Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Serviços Governados pela Comunidade na Borda da Rede

João Victor Barroso Mafra

Campina Grande, Paraíba, Brasil
2020

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Serviços Governados pela Comunidade na Borda da Rede

João Victor Barroso Mafra

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Gerência de recursos em infraestruturas de computação utilitária

Francisco Vilar Brasileiro (Orientador)

Campina Grande, Paraíba, Brasil

M187s Mafra, João Victor Barroso.
Serviços governados pela comunidade na Borda da Rede/João Victor Barroso Mafra. - Campina Grande, 2020.
60f. : il. Color.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2020.
"Orientação: Prof. Dr. Francisco Vilar Brasileiro".
Referências.

1. Computação na Borda. 2. Comunidade. 3. Privacidade. I. Brasileiro, Francisco Vilar. II. Título.

CDU 004.7(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA ITAPUANA SOARES DIAS CRB-15/93

SERVIÇOS GOVERNADOS PELA COMUNIDADE NA BORDA DA REDE

JOÃO VICTOR BARROSO MAFRA

DISSERTAÇÃO APROVADA EM 30/07/2020

FRANCISCO VILAR BRASILEIRO, Ph.D, UFCG
Orientador(a)

FÁBIO JORGE ALMEIDA MORAIS, Dr., UFCG
Examinador(a)

ROSTAND EDSON OLIVEIRA COSTA, Dr., UFPB
Examinador(a)

CAMPINA GRANDE - PB

Resumo

A popularização de smartphones cada vez mais ricos em recursos computacionais permitiu o surgimento de uma grande variedade de aplicações nos últimos anos. Tipicamente, essas aplicações usam infraestruturas remotas na nuvem para processar dados. Em muitos casos, o dado a ser processado (ou parte dele) é coletado pelos dispositivos dos usuários e depois enviado para nuvem. Nessa arquitetura, o provedor de aplicação de nuvem centraliza as ações e é o único responsável por definir a governança da aplicação e dos dados envolvidos. Isso não é satisfatório do ponto de vista da privacidade e pode não ser viável a longo prazo, devido aos custos associados para manter o serviço executando na nuvem. Este trabalho propõe uma arquitetura na qual o serviço é governado pelos usuários de uma comunidade que possuem algum problema em comum para resolver. Para tornar isso possível, foram usados os conceitos de Sensoriamento Participativo, Redes Sociais Móveis e Computação na Borda, que permite o processamento próximo às fontes geradoras de dados, como os próprios dispositivos dos usuários. A arquitetura proposta desse novo modelo de serviço é detalhada, e em seguida é descrito um estudo de caso para avaliar a viabilidade e a qualidade da solução proposta comparada com outras soluções já existentes. O estudo de caso se deu através da execução de experimentos de simulação usando dados do sistema de transporte público. Os resultados mostram que a abordagem proposta é viável, e consegue agregar tantos dados quanto as abordagens já existentes que usam servidores remotos dedicados. Além disso, diferentemente das políticas de compartilhamento "tudo ou nada" das abordagens atuais, a abordagem proposta permite que os usuários configurem de forma autônoma o *trade-off* existente entre o compartilhamento de seus dados privados e o desempenho que a aplicação pode atingir.

Abstract

The popularization of resource-rich smartphones has enabled a wide range of new applications to emerge. Typically, these applications use a remote cloud to process data. In many cases, the data processed (or part of it) is collected by the users' devices and sent to the cloud. In this architecture, the external cloud provider is the sole responsible for defining the governance of the application and all its data. This is not satisfactory from the privacy viewpoint, and may not be feasible in the long run. This document proposes an architecture in which the service is governed by the users of a community who have a common problem to solve. To make it possible, the concepts of Participatory Sensing, Mobile Social Networks (MSN) and Edge Computing were used, which enable data processing closer to the data sources (i.e. the users' devices). This work describes the proposed architecture and a case study to assess the feasibility and quality of our solution compared with other solutions already in place. The case study uses simulation experiments fed with real data from the public transport system of Curitiba, a city in the South of Brazil with a population of approximately 2 million people. The results show that the proposed approach is feasible, and can aggregate as much data as current approaches that use remote dedicated servers. Differently from the all-or-nothing sharing policy of current approaches, the approach proposed allows users to autonomously configure the trade-off between the sharing of private data, and the performance that the application can achieve.

Agradecimentos

Gostaria de agradecer primeiramente a Deus pela saúde, pela força e por guiar o meu caminho durante essa jornada. Depois, aos meus pais Milton e Graça, por sempre acreditarem em mim e proverem todo suporte necessário durante toda minha vida de estudante para que eu pudesse chegar até aqui, além do meu irmão, Luis Eduardo, hoje também um companheiro de projeto no laboratório.

Ao meu orientador, Fubica, pela paciência e por todo aprendizado transmitido desde quando eu era apenas um graduando chegando ao meu primeiro projeto de pesquisa 4 anos atrás. Esse mesmo agradecimento se estende à professora Raquel, que apesar de não ser minha orientadora de forma oficial, participou de inúmeras discussões que culminaram na realização deste trabalho.

Ao professor Nazareno e seu aluno Tarciso, por disponibilizarem os dados utilizados no estudo de caso deste trabalho.

A todos os amigos que fiz no Laboratório de Sistemas Distribuídos, em especial a Fábio e Giovanni, com os quais dividi a mesma sala por um período considerável, pela ajuda em diversos momentos e por transmitirem seus conhecimentos como pesquisadores mais experientes.

Aos meus amigos que me acompanham desde a infância até os dias atuais, em especial Antônio, Chico e Gabriel Leão, além de vários amigos que fiz durante a minha caminhada universitária como Edval, Laércio, Ítalo, Sales e Wayne. Obrigado por sempre acreditarem em mim e por todos os momentos de descontração.

Para finalizar, gostaria de agradecer o apoio financeiro da Ericsson Telecomunicações e do povo brasileiro.

Conteúdo

Acrônimos

QoS - *Quality of Service*

GPS - *Global Positioning System*

IA - *Inteligência Artificial*

FL - *Federated Learning*

MSN - *Mobile Social Network*

P2P - *Peer-to-Peer*

AODV - *Ad hoc On-demand Distance Vector*

URBS - *Urbanização de Curitiba S/A*

UTFPR - *Universidade Tecnológica Federal do Paraná*

BFS - *Breadth-First Search*

WSS - *Within-cluster Sum of Squares*

Lista de Símbolos

Modelo de simulação

t_a - horário real de partida de um usuário do ponto de ônibus

t_c - horário real de chegada de um usuário no ponto de ônibus

t_r - horário em que um usuário fez uma requisição ao serviço

t_e - horário estimado pelo serviço para que o usuário se dirija até o ponto de ônibus

t_d - horário da saída do próximo ônibus após t_e

b - linha de ônibus arbitrária

$R^b_{t_r}$ - requisição feita por um usuário no horário t_r desejando saber o horário de partida do próximo ônibus da linha b

Configuração de compartilhamento de dados

t - timestamp arbitrário

u - usuário provedor de dados

u' - usuário receptor de dados

m - número de usuários do sistema dos quais um usuário pode receber dados

i - identificador da rede social real usada na simulação

Métricas

DP - proporção de requisições que podem ser respondidas usando dados do passado

QD - quantidade de dados usados para fazer uma predição

TEP - tempo de espera na parada

PP - frequência com a qual os usuários não conseguem pegar o ônibus considerando uma janela de uma hora

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Nos últimos anos alcançamos estatísticas que dizem existir mais telefones celulares do que escovas de dentes no planeta. Estar conectado o tempo inteiro é a nova ordem. O telefone celular é um dispositivo de processamento e comunicação que serve para informar, guiar, entreter e, algumas vezes, fazer chamadas telefônicas. Os dispositivos atuais são inteligentes e equipados com todas os recursos de um micro computador pessoal. Além do mais, um smartphone é um dispositivo móvel com sensores como o GPS (*Global Positioning System*), sensores de luz ambiente e proximidade, microfones, acelerômetros e giroscópios, apenas para mencionar alguns. Com esses micro computadores online o tempo inteiro podemos imaginar uma grande variedade de novas possibilidades, aplicações, e ao mesmo tempo novos desafios para lidar.

Em 2006 surgiu o conceito de Sensoriamento Participativo [6], que utiliza os recursos dos smartphones dos usuários, naquele momento ainda bem limitados, para coletar informações úteis do ambiente. Anos depois, esse conceito de Sensoriamento Participativo foi estendido para uma nova dimensão, quando essas capacidades de sensoriamento funcionam em associação com participantes de uma Rede Social Móvel (MSN, do inglês *Mobile Social Network*) [9]. Essa nova maneira de fazer software explora os dispositivos móveis (não limitando-se apenas aos smartphones, mas também usando outros acessórios inteligentes) para coletar, analisar e compartilhar conhecimento, em uma técnica conhecida como *mobile*

crowdsensing. Sensoriamento Participativo e *crowdsensing*[1] podem ser utilizados por um conjunto de usuários para atingir algum objetivo comum. Por exemplo, a aplicação *GreenGPS* utiliza dados coletados do ambiente para mapear o consumo de combustível nas ruas da cidade, permitindo que os motoristas encontrem as rotas mais eficientes em termos de consumo de combustível para seus veículos em pontos de destino arbitrários [8]. Muitas aplicações como essa surgiram [20, 15, 19], todas elas explorando as capacidades de sensoriamento participativo e oportunístico dos dispositivos móveis.

Todas essas aplicações possuem algo mais em comum do que apenas a coleta de dados do ambiente de forma colaborativa. Os participantes que alimentam o sistema estão engajados por um objetivo comum que é alcançado de forma colaborativa. Elas são criadas, anunciadas e, após isso, usuários interessados se juntam a elas, coletando dados de forma colaborativa e usando-as eventualmente. As aplicações estão hospedadas em infraestruturas de nuvem e, portanto, requerem algum tipo de patrocínio, gerenciamento e suporte técnico para torná-las possíveis.

Na última década, vimos também a evolução do Aprendizado de Máquina e outras técnicas de Inteligência Artificial (IA) com o propósito de extrair predições/respostas úteis a partir dos dados brutos que foram coletados de alguma maneira. Quando esses dados são coletados de forma colaborativa, como nos exemplos anteriores, eles são tradicionalmente enviados para um provedor de aplicação de nuvem centralizado, onde modelos de aprendizado de máquina são treinados. Novamente, há a necessidade de patrocínio e suporte técnico, já que o modelo global está hospedado em uma infraestrutura de nuvem remota.

Além disso, quando o dado envolvido é sensível, questões de privacidade devem ser levadas em consideração. Para tentar solucionar esse problema, uma nova arquitetura para construir tais modelos de aprendizado de máquina foi concebida: Aprendizado Federado (FL, do inglês *Federated Learning*) [4]. Nela, modelos de IA são construídos independentemente nos dispositivos de cada usuário e depois unidos com o modelo global que está localizado na nuvem.

[1]Em uma tradução literal, o termo significa "sensoriamento em grupo", e refere-se à técnica de coletar dados usando os dispositivos móveis de indivíduos, e que serão usados para algum propósito.

1.2 Definição do Problema

O Sensoriamento Participativo, as Redes Sociais Móveis e o Aprendizado Federado são tecnologias que permitem que o conhecimento compartilhado seja construído sobre dados brutos coletados de forma colaborativa. Contudo, todos eles precisam de um serviço de hospedagem logicamente centralizado onde a aplicação é executada, geralmente na nuvem.

A arquitetura tradicional cliente-servidor traz para esse cenário um ponto único de falha e exposição. Nesse quesito, a preocupação não é com falhas de hardware e software, que podem ser tratadas com relativa facilidade empregando várias técnicas de replicação. O foco está em problemas relacionados ao ciclo de vida da aplicação, governada, nesse caso, por um provedor de aplicação de nuvem. Manter a governança da aplicação nas mãos do provedor de aplicação de nuvem pode ser pouco interessante por uma série de razões. Por exemplo, os usuários estão sujeitos a mudanças unilaterais nas políticas de uso, decididas pelo provedor. Em situações mais graves, os provedores podem falir ou simplesmente decidir parar de oferecer suporte ao serviço, geralmente sem qualquer responsabilidade em relação aos usuários.

Além disso, o fato de todos os dados serem mantidos por essa única entidade pode levantar questões sobre privacidade e uso indevido dos dados dos usuários, que possuem um controle limitado sobre como esses dados são armazenados na nuvem. Em geral, as aplicações expõem uma política de uso dos dados pessoais, que pode ser lida e aceita (ou não) pelos usuários. Apesar de deixar os usuários, em tese, cientes do uso de suas informações, esse modelo de governança dos dados pode ser arriscado para os usuários. Primeiramente, o provedor da aplicação pode fornecer poucos detalhes, ou até mentir, sobre como esses dados são utilizados. O Facebook, por exemplo, recebeu uma multa severa do governo dos Estados Unidos da América pelo uso de forma indevida dos dados de usuários[2]. Além do mais, a utilização da aplicação está muitas vezes condicionada à aceitação dos termos de uso dos dados. Sendo assim, caso o usuário não concorde com a política, ele pode não conseguir usufruir do serviço. Em outras palavras, geralmente é uma decisão do tipo "tudo ou nada".

Este trabalho refere-se a todas as problemáticas acima como *a necessidade de governança externa*. A governança externa tira da comunidade de usuários o direito de decidir

[2]https://gauchazh.clicrbs.com.br/mundo/noticia/2019/07/eua-aplicam-multa-recorde-de-us-5-bi-ao-facebook-por-uso-de-dados-pessoais-cjyh9k3zi006n01k0lomd4bc1.html

como os dados são compartilhados, onde são armazenados, quando e por quem são usados. Além disso, requer recursos extras e algum tipo de suporte técnico (e custos associados) para manter a aplicação executando na nuvem.

1.3 Objetivos

O objetivo deste trabalho é propor e avaliar um novo modelo de aplicação que concede a governança do serviço oferecido à própria comunidade de usuários que o utiliza. Sendo assim, não há dependência de um provedor de aplicação de nuvem externo, como ocorre naturalmente nas tradicionais aplicações baseadas em nuvem (do inglês, *cloud-based applications*).

Os seguintes objetivos específicos foram definidos:

- Definir um modelo arquitetural para os Serviços Governados pela Comunidade, incluindo os componentes envolvidos, como eles atuam para garantir o funcionamento do serviço, além dos requisitos necessários para as aplicações que podem utilizar o modelo proposto.

- Identificar um estudo de caso que se encaixe nos requisitos do modelo proposto e avaliar a viabilidade da solução nesse estudo de caso por meio de experimentos de simulação.

- Utilizar o mesmo estudo de caso para comparar o modelo proposto com outras soluções já existentes, como por exemplo a arquitetura tradicional baseada em nuvem ou arquiteturas de computação na borda que utilizam servidores centralizados instalados em pontos estratégicos.

- Estudar diferentes níveis de confiança entre os usuários da comunidade durante o compartilhamento dos dados. Em situações reais, espera-se que os indivíduos queiram compartilhar seus dados coletados apenas com pessoas nas quais confiam e não com todos os usuários do serviço. A depender da rede de confiança de um indivíduo, ele pode obter mais ou menos informações.

1.4 Solução Proposta

Neste trabalho propõe-se a ideia de **Serviços Governados pela Comunidade**, um modelo de aplicação que visa dar a governança do serviço oferecido aos próprios membros da comunidade que o utilizam. Ele modifica a governança comumente vista nas aplicações atuais, referenciada neste trabalho como *governança externa*, na qual o controle da vida da aplicação e dos dados é feito por um provedor de aplicação de nuvem externo, ainda que no meio desse processo exista algum consentimento por parte dos usuários em relação à política de uso dos dados.

Em um serviço governado pela comunidade, os indivíduos precisam de um serviço específico cuja qualidade será melhor se eles compartilharem dados entre si. Além disso, os usuários estão conectados por uma MSN na qual os indivíduos têm interesses semelhantes. Motivados pelo seu interesse comum, eles coletam dados do ambiente utilizando sensores presentes em seus dispositivos móveis e trocam esses dados com seus pares confiáveis para que possam tomar melhores decisões.

Um exemplo de aplicação nesses moldes seria o caso de vizinhos compartilhando dados coletados da vizinhança pelos seus celulares com o objetivo de obter um relatório atualizado sobre a qualidade do ar do ambiente onde vivem. Além desse, usuários em um ambiente amplo e fechado, como um shopping center, podem estar interessados em locais com menos poluição sonora e podem recorrer a dados coletados coletivamente para identificar esses locais rapidamente [22].

Os Serviços Governados pela Comunidade exploram a ideia de Sensoriamento Participativo, porém limitando a troca de dados aos parceiros confiáveis da rede social e excluindo os serviços de terceiros, como provedores de nuvem. Para fazer isso, os Serviços Governados pela Comunidade seguem uma arquitetura par-a-par (P2P, do inglês *peer-to-peer*): os dados são gerados e processados na borda da rede sem a necessidade de serem transferidos para a nuvem para serem processados. Sendo assim, o modelo proposta usa o poder de processamento dos próprios dispositivos na borda da rede, como os smartphones, para coletar, processar e armazenar os dados compartilhados.

O paradigma de Computação na Borda [24] é um componente essencial da ideia proposta neste trabalho. Os mesmos dispositivos que estão coletando os dados também estão

processando e analisando os mesmos. Dessa forma, evita-se a inundação de dados na rede, economiza-se largura de banda e reduz-se a latência da aplicação, provendo uma melhor experiência para o usuário.

1.5 Contribuição

As principais contribuições deste trabalho são: i) o modelo dos **Serviços Governados pela Comunidade**, que permitem que a governança da aplicação seja exercida pelos usuários da mesma e ii) o **estudo de caso** que avalia a viabilidade do modelo em termos de quantidade de dados agregada, além de ilustrar um exemplo no qual conseguimos obter uma melhor qualidade de serviço (*QoS*, do inglês *Quality of Service*) quando mais dados são agregados. Parte dos resultados deste trabalho foram apresentados e publicados na *International Conference on Cloud Computing and Services Science (CLOSER 2020)* [16].

1.6 Organização do Trabalho

O restante deste trabalho discute essa nova abordagem de serviço, incluindo seu modelo arquitetural, o *modus operandi*, os requisitos das aplicações que podem utilizá-lo e possíveis limitações ocasionadas pelo uso de dispositivos móveis, além de um estudo de caso através de experimentos de simulação alimentados com dados reais. O Capítulo 2 apresenta conceitos importantes e os trabalhos relacionados aos tópicos que englobam a solução proposta. O Capítulo 3 define e modela o funcionamento dos Serviços Governados pela Comunidade, além de tratar dos requisitos das aplicações elegíveis. O Capítulo 4 propõe um estudo de caso do serviço proposto no sistema de transporte público de ônibus, bem como descreve os materiais e métodos utilizados nos experimentos de simulação realizados, seguido dos resultados e discussões. Por fim, as conclusões são tratadas no Capítulo 5.

Capítulo 2

Fundamentação Teórica e Trabalhos Relacionados

O objetivo desse capítulo é organizar os conceitos importantes e explorar os trabalhos relacionados à solução proposta no Capítulo 3 e ao estudo de caso descrito no Capítulo 4.

2.1 Fundamentação Teórica

A solução proposta neste documento busca conciliar os conceitos de aplicações *crowdsensing* (apoiadas pelo sensoriamento participativo), MSN e Computação na Borda. Sendo assim, é possível coletar informações do ambiente usando o sensoriamento participativo, compartilhá-las com pessoas em uma zona próxima usando redes sociais móveis e ainda processá-las na borda da rede, sem a necessidade de envio para um servidor na nuvem.

O Sensoriamento Participativo diz respeito à coleta de dados do ambiente por parte de um conjunto de indivíduos com o objetivo de formar um corpo de conhecimento. Sua arquitetura inicial é descrita no trabalho de Burke et al. [6], acompanhada de exemplos de possíveis aplicações nas áreas de saúde pública e planejamento urbano. *Mobile Crowdsensing* representa uma versão mais consolidada e mais bem definida do Sensoriamento Participativo. A ideia é que cidadãos comuns usem seus celulares para coletar dados que serão enviados para a nuvem, onde serão unidos e processados para extrair alguma inteligência [9]. O crescente uso de smartphones permitiu o surgimento das Redes Sociais Móveis, que permitem a interação entre indivíduos com interesses semelhantes que estão próximos uns dos outros usando uma

arquitetura P2P [26]. Assim, as pessoas podem não apenas enviar os dados coletados para a nuvem, mas também compartilhá-los com outros membros da comunidade.

O trabalho de Alsheikh et al. [2] apresenta uma arquitetura geral desse modelo tradicional de aplicações que envolvem a coleta de dados e o uso da nuvem para processá-los. A Figura 2.1, retirada daquele trabalho, mostra a presença de vários dispositivos na borda da rede capazes de coletar informações do ambiente, como celulares, sensores presentes em casas e carros, e acessórios inteligentes como pulseiras e relógios. Os dados coletados são enviados através da Internet para a nuvem, onde eles são usados de fato. Esses dados podem ser visualizados ou treinados usando técnicas de IA.

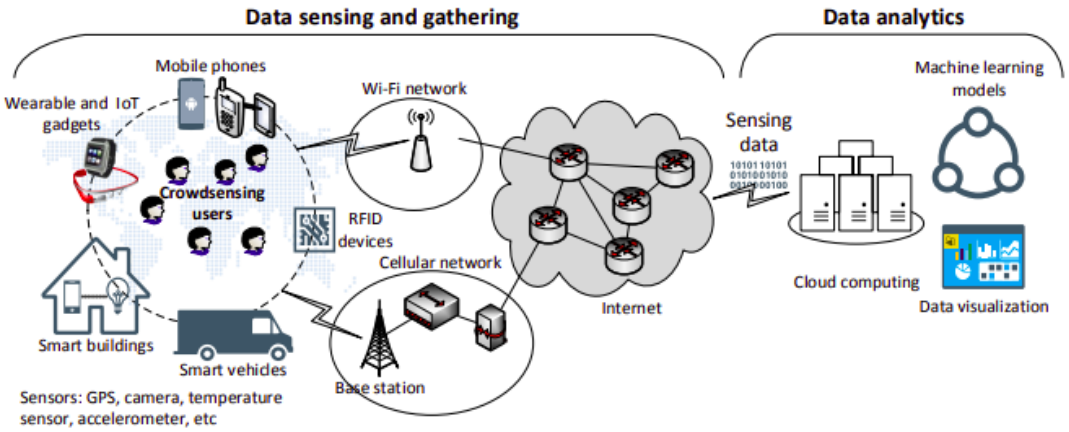

Figura 2.1: Arquitetura geral de uma aplicação crowdsensing [2]

Para mitigar questões a respeito da privacidade dos dados coletados que são enviados para a nuvem, o Aprendizado Federado [4] permite que técnicas de IA sejam usadas em cada dispositivo, gerando modelos locais que são depois unidos na nuvem. Nesse caso, não há troca de dados entre os usuários e o treinamento que ocorre em seus dispositivos é feito usando apenas os dados armazenados localmente. Apesar de mitigar a preocupação com a privacidade dos dados, ainda há dependência de uma infraestrutura na nuvem onde o modelo global será construído.

Com o objetivo de melhorar o tempo de resposta das aplicações e economizar largura de banda, surgiu o paradigma da Computação na Borda (do inglês, *Edge Computing*), referenciado muitas vezes como Computação na Névoa (do inglês, *Fog Computing*) [5, 24]. Ele estende o paradigma da Computação na Nuvem considerando recursos que residem entre os usuários finais e a nuvem centralizada, e que fornecem serviços de computação, armazena-

mento e rede, próximos aos usuários. Assim, os dados coletados no modelo de aplicação descrito na Figura 2.1 seriam processados em servidores instalados mais próximos dos dispositivos que geram os dados, ou até nos próprios dispositivos, como propõe a solução apresentada neste trabalho. A Figura 2.2 mostra uma ideia geral dessa arquitetura. O objetivo é que os dados coletados pelos *edge devices* (ou parte deles) não sejam enviados para a camada *cloud* e sejam processados em uma camada mais próxima dos usuários (*edge nodes*).

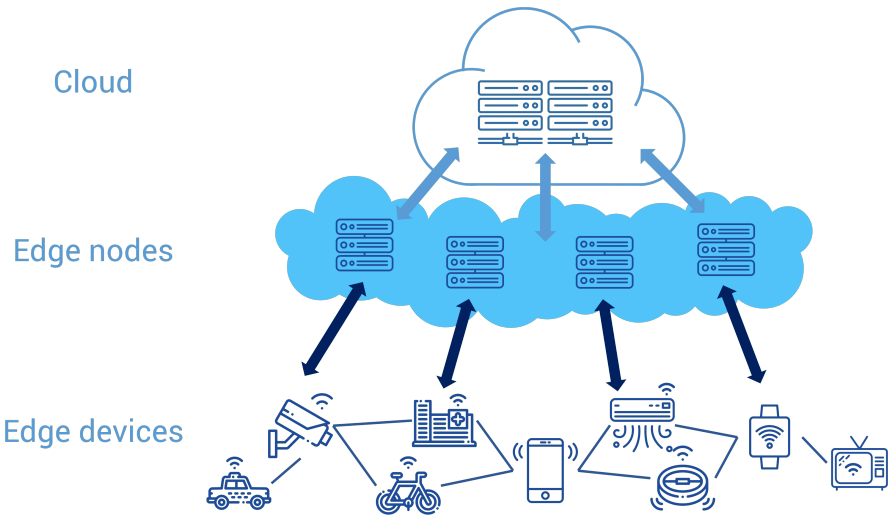

Figura 2.2: Arquitetura geral do paradigma de Computação na Borda. Retirado de https://www.alibabacloud.com/knowledge/what-is-edge-computing

2.2 Trabalhos Relacionados

Boa parte dos trabalhos abordam os conceitos atrelados ao modelo proposto de forma separada. Sendo assim, essa seção explora inicialmente os conceitos de aplicações crowdsensing e computação na borda separadamente, para depois elencar os trabalhos que unem ambos os conceitos, se aproximando mais da ideia proposta neste documento.

2.2.1 Aplicações Crowdsensing

Conforme apresentado na seção anterior, os usuários tipicamente coletam informações usando seus dispositivos móveis e enviam os dados para uma nuvem centralizada onde modelos de IA são treinados. Uma grande variedade de aplicações pode ser construída usando esse modelo.

CenceMe [18] é uma aplicação que explora o poder de sensoriamento dos telefones celulares para inferir informações do ambiente e compartilhá-las em redes sociais como o Facebook. Esse tipo de funcionalidade já é bastante presente nas redes sociais atuais. Facebook e Instagram, por exemplo, inferem a localização atual das pessoas usando um sensor presente em seus dispositivos (GPS) e permite que essa informação seja postada.

O *GreenGPS* [8] permite que os usuários coletem dados enquanto estão se locomovendo nos seus veículos. O serviço utiliza os dados coletados para recomendar rotas nas quais os motoristas economizarão combustível. Reddy et al. [20] propõem algo similar, mas com o objetivo coletar dados para identificar rotas mais seguras para pessoas que andam de bicicleta.

O trabalho de Ludwig et al. [15] propõe um monitor que auxilia o trabalho de serviços de emergência em casos de desastres naturais, como tufões e inundações. A coleta de informações pode ser feita de forma participativa por parte de voluntários, ou a partir de redes sociais. Outro exemplo é o *ExposureSense* [19], que monitora as atividades diárias das pessoas usando sensoriamento participativo para estimar o nível de exposição à poluição que aqueles indivíduos estão sujeitos no seu dia-a-dia.

Por fim, no trabalho de Zhou et al. [27], usuários de uma comunidade possuem um objetivo comum de saber o horário de chegada dos ônibus em determinado ponto. Para isso, eles usam seus telefones celulares para coletar informações enquanto se locomovem e assim obter informações que ajudem a realizar as predições.

Todas esses trabalhos provém um serviço para o benefício de uma comunidade de usuários que compartilham de um objetivo comum, que é um importante pilar da ideia proposta neste documento. Entretanto, todo dado coletado pelos usuários e usado para prover o serviço para comunidade é enviado para ser processado em uma infraestrutura de nuvem remota. A solução proposta neste documento visa dar o poder para que os usuários coletem, processem e governem seus dados e a aplicação, sem a necessidade da presença de um provedor de aplicação de nuvem.

2.2.2 Computação na Borda/Névoa

Por muitas vezes os conceitos de Computação na Borda e Computação na Névoa se misturam e são genéricos até certo ponto. Em tese, qualquer infraestrutura de processamento localizada

próxima das fontes geradores de dados, e não na nuvem, está inserida nessa definição.

O trabalho de Satyanarayanan et al. [23] propõe a implantação de datacenters de pequena escala (chamados *cloudlets*) próximos às fontes geradoras de dados, aproveitando a tecnologia de máquinas virtuais. Dessa forma, os usuários podem acessar o serviço através de uma rede local e de forma mais rápida do que se fossem se comunicar com a nuvem de fato.

Para evitar um intenso tráfego desnecessário de dados até a nuvem, Aazam e Huh [1] propõem que os dados envolvidos na aplicação possam ser pré-processados em roteadores inteligentes no meio do caminho.

Luan et al. [14] citam em seu trabalho outras possibilidades onde o conceito de Computação na Borda pode ser aplicado, como por exemplo, a implantação de servidores de processamento em locais públicos como shoppings e estações de ônibus.

Esses são alguns exemplos de tecnologias citadas na literatura que são implantadas próximas às fontes de dados para executar tarefas computacionais. A ideia proposta neste trabalho lança mão de uma versão "mais pura" do paradigma de Computação na Borda, na qual o processamento é feito nos próprios dispositivos que são as fontes dos dados, como por exemplo os próprios celulares dos usuários.

2.2.3 Aplicações Crowdsensing e Computação na Borda

O trabalho de Marjanović et al. [17] propõe uma arquitetura hierárquica para serviços que usam sensoriamento participativo. Nesse caso, os dados coletados pelos usuários podem ser pré-processados em servidores instalados entre os usuários e a nuvem, gerando modelos intermediários que depois vão para nuvem. A ideia de Zhou et al. [28] é similar, mas os servidores intermediários são usados para fazer uma filtragem preliminar dos dados, eliminando possíveis dados maliciosos ou irrelevantes, e assim diminuindo a carga de dados que é enviada à nuvem para ser processada. Ambos os trabalhos usam o paradigma de Computação na Borda por meio da implantação de servidores entre os usuários e a nuvem, ou seja, há a dependência de uma entidade que centraliza a governança do serviço.

Bellavista et al. [3] também combinam as duas características no serviço proposto, mas o foco é a formação de uma rede *ad-hoc*[1] de maneira oportunística com os dispositivos dos

[1] Algo que é elaborado sob demanda para um propósito específico.

usuários da comunidade. A aplicação monitora regiões e detecta pontos cuja concentração de pessoas é suficiente para formar uma rede que será usada para processar tarefas.

Kundig et al. [12] sugerem uma arquitetura dirigida pela comunidade que reúne os dispositivos dentro de uma zona de proximidade local para formar um ambiente de Computação na Borda colaborativa em uma topologia *mesh*.

Apesar de usarem os próprios dispositivos dos usuários para fazer o processamento de tarefas, ambos os trabalhos não abordam a coleta e o compartilhamento de dados entre esses usuários, que possuem algum objetivo comum a ser alcançado.

2.3 Considerações

Os Serviços Governados pela Comunidade assumem que os usuários se encontram em algum ponto e compartilham dados coletados com quem eles confiam, formando uma Rede Social Móvel vinculada a uma arquitetura P2P, que permite que os dispositivos dos usuários sejam tanto consumidores de dados (clientes) quanto provedores de dados (servidores).

Em relação à coleta de dados usando sensoriamento participativo, a maioria dos trabalhos considera que esses dados são enviados para a nuvem para serem processados. Além disso, os trabalhos que consideram o uso do paradigma de Computação na Borda para tal fim, geralmente apontam para uma infraestrutura implantada entre o usuário e a nuvem, cujo objetivo é promover algum pré-processamento do dado, antes do mesmo ser enviado à nuvem. Nesse caso, ainda há uma necessidade de governança externa para agregar os dados coletados pelos indivíduos.

Os trabalhos que consideram o uso dos próprios dispositivos dos usuários para processar tarefas focam na formação de uma rede *ad-hoc* com esses dispositivos e em explicar como as tarefas de processamento podem ser distribuídas entre esses dispositivos que formam a rede. Portanto, não discutem em qual momento os usuários da comunidade coletam dados do ambiente e compartilham entre si, muito menos como isso pode ser feito.

Para verificar a viabilidade e a eficácia dos serviços governados pela comunidade na borda da rede, foi realizado um estudo de caso baseado em simulação, alimentado com dados reais (melhor descrito na Seção 4.1). Esta aplicação tem como objetivo estimar os horários de partidas de ônibus urbanos usando dados coletados pelos usuários no passado.

Uma aplicação similar é apresentada no trabalho de Zhou et al. [27] (Seção 2.2.1). Além de informações do passado, o modelo dos autores também utiliza informações em tempo real. No entanto, a aplicação definida usa uma nuvem remota para processar os dados, enquanto a proposta deste trabalho é baseada puramente nos princípios da Computação na Borda. Daqui em diante, o termo Computação na Borda refere-se à sua forma mais pura citada anteriormente, na qual o processamento é feito nos próprios dispositivos dos usuários.

Capítulo 3

Serviços Governados pela Comunidade

3.1 Definição

Tipicamente, aplicações que dependem do sensoriamento participativo dos usuários fazem uso do paradigma da Computação na Nuvem, onde os dados coletados na borda da rede são armazenados e processados com algum propósito, como por exemplo para fazer predições usando técnicas de IA.

Esse modelo de serviço expõe algumas limitações, entre as quais a necessidade de governança externa. Dessa forma, há uma total dependência de um provedor de aplicação de nuvem centralizado, que pode simplesmente parar de dar suporte ao serviço e que toma as decisões relacionadas ao uso dos dados. Os usuários que contribuem para o serviço provido, coletando dados usando seus dispositivos móveis, possuem pouco controle sobre como esses dados, muitas vezes sensíveis, são armazenados e usados na nuvem.

Os *Serviços Governados pela Comunidade* têm por objetivo tirar a governança do serviço oferecido de um provedor de aplicação de nuvem e concedê-la aos próprios usuários do serviço. Dessa forma, os indivíduos têm um maior controle sobre a utilização dos seus dados coletados, armazenando-os localmente e compartilhando apenas com quem eles confiam. Além disso, o processamento pode ser feito na borda da rede, usando os próprios dispositivos que coletam as informações do ambiente, evitando a necessidade de patrocínio para manter a aplicação executando nuvem. Para garantir essa governança aos usuários, o serviço proposto possui alguns pilares importantes:

- **Sensoriamento Participativo:** Coleta de dados do ambiente usando sensores instalados em smartphones, relógios, pulseiras, entre outros dispositivos que estão na borda da rede. Para garantir o engajamento dos indivíduos, é importante que eles possuam algum **objetivo comum** a ser satisfeito pelo serviço como forma de incentivo.

- **Compartilhamento de dados:** Os usuários do serviço podem se comunicar entre si usando uma **MSN** e, portanto, podem trocar os dados coletados individualmente com seus pares confiáveis. Aproveitando o **padrão de mobilidade** dos indivíduos, isso pode ser feito em algum momento no qual eles estão em uma certa zona de proximidade.

- **Processamento na borda da rede**: Utilizando o paradigma de Computação na Borda, os dados coletados são processados nos próprios dispositivos móveis dos usuários.

Assim, uma comunidade de usuários possui um objetivo comum e aproveitará o poder dos seus próprios dispositivos móveis para coletar e processar os dados coletados. Entre a coleta e o processamento dos dados, os usuários podem compartilhar seus dados com usuários confiáveis da mesma comunidade.

Já que tudo é feito na borda da rede, a necessidade de um servidor central executando na nuvem é evitada. A remoção do ponto único de falha representado pelo servidor executando na nuvem aumenta a robustez do serviço, elimina o gargalo na comunicação e permite que os usuários definam e gerenciem conjuntamente a governança do serviço, mitigando também preocupações com a privacidade dos dados coletados pelos mesmos.

3.2 Modelo do sistema

O sistema é composto por vários dispositivos pessoais executando o serviço comunitário de análise de dados. Os usuários utilizam um agente da aplicação instalada em seus dispositivos para coletar e compartilhar dados de maneira participativa, além de consultar o serviço de fato. O agente do serviço que é executado no dispositivo móvel de cada usuário é ilustrado na Figura 3.1. Ele consiste de seis módulos: *participatory sensor*, *community sensor*, *community data collector*, *community data filter*, *model builder* e *query dispatcher*.

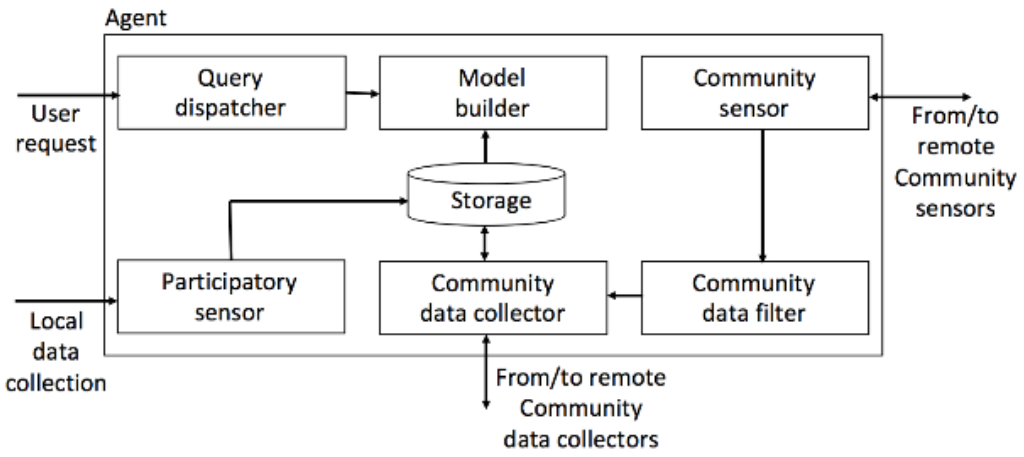

Figura 3.1: Componentes do agente

- **Participatory sensor**: responsável por coletar dados do ambiente nas proximidades do dispositivo. Isso pode ser feito usando as próprias funcionalidades já existentes nos smartphones atuais, como GPS, acelerômetros e microfones. No estudo de caso descrito neste trabalho, o GPS tem um papel fundamental para coleta de locais e horários.

- **Community sensor**: trata de descobrir outros membros da comunidade que estão próximos do dispositivo. Assim, se dois membros da comunidade de usuários do serviço estão em uma zona próxima, o dispositivo de um consegue detectar a presença do outro como um membro ativo da comunidade.

- **Community data collector**: contacta outros membros da comunidade com o objetivo de aumentar a quantidade de dados disponível localmente. Após detectar a presença de um usuário próximo usando o *community sensor*, o indivíduo pode contactá-lo para obter os dados coletados por ele até o momento. Da mesma forma, esse mesmo indivíduo pode ser contactado por outro membro da comunidade que deseja obter seus dados. A permissão ou não para prover dados a alguém ou receber dados de alguém é controlada pelo componente seguinte, o *community data filter*. O *community sensor* e o *community data collector* são os únicos componentes envolvidos durante a comunicação com outros usuários da comunidade.

- **Community data filter**: regula que dados podem ser compartilhados com outros membros da comunidade, em ambas as direções, isto é, para quem o dado local do

dispositivo pode ser compartilhado, e de quem dados podem ser requisitados. Essa atividade depende do nível de confiança dos usuários para com os outros membros da comunidade. Sendo assim, o indivíduo pode limitar o compartilhamento de sua informação com as pessoas nas quais ele confia, e limitar os provedores de dados entre as pessoas nas quais ele confia.

- **Model builder**: responsável por criar o modelo do serviço que responderá as consultas dos usuários a partir de todos os dados disponíveis localmente. Como já dito anteriormente, os usuários coletam e compartilham dados com algum propósito. Esse modelo vai se apropriar das informações disponíveis localmente, coletadas pelo próprio usuário ou obtidas de outros usuários da comunidade no passado, para responder alguma requisição, a depender do propósito do serviço oferecido.

- **Query dispatcher**: provê a interface do serviço para o usuário, na qual ele requisita alguma resposta que será gerada a partir de inferências realizadas pelo modelo construído pelo *Model builder*.

Para exemplificar como esses componentes interagem, considere dois membros A e B da comunidade. Ambos coletam informações do ambiente usando o *participatory sensor* e as armazenam localmente no *storage* de seus respectivos dispositivos. Quando A e B estão em algum lugar em comum, o *community sensor* de A detecta a presença de B e vice-versa. O *community data collector* de A se comunica com o *community data collector* de B para obter os dados que B coletou até o momento. Durante esse processo, o *community data filter* de A verifica se B é um usuário confiável para se obter aquelas informações e o *community data filter* de B verifica se A é um usuário confiável para que ele possa prover sua informação coletada. Passando pelo filtro com sucesso, A armazena no *storage* de seu dispositivo as informações coletadas por B no passado. Eventualmente, o usuário A faz uma requisição ao serviço usando o *query dispatcher*. Nesse momento, o modelo criado pelo *model builder* utiliza os dados armazenados no *storage* de A, que nesse momento reúne os dados coletados pelo próprio usuário A mais os dados obtidos de B, para gerar a resposta da requisição.

3.3 Requisitos da aplicação

Os dispositivos pessoais dos usuários são aptos a coletar dados do ambiente de forma passiva ou ativa. Além disso, esses dispositivos se conectam em algum momento. Essa conexão pode ser através da formação de uma rede local comum ou mesmo de uma conexão à Internet. A tecnologia *bluetooth* é um exemplo de comunicação sem fio simples que permite que dispositivos próximos se comuniquem. Entretanto, a limitação de distância pode ser um problema. O trabalho de Zhuang et al. [29] trata do design e implementação de um *middleware* com uma abordagem baseada em WiFi, permitindo que os usuários se comuniquem com dispositivos próximos através de distâncias maiores e com uma largura de banda bem maior comparado ao *bluetooth*. Essa solução utiliza o protocolo AODV (do inglês, *Ad hoc On-demand Distance Vector*) e é suficiente para aplicações comuns, como compartilhamento de fotos e jogos com múltiplos jogadores. Esse é um exemplo de tecnologia que pode ser aplicada na arquitetura descrita na seção anterior.

Os usuários compartilham de algum interesse comum por algum serviço e formam uma comunidade. A criação da comunidade permite que os usuários compartilhem os dados coletados com seus pares confiáveis (sensoriamento participativo). Esse tipo de serviço envolve análise de dados através de modelos analíticos ou de aprendizado de máquina, e espera-se que a qualidade das respostas providas pelo modelo seja proporcional à quantidade e diversidade dos dados coletados e utilizados. Assim, quanto mais dados estiverem disponíveis para os usuários, mais benefícios eles poderão obter com o serviço.

O uso dos dispositivos de borda para executar o serviço comunitário e a coleta dados pode ser um limitante para essas atividades. Em outras palavras, a execução do serviço e a tarefa de sensoriamento devem ser leves e, idealmente, o consumo da bateria ocasionado por essas atividades deve ser aceitável para o usuário. Além disso, o consumo com armazenamento de dados também deve ser baixo. Existem várias maneiras de mitigar o impacto dessas limitações. Por exemplo, o treinamento de modelos de aprendizado de máquina, que tende a ser um procedimento intensivo de computação, pode ser executado apenas quando o dispositivo estiver totalmente carregado e conectado ao carregador. Como alternativa, modelos preditivos mais simples podem ser usados para reduzir a demanda de computação (o estudo de caso mostra um exemplo). Em relação ao armazenamento de dados, é possível

usar uma abordagem de janela deslizante para descartar dados antigos que não serão mais úteis para o modelo preditivo.

Capítulo 4

Estudo de Caso: Sistema de transporte público

4.1 Descrição

A fim de avaliar a viabilidade do modelo dos Serviços Governados pela Comunidade em termos de quantidade de dados obtidos e qualidade do serviço provido, foi escolhido um estudo de caso que se encaixa nas características da aplicação discutidas acima (na Seção 3.3). Ele foi apoiado por uma base de dados reais disponível, que foi gerada a partir de dados do transporte público de ônibus da cidade de Curitiba/PR.

A partir da formação de uma comunidade, estudantes universitários podem tirar proveito de seus padrões de mobilidade coletiva e dos sensores presentes em seus smartphones, como o GPS, com objetivo de prover dados para um serviço que venha a atender as necessidades dessa comunidade, como por exemplo, saber os horários de partidas dos ônibus que saem das proximidades do campus no qual estudam.

Em muitas cidades, os horários dos ônibus urbanos são divulgados e seguidos de maneira muito rigorosa. Nas cidades que usam a tecnologia de maneira intensa, os ônibus podem ser equipados com sensores e rastreados em tempo real usando um serviço baseado em nuvem, ou mesmo usando soluções baseadas em 5G [13], para que atrasos imprevistos possam ser identificados e rotas de ônibus alternativas possam ser escolhidas. No entanto, em muitos lugares, especialmente nas grandes cidades dos países em desenvolvimento, pode ser difícil saber a hora exata em que um ônibus sairá de um ponto de ônibus. Nesses locais, os horá-

rios fornecidos pelas empresas de ônibus raramente são seguidos, devido a vários motivos, incluindo engarrafamentos, manutenção imprevista, ou até falta de organização por parte das empresas. O desconhecimento do horário real de partida do ônibus pode aumentar o tempo de espera no ponto de ônibus, levando à perda de tempo e, em situações mais graves, pode tornar os passageiros suscetíveis à violência urbana. Infelizmente, assaltos em pontos de ônibus são situações rotineiras no Brasil[1,2]. O estudo de caso foca nesses lugares que não conseguem prover uma tabela de horários cumprida regularmente, e em uma comunidade específica de usuários de ônibus: estudantes de um grande campus universitário.

Um grande número de estudantes universitários usa o transporte público todos os dias da semana para ir de casa para a universidade e voltar. Esses alunos compartilham um interesse comum em relação aos horários de saída de ônibus. Por exemplo, todos gostariam de saber o horário exato que seu ônibus vai sair do ponto. Além disso, muitos deles moram no mesmo bairro e usam as mesmas linhas de ônibus. Ao formar uma comunidade, eles podem tirar proveito de seu padrão de mobilidade coletiva, que pode ser explorado por um serviço governado pela comunidade.

Sempre que saem da universidade em um ônibus, os estudantes podem coletar informações sobre qual linha de ônibus foi usada e a que horas o ônibus partiu da universidade. Quando os alunos retornam à universidade, todas as informações que eles coletaram anteriormente estão disponíveis em seus dispositivos. Portanto, nesse cenário, os alunos que estão online no campus ao mesmo tempo podem compartilhar seus dados coletados com quem eles confiam, seguindo o modelo descrito na Seção 3.2. Esses dados coletados e compartilhados são então processados e analisados para satisfazer demandas comuns desta comunidade. Como discutido anteriormente, esse processamento pode ser realizado localmente sem a necessidade de envio de dados para nuvem.

Com isso em mente, o objetivo comum dos usuários neste estudo de caso é estimar o horário de partida de ônibus de uma linha específica próximo às paradas de ônibus de um campus universitário, usando apenas dados de viagens anteriores coletados pela comunidade universitária que usa esse meio de transporte. A ideia é então avaliar como o modelo

[1]http://diariogaucho.clicrbs.com.br/rs/policia/noticia/2019/06/passageiros-relatam-que-paradas-de-onibus-viraram-pontos-de-assalto-em-porto-alegre-10944083.html

[2]https://www.folhavitoria.com.br/policia/noticia/06/2020/criminoso-faz-arrastao-e-rouba-passageiros-que-estavam-em-pontos-de-onibus-em-vila-velha

de serviço proposto neste trabalho se comporta em comparação com outros cenários, como um serviço típico que utiliza a nuvem como *back-end*, em relação à quantidade de dados agregada para responder as requisições dos usuários. Além disso, foi ilustrado um modelo simples de predição que utiliza os dados agregados, permitindo também a avaliação da qualidade do serviço oferecido.

4.2 Materiais e Métodos

4.2.1 Dados

Foram utilizados dados de transporte público da cidade de Curitiba para executar o estudo de caso. Uma breve caracterização dos dados e como eles são coletados pode ser vista no trabalho de Braz [7]. Seu estudo utiliza dados de programação dos ônibus, dados brutos de GPS e dados do cartão de embarque para reconstruir viagens de ônibus a nível de passageiro, a partir dos dados anonimizados originais providos pela agência municipal de planejamento urbano da cidade de Curitiba (URBS). No rastro reconstruído, cada registro representa uma viagem feita por um usuário. Desse registro, é possível obter a parada da qual o ônibus partiu, o horário de saída e a linha associada àquele ônibus. Um exemplo de um evento pode ser o de um usuário com id 123456 que deixou o ponto de ônibus às 8h no dia 12 de maio usando a linha 500.[3]

A partir do mapa da cidade, foram selecionadas apenas as paradas de ônibus nas proximidades da universidade alvo, no caso deste trabalho a Universidade Tecnológica Federal do Paraná (UTFPR), e foram filtradas apenas as viagens que têm como origem ou destino aqueles pontos de ônibus selecionados. Essas viagens foram então enfileiradas representando um conjunto de eventos de chegadas e saídas de usuários no campus em ordem cronológica.

Além disso, algumas adaptações aos rastros e suposições tiveram que ser feitas, para que de alguma forma, pudéssemos estabelecer os períodos em que um usuário estava no campus. Eles são sinalizados pelos eventos de chegada e saída nos rastros e para cada dia e cada usuário, podem existir zero ou mais desses eventos. Quando um evento de chegada é seguido por um evento de partida no mesmo dia, assume-se que o usuário permaneceu no

[3]Existem mais atributos em cada viagem, mas foram citados apenas aqueles necessários para implementar o estudo de caso.

campus pelo período compreendido entre o horário de chegada e o horário de partida. No entanto, se apenas um evento de partida estiver presente, sem um evento de chegada anterior no mesmo dia, arbitra-se que o usuário chegou ao campus uma hora antes do horário do evento de partida. Da mesma forma, se um evento de chegada estiver presente sem um evento de partida posterior, arbitra-se que o usuário permaneceu no campus por uma hora desde sua chegada. Essa falta de informação sobre chegada ou saída de um usuário em um determinado dia pode acontecer tanto por algum problema na coleta dos dados originais, ou mesmo devido às situações comuns do dia-a-dia, como o caso de um estudante que chegou ao campus usando ônibus, mas partiu usando outro meio de transporte.

Feitas as adaptações, o rastro que alimenta o simulador possui 79.907 eventos de chegada (29.561) ou saída (45.346) em paradas de ônibus próximas da universidade entre maio de 2017 e julho de 2017 de 18.662 usuários diferentes.

4.2.2 Modelo de simulação

Todo evento saída do campus que aparece no rastro considerado pelo simulador gera uma requisição ao serviço. Seja t_a o horário no qual o evento de saída foi registrado no rastro. Assume-se que em algum horário t_r, antes ou igual a t_a, o usuário queria saber o horário estimado que ele(a) deveria deixar o campus e ir até o ponto de ônibus se prefere pegar uma linha específica de ônibus. Em outras palavras, antes de sair do campus no tempo t_a, o usuário pergunta ao serviço: "Em que horário eu devo ir ao ponto de ônibus para pegar a linha de ônibus b de tal forma que eu espere o menor tempo possível na parada?"Arbitrou-se o horário t_r como sendo o tempo no qual a solicitação foi emitada para o serviço, que é sorteado a partir de um intervalo de tempo que se inicia no máximo entre o último evento de chegada do usuário t_c e uma hora antes do horário real da saída t_a, e vai até o horário real da saída que está registrado nos dados t_a, seguindo uma distribuição uniforme.

$$t_r = U(max(t_c, t_a - 1h), t_a)$$

Por exemplo, se o usuário possui um evento de chegada às 12h e um evento de saída às 18h, o horário da requisição estará entre 17h e 18h. Entretanto, se o usuário possui um evento de chegada às 17h20 e um evento de saída às 18h, o horário da requisição deverá estar

entre 17h20 e 18h. Uma requisição feita no horário t_r solicitando a hora estimada (t_e) para ir ao ponto de ônibus nas proximidades do campus para obter o próximo ônibus da linha b é denotada por $R^b_{t_r}$.

Já que t_a não é conhecido pelo serviço, dada uma requisição $R^b_{t_r}$, é usado um algoritmo de predição que é alimentado com dados de viagens passadas disponíveis. O objetivo é estimar o horário mais apropriado para o usuário dirigir-se ao ponto de ônibus para pegar o próximo ônibus da linha b. O algoritmo de predição almeja minimizar o tempo de espera na parada de ônibus. O objetivo do trabalho não é prover a melhor solução para esse problema, mas sim entender como a quantidade de dados disponíveis impacta na performance dessa solução em particular. Sendo assim, foi escolhido um algoritmo extremamente simples que está alinhado com os requisitos necessários da aplicação discutidos na Seção 3.3 e que simplesmente recomenda o menor horário de saída da linha de ônibus contido nos dados do passado disponíveis (para qualquer dia da semana) entre t_r, quando a requisição foi feita, e a próxima hora. Assim, se um usuário faz uma requisição às 8h05, o algoritmo utilizará todos os dados históricos disponíveis de viagens que ocorreram entre 8h05 e 9h05 em qualquer dia do passado, e recomendar o menor horário de saída como sendo o mais apropriado. Para simplificar, não é levado em consideração o tempo que o usuário necessita para caminhar até a parada de ônibus.

A quantidade de dados históricos disponíveis para o algoritmo de predição depende de como os usuários do serviço se comportam. São consideradas diferentes configurações para isso. Em particular, considera-se casos nos quais os usuários não compartilham dados, nem entre eles e nem com servidores centralizados, casos nos quais os dados tornam-se disponíveis em servidores centralizados em diferentes instantes de tempo, e casos nos quais os dados são trocados entre os usuários que estão no campus ao mesmo tempo. Como discutido antes, o dado armazenado no dispositivo do usuário, coletado diretamente por ele ou recebido de outros usuários, pode ser mantido apenas por algum tempo. O rastro utilizado possui uma duração de apenas três meses, logo, nas simulações realizadas, considera-se que todo dado coletado nos dispositivos é mantido até o fim da simulação. A seguir estão apresentadas as configurações de compartilhamento dos dados que foram avaliadas nessa prova de conceito.

4.2.3 Configurações

- *Baseline*: A configuração baseline é a mais conservadora e ingênua possível. Ela não utiliza nenhum dado histórico para estimar o horário para ir até a parada de ônibus. Ela simplesmente sugere o tempo da requisição (t_r) como sendo tal horário.

- *Offline*: Nessa configuração, o modelo é construído usando apenas viagens coletadas pelo próprio usuário que faz a requisição, representando uma situação na qual os usuários nunca compartilham seus dados com outros membros da comunidade.

- *Cloud-based*: Aqui, todo o dado coletado torna-se disponível no momento da coleta, já que nesse caso a informação é enviada para uma nuvem centralizada. Toda informação disponível no servidor pode ser utilizada pelo algoritmo de predição usado para responder as requisições dos usuários.

- *Cloudlet-based*: Nessa configuração considera-se a existência de um servidor local no campus universitário que é acessível apenas quando o usuário está no campus. O dado torna-se disponível para esse servidor sempre que um usuário chega ao campus com novas informações coletadas desde sua última saída (e não no momento exato da coleta do dado, como na configuração anterior).

- **Community-governed**: Nesse caso, um usuário u que está no campus no tempo t vai compartilhar seus dados com um usuário u', desde que u' também esteja no campus no tempo t, u esteja disposto a compartilhar dados com u', e u' confie em u como um provedor de dados. Nesse caso, todo dado que u coletou até aquele ponto do tempo (diretamente ou indiretamente) estará disponível para u'. Todo dado disponível nos dispositivos do usuário podem ser considerados pelo algoritmo de predição usado para responder requisições.

Claramente, a quantidade de dados usada para responder uma requisição na configuração offline, exceto por casos extremos incomuns, é normalmente menor que a quantidade de dados usada na configuração community-governed, que por sua vez, é menor que a quantidade usada nas configurações cloudlet e cloud. O foco é avaliar o quão viável é usar apenas os dados disponíveis na configuração community-governed e também comparar a acurácia das estimativas feitas usando esses diferentes níveis de informações disponíveis. Além

disso, foram considerados diferentes níveis confiança entre os usuários da comunidade para a configuração community, o que leva a diferentes quantidades de dados disponíveis para a realização de predições, conforme descrito a seguir.

4.2.4 Definindo relação de confiança entre usuários

Inicialmente, considera-se que todos os usuários confiam uns nos outros. Portanto, se dois membros da comunidade estiverem no campus ao mesmo tempo, eles poderão trocar seus dados. Isso permite avaliar o melhor desempenho que a configuração community pode oferecer. Em seguida, considera-se o caso em que todos os usuários no rastro têm uma lista com o mesmo número m de outros membros da comunidade que desejam compartilhar dados com eles e avaliamos o desempenho do algoritmo, já que o valor de m varia de 10 a 80 (refere-se a essas configurações como community-m, $m \in \{10, 20, 30, 40, 50, 60, 70, 80\}$). Os m provedores de dados de um usuário u são escolhidos aleatoriamente a partir de todos os outros membros da comunidade. Claramente, essa suposição não é realista, mas ajuda a entender como o desempenho do algoritmo diminui à medida que a quantidade de informações disponíveis diminui. Por último, foram consideradas relações de confiança a partir de redes sociais reais para arbitrar quais membros da comunidade confiam uns nos outros e avaliar o desempenho do serviço governado pela comunidade em um ambiente mais realista.

Para o último caso descrito acima, adotou-se a seguinte abordagem. Foram extraídas 100 redes sociais do Facebook que conectam estudantes matriculados em universidades nos Estados Unidos da América [25]. As redes sociais foram agrupadas considerando todos os atributos vindos do grafo original e após isso foi feita a seleção de um representante de cada grupo. Em seguida, para cada grafo das redes sociais consideradas, foi escolhido aleatoriamente um membro do grafo e iniciada uma busca em largura (BFS, do inglês *Breadth First Search*) a partir desse nó, até que o número de nós percorridos no grafo fosse igual ao número de usuários do rastro do transporte público. Nesse ponto, é criado um novo grafo da rede social contendo apenas os nós que foram percorridos e as conexões que esses nós tinham com os outros nós. Por fim, cada usuário do rastro do transporte público é mapeado aleatoriamente para um nó diferente do grafo da rede social. O modelo assume que dois usuários que estavam conectados no grafo confiam um no outro.

O algoritmo *k-means* [11] foi usado para agrupar as 100 redes sociais. Os métodos da

silhueta [21] e da soma dos quadrados dentro do agrupamento (WSS, do inglês *Within-cluster Sum of Squares*) [10] foram usados para definir um valor para o número de grupos k (ver Figura 4.1 e Figura 4.2). Baseado nesses resultados, foi escolhido um $k = 7$.

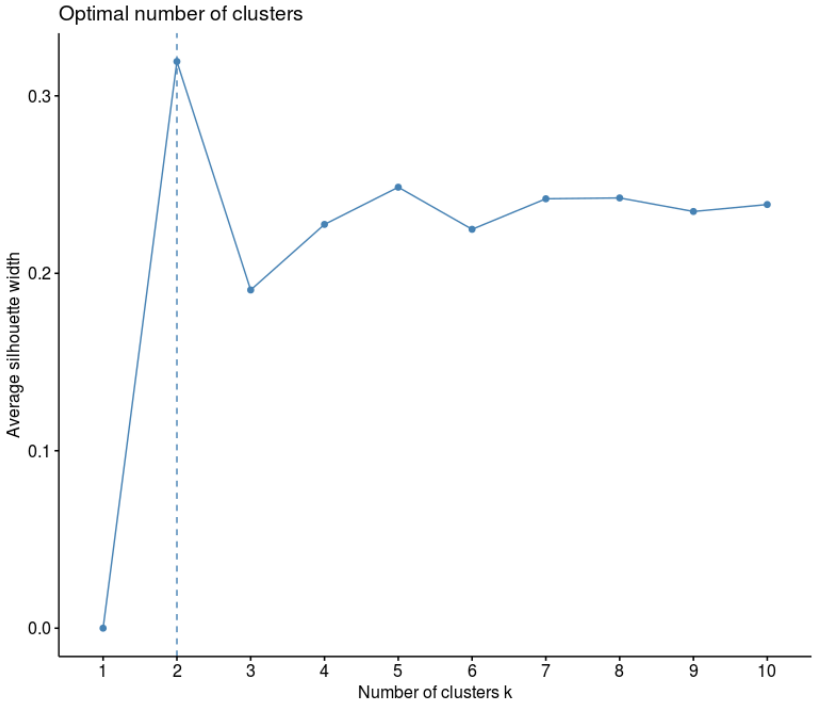

Figura 4.1: Gráfico gerado pelo método da silhueta

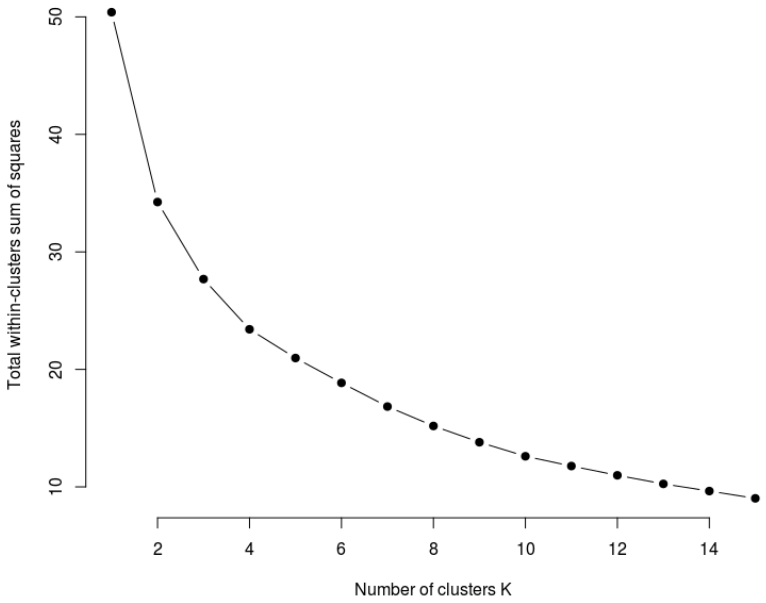

Figura 4.2: Gráfico gerado pelo método WSS

Após agrupar as 100 redes sociais em 7 grupos, foi escolhida a rede mais próxima do centroide de cada grupo como sendo a rede que representa aquele grupo (refere-se a essas configurações como community-real-i, $i \in \{1, 2, 3, 4, 5, 6, 7\}$).

4.2.5 Design experimental

Os diferentes usuários do rastro possuem perfis diferentes. Em particular, a maioria dos usuários possui apenas um evento de partida e nenhum evento de chegada. Nesse caso, eles nunca voltam ao campus com a sua única viagem coletada para compartilhar com os outros membros da comunidade. Dessa forma, esses usuários compartilham essa informação coletada apenas quando a configuração cloud-based é usada. Além disso, a distribuição das "viagens de retorno" realizadas por diferentes usuários é bastante enviesada, como pode ser visto na Figura 4.3. "Viagens de retorno" são eventos de chegada do usuário no campus nos quais ele traz consigo informações sobre alguma viagem realizada no passado. Muitos usuários possuem apenas uma ou duas viagens que são coletadas e que podem ser compartilhadas com outros membros no futuro, mas existem usuários com até 30 viagens.

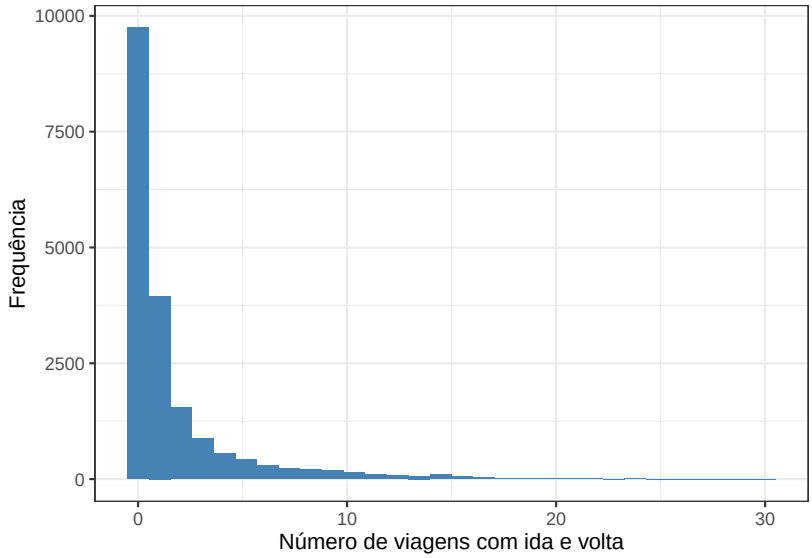

Figura 4.3: Distribuição do número de "viagens de retorno" em todo rastro

Por causa dessa assimetria nos dados, além de considerar todo rastro que foi gerado a partir dos dados originais, foram considerados outros dois subconjuntos do mesmo: i) um que descarta todos os usuários que não possuem viagens de retorno; e ii) um que considera apenas os usuários mais ativos da comunidade. Para o primeiro caso, isso implica retirar do rastro os usuários que nunca contribuíram para comunidade, ou seja, eles até podem ter feito alguma viagem partindo do campus e coletado essa informação, mas nunca retornaram no futuro para ter a oportunidade de compartilhá-la com outros membros. Para o último caso, foi computado o 90-percentil do total de viagens de retorno de cada usuário (entre os que possuíam alguma viagem), o que levou o experimento a considerar apenas os usuários que tornaram disponíveis no mínimo 9 viagens, ou seja, partiram do campus, coletaram a informação e as compartilharam em algum momento no futuro quando voltaram no mínimo 9 vezes. O resumo dos rastros considerados é apresentado na Tabela 4.1.

Foram executados três experimentos de simulação que diferem nos valores usados em dois fatores: os rastros e as configurações de compartilhamento de dados apresentadas na Seção 4.2.3. No primeiro experimento, assume-se que todos os usuários confiam uns no outros, ou seja, não há restrições para o compartilhamento de dados entre eles. Esse representa o melhor caso possível da configuração community em termos de quantidade de informações que podem ser compartilhadas e agregadas. Os outros dois experimentos consideram a existência de redes sociais (sintéticas ou reais), que limitam o compartilhamento de da-

Tabela 4.1: Resumo dos rastros considerados nos experimentos

Rastro	**Descrição**	**Usuários**
Completo	Considera todos os usuários do rastro original	18 662
Colaboradores	Considera os usuários que fizeram ao menos uma "viagem de retorno"	8 918
Ativos	Considera os usuários que fizeram ao menos 9 “viagens de retorno”	845

dos entre usuários, conforme discutido na Seção 4.2.4. As instruções para reprodução dos experimentos estão disponíveis no Apêndice A.

Somente o primeiro experimento considera os três rastros descritos, pois os resultados desse experimento mostram que esse fator tem um impacto mínimo nos resultados. Nos outros dois experimentos, foi usado apenas o rastro com os usuários mais ativos, pois esses são os usuários que mais se beneficiariam com o serviço. Os valores considerados para os fatores em cada experimento estão resumidos na Tabela 4.2.

Tabela 4.2: Resumo do design experimental

Experimento	Rastro	Configuração de compartilhamento dos dados
I	Completo Colaboradores Ativos	Baseline Offline Cloudlet Cloud Community (todos os usuários confiam uns nos outros)
II	Ativos	Community-m (todos os usuários confiam em m outros usuários)
III	Ativos	Community-real-i (confiança baseada no grafo de 7 redes sociais reais)

As configurações community no experimento II e no experimento III têm um compor-

tamento estocástico. No Experimento II, os m usuários que podem fornecer dados a um usuário específico u são sorteados aleatoriamente a partir do rastro. No experimento III, o nó inicial do algoritmo BFS é escolhido aleatoriamente na rede social e, após o novo grafo ser gerado, existe um mapeamento aleatório dos usuários no rastro para os nós do novo grafo (ver Seção 4.2.4). Assim, para essas configurações, a execução dos experimentos foi replicada várias vezes — 50 para o experimento II e 500 para o experimento III.

4.2.6 Métricas

O foco da avaliação do serviço proposto está na quantidade de dados que os usuários podem agregar e que serão utilizados para realizar as predições em cada configuração. Para fazer isso, foram coletados dados dos experimentos de simulação para computar a quantidade de dados históricos utilizados para responder cada requisição, além da proporção de requisições que podem ser respondidas usando algum dado do passado.

Apesar do foco na agregação dos dados, também foi ilustrado um exemplo de um modelo preditivo simples (descrito na Seção 4.2.2) no qual uma maior agregação de dados possibilita a entrega de uma melhor qualidade de serviço. Para avaliar a *QoS*, foram coletadas as métricas tempo de espera na parada e proporção de perda dos ônibus.

Essa relação entre a quantidade de dados obtidos e a qualidade das predições pode variar, dependendo do nicho da aplicação, da característica dos dados coletados e do algoritmo de predição utilizado. Como já dito anteriormente, o objetivo deste trabalho não é encontrar a melhor solução para o problema de estimar horários de partida de ônibus, mas mostrar que o modelo proposto é factível, e avaliar a quantidade de dados que os usuários podem agregar em diferentes configurações de compartilhamento. A seguir, as métricas introduzidas anteriormente são melhor detalhadas:

- Proporção de requisições $R^b_{t_r}$ que podem ser respondidas usando dados do passado (DP): Como descrito antes, quando o usuário quer saber em que horário ele(a) deve ir até o ponto de ônibus, o algoritmo que executa em seu dispositivo utilizará dados de viagens passadas. Entretanto, quando não há dado disponível para aquele intervalo de tempo associado à requisição ($[t_r, t_r + 1h]$), ele utiliza a estratégia baseline que sugere uma saída imediata. Essa métrica visa medir a proporção de requisições nas quais as

predições são feitas baseadas em dados do passado, sem portanto usar a configuração baseline.

- Quantidade de dados usada pela realizar as predições (QD): Essa métrica indica quantas viagens do passado foram usadas para responder uma dada requisição. Como já discutido anteriormente, espera-se que quanto maior for a base de dados disponível, melhor as previsões do modelo. Ela é simplesmente medida em número de viagens.

- Tempo de espera na parada (TEP): Essa métrica mede, em minutos, por quanto tempo o usuário esperou na parada até que o próximo ônibus chegasse. Essa próxima saída t_d não necessariamente é a mesma saída no tempo t_a registrada no rastro e que originou a requisição em um primeiro momento. Isso acontece pois o horário no qual o usuário se dirige até a parada (t_e, estimado pelo algoritmo de predição) pode ser anterior a t_a — nesse caso, o usuário pode pegar um ônibus da mesma linha b que saiu do ponto de ônibus depois de t_e e antes de t_a —, ou posterior a t_a — caso no qual ele não tem garantia que haverá outro ônibus da linha b saindo em um horário após t_a. Para evitar que o usuário espere indefinidamente, assume-se que se um ônibus da linha b não chega até uma hora após t_e, então o usuário abandonou o ponto de ônibus e registra-se que o tempo de espera foi de uma hora.

- Proporção de perda (PP): Essa métrica indica a frequência com a qual os usuários não conseguem pegar o ônibus considerando uma janela de uma hora. Em outras palavras, a porcentagem de requisições $R^b_{t_r}$ tal que não existe ônibus da linha b saindo no tempo t_d, $t_e \leq t_d \leq t_e + 1h$.

As métricas DP e PP foram computadas apenas para o Experimento I e reportadas como o valor computado para a única simulação executada de cada cenário, já que os resultados são determinísticos. As outras duas métricas foram computadas para todos os três experimentos. A métrica QD é reportada como o valor médio para todas as requisições processadas das simulações executadas apenas uma vez. Para os experimentos replicados (que envolvem community-m e community-i), foram computadas as médias de cada execução e reportadas como média das médias juntamente com o intervalo de confiança associado com 95% de confiança. A métrica TEP é reportada de forma similar, mas usando a mediana ao

invés da média. A mediana foi usada como estatística para avaliar o tempo de espera pois a distribuição dos valores não é simétrica e a média pode ser afetada por valores extremos. O Apêndice B reporta as distribuições dos valores de QD e TEP para todas os cenários do primeiro experimento.

4.3 Resultados e Discussões

4.3.1 Análise do melhor caso: sem restrições para o compartilhamento de dados

Tabela 4.3: Resumo de todas as métricas

Rastro	Configuração	QD	DP	PP	TEP
Completo	Baseline	0.0	0.0%	0.008%	5.48 min
	Offline	1.3	39.8%	0.366%	5.13 min
	Community	51.9	92.6%	0.602%	3.65 min
	Cloudlet	53.7	96.0%	0.965%	3.35 min
	Cloud	57.1	97.7%	0.882%	3.30 min
Colaboradores	Baseline	0.0	0.0%	0.005%	5.40 min
	Offline	1.6	50.7%	0.460%	4.81 min
	Community	54.0	94.5%	0.547%	3.33 min
	Cloudlet	55.6	96.7%	0.744%	3.21 min
	Cloud	57.1	97.7%	0.747%	3.33 min
Ativos	Baseline	0.0	0.0%	0.008%	5.36 min
	Offline	3.6	75.8%	0.239%	4.46 min
	Community	53.4	96.0%	0.168%	3.16 min
	Cloudlet	53.9	96.5%	0.186%	3.16 min
	Cloud	54.4	97.0%	0.195%	3.13 min

A Tabela 4.3 mostra os resultados para DP, QD, PP, e TEP em cada configuração. Em geral, a configuração community consegue agregar tantos dados quanto nas configurações cloudlet e cloud, além de conseguir agregar bem mais dados que uma situação na qual não há troca de dados entre os membros da comunidade. Além disso, quanto mais dados disponíveis, melhor se sai o algoritmo de predição.

Agregação de dados

A partir da quantidade de informação usada em cada uma das predições (acumuladas localmente no dispositivo do usuário ou agregadas em um servidor centralizado, a depender da configuração simulada), foi calculado um intervalo de confiança para média com 95% de confiança. Como é possível ver na Figura 4.4, a configuração community apresentou resultados bem próximos das configurações cloudlet e cloud, onde há a existência de um servidor central que agregue os dados. São usados, em média, entre 52 e 57 dados históricos para gerar o modelo que responde as requisições nessas três configurações, considerando todos os rastros, como mostra a Tabela 4.3. Considerando o rastro "Ativos", a diferença da configuração community para as outras duas é ainda menor, e os intervalos de confiança das três configurações se sobrepõem. Além disso, na configuração community consegue-se agregar bem mais dados que na configuração offline, onde não há nenhum tipo de compartilhamento de dados entre os usuários. No rastro "Ativos", por exemplo, são usados, em média, apenas 3 informações do passado para gerar o modelo de predição.

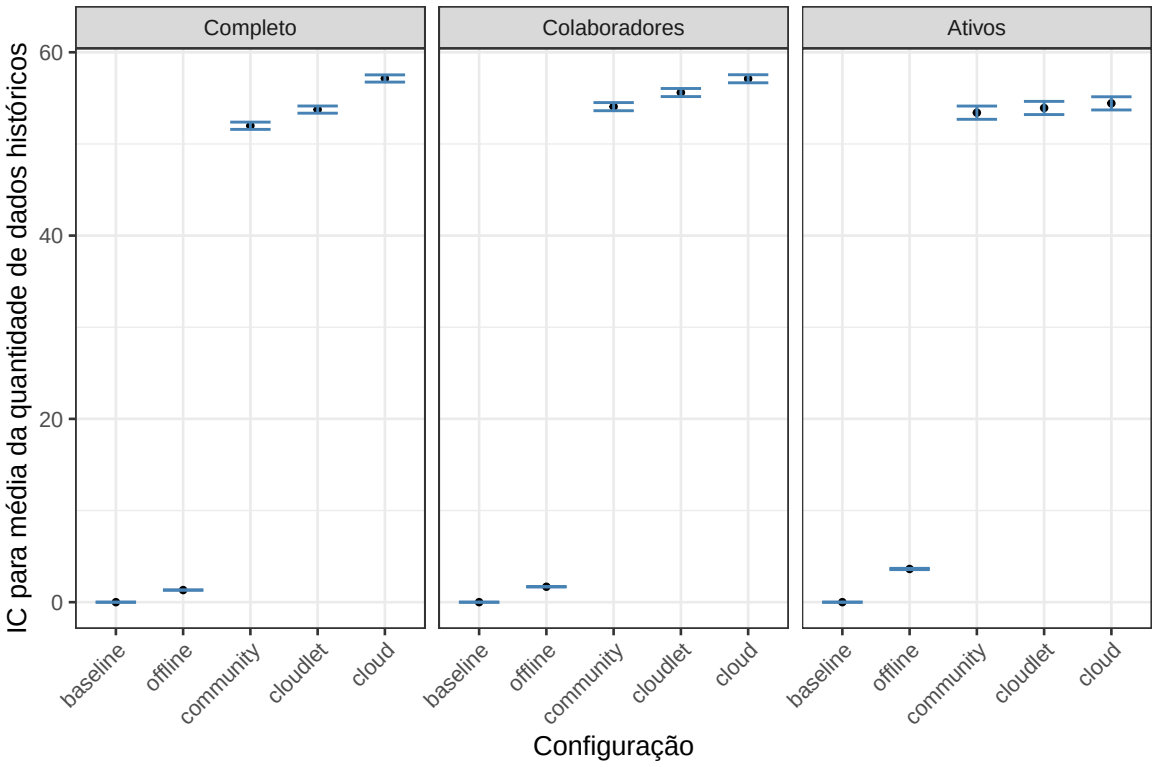

Figura 4.4: Intervalos para média da quantidade de dados com confiança de 95%

A única configuração que é substancialmente afetada pelos diferentes rastros é a offline,

mais especificamente na métrica DP (ver Tabela 4.3). Quando a configuração offline é usada com o rastro completo, apenas 40% das requisições são respondidas baseadas em dados passados coletados pelo usuário (DP). Os outros 60% das requisições recorrem à estratégia baseline devido à falta de dados. Isso acontece pois nesse rastro metade dos usuários fazem apenas uma requisição ao longo do tempo, e não há nenhum dado coletado por eles anteriormente. O valor de DP aumenta para 76% quando a configuração offline é usada com o rastro "Ativos". Isso é esperado, já que nesse rastro todos os usuários viajaram e coletaram dados ao menos 9 vezes como já foi dito anteriormente. Logo, ao fazerem uma requisição ao serviço, é alta a chance de eles já terem coletado alguma informação no passado que está armazenada em seu smartphone localmente e que portanto pode ser utilizada pelo modelo de predição.

Ainda considerando a métrica DP, percebe-se que a diferença entre a configuração comunnity e a configuração cloud-based, que é a melhor configuração possível em relação à quantidade de dados disponível, não é maior que 5.1% no rastro completo (92.6% x 97.7%). No rastro "Ativos", a diferença é ainda menor (1% apenas).

Qualidade das predições

Como discutido anteriormente, cada requisição possui um tempo de espera (TEP) associado e foi calculada a mediana para cada cenário. Quanto menor o tempo de espera, melhor. A Figura 4.5 mostra os intervalos de confiança para as medianas dos tempos de espera em cada configuração de compartilhamento de dados de cada rastro considerado, com um nível de confiança de 95%. Como podemos ver, em cenários em que há mais dados disponíveis, o tempo de espera é menor no geral. A maior evolução ocorre quando passamos da configuração offline para a configuração community. Nas configurações cloudlet e cloud, há uma diminuição nos tempos de espera, mas não é substancial.

O teste Wilcoxon-rank-sum confirma que, em média, existem apenas diferenças estatísticas significativas entre as configurações cujos intervalos de confiança não se sobrepõem. Considerando o rastro completo, os tempos de espera medidos para a configuração community são menores que os medidos na configuração offline e pouco maiores que os tempos de espera da configuração cloud. Não há diferença significativa entre a configuração community e a configuração cloudlet. Para os outros dois rastros, não há diferença estatística

entre os tempos de espera das configurações community, cloudlet e cloud. Esses resultados indicam que a rede comunitária formada pelos usuários que estão online ao mesmo tempo pode ser tão boa quanto os casos em que há um servidor central para agregar todos os dados coletados no que diz respeito à qualidade das predições feitas.

A proporção de perda de todos os ônibus para todos os cenários simulados é muito pequena, chegando no máximo em 0.96%, e menor que 0.24% para todos os cenários que consideram o rastro dos usuários mais ativos (ver Tabela 4.3).

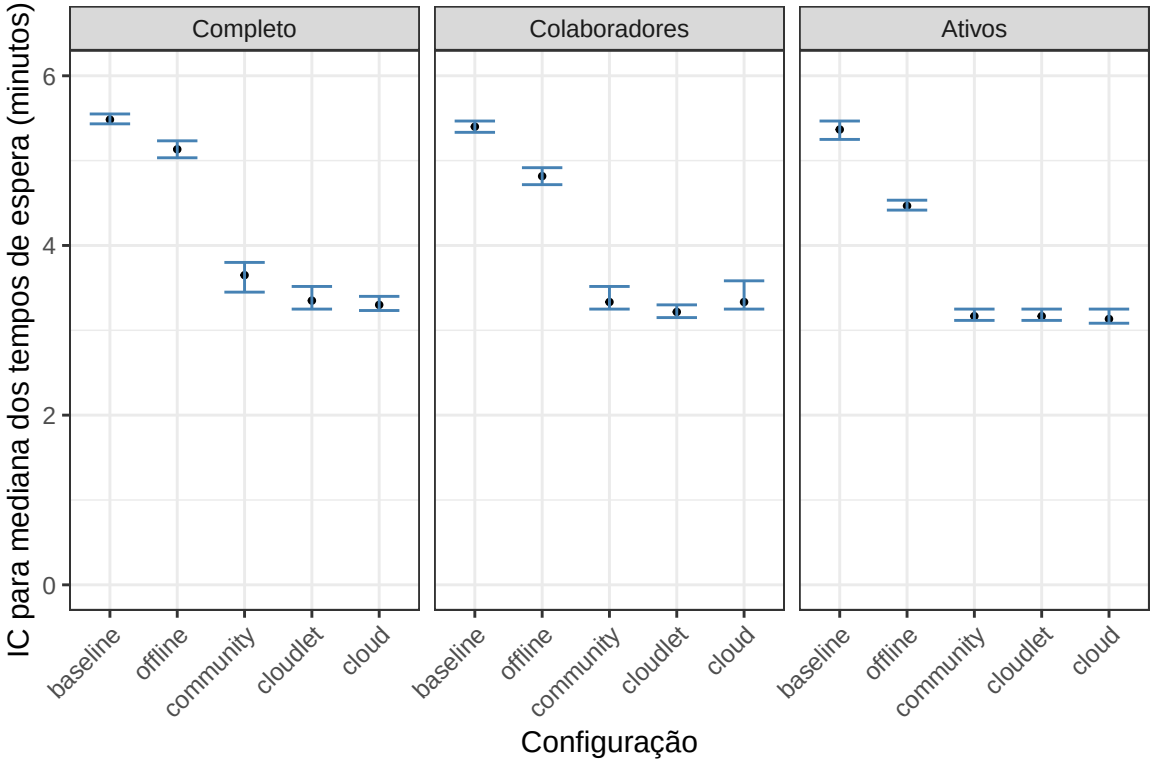

Figura 4.5: Intervalos para mediana do tempo de espera com confiança de 95%

Também foi medida a diferença entre a a configuração baseline e todas as outras configurações. Para isso, as requisições foram pareadas em cada uma das configurações. A tabela 4.4 mostra a proporção de requisições nas quais o tempo de espera foi melhor (ou seja, menor), pior (ou seja, maior) e igual à configuração baseline.

Nas configurações com mais dados disponíveis para predição, a proporção de requisições nas quais o tempo de espera foi melhor que a configuração baseline aumenta. Novamente, a principal diferença entre os rastros está na configuração offline. Como dito anteriormente, a estratégia baseline é usada em 60% das requisições da configuração offline com o rastro com-

pleto. Isso significa que nesses casos, os resultados da configuração offline são os mesmos da baseline. Em apenas 23% dos casos são vistos melhores resultados para a configuração offline. No rastro "Ativos", essa proporção aumenta para 46%.

A configuração community teve tempos de espera menores para mais de 77% das solicitações. As configurações cloudlet e cloud são melhores que a baseline em mais de 80% dos casos. Mais uma vez, no rastro "Ativos" o resultado da configuração community se aproxima ainda mais das configurações cloudlet e cloud.

Nesse Experimento I, questões sobre as relações de confiança entre os usuários não são levadas em consideração e considera-se que todos os usuários da rede comunitária confiam uns nos outros. Nos próximos experimentos foi avaliado como o desempenho do serviço governado pela comunidade é impactado pelas relações de confiança entre os usuários.

Tabela 4.4: Comparação dos tempos de espera entre a configuração baseline e as outras configurações

Rastro	**Configuração**	**Melhor**	**Pior**	**Igual**
Completo	Offline	23.4%	16.4%	60.2%
	Community	77.7%	13.7%	8.6%
	Cloudlet	81.0%	13.8%	5.2%
	Cloud	83.8%	12.6%	3.6%
Colaboradores	Offline	29.9%	20.8%	49.3%
	Community	79.8%	13.6%	6.6%
	Cloudlet	82.1%	13.3%	4.6%
	Cloud	83.2%	13.2%	3.6%
Ativos	Offline	46.0%	29.8%	24.2%
	Community	82.0%	12.8%	5.2%
	Cloudlet	82.5%	12.8%	4.7%
	Cloud	82.9%	12.9%	4.2%

4.3.2 Análise do compartilhamento de dados usando as restrições de redes sociais sintéticas

Neste experimento, foi considerado um cenário em que cada usuário do rastro "Ativos" pode receber informações de exatamente m outros usuários. A Figura 4.6 e a Figura 4.7 mostram, respectivamente, os intervalos com confiança de 95% para a média de QD e para a mediana de TEP, para todas as configurações simuladas no Experimento I, além de várias configurações para o cenário community-m, simulados no Experimento II. Os valores de todas as métricas de interesse obtidos no Experimento II estão disponíveis no Apêndice C.

Note que os intervalos de confiança do Experimento II possuem uma natureza diferente daqueles mostrados no Experimento I. No primeiro experimento, calculou-se intervalos para mediana e média de TEP e QD respectivamente a partir das requisições individuais daquelas simulações, mas os resultados finais são determinísticos. Assim, há só uma mediana para TEP e uma média para QD nesses casos, que estão em vermelho nas figuras a seguir. Como já dito anteriormente, os m usuários dos quais um indivíduo pode receber dados foram sorteados aleatoriamente, e 50 repetições foram realizadas. Cada uma delas gerou uma mediana para o tempo de espera e uma média para a quantidade de dados, e partir delas um intervalo foi calculado para a média das estatísticas de interesse (mediana para TEP e média para QD).

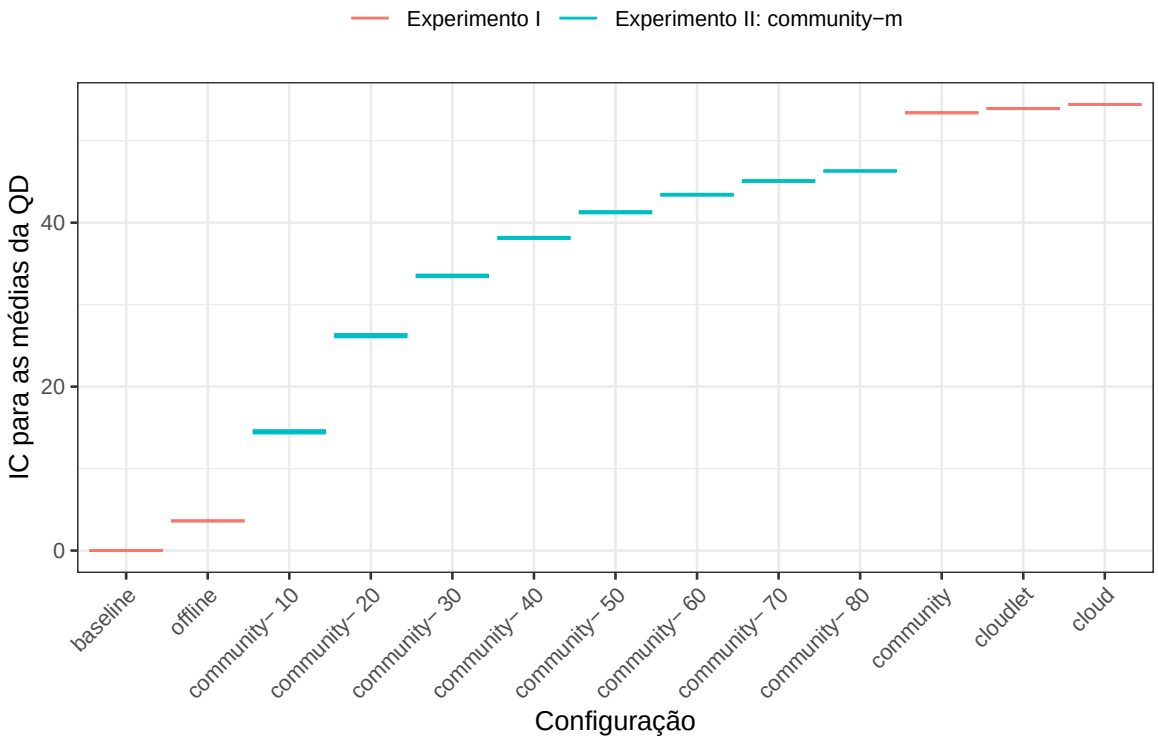

Figura 4.6: Intervalos de confiança para a média das médias da quantidade de dados por configuração

Como esperado, à medida que m aumenta, aumenta também a quantidade média de informações usadas por predição. Como resultado, a mediana de TEP diminui à medida que m aumenta. Além disso, para um m tão baixo quanto 20, a quantidade de dados utilizada para realizar as predições já é, em média, bem maior que a obtida na configuração offline (26.2 x 3.6). A análise se aplica para a métrica TEP, cujo resultado alcançado também já é bem melhor do que o obtido quando a configuração offline é usada — enquanto a configuração community-20 aumenta o TEP em $24,3\%$ quando comparado à configuração cloud, para a configuração offline, o aumento de TEP é de $42,5\%$ (consulte a Tabela 4.3). Permitindo que o usuário receba informação de outros 80 usuários (community-80), o resultado já é bem próximo da configuração community, onde todos confiam em todos, tanto para quantidade dados, como para o tempo de espera na parada.

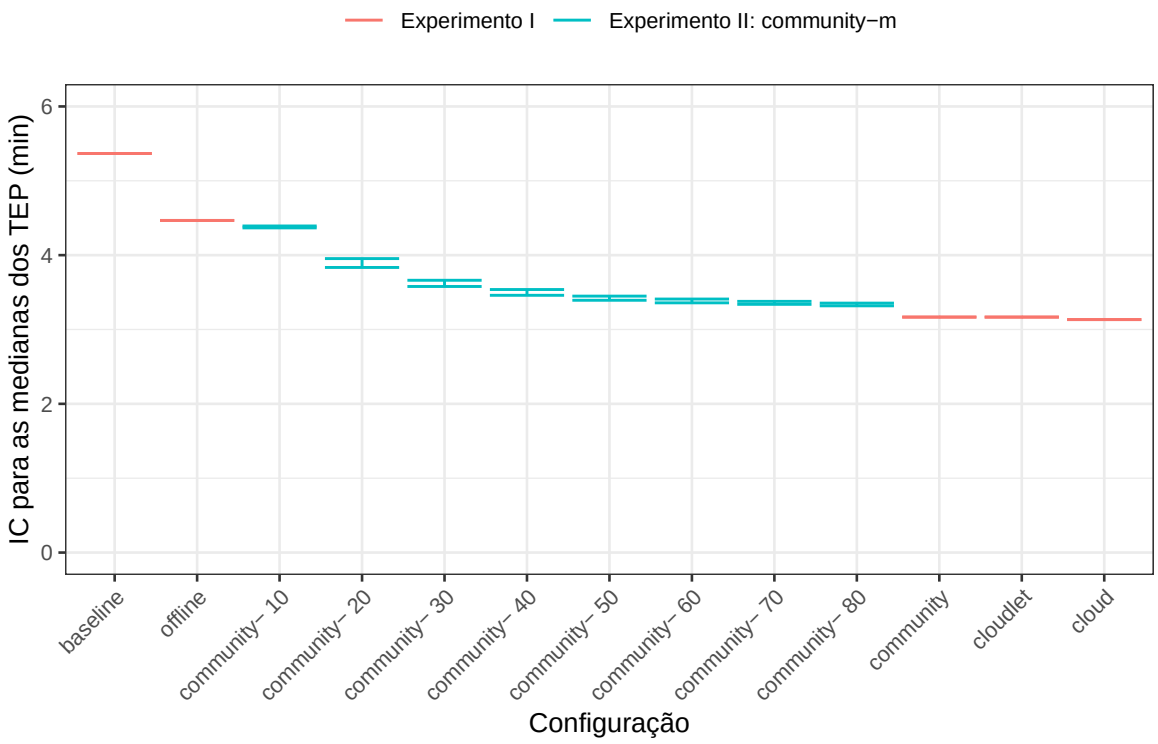

Figura 4.7: Intervalos de confiança para a médias das medianas dos tempos de espera por configuração

4.3.3 Análise do compartilhamento de dados usando as restrições de redes sociais reais

Agora foi considerado um cenário mais realista em que as relações de confiança reais são usadas para informar ao modelo de simulação sobre quais dados podem ser compartilhados (consulte a seção 4.2.5). Para efeitos de comparação, foram usados alguns resultados do Experimento I e do Experimento II. A Tabela 4.5 mostra resultados para uma única execução de experimento (Baseline, Offline, Community, Cloudlet e Cloud), resultados para execuções de experimentos replicados (community-m e community-real-i) e um resultado que agrega todas as requisições de todas as simulações replicadas envolvendo redes reais (community-real-all). As colunas QD e TEP representam, respectivamente, a média da quantidade de dados disponíveis e a mediana do tempo de espera para os casos em que um único experimento é realizado, e a média das médias e a média das medianas para os casos em que os experimentos replicados são executados. Para este último, a tabela também mostra os

intervalos de confiança de 95% (colunas QD (I.C.) e TEP (I.C.)). Os valores das métricas complementares DP e PP do Experimento III estão disponíveis no Apêndice D.

Tabela 4.5: Resumo das métricas QD and TEP (em minutos)

Configuração	**QD**	**QD (I.C.)**	**TEP**	**TEP (I.C.)**
Baseline	0.0	-	5.36	-
Offline	3.6	-	4.46	-
Community-20	26.2	[26.0, 26.3]	3.89	[3.83, 3.95]
Community-real-1	27.9	[27.4, 28.4]	3.91	[3.88, 3.93]
Community-real-2	30.4	[30.2, 30.6]	3.80	[3.78, 3.81]
Community-real-3	33.6	[33.1, 34.0]	3.71	[3.69, 3.73]
Community-real-4	34.3	[34.1, 34.5]	3.67	[3.66, 3.69]
Community-real-5	35.3	[35.2, 35.3]	3.63	[3.62, 3.65]
Community-real-6	38.0	[37.7, 38.3]	3.54	[3.52, 3.55]
Community-real-7	38.4	[38.3, 38.6]	3.53	[3.51, 3.54]
Community-real-all	34.0	[33.8, 34.1]	3.68	[3.67, 3.69]
Community-40	38.1	[38.0, 38.2]	3.50	[3.46, 3.53]
Community	53.4	-	3.16	-
Cloudlet	53.9	-	3.16	-
Cloud	54.4	-	3.13	-

A partir dos resultados podemos concluir que a relação de confiança existente nas redes sociais reais faz com que o sistema funcione em um nível que esteja entre as configurações community-20 e community-40 do Experimento II. Como mencionado anteriormente, os resultados para a configuração community-20 já são muito melhores que a configuração offline, tanto para quantidade de dados agregada, como para qualidade das predições. Por outro lado, quando comparado ao desempenho das configurações cloudlet e cloud, o TEP médio do serviço governado pela comunidade (community-real-all) é, respectivamente, $16,5\%$ e $17,6\%$ maior. Nesse estudo, esse é o preço médio pago por limitar a exposição dos seus dados apenas a um certo grupo de usuários. Assim, se a privacidade não for uma preocupação, o serviço governado pela comunidade poderá agregar mais dados e fornecer um serviço que tenha um desempenho tão bom quanto as configurações cloudlet e cloud, mas sem as limitações de governança externa que acompanham a necessidade de um provedor

centralizado.

Capítulo 5

Conclusões

Este trabalho propõe uma arquitetura na qual os próprios usuários definem a governança de aplicações que usam dados coletados do ambiente para prover algum tipo de serviço. Na arquitetura tradicional desse tipo de aplicação, usualmente baseada em nuvem, os dados são coletados do ambiente usando dispositivos de borda, como sensores e smartphones, e depois são enviados para nuvem, onde são armazenados e processados. Esse tipo de modelo expõe algumas limitações que são atacadas pela solução proposta, relacionadas principalmente à governança da aplicação e dos dados envolvidos, totalmente controlada por um provedor de aplicação de nuvem centralizado.

No modelo proposto, usuários podem definir a governança de um serviço criado para o bem da comunidade usando os princípios de Sensoriamento Participativo, MSN e Computação na Borda. A ideia é que os membros da comunidade usem os recursos de seus dispositivos móveis na borda da rede para coletar dados do ambiente, compartilhá-los com outros membros e processá-los no próprio dispositivo, sem precisar enviá-los para uma nuvem remota. Isso evita a necessidade de governança externa para dados e aplicação, além de fornecer um maior controle sobre quem tem acesso direto aos dados que foram coletados.

A utilização dos padrões de mobilidade dos usuários, bem como da capacidade dos seus dispositivos para coletar informações, é bastante usada em inúmeras aplicações, mas na maioria dos casos, uma infraestrutura de nuvem centralizada é necessária para armazenar e processar os dados, conforme apresentado no Capítulo 2. Além disso, as aplicações que fazem uso do paradigma de Computação na Borda lançam mão de uma abordagem mais conservadora, na qual servidores instalados entre os usuários e a nuvem servem apenas para fazer

um pré-processamento dos dados. Isso não resolve os problemas associados à governança da aplicação.

Apesar de oferecer os meios necessários para que os usuários governem as aplicações e os dados, os dispositivos da borda da rede (i.e. smartphones) possuem características que podem limitar o modelo proposto, como capacidades de processamento e armazenamento limitadas. Isso pode ser mitigado, por exemplo, usando modelos preditivos simples que demandem pouca computação, ou descartando dados antigos. Dessa forma, nem todas as aplicações dirigidas pela coleta de dados podem ser elegíveis a utilizar a arquitetura proposta, a exemplo de aplicações que necessitem de um armazenamento massivo de dados e de um complexo processamento.

Com o objetivo de avaliar a viabilidade do modelo proposto em termos da quantidade de dados que podem ser agregados pelos usuários, e que serão utilizados por algum modelo preditivo, foi elaborado um estudo de caso (descrito no Capítulo 4) no contexto de transporte público de ônibus, usando dados da cidade de Curitiba/PR. Nessa prova de conceito, estudantes universitários estão interessados em saber o horário de partida do primeiro ônibus de uma linha específica em um período de uma hora usando dados históricos de viagens. Para isso, eles compartilham dados coletados com outros membros com o objetivo de alimentar algum gerador de modelo de predição responsável por responder as requisições de fato. Além da configuração de compartilhamento de dados proposta pelo trabalho, na qual indivíduos de uma comunidade compartilham informações entre si quando estão em uma zona próxima, usando seus dispositivo móveis, foram usadas outras configurações para efeitos de comparação. Essas configurações simulam desde situações mais simples, como cenários nos quais não há compartilhamento de dados, até abordagens usadas atualmente, como a agregação de dados coletados na borda da rede em um servidor na nuvem, ou em servidor instalado mais próximo dos usuários.

Os resultados e discussões (Seção 4.3) foram apresentados sob a forma de três experimentos diferentes. No primeiro experimento, assume-se que, na configuração community, todos os usuários confiam uns nos outros. Assim, para compartilharem dados entre si, apenas precisam estar online ao mesmo tempo. Os resultados mostraram que é possível agregar dados suficientes da comunidade de usuários para realizar boas predições. A quantidade de dados agregada é bem maior do que um único usuário poderia coletar (offline). Além disso,

quando a privacidade não é uma preocupação, a quantidade de dados agregada está próxima das abordagens onde um servidor central é necessário (cloudlet e cloud), sem enfrentar os riscos associados à necessidade de governança externa.

Os outros dois experimentos visaram avaliar o impacto das mudanças no nível de confiança entre os usuários na configuração community. Em outras palavras, simula cenários nos quais os usuários limitam a exposição de seus dados. Em um deles, assume-se que o usuário pode receber informação de exatamente m outros usuários aleatórios. No outro, foram utilizadas redes sociais reais para definir as relações de confiança entre os usuários. Os resultados mostraram que, quando os usuários limitam a exposição de seus dados, compartilhando apenas com quem confiam, a quantidade de dados agregada e a qualidade das predições são afetadas, mas ainda assim apresentam resultados razoáveis. Quanto maior é a rede do usuário, mais dados ele pode obter e melhores são as predições feitas. Sendo assim, o modelo proposto permite uma certa flexibilidade de forma a negociar a diminuição da exposição de dados em troca de perda de desempenho e vice versa. Portanto, o usuário pode deliberar, de forma autônoma e sem se sujeitar à política de uso de dados de um provedor de aplicação de nuvem externo, se prefere expor mais ou menos os seus dados em troca de uma menor ou maior qualidade de serviço.

Bibliografia

[1] M. Aazam and E. Huh. Fog computing and smart gateway based communication for cloud of things. In *2014 International Conference on Future Internet of Things and Cloud*, pages 464–470, Aug 2014.

[2] Mohammad Abu Alsheikh, Yutao Jiao, Dusit Niyato, Ping Wang, Derek Leong, and Zhu Han. The accuracy-privacy trade-off of mobile crowdsensing. *IEEE Communications Magazine*, 55(6):132–139, 2017.

[3] P. Bellavista, S. Chessa, L. Foschini, L. Gioia, and M. Girolami. Human-enabled edge computing: Exploiting the crowd as a dynamic extension of mobile edge computing. *IEEE Communications Magazine*, 56(1):145–155, Jan 2018.

[4] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019.

[5] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[6] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *In: Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.

[7] Tarciso Braz de Oliveira Filho. *Inferring passenger-level bus trip traces from sche-*

dule, positioning and ticketing data: methods and applications. Master dissertation, Universidade Federal de Campina Grande, Paraíba, Brasil, 2019.

[8] Raghu K. Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F. Abdelzaher. Greengps: A participatory sensing fuel-efficient maps application. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 151–164, New York, NY, USA, 2010. ACM.

[9] B. Guo, Z. Yu, X. Zhou, and D. Zhang. From participatory sensing to mobile crowd sensing. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 593–598, March 2014.

[10] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[11] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition, 1975.

[12] Stéphane J. Kuendig, Jose Rolim, Konstantinos M. Angelopoulos, and Mahmood Hosseini. Crowdsourced edge: a novel networking paradigm for the collaborative community. Technical report, 2019. ID: unige:114607; Paper submitted for publication at the Global IoT Summit 2019.

[13] Thorsten Lohmar, Ali Zaidi, Håkan Olofsson, and Christer Boberg. Driving transformation in the automotive and road transport ecosystem with 5G. *Ericsson Technology Review*, 2019.

[14] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, and Limin Sun. Fog computing: Focusing on mobile users at the edge. *CoRR*, abs/1502.01815, 2015.

[15] Thomas Ludwig, Christian Reuter, Tim Siebigteroth, and Volkmar Pipek. Crowdmonitor: Mobile crowd sensing for assessing physical and digital activities of citizens during emergencies. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 4083–4092, New York, NY, USA, 2015. ACM.

[16] João Mafra, Francisco V. Brasileiro, and Raquel Vigolvino Lopes. Community-governed services on the edge. In Donald Ferguson, Markus Helfert, and Claus Pahl, editors, *Proceedings of the 10th International Conference on Cloud Computing and Services Science, CLOSER 2020, Prague, Czech Republic, May 7-9, 2020*, pages 498–505. SCITEPRESS, 2020.

[17] M. Marjanović, A. Antonić, and I. P. Žarko. Edge computing architecture for mobile crowdsensing. *IEEE Access*, 6:10662–10674, 2018.

[18] Emiliano Miluzzo, Nicholas D. Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B. Eisenman, Xiao Zheng, and Andrew T. Campbell. Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 337–350, New York, NY, USA, 2008. ACM.

[19] B. Predić, Z. Yan, J. Eberle, D. Stojanovic, and K. Aberer. Exposuresense: Integrating daily activities with air quality using mobile participatory sensing. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 303–305, March 2013.

[20] Sasank Reddy, Katie Shilton, Gleb Denisov, Christian Cenizal, Deborah Estrin, and Mani Srivastava. Biketastic: Sensing and mapping for better biking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1817–1820, New York, NY, USA, 2010. ACM.

[21] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

[22] L. Ruge, B. Altakrouri, and A. Schrader. Soundofthecity - continuous noise monitoring for a healthy city. In *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 670–675, March 2013.

[23] Mahadev Satyanarayanan, Victor Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 2009.

[24] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, Oct 2016.

[25] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of Facebook networks. *Phys. A*, 391(16):4165–4180, Aug 2012.

[26] Flora S. Tsai, Wenchou Han, Junwei Xu, and Hock Chuan Chua. Design and development of a mobile peer-to-peer social networking application. *Expert Systems with Applications*, 36(8):11077 – 11087, 2009.

[27] P. Zhou, Y. Zheng, and M. Li. How long to wait? predicting bus arrival time with mobile phone based participatory sensing. *IEEE Transactions on Mobile Computing*, 13(6):1228–1241, June 2014.

[28] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez. Robust mobile crowd sensing: When deep learning meets edge computing. *IEEE Network*, 32(4):54–60, 2018.

[29] Tiancheng Zhuang, Paul Baskett, and Yi Shang. Managing ad hoc networks of smartphones. *International Journal of Information and Education Technology*, 3(5):540, 2013.

Apêndice A

Instruções para reprodução dos experimentos

O simulador utilizado nos experimentos foi construído usando a linguagem Java e os resultados foram analisados utilizando a linguagem R. A seguir são explicados os passos a serem seguidos para reprodução dos resultados reportados neste trabalho.

A.1 Funcionamento do simulador

O simulador construído[1] é capaz de executar experimentos de simulação usando diferentes rastros e diferentes políticas de compartilhamento de dados. Para executar um caso de simulação basta executar o comando:

- *java -r target/cbas.jar conf/simulation.conf*

O executável *target/cbas.jar* presente no repositório contempla o código atual com o qual os experimentos de simulação foram executados. Em caso de modificação no código, o executável pode ser gerado novamente usando o seguinte comando no diretório raíz:

- *mvn package -Dmaven.test.skip=true*

[1] O download do simulador (community-governed-simulator.zip) pode ser feito em https://drive.google.com/drive/folders/1wyRCbAJLITbJ2a0ZIyS_4t9UY3xL-zOT?usp=sharing. Apesar de o repositório ser privado, a solicitação para acesso pode ser feita no próprio repositório.

O arquivo conf/simulation.conf é o arquivo de entrada para o simulador e possui os seguintes parâmetros a serem preenchidos:

- **input_trace_path**: caminho para o arquivo contendo todos eventos dos usuários.
- **presence_trace_path**: caminho para o arquivo auxiliar contendo todos os intervalos de tempo nos quais os usuários estiveram no campus.
- **fully_connected_network**: booleano que indica se no cenário a ser executado todos os usuários confiam uns nos outros (true) ou se esse compartilhamento é limitado por uma rede social (false).
- **real_network**: booleano que indica se no cenário a ser executado a rede social utilizada é real (true) ou sintética (false).
- **friends_network_path**: caminho para o arquivo contendo o grafo que conecta os usuários da comunidade, se for o caso.
- **output_result_url**: caminho para o destino dos resultados após a execução da simulação. Os resultados são escritos em um arquivo sqlite3 ao fim da execução.
- **data_usage_policy_class**: configuração de compartilhamento de dados a ser utilizada no experimento.
- **wait_limit**: tempo de espera limite no ponto de ônibus. Usado para computar os tempos de espera ao fim da simulação.

A.2 Arquivos de entrada

A.2.1 input_trace_path

A sequência de eventos dos usuários é descrita por um arquivo contendo as seguintes colunas:

- **id**: identificador do evento.
- **type**: indica o tipo do evento. ARRIVAL | DEPARTURE.

- **timestamp**: horário do evento no formato YYYY-MM-dd HH:mm:ss. Se o evento for ARRIVAL, indica o horário da chegada do usuário ao campus e que portanto ele tornou sua informação disponível para outros membros da comunidade. Se o evento for DEPARTURE, indica o timestamp t_r da requisição feita pelo usuário para sair do campus.
- **start_time**: se o evento for do tipo ARRIVAL, indica o timestamp da viagem coletada no passado que o usuário traz consigo naquele momento para o campus. Se o evento for do tipo DEPARTURE, indica o horário real da viagem que gerou a requisição no tempo t_r.
- **origin**: identificador do ponto de ônibus de origem.
- **dest**: identificador do ponto de ônibus de destino.
- **route**: identificador da linha de ônibus usada.
- **card_num**: identificador do usuário.
- **first_departure**: Existe apenas para os eventos do tipo DEPARTURE e indica a saída do próximo ônibus da linha após o timestamp t_r da requisição. É usada ao fim da simulação para gerar a métrica do tempo de espera

A.2.2 presence_trace_path

Rastro auxiliar que reúne todas os eventos de chegada e saída dos usuários no formato *CARD_NUM, ARRIVAL_TIMESTAMP, DEPARTURE_TIMESTAMP*, com os dois timestamps em segundos.

A.2.3 friends_network_path

Grafo que indica as relações de confiança entre os usuários da comunidade. Para o caso da redes sintéticas, cada linha do arquivo deve estar no seguinte formato:

CARD_NUM_0, CARD_NUM_1, CARD_NUM_2, ..., CARD_NUM_m, indicando um usuário seguido de outros m usuários que podem prover dados para ele.

Para o caso da redes reais, cada linha do arquivo deve estar no seguinte formato:

CARD_NUM_0, CARD_NUM_1, indicando que o usuário 0 confia no usuário 1 e vice versa.

A.3 Automatização da execução dos experimentos

O repositório do simulador contém todos os arquivos de configuração usados nos experimentos na pasta *conf* já preenchidos, além do executável compilado na pasta target. Além disso, a pasta *input_trace* contém todos os arquivos de entrada já separados pelo rastro utilizado (completo, colaboradores e ativos). As 400 redes sintéticas e 3500 redes reais usadas estão disponíveis para download[2]. Os arquivos networks-community-m.tar.gz e networks-community-i.tar.gz devem ser descompactados na pasta *input_trace* antes das reproduções dos experimentos II e III, respectivamente.

Na pasta raiz do simulador existem scripts que executam os três experimentos do trabalho. Para executar o experimento I, onde não há redes que limitam o compartilhamento de dados, basta executar:

- *bash run_experimentI.sh*

Os resultados serão gerados na pasta *output-experimentI*.

Para executar o experimento II, usando redes sintéticas (e assumindo que elas foram descompactadas corretamente, conforme instruído anteriormente), basta executar:

- *bash run_experimentII.sh $NUM_REPETICOES*

Os resultados serão gerados na pasta *output-experimentII*.

Para executar o experimento III, usando redes reais (e assumindo que elas foram descompactadas corretamente, conforme instruído anteriormente), basta executar:

- *bash run_experimentIII.sh $NUM_REPETICOES*

Os resultados serão gerados na pasta *output-experimentIII*.

[2]https://drive.google.com/drive/folders/1wyRCbAJLITbJ2a0ZIyS_4t9UY3xL-zOT?usp=sharing

A.4 Saída dos experimentos

Cada simulação executada vai gerar um arquivo sqlite3 com os resultados. As seguintes colunas estão presentes no arquivo:

- **id:** identificador da requisição do usuário.
- **timestamp:** horário da requisição no formato YYYY:MM:dd HH:mm:ss.
- **user:** identificador do usuário que fez a requisição.
- **route:** linha ônibus para qual foi feita a requisição.
- **prediction:** timestamp estimado pelo algoritmo de predição no formato YYYY:MM:dd HH:mm:ss.
- **used_information:** quantidade de dados usada pelo algoritmo para responder aquela requisição.
- **next_departure_miss:** o timestamp da saída do próximo ônibus, caso o usuário perca o primeiro, também no formato YYYY:MM:dd HH:mm:ss. Pode assumir o valor nulo caso o usuário pegue o primeiro ônibus.
- **waiting_time:** inteiro que representa o tempo de espera na parada considerando que o usuário seguiu a recomendação do algoritmo. Representa a diferença entre o horário recomendado e o horário de saída do próximo ônibus. Caso o usuário perca o primeiro ônibus e não existam mais saídas daquela linha naquele dia, esse valor é -1. Caso o tempo de espera seja igual a -1 ou maior que 1 hora, assuma-se no script de análise que o usuário esperou uma hora.

A.5 Análise de resultados

Para gerar as tabelas e gráficos dos resultados[3], existem scripts dentro do diretório *R* que reproduzem a análise da seção 4.3.

[3]As saídas das simulações que foram usadas na análise do estudo de caso podem ser baixadas no link: https://drive.google.com/drive/folders/1wyRCbAJLITbJ2a0ZIyS_4t9UY3xL-zOT?usp=sharing

Para gerar os resultados do experimento I, basta executar o seguinte comando passando como argumento a pasta onde estão os resultados:

- *Rscript R/analysis-experimentI.R $OUTPUT_EXPERIMENTI*

Para gerar os resultados do experimento II, basta executar o seguinte comando passando as pastas onde estão os resultados dos experimentos I e II, seguido do número de repetições usadas no experimento II:

- *Rscript R/analysis-experimentI.R $OUTPUT_EXPERIMENTI $OUTPUT_EXPERIMENTII $NUM_REPETICOES*

Para gerar os resultados do experimento III, basta executar o seguinte comando passando como parâmetro a pasta onde estão os resultados e o número de repetições usadas no experimento III:

- *Rscript R/analysis-experimentIII.R $OUTPUT_EXPERIMENTIII $NUM_REPETICOES*

Apêndice B

Distribuição da quantidade de dados e tempos de espera em todas as simulações do Experimento I

As figuras abaixo mostram a distribuição dos valores de tempo de espera e quantidade de dados usados em todas as predições de todas as simulações do Experimento I. A configuração baseline foi excluída da Figura B.2, pois em todas as requisições o valor de QD é 0.

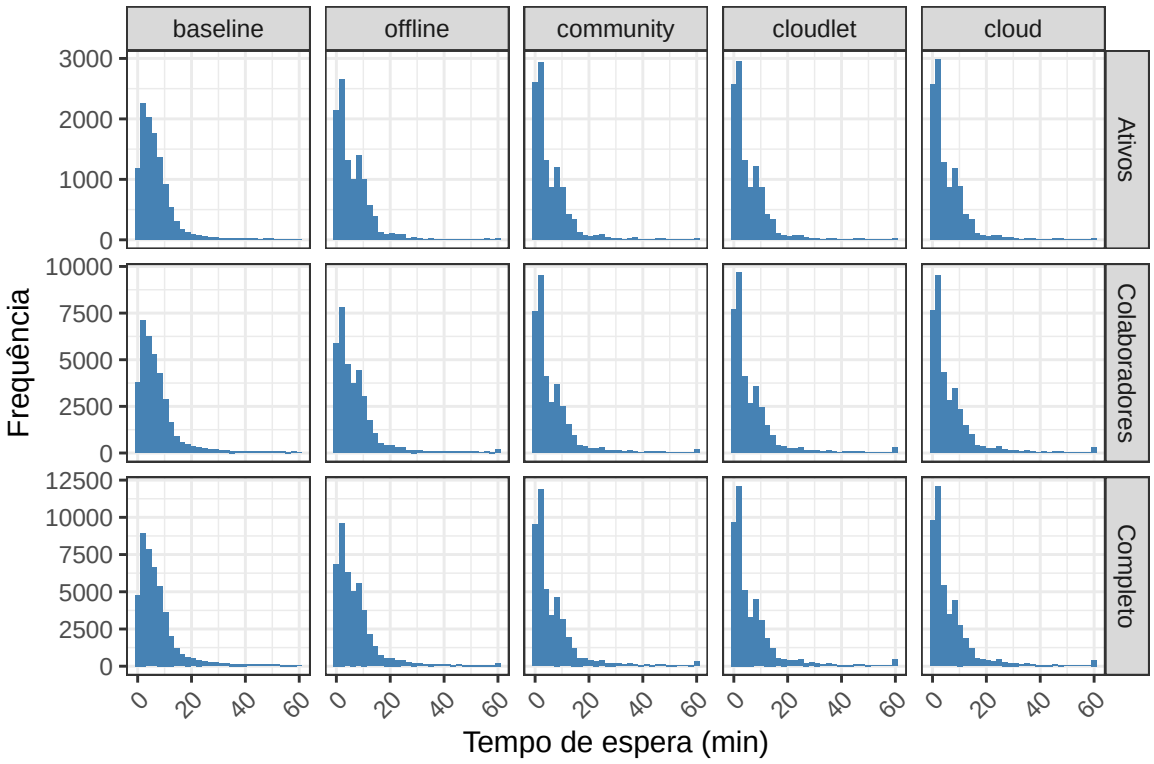

Figura B.1: Histograma dos tempos de espera obtidos em todas requisições de todas as configurações (em minutos)

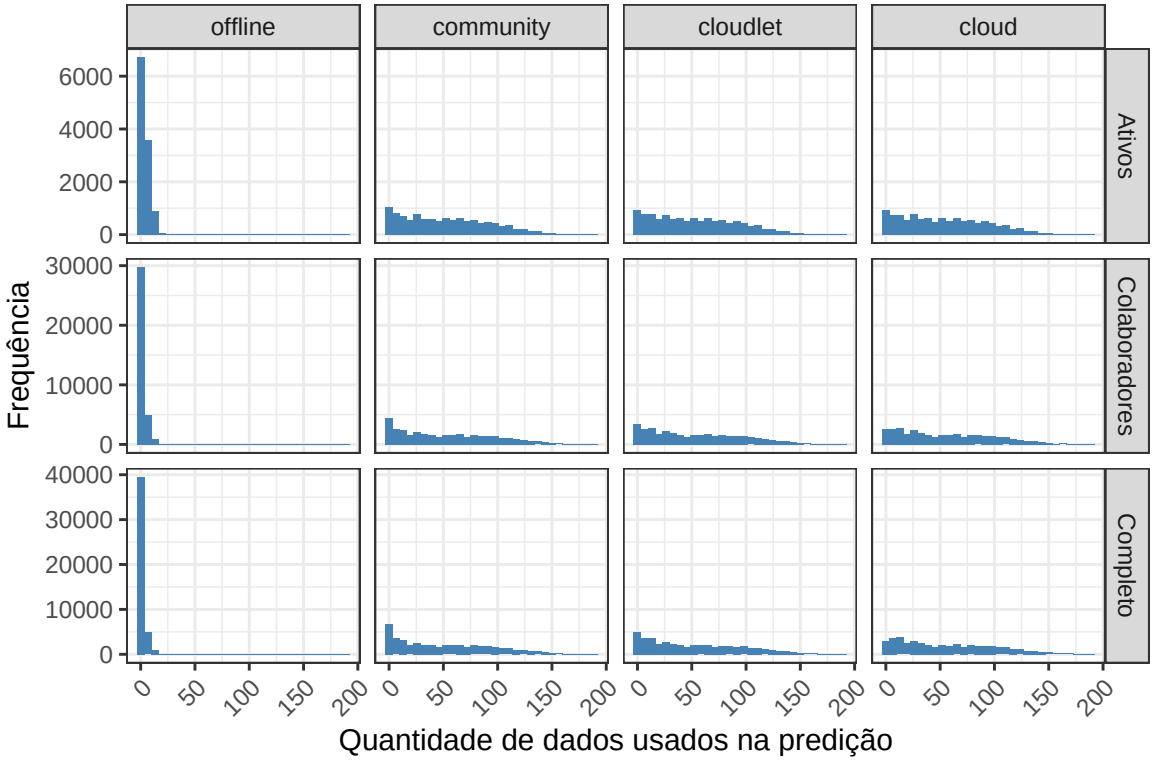

Figura B.2: Histograma para a quantidade dados usados em todas requisições de todas as configurações

Apêndice C

Experimento II: sumário de todas as métricas

A tabela a seguir complementa os resultados apresentados na Seção 4.3.2, mostrando o valor médio obtido para as métricas QD, DP, PP e TEP nas 50 repetições, além dos respectivos intervalos de confiança.

Tabela C.1: Resumo das métricas QD, DP (%), PP (%) e TEP (minutos) e seus respectivos intervalos com 95% de confiança.

Configuração	**QD**	**QD (I.C)**	**DP**	**DP (I.C)**	**PP**	**PP (I.C)**	**TEP**	**TEP (I.C)**
Community-10	14.5	[14.3,14.6]	83.0	[83.0, 83.1]	0.22	[0.22,0.23]	4.38	[4.36,4.39]
Community-20	26.2	[26.0, 26.3]	86.4	[86.3, 86.5]	0.19	[0.19,0.20]	3.89	[3.83, 3.95]
Community-30	33.5	[33.4, 33.6]	88.4	[88.4, 88.5]	0.19	[0.18,0.19]	3.62	[3.57, 3.66]
Community-40	38.1	[38.0, 38.2]	89.7	[89.6, 89.8]	0.18	[0.18, 0.19]	3.50	[3.46, 3.53]
Community-50	41.3	[41.2, 41.4]	90.5	[90.5, 90.6]	0.18	[0.18, 0.19]	3.42	[3.39, 3.45]
Community-60	43.4	[43.3, 43.5	91.3	[91.3, 91.4]	0.19	[0.18, 0.19]	3.38	[3.35, 3.41]
Community-70	45.0	[45.0, 45.1]	91.9	[91.8, 91.9]	0.18	[0.18, 0.19]	3.35	[3.33, 3.38]
Community-80	46.3	[46.2, 46.4]	92.3	[92.3, 92.4]	0.18	[0.18,0.19]	3.33	[3.31, 3.35]

Apêndice D

Experimento III: métricas DP e PP

A tabela a seguir complementa os resultados apresentados na Seção 4.3.3, mostrando o valor médio obtido para as métricas DP e PP nas 500 repetições, além dos respectivos intervalos de confiança.

Tabela D.1: Resumo das métricas $DP(\%)$ e $PP(\%)$ e seus respectivos intervalos com 95% de confiança.

Configuração	**DP**	**DP (I.C)**	**PP**	**PP (I.C)**
Community-real-1	86.6	[86.4, 86.7]	0.20	[0.20,0.20]
Community-real-2	87.3	[87.2, 87.3]	0.20	[0.20,0.20]
Community-real-3	88.1	[88.0, 88.3]	0.20	[0.19,0.20]
Community-real-4	88.4	[88.3, 88.4]	0.20	[0.19, 0.20]
Community-real-5	88.5	[88.5, 88.5]	0.19	[0.19, 0.19]
Community-real-6	89.6	[89.5, 89.7]	0.19	[0.19, 0.19]
Community-real-7	89.7	[89.6, 89.8]	0.19	[0.19, 0.19]
Community-real-all	88.3	[88.3, 88.4]	0.20	[0.19,0.20]