena nenhuma informação relacionada com o ambiente ou com suas ações anteriores.

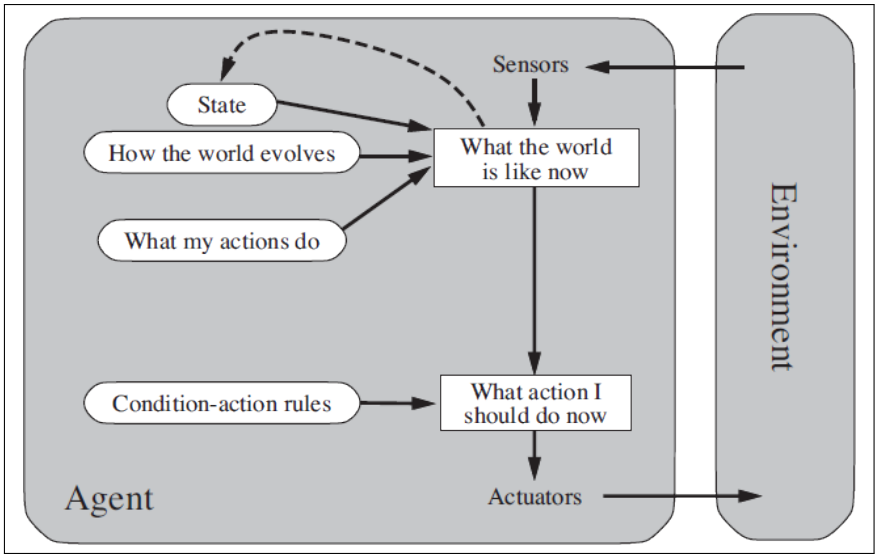
Figura 11 – Agente reativo simples



Fonte: (RUSSELL; NORVIG, 2016, p. 49)

Já o agente reativo com estado interno, apresentado na Figura 12, mantém suas percepções sobre o ambiente armazenadas internamente. O estado interno é atualizado conforme o agente atua ou o ambiente muda, formando o modelo de mundo que o agente constrói. Esses dois primeiros tipos de agente possuem várias limitações em termos de autonomia e precisam ser muito bem planejados para funcionar em ambientes complexos. Por isso mesmo, são utilizados comumente em ambientes pequenos, pouco complexos, totalmente observáveis, episódicos, e deterministas.

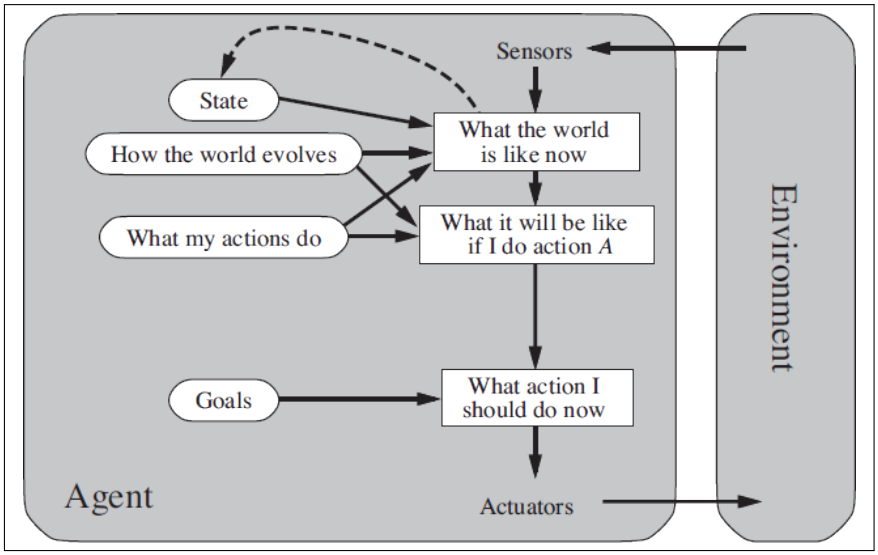
Figura 12 – Agente reativo com estado interno



Fonte: (RUSSELL; NORVIG, 2016, p. 51)

Agentes cognitivos, conforma demonstrado na Figura 13, além de utilizar suas percepções para manter o modelo de mundo, utilizam algoritmos que considerem as mudanças que sua atuação causará no ambiente. Dessa forma, ele pode escolher sua atuação de forma a maximizar sua métrica de desempenho e alcançar seus objetivos. Esse tipo de agente é mais complexo que os agentes reativos, mas também é mais autônomo e flexível. Esse tipo de agente é ideal para ambientes deterministas, em que a resultado da ação pode ser calculado.

Figura 13 – Agente cognitivo com estado interno e objetivo

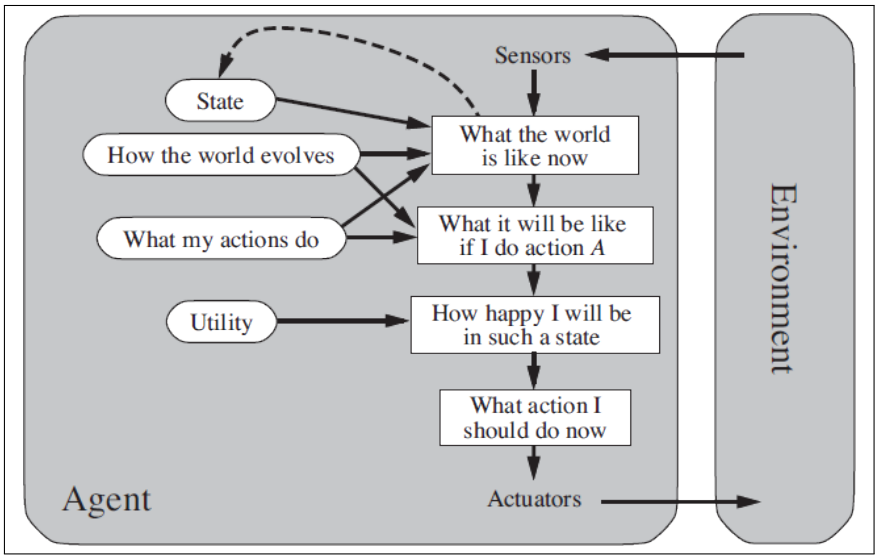


Fonte: (RUSSELL; NORVIG, 2016, p. 51)

O agente otimizador, na Figura 14, tem um componente chamado função de utilidade que deve ser criada para verificar o impacto de suas ações sobre o ambiente. Além disso,

através da função de utilidade, o agente consegue calcular os resultados e fazer previsões sobre comportamentos que não pode calcular com precisão, especialmente quando ele não tem a visão global através dos seus sensores.

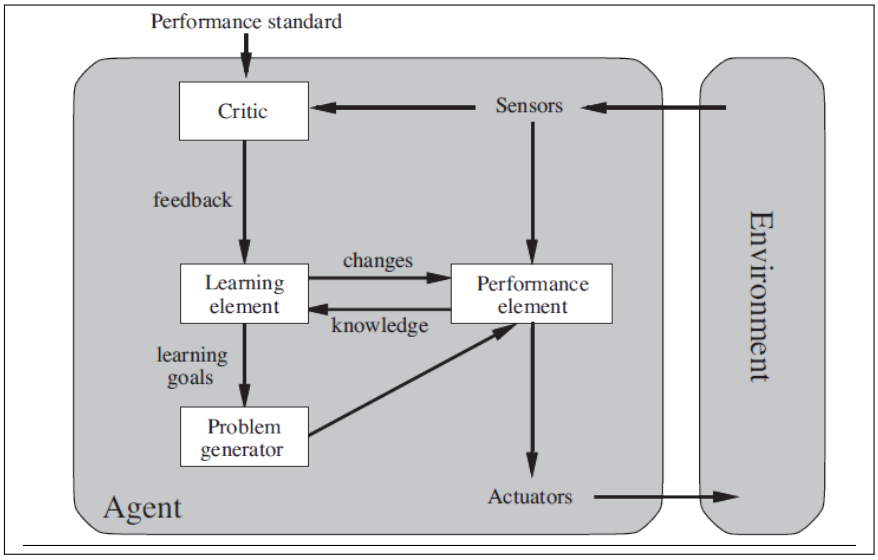
Figura 14 – Agente otimizador com função de utilidade



Fonte: (RUSSELL; NORVIG, 2016, p. 54)

Por fim, existe o agente adaptativo, na Figura 15, que aprende com suas decisões passadas e toma decisões melhores para alcançar seu objetivo. Esse tipo de agente, quando é bem planejado, não tem restrições de ambiente, ou seja, pode atuar em qualquer ambiente que for colocado.

Figura 15 – Agente adaptativo com suporte ao aprendizado



Fonte: (RUSSELL; NORVIG, 2016, p. 55)

2.7.1 Ambiente

O ambiente pode ser definido através de suas classes e suas propriedades. O ambiente pode pertencer às classes de ambiente físico, ambientes de software, e ambientes de realidade virtual, como simulações de ambientes físicos. Quanto às propriedades, o ambiente pode ser classificado quanto à:

- observabilidade;
- determinismo;
- sequencialidade;
- dinâmica;
- continuidade;
- quantidade de agentes;
- complexidade.

Quanto à observabilidade, um ambiente pode ser parcialmente observável ou totalmente observável. Isso depende dos sensores do agente em termos de alcance, quantidade de precisão, por exemplo. Quanto ao determinismo, o ambiente pode ser determinista ou estocástico. O ambiente é determinista quando o próximo estado, ou o estado atual, pode ser completamente determinado a partir dos estados anteriores e das ações do agente. Por outro lado, ele pode ser estocástico quando o estado atual, ou estados futuros, contém um componente aleatório ou probabilístico.

Quando à sequencialidade, o agente pode ser sequencial ou episódico. O ambiente é sequencial quando a ação do agente pode ter impactos futuros, ou seja, a ação do agente altera o ambiente de tal forma que essa alteração pode ser percebido em vários momentos. Por outro lado, um ambiente é episódico quando existe um ciclo sensoriamento-ação bem determinado de forma que o ambiente pode ser percebido em cada episódio. Quanto à dinâmica, um ambiente pode ser dinâmico ou estático. Um ambiente dinâmico pode mudar enquanto a agente está planejando sua próxima ação. O ambiente estático não muda, ou muda apenas de acordo com a ação do agente.

Em relação à continuidade, um ambiente pode ser discreto e contínuo. As duas classificações variam de acordo com o tipo de dados que o ambiente fornece, podendo ser discreto, quando há separação clara entre valores, ou contínuo. Quanto à quantidade de agentes, o sistema pode ser classificado como agente único ou multiagente. No primeiro tipo, apenas um agente percebe o ambiente e atua sobre ele. No segundo tipo, existem vários agentes interagindo com o ambiente e as ações de todos eles podem influenciar umas às outras.

Por fim, um ambiente tem uma complexidade. Não é um fator objetivo, mas que pode ser medido em relação à interações entre as propriedades, além da quantidade e dos tipos de elementos que o formam. A complexidade guia a relação entre o ambiente e agente.

2.7.2 IA distribuída ou Inteligência coletiva

Existem casos em que vários agentes devem trabalhar em conjunto para alcançar um objetivo comum em cenários que são complexos e que apresentam problemas complexos. Em alguns casos, o ambiente pode ser modelado como um agente que interage com os outros. Esse tipo de sistema é conhecido como Multiagente.

Neles, cada agente tem a visão do objetivo a ser alcançado. Esse tipo de sistema comumente não apresenta o conhecimento global sobre o problema, nem uma divisão clara de tarefas, e os agentes devem convergir para alcançar o objetivo de forma totalmente distribuída. Esse esquema é diferente do conceito de um sistema distribuído tradicional, ou mesmo de uma resolução distribuída de tarefas, onde os elementos cooperam para a solução do problema.

2.7.2.1 Teoria dos Jogos

Em cenários como esses, os agentes podem agir para atingir objetivos próprios ou podem cooperar. O conceito mais comum relacionado com esse tipo de sistema é a Teoria dos Jogos. Trata-se de um conceito de análise sobre a interação de agentes inteligentes em um ambiente incerto, além de possíveis outras fontes de observabilidade parcial.

A teoria dos jogos pode ser utilizada de, pelo menos, duas formas: *Agent Design*, e *Mechanism Design*. Em *Agent Design*, a Teoria dos Jogos auxilia na análise das decisões do agente e na computação da utilidade esperada a partir de cada ação ou decisão para maximizar sua função de utilidade. Já o *Mechanism Design* pode ser utilizado para a criação de regras que os agentes devem seguir para atingir algum bem comum e, ainda assim, maximizar funções de

utilidade internas. *Mechanism Design* também é utilizado para construir sistemas multiagentes que buscam resolver problemas computacionais complexos.

A Teoria dos Jogos analisa esses sistemas multiagentes a partir dos tipos de jogos que eles representam. Podemos considerar os seguintes tipos:

- Jogos de movimento único;
- Jogos repetitivos
- Jogos sequenciais

Nos Jogos de Movimento único, todos os agentes tomam a decisão ao mesmo tempo, sem conhecimento das ações dos outros e com o conhecimento que têm do ambiente. Esse tipo de jogo é definido em três componentes: jogadores, que são os agentes tomando decisões; ações, que são resultados das decisões dos agentes; e uma função de recompensa, a qual os agentes usam para calcular a utilidade de suas jogadas.

Desde o começo do jogo, cada agente adota uma estratégia. Em jogos do tipo "movimento único"é comum que exista um componente de aleatoriedade nessa estratégia, especialmente porque não é possível prever o comportamento de outros agentes. A solução do jogo é a estratégia, ou o perfil de estratégia em que cada agente adota uma estratégia racional. Todo jogo tem, pelo menos, um "Equilíbrio de Nash". O equilíbrio é alcançado quando nenhum agente pode se beneficiar mesmo mudando sua estratégia.

Quando os agentes precisam se comunicar, em quaisquer tipos de jogos, denominamos "jogos de coordenação". Eles podem fazer isso através de convenções antes do jogo, ou por negociações para encontrar uma solução benéfica para todos.

Jogos de soma zero são aqueles em que a soma de todas as recompensas é igual a zero. Em muitos casos, esse tipo de jogos é resolvido através de programação linear pois a maximização do objetivo pode ser resultado de restrições que a tornam linear. Tecnicamente, é possível encontrar a solução em tempo polinomial. Por outro lado, encontrar o equilíbrio em jogos de soma não zero pode ser mais complicado e requerer o uso de outros passos como:

1. enumerar os conjuntos possíveis de ações de estratégias mistas variando a quantidade de ações (o que pode ser feito em tempo exponencial);

2. verificar o equilíbrio em cada uma das possibilidades, resolvendo as equações e inequações resultantes das funções de utilidade. Conforme a quantidade de jogadores cresce, o problema aumenta muito em complexidade.

Os jogos repetitivos são o caso mais simples de jogos de múltiplos movimentos em turnos com as mesmas regras. Muitas vezes podem ser reduzidos apenas à ação final, dependendo do tipo de observabilidade e das regras impostas. Diferentemente do jogo de movimento único, aqui, cada agente tem conhecimento das ações dos outros agentes e pode tomar decisões a partir disso.

Os jogos repetitivos possuem um conjunto diferente de análises e estratégias. A mais comum, e que foi provada ser bastante robusta contra outros tipos de estratégias, é a "olho por olho". Nela, agentes simplesmente repetem a ação dos outros agentes.

Por fim, os jogos sequenciais consistem numa sequência de turnos, que não necessariamente precisam ter as mesmas regras. Este tipo é bem representado por uma árvore de decisões, que é conhecida pelos especialistas na área pelo nome de "extensive form". Cada árvore possui: um estado inicial S_0; uma função ou regra JOGADOR(s) que indica qual agente deve agir e quando; uma função ou regra AÇÃO(s) que determina as possíveis ações em um dado momento; uma função RESULTADO(s, p) definida em estados finais do jogo que determina a utilidade de cada agente.

2.7.2.2 Mechanism Design

A Teoria dos Jogos é utilizada, prioritariamente, para analisar ambientes que estão em equilíbrio e tentar demonstrar seu funcionamento. Portanto, não se propõe a controlar esses agentes. A teoria dos jogos, assim, pressupõe que os agentes são completamente independentes e que agem de forma não prevista por qualquer tipo de entidade.

Aqui entra o *Mechanism Design* que, com uma proposta partindo da lógica inversa, busca definir quais regras o jogo deve ter para maximizar a função de utilidade, ou bem-estar, global. Por esse motivo, *Mechanism Design* também é conhecido por "Teoria dos Jogos Inversa". O objetivo é construir sistemas inteligentes baseados em agentes simples com funções predeterminadas.

Formalmente, um mecanismo consiste em:

1. uma linguagem para descrever as estratégias permitidas;

2. um tipo especial de agente, o "centro", que coleta relatórios das escolhas dos agentes no jogo; e

3. uma regra para o resultado, conhecida dos agentes, que o "centro"usa para determinar as recompensas de cada agente, dadas as escolhas de estratégias.

Leilões são o principal exemplo de *Mechanism Design*. Trata-se de um mecanismo de gerência de recursos para distribuir/vender algum bem para um conjunto de compradores/licitantes. No caso mais simples, existe apenas um bem para ser vendido. Os agentes disputam esse bem dando lances de forma que o agente que tiver a maior avaliação interna para o bem, ganha o direito de obtê-la através desse valor. O leilão é baseado na premissa de que algum agente tem uma avaliação interna que compensa pagar um valor acima do valor comum.

O leilão inglês é o modelo mais conhecido. Nele, agentes dão lances crescentes. O Agente Centro, ou leiloeiro, começa pedindo um valor mínimo, também chamado de reserva, que também é o lance mínimo. O leilão começa quando algum licitante consegue pagar aquele valor. Todos os licitantes ouvem os lances uns dos outros e podem decidir, ou não, dar um lance mais alto que o anterior. A partir daí, o leiloeiro acrescenta um pequeno valor em cada rodada, até que nenhum licitante esteja disposto a pagar. Assim, o leilão acaba. Um leilão é dito "eficiente"quando os bens vão para o agente que os avaliam melhor, resultado numa boa distribuição dos recursos. O leilão inglês é eficiente nesse sentido, além de se viabilizar o aumento de receita do licitante. Além dessas, o leilão deve ter outras propriedades que evitem que os agentes formem conluios. Isso pode acontecer quando o leilão não é bem projetado e os agentes conseguem trocar informações, de forma explícita ou implícita, de forma que as maximizações esperadas não aconteçam. Mesmo diante dessas possibilidades, é possível incrementar o leilão e criar mecanismo que impeçam os conluios como:

- um preço inicial mais alto, eliminando participantes indesejados, mas limitando a participação de possíveis agentes confiáveis;

- utilização de lances ocultos de jogada única, aumentando a possibilidade de lances independentes e verdadeiros, ou seja, sem combinação de valores;

- incentivos para a participação de mais licitantes, levando em conta o tempo/processamento perdido por agentes que não consigam dar lances cada vez maiores.

Todos esses mecanismos possuem desvantagens que precisam sem analisadas cuidadosamente de acordo com o planejamento do mecanismo de leilão. Assim, um mecanismo em que os agentes possam usar apenas suas estratégias pre-determinadas, sem precisar se preocupar com possíveis conluios, ou agentes maliciosos, é chamado "Leilão à prova de estratégias". Tal mecanismo deve ser projetado de forma a que seus participantes sejam levados a revelar seus valores reais, criando, assim, uma verdadeiro "bem-estar social". Qualquer leilão pode ser ajustado para se tornar "à prova de estratégia".

Como exemplo, temos que o leilão inglês tem boas propriedades gerais. Entretanto, o agente não é incentivado a demonstrar sua avaliação real do bem leiloado, mas apenas uma avaliação mínima acima do valor inicial. Além disso, o leilão inglês pode ter um alto custo de comunicação, onde todos os licitantes realizam rodadas sucessivas de envio de lances até que a avaliação máxima daquele conjunto de agentes seja alcançada.

Uma alternativa que pode ser aplicada para aliviar esses dois problemas é o "leilão com lances fechados". Nesse tipo de leilão, cada licitante envia um lance ao qual os outros agentes não têm acesso. Dessa forma, ele é incentivado a fornecer sua avaliação real do produto, independente dos lances dos outros. Além disso, a tendência é que o leilão tenha apenas uma rodada, já que um dos agentes enviará seu lance com avaliação verdadeira que é maior que os lances dos outros agentes. Ainda assim, agentes podem gastar tempo/processamento analisando leilões anteriores para estimar possíveis lances que reflitam melhor a realidade global.

Para evitar esse tipo de análise, e ainda melhorar o mecanismo, existe os leilões em que o licitante vencedor para o segundo maior preço em vez do lance que ele mesmo deu. Essa modificação permite que os agentes sejam mais incentivamos a mostrar seus valores verdadeiros pois, no fim, ainda conseguirão ter boa noção do mercado e podem utilizar essa informação nos leilões subsequentes para ajustar sua avaliação interna e o valor do seu lance. Esse tipo de leilão, "de lance oculto e segundo melhor preço", é conhecido como leilão de Vicrey e é bastante usado na construção de sistemas de IA devido às suas propriedades de comunicação e revelação de avaliação real.

Por fim, *Mechanism Design* pode propor mecanismos para a distribuição de "bens comuns". Caso exista um bem comum, porém limitado, que nenhum agente precise pagar para ter acesso, esse recurso tem a tendência de ser explorado de forma que não aumente a utilidade total dos agentes, ou o "bem-estar social". Essa é "tragédia dos comuns". Para superá-la, é preciso criar uma forma de cobrança que maximize o bem-estar comum ou, em outras palavras, que

minimize a utilização errada do bem.

Simplesmente inquirir os agentes pela sua avaliação, ou o quanto eles se beneficiariam daquele bem, não é uma boa estratégia, pois poderia levar todos os agentes a manifestarem valores altos para, assim, terem o acesso desejado ao recurso. Um mecanismo conhecido para lidar com esse tipo de problema é o Vickrey-Clarke-Groves (VCG). A ideia é que cada agente pague uma taxa, ou imposto, equivalente à perda de utilidade global que ocorre devido à presença do agente no jogo da seguinte forma:

1. o "Agente central"pede que cada outro agente informe sua avaliação por receber o item;
2. ele aloca os recursos para um subconjunto dos agentes e usa o resultado para calcular um valor de utilidade somando todas as avaliações relatadas;
3. depois disso, faz várias rodadas de somas das avaliações, em cada rodada, deixa um dos agentes de fora do cálculo;
4. além disso, calcula a utilidade que maximizaria a utilidade global caso esse mesmo agente não estivesse presente;
5. por fim, cada agente paga um imposto/taxa resultante da subtração entre o valor do cálculo da utilidade maximizada (4) e o valor resultante das operações com as avaliações reportadas (3).

Essa taxa/imposto compensa o fato de que alguns agentes não puderam participar do leilão por terem avaliações baixas sobre o bem em questão. Com esse mecanismo, os agentes são incentivados a relatar suas avaliações reais pois todos sabem que o Agente Central tentará maximizar a utilidade global. Assim, o agente participante tende a relatar sua avaliação real pois sabe que o valor do imposto depende apenas dos outros agentes.

O *Mechanism Design* propõe quatro funções para a avaliação dos resultados da alocação recursos baseadas na utilidade dos agentes (BRANDT; CONITZER; ENDRISS, ; WEISS, 2013; LUJAK; SLAVKOVIK, 2017):

- *Bem-estar social Utilitário* : $u(C,S) = \sum_{c,s} V_{c,s}$
- *Bem-estar social Igualitário* : $e(C,S) = min_{c,s} V_{c,s}$
- *Bem-estar social Elitista* : $el(C,S) = max_{c,s} V_{c,s}$

- *Bem-estar social de Nash* : $n(C, S) = \prod_{c,s} V_{c,s}$

A avaliação do bem-estar utilitário consiste na soma das utilidades dos agentes. Torna-se, assim, intuitivo medir a qualidade geral da alocação: quanto maior o resultado da função de utilidade, melhor a alocação ou melhor o bem-estar. Entretanto, a função utilitária falha em mostrar discrepâncias internas ou injustiças na alocação, uma vez que demonstra apenas uma visão geral.

O bem-estar igualitário tem como foco analisar a alocação baseada nos agentes com menor função de utilidade. Trata-se uma análise mais justa, já que demonstra o estado dos piores agentes. Entretanto, algoritmos de otimização baseados apenas na função igualitária possuem complexidade elevada e a análise é conhecida como Problema do Papai Noel (BANSAL; SVIRIDENKO, 2006). A análise deste problema está fora do escopo deste trabalho uma vez que lançamos a métrica apenas para comparação com futuras implementações.

A avaliação elitista contém pouca relação com resultados justos de alocação de recursos. Ao contrário da análise igualitária, a função elitista avalia a alocação em termos dos agentes que conseguiram a melhor utilidade não demonstrando como a alocação realmente aconteceu. Entretanto, pode ser uma análise útil em situações em que o fato de, pelo menos, um agente alcançar um objetivo pode ser um resultado aceitável.

Por fim, o bem-estar de Nash é a função dos produtos das funções de utilidade dos agentes. Ela tende a demonstrar com mais fidelidade o resultado da alocação pois agentes com funções baixas tendem a reduzir os resultados gerais.

2.7.3 Projeto de Agente

Em termos de projeto, (MACAL; NORTH, 2005) propõem uma série de perguntas que devem ser respondidas e que ajudam na criação de um modelo inicial de um sistema utilizando agentes (tradução livre).

1. Qual problema específico deve ser resolvido pelo modelo? Quais perguntas específicas o modelo deve responder? Que valor agregado a modelagem baseada em agente traria para o problema que outras abordagens de modelagem não podem trazer?

2. Quais devem ser os agentes no modelo? Quem são os tomadores de decisão no sistema? Quais são as entidades que possuem comportamentos? Quais dados sobre agentes são

simplesmente descritivos (atributos estáticos)? Quais atributos do agente seriam calculados endogenamente pelo modelo e atualizados nos agentes (atributos dinâmicos)?

3. Qual é o ambiente dos agentes? Como os agentes interagem com o ambiente? A mobilidade de um agente através do espaço é uma consideração importante?

4. Quais comportamentos do agente são de interesse? Que decisões os agentes tomam? Quais comportamentos estão sendo usados? Quais ações estão sendo tomadas pelos agentes?

5. Como os agentes interagem uns com os outros? E com o ambiente? Quão expansivas ou focadas são as interações do agente?

6. De onde os dados do modelo se originam, especialmente aqueles sobre os comportamentos dos agentes?

7. Como podemos validar o modelo, especialmente os comportamentos do agente?

As respostas a essas perguntas dão indicações sobre os objetivos e sobre o PEAS do agente, além de apontar uma arquitetura adequada de acordo com o ambiente previsto. Além disso, elas podem indicar a metodologia para resolver o problema. Em termos de implementação, o agente pode ser implementado em termos de componentes, ambientes e desempenho.

2.8 CONSIDERAÇÕES FINAIS

A IoV é resultado, mas também pode dar suporte a uma série de avanços tecnológicos relacionados com o 5G. Com base nas demandas futuras de serviço, os requisitos de 5G foram definidos com relação às capacidades dos atuais sistemas 4G, da seguinte forma (AGIWAL; ROY; SAXENA, 2016; AKYILDIZ et al., 2016; CHÁVEZ-SANTIAGO et al., 2015):

- aumento de 10 a 100x a taxa de dados do usuário atingindo 10Gbps;
- 100x mais dispositivos conectados;
- alta largura de banda por unidade de área, alcançados pelo gerenciamento dinâmico do espectro;
- redução de latência de 10x, implicando em latência de 1ms fim a fim;

- redução de 90% no consumo de energia, desenvolvendo tecnologias verdes apoiadas por órgãos de padronização;
- aumento da vida útil da bateria 10x e quase 100% de cobertura e disponibilidade ;
- menor custo de implantação e manutenção de infraestrutura .

A IoV será positivamente impactada por se valer de tecnologias avançadas de redes sem fio, controle e automação da rede, coleta de dados de sensores, e virtualização para formação de redes dinâmicas. Por outro lado, uma vez que os veículos podem participar da coleta, distribuição e processamento de dados, eles podem prover recursos ajudando a atender a alguns requisitos como: alta largura de banda por unidade de área, através da da distribuição da transmissão; redução de latência, uma vez que eles mesmos podem prover serviços uns aos outros, bem como aos usuários; e redução no consumo de energia, pois podem coletá-la do ambiente ou mesmo ajudar na geração.

Acreditamos que todas essas melhorias podem ser alcançadas através do atendimento aos dois últimos requisitos: cobertura e disponibilidade, e redução do custo de implantação. Uma vez que os veículos estão aptos a participar da cadeia *Fog*, eles podem ser responsáveis por levar cobertura, serviço e disponibilidade à regiões antes não alcançadas, seja àqueles em que não existem serviços de outras tecnologias, seja tratando parte dos serviços no modelo semelhante ao de micro-cells.

Por fim, acreditamos que seja necessário fazer algumas considerações sobre a relação entre dinâmica veicular e troca de dados da IoV. Por um lado, o congestionamento de veículos pode criar uma situação que viabiliza as comunicações dentro da IoV, especialmente quando elas são gerenciadas apropriadamente. Por outro lado, os sistemas de transporte inteligente, ITS, têm sido projetados para aumentar o conforto, a segurança e também reduzir o tempo de viagem quando possível, especialmente dentro das cidades. A longo prazo, isso fará com que os congestionamentos diminuam, especialmente através de um controle fino e inteligente das rotas de cada veículo autônomo. Com isso, os modelos de previsão de mobilidade podem se tornar defasados e será possível até mesmo controlar a mobilidade de veículos autônomos para facilitar a movimentação.

3 REVISÃO DE LITERATURA

Esta seção descreve, relaciona e critica os principais trabalhos relacionados a esta tese. O conteúdo dos trabalhos gira em torno computação na borda (*Edge*, *Fog*), mobilidade, migração de código e alocação de recursos. Além disso, abordaremos como esses conceitos são trabalhados e avaliados atualmente em termos de modelagem, simulação e implementação. Por fim, demonstramos o estado da arte em termos de VFC e IoV, na qual os veículos participam da computação e quais são os mecanismos e algoritmos envolvidos neste processo. Nosso objetivo é sintetizar princípios de *design*, critérios de avaliação e operação de VFC e IoV nas próximas seções.

3.1 COMPUTAÇÃO *FOG*: MODELAGEM, SIMULAÇÃO E PROVAS DE CONCEITO

Modelagem matemática é, talvez, a melhor forma de começar a investigar um tema de interesse (JAIN, 1990). Mesmo assim, é necessário comparar os resultados com futuras simulações e implementações. Nesse sentido, dois trabalhos buscam investigar o funcionamento de Computação *Fog* através de modelagem e simulação (SARKAR; MISRA, 2016; GUPTA et al., 2017).

Os dois trabalhos são fundamentados no modelo básico de computação em Nuvem, que é baseado em requisição/resposta. O trabalho de modelagem (SARKAR; MISRA, 2016) apresenta modelos para investigar duas métricas, que são básicas em computação *Fog*: latência de resposta e consumo energético. Os autores compararam as métricas em relação ao funcionamento em nuvem e em *Fog* apontando economias reais desses dois recursos no ambiente *Fog*. Assim, eles também conseguem chegar a conclusão de que a diferença de latência e consumo energético entre *Cloud* e *Fog* aumenta proporcionalmente com a quantidade de dispositivos. Entretanto, os autores criam premissas muito simplificadas como é o caso da transmissão de dados entre dispositivos *Fog*, que é dada como insignificante. Devido ao mesmo pressuposto, o trabalho não analisa a questão de mobilidade ou *handover*. É importante destacar que o modelo não usa o processamento dos dispositivos terminais, conforme as definições de *Fog*.

O simulador iFogSim (GUPTA et al., 2017) se apresenta como uma ferramenta para investi-

gação do funcionamento de *Fog* em termos de gerenciamento de recursos. O simulador contém suporte a dispositivos IoT, gerenciamento de recursos e monitoramento energético. Os autores também apontam um funcionamento *Fog* semelhante a (HONG et al., 2013), baseado em requisição/resposta, ou requisição/processamento/atuação, além do modelo de processamento de *streams*, e na descoberta de um dispositivo *Edge* que possa processá-las.

Alguns trabalhos como (HONG et al., 2013) deixam muito claro como *Fog* deve funcionar em algumas situações, conforme demonstrado na seção 2.5.1. Trabalhos como esses são importantes na busca pela computação ubíqua e sua correta avaliação. A análise do funcionamento desse tipo de computação indica caminho importantes para sua avaliação. Os autores apontam e avaliam seu trabalho em termos de latência/alcance-distância da requisição-resposta, e distribuição da carga de processamento em dispositivos *Fog* que, na classificação do artigo, não abrange os dispositivos finais. Os autores avaliam essas métricas em termos de mobilidade, mas não avaliam o gasto energético dessas requisições. Acreditamos que isso indique a falta de um modelo para esse tipo de avaliação Fog na ferramenta Omnet++ (VARGA, 1999). Pretendemos que esta tese contenha as ferramentas necessárias para tal avaliação.

(TEERAPITTAYANON; MCDANEL; KUNG, 2017) prevêem a distribuição completa de processamento até os dispositivos finais. O trabalho discute a possibilidade de usar Distributed Deep Neural Networks (DDNN) para processamento de dados relacionados ao reconhecimento de padrões, que é um tipo de processamento característico de IoT. Para tando, os autores utilizam técnicas de agregação dos vetores de saída e avaliam sua proposta em aplicações de reconhecimento facial. Entretanto, apesar da distribuição, os autores assumem, por inferência, que existe uma redução da latência de processamento das requisições e não demonstram experimentalmente. Também podemos criticar a existência e distribuição de apenas um tipo de aplicação por vez nos cenários avaliados. A ausência de uma dinâmica ou execução de aplicações paralelas precisa ser melhor avaliada e justificada devido à dinâmica e à quantidade de dispositivos móveis e IoT existes e previstos para os próximos anos.

Esses trabalhos contêm as seguintes contribuições para a proposta FogWise:

- buscar o caminho da modelagem e simulação para a compreensão dos mecanismos de funcionamento *Fog* em seus detalhes;
- modelagem do modelo de funcionamento de *Fog* baseado em requisição/processamento/atuação;

- proposta para a investigação do gerenciamento de recursos em VANET, que nos levou a iniciar a investigação a partir da visão interna dos dispositivos
- perspectiva de utilizar os dispositivos finais, no caso os veículos, como extensão do modelo *edge-fog*;
- necessidade de uma investigação mais profunda sobre a participação dos dispositivos finais, buscando determinar e justificar a execução de aplicações paralelas e identificar as características e requisitos de cada caso.

3.2 MIGRAÇÃO DE RECURSOS E CÓDIGO EM FOG

Acreditamos que um dos principais problemas para fazer com que os dispositivos finais, especialmente aqueles com limitações computacionais ou com alta mobilidade, é migração das aplicações e serviços necessários. Assim, investigaremos alguns trabalhos que podem conter diretrizes e contribuições importantes nessa área. Perceberemos que grande parte dos trabalhos discute migração de VM ou se limita a avaliar dispositivos executando um só tipo de aplicação. Alguns trabalhos apontam estratégias mais direcionadas para desempenho, como re-compilações e conteiners.

O trabalho de (SATOH, 2013) realiza uma distribuição de tarefas para processamento de *streams* de dados através de uma variação no algoritmo de MapReduce. Com isso ele consegue realizar processamentos distribuídos abstraindo grande parte das dificuldades relacionados ao desenvolvimento e gerenciamento de aplicações distribuídas. Mesmo assim, ainda é possível identificar a dependência de paradigma. Como qualquer processamento *Fog*, o trabalho prevê processamento próximo à fonte dos dados (dispositivos IoT) e agregação dos resultados, justificando de forma semelhante a distribuição em (TEERAPITTAYANON; MCDANEL; KUNG, 2017). O autor chama a atenção para a diversidade de dados gerados por um conjunto de dispositivos IoT numa determinada área. O autor defende que essa diversidade deve ser analisada para tirar o foco dos dados em si, e focar na distribuição e execução distribuída de tarefas (*"Task deployment rather than data deployment"*). Ele também chama atenção para a necessidade de computação paralela e assíncrona. Acreditados, então, que a questão dos dados e da distribuição deve ser tratada através da utilização de interfaces padronizadas e ou da disponibilização de metodologias padronizadas para tratamento desses dados. Nesse contexto,

a computação *Fog* baseada em uma distribuição de aplicações de forma transparente, fornece uma plataforma para o tratamento desses dados.

O trabalho de (SATOH, 2013) ainda apresenta dados importantes sobre o levantamento de requisitos e princípios de *design*, que serão utilizados e estendidos na proposta FogWise. O autor avalia seu trabalho em termos de duplicação e *deployment*, o que nos dá diretrizes importantes sobre a avaliação da proposta FogWise para formar os componentes do tempo total de serviço.

Alguns trabalhos focam na migração de VM ou na utilização de metodologias avançadas de *deployment*, e possuem contribuições importantes para FogWise em relação ao funcionamento de uma arquitetura desse tipo. É o caso do trabalho de (SHUJA et al., 2017). O foco do artigo é defender a utilização de métodos avançados de *deployment* e compilação para agilizar a execução do código após a migração em *Edge* e MCC. Particularmente, os autores defendem o método Dynamic Binary Translation (DBT), que permite compilação para diferentes plataformas (ex.: x86, ARM), em detrimento dos métodos de virtualização de aplicação, como contêineres, e de migração de VM. A discussão ocorre porque os dois últimos métodos geram *overhead* em relação ao tamanho e carga útil da aplicação. Os resultados são satisfatórios em relação ao desempenho final da aplicação, ficando na casa dos milissegundos, mas o tempo de migração não é trabalhado. A principal contribuição deste trabalho é a demonstração da possibilidade de remoção de *overhead* de computação e, até mesmo de migração de código, através de técnicas específicas.

Mesmo os trabalhos relacionados com migração de VM podem contribuir para a proposta FogWise. O trabalho de (AMENDOLA; CORDESCHI; BACCARELLI, 2016) apresenta várias discussões sobre o processo de migração de VM, dos quais podemos aproveitar técnicas e metodologias. O trabalho indica a otimização do gerenciamento de banda em redes sem fio 5G como solução para a migração. Aponta também que mecanismos de migração são necessários para atingir os objetivos de Ultra-Reliable Low-Latency Communications (URLLC) e defende, inclusive, que as métricas relacionadas com a latência de uma requisição serão mais importantes no futuro do que as próprias relações de vazão entre os pontos de rede. Dessa forma, grande parte da vazão nas redes de núcleo e *backhaul* pode ser deixada para migração de serviços (ex.: VM) e dados multimídia. Assim, uma vez que os serviços estão na borda da rede, as requisições e respostas podem ser distribuídos. Os autores também defendem que já existem mecanismos eficientes para elasticidade de serviços em arquiteturas *Cloud* e MCC, apontando uma lacuna em aberto para que a proposta FogWise trabalhe elasticidade na cadeia

Fog.

Outros autores trabalham a migração baseada em contêineres . É o caso de (BROGI et al., 2017), que apresenta uma arquitetura baseada em provedores e consumidores de serviços, além da caracterização de um *fog node* como *handler* de *streams* de dados de IoT. A arquitetura também apresenta alguns elementos como um controlador de *fog node* (Fog Node Controller (FNC)), Aplicações Autonômicas Autonomic Application (AA) e Controladores de Aplicações Autônomas Autonomic Application Controller (AAC). Apesar de não deixar isso claro em sua implementação, os autores defendem que o dispositivo *fog* deve ter algum nível de lógica autonômica interna, ou seja, algum nível de inteligência baseada em ML, que permita que ele participe dos processos de requisição/liberação de recursos, da elasticidade (*scaling out* e/ou *scaling in*) dos serviços e aplicações da rede, além da distribuição de serviços e do processamento distribuído das aplicações. Também existe a menção à migração online de serviços, ou seja, uma migração que acontece durante o processamento dos dados e, mesmo assim, resulta numa entrega correta e íntegra, semelhante ao mecanismo de MapReduce defendidos por (SATOH, 2013). Apesar de defenderem a participação dos dispositivos finais na computação, os autores utilizam um servidor com 48 núcleos para sua avaliação. Isso faz com que o trabalho fique ainda mais distante do poder de processamento preciso para veículos e especialmente para dispositivos IoT.

O trabalho de (DUPONT; GIAFFREDA; CAPRA, 2017) também apresenta discussões sobre a utilização de contêineres no contexto de *Fog* para IoT. Apesar de não apresentar avaliação, os autores discutem migração vertical, da nuvem para *gateways* IoT, e horizontal, entre *gateways*. O processo de migração é disparado através de uma notificação dos dispositivos IoT de acordo com a área em que estão. Além disso, os autores apresentam casos de uso na área de saúde e na área da indústria, cada um com requisitos diferentes, mas que podem ser beneficiados com *Fog*, especialmente através da agregação de dados, separação de tarefas e redução da carga da rede.

Por fim, (SOUA; TOHME, 2018) apresenta uma arquitetura com a participação dos veículos como dispositivos *Fog*, seja processando dados de outros veículos, dispositivos IoT ou mesmo dos dispositivos dos usuários. Essa arquitetura deve ter auxílio de SDN. Apesar das propostas, a discussão fica num nível muito alto e nem sequer leva em conta os modelos de programação IoT-IoV apresentados pela literatura e nem se preocupa em como essas aplicações são distribuídas entre os veículos.

Os trabalhos sobre migração de código em ambientes *Fog*, contribuem das seguintes formas

para a proposta FogWise:

- possibilidade de que os nós finais participem da cadeia de processamento;
- recomendação para que o foco seja dado em tarefas em vez de dados, o que combina com o conceito recente de Micro Serviços;
- exemplos de requisitos e princípios de design;
- indicação para o desenvolvimento de mecanismos de segurança que garantam computação e transporte seguro dos dados (privacidade, integridade) além de garantia de dispositivos seguros que possam receber a migração;
- possibilidade de planejar a utilização de técnicas avançadas de compilação para transmissão e execução otimizada de código após migração;

3.3 ALGORITMOS E INTELIGÊNCIA NA ORQUESTRAÇÃO DE RECURSOS EM COMPUTAÇÃO *FOG*

A orquestração de recursos e de serviços sempre foi um tema de interesse em todos os níveis da hierarquia de computação em Nuvem. (Saraiva de Sousa et al., 2019). No contexto atual, em que a nuvem migra para a borda da rede, não são apenas dados que trafegam pelas redes de acesso e *backhaul*. No ambiente de nuvem tradicional, baseado em *datacenter* a comunicação entre servidores, e mesmo entre arquiteturas de nuvem, acontece por meio de links de altíssima capacidade e.g. Fibra Ótica. Nesse contexto, a migração de serviços acontece prioritariamente através de migração de VM, que apresentam pouco impacto aos *links*. Esta subseção se limitará a discutir alguns mecanismos mais avançados de migração de serviços e que envolvam algum algoritmo ou inteligência. Os mecanismos iniciais e os mais simples foram apresentados na seção 2.5.2.

O contexto de IoV apresenta mais dificuldades devido a suas características. Os links podem não ser tão rápidos quanto em data centers, e a maior parte das conexões são *wireless*. Além disso, a dinâmica é muito maior, pois os dispositivos são móveis e realizam *handovers* constantes. Esse novo cenário requer estratégias ainda mais avançadas para manter o nível dos serviços atuais e chegar à realização completa de *Fog* e URLLC.

O trabalho (ALAM; TUN; HONG, 2016) apresenta uma análise sobre a decisão de *offloading* num cenário de *Mobile Fog*. Assim como a maior parte dos outros trabalho na área, a

arquitetura é hierárquica, ou seja, parte de uma arquitetura de nuvem centralizada e desce hierarquicamente passando por *Edge* até a *Mobile Fog*. É importante destacar que, assim como em vários trabalhos atuais, os dispositivos finais não participam dos processos de computação e apenas delegam seus processamentos para a *Mobile Fog*. No artigo, o problema do *offloading* e da mobilidade são modelados através da interação entre Agentes Inteligentes. Os agentes monitoram o ambiente e usam a experiência aprendida para entrar e sair em grupos, ou *clusters*, através de um mecanismo de Dissolution and Formation of Groups (DFG). O algoritmo de aprendizagem por reforço é alimentado com informações relativas à execução de um bloco de aplicação, que geralmente é um método ou um conjunto pequeno de métodos. Pra isso, é preciso estimar a execução de um método em termos de RAM, utilização de CPU, duração da execução, e quantidade de execuções. Além disso, os autores destacam que é importante saber se aquele bloco de código é passível de *offload* ou não. Ao final, os autores conseguiram pequenas reduções no tempo de execução de um código e no consumo energético dos dispositivos móveis.

Já (OUYANG; ZHOU; CHEN, 2018) modela a migração de serviços na borda da rede (*Edge*) como a minimização da latência do serviço de longo prazo sob as restrições do orçamento de custo de longo prazo como uma otimização estocástica. Análises posteriores os levam a tratá-la como um Problema de Otimização de Lyapunov. Outra grande contribuição deste trabalho é a demonstração de que não é necessário se basear em padrões de mobilidade dos usuários para essa decisão. A detecção e análise de padrões de mobilidade sempre foi uma das principais alternativas para alimentar as decisões sobre *handover* em redes sem fio. Os autores de (OUYANG; ZHOU; CHEN, 2018) apontam que a análise agora está em um outro patamar e que a dinâmica é tão grande que usar padrões de mobilidade poderia ter um custo computacional elevado.

Assim como (OUYANG; ZHOU; CHEN, 2018), o trabalho de (MAO; ZHANG; LETAIEF, 2016) também utiliza otimização de Lyapunov na decisão de *offloading* em ambiente *Mobile Fog*. O algoritmo é utilizado por sua característica de otimização apenas com um *snapshot* do estado do dispositivo em termos de características de processamento e código. O algoritmo precisa ser alimentado com dados de caráter estocástico para ajudar nessa decisão. O trabalho apresenta essa análise utilizando dispositivos móveis com Energy Harvesting (EH), ou seja, que têm certa capacidade de coletar energia do ambiente de alguma forma e isso impacta a decisão sobre o *offloading*. A maior parte dos resultados leva isso em consideração e demonstra que, em muitos momentos, o dispositivo pode alterar sua decisão de manter o processo de

sua tarefa localmente, distribuir na MEC ou enviar para a nuvem. Os resultados também dão indicações importantes sobre como avaliar arquiteturas desse tipo. É necessário avaliar não apenas as métricas básicas de *Edge-Fog*, como latência e consumo energético, mas é preciso avaliar como as requisições estão sendo geradas e atendidas. Isso envolve avaliar: probabilidade de chegada de uma requisição, probabilidade de a requisição ser atendida, estimativa de que a requisição seja atendida de acordo com um determinado limiar de tempo de resposta, e probabilidade de descarte de requisição. As indicações são importantes, especialmente num cenário de maior mobilidade e dinâmica, como é a característica de IoV que será atendida pela proposta FogWise.

Esses trabalho contém contribuições importantes para a proposta FogWise como:

- Necessidade de utilizar algoritmos avançados de otimização e/ou ML na decisão de *offloading*;
- Necessidade de um suporte à análise sobre migração de código, mesmo que na ausência de mobilidade;
- Necessidade de ir além dos modelos tradicionais de previsão de mobilidade para tratar URLLC;
- descrição de uma entidade na rede, baseada nos padrões atuais do 3GPP, que certifica os dispositivos indicando quais deles estão aptos para realizar *offloading*;
- sugestão do acesso à nuvem e *fog* através de rede Wi-Fi e LTE;
- servidores *fog* podem ser agentes inteligentes agindo ativamente e aprendendo com a dinâmica da rede;
- identificação de certos níveis de recomendações comuns/padronização do funcionamento *Fog* conforme encontrado em outros trabalhos relacionados (SARKAR; MISRA, 2016);
- necessidade de avaliações complementares em termos de avaliação de requisições e respostas, bem como a probabilidade de que essas requisições sejam geradas ou atendidas de acordo com a taxa de distribuição de código.

3.4 PARTICIPAÇÃO DOS VEÍCULOS NA COMPUTAÇÃO *FOG*: COMPUTAÇÃO *FOG* VEICULAR

Vários trabalhos na literatura mencionam a participação de vários tipos de dispositivos na cadeia de computação *Fog*, desde elementos de rede a dispositivos móveis, laptops e tablets (HONG et al., 2013; HONG; VARGHESE, 2019; BROGI et al., 2017; TEERAPITTAYANON; MCDANEL; KUNG, 2017). Nesta seção, discutiremos alguns cenários, condições e situações nas quais os veículos podem participar do *Fog Computing*, especialmente enquanto estão de deslocando nas vias. Os trabalhos desta seção foram selecionados com base em uma revisão sistemática, com dois critérios de inclusão principais: veículos realizam computação; e contêm avaliação de desempenho desse cálculo. Assim, selecionamos quatro artigos. Além disso, um processo de *snowballing* nos ajudou a incluir os dois primeiros artigos (NING; HUANG; WANG, 2019; HOU et al., 2016) porque eles são referenciados nos anteriores e são considerados essenciais para contextualizar os cenários e aplicações em que os veículos podem participar. Por último, coletamos informações de dois artigos (SATOH, 2013; MOURADIAN et al., 2017) que afirmam claramente os requisitos e critérios de avaliação para sistemas de computação *Fog*.

A proposta de (HOU et al., 2016) apresentou seminalmente o conceito de VFC propondo o uso do poder de processamento de veículos para aplicações em Cidades Inteligentes. Os autores apresentaram quatro casos de uso em que os veículos podem participar, seja como infraestrutura de comunicação ou de computação, estacionados ou em movimento. Em seguida, os autores realizaram avaliações de cada cenário para indicar as condições de disponibilidade: situações com variação de densidade, tempo e velocidade. Entretanto, os autores não avaliaram a influência do poder computacional, da largura de banda, nem latência. Os resultados indicam que sua solução atendia aos requisitos de latência em padrões específicos de movimento, densidade e estacionamento. Além disso, há uma tendência para comunicação probabilística e computação semelhante a Delay-Tolerant Netwoking (DTN) (BURGESS et al., 2006).

O trabalho de (NING; HUANG; WANG, 2019) dá uma descrição prática de VFC ao descrever e avaliar um Sistema de Gestão de Tráfego - Traffic Management System (TMS). Os autores motivam seu trabalho com base nas necessidades crescentes de QOS das aplicações. Eles argumentam que pode ser caro usar tecnologias relacionadas à nuvem, como MCC e MEC, bem como instalar novas infraestruturas, seja celular ou RSU. Em seguida, eles descrevem uma arquitetura de três camadas (*Cloud*, cloudlet e *Fog*), mas apontam que os veículos na *Fog* podem ser o elemento de processamento primário da rede. Quando a *Fog* está sobrecarre-

gada, as cloudlets, ou mesmo a nuvem, realizam o processamento para as aplicações. O TMS funciona através do processamento de fluxos de dados de sensores e câmeras de uma região de uma cidade nos veículos ou cloudlets naquela região. Esse processamento gera notificações sobre ocorrências nas vias, como acidentes ou mudanças de rota necessárias e envia de dados consolidados para que a nuvem possa processar as informações em maior escala e atender usuários de todas as regiões da cidade. A avaliação usa dados de rotas de veículos de cidades chinesas que costumam ter alta densidade de tráfego. Os autores convertem a passagem dos veículos por essas rotas em taxas de geração de mensagens ou em nuvens de veículos estacionados. A avaliação foca na distribuição de tarefas entre os veículos e no cálculo do tempo médio de resposta do TMS sob determinadas condições. Os autores comparam sua solução com uma estratégia de distribuição aleatória de tarefas e destacam numericamente quais são as melhorias. O tempo de resposta permanece num limiar entre 200ms e 1000 ms quando os veículos participam da computação. Esse tempo envolve o tempo de troca de mensagens, bem como a própria computação.

(VILALTA et al., 2018) apresentam uma prova de conceito de uma arquitetura em que um veículo controlado remotamente processa dados de sensores internos e externos para determinar suas melhores ações. Embora seja apenas um protótipo em pequena escala, os autores integram vários algoritmos e tecnologias que já são encontrados no mercado para fazer a aplicação funcionar da forma mais realista possível. Isso inclui a execução da aplicação em contêineres num Raspberry PI e o modelo YANG para descrição de serviço além de *middleware* com suporte a REST e JSON. Os autores argumentam que, dessa forma, podem implementar sua solução em qualquer ambiente real. Eles avaliam a capacidade do sistema em termos da quantidade de comandos de controle que precisam de dados provenientes dos sensores. Em média, cada comando é executado em 260 ms, que inclui o processamento do sensor e a transmissão da mensagem.

(TSAUR; YEH, 2018) investigam os problemas de latência gerados pelo aumento no nível de segurança em aplicações da IoV. Os autores argumentam que muitos das aplicações que dependem de notificações podem ter QoS degradado se precisarem preservar a integridade ou a confidencialidade das mensagens, seja por problemas na implementação ou pela necessidade de retransmissão ou renegociação dos protocolos. Para lidar com esses desafios, os autores apresentam um Esquema de Notificação para Comportamento Anormal do Motorista - Driver-Abnormal Notification Scheme (DANS), uma aplicação *Edge-Fog*-IoT para relatar comportamento anormal do motorista. Nessa aplicação, os veículos processam dados dos sen-

sores internos do veículo para verificar o comportamento do motorista durante a condução, bem como processam dados do ambiente circundante. Durante o processamento, a aplicação DANS procura por padrões no nível de atenção do motorista ou na mobilidade do veículo para gerar notificações para outros motoristas, caso detecte um comportamento anormal e que possa gerar um acidente. Devido ao processamento da informação, existe a necessidade de preservar a privacidade do motorista. Para isso, o veículo precisa processar os dados utilizando mecanismos de autenticação com uma Autoridade Confiável - Trusted Authority (TA). Depois da autenticação, o veículo está pronto para processar os dados dos sensores e operações relacionadas à segurança, como multiplicação, emparelhamento bilinear e criptografia simétrica. Os resultados indicam que a maior parte das operações de segurança têm tempo de processamento por volta de 1 ms. Além disso, a latência total de ponta a ponta após todo o processamento e trocas de mensagens é de cerca de 160 ms, o que representa entre reduções entre 80 % e 90 % em relação aos algoritmos comparados.

(NETO et al., 2018) investigam problemas de latência em sistemas de segurança de tráfego quando a aplicação depende da nuvem. Os autores argumentam que esquemas reativos para lidar com crimes e acidentes não são válidos. Para resolver esses problemas, os autores propõem uma Estrutura *Fog* para Segurança Pública Inteligente em Ambiente Veicular - Fog-Framework for Intelligent Public Safety In Vehicular Environment (FISVER). As suas principais contribuições são um sistema para uso de vídeo dos veículos (*dash cam*) para detecção de crimes e situações gerais de ameaça; uma infraestrutura VFC que serve de interface entre os veículos geradores dos vídeos e a polícia; e um sistema de eventos para usuários cadastrados. O sistema possui três camadas: coleta de dados, analytics e processamento de imagens de sensores nos veículos; infraestrutura *Fog*; e Nuvem. Os autores implementaram sua arquitetura usando Raspberry Pi como o elemento *Fog*. Entretanto, eles não avaliam essa arquitetura em um ambiente real e apenas criam uma sequência de imagens que o sistema deve detectar e disparar as notificações necessárias. Eles variam a taxa de imagens geradas usando o que eles chamam de Teste de Implementação Típica, cuja capacidade nominal é de 3600 análises em 10 minutos. Para a avaliação, os autores usam cargas de 30 %, 60 %, 90 % e 120 % da capacidade nominal. Seus resultados indicam que FISVER com processamento nos veículos pode economizar recursos de processamento (27 %), rede (47,99 %) e energia (62,14 %). Além disso, os autores propõem princípios de *Fog* para segurança: detecção rápida e precisa de eventos em tempo real; implantação de forma que forneça serviços em tempo real; integração com *backend* de sistemas da cidade inteligente; sistemas baseados em eventos ou

gatilhos.

(CASTELLANO; RISSO; LOTI, 2018) argumentam que fazer dispositivos *Fog* se comunicarem, especialmente dispositivos móveis, pode não ser trivial. Assim, os autores analisam a aplicabilidade da computação *Fog* em um ambiente real de IoT Industrial (IioT), propondo uma arquitetura que habilita DTN baseadas em *clusters* de dispositivos. Além disso, eles avaliam a solução proposta em um protótipo de código aberto. O artigo apresenta pelo menos duas aplicações: monitoramento e visualização de dados brutos da planta industrial; e filtragem e agregação de dados para analytics. Além disso, os autores projetam a atualização dos algoritmos e modelos para analytics usando os mesmos mecanismos DTN. Inicialmente, eles avaliam a formação de DTNs em uma implementação usando Raspberry Pi e gaiolas de Faraday para simular conectividade intermitente e possíveis desconexões. Com base nos resultados, os autores emulam uma rede usando SDN, Open Vswitch (OVS) e docker para simular as conexões e operação de veículos em movimento que precisam processar dados das aplicações mencionadas. Em seguida, avaliam a probabilidade e a duração dos contatos e como isso pode afetar o funcionamento das aplicações em um ambiente *Fog* totalmente distribuído. Durante os experimentos, a latência de comunicação permaneceu baixa, utilizando tanto o protocolo IP quanto o protocolo DTN, atingindo no máximo 50 ms. Os autores também mostram como o tempo de vida do pacote DTN (*bundle*) e a probabilidade dos contatos influenciam a distribuição do *bundle*. Embora este seja um ambiente DTN, os resultados podem indicar a viabilidade da troca de dados do sensor entre dispositivos em movimento, especialmente em casos onde eles estão agrupados ou com alta mobilidade. Esta solução é adequada para aplicações que toleram os seguintes requisitos: 2s para conectividade de dispositivo, probabilidade de contato maior que 10 %, 95 % de dados entregues em 3 minutos e perda de pacote menor que 2 %.

(PEREIRA et al., 2019) investigam a existência de dispositivos móveis, principalmente ônibus e táxis, como consumidores e geradores de dados. Para tanto, propõem uma arquitetura *Fog* para redes veiculares e avaliam uma prova de conceito de analytics para duas aplicações: detecção de anomalias de tráfego e previsão do tempo de chegada de ônibus nas paradas. Todos as OBUs são proativas, analisam os dados por onde passam e geram notificações para os usuários da região. A solução distribui a carga de trabalho de acordo com o tempo de vida dos dados: RSUs armazenam 15 dias de dados, e a nuvem mantém até 2 meses de dados. A avaliação investigou a precisão dos algoritmos de acordo com os dados coletados na *Fog* e na Nuvem. A latência de processamento, tanto da Nuvem quanto da *Fog*, é extraída de conjuntos de dados reais, bem como de tempos de viagem de veículos e ônibus. Os autores consideram

"anomalia", uma latência acima do quarto quartil de todos os tempos analisados. Os resultados mostraram que o *Fog*, tendo características de localização, pode executar ambas as aplicações com mais precisão, mesmo usando menos dados. No entanto, a nuvem faz previsões melhores em relação à chegada de ônibus em algumas situações, pois é menos sensível a mudanças locais e fatores de curto prazo, como algum tráfego anormal pontual causado por pequenos acidentes. Os autores argumentam que uma solução ótima deve combinar dados de curto, médio e longo prazo. O uso de *Fog* também gera menos notificações de atraso de ônibus, embora os autores tenham encontrado uma relação entre a hora do dia e a latência na entrega das mensagens. A latência do serviço praticamente aumenta com a hora do dia. Provavelmente devido ao aumento do uso de redes ao longo do tempo nas redes de acesso. Isso acaba por incentivar o uso de *Fog*, que mantém latência constante e baixa.

Os trabalhos desta subseção contêm as seguintes contribuições para a proposta FogWise:

- a participação dos veículos na VFC só é possível sob condições específicas que podem depender do tipo de aplicação, da rede e dos próprios veículos, mas esse limite ainda precisa ser encontrado;
- VFC é um das principais tecnologias habilitadoras de um verdadeiro ITS pois os veículos podem fornecer seu poder de processamento para vários tipos de aplicações de tráfego, usuários e segurança;
- o principal objetivo da VFC deve ser reduzir a latência das aplicações em relação à execução à nuvem ou mesmo em relação a execução interna.

3.5 ALOCAÇÃO DE RECURSOS E DISTRIBUIÇÃO DE TAREFAS EM COMPUTAÇÃO *FOG* VEICULAR

Da mesma forma que na seção anterior, selecionamos trabalhos com base em revisão sistemática com um critério principal de inclusão: trabalhos com foco em balanceamento de carga em VFC nos quais os veículos são dispositivos de processamento. Acreditamos que isso é importante uma vez que alguns trabalhos mencionam VFC, mas se referem a ela apenas como uma extensão da computação em nuvem especialmente adaptada para servir ITS e *infotainment* em veículos (YAN et al., 2019). Embora VFC seja inerentemente sem fio e distribuída, a maioria dos trabalhos não leva em consideração o comportamento errático da rede, como erros e colisões; e

a maioria deles requer suporte de infraestruturas centralizadas, especialmente RSU e Nuvem. Parte deles considera a existência de *third parties* que gerenciam a alocação de recursos e atribuição de tarefas de forma centralizada. Assim, apenas um dos trabalhos que selecionamos apresenta uma arquitetura completamente distribuída (KLAIMI; SENOUCI; MESSOUS, 2018).

(NING; HUANG; WANG, 2019) investigam como minimizar a latência em aplicação *Fog* para gerenciamento de tráfego em tempo real, equilibrando a carga entre *Cloud*, *cloudlets*, VCC e VFC. Para isso, utilizam uma série de algoritmos, como algoritmo de marca e limite (*brand-and-bound algorithm*)(NARENDRA; FUKUNAGA, 1977) e Edmonds-Karp(ZADEH, 1972). Os autores estimam o tempo de resposta da aplicação, mas argumentam que isso pode ser impraticável em um cenário tão dinâmico em que os veículos entram e saem da VFC de maneira incontrolável. Mesmo assim, os algoritmos utilizados reduzem o tempo de resposta quando comparados a uma abordagem aleatória para balanceamento de carga e distribuição de tarefas. Apesar de descrever os protocolos de negociação, os autores não consideram apropriadamente o comportamento da rede, deixando espaço para futuras investigações.

(XU et al., 2018) também investigam como reduzir a latência do serviço enquanto investigam o problema de atribuição de tarefas, propondo um algoritmo de correspondência baseado em preço (*price-based matching algorithm*). Os equipamentos do usuário - User Equipment (UE) - constroem uma lista de servidores *Fog* preferidos antes de enviar solicitações. Quando um servidor *Fog* recebe mais de um pedido, ele aumenta seu preço para que possa cair em preferência nas listas de UEs até que o sistema geral encontre correspondência 1 para 1 entre clientes e servidores. Os resultados indicam que a latência do serviço aumenta com o número de UE, pois eles precisam de mais iterações para chegar a uma solução ideal. Por outro lado, essa latência diminui com o aumento do número de iterações sob um número fixo de clientes. Nessas condições, argumentamos que tal sistema parece inviável para aplicativos sensíveis à latência. Além disso, os autores não consideram os impactos do funcionamento da rede nas negociações para construir uma lista de servidores preferenciais.

(ZHOU et al., 2019) demonstram preocupações de que os trabalhos anteriores não investigaram a falta de incentivos e protocolos de atribuição de tarefas no VFC, visto que os veículos são bens privados e podem optar por não fornecer recursos para a rede. Assim, eles aproveitam a Teoria de Contratos e o Algoritmo de Correspondência Estável Baseado em Preço (*Price-based Stable Matching Algorithm*) para reduzir a latência do serviço. Servidores *Fog* recebem contratos contendo requisitos e recompensas para participarem do VFC. Depois de aceitar as condições, eles entram na VFC e os clientes classificam os veículos de acordo com a latência

de transmissão, latência de serviço, tamanho da tarefa e custo correspondente. As avaliações indicam que a latência aumenta com o número de tipos de contratos, o número de UEs e com a velocidade de deslocamento dos servidores *Fog*. Por outro lado, a latência diminui com o aumento da quantidade de iterações e com o afrouxamento das restrições. Especialmente a segunda conclusão indica que tal mecanismo não pode ser direcionado para aplicações de baixa latência na VFC.

Zhu et al. em (ZHU et al., 2018b; ZHU et al., 2018a) discutem como o Folo (*Fog Following Me*) ajuda a reduzir a latência e a perda de qualidade de serviço ao alocar dinamicamente tarefas acionadas por evento. Os autores argumentam que a VFC apresenta uma compensação entre a latência do serviço e a qualidade da resposta. Assim, eles adaptam algoritmos de programação linear e a otimização de enxame de partículas para balancear essa relação. Os autores investigam como clientes, servidores e infraestrutura (*Edge*, RSU) podem negociar para permitir a descoberta de serviço eficiente e seleção de servidor para *streaming* de vídeo e aplicativos de detecção de objetos em tempo real. Os resultados indicam que a programação linear é melhor na redução da latência, enquanto a otimização de partículas por enxame mantém os níveis de qualidade altos.

(Lin et al., 2018) visam reduzir a latência do serviço alocando largura de banda de rede para 4 tipos de serviços: Serviços Elásticos Tradicionais (transmissão de dados, HTTP), Serviços Elásticos Interativos (bate-papo online), Serviços em tempo real pesados (VoIP) e Serviços em tempo real leves (*Live Streaming*). Os autores propõem uma solução em duas etapas para alocar esses recursos. Primeiro, eles obtêm uma alocação sub ótima baseada no Algoritmo de Lagrange, que resulta em sobreposições. Depois disso, um algoritmo de seleção calcula a utilidade de cada subfaixa de largura de banda até encontrar uma solução ótima. Os resultados indicam que a junção das duas etapas aumenta a capacidade da rede de se manter *online* (*survivability*).

Embora não investigando problemas de latência em VFC, (KLAIMI; SENOUCI; MESSOUS, 2018) visam minimizar a perda de pacotes e aumentar a taxa de admissão de tarefas para aplicações de alta prioridade mesmo em meio à dinâmica das redes veiculares. Eles modelam o problema com base na entrada e saída de veículos nos estacionamentos, enquanto uma variável auxiliar ajuda a aumentar ou diminuir a prioridade de acordo com a demanda. Eles propõem uma solução para este problema por meio de um jogo descentralizado para a tomada de decisão sobre a utilização dos recursos dos veículos estacionados. Enquanto isso, uma variável auxiliar ajuda a aumentar ou diminuir a prioridade de acordo com a demanda. Assim, eles aumentam

a função de utilidade e, consequentemente, a taxa de resposta para aplicativos *Fog* de alta prioridade enquanto otimizam a utilização da CPU.

(YAO et al., 2018) também estão preocupados com a aleatoriedade de chegadas e partidas de veículos estacionados. Assim, eles propõem uma arquitetura VFC baseada na execução das aplicações em contêineres que tem um Método de Construção e Método de Acesso baseado nas negociações com uma Autoridade Confiável - TA. O Método de Construção tem quatro partes: inicialização, registro, formação e revogação. Quando um cliente precisa acessar a VFC, ele passa por quatro fases: registro, autenticação, login e acesso. Embora segura, essa estrutura não é adequada para aplicativos sensíveis à latência, pois o tempo de autenticação pode retardar o acesso aos recursos da VFC.

(ZHANG; WANG; WEI, 2019) propõem aumentar a disponibilidade de recursos em VFC por meio de um mecanismo de seleção de locais de estacionamento baseado em leilão. O mecanismo funciona através da distribuição de incentivos para que servidores *Fog* estacionem em locais específicos e, assim, possam prestar seus serviços de computação aos clientes das proximidades. Assim, servidores *Fog* recebem prioridades em estacionamentos, mas também recebem incentivos para emprestar poder de processamento. Os recursos disponíveis aumentam à medida que o valor que os veículos recebem também aumenta. Por outro lado, as negociações exigem mais rodadas à medida que aumenta o número de servidores que exigem estacionamento e isso acaba por dificultar a criação de VFCs para servir aplicações de baixa latência.

(ZHAO et al., 2019) também investigam a falta de recursos disponíveis e a falta de incentivo para participação na VFC, bem como a complexidade para realizar *offloading* de tarefas em meio à possíveis colisões, ou seja, quando vários servidores recebem a mesma requisição de processamento. Os servidores *Fog* usam Aprendizagem por Reforço Profundo - Deep Reinforcement Learning (DRL) - para decidir se respondem às solicitações após receber um contrato de recompensa da RSU. A métrica de desempenho é resultado da normalização de outras métricas como a latência do sistema, consumo de energia, taxa de conclusão das tarefas e recursos que os veículos disponibilizam para a rede. Os resultados indicam um aumento significativo na disponibilidade de recursos conforme o aumento de rodadas de treinamento da rede neural. Também existe relação inversa com outros parâmetros da simulação, como é o caso da quantidade de veículos, e da própria rede neural, como o tamanho do batch de amostra, memória de replay e gamma.

Por fim, (Peng; Ota; Dong, 2020) investigam um mecanismo de leilão duplo baseado em

múltiplos atributos para alocação de recursos em VFC. Os autores usam localização, reputação e poder de computação dos dispositivos *Fog* para definir pares cliente-servidor. Eles utilizaram um simulador de tráfego para calcular as interações entre os veículos e utilizaram ferramentas externas para calcular as possíveis interações de comunicação entre eles, pois o simulador VISSIM[1] não oferece recursos de comunicação. Assim, eles avaliaram o desempenho dos mecanismos de leilão e apresentaram como seu algoritmo supera dois outros chamados McAfee e STGA. No entanto, argumentamos que seus resultados podem parecer incompletos devido à falta de recursos de rede. Por exemplo, a complexidade de tempo de seus mecanismos propostos em execução em 50 compradores/vendedores permanece abaixo de 5 ms. Em nossos experimentos, detectamos uma quantidade considerável de erros de mensagem e erros de bit conforme a densidade aumenta, o que pode afetar os resultados. Portanto, argumentamos que avaliações adicionais podem ser necessárias em diferentes condições de rede.

Desta seção, observamos as seguintes contribuições para a proposta FogWise:

- ainda existe uma demanda por investigar a relação da VFC com as limitações da rede veicular (Peng; Ota; Dong, 2020; XU et al., 2018);
- veículos podem participar ativamente da rede como agentes inteligentes (Peng; Ota; Dong, 2020);
- a maior parte dos trabalhos investigam o impacto do gerenciamento dos recursos (computacionais, rede) na latência dos serviços (NING; HUANG; WANG, 2019; XU et al., 2018; ZHOU et al., 2019; ZHU et al., 2018b)
- VFC deve ser usada para aumento de disponibilidade (KLAIMI; SENOUCI; MESSOUS, 2018; YAO et al., 2018; ZHANG; WANG; WEI, 2019);

[1] https://www.ptvgroup.com/en/solutions/products/ptv-vissim/

Tabela 3 – Características e contribuições dos trabalhos relacionados

Artigo	R1 Computação	R2 Qos	R3 Escalabilidade	R4 Mobilidade	R5 API	R6 Dinâmica	R7 Armazenamento	R8 Autonomia	R9 RPRA	R10 Área	R11 Seleção
(VILALTA et al., 2018)	Veículos Autônomos (Protótipo)	S (Reduzir Latência)	N	Apenas menciona	YANG	N (Único Veículo)	N (Apenas processa comandos)	Veículos Autônomos	S (Comandos e leilão)	N (Único veículo)	N (Único veículo)
(TSAUR; YEH, 2018)	Procedimentos de Segurança	S (Latência reduzida para segurança)	Não avaliado	S (Mobilidade gera perda de pacotes)	Indefinido	S (Processamento distribuído)	Indefinido	Indefinido	S (Notificações após processamento)	S (*Trusted authority*)	N (Indefinido)
(CASTELLANI RISSO; LOTI, 2018)	*Analytics*	S (Redução de latência através de processamento distribuído)	S (Redes *FOG* isoladas)	N (Emulado por SDN)	MQTT	S (Conexões aleatórias)	S (*Bundles* DTN)	S (Conexões aleatórias)	S (*Analytics* preliminar)	S (Redes DTN isoladas)	Indefinido
(NETO et al., 2018)	*Analytics* + Video	N (Apenas para economizar recursos)	Não avaliado	Não avaliado	Indefinido	N (*Workloads* controlados)	N (Reduzir e eliminar armazenamento)	N (Economias aleatórias)	S (Identificação de ameaças de segurança)	S (Dados processados dependendo da região)	N (Indefinido)
(PEREIRA et al., 2019)	*Analytics*	S (Reduzir a latência de notificações)	N (Indefinido)	S (*Traces* reais)	Indefinido	S (Densidade)	S (Vídeo e dados dos sensores)	N	S (*Analytics* e notificações)	S (Fog-nodes fixos)	N (Indefinido)
(Peng; Ota; Dong, 2020)	Tarefas	S (Reduzir latência para clientes e servidores)	S (50 150 veículos)	S (*Traces* reais)	Indefinido	S (Densidade e relações clientes-servidores)	S (Indefinido)	Leilão	S (Indefinido)	S (RSU)	N (Leilão)
FogWise	Tarefas	S (reduzir latência para clientes e servidores)	S (densidade)	Veículos entrando e saindo da área de um *Fog Node*	WAVE	Veículos entrando e saindo da área de um *Fog Node*	Durante a execução da tarefa	Leilão	S (ITS, processamento de imagem)	S (densidade)	S (Leilão)

Fonte: Elaborada pelo autor (2021)

3.6 QUADROS COMPARATIVOS DOS TRABALHOS RELACIONADOS

De acordo com as discussões realizadas, e as contribuições identificadas em cada trabalho, construímos as Tabelas 3 e 4. A tabela 3 contém um sumário dos principais trabalhos relacionados a esta tese e que contribuíram diretamente com a definição dos do tema principal da pesquisa e com os princípios de *design* que serão apresentados na seção 4. Já a tabela 4 contém os principais parâmetros e métricas utilizados nos trabalhos anteriores no contexto de Computação *Fog*.

Tabela 4 – Quadro comparativo e sumarizado dos trabalhos relacionados

Trabalho	Parâmetros	Métricas
Modelagem e Simulação		
(SARKAR; MISRA, 2016)	Quantidade de Nodos	Latência e Consumo
(GUPTA et al., 2017)	Topologias	Latência, Energia e utilização da rede
(HONG et al., 2013)	Alcance das requisições, quantidade de nodos.	Latência, Carga de Trabalho
(TEERAPITTAYANON; MCDANEL; KUNG, 2017)	Quantidade de dispositivos, falha e dados parciais.	Precisão do algoritmo, Latência de Agregação de Dados
Migração de Recursos		
(SATOH, 2013)	Tamanho do Agente (KB)	Custo de duplicação do agente (ms), Custo de implantação (ms)
(SHUJA et al., 2017)	Poder computacional e arquitetura	RTT (ms), Consumo energético (mW)
(AMENDOLA; CORDESCHI; BACCARELLI, 2016)	Tipo de rede	Consumo energético (Joules),
(BROGI et al., 2017)	Linha do tempo	Utilização

Tabela 4 continua na próxima página

Tabela 4 – continuando da página anterior

Trabalho	Parâmetros	Métricas
(DUPONT; GIAFFREDA; CAPRA, 2017)	NA	NA
(SOUA; TOHME, 2018)	NA	NA
	Orquestração em Fog	
(ALAM; TUN; HONG, 2016)	Cluster Heads, Quantidade de Nodos	Latência e Consumo
(OUYANG; ZHOU; CHEN, 2018)	Quantidade de Nodos, Custo de migração, parâmetros de migração.	Latência
(MAO; ZHANG; LETAIEF, 2016)	Requisições, Distância do servidor MEC, requisito da aplicação	Custo de execução, latência, energia
	Participação dos Veículos	
(HOU et al., 2016)	Traces de Veículos	Estimativas sobre: densidades; taxa de entrada, saída e permanência da rede, Disponibilidade de recursos.
(VILALTA et al., 2018)	*Timeline*	Tempo de resposta (us)
(TSAUR; YEH, 2018)	Quantidade de notificações, Densidade de veículos	Latência de verificação (ms), Razão da latência de verificação (ms), *Overhead* de comunicação (B), Latência fim a fim (s), Perda de mensagens (%)
(NETO et al., 2018)	Carga de notificações em relação à carga Nominal	Eficiência de CPU, Economia de Energia,

Tabela 4 continua na próxima página

Tabela 4 – continuando da página anterior

Trabalho	Parâmetros	Métricas
(CASTELLANO; RISSO; LOTI, 2018)	Probabilidade de conexão entre Dispositivos, duração da conexão, *Timeline*	Taxa de entrega de mensagens, Tempo médio de entrega (s), *Bundles* entregues (%), *Bundles* por dispositivos (#)
(PEREIRA et al., 2019)	*Timeline*	Predição de atraso nas paradas, Taxa de erro na precisão, Latência entre *cloud* e RSU
Alocação de Tarefas em VFC		
(NING; HUANG; WANG, 2019)	Taxa de mensagens do sistema, Quantidade de veículos estacionados	Tempo de resposta (s)
(XU et al., 2018)	Quantidade de iterações, Quantidade de UE	Latência de rede
(ZHOU et al., 2019)	Tipo de itens no contrato, tipo de contrato, Quantidade de veículos, Quantidade de UE, Tamanho do dado, Capacidade computacional, Requisito de latência, Velocidade dos veículos.	Utilidade da BS, Utilidade dos veículos, Bem-estar social, latência da rede
(ZHU et al., 2018b)	Estratégias de alocação, Capacidade de memória, tecnologia de rede (LTE, DSRC).	Latência (ms), Transbordo de memória, Vazão de rede (kbps)

Tabela 4 continua na próxima página

Tabela 4 – continuando da página anterior

Trabalho	Parâmetros	Métricas
(Lin et al., 2018)	Banda disponível, Tipos de serviços, Banda já alocada, Categoria	Utilidade, Alocação ótima
(KLAIMI; SENOUCI; MESSOUS, 2018)	Intervalo de tarefas de alta ou baixa prioridade	Utilidade
(YAO et al., 2018)	Taxa de entrada de veículos na rede, Tempo de permanência na rede, taxa de saída da rede, Tempo de execução das tarefas, Tempo de migração das tarefas	Vazão do sistema (tarefas/-min)
(ZHANG; WANG; WEI, 2019)	Preço do *offloading*, quantidade de veículos, vazão da rede, abordagem de alocação, Rodadas de negociação	Preço do *offloading*, quantidade de veículos, vazão da rede,
abordagem de alocação		
(ZHAO et al., 2019)	Quantidade de episódios de treinamento, memória de *replay*, *batch* de amostras, quantidade de veículos, Gamma (parâmetro DRL)	Desempenho normalizado do sistema (DRL)
(Peng; Ota; Dong, 2020)	*Timeline*, Quantidade de compradores e vendedores.	Preço, Pares formados, Utilidade, Complexidade computacional, Tempo de leilão (ms)

Tabela 4 continua na próxima página

Tabela 4 – continuando da página anterior

Trabalho	Parâmetros	Métricas
FOGWISE (esta tese)	Capacidades, cargas, Requisições, aplicações, e vazão da rede	Latência, enfileiramentos, taxa de sucesso de requisições

Fonte: Elaborada pelo autor (2021)

3.7 CONSIDERAÇÕES FINAIS

Os trabalhos discutidos nesta seção apresentam várias nuances necessárias para análise, modelagem, simulação e implementação de uma VFC. Acreditamos ter identificado algumas lacunas nesses trabalhos, e elas serão incrementalmente preenchidas no arcabouço FogWise. Essas lacunas são:

- Falta de análise, modelagem e simulação da utilização de dispositivos da VFC levando em conta as características e limitações reais das redes veiculares;
- Existem trabalhos de *Fog* com análise de mobilidade, mas apenas sugerem apesar de não avaliar as aplicações VFC de baixa latência;
- existem trabalhos que investigam *offloading* e distribuição de tarefas em VFC, mas não fazem isso em cenários de aplicações de baixa latência;
- a definição de VFC reza que ela seja adaptável e distribuída, mas muitas propostas ainda dependem de um ponto de controle centralizado.

4 FOGWISE

Esta seção apresenta FogWise, um *framework* proposto nesta tese e que contém modelo de análise para VFC, arquitetura e algoritmos para habilitar aplicações de baixa latência em VFC do ponto de vista da rede. Trata-se de um trabalho de análise, investigação e caracterização de mecanismos essenciais em VFC que estão sob o leque que chamamos "gerenciamento de recursos": distribuição de aplicações e alocação de tarefas. Para isso, os veículos devem ter a capacidade de responder a requisições e, especialmente, ter os códigos das aplicações necessários para processá-las. FogWise descreve e analisa os elementos necessários para que os veículos recebam os códigos que necessitam para responder às requisições externas, que podem ser originadas de outros veículos, dispositivos móveis de usuários ou de dispositivos IoT. Por fim, propomos um modelo de leilão com participação espontânea dos veículos com objetivo principal de reduzir a latência na execução das aplicações.

4.1 MOTIVAÇÃO, REQUISITOS E PRINCÍPIOS DE DESIGN

A busca pela computação ubíqua não é nova mas voltou à discussão com a emergência de novas tecnologias como SDN, NFV e a evolução das redes sem fio em questões de *handover* e gerenciamento de mobilidade (CHEN; JOSHI; FININ, 2001). Muito se fala sobre a definição de estratégias para migração de VMs e dos desafios de realizar isso em ambientes de grande mobilidade, especialmente devido à sobrecarga gerada durante o processo de negociação, instanciação, transferência de dados e inicialização da aplicação virtualizada, quando comparado ao tempo de execução da aplicação diretamente sobre um ambiente não virtualizado (RODRIGUES et al., 2017; BAKTIR; OZGOVDE; ERSOY, 2017; BOUKERCHE; ROBSON, 2018; HAO et al., 2017; RODRIGUES et al., 2017). Acreditamos, então, que a migração de serviços deve se basear em outras estratégias que não tenham sobrecarga elevada de virtualização, e que possam permitir uma execução que atenda aos requisitos das aplicações previstas e planejadas para VFC. Além disso, é preciso analisar quando e como as requisições geradas pelos dispositivos de usuários, ou mesmo IoT serão atendidas, especialmente quando a etapa da distribuição e orquestração de recursos for ultrapassada.

Mesmo assim existe um impasse importante nas estratégias adotadas para operações de migração. Migrar aplicações específicas para determinadas plataformas envolve um certo nível de recompilação dos códigos, o que pode tomar tempo antes e/ou depois da migração. Nesse caso, os participantes da migração de código devem dispor dos mecanismos de conversão, ou se valer de modelos semelhantes às lojas de aplicativos, repositórios de controle de versão ou mesmo qualquer mecanismo que permita a execução dos mesmos códigos nos mais diversos dispositivos (SHUJA et al., 2017). Conforme o nível de virtualização aumenta, o tempo necessário para a conversão de plataformas diminui, mas o tempo total para que uma aplicação esteja pronta para executar, pode aumentar. É o caso de migrações específicas baseadas em mecanismos como Contêineres (Docker) (BARBOSA; JúNIOR; DIAS, 2020) e até mesmo VMs (AMENDOLA; CORDESCHI; BACCARELLI, 2016; BROGI et al., 2017).

Na outra ponta, o nível de mobilidade e a imprevisibilidade em determinar os recursos disponíveis em VFC podem inviabilizar a distribuição ótima de aplicações devido à complexidade computacional (NING; HUANG; WANG, 2019; YAO et al., 2018; BARIK et al., 2021; FENG et al., 2020). Caso a distribuição não seja otimizada, surge outro problema em relação à alocação de tarefas, ou seja, sobre a decisão de qual servidor *Fog* vai responder às requisições em meio a um *pool* de recursos.

Para atingir o objetivo de ter uma computação realmente ubíqua, é necessário projetar uma *framework* que envolva todos os elementos que possam fornecer computação. Grande parte da literatura atual, como demonstrando na revisão de literatura (Capítulo 3), considera que a abrangência do conceito de *Fog* vai da arquitetura de nuvem centralizada/público-privada, passando por servidores colocados na borda da rede (Edge), até os dispositivos de rede (middleboxes, AP, roteadores, *switches*, etc...), mas não chega até os dispositivos finais, deixando essa lacuna aberta (HONG; VARGHESE, 2019; MOURADIAN et al., 2017). Entretanto, com a emergência da IoT alguns trabalhos já preveem (BROGI et al., 2017; SATOH, 2013) e avaliam o processamento completamente distribuído, ainda mais próximo a origem dos dados (SATOH, 2013; TEERAPITTAYANON; MCDANEL; KUNG, 2017; HOU et al., 2016; TSAUR; YEH, 2018; NETO et al., 2018; CASTELLANO; RISSO; LOTI, 2018) . Ainda assim, falta uma avaliação e uma previsão adequada de como ocorre a distribuição das aplicações, e a resposta à requisições e serviços no ambiente mais dinâmico e móvel que se apresenta para o futuro próximo.

Mesmo que os trabalhos prevejam e analisem uma distribuição completa, eles acabam por avaliar as aplicações individualmente, ou seja, teorizam e avaliam o funcionamento de apenas uma aplicação sem levar em consideração os detalhes e influências de outras aplicações que po-

dem estar executando paralelamente. Cenários com aplicações únicas não são realistas quando se leva em consideração a quantidade de usuários móveis (*smartphones*), veículos autônomos conectados e aplicações IoT previstas para os próximos anos (ATZORI; IERA; MORABITO, 2010; MAI; SCHLESINGER, 2011).

FogWise busca a realização da computação ubíqua integrando os dispositivos finais, neste caso os veículos da VFC, à cadeia de computação e disponibilizando seu poder computacional para toda a rede. Para tanto, em meio a quantidade de avaliações em nível de rede, de computação na borda, distribuição de processamento na borda ou na névoa, além do controle e distribuição de requisições, é preciso avaliar o funcionamento de várias aplicações simultâneas em todos os dispositivos, além da distribuição de código de aplicações entre os dispositivos de maior mobilidade.

FogWise busca atender aos requisitos de aplicações que precisam de baixa latência, além de seguir princípios de design propostos na literatura. Além disso, precisa apresentar mecanismos suficientemente robustos e avaliados para ser capaz de sustentar os padrões para as próximas gerações de telecomunicações e computação, especialmente naqueles ambientes de alta mobilidade e alta dinâmica de dispositivos e/ou aplicações. A literatura, em duas discussões, e especialmente apresentação de definições, apresenta requisitos (SATOH, 2013) e critérios de avaliação em termos arquiteturais e algorítmicos (MOURADIAN et al., 2017) que uma cadeia de computação *Fog* deve ter. Podemos utilizá-los tanto para avaliar a criação das aplicações, quanto para determinar se uma aplicação pode mesmo ser enquadrada como *Fog*. Os princípios de design, que até mesmo ajudam na definição do que é *Fog* e VFC são listados a seguir:

- R1: todos os nodos devem ter capacidade de executar tarefas de gerenciamento ou processamento, mas podem não ter códigos para essas tarefas em sua inicialização. Esses mecanismos devem ser transparentes em dispositivos heterogêneos em termos de domínio, características de processamento, armazenamento e rede (SATOH, 2013; MOURADIAN et al., 2017).

- R2: computação *Fog* deve atender a requisitos de QOS e QOE, especialmente em relação à variação de latência, através de módulos arquiteturais e migração de serviços entre estratos (*cloud-edge-fog*) (MOURADIAN et al., 2017; HONG et al., 2013);

- R3: sistemas *Fog* devem dar suporte à alta escalabilidade e elasticidade pois podem precisar lidar com milhões de dispositivos IoT;

- R4: a cadeia *Fog* deve dar suporte à mobilidade, seja dos geradores de dados (ex.: sensores, *smartphones*) seja dos dispositivos que proveem computação (MOURADIAN et al., 2017). FogWise pretende apresentar soluções robustas para suporte à computação *Fog* em altos níveis de mobilidade;

- R5: *Fog* deve suportar interoperabilidade entre Federações. Acreditados que a utilização de APIs padronizadas ou arcabouços com suporte à virtualização podem ser uma passo importante nessa direção (MOURADIAN et al., 2017; SHUJA et al., 2017);

- R6: sistemas *Fog* rodam em redes distantes (fisicamente e em saltos) do núcleo e geralmente são baseadas em redes sem fio, são dinâmicas, e apresentam baixa vazão de dados e desconexões frequentes;

- R7: os sistemas clientes não precisam, obrigatoriamente, dispor de algum mecanismo de armazenamento, e os dados podem ser mantidos em armazenamento (permanentes ou temporários) dos nodos *Fog*;

- R8: um dispositivo *Fog* deve ter uma lógica autonômica interna que o permita participar do processo de elasticidade (*scaling out* e/ou *scaling in*) dos serviços e aplicações da rede (BROGI et al., 2017);

Por fim, definimos, o escopo desta pesquisa considera que o modelo de programação RPRA deve ficar contido dentro da duração do contato entre o dispositivo que faz a requisição (IoT ou mesmo outro veículo), e o dispositivo que responde, que no caso FogWise, é outro veículo. Além disso, é preciso criar um mecanismo eficiente de atendimento às requisições. Numa dada região, é preciso ter uma quantidade mínima de veículos capazes de responder às demandas. Não é desejável deixar que apenas um veículo possa atendê-las. Por outro lado, uma quantidade muito grande pode gerar sobrecarga para distribuição e conflito de respostas. Ao mesmo tempo que a redundância gera resiliência e disponibilidade, ela também cria o problema sobre a decisão sobre qual ou quais veículos devem atender à determinadas requisições para evitar desperdício de recursos. Então, definimos os seguintes princípios de design adicionais:

- R9: todo o processo de requisição, processamento e resposta deve acontecer dentro da duração do contato entre os dispositivos que estão participando da transação *Fog*. Esse princípio é suportado por investigações realizadas em outros trabalhos (ZHU et al., 2010; XIAO; KUI, 2015), mas ainda não havia sido proposto e avaliado. O tempo de atuação

pode ultrapassar a duração do contato, uma vez que se trata de um processamento da reposta;

- R10: uma região tem limites para seu processamento que são determinados pelo número de servidores, as aplicações que eles podem executar e os limites do espectro eletromagnético;
- R11: o atendimento às requisições depende de um mecanismo, preferenciamento distribuído, para seleção de um veículo/serviço em meio aos outros.

4.2 PRESSUPOSTOS SOBRE O SISTEMA

Boa parte das características atuais e futuras da IoV já foi apresentada. Mesmo assim, é preciso determinar os pressupostos específicos dos quais FogWise depende para funcionar, além das situações nas quais se encaixa com maior ou menor precisão.

Inicialmente, o sistema FogWise necessita de protocolos e algoritmos adaptáveis, e tão simplificados quanto possíveis, para que o tempo de resposta a uma requisição seja otimizado. Acreditamos que, em alguns contextos, isso pode ser alcançado através da utilização de SDN, ou especificamente de Wireless Software-Defined Networks (WSDN), devido ao contexto apresentado. SDN pode dar um suporte em relação à sinalização e negociação antecipada de rotas e de encaminhamentos. O auxílio não está apenas em nível de redes cabeadas no *backhaul*, que é formado pelos componentes dTcr (*Delay* - ΔT - *Cloud-Router*), dTre (*Delay Router-Edge*) e dTrr (*Delay Router-RSU*) da figura 17), e que sempre se apresentou como o foco inicial principal da computação na borda, mas também em nível de rede sem fio (HAQUE; ABUGHAZALEH, 2016). *Frameworks* que se utilizam de SDN, NFV e *Fog* em ambiente IoV estão sendo desenvolvidos e analisados em paralelo a esta tese (BARBOSA; JúNIOR; DIAS, 2020).

Em relação a esses *frameworks*, num primeiro momento é preciso construir alguns pressupostos, mas todos eles serão trabalhados e especificados pela proposta FogWise junto aos princípios de design que serão apresentados e discutidos. Os principais pressupostos são:

- as hierarquias de computação (*Cloud, Edge, Fog*) distribuem as aplicações, ou os códigos das aplicações, utilizando SDN para realizar a distribuição eficiente, correta e completa de aplicações, se valendo possivelmente de mecanismos de *handover*, especialmente nas Road Side Units (RSU);

- as requisições são recebidas por um conjunto de veículos que devem decidir sobre atendê-las ou não de acordo com algum critério;
- veículos priorizam as aplicações executadas por seus ocupantes e mantém pelo menos uma *thread* voltada para a execução de aplicações de terceiros na VFCs;
- veículos têm acesso contínuo a bancos de dados contendo estimativas de tempo de execução (*profiling*) para diversos tipos de aplicações de acordo com o poder de processamento e o tipo de hardware disponível, seja através de requisições à nuvem, seja mantendo uma cópia interna (AMENDOLA; CORDESCHI; BACCARELLI, 2016);
- veículo possuem capacidade de armazenamento maior que os *smartphones* atuais (centenas de Gigabytes), portanto, o problema da alocação de recursos pode ser analisado com folga, uma vez que existe espaço disponível para armazenar centenas de aplicações diferentes;
- existe um mecanismo de autenticação mútua, de baixa latência, que utiliza chaves públicas ou *blockchain* para que os veículos criem comunicações seguras (KAUR et al., 2019; YAO et al., 2018; TSAUR; YEH, 2018).

FogWise depende de algumas características dos veículos e das aplicações. Pressupomos, também, que todos os veículos, ou grande parte deles, são autônomos. Isso faz com que eles precisem ter algumas características como:

- capacidade de carregar poder de processamento maior que os dispositivos móveis tradicionais (*smartphones*), lidando com centrais multimídia, agregadores de dados de sensores e, até mesmo, pequenos servidores (SOUA; TOHME, 2018);
- capacidade de coleta de energia (*energy harvesting*) seja com a queima de combustíveis fósseis até veículos completamente elétricos com sistemas de coleta de energia fotovoltaica e dos freios (MAO; ZHANG; LETAIEF, 2016)

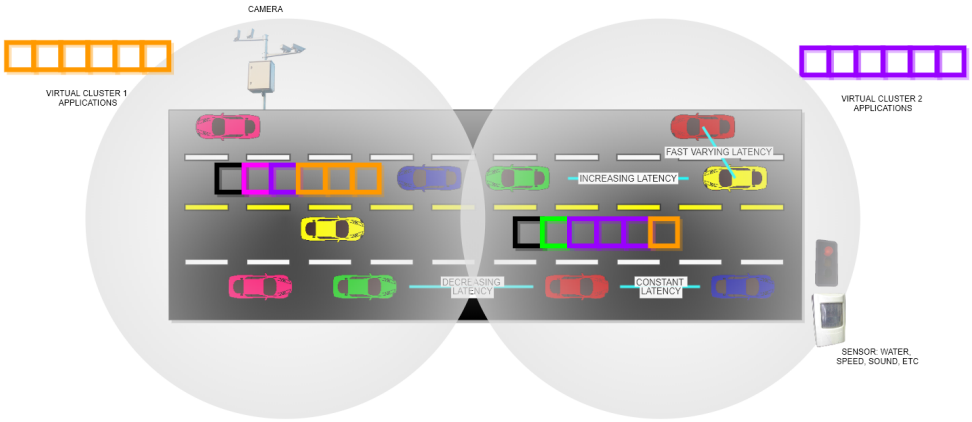
Se valendo dessas características, os veículos participarão de algumas situações, ou participarão de tarefas/aplicações como:

- servir e analisar dados uns dos outros, seja para verificar/validar resultados, seja para dividir carga de processamento;

- servir aos usuários móveis no interior dos veículos, especialmente recebendo *offloading*;
- servir aos dispositivos IoT nas imediações ou servir de mula de dados para processamento/agregação.

A Figura 16 apresenta algumas características necessárias para que os veículos participem da interação *cloud-fog*. Para fornecer processamento uns aos outros, bem como aos dispositivos IoT que os cercam, eles precisam fornecer API para receber requisições e enviar respostas. Eles devem conter as aplicações necessários para realizar os processamentos. Na proposta FogWise, abstraímos as APIs como distribuições de probabilidade que indicam as aplicações que os veículos possuem, as aplicações que eles terão de acordo com a área em que se encontram, e as requisições que eles estão recebendo.

Figura 16 – FogWise. Relações entre veículos e aplicações.



Fonte: Elaborada pelo autor (2021)

Os *clusters* (Virtual Cluster 1 e 2) representam conjuntos de aplicações necessárias em um determinado local. Eles são formados a partir dos requisitos de usuários de dispositivos móveis, de outros veículos ou mesmo de aplicações IoT. Um estádio de futebol, por exemplo, pode conter aplicações de redes sociais, *streaming* de vídeo ou processamento de dados de sensores como câmeras, incêndio e outros em seus *clusters*. Cruzamentos nas vias podem conter aplicações relacionadas com direção autonômica ou de processamento dos dados de semáforos ou sensores de trânsito em geral. Alamedas de negócios, reservas ambientais ou locais sujeitos a desastres ambientais como vulcões, terremotos ou furacões podem ter aplicações específicas que os veículos podem receber e priorizar nas situações de risco.

O *framework* FogWise se encaixa em um cenário em que os veículos estão aptos a receber requisições conforme os veículos passam por esses *clusters* e baixam as aplicações. A figura

16 indica que os veículos de cor rosa e verde possuem:

- aplicações padrão relacionadas, por exemplo, a direção autonômica, mapas, processamento de sensores e sinais do ambiente, etc, na cor preta;
- aplicações comuns entre os usuários mais frequentes do veículo como multimídia, alguma rede social, mensagens ou jogos, na cor do veículo (rosa ou verde);
- aplicações específicas de IoT ou de outros usuários do clusters no qual os veículos passam ou podem vir a passar em algum momento, na cor do cluster (laranja ou roxo).

Previsões da academia e da indústria, estimam que haverá 1 dispositivo IoT por metro quadrado, mesmos assim é difícil quantificar quantos deles participarão ou dependerão da cadeia de processamento *Fog*, ou mesmo quais serão as taxas de dados/requisições gerados (CHEN; JOSHI; FININ, 2001; IRAJI; MOGENSEN; RATASUK, 2017). Nesse sentido, de acordo com a figura 16, ainda é difícil prever todas as situações nas quais os veículos autônomos estarão envolvidos, ou quais ou quantas serão as aplicações que os veículos deverão executar ou processar de forma compartilhada. A literatura propõe várias aplicações, como aquelas listadas na seção 2, mas a carga e as necessidades de cada uma demandam caracterizações muito específicas, especialmente no que diz respeito à utilização de cada uma em termos de área.

Dessa forma, a proposta FogWise não se destina a criar ou propor mais uma aplicação para IoV como os outros trabalhos, especialmente os trabalhos relacionados fazem. Mais do que isso, um dos principais objetivos é propor um modelo que atenda a diversas demandas de aplicações IoV, e especialmente IoT, que são executadas de forma distribuída.

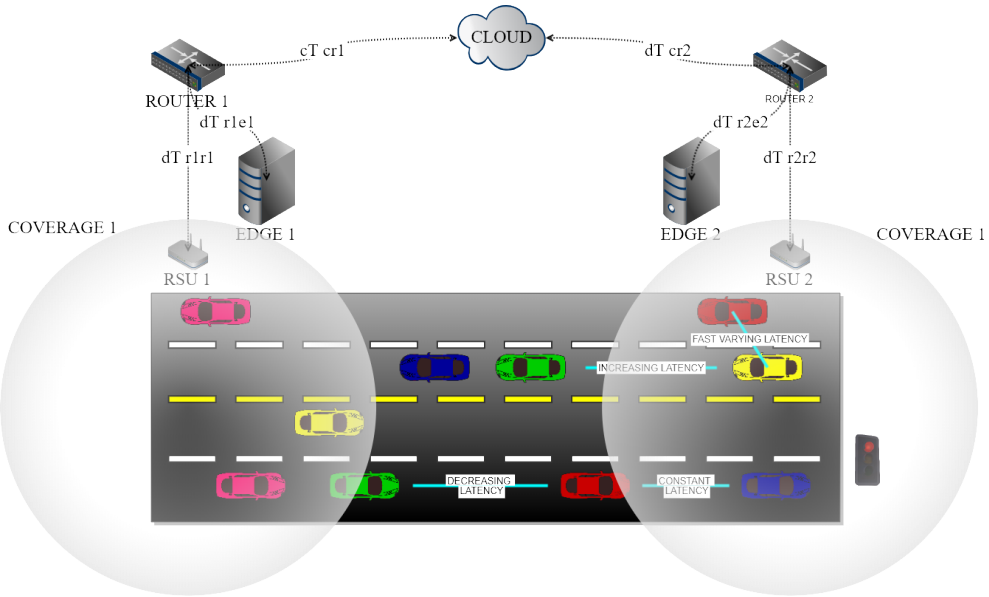
O modelo deve ser adaptável de tal forma que atenda a diversas nuances da IoV, dependendo da evolução das tecnologias. Por um lado, deve ser possível utilizá-lo para determinar o comportamento dos veículos em termos das aplicações que executarão. Nesse caso, os parâmetros de maior interesse estão relacionados com as aplicações, ou seja, os tipos de aplicações existentes, a quantidade de dispositivos envolvidos e a taxa de requisições-respostas necessárias.

Por outro lado, o modelo pode ser usado para determinar como posicionar elementos de rede (RSU, *edge*), as aplicações ou determinar seu funcionamento de acordo com os recursos disponíveis nos veículos. Nesse caso, os parâmetros de interesse são as características de processamento dos veículos, as aplicações que eles podem rodar, a quantidade e os tipos de aplicações que eles podem executar.

4.3 ARQUITETURA DE REFERÊNCIA PARA O FRAMEWORK FOGWISE

A Figura 17 demonstra como todos os elementos podem cooperar para fazer parte de uma VFC. A arquitetura é formada pela infraestrutura de nuvem centralizada (*Cloud*), servidores estrategicamente posicionados próximos a borda da rede dando suporte a uma infraestrutura de nuvem (*EDGE* 1, *EDGE2*), e dispositivos *middleboxes* na borda da rede (*ROUTER*, RSU), geralmente referenciados na literatura como dispositivos *Fog*, além dos Veículos.

Figura 17 – Principais elementos da arquitetura que dá suporte a FogWise



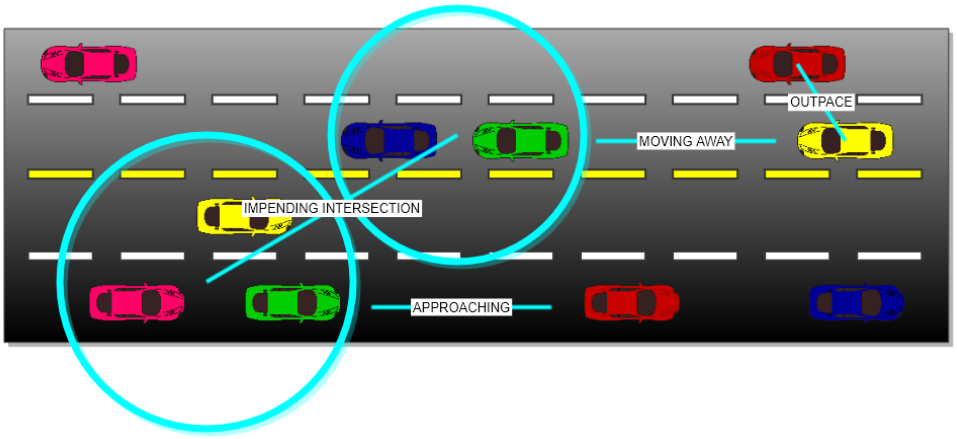
Fonte: Elaborada pelo autor (2021)

A Figura 17 indica que o tempo de resposta de uma requisição que vai até a arquitetura de nuvem centralizada tem pelo menos 3 componentes de latência: veículo-RSU, RSU-router, roteador-*Cloud*, destacando, claro que é possível encontrar vários roteadores ao longo do caminho (*path*). Caso o processamento seja realizado nas arquiteturas de nuvem próximas à borda da rede (*Edge*), também podemos observar as situações com vários componentes. O primeiro, com três componentes, é o mesmo na arquitetura de Nuvem. O segundo, com dois, é alcançado quando os servidores estão conectados diretamente à infraestrutura, seja RSU, seja LTE. As requisições podem, ainda, ser atendidas nos próprios equipamentos de rede (*middleboxes* como a RSU ou mesmo um roteador. Nesse caso, observamos, na melhor das hipóteses, apenas um componente de latência.

As figuras 17 e 18 contêm também referências a um componente de disponibilidade ainda pouco investigado nas avaliações de Computação *Fog*, especialmente no contexto de migração e disponibilidade de serviços/código, que é o tempo de contato. Alguns autores (ZHU et al.,

2010; XIAO; KUI, 2015) confirmaram que o tempo de contato entre veículos, assim como o tempo de contato entre veículos e interseções, nas quais podemos considerar a existência de sensores IoT variados, é exponencial. Essa característica é útil pois aproxima as características das redes veiculares dos servidores web tradicionais e serve de parâmetro para os experimentos iniciais da proposta FogWise.

Figura 18 – Características relativas da mobilidade dos veículos



Fonte: Elaborada pelo autor (2021)

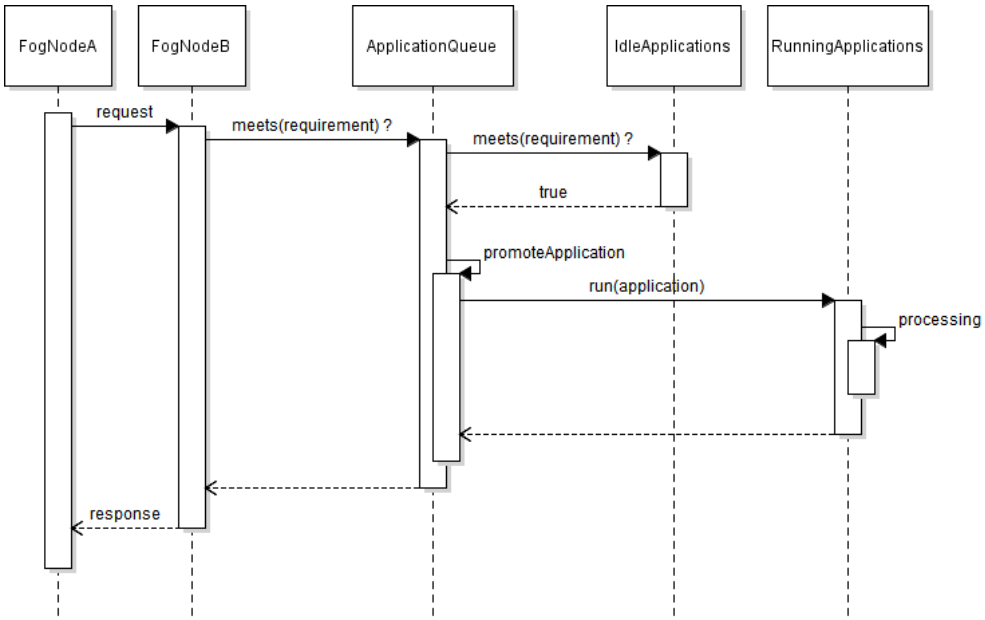
A figura 18 demonstram que os veículos podem estar se afastando, se aproximando ou mesmo mantendo uma distância variável. O modelo inicial, que ainda será refinado, utiliza a direção, a velocidade e a vazão da rede para que os veículos possam prever a quantidade de dados que poderão trocar com os vizinhos ou mesmo com RSU para baixar as aplicações ou parte delas. Depois de baixadas, as aplicações servirão para atender aos outros veículos, aos usuários e até mesmo à IoT com latência próxima de zero.

Para este tipo de arquitetura, e para o escopo deste trabalho, pressupomos um modelo de programação *Fog* que se apresenta propício para este e outros ambientes relacionados com IoT e IoV: o RPRA (HONG et al., 2013), mostrado nos diagramas das Figuras 19 e 21. A figura 19 demonstra que o funcionamento padrão de uma aplicação *Fog* deve começar com uma requisição entre dois dispositivos de forma ubíqua, ou seja, não importa quem faz a requisição e quem processa. Nesse caso, o FogNodeA representa um veículo qualquer ou um dispositivo IoT, ou mesmo um equipamento de usuário, e o FogNodeB representa outro dispositivo que contém uma API e a capacidade de processar aquela requisição. Como se trata de VANET, o modelo mais comum é que uma requisição seja recebida por vários veículos que estejam no alcance de transmissão do sinal sem fio. Quando recebe uma requisição, o veículo realiza uma consulta à sua base de aplicações sobre a possibilidade de atender àquela requisição. Caso sua

base contenha a aplicação correspondente, o veículo processa o dado e responde a quem o requisitou ou encaminha para atuação.

O fluxo positivo também está representado na figura 20, que destaca a participação dos elementos envolvidos no processo. As aplicações notificam ou são detectadas pela rede que dispara o download das aplicações necessárias para a execução. Durante o trajeto, o veículo recebe a aplicação de acordo com a região e as aplicações existentes e, assim, pode prestar seus serviços.

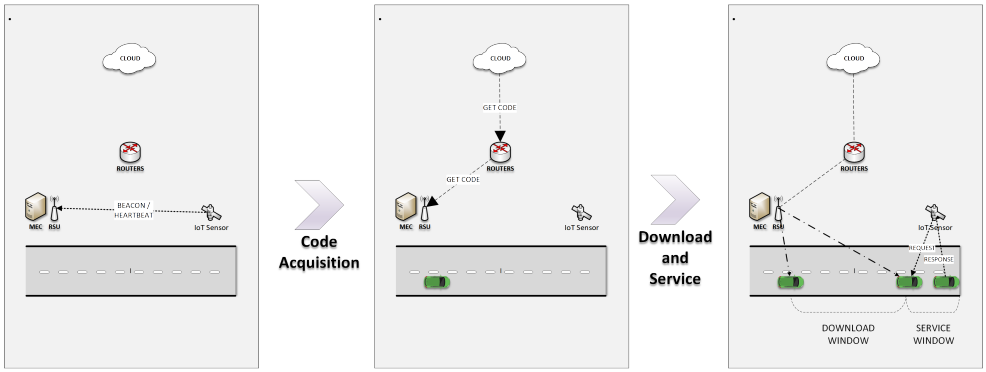
Figura 19 – FogWise: Fluxo positivo



Fonte: Elaborada pelo autor (2021)

A figura 21 demonstra uma proposta para o fluxo negativo, no qual uma requisição não é atendida e o FogNodeB, responsável por ela, deve notificar a Nuvem sobre o evento. É importante destacar que, uma vez que o veículo não possa atender uma requisição, ele pode

Figura 20 – FogWise: Fluxo positivo em funcionamento

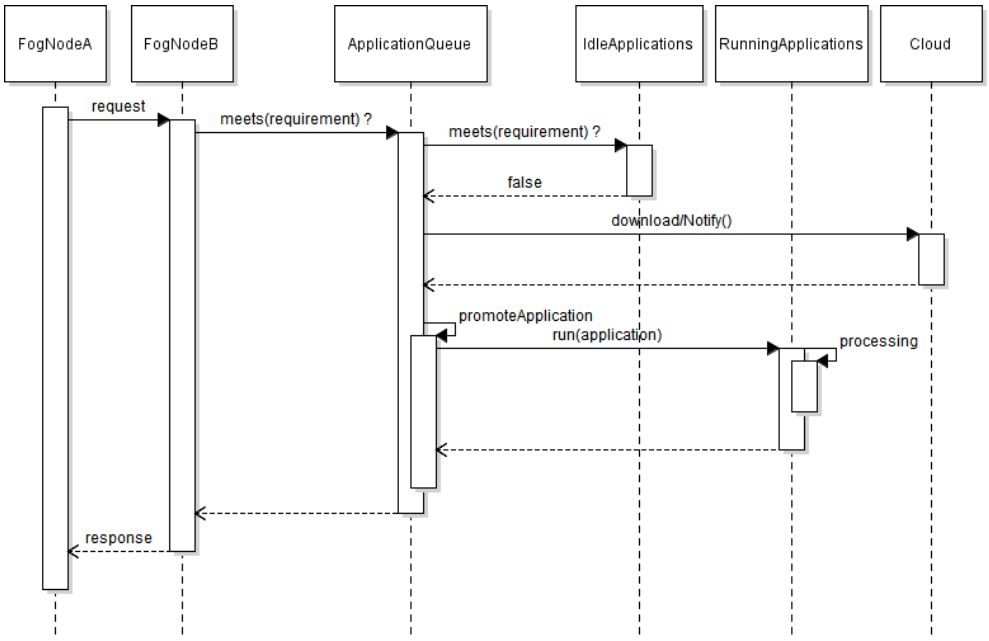


Fonte: Elaborada pelo autor (2021)

optar por não realizar o *download* dos códigos para aquela situação específica, podendo recebê-la para utilização futura. Isso acontece para manter os níveis de QOS ou mesmo para que outro veículo na área pode responder. Esse caso precisa ser analisado e avaliado com mais detalhes, mas até o momento espera-se que o veículo contenha as aplicações para as quais deve disponibilizar uma resposta. O fluxo negativo, ao gerar uma notificação, pode ter algumas consequências para a rede como:

- alimentar uma base de dados sobre aplicações que são ou não são atendidas por determinados veículos em determinados locais, e isso pode ajudar tanto na decisão futura sobre a distribuição de código para determinados dispositivos, quanto na decisão futura dos agentes sobre responder ou não a uma requisição;
- agilizar a decisão de resposta por outros veículos na mesma rede;
- favorecer a decisão de um dispositivo de não fazer *offload* de sua computação.

Figura 21 – FogWise: Fluxo negativo



Fonte: Elaborada pelo autor (2021)

Acreditamos, contudo, que a latência de serviço é um requisito ainda mais importante que vazão (AMENDOLA; CORDESCHI; BACCARELLI, 2016) dependendo da aplicação, especialmente naquelas que envolvem participação ou necessidades de seres humanos (YI; LI; LI, 2015) (ex.: condução autônoma, prevenção de acidentes, internet tátil, jogos e multimídia em tempo real) e, assim, esse componente pode ser ainda mais reduzido caso os dispositivos finais também

participem. FogWise pressupõe que os dispositivos finais, no caso, os veículos, participem do processamento. Para isso, deve haver um mecanismo para a distribuição de aplicações, de forma que os veículos possam executar aplicações uns dos outros, além de servir de infraestrutura para execução de aplicações de dispositivos móveis (*smartphones*, *tablets*, *laptops*) e de dispositivos IoT no geral (sensores e atuadores).

4.4 METODOLOGIA FOGWISE

Feitas essas considerações sobre aplicações e latência de serviço, acreditamos que é preciso criar metodologias incrementais até chegar à realização de uma VFC totalmente autônoma e transparente, de acordo com as definições de IoV e *Fog*. A modelagem inicial carrega elementos de teoria das filas para a compreensão do mecanismo RPRA e identificação de possíveis padrões que possam alimentar algoritmos de ML e otimização no auxílio da resolução dos problemas relacionados com o gerenciamento de recursos que serão apresentados a seguir.

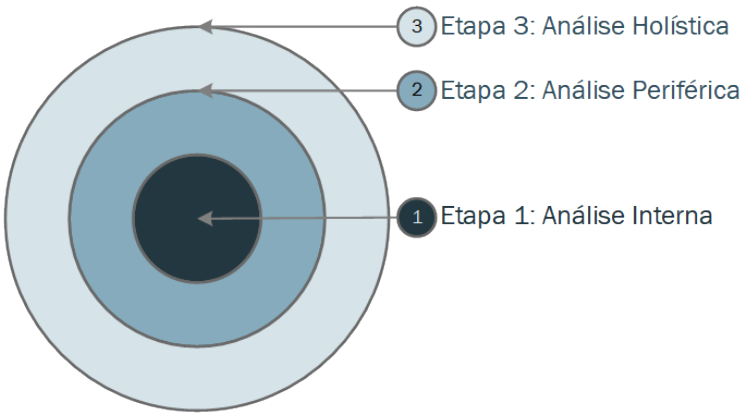
Assim, a metodologia segue a seguinte ordem. Analisamos, inicialmente, o dispositivo terminal, no caso um veículo, através de modelagem da relação entre aplicações, dados processados e requisições. Com isso é possível coletar informações importantes sobre o funcionamento interno que podem ser utilizadas para futuras implementações e tomadas de decisão sobre a própria pesquisa. Em seguida, passamos à análise do ambiente externo em pequena escala, ou o periférico, envolvendo as alguns processos de decisão imediata como a escolha do veículo que responderá a determinadas requisições, ou escolha do conjunto de aplicações a ser baixada dentro do conjunto total disponível. Por fim, deve ser realizada a análise holística, que abrangerá os processos de decisão envolvendo grandes conjuntos de aplicações IoT e IoV, além de veículos e a interação *Fog-Edge-Cloud*. De acordo com a Figure 22, este trabalho contém elementos da Análise Interna e Periférica. Além disso, deixamos elementos suficientes para que a análise holística seja alcançada.

4.4.1 Análise Interna: A estrutura do Nó *Fog*

A figura 23 contém as principais representações desse modelo. Existem três filas principais que se adequam ao modelo de programação RPRA:

- uma fila de requisições que chegam ao veículo, na cor preta;

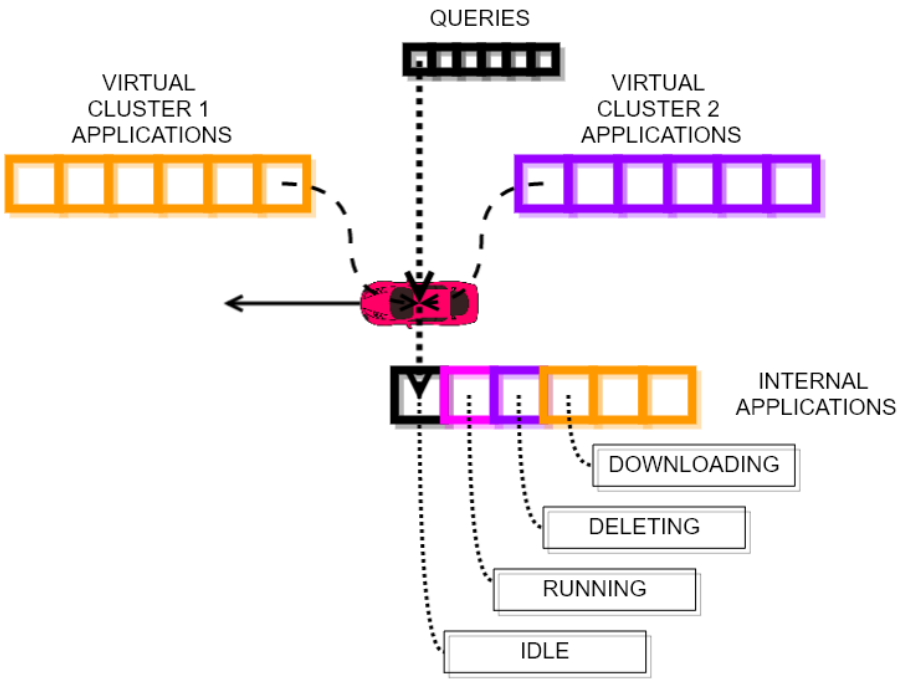
Figura 22 – Método FogWise



Fonte: Elaborada pelo autor (2021)

- uma fila de aplicações que o veículo contém no momento, na cor rosa;
- uma fila de aplicações que o veículo pode processar, está recebendo ou descartando de acordo com o *cluster*, ou *clusters*, em que está conectado, nas cores rosa, laranja e roxo.

Figura 23 – Visão interna de um veículo participando de uma FogWise



Fonte: Elaborada pelo autor (2021)

As análises das interações entre essas três filas foi usada como base para um modelo de simulação independente e, posteriormente, foram incluídas no simulador Omnet++ (VARGA, 1999), que provê um arcabouço completo para a avaliação em diversos cenários de rede e

mobilidade, especialmente quando integrado com o simulador SUMO(Lopez et al., 2018) e o projeto VEINS (Sommer; German; Dressler, 2011).

4.4.1.1 Modelo Inicial de Simulação

O modelo inicial de simulação não continha um suporte específico em nível de rede. Mesmo assim, é possível emular cenários onde há maior ou menor variação na quantidade de aplicações ao redor do veículo, a taxa de requisição de dados, além de prever como este se comportará de acordo com a vazão disponível na rede, seus recursos internos (*capacity*) e as características das aplicações (*load*). Assim, realizamos descrições e avaliações iniciais para validação do modelo de simulação independente.

O modelo trata esses elementos, que serão os parâmetros de parte das simulações, como distribuições de probabilidade, que podem, futuramente, ser alimentadas de acordo com dados coletados na literatura. Utilizando-os como parâmetros, é possível modelar o comportamento do veículo até mesmo de parâmetros não previstos, como a velocidade e a quantidade de veículos ou a quantidade de aplicações IoT ao redor, tudo isso em termos das aplicações que eles estão executando. Existem distribuições para representar:

- a carga das aplicações;
- as capacidades dos veículos;
- o tipo de aplicações que o veículo carrega internamente;
- o tipo de aplicações existentes no ambiente.

Destacamos que o tipo da aplicação pode determinar outras características como o tamanho do seu código em *bytes* e os requisitos de QoS. Além disso, outros elementos específicos das redes veiculares foram inseridos no modelo inicial e foram subsequentemente substituídos por um *framework* de simulação:

- vazão da rede;
- latência de propagação;
- previsão da velocidade dos veículos e, consequentemente, previsões sobre distância percorrida e interações entre veículos, sensores e usuários, de acordo com a Figura 18;

- a capacidade de armazenamento dos veículos.

A proposta de criação de um modelo com características específicas para IoV com *Fog*, além das interações entre todos esses elementos/parâmetros constituem algumas das principais contribuições desta proposta até aqui, especialmente quando comparados com outros trabalhos (SARKAR; MISRA, 2016; HONG et al., 2013; GUPTA et al., 2017). As primeiras simulações se baseiam no Fluxo Positivo, demonstrados nas Figuras 19 e 20, e através delas, destacaremos as contribuições. A conjunção entre a carga das aplicações e a capacidade dos veículos resulta no tempo de execução da aplicação num ambiente VFC, mas isso é comumente feito nos outros simuladores apresentados na literatura. FogWise, por sua vez, adiciona o tempo de migração do código da aplicação ao tempo total do serviço. Ao final, também é possível verificar, através de métricas, a latência da migração do código, a latência total do serviço, além da quantidade de requisições atendidas e não atendidas a depender se o veículo já possuía ou não os códigos necessários para atendê-las. Essas métricas resultam da interação com os outros parâmetros como: a velocidade do veículo, a vazão da rede e a latência de propagação no meio sem fio.

Outras interações podem ser analisadas a partir daí, mas sempre partindo do ponto de vista interno. Por exemplo, a velocidade do veículo, a vazão da rede e o tipo de aplicação, especialmente do que diz respeito ao tamanho do *download* necessário, podem determinar se um veículo conseguirá ter acesso ao código de uma aplicação e atender a uma requisição, dentro do intervalo de interação com uma RSU ou com a aplicação IoT.

4.4.2 Análise Periférica

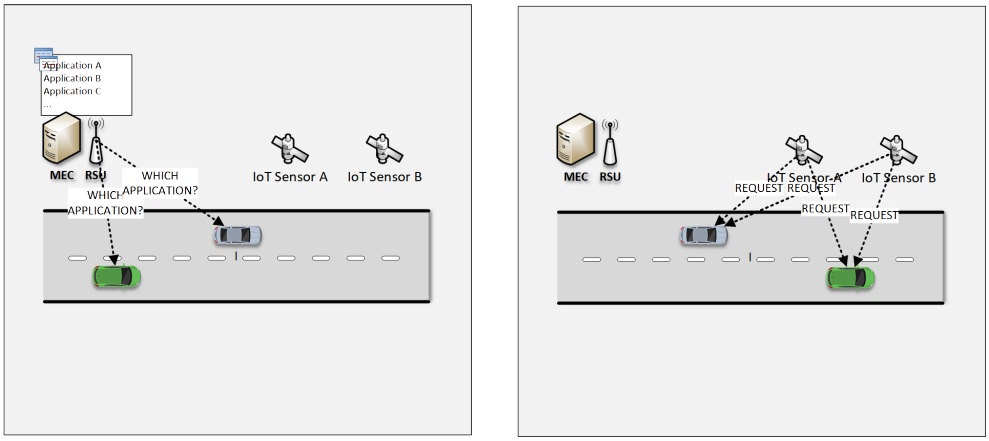
O objetivo da análise periférica é modelar um veículo como um Agente Inteligente (RUSSELL; NORVIG, 2016; SATOH, 2013; ALAM; TUN; HONG, 2016) capaz de basear suas decisões de tratamento de requisições, processamento, *offloading* e enfileiramento apenas no seu estado interno e nas informações que capta do ambiente. Como mencionado anteriormente, o estado interno é basicamente formado pelas filas de aplicações e requisições. O ambiente é apreendido a partir das requisições recebidas e das mensagens trocadas.

Nessa etapa surge a necessidade de analisar minunciosamente duas decisões importantes, representadas na Figura 24: a decisão de quais códigos o veículo deve baixar, dado que existam muitas aplicações que precisam de atendimento nas regiões onde o veículo poderá passar de acordo com sua rota; e a decisão de resposta na presença de mais de um veículo capaz

de atender a uma requisição. A Figura 24 mostra dois veículos recebendo a aplicação e se aproximando de sensores IoT que precisam de respostas rápidas para realizar alguma atuação. Quando as duas requisições são disparadas, FogWise deve ser acionado nos dois veículos para que eles tomem a decisão sobre qual requisição cada um deve atender para otimizar a latência total do serviço para as duas aplicações.

Na solução ótima, tanto nesse cenário quanto em cenários mais complexos, cada veículo responde a uma requisição diferente como resultado de uma decisão interna e autônoma. Pretendemos reduzir ao máximo, ou mesmo eliminar, quaisquer negociações que impeçam as aplicações de se manterem no nível de QOS requerido pelas aplicações previstas para IoV, especialmente as aplicações de segurança. O modelo criado para representar o funcionamento interno do veículo tem flexibilidade para simular diversas situações de mobilidade, especialmente no que diz respeito à interação entre aplicações executando em paralelo no ambiente, quer sejam elas IoT, dos veículos ou dos usuários, de forma que seja possível determinar as situações em que as aplicações conseguirão se manter dentro de dos requisitos esperados e também quando não conseguirão.

Figura 24 – Problemas de decisão FogWise



APPLICATION PROBLEM RESPONSE-DECISION PROBLEM

Fonte: Elaborada pelo autor (2021)

O Problema da Aplicação é caracterizado pela distribuição das aplicações. O modelo de análise interna já prevê que as aplicações contém um tamanho de download para o código. Baseado em seu trajeto, armazenamento interno e nas características das aplicações que pode/pretende atender, é preciso decidir quais códigos o veículo deve baixar. Isso pode ser decidido pelo próprio veículo ou por algum algoritmo rodando na nuvem. Essa decisão de implementação será tomada dependendo dos resultados obtidos com a evolução do modelo de

agentes inteligentes.

Nesse cenário, o controle fino pode ser feito às custas de uma alta quantidade de downloads/migrações por área, além da necessidade de um controle fino sobre as características do tráfego dos veículos. Essa forma pode garantir que sempre haverá um veículo com capacidade de atender á requisição. Outra possibilidade é que exista um algoritmo que decida com muita precisão qual dos veículos tem maior probabilidade de atender a uma aplicação IoT. Esse algoritmo pode rodar na nuvem ou pode ficar a cargo do veículo decidir sobre qual código baixar em meio a um conjunto disponível. Entretanto, com apenas um veículo, não é possível garantir resiliência e disponibilidade, e a requisição fica sujeita a erros no próprio download da aplicação, além da dependência do tratamento posterior dos dados.

O segundo problema, a decisão sobre responder a uma requisição, deriva da possível solução do primeiro. Uma vez que tudo indica que é necessário que mais de um veículo possua os códigos para responder às requisições em uma área, é preciso criar mecanismos para determinar qual deles atenderá. A teoria básica de redes sugere que o problema seja resolvido com sinalização e negociação com algoritmos de balanceamento de carga (KUROSE, 2005). Dadas as características dinâmicas da IoV, além dos requisitos esperados para computação *Fog*, acreditamos que qualquer negociação mais elaborada, e que precise ser realizada no momento da requisição, pode ser impraticável nesses ambientes.

Conforme nos aprofundamos na análise do funcionamento da VFC, observamos que, apesar de as tecnologias de rede, nuvem e veículos não terem evoluído conforme as previsões anteriores, o poder computacional e de armazenamento dos dispositivos móveis têm aumentado a um ritmo acelerado (CERWALL et al., 2020). Assim, podemos considerar que os veículos podem armazenar uma grande quantidade de aplicações enquanto se desloca e pode realizar downloads rápidos em poucos segundos de aplicações. Baseado nesse pressuposto, direcionamos esforços para o problema da decisão sobre a resposta, elaborado nesta seção de análise periférica.

Consideramos que a redundância de respostas é a solução mais simples, ou seja, todos os veículos que receberem a requisição, podem respondê-la. Nesse cenário, o veículo recebe de antemão todos os códigos das aplicações que geram requisições nos locais por onde passará de acordo com seu trajeto pre-estabelecido através do uso de mapas online, por exemplo. No modelo simulação inicial, existem apenas a latência referente à duração do *download* dos códigos das aplicações. O veículo, então, simplesmente responde à requisição, sem verificar redundância.

Tal mecanismo contém algumas falhas claras em relação aos requisitos de redes e *Fog*.

A princípio, não se pode esperar que a rede sempre tenha recursos abundantes para atender às aplicações. Além disso, enquanto vários veículos estão respondendo à mesma requisição, outras aplicações podem ficar sem serviço. Dessa forma, a decisão sobre o serviço é crucial para o funcionamento da cadeia de computação *Fog*, e à busca pelos requisitos estipulados.

4.4.2.1 Projeto Inicial de Veículo como Agente Inteligente

O projeto do veículo como agente inteligente é baseado nas perguntas propostas por (MACAL; NORTH, 2005). As respostas foram dadas de acordo com as características apresentadas e os resultados esperados para a metodologia FogWise.

1. Qual é o problema específico deve ser resolvido pelo modelo? Quais perguntas específicas o modelo deve responder? Que valor agregado à modelagem baseada em agentes inteligentes traria para o problema que outras abordagens de modelagem não podem trazer?

As principais questões a serem respondidas pelo modelo baseado em agente são: como dar respostas a 100% das requisições das aplicações ao redor do veículo (IoT, outros veículos e usuários), minimizando ao máximo a redundância? Como modelar os participantes da atividade como agentes inteligentes de forma a minimizar a necessidade de negociação para determinar qual veículo atende a qual requisição?

O valor específico obtido com a execução do sistema de agentes está relacionado com a utilização de agentes inteligentes e independentes de forma a reduzir da negociação necessária para a tomada de decisão. Os agentes terão acesso às informações do ambiente através de sensores, e *broadcasts* na rede. Com isso, eles construirão sua visão de mundo e tomarão as decisões sobre responder ou não às requisições que receberem. As principais métricas resultantes serão a redundância de respostas e a quantidade de requisições atendidas.

2. Quais devem ser os agentes no modelo? Quem são os tomadores de decisão no sistema? Quais são as entidades que possuem comportamentos? Quais dados sobre agentes são simplesmente descritivos (atributos estáticos)? Quais atributos do agente seriam calculados endogenamente pelo modelo e atualizados nos agentes (atributos dinâmicos)?

Os agentes, inicialmente, são veículos. Mais precisamente, são as OBUs dos veículos que precisam rodar aplicações e decidir sobre responder ou não a uma requisição, dado um conjunto de informações contextuais. Eles coletam informações do ambiente, mas também proveem

informações para a rede, na forma de estatísticas de seu comportamento. Isso tudo deve ser implementado de forma tão descentralizada quanto possível, sempre buscando avaliar o limite de uma VFC totalmente distribuída.

3. Qual é o ambiente dos agentes? Como os agentes interagem com o ambiente? A mobilidade de um agente através do espaço é uma consideração importante?

O ambiente do agente é formado por um conjunto de informações que o veículo-agente pode sensoriar, e outras que ele recebe da nuvem, como o banco de dados de *profiling*. Essas informações podem ser divididas em dois eixos. O primeiro, trata das informações que o veículo sensoria através de sensores ou mesmo de suas interfaces aéreas; o segundo trata das informações que o veículo recebe da rede. O ambiente, então, é formado por:

- características percebidas pelo do veículo sobre si mesmo: posição, velocidade, destino;
- características dos vizinhos: quantidade de vizinhos, características do contato;
- aplicações que ele possui;
- aplicações disponíveis no ambiente imediato, e aquelas que deverá ter no futuro;
- requisições que ele recebeu, recebe e, possivelmente, receberá;
- informações de *profiling* de todas essas aplicações (carga, tipo, requisitos).

4. Quais comportamentos do agente são de interesse? Que decisões os agentes tomam? Quais comportamentos estão sendo usados? Quais ações estão sendo tomadas pelos agentes?

Alguns comportamentos dos agentes são importantes. A mobilidade é o primeiro, porque a dinâmica pode influenciar os resultados. O segundo é a própria decisão sobre responder ou não a uma requisição, que impactará fortemente no funcionamento geral da rede. Além disso, ele pode ter que decidir sobre atender a mais de uma requisição, dependendo, é claro, da relação capacidade/carga. Para decidir sobre atender uma requisição, o agente precisa calcular o resultado de uma função que facilitará a escolha, ou participar de algum tipo de negociação. Essa função está definida, inicialmente, apenas como uma relação capacidade/carga, mas que será incrementada com as outras características obtidas através do modelo inicial FogWise. No fim, a ação do agente é responder a uma requisição em meio a um conjunto de requisições que podem, por sua vez, ser atendidas ou não.

5. Como os agentes interagem uns com os outros? E com o ambiente? Quão expansivas ou focadas são as interações do agente?

A princípio, os agentes não interagem diretamente uns com os outros. Mesmo assim, eles devem cooperar para atingir o objetivo comum, que é o atendimento a todas as requisições e a eliminação da redundância. A interação entre agentes-veículo é direta, pois todos compartilham o meio sem fio e devem estar cientes de que existem outros veículos recebendo os mesmos códigos e tentando responder às mesmas requisições enquanto negociam através de um leilão visando bem comum, isto é, o atendimento ás requisições com menor latência possível.

A interação com o ambiente se dá através do atendimento às requisições. Eventualmente veículos enviam seus resultados estatísticos para a nuvem, que consolida os dados, e utiliza mecanismos para incrementar o poder de decisão dos outros veículos no futuro.

6. De onde os dados do modelo de originam, especialmente aqueles sobre os comportamentos dos agentes?

A maior parte dos dados que controlam o comportamento dos agentes-veículos são gerados pela própria rede, por outros veículos e pelas aplicações.

7. Como podemos validar o modelo, especialmente os comportamentos do agente?

O modelo deve ser validado utilizando duas métricas principais: percentual de requisições atendidas e redundância das aplicações. Esperamos atingir o balanço adequado dentro de certas características, de forma que seja possível manter as duas métricas dentro dos valores esperados. A quantidade de requisições atendidas deve estar muito próxima de 100%, enquanto a quantidade de redundâncias deve estar próxima de 0%.

4.4.2.2 Modelo de leilão para decisão sobre a resposta à requisição

Uma vez que a VFC é descentralizada, os métodos usuais de conexão cliente-servidor não são adequados (HONG et al., 2013). Desde o seu início, VANETs devem operar essencialmente com aplicativos de salto único, seja para fornecer serviços mais rápidos ou para evitar complexidade desnecessária (Kenney, 2011). Assim, aproveitamos protocolos padronizados para VANETs para decidir o melhor servidor para um cliente usando duas variações de um método de negociação amplamente conhecido, *Fire and Forget* (VOLTER; KIRCHER; ZDUN, 2013), aprimorado com alguns recursos que se assemelham a métodos de negociação baseados em leilão normalmente usados como mecanismos de coordenação em sistemas distribuídos (Kayal; Liebeherr,

2019) e *mechanism design* (RUSSELL; NORVIG, 2016; LUJAK; SLAVKOVIK, 2017). Argumentamos que o tempo de resposta pode ser melhorado removendo as negociações usuais entre cliente e servidor às custas de algum grau de redundância de processamento, que esperamos minimizar.

Utilizamos a primeira variação, que chamamos de "Tratamento de Resposta", como *baseline*. O cliente envia a solicitação de processamento à rede; todos os servidores ao alcance o recebem essa requisição e iniciam o processamento. Se o veículo tiver uma tarefa atribuída, a nova solicitação espera em uma fila. O veículo com maior poder de processamento e/ou a menor fila em determinado momento tende a responder primeiro. Todos os veículos percebem a resposta devido ao meio sem fio compartilhado e todos os outros servidores cancelam sua tarefa atual.

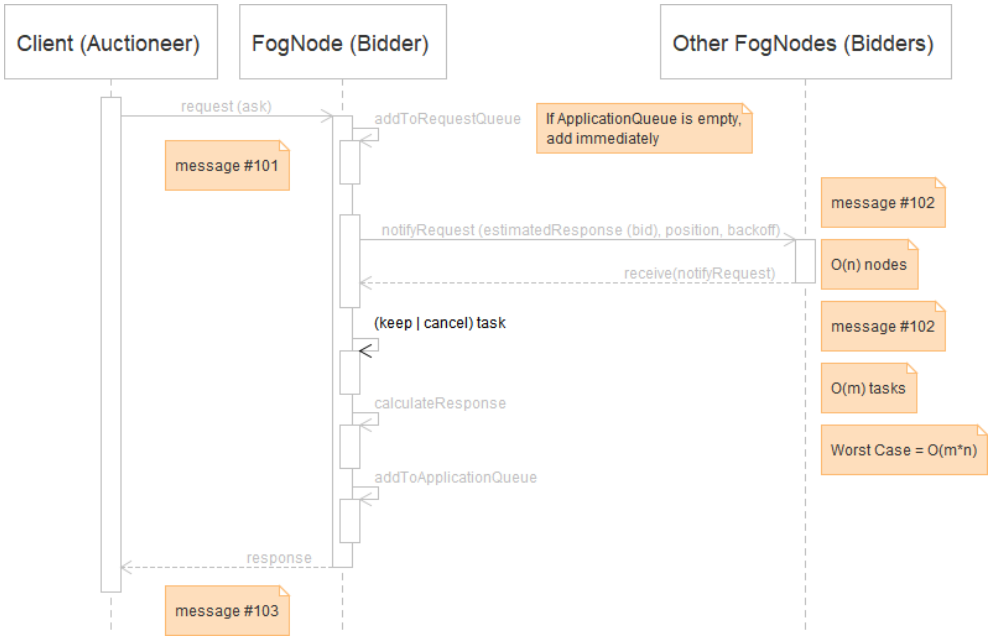
Essa variação tem desvantagens e vantagens. Por um lado, um veículo pode assumir a maior parte das tarefas, criando assim um processamento desnecessário e redundante em todos os outros veículos, que chamamos de servidor ganancioso. Por outro lado, os veículos trocam menos mensagens e a rede pode manter baixa utilização.

Confrontamos esse *baseline* com um One Shot Open First Price Auction (OSOFPA) com restrições de tempo que usa o tempo de processamento como lance. De acordo com os princípios do *Mechanism Design*, o objetivo desse mecanismo é viabilizar uma distribuição mais justa de recursos. Os veículos clientes desempenham o papel de vendedores, que disponibilizam as requisições de processamento, mas também de leiloeiros e transmitem suas tarefas à rede na forma de solicitações. Outros *Fog Nodes* desempenham o papel de licitantes dispostos a fazer lances para tarefas de processamento e tentar responder o mais rápido possível. O OSOFPA consiste em um leilão de rodada única com lances abertos, onde o lance mais baixo vence o leilão. Uma VFC pode conter vários leilões simultâneos onde existem subconjuntos independentes de clientes e servidores.

O funcionamento do leilão é ilustrado na Figura 25, assim como nos Algoritmos 1 e 2 . Assim como no *baseline*, todos os servidores em uma área recebem a solicitação de processamento através de *broadcast*. Assim que receberem uma solicitação, eles iniciam o processamento. Se os servidores estiverem processando alguma tarefa, eles colocarão a solicitação recém-recebida em uma fila de solicitações. Nesse ínterim, estimam o tempo de processamento da solicitação recebida, levando em consideração todo o tempo de processamento da fila, aproveitando os bancos de dados de *profiling* das aplicações.

O Algoritmo 1 demonstra o processo executado em, pelo menos, duas *threads* distintas no

Figura 25 – Funcionamento do leilão



Fonte: Elaborada pelo autor (2021)

servidor: uma voltada para gerenciamento e outra realmente para o processamento. Observamos a identificação do tipo de mensagem e a ação decorrente de cada tipo. O enfileiramento imediato, em caso de uma requisição, seguido pelo envio do lance (*bid*); e a análise dos lances que outros servidores enviaram para a rede. Outra *thead* fica responsável por monitorar a fila de tarefas recebidas, enfileiradas e concluídas.

Código Fonte 1 – Funcionamento Simplificado do Servidor

```
/* Thread 1
foreach message m do:
    if m == request then:
        enqueueForProcessing(m[uuid]);
        sendBid();
    end;
    if m == bid then
        if bid[uuid].estimated < this.bid[uuid].estimated then:
            stopProcessing(m[uuid]);
        end;
    end;
end;

/*Thread 2
if m[uuid].finished then:
    sendResponse(r[uuid]);
end
```

Fonte: Elaborada pelo autor (2021)

Já o algoritmo 2 demonstra um processo do lado do cliente, que envia as requisições e monitora quais delas estão sendo concluídas.

Código Fonte 2 – Funcionamento Simplificado do Client

```
/* Thread 1
while application.running() do:
    sendRequests();
end;

/* Thread 2
foreach response r do:
    if r[uuid] == m[uuid] then:
        accept();
    else
        ignore();
    end;
end;
```

Fonte: Elaborada pelo autor (2021)

Do ponto de vista formal, o OSOFPA começa quando clientes $C = \{C_1, C_2, C_3, ..., C_n\}$ geram mensagens em *broadcast* contendo uma requisição (*Ask*) de processamento (A_n). Quando um servidor recebe essa requisição, ele consegue estimar a carga que essa execução pode infligir ao seu sistema de processamento através do tipo de aplicação contido na mensagem, de acordo com o pressuposto de consultar um banco de dados de *profiling*. Os servidores $S = \{S_1, S_2, S_3, ..., S_m\}$ estimam o tempo de processamento com base na carga de (A_n) e transmitem um lance contendo o tempo estimado. A mensagem de rede também contém um identificador (ID) de solicitação e a posição atual que pode ser usada em implementações futuras. Seja H_m o lance do servidor S_m. A equação 4.1 demonstra que o tempo de resposta de um lance é determinado dividindo a carga da aplicação (A_n) e o poder de processamento (PP_m) do veículo adicionado pela soma das tarefas anteriores que ele pode estar processando atualmente, ou seja, quando há uma fila de processamento no *Fog Node*. Este é um lance aberto, assim outros servidores podem cancelar o lance ou a tarefa se sua estimativa interna for superior ao lance que eles percebem da rede.

Neste modelo, o vencedor é determinado de acordo com a Equação 4.2. O vencedor (*winner*) (W_n) do leilão é o servidor que envia o lance contendo a menor estimativa de tempo de resposta dentro das restrições de tempo. Assim, se houver pelo menos um lance $T(H_m)$ dentro dos requisitos do cliente $R(n)$, este é considerado o melhor lance. Para as avaliações em nível

de análise, desconsideramos o cálculo do requisito. O cálculo do requisito tem a tendência de apenas aumentar a taxa de requisições não atendidas. Assim, no capítulo 5 utilizamos uma regra de corte a partir dos resultados para determinar quais cenários são mais propensos a atender requisições de acordo com os requisitos de VFC.

Uma regra de desempate determina que o cliente escolha aleatoriamente quando houver lances repetidos. Este leilão é limitado pelo tempo, pois os clientes recebem uma resposta, mesmo se houver pelo menos um lance. Dadas as restrições de tempo, nem todos os servidores podem enviar lances em tempo hábil por algum motivo, como erros de rede. Assim, os clientes podem não receber o lance ideal em todos os leilões, mas receber um lance abaixo do ideal que atende aos requisitos. Mantivemos a notação consistente com (Peng; Ota; Dong, 2020) sempre que possível.

$$H_m = \frac{A_n}{PP_m} + \sum_{0}^{n-1} \frac{A_n}{PP_m} \tag{4.1}$$

$$W_n = \underset{m}{\operatorname{argmin}}(H_m), \exists H_m : T(H_m) < R(n) \tag{4.2}$$

Após a estimativa, cada servidor transmite uma notificação (lance) contendo informações suficientes para permitir que outros servidores mantenham ou cancelem suas tarefas. Neste momento, consideramos "ID da tarefa"e "tempo estimado de resposta"como informações cruciais para otimizar o tempo de resposta aos clientes. A identificação da tarefa é usado com um *hash* que permite que os servidores encontrem tarefas mais rapidamente na fila quando precisam cancelar algum processamento. O tempo de resposta estimado pode permitir que outros servidores comparem suas estimativas e cancelem a tarefa de menor tempo de resposta na notificação (lance).

Para reduzir a complexidade computacional, mantivemos a duração da fila como uma informação separada que é calculada de acordo com cada atualização (chegada de requisições ou cancelamento de processamentos). Gerenciamos cada alteração na fila, atualizando assim a duração da fila sempre que uma nova tarefa entra ou sai da fila.

Diferente do "tratamento de resposta", um leilão tende a não cair no problema de "servidor ganancioso"e tende a nivelar o nível de filas entre os servidores, especialmente se o servidor perceber uma taxa de solicitação significativa. Assim, o número de mensagens trocadas na rede pode aumentar com o número de servidores, especialmente se a notificação do servidor mais rápido atrasar ou apresentar erros de rede.

Na melhor das hipóteses, a notificação do servidor com a maior capacidade de processamento, ou o tempo de processamento estimado mais curto considerando a fila de tarefas, atinge todos os outros servidores permitindo que eles cancelem suas tarefas. Nesse caso, os servidores comparam seu "ID de tarefa"e "tempo de resposta estimado"com as informações do lance. Se o tempo de resposta estimado para esse "ID de tarefa"for maior do que o tempo de resposta no lance, eles cancelam a tarefa ou a removem da fila.

No pior caso, a notificação do servidor mais rápido pode ser a última a chegar nos outros, impedindo que todos os outros servidores cancelem suas tarefas. Além disso, essa resposta (lance) pode ser perdida devido a erros de rede. Além disso, essa notificação pode chegar ao servidor mais lento e fazer com que cancele a última tarefa da fila. Considerando essa natureza e o comportamento semelhante de ambas as abordagens, considerando os nós da rede como *threads* separados como "m"(RANGANATHAN; CAMPBELL, 2007), e as tarefas em uma fila como "n", a complexidade algorítmica de pior caso desses mecanismos é $O(m * n)$.

A latência de serviço é o tempo entre o envio da solicitação e o recebimento da resposta pelo cliente e afeta fortemente a complexidade temporal. Assim, implementamos o leilão usando *mechanism design* de forma a dar uma resposta garantida, embora algumas vezes abaixo do ideal sob condições e cenários variados. Esse modelo de leilão permite que os servidores respondam aos clientes de acordo com os requisitos de latência de serviço. Os servidores negociam pelo menor tempo de resposta possível. Nos melhores cenários, esperamos que, pelo menos, um servidor ofereça o menor tempo de resposta possível, mesmo se o leilão não esteja concluído e não haja um vencedor dentro das restrições de tempo.

Nesse contexto, podemos dizer que a latência representa a avaliação dos clientes e o custo dos servidores, uma vez que detém os principais requisitos para redes VFC. Para os clientes, representa o custo que eles têm para participar do *Fog*, ou seja, eles dependem da *Fog* para ter acesso a seus serviços. Para o servidor, representa seus custos, principalmente o enfileiramento de tarefas, e o tempo que ele estará ocupado com processamentos. Para ambos, quanto menor for a latência, maior será a avaliação, pois o cliente terá sua resposta em tempo, e o servidor estará livre para realizar novos processamentos . A equação 4.3 representa esta relação uma vez que os clientes só tendem a participar em leilões quando a sua avaliação para processar as suas tarefas V_c é inferior à avaliação para usar os serviços da *Fog* V_{fog}.

$$V_c > T_{\Delta t} + V_s \tag{4.3}$$

Como o objetivo do leilão é minimizar a latência do serviço para os clientes pretendemos encontrar o melhor cenário de acordo com as propriedades de bem-estar social (BRANDT; CONITZER; ENDRISS, ; LUJAK; SLAVKOVIK, 2017), considerando as funções de avaliação:

- *Bem-estar social Utilitário* : $u(C, S) = \sum_{c,s} V_{c,s}$
- *Bem-estar social Igualitário* : $e(C, S) = min_{c,s} V_{c,s}$
- *Bem-estar social Elitista* : $el(C, S) = max_{c,s} V_{c,s}$
- *Bem-estar social de Nash* : $n(C, S) = \prod_{c,s} V_{c,s}$

4.4.3 Análise Holística

No estado atual do trabalho, avaliamos o funcionamento de uma VFC completamente distribuída. A análise holística se destina a avaliar a integração da VFC com os outros estratos da Computação em Nuvem. Para essa integração, os dados oriundos das análises interna e periférica podem alimentar algoritmos online de ML que otimizem a distribuição de aplicações a partir de mecanismos externos.

É necessário escolher entre algoritmos online que possam ser alimentados por dados dos outros veículos quando eles tiverem oportunidade de trocar dados (TEERAPITTAYANON; MCDANEL; KUNG, 2017). Algoritmos desse tipo apresentam dois benefícios claros: os veículos poderão participar do cálculo da distribuição dos programas que possuem, através de análises usando seus estados interno e externo, e podem se valer dos mesmos algoritmos para tratar as aplicações IoT tradicionais.

4.5 CONTRIBUIÇÕES DA PROPOSTA FOGWISE

A principal contribuição da proposta FogWise está na análise e avaliação dos limites de uma VFC totalmente distribuída através de um mecanismo de leilão. Demonstramos um *framework* completo e detalhado, baseado em três etapas de análise das características de aplicações e dos veículos, além da identificação de princípios de design e cenários de funcionamento de uma VFC.

Pelas definições de IoV e *Fog*, observamos que existe um objetivo claro de reduzir a latência de serviços. Como existe um objetivo bem definido, essa característica aproxima essa análise

do *Mechanism Design* que busca propor mecanismos que maximizem a utilidade geral dos agentes (RUSSELL; NORVIG, 2016). Agentes, no caso os servidores, podem ter apenas uma decisão em relação a manipulação: aumentar seu tempo de resposta para, possivelmente, reduzir o tamanho das duas filas gradativamente. Assim, eles podem beneficiar os clientes atuais. Entretanto, tal abordagem é impraticável quando existe um incentivo econômico fixo em termos de tempo ou ciclos de processamento, que é o modelo mais comum de cobrança de nuvem. Em vez de deixar os servidores criarem suas próprias lógicas para participar da VFC,o ideal, neste caso, é alimentá-los com regras que busquem otimizar o bem-estar total.

5 AVALIAÇÃO E RESULTADOS

Nesta seção, são apresentados os experimentos realizados para avaliar o comportamento e a participação de veículos numa VFC completamente distribuída, em diversos níveis de mobilidade e diversas características de contato entre eles. Para isso, é preciso avaliar uma série de situações passando pela mobilidade, características internas previstas para processamento, até o protocolo de negociação necessário para a distribuição de tarefas, isto é, o modelo de leilão. Esta seção apresenta, também, a evolução da metodologia de avaliação da VFC distribuída desde uma versão sem as condições de mobilidade, até o momento em que essas características foram inseridas. Assim, as avaliações demonstradas aqui servem para determinar quais cenários, em termos de características e parâmetros, são mais ou menos propícios para os veículos participarem da VFC.

5.1 METODOLOGIA DE AVALIAÇÃO E PRESSUPOSTOS INICIAIS

A avaliação da proposta FogWise contém uma série de passos de acordo com as características apresentadas na Figura 22 que consiste em:

- Análise interna;
- Análise Periférica
- Análise Holística

5.2 ANÁLISE INTERNA

Nesse primeiro momento, foi investigado e modelado o comportamento interno de um nó *Fog* para, assim, determinar as características que guiarão as próximas etapas. Todas as nuances que resultam desta etapa, como levantamento de requisitos e resultados, servem de base teórica e metodológica para o protocolo de distribuição de tarefas na VFC. Destacamos que os parâmetros e métricas escolhidos servem para identificar como as interações entre a

dinâmica *Fog* e IoV pode contribuir para atender aos requisitos estritos das aplicações sempre buscando atender aos princípios de design.

O modelo de simulação inicial, que ainda não contém o arcabouço completo de rede e mobilidade, contém avaliações com limite de 10ms para latência. O valor de 10ms para a latência de serviço, descontando o valor da rede que será demonstrado na análise periférica, foi escolhido como referência por ser mencionado como um valor a ser atingido para aplicações móveis de missão-crítica (*mission-critical mobile applications*) (BONOMI et al., 2012; SOUA; TOHME, 2018; OUYANG; ZHOU; CHEN, 2018). Para as avaliações que já contêm o arcabouço completo, utilizamos o limiar de 50 ms, sugerido pelo ETSI, nas simulações (ETSI, 2018; SARKAR; MISRA, 2016). Mesmo assim, os resultados das simulações, no Apêndice A indicam que é possível alcançar limiares com latência de serviço ainda menores.

A análise interna contém uma série de etapas a serem executadas em busca dos objetivos desejados:

- avaliar o comportamento das aplicações e requisições em relação ao estado interno do veículo;
- determinar as características de mobilidade de acordo com percepção do estado interno do veículo;
- determinar o tempo que os veículos dispõem para trocar dados de uma transação, de acordo com as velocidades, vazão disponível e latência de requisição/processamento;
- selecionar e justificar os parâmetros e métricas selecionados, especialmente aqueles que forem inovadores em relação à literatura, dando especial atenção àqueles que conseguem atingir os requisitos das aplicações, especialmente deixando o tempo total do serviço abaixo dos 50ms.

Durante essa análise, houve a necessidade de criar um modelo que se adequasse à necessidade que cada veículo pudesse agir como servidor. O modelo de simulação inicial foi desenvolvido em Java[1] e foi adaptado para ser integrado ao simulador Omnet++(VARGA, 1999). Na análise periférica, descrita na próxima seção, as adaptações serviram para avaliar especialmente as características de mobilidade, que os trabalhos relacionados avaliam parcialmente. Assim, para o contexto de migração e serviço identificamos que os parâmetros de interesse são:

[1] https://bitbucket.org/ffmjunior/thesis-fog-vanet/src/master/

1. capacidade do dispositivo;
2. carga gerada pela aplicação;
3. vazão da rede;
4. tamanho da aplicação;
5. taxa e tipos de requisições por unidade de tempo;
6. quantidade e tipos de aplicações.

A simulação consiste num *Fog Node* recebendo requisições. Esse dispositivo contém uma capacidade computacional e ele recebe uma série de requisições de vários tipos e sob determinadas taxas. Cada requisição vem de uma aplicação que acarreta uma determinada carga ao dispositivo e isso resulta em um tempo de processamento. As requisições são enfileiradas e processadas em ordem. Do enfileiramento, conseguimos obter métricas relacionadas com o tempo de execução e a taxa de atendimento de vários tipos de aplicações sob várias taxas de requisições de acordo com as tabelas 5, 6, e 7, conforme o experimento.

5.2.1 Relação entre a Capacidade do Dispositivo e a Carga da Aplicação

Os parâmetros 1 e 2 são estreitamente relacionados com as métricas de servidores tradicionais (JAIN, 1990). Na análise interna, carga e capacidade estão modelados em termos de seu desempenho relativo, ou seja: existe uma unidade virtual de processamento por unidade de tempo. Dada uma carga de uma determinada aplicação, ela será processada em quantas unidades de tempo forem necessárias de acordo com seu valor. Daí, obtemos seu tempo de execução.

Esse modelo se assemelha a Millions of Instructions Per Second (MIPS), que apesar de sua simplicidade, conforme reconhecido em (JAIN, 1990), ainda tem sido usado em simuladores recentes no contexto *Fog*, de forma que justificamos sua adoção (SARKAR; MISRA, 2016; GUPTA et al., 2017). Destacamos, ainda, que utilizamos uma modelagem que pode ser convertida para outras medidas de desempenho de sistemas de acordo com a necessidade, como MIPS, Millions of Floating-Point Operations Per Second (MFLOPS) ou outras medidas que a literatura possa fornecer. Isso foi feito de forma a facilitar uma posterior implementação em simulador (Omnet++). Além disso, esse modelo pode ser adaptado para variações que diferenciam

aplicações com diferentes pesos para características como processamentos lógico-aritméticos, armazenamento, desvios de fluxo, vetores e cache (JAIN, 1990).

Inicialmente, o modelo suporta a investigação da relação entre poder de processamento dos veículos e as características das aplicações em funcionamento nesse tipo de ambiente. O gráfico da Figura 26 demonstra uma relação entre essas características. Os experimentos fatoriais demonstram que a relação entre a carga das aplicações e a capacidade dos dispositivos deve ser maior que 1/100 para que o serviço se mantenha abaixo dos 50ms recomendados pelo ETSI, especialmente quando se almeja atingir os 10ms das aplicações de missões-críticas.

As Figuras 26a e 26b demonstram a mesma avaliação, mas com taxas de requisições com λ igual a 1/50 e 1/250, respectivamente. Observamos que a diminuição da taxa gera uma redução significativa da latência quando a relação capacidade-carga é menor que 1/100. Quando a relação é maior, há pouca variação na latência do serviço.

Tabela 5 – Parâmetros para a Demonstração da Relação entre a Capacidade do Dispositivo e a Carga da Aplicação

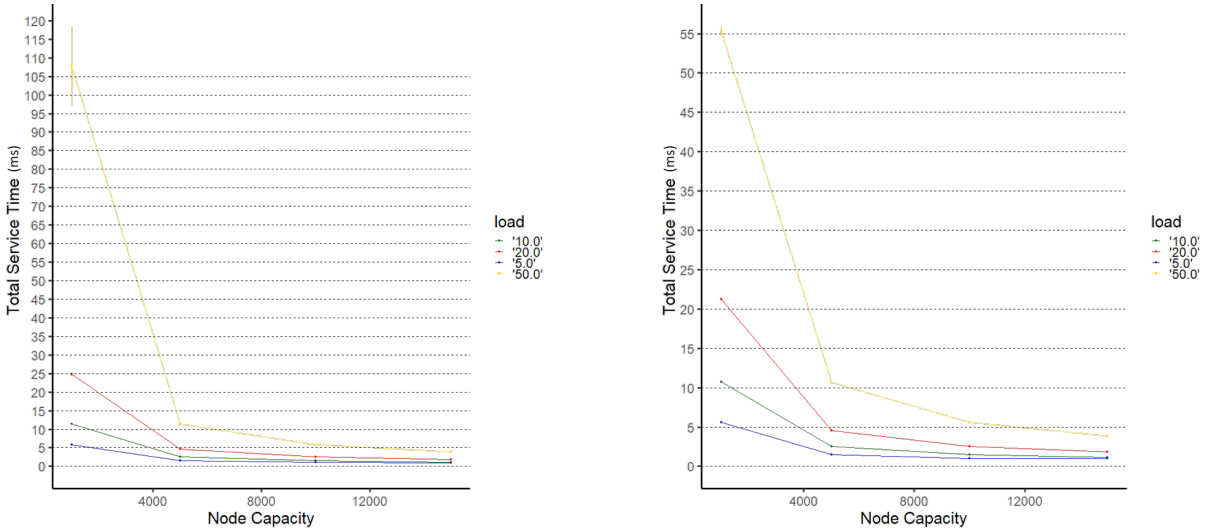
Parâmetro	Experimento 2	Experimento 6
Vazão da Rede (Mbps)	10	10
Tamanho Médio da Aplicação (MB)	18	18
Desvio Padrão do Tamanho da Aplicação (MB)	10	10
Tipo de Aplicação	10	10
Desvio Padrão do Tipo de Aplicação	1	1
ID de Requisição	10	10
Desvio Padrão do ID da Aplicação	3	3
Duração da Simulação (h)	1	1
Taxa de Requisições (λ)	50	250
Capacidade do Dispositivo	1000, 5000, 10000, 15000	1000, 5000, 10000, 15000
Carga das aplicações	5, 10, 20, 50	5, 10, 20, 50

Fonte: Elaborada pelo autor (2021)

5.2.2 Aplicações e Requisições: Distribuição, Migração e Tipos

Após a avaliação capacidade-carga, inicia-se uma etapa fundamental para a compreensão da cadeia *Fog*: a relação entre aplicações e requisições. Os parâmetros "tamanho da aplicação"e "vazão"são importantes para avaliar a distribuição das aplicações. Essa relação pode ajudar a determinar com quanta antecedência pode ser necessário realizar o download, migração, replicação das aplicações para que o serviço seja satisfatório. Por fim, os dois últimos

Figura 26 – Demonstração da Relação entre a Capacidade do Dispositivo e a Carga da Aplicação



Fonte: Elaborada pelo autor (2021)

parâmetros (5 e 6), relacionados com quantidades e tipos de requisições e aplicações, ajudam a modelar como o dispositivo *Fog* percebe o ambiente no qual está inserido.

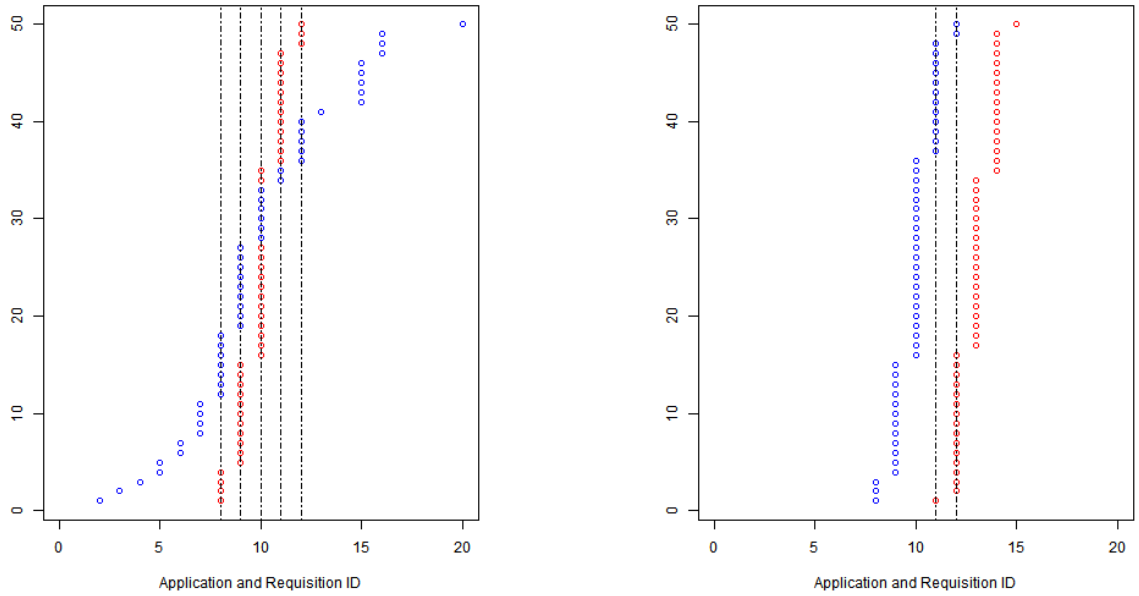
A taxa e o tipo de requisições pode ser derivado da quantidade de dispositivos IoT, IoV, ou mesmo de usuários, nas redondezas e da tipo de aplicação que cada um carrega, tudo isso em função do tempo e da velocidade do veículo. Os dois sub-parâmetros, quantidade e tipo, são modelados separadamente pois possuem funções diferentes nos cenários. Cada um deles é representado por uma distribuição de probabilidade que determina seus valores. Atualmente, a taxa de requisições está representada por uma distribuição exponencial, com parâmetro λ; e o tipo de requisição está representada por uma distribuição Normal, das quais extraímos valores inteiros representando tipos de aplicações. Com isso, esperamos modelar, por exemplo, a existência de aplicações mais comuns e mais raras entre todas aquelas previstas na literatura.

Uma metodologia semelhante é aplicada ao último parâmetro, em especial aos tipos de aplicações. As quantidades e tipos de aplicações são determinadas por distribuições Normais. Com isso, é possível filtrar as aplicações que um veículo pode executar em meio a todas as requisições que ele recebe. Também é possível usar as distribuições variando seus parâmetros de média e desvio, para modelar diversos cenários onde os veículos possuem mais ou menos aplicações, ou tipos de aplicações, em relação ao total de aplicações do ambiente.

Essa lógica está representada na Figura 27 onde é possível observar uma situação em que um veículo (pontos vermelhos) possui menos aplicações que o total do ambiente (pontos azuis) ou possui uma parte delas. As linhas tracejadas na vertical indicam as correspondências,

ou seja, as situações onde as requisições encontram aplicações que podem atendê-las dentro dos veículos. Os valores obtidos a partir das distribuições representam identificadores das requisições ou das aplicações. Quando uma aplicação recebe uma requisição cuja identificação corresponde às aplicações que o veículo contém, então aquela requisição pode ser satisfeita. Salientamos que a distribuição normal pode ser substituída por qualquer outra para se adequar às futuras avaliações empíricas ou à revisões de literatura.

Figura 27 – Relação entre requisições e aplicações



(a) (b)

Fonte: O autor

Em relação às aplicações, e seu contexto de migração, requisição e resposta, o modelo contém aplicações que podem ser baixadas da RSU, a taxa de download para essas aplicações, além das características de taxa de requisições e tipos. O modelo será estendido futuramente para a migração entre veículos. A Figura 28 contém um gráfico indicando a taxa de resposta às requisições que um nó *Fog* recebe de acordo com a vazão da rede. Os parâmetros para esse experimento estão na Tabela 6. Esse é um resultado importante pois, futuramente, pode indicar a antecedência necessária para ativar a migração/replicação das aplicações. Quanto menor a vazão da rede, menos aplicações são atendidas. Isso ocorre porque o veículo demora mais para receber o código das aplicações e pode ser agravado pelo aumento na quantidade de tipos de aplicações. A queda de vazão pode ser causada por vários fatores como interferências, quantidade de veículos e erros na rede, que serão avaliados com mais detalhes na fase de

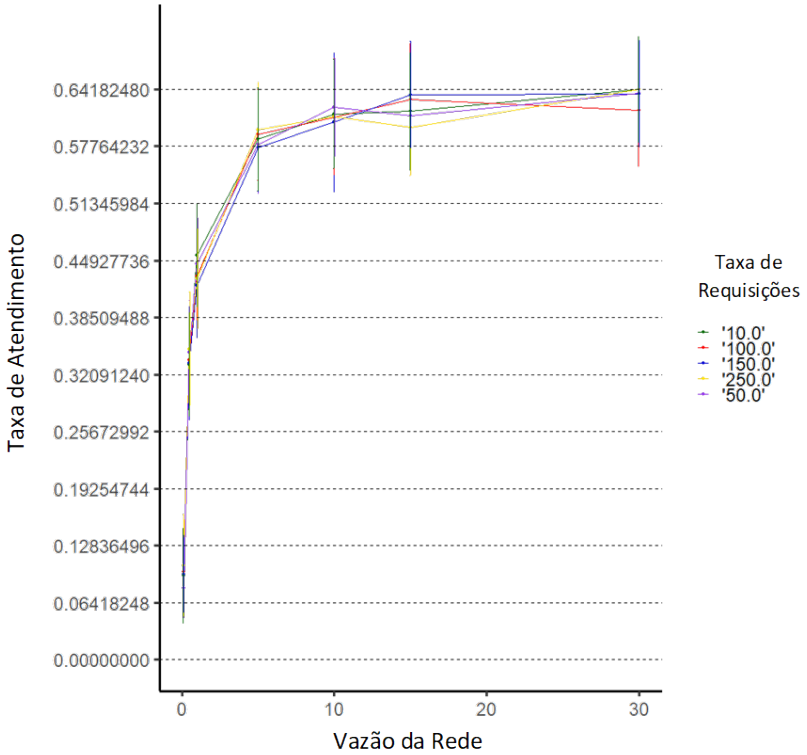
avaliação periférica e holística. A taxa de requisições não impacta diretamente na taxa de atendimento.

Tabela 6 – Parâmetros para o experimento 28

Parâmetro	Experimento
Vazão da Rede (Mbps)	0.1, 0.5, 1.0, 5.0, 10.0, 15.0, 30.0
Tamanho Médio da Aplicação (MB)	18
Desvio Padrão do Tamanho da Aplicação	10
Tipo de Aplicação	10
Desvio Padrão do Tipo de Aplicação	1
ID de Requisição	10
Desvio Padrão do ID da Aplicação	3
Duração da simulação (h)	1
Taxa de Requisições (λ)	10, 50, 100, 150, 250
Capacidade do Dispositivo	5000
Carga das aplicações	5

Fonte: Elaborada pelo autor (2021)

Figura 28 – Taxa vs Atendimento



Fonte: O autor

As avaliações apresentadas na Figura 29, cujos parâmetros estão na Tabela 7, demonstram o impacto da variabilidade das aplicações em relação ao atendimento às requisições. O principal mote dessa avaliação é a comparação entre a variabilidade das aplicações. A avaliação 29a

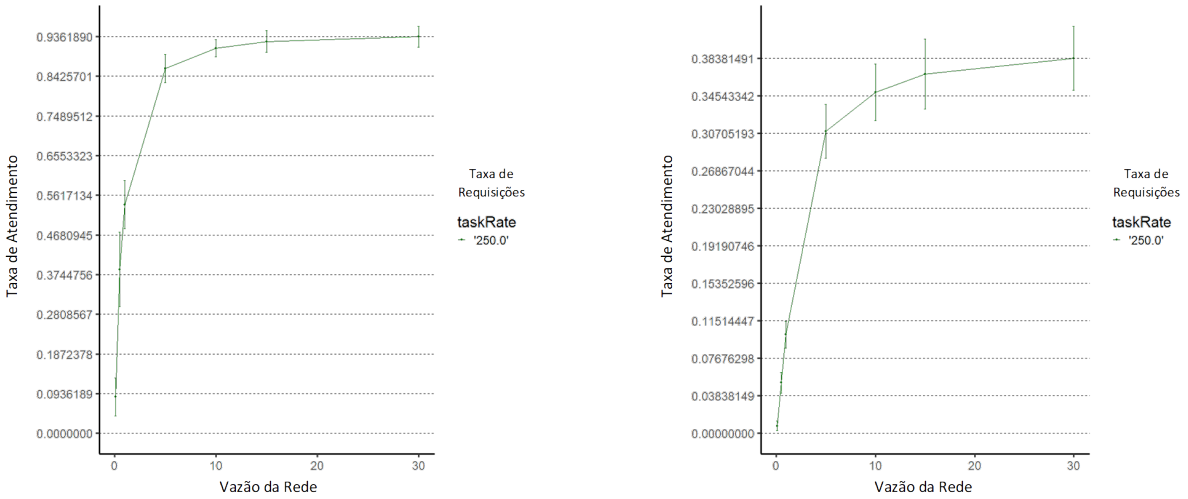
possui identificadores das aplicações e das requisições com alta correspondência. Nesse caso, a pouca variabilidade na quantidade de aplicações gerou uma taxa de atendimento que chega a 93% quando a vazão da rede permite o download/migração das aplicações em tempo hábil. A avaliação 29b também possui identificadores requisição-aplicação com alta correspondência, mas a variabilidade é muito mais alta, em termos de tipos de aplicações/requisições. Mesmo quando a rede provê altas taxas de download, a taxa de serviço chega a apenas 38% dentro do tempo de simulação, e não apresenta tendência de melhoras.

Tabela 7 – Parâmetros para a Demonstração da Relação entre a Capacidade do Dispositivo e a Carga da Aplicação

Parâmetro	Experimento 29a	Experimento 29b
Vazão da Rede (Mbps)	0.1, 0.5, 1.0, 5.0, 10.0, 15.0, 30.0	0.1, 0.5, 1.0, 5.0, 10.0, 15.0, 30.0
Tamanho Médio da Aplicação (MB)	18	18
Desvio Padrão do Tamanho da Aplicação	10	10
Tipo de Aplicação	10	100
Desvio Padrão do Tipo de Aplicação	3	30
ID de Requisição	10	100
Desvio Padrão do ID da Aplicação	3	30
Duração da Simulação (h)	1	1
Taxa de Requisições (rps)	250	250
Capacidade do Dispositivo	5000	5000
Carga das aplicações	5	5

Fonte: Elaborada pelo autor (2021)

Figura 29 – Demonstração da Relação entre a variação de vazão e tipos de aplicações com a taxa de atendimento de requisições. Em (a), as distribuições que representam as aplicações e as requisições possuem as mesmas características. Em (b), existe uma diferença em relação ao desvio padrão dos identificadores das aplicações e das requisições, gerando uma quantidade de requisições que não são atendidas devido à falta de aplicações baixadas.



(a) (b)

Fonte: Elaborada pelo autor (2021)

5.3 ANÁLISE PERIFÉRICA

5.3.1 Resultados preliminares da modelagem do veículo como Agente Inteligente

Nesta etapa, o maior desafio é integrar a modelagem do ambiente (mobilidade e requisições) os dados adquiridos com as modelos de funcionamento das aplicações e das características de processamento dos veículos. Um dos principais objetivos desta tese é encontrar as situações em que a latência dos serviços VFC atende à aplicações com requisitos críticos.

A modelagem inicial consistiu no agente mais simples e que praticamente já está implementado na modelagem interna. Trata-se do agente que responde à qualquer requisição que recebe, sem utilizar seus conhecimentos prévios ou informações extras para tomar a decisão sobre responder ou não. Esse tipo de agente gera muito desperdício de recursos computacionais pois vários veículos podem responder à mesma requisição, caso os *clusters* não estejam ajustados e as informações internas não estejam compatíveis.

Num segundo momento, foi desenvolvido um agente reativo que responde às requisições às quais ele acredita que pode prover uma resposta em menor tempo. Para isso, o agente utilizar um limiar preestabelecido para a relação capacidade/carga, conforme os resultados

da modelagem interna. O limiar foi estabelecido de acordo com os resultados apresentados nos gráficos da figura 26b. Determinamos que um veículo só pode responder às requisições caso ela consiga manter a latência de resposta abaixo dos 10ms, que é bem inferior aos 50ms sugeridos para várias aplicações críticas da Internet dos Veículos, especialmente no que se trata de segurança dos passageiros (ETSI, 2018) .

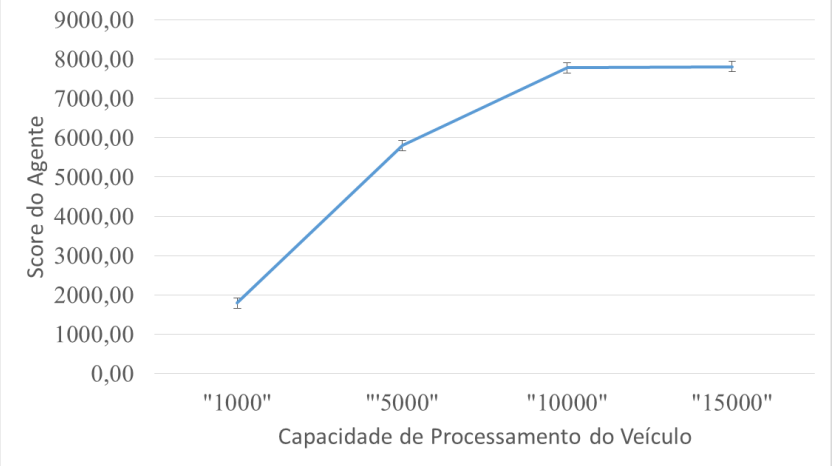
Nas simulações com o agente reativo que utiliza apenas as informações de capacidade/-carga, foram utilizados os mesmos parâmetros dos experimentos 2 e 6, que estão elencados na Tabela 26. Como medida inicial de desempenho determinamos que quando o agente responde a uma requisição ele ganha uma pontuação de 10, e quando não responde a uma requisição, ele ganha uma pontuação de -1. Cada agente realiza 1000 rodadas de tentativas de responder a uma requisição que recebe. Esse experimento foi repetido 100 vezes com cada combinação de parâmetros para a obtenção dos resultados. Observamos um comportamento ascendente com tendência a uma estabilização conforme a capacidade de processamento do veículo aumenta. Esses resultados podem indicar, após análises mais aprofundadas, que veículos com maior poder de processamento tentem a atender mais requisições.

Tabela 8 – Resultados para os Scores dos agentes

Capacidade	Desempenho	Desvio Padrão
1000	1796,73	119,11
5000	5801,24	142,59
10000	7777,34	130,88
15000	7806,6	137,02

Fonte: Elaborada pelo autor (2021)

Figura 30 – Gráfico de scores do agente de acordo com as relações capacidade/carga



Fonte: Elaborada pelo autor (2021)

Os próximos passos tratam da adição de mais informações ao modelo interno do agente para que ele possa tomar as decisões sobre responder ou não a uma requisição com o menor tempo. Esperamos que isso gere uma variabilidade nos resultados, e que as respostas às requisições comecem a ser dadas por outros veículos que não apenas aqueles com maior poder computacional, especialmente quando a mobilidade for levada em consideração. Ao mesmo tempo, esperamos que não haja duplicação de respostas na rede, pois isso pode indicar desperdício de recursos. Nesse caso, uma das principais medidas que pretendemos otimizar é a redundância de respostas obtidas.

5.3.2 Avaliação de leilão para alocação de recursos em VFC

Utilizamos o simulador VEINS(Sommer; German; Dressler, 2011) para realizar a avaliação do FogWise. VEINS é uma plataforma para simular aplicações veiculares com fidelidade. Esta plataforma de simulação híbrida foi acoplada bidirecionalmente a simuladores comumente usados e validados, ou seja, o simulador de rede OMNeT ++(VARGA, 1999) e o simulador de tráfego microscópico SUMO(Lopez et al., 2018). VEINS fornece várias extensões para simulação precisa de redes veiculares e aplicações, nomeadamente modelos de camadas inferiores DSRC/WAVE, modelos de propagação personalizados, cenários veiculares representativos, entre outros. Os dois métodos para tratamento de solicitações foram implementados na instância do veículo no OMNet++.

O objetivo dessa integração é investigar e avaliar o comportamento interno de um nó *Fog* (ou seja, um veículo) sob certas condições de densidade de veículos, taxa de solicitação e mobilidade. Quando uma simulação começa, o núcleo da simulação seleciona aleatoriamente alguns veículos para se tornarem clientes e os outros se tornam servidores, ou seja, *Fog Nodes* de acordo com o parâmetro *Client Ratio*. Em seguida, os clientes começam a gerar solicitações de acordo com determinados padrões de aplicações reais. Os *Fog Nodes* nas proximidades recebem essas solicitações e negociam sobre como processá-las. Comparamos dois métodos de tratamento de solicitações: i) *baseline* baseado na resposta e ii) baseado em leilão.

Avaliamos FogWise referenciando parâmetros e métricas de trabalhos relacionados discutidos na Seção 3 e listados nas tabelas daquela seção. Ambas as propostas foram avaliadas em um cenário de Manhattan com 300 m de lado, criando um ambiente microscópico de simulação com área controlada. Dessa forma, pretendemos isolar melhor o impacto dos parâmetros sobre os resultados. Com esse cenário, buscamos modelar a mobilidade através da entrada e

Tabela 9 – Parâmetros dos Experimentos

Parâmetro	Valor
Mapa	
Mapa	Manhatam 300 m (lado)
Densidade de Veículos (veículos por km²)	33, 44, 111, 155
Rede	
Vazão da Rede (Mbps)	27
Clientes	
Client Ratio	1/2; 1/3
Taxa de Requisições (por segundo)	1, 10, 33.3, 100
Média do Tipo de Aplicação	100
Dispersão do Tipo de Aplicação	1, 5, 10
Média do Tipo de Requisição	100
Dispersão do Tipo de Requisição	1, 5, 10
Média do Tipo de Aplicação	1000.0
Dispersão do Tipo de Aplicação	200.0
Tamanho da Requisição (B)	0, 300, 2197, 16695, 34951
Fog Nodes (servidores)	
Capacidade de Processamento	100000
Dispersão da Capacidade de Processamento	5000
Método de Notificação	1, 2

Fonte: Elaborada pelo autor (2021)

saída dos veículos do ambiente microscópico de simulação.

Variamos a proporção cliente/servidor(Peng; Ota; Dong, 2020) e a densidade de veículos para criar cenários de acordo com a literatura. Argumentamos que a densidade impacta criticamente o desempenho das redes sem fio gerando problemas como Taxa de Erro de Bit - Bit Error Rate (BER), - erros de mensagem e interferência. Podemos avaliar a taxa geral de solicitação que um servidor pode esperar em uma VFC totalmente descentralizada, mas mantendo as garantias de QOS com base nas taxas de solicitação de cada cliente. Extraímos valores de densidade de veículos de outros trabalhos(Peng; Ota; Dong, 2020; Rivoirard et al., 2017), e os listamos na Tabela 9.

Uma boa parte das aplicações em IoV envolve processamento de multimídia e agregação de sensores (Lin et al., 2018; YANG et al., 2017; MOURADIAN et al., 2017) . Construímos uma simulação onde alguns veículos atuam como clientes e alguns atuam como servidores (Peng; Ota; Dong, 2020). Assim, programamos veículos para gerar solicitações que imitam essas aplicações em relação ao tamanho e à taxa da mensagem. Os clientes geram solicitações a taxas próximas daquelas de aplicações reais, que são: 1 RPS como linha de base; 10 RPS para representar um processamento de imagem em lote (Rivoirard et al., 2017; Zhang et al., 2019); 33 RPS para

representar o processamento de imagem única e 100 RPS(Rivoirard et al., 2017) para encontrar casos extremos, bem como limites de rede / processamento.

Também avaliamos como os *Fog Nodes* funcionam e como a rede se comporta quando os tamanhos de solicitação se assemelham a aplicativos reais, como processamento de imagem e vídeo:

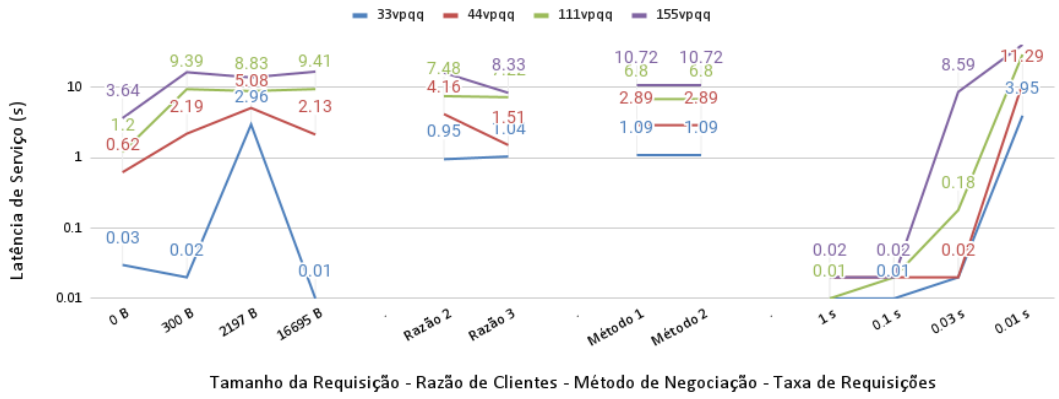
- 0 byte é a linha de base. Isso representa uma solicitação simulada com alguns identificadores para o servidor manipulá-la;
- 300 bytes para aplicações de telemetria de redes veiculares23;
- 2197, 16695 e 34951 bytes referem-se a tamanhos de quadros de vídeos com resoluções 144p, 480p e 720p, respectivamente. Essas são resoluções comuns de vídeos de redes sociais e de aplicativos que processam imagens e vídeos.

FogWise modela capacidade de processamento e carga das aplicações termos de desempenho relativo, ou seja, há uma unidade de processamento virtual por unidade de tempo. Uma aplicação será processada em quantas unidades de tempo forem necessárias de acordo com seu valor, dada a sua carga. Portanto, obtemos o tempo de execução. Assim, nos assemelhamos a Milhões de instruções por segundo (MIPS), mas abstraindo detalhes do hardware, que é uma falha conhecida do modelo(JAIN, 1990; SARKAR; MISRA, 2016; GUPTA et al., 2017). Embora pareçam simples, são uma *baseline* importante para as avaliações de heterogeneidade a seguir.

Propomos uma medida de dispersão para as métricas "Dispersão da Capacidade de Processamento"e "Dispersão da Carga da Aplicação"para adicionar um componente de variabilidade, e possível realidade, aos experimentos. Argumentamos que a capacidade de processamento e a carga do aplicativo estão sujeitas a variabilidade desconhecida no tempo de processamento, dependendo das interações complexas entre os componentes computacionais do servidor, e que há um limite para a complexidade das simulações e modelos. Assim, optou-se por representar essa variabilidade como valores resultantes de distribuições de probabilidade(JAIN, 1990).

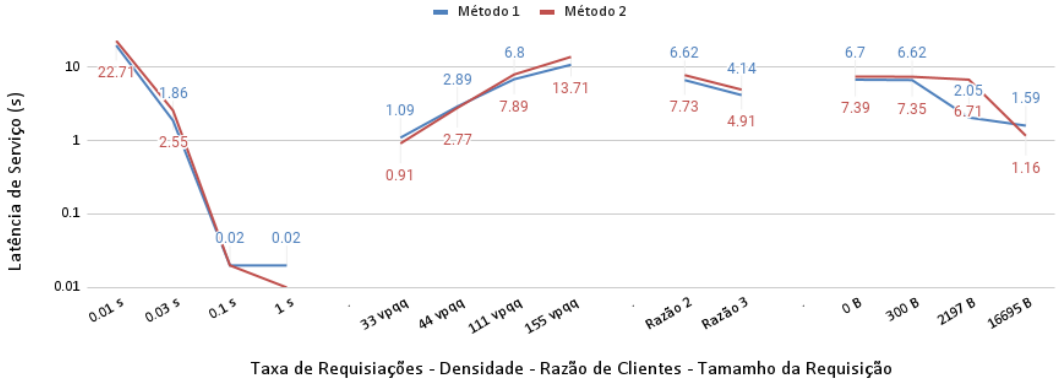
Os parâmetros Capacidade de Processamento e Carga da Aplicação são configurados de acordo com a Tabela 9. Consideramos que esses valores indicam que o veículo pode processar aplicações em tempo real, como detecção de objetos, e respeitar os requisitos: cerca de 33,3 ms para cada quadro em um vídeo rodando a 30 fps, possivelmente incluindo latência de rede.

Figura 31 – Latência de Serviço - Densidade de Veículos



Fonte: Elaborada pelo autor (2021)

Figura 32 – Latência de Serviço - Tamanho da Mensagem

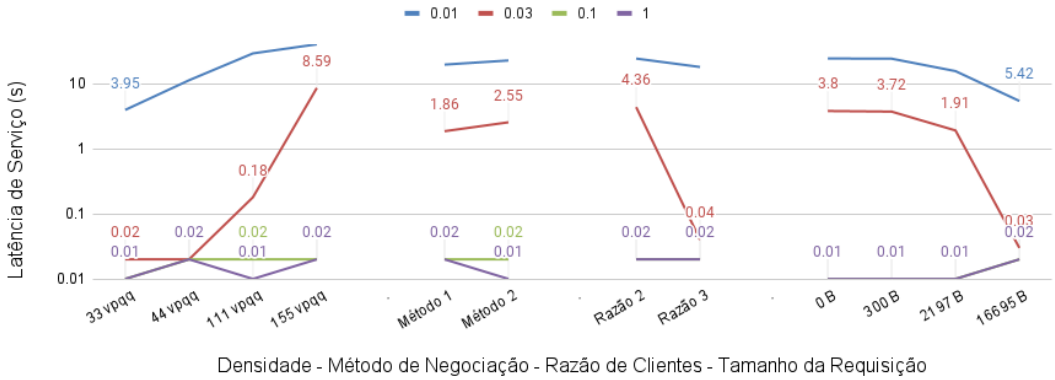


Fonte: Elaborada pelo autor (2021)

5.3.3 Metodologia de avaliação: Clientes

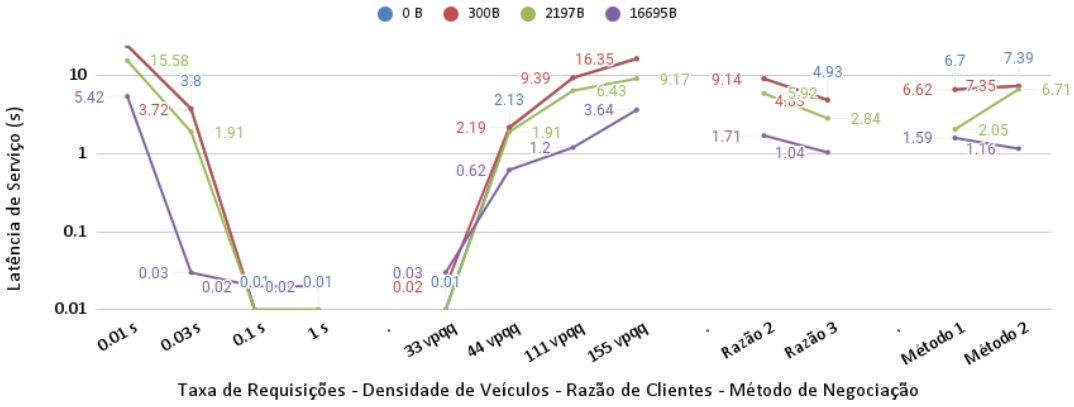
Realizamos experimentos em uma ampla gama de métricas relacionadas à VFC. As Métricas de Cliente e Métricas Comuns expressam os detalhes de cada cenário, especialmente em relação às requisições e densidades. As Métricas de Servidor representam o funcionamento de veículos como *Fog Nodes*. O ETSI definiu que as aplicações para a segurança dos passageiros (por exemplo, direção autônoma e detecção de pedestres) precisam manter seu funcionamento abaixo de 50 ms (ETSI, 2018; SARKAR; MISRA, 2016). Este valor pode se referir a aplicativos relacionados à direção autônoma, direção tele-operada (d'Orey et al., 2016), notificações ao motorista, prevenção de acidentes e outros, que envolvem, inerentemente, processamento de vídeo e agregação de dados de sensores. Com isso, baseamos nosso argumento na necessidade de que qualquer aplicação deve atender a esse requisito, mesmo funcionando em VFC. Portanto, usamos esse valor como uma referência para encontrar os melhores conjuntos de características

Figura 33 – Latência de Serviço - Taxa de Requisições



Fonte: Elaborada pelo autor (2021)

Figura 34 – Latência de Serviço - Tamanho da Requisição



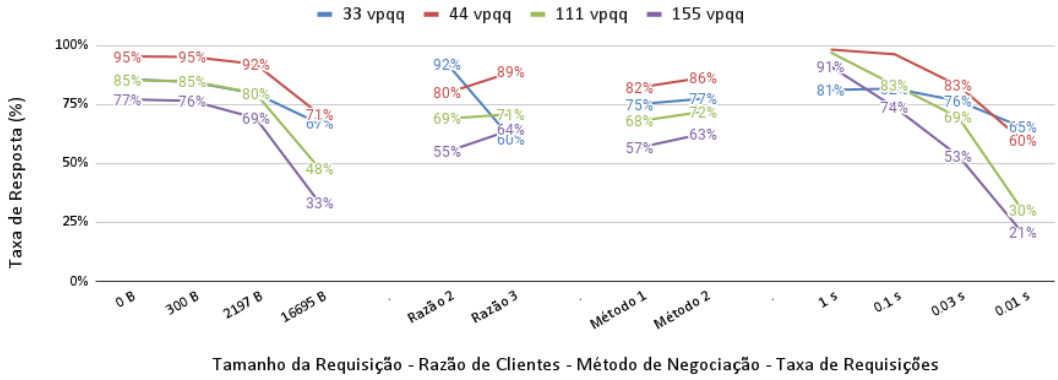
Fonte: Elaborada pelo autor (2021)

que permitem que os veículos participem de uma VFC completamente distribuída.

As seguintes métricas relacionadas ao cliente, rede e leilão são consideradas:

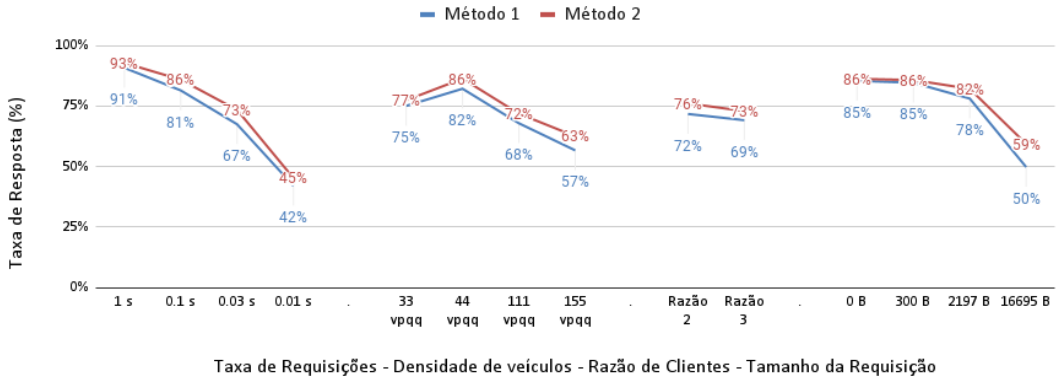
- **Latência de Serviço (L-svc)** [Cliente]: é definido como o intervalo de tempo entre a geração de uma requisição e o recebimento da resposta;
- **Taxa de Resposta (T-res)** [Cliente]: é definida como a parte das solicitações que receberam resposta adequada. Esta métrica pode indicar se a rede ou os *Fog Nodes* estão com alta utilização. Em termos de sistemas distribuídos, essa métrica também indica a Disponibilidade e a Confiabilidade do sistema de acordo com o cenário apresentado.
- **Pacotes perdidos por SNIR** [Rede]: é definido como o número total de pacotes perdidos por erros de bit.

Figura 35 – Taxa de Resposta - Densidade de Veículos



Fonte: Elaborada pelo autor (2021)

Figura 36 – Taxa de Resposta - Método de Leilão

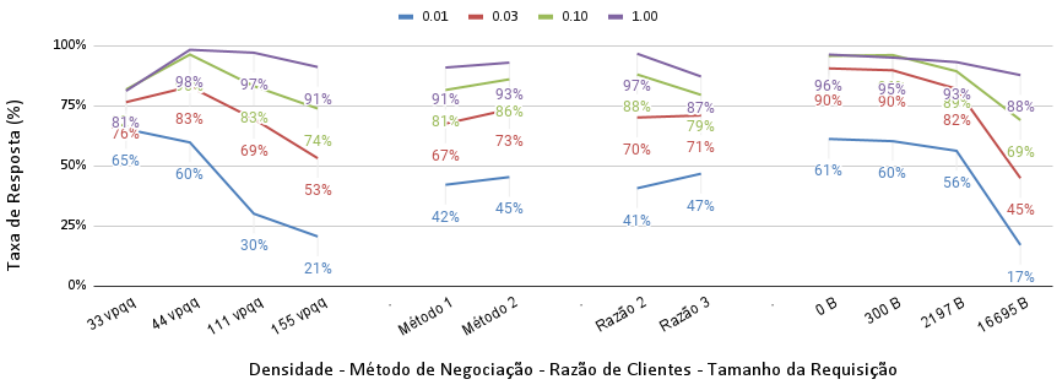


Fonte: Elaborada pelo autor (2021)

- **Pacotes perdidos por RXTX** [Rede]: é definido como o número total de pacotes perdidos quando ocorrem colisões de pacotes.
- **Requisições de Terceiros** (R-ter) [Leilão]: registra as respostas recebidas por um veículo que não se destinava a ele. Isso é causado principalmente devido ao meio sem fio compartilhado.
- ***Overhead* de Requisições** (O-req) [Leilão] - mede as respostas duplicadas quando dois ou mais veículos acabam respondendo à mesma solicitação mesmo após o leilão. Podemos usar essa métrica para avaliar a eficácia do Protocolo de Leilão.

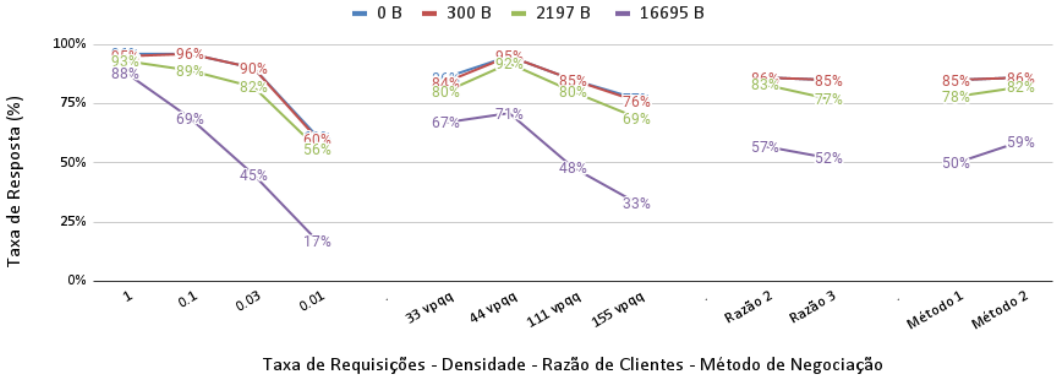
Realizamos experimentos fatoriais com todas as combinações de parâmetros. Coletamos os valores das métricas de cada cliente e de cada servidor. Finalmente, calculamos a média de todos os valores para o conjunto de veículos e consolidamos os resultados como uma média para cada simulação. Em seguida, individualizamos os fatores para identificar sua relevância

Figura 37 – Taxa de Resposta - Taxa de requisições



Fonte: Elaborada pelo autor (2021)

Figura 38 – Taxa de Respostas - Tamanho da Requisição



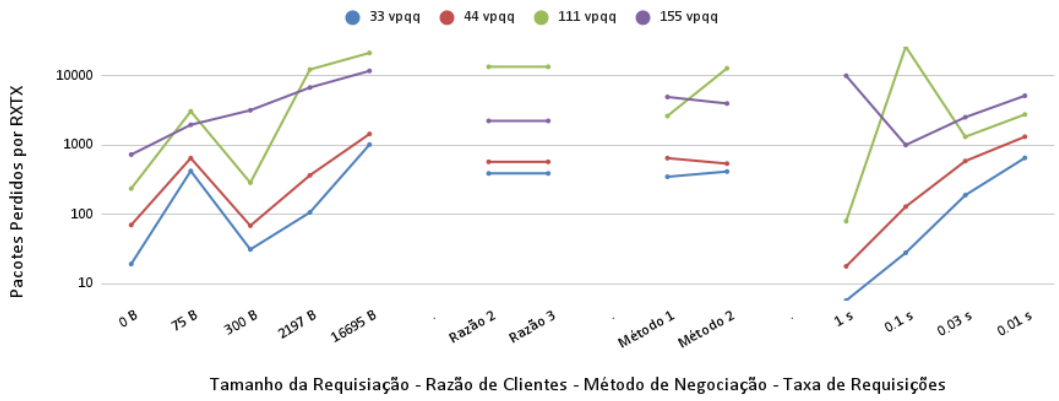
Fonte: Elaborada pelo autor (2021)

dentro de cada cenário. A quantidade de amostras dentro de cada simulação nos permite usar o Teorema do Limite Central para justificar que as amostras dentro de cada experimento formam uma distribuição normal. Assim, comparamos as médias de cada métrica para cada uma das simulações (JAIN; PAUL, 2013). Por fim, realizamos testes estatísticos para avaliar a diferença entre as médias de cada simulação. Verificamos que todas as amostras isoladas por fatores apresentam diferença estatisticamente significativa ($p < 0{,}05$).

5.3.3.1 Avaliação em nível de Rede

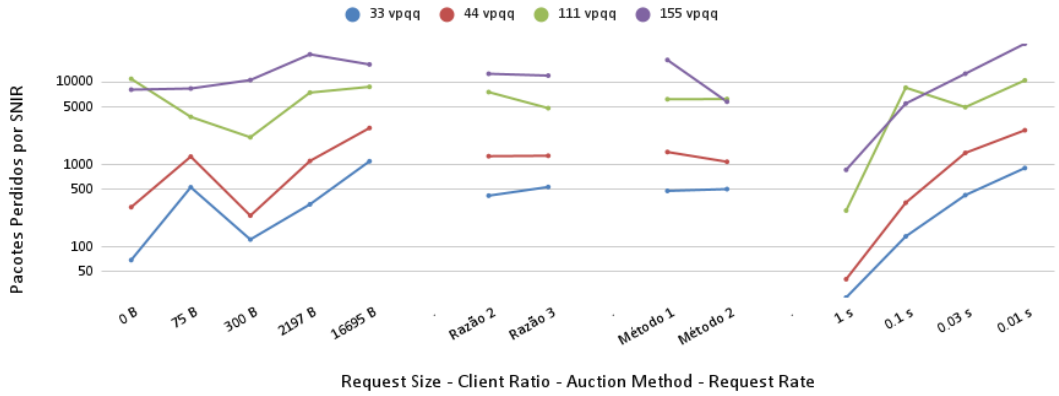
Organizamos os resultados da latência nas Figuras 31, 32, 33 e 34 para demonstrar a influência de cada parâmetro. As Figuras exibem os resultados em uma escala logarítmica devido à grande variabilidade de valores. A Figura 31 exibe os resultados de L-svc em função da densidade. As curvas mostram que um aumento na densidade resulta em um aumento

Figura 39 – Pacotes perdidos por RXTX



Fonte: Elaborada pelo autor (2021)

Figura 40 – Pacotes perdidos por SNIR



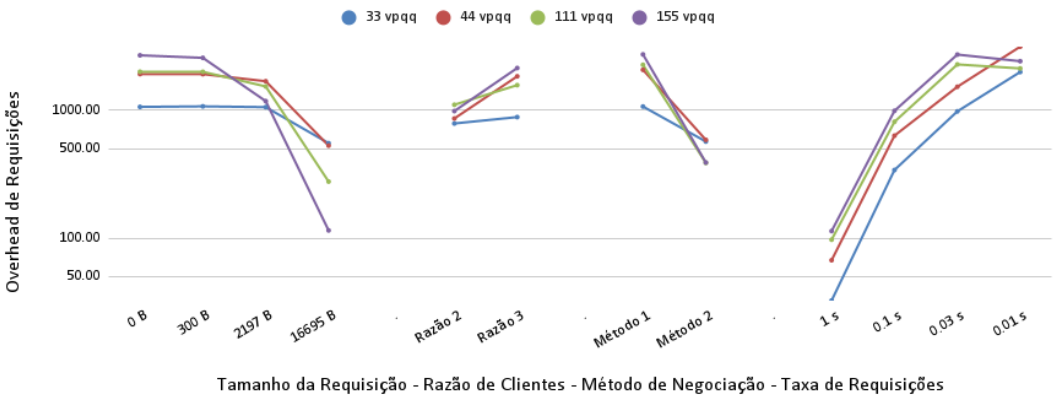
Fonte: Elaborada pelo autor (2021)

exponencial em L-svc. Na densidade mais baixa, L-svc permanece abaixo de 30ms em média.

A taxa de solicitação também tem um forte impacto nessa métrica de acordo com as Figuras 31, 32, 33 e 34. Na taxa de requisições base, 1 RPS, L-svc permanece entre 10 ms e 20 ms. Aumentos significativos começam a ocorrer em taxas acima de 0,1s e densidades acima de 111 VPQQ. A Figura 32 exibe os resultados de L-svc agrupados de acordo com o método de leilão. Não observamos diferenças significativas na forma da curva, indicando que os fatores influenciam igualmente ambos.

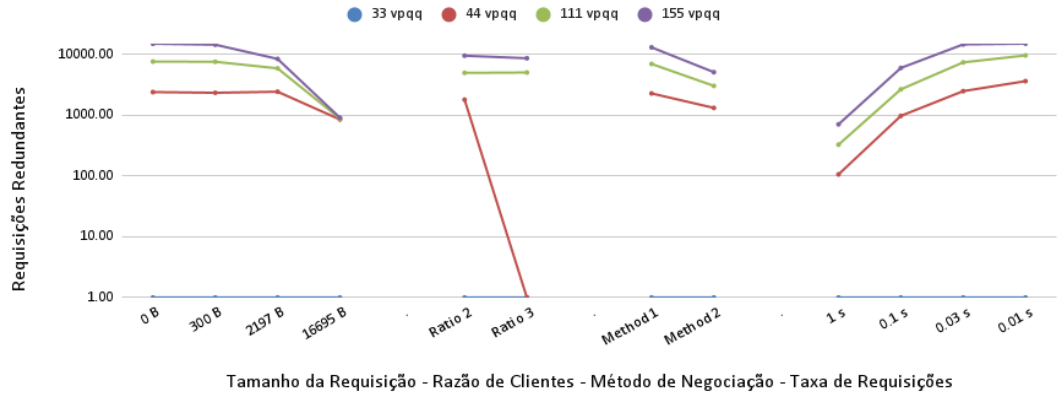
A Figura 33 mostra as discrepâncias mais significativas quando exibimos a L-svc como uma função da taxa de solicitação. As taxas 1 RPS e 10 RPS exibem resultados semelhantes e até mesmo se sobrepõem quando agrupadas de acordo com densidades, método de leilão, proporção de clientes e tamanho das requisições. No entanto, todas as amostras demonstraram diferenças significativas nos testes estatísticos. Na densidade mais baixa, a diferença entre a menor e a maior L-svc médio chega a 395%, e na maior densidade, ultrapassa 2.000%.

Figura 41 – *Overhead* de Requisições



Fonte: Elaborada pelo autor (2021)

Figura 42 – Requisições Redundantes



Fonte: Elaborada pelo autor (2021)

Durante a análise de L-svc, alguns casos se destacaram pela baixa latência, apesar das condições extremas. Em algumas situações, conforme mostrado nas Figuras 32 e 33, o aumento no tamanho da requisição gerou uma redução de L-svc. Argumentamos que isso ocorreu devido ao aumento da quantidade de erros na transmissão de requisições e respostas, pois requisições maiores mantêm a rede ocupada. Além disso, o leilão por notificação tem uma taxa de serviço 5% menor que o leilão por resposta. Acreditamos que isso seja decorrente do maior número de mensagens trocadas durante o leilão, ocasionando queda na taxa de atendimento. Essa avaliação é essencial para destacar que aplicações com altas taxas de *offloading*, como processamento de vídeo em multimídia, precisam ser realizadas em regiões com baixa densidade de veículos ou com alta densidade espectral.

Exibimos os resultados da Taxa De Resposta (T-RES) nas Figuras 35, 36, 37, e 38. A Figura 35 exibe os resultados da T-RES em função da densidade. Observamos que o T-RES diminui com o aumento da densidade. No entanto, a taxa de resposta na menor densidade, 33

VPQQ, T-RES é menor do que na densidade subsequente, 44 VPQQ. Além disso, a proporção entre clientes e servidores afeta fortemente a T-RES pois, na menor proporção, ela tende a aumentar a distância entre os clientes e os servidores, bloqueando assim as respostas. Esses resultados indicam que os clientes podem não encontrar servidores suficientes em densidades mais esparsas. O segundo método de leilão, tratamento de resposta, apresenta maior Taxa De Resposta (TR) entretanto com um maior *overhead* de processamento.

A Figura 36 mostra que o tratamento da resposta da linha de base apresenta alto T-RES, mas ao custo de uma sobrecarga de processamento mais alta. Além disso, a T-RES é inversamente proporcional ao aumento da taxa de requisições, conforme mostrado na Figura 37. A Figura 38 exibe resultados importantes, pois o tamanho da mensagem gera uma elevada taxa de erros.

Duas outras métricas da rede ajudam a avaliar como a densidade afeta os processos de comunicação e o desempenho dos mecanismos de distribuição de tarefas, ou seja, o leilão. As Figuras 39 e 40 mostram os resultados dos erros de rede por RXTX e SNIR. Ambas as métricas têm valores que aumentam com o aumento da densidade e da taxa de requisições.

5.3.3.2 Avaliação do Leilão

Dadas as limitações das mensagens e do canal sem fio, devemos avaliar o sistema de leilão sobre como ele opera sob certas condições de rede. Avaliamos como os aplicativos dependem de leilão. Consideramos que é necessário reduzir a quantidade de mensagens necessárias para negociações para aumentar a taxa de resposta e reduzir a latência total. No melhor do nosso conhecimento, este é o primeiro trabalho de avaliação de um leilão em termos de rede em VFC de baixa latência. Nosso objetivo é identificar os pesos de cada parâmetro nos resultados.

A principal métrica relacionada ao leilão no nível da rede é Overhead De Requisições (O-REQ), no lado do cliente. A Figura 41 apresenta a métrica O-REQ em função dos outros parâmetros. Cada linha representa uma densidade. Os resultados mostram que os valores são diretamente proporcionais à densidade. Além disso, eles são proporcionais ao tamanho da requisição e à proporção entre clientes e servidores. Quando analisamos esses resultados de acordo com a perde por RXTX, na Figura 39, observamos que o tamanho da requisição cria interferência na transmissão de outras requisições ou mesmo respostas. Como os veículos trocam uma quantidade geral menor de mensagens devido a erros, também podemos esperar uma pequena sobrecarga. Além disso, conforme o número de servidores aumenta, há uma

tendência de aumento da sobrecarga de mensagens conforme os servidores precisam negociar sobre o processamento das respostas. Esses resultados corroboram as suposições sobre a tendência de que os veículos devem receber baixas taxas de mensagens em IoV (NETO et al., 2018; CASTELLANO; RISSO; LOTI, 2018), mesmo quando a vazão da rede está no máximo, conforme determinado na Tabela 9, que contém os parâmetros dos experimentos.

Finalmente, a Figura 42 exibe os resultados em O-REQ, que é proporcional à proporção do cliente e à taxa de requisições, mas inversamente proporcional ao tamanho da requisição. Além disso, o número de Requisições De Terceiros (R-TER) aumenta com o leilão. Como argumentamos, um aumento no tamanho da mensagem gera um comportamento anômalo na operação de aplicativos e leilão, pois a rede apresenta erros RXTX e SNIR.

5.3.4 Metodologia de Avaliação: Servidores

As métricas do servidor demonstram o funcionamento de veículos como *Fog Nodes*. Consideramos as seguintes métricas em relação ao servidor e leilão:

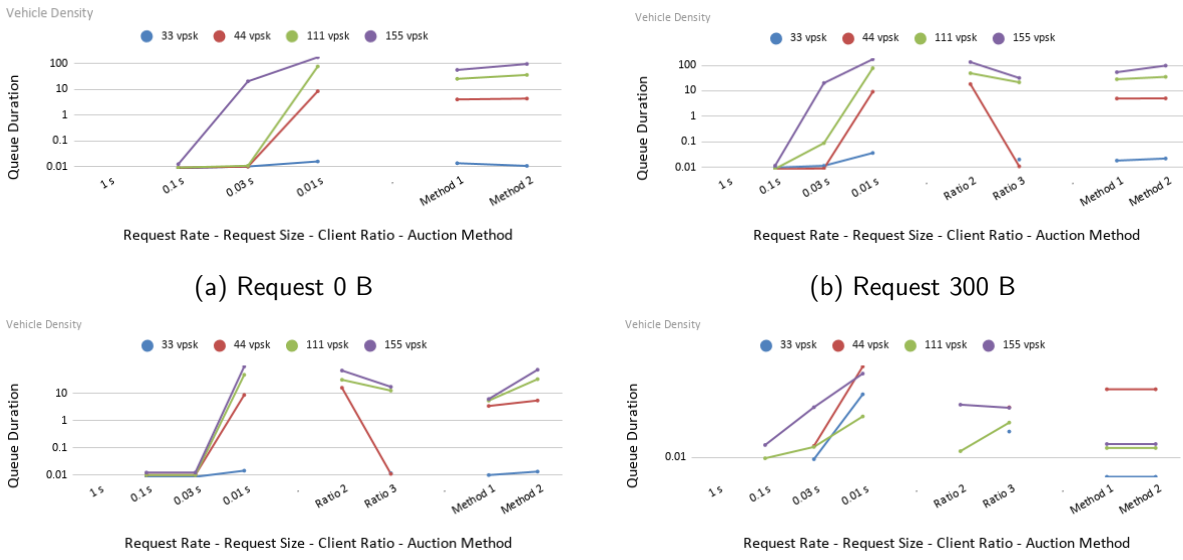
- **Duração da fila** [Servidor]: duração média das filas internas de requisições;
- **Quantidade de Requisições Recebidas** [Servidor]: a quantidade média de requisições recebidas pelos servidores durante a simulação;
- **Quantidade de Requisições Descartados** [Servidor]: requisições descartadas após procedimento de *matching*, ou seja, quando o veículo não contiver a aplicação necessária para realizar o processamento;
- **Quantidade de tarefas aceitas** [Servidor]: tarefas aceitas após procedimento de matching;
- **Quantidade de Tarefas Concluídas** [Servidor]: tarefas finalizadas por um *Fog Node* após o leilão;
- **Quantidade de Tarefas Canceladas** [Leilão]: tarefas canceladas durante o leilão, ou seja, quando o tempo de processamento estimado pelos veículos é maior do que o tempo de processamento estimado no lance;
- **Quantidade de Lances Enviados** [Leilão]: notificações transmitidas pelos servidores quando enviam lances no leilão após estimativa de tempo de processamento;

- **Quantidade de Lances Percebidos** [Leilão]: notificações recebidas pelos servidores durante o leilão;
- **Overhead de Requisições** (RO) [Leilão] : mede respostas duplicadas quando dois ou mais *Fog Nodes* acabam respondendo à mesma solicitação, mesmo após o leilão. Podemos usar essa métrica para avaliar a eficácia do Protocolo de Leilão.

5.3.5 Avaliação dos Servidores

Figura 43 exibe a duração da fila sob diferentes condições. Cada gráfico representa um tamanho de mensagem com os dados agrupados por densidade. Os resultados indicam que a duração da fila aumenta com a densidade e com o método de tratamento de resposta. Em maiores densidades, os servidores recebem mais solicitações, enquanto o método de tratamento de respostas faz com que os servidores mantenham tarefas redundantes por um longo tempo até que o servidor mais rápido responda. Por outro lado, uma proporção menor de clientes reduz a duração da fila. Independentemente do tamanho da requisição, a duração média da fila permanece próxima a 0 (zero) na menor densidade. Quanto maiores os tamanhos das mensagens (2197 e 12695), a duração da fila apresenta uma queda significativa à medida que aumenta o número de erros e diminui o número de solicitações que chegam aos servidores.

Figura 43 – Queue duration, in milliseconds according to the emulated applications



(a) Request 0 B

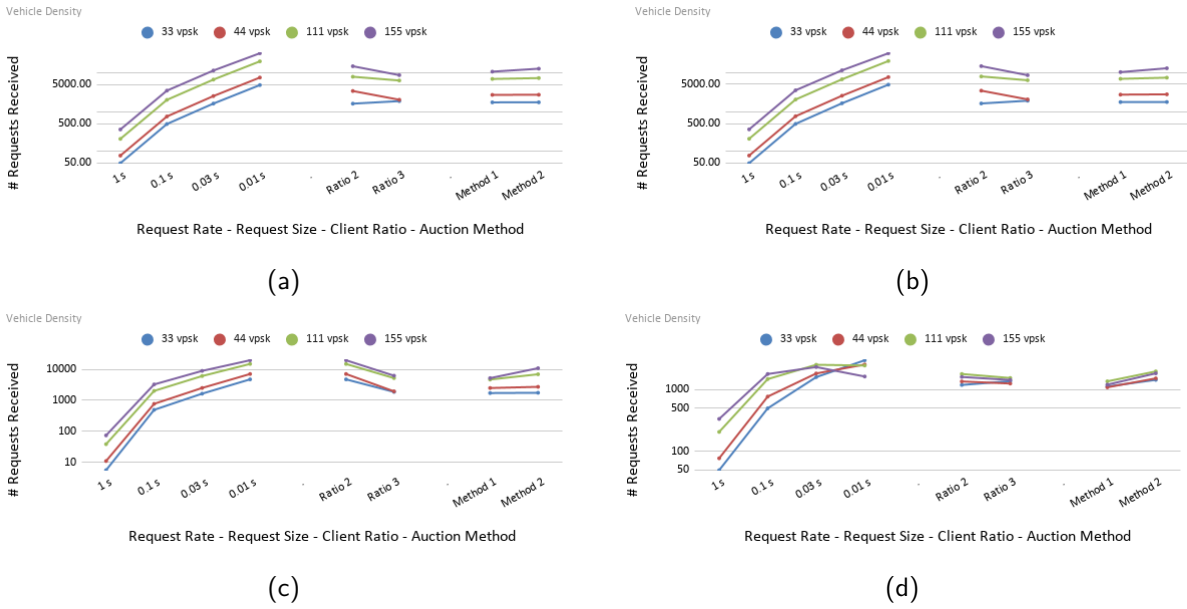
(b) Request 300 B

(c) Request 2197 B

(d) Request 16695 B

Fonte: Elaborada pelo autor (2021)

Figura 44 – Number of requests received



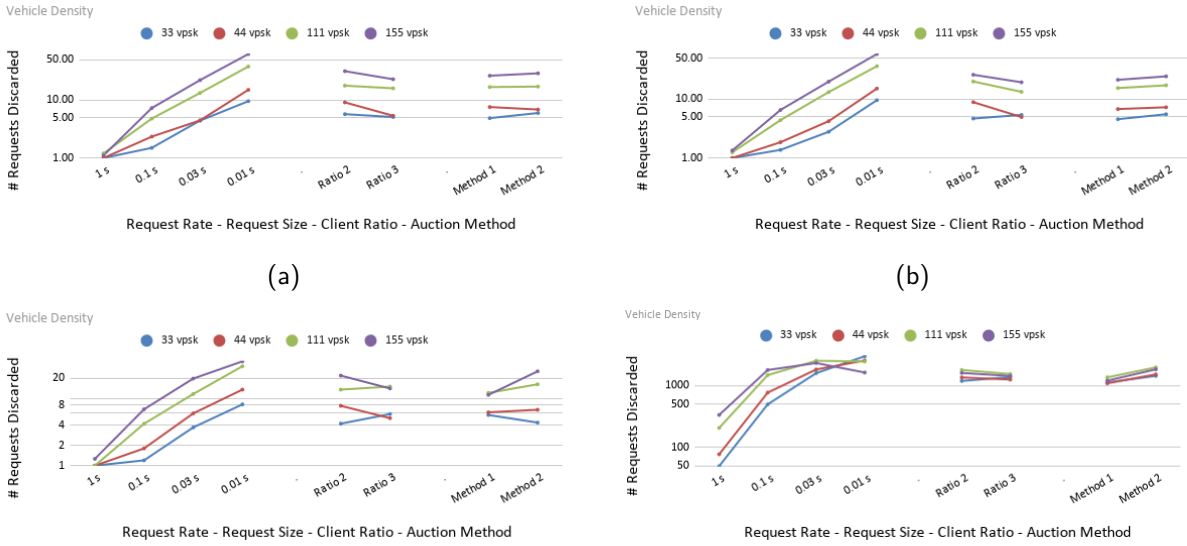
Fonte: Elaborada pelo autor (2021)

A Figura 44 exibe os resultados referentes às requisições recebidas. Essa métrica segue o mesmo padrão da duração da fila, ou seja, os valores aumentam com a densidade e o método; enquanto cai com a proporção de clientes. Além disso, os valores diminuem à medida que o tamanho da mensagem aumenta devido a erros. Ainda assim, a Figura 44d exibe um comportamento anômalo conforme o número de solicitações recebidas nos tamanhos de mensagem maiores e a densidade mais alta cai abaixo das densidades mais baixas. A Figura 45 exibe as solicitações descartadas e apresentam um padrão semelhante.

O número de tarefas aceitas, conforme exibido na Figura 46, apresenta padrões que são diferentes das métricas anteriores, pois a taxa de requisições tem grande influência sobre ela. Na maior taxa de requisições, o número de tarefas aceitas apresenta uma queda significativa, causada por erros de rede. Conforme a proporção de clientes diminui, comete os erros de rede e o número de tarefas aceitas aumenta. Em relação ao método de negociação, o método de tratamento de resposta apresenta uma vantagem nesta métrica quando as mensagens são menores. Conforme o tamanho da mensagem aumenta, o método de leilão faz os servidores aceitarem proporcionalmente mais mensagens.

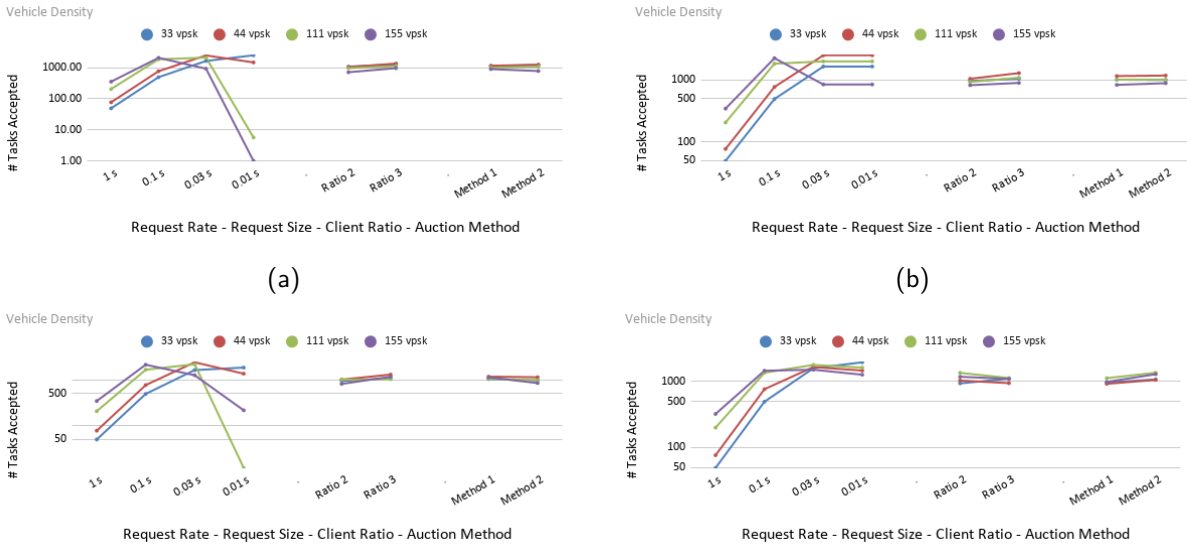
O número de tarefas canceladas, conforme mostrado na Figura 47, é crucial para avaliar os métodos de negociação. Inicialmente, observamos que essa métrica é proporcional ao número de tarefas recebidas. Em densidades mais baixas, o número reduz com menos clientes. Por outro lado, o valor aumenta com o método de tratamento de resposta, indicando níveis mais

Figura 45 – Requests Discarded



Fonte: Elaborada pelo autor (2021)

Figura 46 – Number of tasks accepted

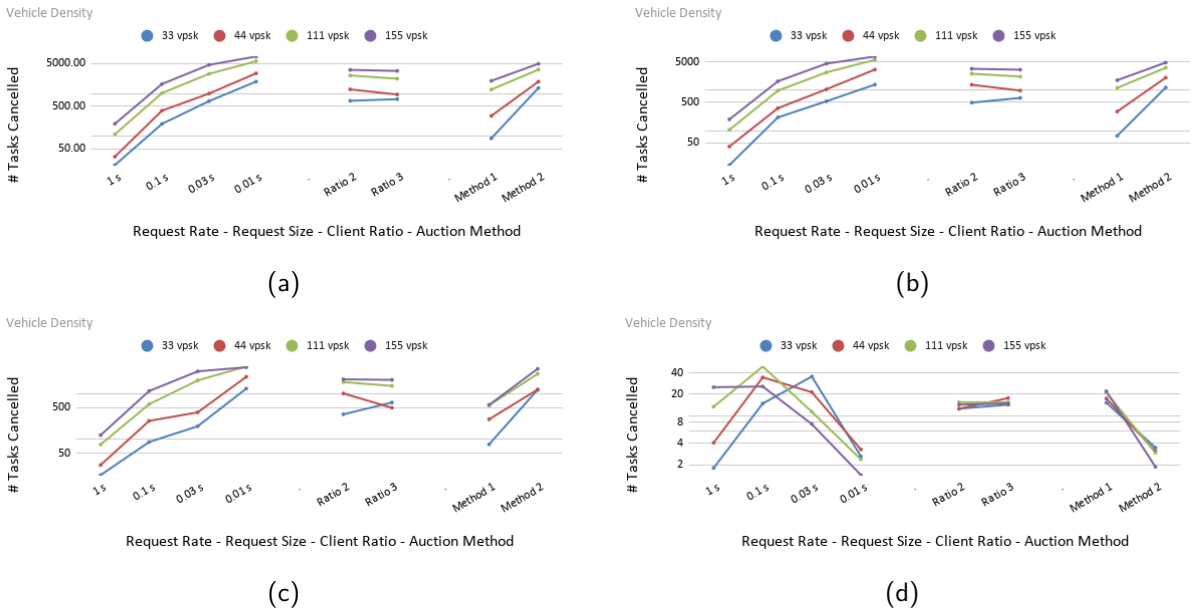


Fonte: Elaborada pelo autor (2021)

baixos de redundância. Essa tendência reverte para o maior tamanho de mensagem, pois os erros de rede reduzem drasticamente o número de solicitações aceitas.

O número de tarefas concluídas, apresentado na Figura 48, indica os resultados após todas as etapas do mecanismo de leilão terem sido concluídas. Assim como algumas das métricas anteriores, o número de tarefas concluídas cai muito com os erros de rede caudado pelo aumento da densidade ou do tamanho das mensagens. Essa métrica exibe resultados

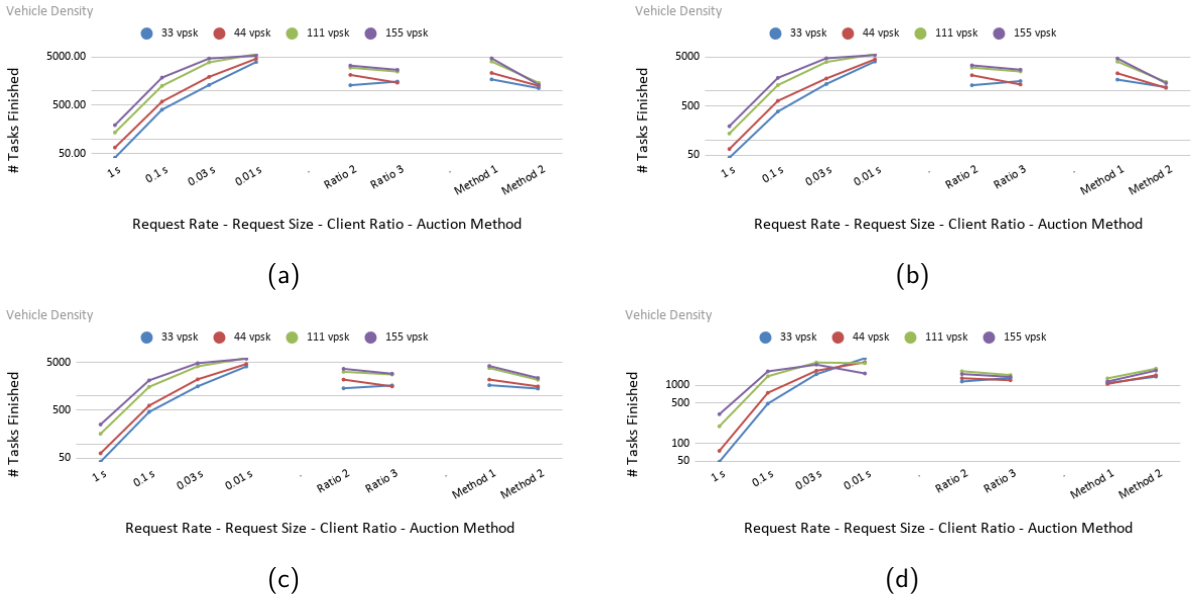
Figura 47 – Number of tasks cancelled



Fonte: Elaborada pelo autor (2021)

extremos, conforme mostrado na Figura 48d. Além disso, no maior tamanho de mensagem, o método de tratamento de resposta supera o leilão. Em tamanhos de solicitação menores, o leilão supera o método de tratamento de resposta.

Figura 48 – Number of tasks finished

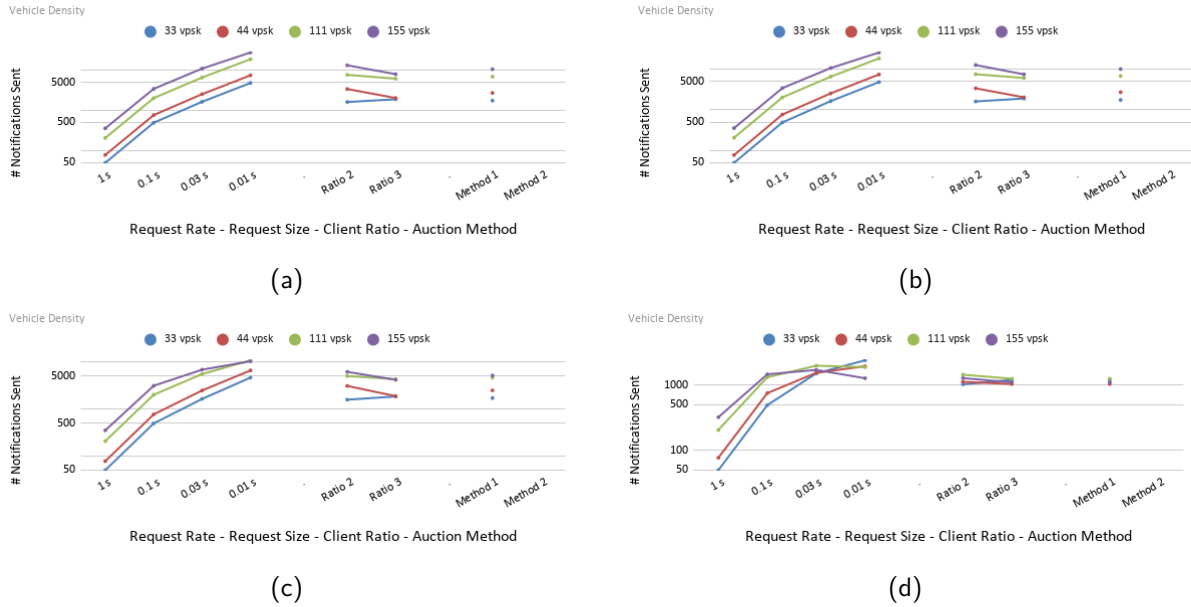


Fonte: Elaborada pelo autor (2021)

Por fim, as Figuras 49 e 50 mostram a quantidade média de notificações (lances) trocadas por veículos durante o leilão. Essa métrica pode representar a sobrecarga de mensagem que o leilão causa numa VFC, pois não conta no método de tratamento de resposta. Nos valores

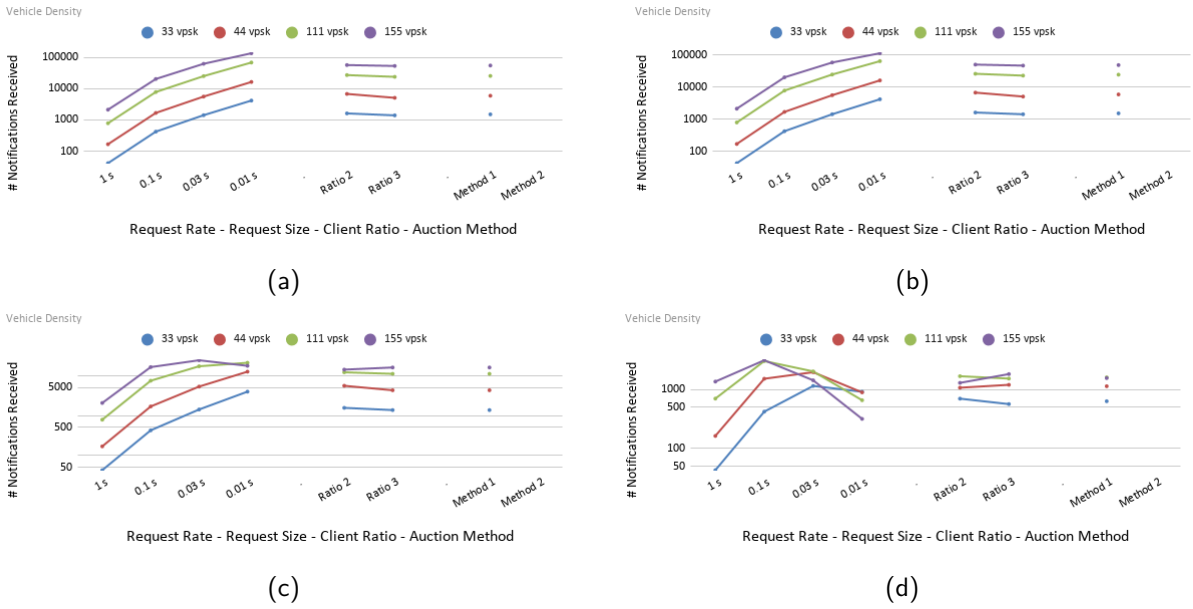
mais baixos dos parâmetros , o *overhead* é de cerca de 42 mensagens, chegando a 135 mil mensagens, em média, na maior taxa e densidade quando o tamanho da mensagem é a linha de base. O número de notificações segue uma tendência crescente junto com a taxa de solicitação e densidade, mas outras métricas sofrem com erros de rede. No tamanho máximo da mensagem, a quantidade de notificações trocadas é a menor. Além disso, a convergência do tamanho da mensagem e da taxa de solicitação causa fenômenos anômalos. Em valores extremos de parâmetros, o número de notificações trocadas fica abaixo de outros valores, como mostrado nas Figuras 50d e 49d.

Figura 49 – Number of Notifications Sent



Fonte: Elaborada pelo autor (2021)

Figura 50 – Number of Notifications Received

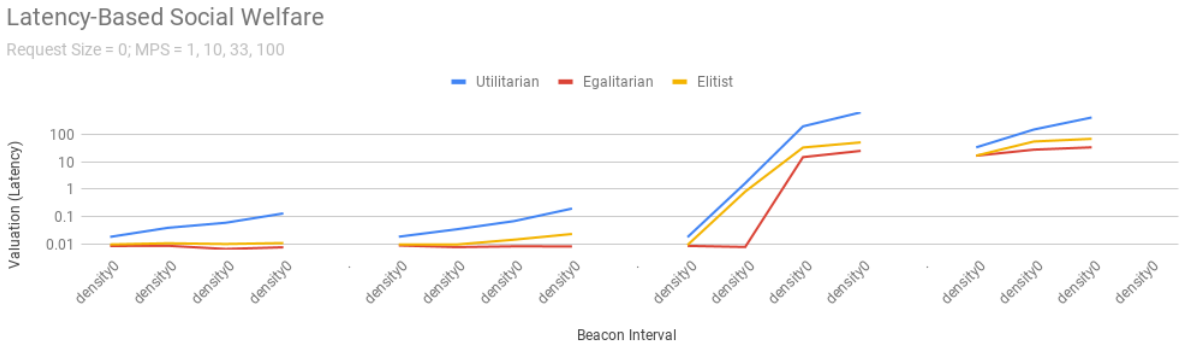


Fonte: Elaborada pelo autor (2021)

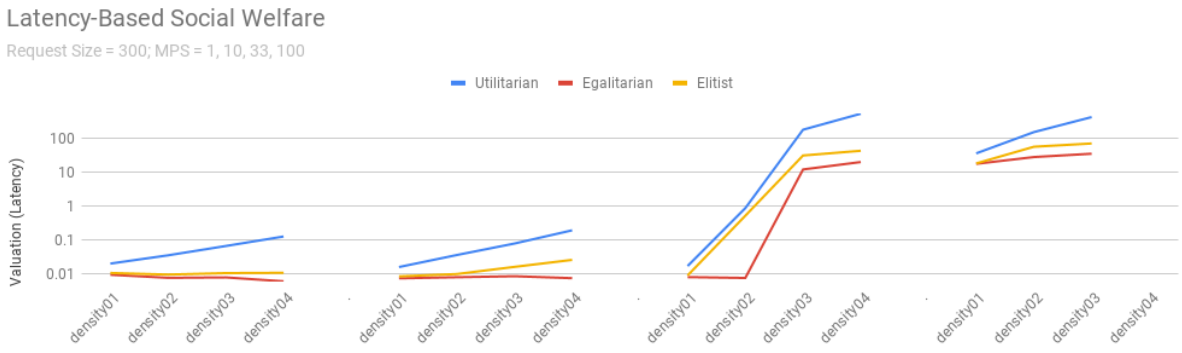
5.4 AVALIAÇÃO DE BEM-ESTAR SOCIAL

Os resultados da avaliação da bem-estar social na Figura 51 corroboram com outros resultados. Quanto menor for a avaliação/custo, melhores serão os leilões. Isso significa que clientes e servidores podem receber/entregar serviços com latência reduzida. Todas as métricas apresentam resultados semelhantes quando agrupadas por parâmetros relevantes: densidade e taxa de mensagem. Ao aumentar a taxa de mensagens, o bem-estar social, ou seja, os custos, aumenta em todos os cenários. Os resultados indicam um ponto de inflexão perceptual em torno de 33 RPS, exceto quando o tamanho da mensagem é 16695 B, cujos outros resultados indicam comportamento anômalo.

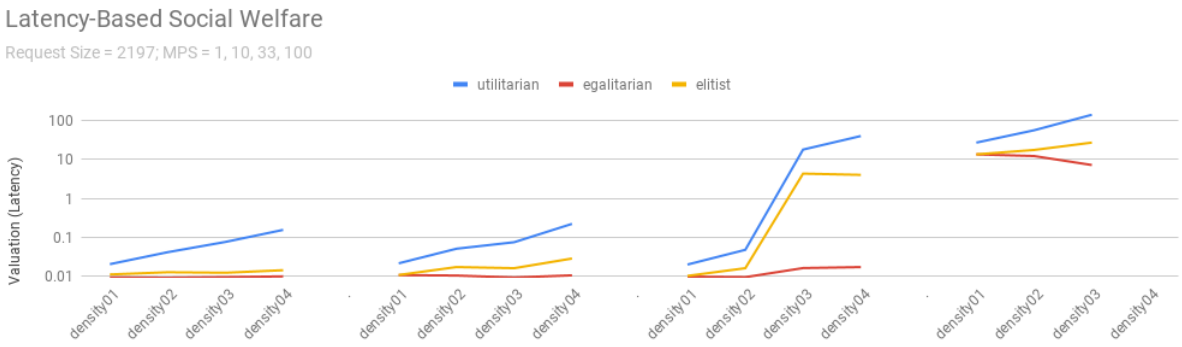
Figura 51 – Utilitarian, Egalitarian and Elitist social welfare.



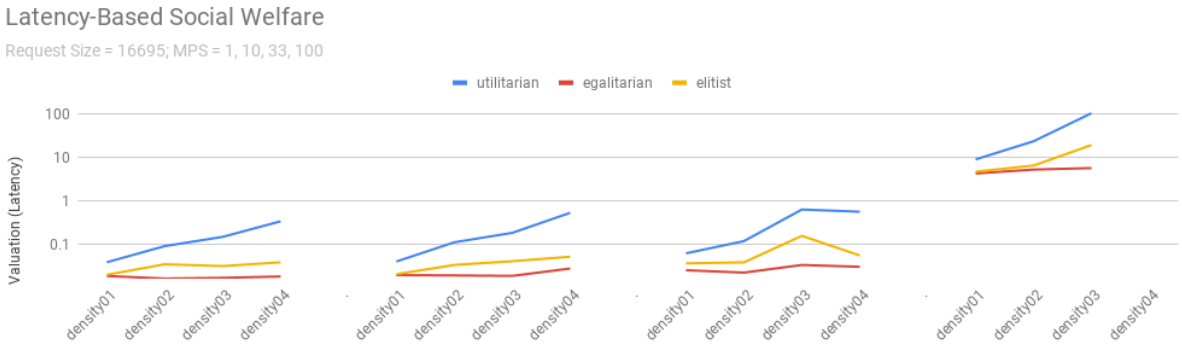
(a) Request 0 B



(b) Request 300 B



(c) Request 2197 B



(d) Request 16695 B

Fonte: Elaborada pelo autor (2021)

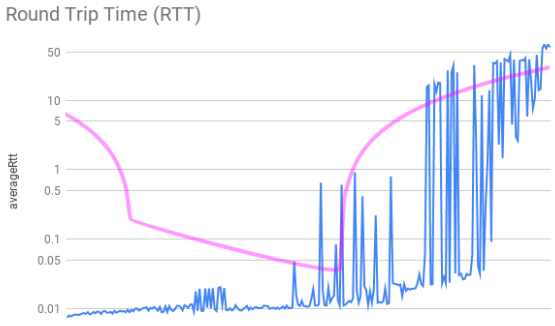
5.5 ANÁLISE DE PARETO PARA DETERMINAÇÃO DE MELHORES CENÁRIOS

Dadas a quantidade de parâmetros e a quantidade de simulações, utilizamos um mecanismo de Fronteiras de Pareto (LINDEN, 2008) para resolver possíveis *trade-offs* entre soluções e determinar quais são aquelas que realmente conseguem se manter dentro dos requisitos de VFC. Para isso, utilizamos duas métricas principais do lado do cliente (latência de serviços e taxa de atendimento) para determinar as melhores soluções. Cada solução é identificada por uma tupla indicando os parâmetros daquele cenário. Por exemplo, o cenário (density00, 1, 1, 2, 300) indica a densidade 33 VPQQ, 1 RPS, método de negociação por leilão usando notificações, razão de clientes 1/2, e requisição de 300 B.

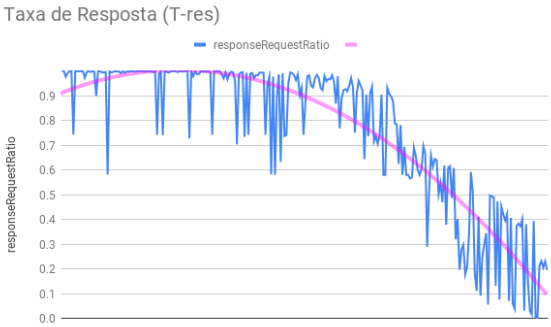
As imagens da Figura 52 classificam as simulações de acordo com os resultados da tabela do Apêndice A. As linhas de tendência polinomial ajudam na visualização. As figuras 52a e 52b mostram que as melhores soluções possuem baixa latência e alta taxa de atendimento de requisições. Em termos dos identificadores de simulação, apresentados no Apêndice A, tratam-se, prioritariamente, dos cenários com baixa densidade ou baixa taxa de geração de mensagens. Entre elas, aparecem alguns casos com valores divergentes entre as métricas como, por exemplo, densidade mais alta e taxa de mensagens mais baixa.

As imagens da Figura 52 além da tabela no Apêndice A podem servir de base para a comparação com outros trabalhos que surjam futuramente.

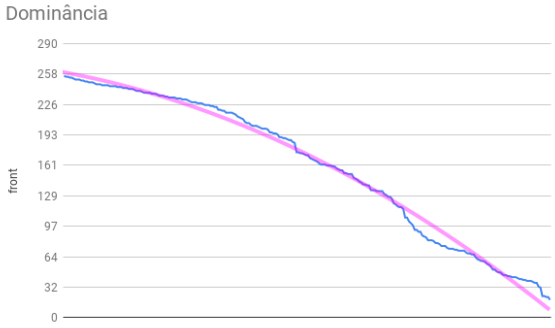
Figura 52 – Relações de pareto



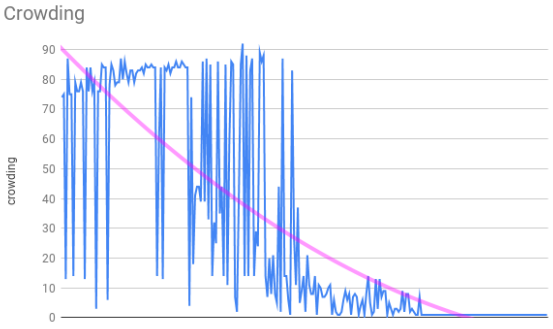
(a) Medida de Pareto para L-svc (apresentado em escala logarítimica para melhor visualiação)



(b) Medida de Pareto para T-res



(c) Medida de dominância



(d) Medida de Crowding de Pareto usada para desempate

Fonte: Elaborada pelo autor (2021)

5.6 CONSIDERAÇÕES FINAIS

Os resultados apontam que dois parâmetros são especialmente importantes nesse contexto: a taxa de requisições, que pode ajudar na visão interna do dispositivo *Fog* em relação ao ambiente; e a densidade de veículos que impactam diretamente nos erros de rede. Esse último resultado, especialmente, aponta uma necessidade de investigação a variação na quantidade de aplicações IoT e ITS que podem requerer serviços no futuro, problema esse que ainda não foi investigado com detalhes na literatura especializada.

Além disso, os resultados indicam que o método de leilão, que pode ser um candidato para o gerenciamento de recursos em VFC, funciona bem apenas sob condições restritas de densidade e rede. Por outro lado, um método alternativo de leilão, o tratamento de respostas, pode funcionar melhor em redes mais densas, especialmente devido ao alto *overhead* de mensagens. Isso aponta que é necessário integrar mecanismos de leilão com outros mecanismos de gerenciamento de transmissão de pacotes como, por exemplo, WSDN.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A emergência do 5G depende da adoção e da integração de uma série de tecnologias. A IoV se utilizará dessa sinergia, especialmente do controle e automação de rede, para se consolidar, mas, ao mesmo tempo, proverá recursos para que todos os princípios de *design* sejam contemplados, como um representante da computação *Fog*. Acreditamos que a computação Fog é um dos pilares do 5G por ser aglutinadora de todas as outras tecnologias, além de conectar os princípios de *design* e atender aos requisitos das aplicações que precisam de baixa latência de serviço.

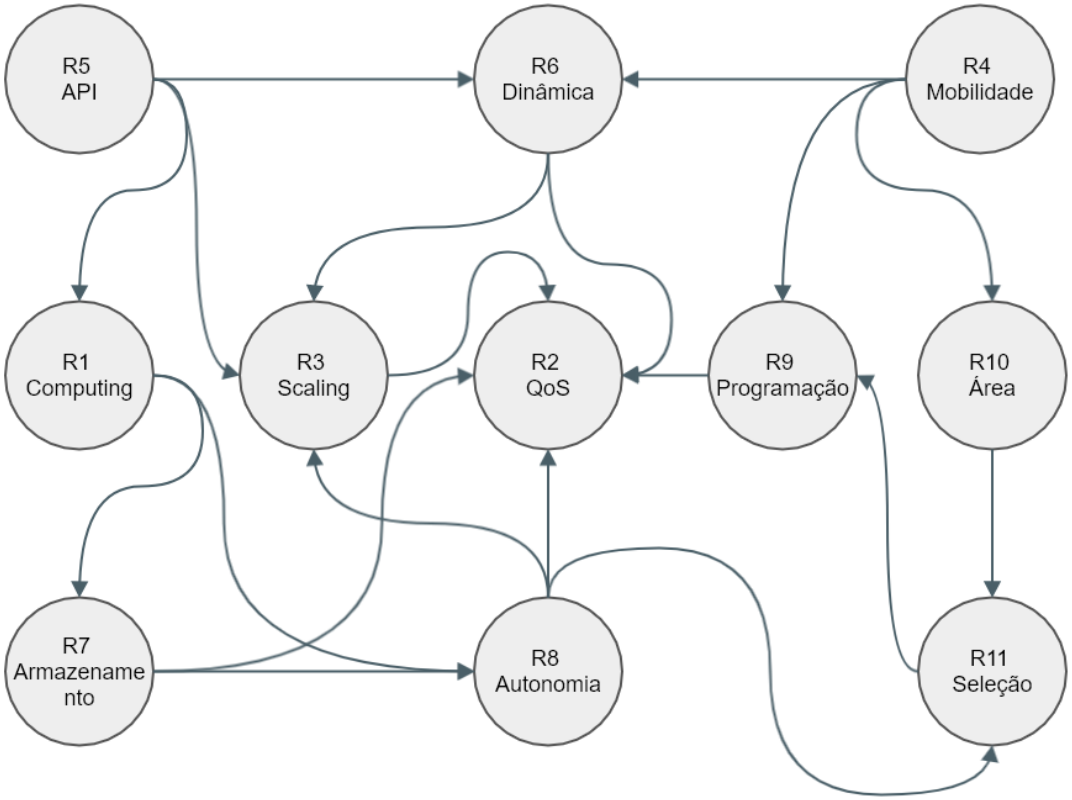
Dispositivos *Fog* podem prover serviços, dados, processamento e infraestrutura uns para os outros. Ao mesmo tempo, eles podem depender de tecnologias de controle, como SDN e virtualização, como NFV, para a formação das redes complexas e autonômicas das quais dependem para funcionar. Mesmo assim, como previsto na teoria da computação na borda, eles podem fornecer serviços, dados, processamento e infraestrutura para as próprias redes complexas e autonômicas das quais dependem. Computação *Fog* se destina a atender requisitos estritos, ao mesmo tempo em que provê recursos para que tais requisitos sejam atingidos.

FogWise atende a todos os princípios de *design* propostos para computação *Fog* através de algoritmos e técnicas específicos ou complementares. No estado atual, o modelo FogWise está atendendo, pelo menos parcialmente, aos princípios de:

- participação na cadeia de processamento (R1), com o uso de veículos, o que dificilmente é feito na literatura;
- atendimento aos requisitos de QoS e QoE (R2), com avaliações que demonstram a possibilidade de alcançar 50ms de latência de serviços;
- existe um nível de escalabilidade (R3) em termos, especialmente, de densidade e taxa de mensagens no qual leilões podem ser o algoritmo principal de alocações de recursos em VFC;
- mobilidade (R4), através da possibilidade de modelar velocidade de veículos e interação com dispositivos IoT do ambiente, além da observação do comportamento dos veículos num ambiente de micro-simulação;
- veículos podem ter uma API (R5) definida que funcione através de leilão de tarefas de execução que funciona caso mais de um servidor esteja disponível;

- o leilão é realizado em uma rede totalmente distribuída sem auxílio de Cloud ou Edge, mesmo apresentando certos limites de funcionamento (R6);
- sistemas clientes não têm necessidade de realizar qualquer processamento sob condições específicas, realizando *offloading* completamente em Fog Nodes servidores (R7);
- Fog Nodes e clientes atuam como agentes inteligentes e possuem toda a lógica interna para realizar seus processamentos através de um leilão (R8);
- é possível realizar vários processamento no tempo de contato entre clientes e servidores no intervalo de poucos milissegundos, o que é suficiente para executar em contatos muito curtos (R9);
- relação aplicação-área (R10), pois os veículos trocam informações dentro da áreas sobrepostas onde acontece o leilão;
- é possível realizar leilões para distribuição de tarefas de baixa latência em VFC (R11).

Figura 53 – Relação entre os princípios de *design* para a consolidação de FogWise



Fonte: Elaborada pelo autor (2021)

6.1 LIMITAÇÕES E TRABALHOS FUTUROS

Mesmo tendo fornecido uma série de contribuições para o campo de pesquisa, esta tese ainda tem algumas limitações. Os trabalhos futuros, assim como outras contribuições em relação à conclusão desta tese, estão intimamente ligados ao aperfeiçoamento do mecanismo de leilão, ou mesmo à integração com outros tecnologias para que ele possa funcionar em cenários que estão além dos limites atualmente demonstrados.

Atualmente, já estamos trabalhando em modelos mais elaborados que atendam a cada vez mais requisitos que reforcem a participação livre de estratégia do *Mechanism Design*, como o leilão com lance fechado e que premie o melhor lance mas cobrando o segundo melhor valor. Está sendo avaliado que estratégias como essa podem melhorar o bem-estar social numa VFC ao custo de uma latência de serviço um pouco maior Assim, pode ser preciso avaliar quais são as aplicações ou contextos que podem se beneficiar de tais melhorias.

Este trabalho analisa e avalia a apenas o funcionamento e os limites de uma rede VFC com-

pletamente distribuída. Observamos que existe um limite muito baixo para que aplicações com alta demanda de processamento e requisito de baixa latência possam funcionar corretamente. Assim, é preciso modelar e avaliar a interação *Cloud-edge-fog* para que o processamento excedente possa ser realizado dentro dos requisitos para todos os clientes. Inicialmente, isso passa pela extensão do modelo atual, que apenas assume a participação dos dispositivos finais na rede. Isso pode ser realizado através da implantação do modelo nos simuladores Omnet++ e Mininet-Wifi.

Além disso, ainda é possível realizar otimizações no modelo completamente distribuído através da integração de novas tecnologias e algoritmos que possam minimizar os erros de rede. Tal otimização pode expandir os limites nos quais o modelo de leilão pode funcionar em VFC. Tecnologias como WSDN podem ser utilizadas nesse contexto.

6.2 PUBLICAÇÕES

Este trabalho possui pelo menos uma publicação em revista contendo seus principais resultados:

- (JUNIOR et al., 2021) DE MENDONÇA JUNIOR, Francisco Ferreira et al. FogWise: On the limits of the coexistence of heterogeneous applications on Fog computing and Internet of Vehicles. Transactions on Emerging Telecommunications Technologies, v. 32, n. 1, p. e4145, 2021.

REFERÊNCIAS

AGIWAL, M.; ROY, A.; SAXENA, N. Next generation 5g wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 18, n. 3, p. 1617–1655, 2016.

AKYILDIZ, I. F.; NIE, S.; LIN, S.-C.; CHANDRASEKARAN, M. 5g roadmap: 10 key enabling technologies. *Computer Networks*, Elsevier, v. 106, p. 17–48, 2016.

AL-SULTAN, S.; AL-DOORI, M. M.; AL-BAYATTI, A. H.; ZEDAN, H. A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, Elsevier, v. 37, p. 380–392, 2014.

ALAM, M. G. R.; TUN, Y. K.; HONG, C. S. Multi-agent and reinforcement learning based code offloading in mobile fog. In: IEEE. *Information Networking (ICOIN), 2016 International Conference on*. [S.l.], 2016. p. 285–290.

ALTAYEB, M.; MAHGOUB, I. A survey of vehicular ad hoc networks routing protocols. *International Journal of Innovation and Applied Studies*, Citeseer, v. 3, n. 3, p. 829–846, 2013.

AMADEO, M.; CAMPOLO, C.; MOLINARO, A. Information-centric networking for connected vehicles: a survey and future perspectives. *IEEE Communications Magazine*, IEEE, v. 54, n. 2, p. 98–104, 2016.

AMENDOLA, D.; CORDESCHI, N.; BACCARELLI, E. Bandwidth management vms live migration in wireless fog computing for 5g networks. In: IEEE. *Cloud Networking (Cloudnet), 2016 5th IEEE International Conference on*. [S.l.], 2016. p. 21–26.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.

BAKTIR, A. C.; OZGOVDE, A.; ERSOY, C. How can edge computing benefit from software-defined networking: a survey, use cases, and future directions. *IEEE Communications Surveys & Tutorials*, IEEE, v. 19, n. 4, p. 2359–2391, 2017.

BANSAL, N.; SVIRIDENKO, M. The santa claus problem. In: *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: Association for Computing Machinery, 2006. (STOC '06), p. 31–40. ISBN 1595931341. Disponível em: <https://doi.org/10.1145/1132516.1132522>.

BARBOSA, F. E. da S.; JúNIOR, F. F. de M.; DIAS, K. L. A platform for cloudification of network and applications in the internet of vehicles. *Transactions on Emerging Telecommunications Technologies*, v. 31, n. 5, p. e3961, 2020. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3961>.

BARIK, R. K.; PATRA, S. S.; KUMARI, P.; MOHANTY, S. N.; HAMAD, A. A. A new energy aware task consolidation scheme for geospatial big data application in mist computing environment. In: *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*. [S.l.: s.n.], 2021. p. 48–52.

BENAMAR, N.; SINGH, K. D.; BENAMAR, M.; OUADGHIRI, D. E.; BONNIN, J.-M. Routing protocols in vehicular delay tolerant networks: A comprehensive survey. *Computer Communications*, Elsevier, v. 48, p. 141–158, 2014.

BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. Fog computing and its role in the internet of things. In: ACM. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.], 2012. p. 13–16.

BOUKERCHE, A.; ROBSON, E. Vehicular cloud computing: Architectures, applications, and mobility. *Computer Networks*, Elsevier, 2018.

BRANDT, F.; CONITZER, V.; ENDRISS, U. Computational social choice. *Multiagent systems*, p. 213–283.

BROGI, A.; MENCAGLI, G.; NERI, D.; SOLDANI, J.; TORQUATI, M. Container-based support for autonomic data stream processing through the fog. In: SPRINGER. *European Conference on Parallel Processing*. [S.l.], 2017. p. 17–28.

BURGESS, J.; GALLAGHER, B.; JENSEN, D. D.; LEVINE, B. N. et al. Maxprop: Routing for vehicle-based disruption-tolerant networks. In: BARCELONA, SPAIN. *Infocom*. [S.l.], 2006. v. 6.

Cai, W.; Leung, V. C. M.; Chen, M. Next generation mobile cloud gaming. In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. [S.l.: s.n.], 2013. p. 551–560.

CASTELLANO, G.; RISSO, F.; LOTI, R. Fog computing over challenged networks: A real case evaluation. In: IEEE. *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. [S.l.], 2018. p. 1–7.

CERWALL, P. et al. Ericsson mobility report november 2020. *Ericsson.com*, 2020.

CHAHAL, M.; HARIT, S.; MISHRA, K. K.; SANGAIAH, A. K.; ZHENG, Z. A survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases. *Sustainable Cities and Society*, Elsevier, 2017.

CHÁVEZ-SANTIAGO, R.; SZYDEŁKO, M.; KLIKS, A.; FOUKALAS, F.; HADDAD, Y.; NOLAN, K. E.; KELLY, M. Y.; MASONTA, M. T.; BALASINGHAM, I. 5g: The convergence of wireless communications. *Wireless Personal Communications*, Springer, v. 83, n. 3, p. 1617–1642, 2015.

CHEN, H.; JOSHI, A.; FININ, T. Dynamic service discovery for mobile computing: Intelligent agents meet jini in the aether. *Cluster Computing*, Springer, v. 4, n. 4, p. 343–354, 2001.

CHIOSI, M.; CLARKE, D.; WILLIS, P.; REID, A.; FEGER, J.; BUGENHAGEN, M.; KHAN, W.; FARGANO, M.; CUI, C.; DENG, H. et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In: SN. *SDN and OpenFlow World Congress*. [S.l.], 2012. v. 48.

d'Orey, P. M.; Hosseini, A.; Azevedo, J.; Diermeyer, F.; Ferreira, M.; Lienkamp, M. Hail-a-Drone: Enabling teleoperated taxi fleets. In: *IEEE Intelligent Vehicles Symposium (IV)*. [S.l.: s.n.], 2016. p. 774–781.

DUPONT, C.; GIAFFREDA, R.; CAPRA, L. Edge computing in iot context: horizontal and vertical linux container migration. *Global Internet of Things Summit (GIoTS). IEEE*, 2017.

ETSI. *Intelligent Transport Systems (ITS). V2X Applications. Part 2: Intersection Collision Risk Warning (ICRW) Application requirements specification*. 2018. Disponível em: <https://www.etsi.org/deliver/etsi_ts/101500_101599/10153902/01.01.01_60/ts_10153902v010101p.pdf>.

EZE, E. C.; ZHANG, S.; LIU, E. Vehicular ad hoc networks (vanets): Current state, challenges, potentials and way forward. In: IEEE. *Automation and Computing (ICAC), 2014 20th International Conference on*. [S.l.], 2014. p. 176–181.

FENG, L.; LI, W.; LIN, Y.; ZHU, L.; GUO, S.; ZHEN, Z. Joint computation offloading and urllc resource allocation for collaborative mec assisted cellular-v2x networks. *IEEE Access*, v. 8, p. 24914–24926, 2020.

GIANG, N. K.; LEUNG, V. C.; LEA, R. On developing smart transportation applications in fog computing paradigm. In: *Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*. New York, NY, USA: Association for Computing Machinery, 2016. (DIVANet '16), p. 91–98. ISBN 9781450345064. Disponível em: <https://doi.org/10.1145/2989275.2989286>.

GUPTA, H.; DASTJERDI, A. V.; GHOSH, S. K.; BUYYA, R. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, Wiley Online Library, v. 47, n. 9, p. 1275–1296, 2017.

HAO, Z.; NOVAK, E.; YI, S.; LI, Q. Challenges and software architecture for fog computing. *IEEE Internet Computing*, IEEE, v. 21, n. 2, p. 44–53, 2017.

HAQUE, I. T.; ABU-GHAZALEH, N. Wireless software defined networking: A survey and taxonomy. *IEEE Communications Surveys & Tutorials*, IEEE, v. 18, n. 4, p. 2713–2737, 2016.

HONG, C.-H.; VARGHESE, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. Association for Computing Machinery, New York, NY, USA, v. 52, n. 5, set. 2019. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3326066>.

HONG, K.; LILLETHUN, D.; RAMACHANDRAN, U.; OTTENWÄLDER, B.; KOLDEHOFE, B. Mobile fog: A programming model for large-scale applications on the internet of things. In: ACM. *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. [S.l.], 2013. p. 15–20.

HOU, X.; LI, Y.; CHEN, M.; WU, D.; JIN, D.; CHEN, S. Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Transactions on Vehicular Technology*, IEEE, v. 65, n. 6, p. 3860–3873, 2016.

HUANG, J.; QIAN, F.; GERBER, A.; MAO, Z. M.; SEN, S.; SPATSCHECK, O. A close examination of performance and power characteristics of 4g lte networks. In: ACM. *Proceedings of the 10th international conference on Mobile systems, applications, and services*. [S.l.], 2012. p. 225–238.

IRAJI, S.; MOGENSEN, P.; RATASUK, R. Recent advances in m2m communications and internet of things (iot). *International Journal of Wireless Information Networks*, Springer, v. 24, n. 3, p. 240–242, 2017.

JAIN, R. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. [S.l.]: John Wiley & Sons, 1990.

JAIN, R.; PAUL, S. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, v. 51, n. 11, p. 24–31, November 2013. ISSN 0163-6804.

JI, X.; YU, H.; FAN, G.; FU, W. Sdgr: An sdn-based geographic routing protocol for vanet. In: IEEE. *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference on*. [S.l.], 2016. p. 276–281.

JUNIOR, F. F. de M.; DIAS, K. L.; D'OREY, P. M.; KOKKINOGENIS, Z. Fogwise: On the limits of the coexistence of heterogeneous applications on fog computing and internet of vehicles. *Transactions on Emerging Telecommunications Technologies*, v. 32, n. 1, p. e4145, 2021. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4145>.

KAUR, K.; GARG, S.; KADDOUM, G.; GAGNON, F.; AHMED, S. H. Blockchain-based lightweight authentication mechanism for vehicular fog infrastructure. In: IEEE. *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. [S.l.], 2019. p. 1–6.

Kayal, P.; Liebeherr, J. Distributed service placement in fog computing: An iterative combinatorial auction approach. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2019. p. 2145–2156.

Kenney, J. B. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, v. 99, n. 7, p. 1162–1182, July 2011. ISSN 1558-2256.

KLAIMI, J.; SENOUCI, S.-M.; MESSOUS, M.-A. Theoretical game approach for mobile users resource management in a vehicular fog computing environment. In: IEEE. *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. [S.l.], 2018. p. 452–457.

KU, I.; LU, Y.; GERLA, M.; GOMES, R. L.; ONGARO, F.; CERQUEIRA, E. Towards software-defined vanet: Architecture and services. In: IEEE. *2014 13th annual Mediterranean ad hoc networking workshop (MED-HOC-NET)*. [S.l.], 2014. p. 103–110.

KUMAR, K.; LIU, J.; LU, Y.-H.; BHARGAVA, B. A survey of computation offloading for mobile systems. *Mobile networks and Applications*, Springer, v. 18, n. 1, p. 129–140, 2013.

KUROSE, J. F. *Computer networking: A top-down approach featuring the internet, 3/E*. [S.l.]: Pearson Education India, 2005.

LEE, E.; LEE, E.-K.; GERLA, M.; OH, S. Y. Vehicular cloud networking: architecture and design principles. *IEEE Communications Magazine*, IEEE, v. 52, n. 2, p. 148–155, 2014.

Lin, F.; Zhou, Y.; Pau, G.; Collotta, M. Optimization-oriented resource allocation management for vehicular fog computing. *IEEE Access*, v. 6, p. 69294–69303, 2018.

LINDEN, R. *Algoritmos genéticos (2a ediçao)*. [S.l.]: Brasport, 2008.

LIU, L.; CHEN, C.; PEI, Q.; MAHARJAN, S.; ZHANG, Y. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, Springer, p. 1–24, 2020.

Lopez, P. A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; Wiessner, E. Microscopic traffic simulation using sumo. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. [S.l.: s.n.], 2018. p. 2575–2582.

LOVECRAFT, H. P. *O Horror Sobrenatural em Literatura*. [S.l.]: Dover Publications, Inc., 1987. 10 p. Tradução de João Guilherme Linke.

LUJAK, M.; SLAVKOVIK, M. A modified vickrey auction with regret minimization for uniform alliance decisions. In: SPRINGER. *International Symposium on Intelligent and Distributed Computing*. [S.l.], 2017. p. 61–72.

MACAL, C. M.; NORTH, M. J. Tutorial on agent-based modeling and simulation. In: IEEE. *Proceedings of the Winter Simulation Conference, 2005.* [S.l.], 2005. p. 14–pp.

MAI, A.; SCHLESINGER, D. Connected vehicles: From building cars to selling, personal travel time well-spent. *Cisco Internet Business Solutions Group (IBSG)*, 2010.

MAI, A.; SCHLESINGER, D. Connected vehicles: Service providers at a crossroads. *Cisco Internet Business Solutions Group, April*, 2011.

MAO, Y.; ZHANG, J.; LETAIEF, K. B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 34, n. 12, p. 3590–3605, 2016.

MELL, P.; GRANCE, T. et al. *The NIST definition of cloud computing*. [S.l.], 2011.

MOURADIAN, C.; NABOULSI, D.; YANGUI, S.; GLITHO, R. H.; MORROW, M. J.; POLAKOS, P. A. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, IEEE, v. 20, n. 1, p. 416–464, 2017.

Mukherjee, M.; Shu, L.; Wang, D. Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Communications Surveys Tutorials*, v. 20, n. 3, p. 1826–1857, 2018.

NARENDRA, P.; FUKUNAGA, K. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, IEEE Computer Society Washington, DC, USA, v. 26, n. 9, p. 917–922, 1977.

NETO, A. J.; ZHAO, Z.; RODRIGUES, J. J.; CAMBOIM, H. B.; BRAUN, T. Fog-based crime-assistance in smart iot transportation system. *IEEE Access*, IEEE, v. 6, p. 11101–11111, 2018.

NING, Z.; HUANG, J.; WANG, X. Vehicular fog computing: Enabling real-time traffic management for smart cities. *IEEE Wireless Communications*, IEEE, v. 26, n. 1, p. 87–93, 2019.

OUYANG, T.; ZHOU, Z.; CHEN, X. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *arXiv preprint arXiv:1809.05239*, 2018.

Peng, X.; Ota, K.; Dong, M. Multiattribute-based double auction toward resource allocation in vehicular fog computing. *IEEE Internet of Things Journal*, v. 7, n. 4, p. 3094–3103, 2020.

PEREIRA, J.; RICARDO, L.; LUÍS, M.; SENNA, C.; SARGENTO, S. Assessing the reliability of fog computing for smart mobility applications in vanets. *Future Generation Computer Systems*, Elsevier, v. 94, p. 317–332, 2019.

RANGANATHAN, A.; CAMPBELL, R. H. What is the complexity of a distributed computing system? *Complexity*, v. 12, n. 6, p. 37–45, 2007. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cplx.20189>.

Rivoirard, L.; Wahl, M.; Sondi, P.; Berbineau, M.; Gruyer, D. Performance evaluation of aodv, dsr, grp and olsr for vanet with real-world trajectories. In: *2017 15th International Conference on ITS Telecommunications (ITST)*. [S.l.: s.n.], 2017. p. 1–7.

RODRIGUES, T. G.; SUTO, K.; NISHIYAMA, H.; KATO, N. Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control. *IEEE Transactions on Computers*, IEEE, v. 66, n. 5, p. 810–819, 2017.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Malaysia; Pearson Education Limited,, 2016.

Saraiva de Sousa, N. F.; Lachos Perez, D. A.; ROSA, R. V.; SANTOS, M. A.; Esteve Rothenberg, C. Network service orchestration: A survey. *Computer Communications*, v. 142-143, p. 69–94, 2019. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366418309502>.

SARKAR, S.; MISRA, S. Theoretical modelling of fog computing: a green computing paradigm to support iot applications. *Iet Networks*, IET, v. 5, n. 2, p. 23–29, 2016.

SATOH, I. A framework for data processing at the edges of networks. In: SPRINGER. *International Conference on Database and Expert Systems Applications*. [S.l.], 2013. p. 304–318.

SCHOETTLE, B.; SIVAK, M. A survey of public opinion about autonomous and self-driving vehicles in the us, the uk, and australia. University of Michigan, Ann Arbor, Transportation Research Institute, 2014.

SHUJA, J.; MUSTAFA, S.; AHMAD, R. W.; MADANI, S. A.; GANI, A.; KHAN, M. K. Analysis of vector code offloading framework in heterogeneous cloud and edge architectures. *IEEE Access*, IEEE, v. 5, p. 24542–24554, 2017.

Sommer, C.; German, R.; Dressler, F. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, v. 10, n. 1, p. 3–15, 2011.

SOUA, A.; TOHME, S. Multi-level sdn with vehicles as fog computing infrastructures: A new integrated architecture for 5g-vanets. In: IEEE. *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. [S.l.], 2018. p. 1–8.

TEERAPITTAYANON, S.; MCDANEL, B.; KUNG, H. Distributed deep neural networks over the cloud, the edge and end devices. In: IEEE. *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. [S.l.], 2017. p. 328–339.

TSAUR, W.-J.; YEH, L.-Y. Dans: A secure and efficient driver-abnormal notification scheme with i ot devices over i o v. *IEEE Systems Journal*, IEEE, v. 13, n. 2, p. 1628–1639, 2018.

VARGA, A. Using the omnet++ discrete event simulation system in education. *IEEE Transactions on Education*, IEEE, v. 42, n. 4, p. 11–pp, 1999.

VILALTA, R.; VÍA, S.; MIRA, F.; CASELLAS, R.; MUÑOZ, R.; ALONSO-ZARATE, J.; KOUSARIDAS, A.; DILLINGER, M. Control and management of a connected car using sdn/nfv, fog computing and yang data models. In: IEEE. *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. [S.l.], 2018. p. 378–383.

VOLTER, M.; KIRCHER, M.; ZDUN, U. *Remoting patterns: foundations of enterprise, internet and realtime distributed object middleware*. [S.l.]: John Wiley & Sons, 2013.

Wang, X.; Ning, Z.; Wang, L. Offloading in internet of vehicles: A fog-enabled real-time traffic management system. *IEEE Transactions on Industrial Informatics*, v. 14, n. 10, p. 4568–4578, 2018.

WEISS, G. *Multiagent Systems*. [S.l.]: MIT Press, 2013.

XIAO, X.; KUI, X. The characterizes of communication contacts between vehicles and intersections for software-defined vehicular networks. *Mobile Networks and Applications*, Springer, v. 20, n. 1, p. 98–104, 2015.

XU, C.; WANG, Y.; ZHOU, Z.; GU, B.; FRASCOLLA, V.; MUMTAZ, S. A low-latency and massive-connectivity vehicular fog computing framework for 5g. In: IEEE. *2018 IEEE Globecom Workshops (GC Wkshps)*. [S.l.], 2018. p. 1–6.

YAN, L.; ZHANG, M.; SONG, C.; WANG, D.; LI, J.; GUAN, L. Deep learning-based containerization resource management in vehicular fog computing. In: OPTICAL SOCIETY OF AMERICA. *Asia Communications and Photonics Conference*. [S.l.], 2019. p. M4A–213.

YANG, F.; LI, J.; LEI, T.; WANG, S. Architecture and key technologies for internet of vehicles: a survey. *Journal of Communications and Information Networks*, Springer, v. 2, n. 2, p. 1–17, 2017.

YAO, Y.; CHANG, X.; MIŠIĆ, J.; MIŠIĆ, V. Reliable and secure vehicular fog service provision. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 1, p. 734–743, 2018.

YI, S.; LI, C.; LI, Q. A survey of fog computing: concepts, applications and issues. In: ACM. *Proceedings of the 2015 workshop on mobile big data*. [S.l.], 2015. p. 37–42.

ZADEH, N. Theoretical efficiency of the edmonds-karp algorithm for computing maximal flows. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 19, n. 1, p. 184–192, 1972.

Zhang, D.; Vance, N.; Zhang, Y.; Rashid, M. T.; Wang, D. Edgebatch: Towards ai-empowered optimal task batching in intelligent edge systems. In: *2019 IEEE Real-Time Systems Symposium (RTSS)*. [S.l.: s.n.], 2019. p. 366–379.

ZHANG, Y.; WANG, C.-Y.; WEI, H.-Y. Parking reservation auction for parked vehicle assistance in vehicular fog computing. *IEEE Transactions on Vehicular Technology*, IEEE, v. 68, n. 4, p. 3126–3139, 2019.

ZHAO, J.; KONG, M.; LI, Q.; SUN, X. Contract-based computing resource management via deep reinforcement learning in vehicular fog computing. *IEEE Access*, IEEE, v. 8, p. 3319–3329, 2019.

ZHOU, Z.; LIU, P.; FENG, J.; ZHANG, Y.; MUMTAZ, S.; RODRIGUEZ, J. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE Transactions on Vehicular Technology*, IEEE, v. 68, n. 4, p. 3113–3125, 2019.

ZHU, C.; PASTOR, G.; XIAO, Y.; LI, Y.; YLAE-JAEAESKI, A. Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In: IEEE. *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. [S.l.], 2018. p. 1–9.

ZHU, C.; TAO, J.; PASTOR, G.; XIAO, Y.; JI, Y.; ZHOU, Q.; LI, Y.; YLÄ-JÄÄSKI, A. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, IEEE, v. 6, n. 3, p. 4150–4161, 2018.

ZHU, H.; FU, L.; XUE, G.; ZHU, Y.; LI, M.; NI, L. M. Recognizing exponential inter-contact time in vanets. In: IEEE. *INFOCOM, 2010 Proceedings IEEE*. [S.l.], 2010. p. 1–5.

APÊNDICE A – TABELA DE DOMINÂNCIAS DE PARETO

A tabela a seguir contém análises de Fronteiras de Pareto baseado em duas métricas de cliente: RTT e Taxa de Resposta. Uma solução domina a outra quando "cada coordenada da avaliação de A é igual ou maior que a respectiva coordenada de B, havendo pelo menos uma das coordenadas de A que seja maior que a respectiva coordenada e B". (LINDEN, 2008)

Para desempate, usamos o critério de *Crowding* que indica que soluções são boas caso elas estejam em regiões pouco povoadas.Dada as características dos dados, especialmente o RTT, usamos o raio de 0.01 unidades na determinação de regiões povoadas. Valores maiores que esses criam regiões extremamente povoadas que não servem para desempate.

Tabela 10 – Métricas de fronteiras de Pareto baseada em Latência de Serviço de Resposta e Taxa de Resposta

Densidades	Taxa de Requisições	Método de Negociação	Razão de Clientes	Tamanho da Requisição	Latência de Serviço	Taxa de Resposta	Dominância	Crowding	Distance
density00	1	1	2	300	0.007	-1.00	256	74	0.000
density00	1	1	2	2197	0.008	-1.00	255	75	0.001
density00	0.01	1	2	300	0.008	-0.98	255	13	0.023
density01	0.1	1	3	34951	0.008	-0.99	254	87	0.008
density03	1	1	3	300	0.008	-1.00	254	75	0.001
density01	1	1	3	300	0.008	-1.00	253	75	0.001
density00	1	1	3	300	0.008	-0.74	252	14	0.257
density01	0.1	1	2	300	0.008	-1.00	252	79	0.001
density03	1	2	3	300	0.008	-1.00	252	76	0.001
density01	1	2	2	34951	0.009	-1.00	251	76	0.001
density01	0.03	2	3	300	0.008	-1.00	251	79	0.002
density03	1	2	3	34951	0.009	-1.00	250	76	0.002

density03	0.1	1	3	34951	0.008	-0.97	250	13	0.026
density01	0.1	2	3	300	0.008	-1.00	249	84	0.004
density02	1	1	3	300	0.009	-1.00	249	76	0.002
density02	1	2	2	34951	0.009	-1.00	249	84	0.004
density00	0.03	2	2	34951	0.009	-1.00	248	76	0.002
density00	0.1	1	2	34951	0.009	-1.00	247	79	0.002
density01	0.01	1	3	2197	0.008	-0.90	247	3	0.100
density02	1	1	2	300	0.009	-1.00	247	76	0.002
density00	1	2	2	34951	0.009	-1.00	246	76	0.002
density01	0.03	1	3	34951	0.009	-1.00	246	85	0.005
density02	0.1	2	2	300	0.009	-0.99	246	84	0.005
density03	0.1	2	3	34951	0.009	-1.00	246	84	0.004
density00	0.03	1	3	300	0.008	-0.58	245	6	0.418
density01	1	2	3	300	0.009	-1.00	245	78	0.002
density03	1	1	2	34951	0.009	-0.99	245	85	0.006
density03	1	1	3	34951	0.009	-1.00	245	83	0.003
density00	1	2	2	2197	0.009	-1.00	244	78	0.002
density00	0.1	2	2	34951	0.009	-1.00	244	79	0.003
density02	0.1	2	3	300	0.009	-1.00	244	79	0.002
density01	0.03	1	2	300	0.009	-0.99	243	87	0.008
density02	1	1	3	34951	0.010	-1.00	243	80	0.002
density02	0.1	1	2	34951	0.009	-0.99	243	86	0.007
density01	0.1	1	2	34951	0.009	-1.00	242	82	0.002
density01	0.1	2	2	2197	0.009	-1.00	242	79	0.003
density02	1	2	3	34951	0.010	-1.00	242	83	0.003
density02	1	1	2	34951	0.010	-1.00	241	83	0.003

density00	0.03	1	2	300	0.009	-1.00	240	79	0.003
density00	0.03	2	2	300	0.010	-1.00	240	82	0.003
density02	1	2	2	300	0.010	-1.00	240	83	0.003
density00	1	2	2	300	0.010	-1.00	239	83	0.003
density01	1	1	2	300	0.010	-1.00	238	84	0.003
density01	0.03	2	2	300	0.009	-1.00	238	82	0.003
density01	0.03	2	3	34951	0.010	-1.00	238	85	0.003
density02	0.03	2	3	34951	0.010	-1.00	238	84	0.003
density02	1	2	3	300	0.010	-1.00	237	84	0.003
density03	0.1	2	2	300	0.010	-1.00	237	85	0.003
density03	0.1	2	3	300	0.009	-1.00	237	84	0.004
density00	1	1	2	34951	0.010	-1.00	236	84	0.003
density00	1	2	3	300	0.009	-0.74	235	14	0.257
density00	0.01	1	2	34951	0.009	-0.99	235	59	0.010
density01	1	2	3	34951	0.011	-1.00	235	84	0.003
density00	0.1	2	3	300	0.009	-0.74	234	13	0.258
density01	1	2	2	2197	0.011	-1.00	234	84	0.004
density00	0.03	2	2	2197	0.010	-1.00	233	83	0.004
density01	0.03	1	2	34951	0.009	-0.99	233	85	0.006
density01	0.03	2	2	34951	0.010	-1.00	233	82	0.003
density02	1	2	2	2197	0.011	-1.00	233	84	0.004
density00	0.1	2	2	2197	0.011	-1.00	232	84	0.004
density01	0.1	2	3	34951	0.010	-1.00	232	86	0.003
density03	1	2	2	34951	0.009	-1.00	232	84	0.004
density00	0.1	1	2	2197	0.011	-1.00	231	84	0.004
density01	0.1	2	2	300	0.011	-1.00	231	86	0.004

density03	1	2	2	300	0.010	-1.00	231	85	0.003
density00	0.1	2	2	300	0.011	-1.00	230	84	0.004
density01	1	2	2	300	0.011	-1.00	229	84	0.004
density00	0.1	2	3	34951	0.009	-0.73	228	4	0.272
density01	1	2	2	16695	0.018	-1.00	228	74	0.010
density02	0.03	1	3	34951	0.009	-0.98	228	18	0.020
density00	0.1	2	2	16695	0.019	-1.00	227	41	0.012
density00	0.01	2	2	34951	0.009	-0.99	227	44	0.012
density02	0.1	1	2	300	0.009	-0.99	227	44	0.012
density00	1	2	2	16695	0.019	-1.00	226	39	0.012
density01	0.03	1	3	300	0.010	-1.00	225	86	0.004
density02	1	2	2	16695	0.019	-1.00	225	39	0.012
density02	0.1	2	2	34951	0.010	-1.00	225	87	0.004
density01	1	2	3	16695	0.020	-1.00	224	33	0.012
density02	0.1	2	2	2197	0.010	-0.99	224	85	0.006
density00	1	1	3	34951	0.009	-0.74	223	14	0.257
density01	1	1	2	16695	0.020	-1.00	223	32	0.012
density01	0.1	2	2	16695	0.020	-1.00	220	25	0.013
density02	0.1	2	3	34951	0.011	-1.00	220	86	0.004
density00	0.03	2	2	16695	0.020	-1.00	219	35	0.012
density01	0.03	1	3	2197	0.009	-0.99	219	44	0.012
density01	1	1	2	34951	0.009	-0.97	217	14	0.029
density01	0.01	2	3	34951	0.010	-1.00	217	85	0.005
density02	0.03	1	3	300	0.009	-0.97	217	11	0.034
density03	1	1	2	300	0.010	-0.99	217	59	0.010
density00	0.1	1	2	300	0.011	-1.00	216	86	0.004

density00	0.03	1	2	34951	0.010	-0.99	215	85	0.006
density02	0.03	2	3	2197	0.009	-0.96	213	7	0.041
density00	0.03	2	3	34951	0.009	-0.70	212	2	0.295
density01	1	1	3	34951	0.010	-0.99	211	41	0.014
density01	0.1	1	3	300	0.010	-0.99	210	85	0.006
density01	0.1	2	2	34951	0.011	-1.00	207	92	0.005
density00	0.01	1	3	300	0.010	-0.73	206	14	0.265
density03	1	2	3	2197	0.010	-0.99	206	88	0.007
density00	1	2	3	34951	0.010	-0.74	204	14	0.257
density01	1	1	2	2197	0.010	-0.99	203	83	0.010
density02	1	1	2	2197	0.010	-0.99	203	87	0.008
density03	0.1	1	2	34951	0.010	-0.97	203	14	0.028
density01	0.01	1	3	34951	0.010	-0.98	202	29	0.016
density01	0.03	2	2	2197	0.010	-0.98	201	24	0.017
density00	0.03	1	2	2197	0.011	-0.99	200	89	0.006
density01	0.1	1	2	2197	0.011	-0.99	200	86	0.007
density02	0.1	1	3	300	0.011	-0.99	200	88	0.007
density00	0.03	1	3	34951	0.010	-0.74	199	14	0.257
density01	0.03	1	2	2197	0.010	-0.94	196	7	0.059
density02	1	1	3	2197	0.010	-0.98	196	20	0.022
density00	0.03	1	3	2197	0.010	-0.58	195	8	0.416
density01	0.1	2	3	2197	0.010	-0.98	195	21	0.022
density00	0.1	2	3	2197	0.010	-0.58	194	8	0.419
density01	0.03	2	3	2197	0.010	-0.90	191	5	0.101
density02	0.1	1	3	34951	0.011	-0.99	191	44	0.013
density00	0.01	1	3	2197	0.010	-0.63	190	2	0.367

density03	0.03	2	3	34951	0.011	-0.99	190	87	0.009
density00	0.1	1	3	34951	0.010	-0.74	189	14	0.262
density00	0.1	1	3	300	0.010	-0.74	188	14	0.258
density03	1	1	3	2197	0.010	-0.95	188	6	0.049
density01	0.01	2	3	300	0.047	-0.99	186	1	0.040
density00	0.01	2	2	300	0.016	-0.99	185	83	0.011
density00	0.01	2	2	2197	0.011	-0.98	175	27	0.016
density03	0.1	1	3	300	0.011	-0.96	175	11	0.036
density01	1	2	3	2197	0.013	-0.99	174	37	0.015
density02	0.03	2	3	300	0.011	-0.90	174	5	0.097
density01	0.1	1	3	2197	0.011	-0.93	173	9	0.066
density00	0.03	2	3	300	0.010	-0.74	172	14	0.257
density03	0.1	1	2	2197	0.011	-0.88	172	2	0.123
density01	0.01	2	3	2197	0.014	-0.98	169	21	0.022
density00	0.1	1	2	16695	0.019	-0.98	168	11	0.021
density03	0.1	2	3	2197	0.011	-0.94	167	8	0.059
density03	1	1	2	2197	0.011	-0.93	166	8	0.066
density03	0.1	1	2	300	0.011	-0.97	165	14	0.029
density03	0.03	2	3	300	0.650	-0.99	163	1	0.643
density01	0.1	1	3	16695	0.018	-0.97	162	11	0.029
density02	1	2	3	2197	0.011	-0.93	162	10	0.068
density03	1	2	2	2197	0.011	-0.92	162	7	0.076
density00	1	1	2	16695	0.020	-0.98	161	7	0.023
density01	1	1	3	2197	0.012	-0.96	161	8	0.041
density03	0.1	2	2	34951	0.014	-0.97	161	10	0.032
density00	0.01	1	2	2197	0.015	-0.97	160	11	0.034

density01	0.01	1	3	300	0.084	-0.98	160	1	0.079
density03	0.1	2	2	2197	0.013	-0.96	158	6	0.044
density02	0.03	1	3	2197	0.011	-0.87	157	2	0.131
density02	0.03	2	2	34951	0.607	-0.98	156	1	0.600
density03	0.1	1	3	2197	0.011	-0.77	156	1	0.231
density02	0.1	2	3	2197	0.011	-0.89	153	2	0.110
density03	0.03	1	3	300	0.012	-0.92	153	6	0.077
density02	0.1	1	2	2197	0.013	-0.93	152	9	0.073
density02	0.1	1	3	2197	0.012	-0.92	152	6	0.081
density03	0.03	1	3	34951	0.014	-0.94	152	8	0.059
density02	0.03	2	2	300	0.914	-0.97	148	1	0.907
density02	1	2	3	16695	0.018	-0.93	147	7	0.067
density02	0.03	1	2	2197	0.012	-0.75	146	8	0.249
density03	0.03	2	3	2197	0.015	-0.91	144	7	0.088
density02	0.03	1	2	34951	0.414	-0.96	143	1	0.408
density01	0.1	1	2	16695	0.021	-0.94	141	4	0.059
density03	1	2	3	16695	0.018	-0.92	141	6	0.080
density00	0.1	1	3	2197	0.011	-0.64	140	1	0.355
density01	1	1	3	16695	0.018	-0.91	140	8	0.095
density00	0.01	1	3	34951	0.012	-0.74	135	14	0.262
density02	1	1	3	16695	0.019	-0.91	135	5	0.090
density02	0.03	1	2	300	0.219	-0.94	135	1	0.221
density00	1	2	3	2197	0.012	-0.72	134	2	0.284
density00	0.01	2	3	300	0.012	-0.74	134	13	0.262
density00	0.01	2	3	2197	0.012	-0.71	134	3	0.293
density00	0.01	2	3	34951	0.013	-0.74	134	13	0.260

density01	0.03	2	3	16695	0.019	-0.91	131	7	0.096
density00	0.03	2	3	2197	0.012	-0.58	130	9	0.419
density00	1	1	3	2197	0.012	-0.58	128	9	0.419
density02	0.03	2	2	2197	0.798	-0.93	128	1	0.794
density02	1	1	2	16695	0.023	-0.91	126	5	0.092
density01	0.1	2	3	16695	0.023	-0.90	121	3	0.102
density00	0.03	1	2	16695	0.022	-0.88	120	1	0.121
density03	1	1	2	16695	0.022	-0.79	118	3	0.211
density03	1	2	2	16695	0.022	-0.78	117	3	0.216
density03	0.03	1	3	2197	0.015	-0.63	117	2	0.374
density02	0.1	2	2	16695	0.022	-0.78	115	3	0.218
density00	1	1	3	16695	0.018	-0.58	106	9	0.419
density01	0.03	1	3	16695	0.019	-0.70	106	2	0.305
density00	0.1	2	3	16695	0.019	-0.58	102	8	0.419
density00	1	2	3	16695	0.019	-0.58	100	8	0.419
density00	0.1	1	3	16695	0.019	-0.57	98	2	0.434
density02	0.1	1	3	16695	0.020	-0.57	93	3	0.429
density03	1	1	3	16695	0.023	-0.69	93	2	0.310
density01	0.03	2	2	16695	0.031	-0.66	91	1	0.343
density02	0.1	2	3	16695	0.023	-0.60	91	1	0.400
density00	0.03	2	3	16695	0.021	-0.58	87	7	0.420
density00	0.01	2	2	16695	0.061	-0.60	86	1	0.399
density01	0.01	2	2	300	15.602	-0.70	85	1	15.597
density01	0.01	1	2	34951	16.459	-0.67	82	1	16.455
density03	0.1	1	3	16695	0.022	-0.29	82	1	0.709
density03	0.1	2	3	16695	0.022	-0.52	82	1	0.482

density01	0.01	2	2	34951	17.214	-0.66	81	1	17.210
density01	0.01	1	2	2197	13.464	-0.61	79	1	13.462
density01	0.01	2	2	2197	17.996	-0.65	79	1	17.992
density01	0.01	1	2	300	17.706	-0.64	78	1	17.703
density00	0.03	1	3	16695	0.023	-0.50	76	1	0.497
density02	0.1	1	2	16695	0.028	-0.55	76	1	0.453
density03	0.1	2	2	16695	0.023	-0.47	76	1	0.530
density03	0.03	1	2	34951	27.145	-0.62	74	1	27.140
density02	0.03	2	3	16695	0.024	-0.38	73	1	0.620
density03	0.03	1	2	300	25.028	-0.61	73	1	25.024
density03	0.03	2	2	34951	31.693	-0.62	73	1	31.688
density01	0.01	2	3	16695	0.032	-0.49	72	1	0.513
density03	0.03	2	2	2197	25.452	-0.61	72	1	25.447
density00	0.01	2	3	16695	0.030	-0.32	71	1	0.680
density01	0.03	1	2	16695	0.032	-0.40	71	1	0.598
density02	0.03	1	3	16695	0.026	-0.20	71	1	0.802
density03	0.1	1	2	16695	0.027	-0.28	71	1	0.721
density02	0.03	2	2	16695	0.031	-0.30	69	1	0.702
density02	0.03	1	2	16695	0.030	-0.18	68	1	0.821
density03	0.03	2	3	16695	0.031	-0.21	68	1	0.790
density00	0.01	1	2	16695	0.060	-0.34	67	1	0.660
density03	0.03	2	2	300	32.234	-0.59	67	1	32.229
density03	0.03	1	2	2197	2.551	-0.51	64	1	2.591
density00	0.01	1	3	16695	0.041	-0.19	62	1	0.816
density03	0.03	2	2	16695	0.032	-0.11	61	1	0.888
density02	0.01	1	3	2197	11.806	-0.41	60	1	11.814

density03	0.03	1	3	16695	0.035	-0.09	60	1	0.907
density01	0.01	1	3	16695	0.307	-0.25	59	1	0.807
density03	0.01	1	3	2197	1.274	-0.30	57	1	1.447
density02	0.01	1	2	2197	13.966	-0.35	56	1	13.974
density03	0.03	1	2	16695	0.091	-0.06	54	1	0.946
density02	0.01	1	3	34951	34.683	-0.50	51	1	34.679
density02	0.01	2	3	2197	34.094	-0.49	51	1	34.091
density02	0.01	2	3	300	36.588	-0.49	49	1	36.584
density01	0.01	2	2	16695	2.299	-0.13	48	1	2.450
density02	0.01	1	3	300	35.398	-0.47	48	1	35.395
density02	0.01	2	3	16695	1.480	-0.08	46	1	1.737
density02	0.01	2	3	34951	39.599	-0.45	45	1	39.595
density03	0.01	1	3	34951	38.107	-0.42	45	1	38.104
density02	0.01	1	2	34951	36.548	-0.40	44	1	36.546
density03	0.01	2	3	34951	45.335	-0.42	44	1	45.331
density01	0.01	1	2	16695	4.490	-0.07	43	1	4.577
density02	0.01	2	2	300	38.788	-0.41	43	1	38.785
density02	0.01	2	2	16695	2.966	-0.06	43	1	3.104
density02	0.01	1	3	16695	2.624	-0.04	42	1	2.787
density02	0.01	1	2	300	37.262	-0.37	41	1	37.260
density03	0.01	1	3	300	38.356	-0.38	41	1	38.353
density02	0.01	2	2	2197	37.594	-0.37	40	1	37.592
density03	0.01	2	3	2197	40.929	-0.40	40	1	40.926
density02	0.01	1	2	16695	5.845	-0.03	39	1	5.917
density02	0.01	2	2	34951	39.193	-0.38	39	1	39.190
density03	0.01	1	2	2197	19.954	-0.15	39	1	19.965

density03	0.01	2	3	16695	7.681	-0.03	38	1	7.734
density03	0.01	1	3	16695	10.678	-0.02	37	1	10.716
density03	0.01	2	3	300	45.439	-0.39	37	1	45.436
density03	0.01	2	2	16695	14.071	-0.01	33	1	14.099
density03	0.01	1	2	16695	14.667	0.00	32	1	14.693
density03	0.01	1	2	34951	56.900	-0.22	23	1	56.898
density03	0.01	2	2	300	63.860	-0.23	23	1	63.858
density03	0.01	2	2	2197	56.183	-0.21	22	1	56.182
density03	0.01	2	2	34951	64.140	-0.23	22	1	64.138
density03	0.01	1	2	300	58.676	-0.20	19	1	58.675

Fonte: Elaborada pelo autor (2021)