MARCUS VINÍCIUS SOUZA COSTA

MECANISMO PARA SELEÇÃO DINÂMICA DE RECURSOS PARA PROVIMENTO CONTROLE-COMO-UM-SERVIÇO EM AMBIENTE DE COMPUTAÇÃO EM NÉVOA DINÂMICO E HETEROGÊNEO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

Orientador: Vitor Barbosa Carlos de Souza

VIÇOSA - MINAS GERAIS
2020

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
C837m
2020

Costa, Marcus Vinícius Souza, 1988-
Mecanismo para seleção dinâmica de recursos para provimento controle-como-um-serviço em ambiente de computação em névoa dinâmico e heterogêneo / Marcus Vinícius Souza Costa. – Viçosa, MG, 2020.
62f. : il. (algumas color.) ; 29 cm.

Orientador: Vitor Barbosa Carlos de Souza.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f.59-62.

1. Redes de computadores. 2. Internet das coisas.
3. Sistemas de computação distribuídos heterogêneos.
I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação.
II. Título.

CDD 22 ed. 004.6

MARCUS VINÍCIUS SOUZA COSTA

MECANISMO PARA SELEÇÃO DINÂMICA DE RECURSOS PARA PROVIMENTO CONTROLE-COMO-UM-SERVIÇO EM AMBIENTE DE COMPUTAÇÃO EM NÉVOA DINÂMICO E HETEROGÊNEO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae.*

APROVADA: 06 de março de 2020.

Assentimento:

Marcus Vinícius Souza Costa
Autor

Vitor Barbosa Carlos de Souza
Orientador

A Deus, por conceder tudo aquilo de que preciso para viver e prosperar: tudo aquilo que eu sei e tudo aquilo que jamais poderei identificar ou mensurar. À minha mãe, Maria de Fátima Souza Costa, por me apoiar em todas as minhas dificuldades. À memória do meu pai, João Batista Martins da Costa, que jamais será esquecido.

Agradecimentos

A Deus, por me conceder a capacidade de concluir o curso e me tornar melhor pessoa e profissional.

Aos meus pais, Maria de Fátima Souza Costa e João Batista Martins da Costa (*in memoriam*) pelo amor e apoio incondicional dados a mim durante toda a vida. Estendo também a todos meus familiares e amigos.

A todos os colaboradores do Departamento de Informática (DPI) da Universidade Federal de Viçosa, pelo talento, suporte e condução exemplar do PPGCC (Programa de Pós-Graduação *Stricto Sensu* em Ciência da Computação). Em especial ao meu orientador, Prof. Dr. Vitor Barbosa Carlos de Souza, pelo apoio, amizade, compartilhamento do conhecimento, paciência e dedicação de incontáveis horas, durante toda a realização da pesquisa.

Aos colaboradores do Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais (IFSUDESTEMG - Campus Muriaé), a destacar os amigos do Núcleo de Informática e Engenharias, especialmente ao Diego Rossi, Gustavo Willam e Paulo Vinícius, pelos esforços para que os meus horários de trabalho pudessem coexistir com as aulas do PPGCC e tornar menos exaustiva a dura tarefa de trabalhar, estudar e realizar as pesquisas de forma concomitante. Destaco também os amigos do setor de Tecnologia da Informação: Neuberth, Rafael, Reginaldo e Saulo, por viabilizar a utilização dos laboratórios de Informática em momentos propícios para que eu pudesse realizar alguns testes experimentais para a realização desta pesquisa.

Gostaria de agradecer também a todos os brasileiros pelos esforços para manter a oferta de ensino de qualidade, público e gratuito provindos pelas Universidades públicas Federais, a destacar a Universidade Federal de Viçosa. Sua missão incomensurável de mudar a vida de todos os alunos, egressos, da sociedade brasileira em geral e de todas as pessoas ao redor do mundo, com ensino e pesquisas de destaque, nas mais diversas áreas do conhecimento, aprimorando o desenvolvimento socioeconômico da Zona da Mata Mineira e de todo o país, é fundamental para uma Nação sólida e próspera. Os meus agradecimentos a todos os servidores, ex-servidores e colaboradores terceirizados desta Instituição.

Por fim, agradeço aos colegas que fiz no PPGCC e a todos aqueles que, direta ou indiretamente, apoiaram a realização deste trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

"Não importa quão estreita a passagem,
Quantas punições ainda sofrerei,
Eu sou o mestre de meu destino:
Eu sou o capitão de minha alma."
(William Ernest Henley - tradução de trecho do poema Invictus)

Resumo

COSTA, Marcus Vinícius Souza, M.Sc., Universidade Federal de Viçosa, março de 2020. **Mecanismo para seleção dinâmica de recursos para provimento controle-como-um-serviço em ambiente de computação em névoa dinâmico e heterogêneo**. Orientador: Vitor Barbosa Carlos de Souza.

A Internet das Coisas é uma revolução tecnológica que traz grandes possibilidades para pessoas, governos, indústria e academia, sendo um dos pilares de diversas aplicações do mundo moderno. Estas aplicações costumam fazer uso intensivo de dispositivos inteligentes para poder atender aos requisitos exigidos, de modo que os mesmos produzem um alto volume de dados, frequentemente migrados para *data-centers* disponibilizados por grandes provedores de serviço em nuvem, para armazenamento e processamento. A latência observada neste modelo pode não atender a requisitos de aplicações com restrição de tempo, motivando o aparecimento de um novo paradigma de computação: a computação em névoa. Um dos principais objetivos deste modelo é apoiar a nuvem, aproveitando o potencial dos dispositivos da borda e provendo baixa latência. Um plano de controle da névoa efetivo necessita conhecer os recursos disponíveis na borda para alcançar os benefícios provindos por este modelo. Devido ao grande número de dispositivos da névoa e da alta dinamicidade e heterogeneidade dos recursos subjacentes, esta tarefa é desafiadora e é foco de interesse da comunidade científica. Uma abordagem recentemente proposta, denominada controle-como-um-serviço, define dispositivos para atuarem como controladores, conhecendo os recursos da borda e selecionando os mais apropriados para a execução do serviço. Neste trabalho, propomos uma estratégia para selecionar dinamicamente potenciais dispositivos da borda para assumir o papel de controlador da névoa, mapeando os recursos disponíveis e provendo controle-como-um-serviço. Desenvolvemos um modelo baseado em classificação, que pondera algumas características chave dos dispositivos e seleciona os mais capazes para assumir este papel. Posteriormente, o modelo foi aprimorado, através de duas extensões, com o objetivo de diminuir o número de trocas desnecessárias entre controladores, mostrando eficiência em todos os cenários avaliados. O mecanismo também mostrou melhor utilização de memória e bateria nos controladores do que o método comparativo.

Palavras-chave: Computação em Névoa. Classificação. Seleção dinâmica de controladores. Internet das Coisas. Controle-como-um-serviço

Abstract

COSTA, Marcus Vinícius Souza, M.Sc., Universidade Federal de Viçosa, March, 2020. **Mechanism for dynamic resource selection for control-as-a-service provisioning in heterogeneous and dynamic fog computing environments**. Advisor: Vitor Barbosa Carlos de Souza.

The Internet of Things is a technological revolution that brings great possibilities for people, governments, industry, and academia, being one of the pillars of several applications in the modern world. These applications often make intensive use of smart devices to meet the required demands, so they produce a high volume of data, often migrated to data-centers available by large cloud service providers, for storage and processing. The latency observed in this model may not meet time-constrained applications requirements, leading to the emergence of a new computing paradigm: fog computing. One of the main goals of this model is to support the cloud by harnessing the potential of edge devices and providing low latency. An effective fog control plan needs to know the resources available at the edge to realize the benefits provided by this model. Due to a large number of fog devices and the high dynamics and heterogeneity of the underlying resources, this task is challenging and is a focus of interest of the scientific community. A recently proposed approach, so-called control-as-a-service, defines devices to act as controllers, knowing the edge capabilities and selecting the most appropriate ones to perform the service. In this work, we propose a strategy for dynamically selecting potential edge devices to assume the role of fog controller, mapping available resources and providing control-as-a-service. We developed a rank-based model that weighs some key characteristics of devices and selects the most capable of assuming this role. Subsequently, the model was enhanced through two extensions to reduce the number of unnecessary exchanges between controllers, showing efficiency in all evaluated scenarios. The mechanism also showed better use of memory and battery in the controllers than the comparative method.

Keywords: Fog Computing. Ranking. Dynamic controller selection. Internet of Things. Control-as-a-Service

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Nos últimos anos, diversas tarefas têm sido apoiadas pela evolução dos dispositivos. A Internet das Coisas (*Internet of Things* - IoT) é um novo paradigma tecnológico que vem modificando a forma com que as mais diversas aplicações interagem com o ambiente, permitindo maior domínio sobre variadas áreas. Com a IoT, empresas e governos podem oferecer estratégias mais robustas e eficientes para diversas soluções, como casas e cidades inteligentes (trânsito, gestão de alimentos, qualidade do ar e água, etc.). A agroindústria pode mapear características ideais para o plantio. Médicos podem gerenciar melhor as características vitais de seus pacientes. Os consumidores podem receber serviços customizados e ofertas ideais baseadas em sua localização real, dentre várias outras aplicações IoT disponíveis.

Kevin Ashton, pesquisador britânico do Massachusetts Institute of Technology (MIT), comentou em Ashton (2009) que foi ele o primeiro a introduzir o conceito, em 1999, no seu trabalho intitulado "*I made at Procter & Gamble*". O termo inicialmente era vinculado apenas a tecnologia de identificação por radiofrequência (*radio frequency identification* - RFID) e Redes de Sensores sem fio (RSSF). Desde então, os dispositivos aumentaram substancialmente suas capacidades computacionais, a preços relativamente baixos. Processadores mais velozes e maior capacidade de armazenamento permitem que os dispositivos possam processar e armazenar mais dados. Baterias mais eficientes são capazes de garantir maior disponibilidade dos dispositivos. Melhor conectividade permite acessar e estender os recursos computacionais remotamente, bem como mais alternativas para sensoriamento remoto. Alinhado a esta evolução, os dispositivos são utilizados atualmente, em larga escala, para os mais diversos fins. A Figura 1.1 mostra a expansão da IoT e a previsão de crescimento para os próximos anos.

O emprego massivo dos dispositivos produz um aumento significativo no volume de dados. Por exemplo, para gerenciar as características importantes de uma aeronave, 10TB de dados são produzidos em apenas 30 minutos de voo, da mesma forma que 75TB/semana são gerados por um campo de petróleo *offshore* e 1TB/dia por uma grande refinaria de petróleo (Anawar et al., 2018). Só no ano de 2015, uma média mensal de 15PB foram produzidos e trafegados na Internet, por aproximadamente 97 bilhões de *wearables* (Baktir et al., 2017). No mesmo ano, uma média 20TB/dia foram produzidos por sensores meteorológicos, radares, satélites e observatórios em todo o planeta (Alliance, 2015). Boa parte dos dados gerados pelos dispositivos são processados e armazenados em grandes *data-centers* localizados na nuvem.

O modelo de computação de nuvem móvel (*mobile cloud computing* - MCC) é um paradigma de computação para aplicações móveis, em que o processamento e armazenamento de dados são movidos dos dispositivos móveis para plataformas de computação centralizadas e poderosas, localizadas na nuvem (Dinh et al., 2013). Em casos onde possa existir limitação de conectividade com a nuvem entre os dispositi-

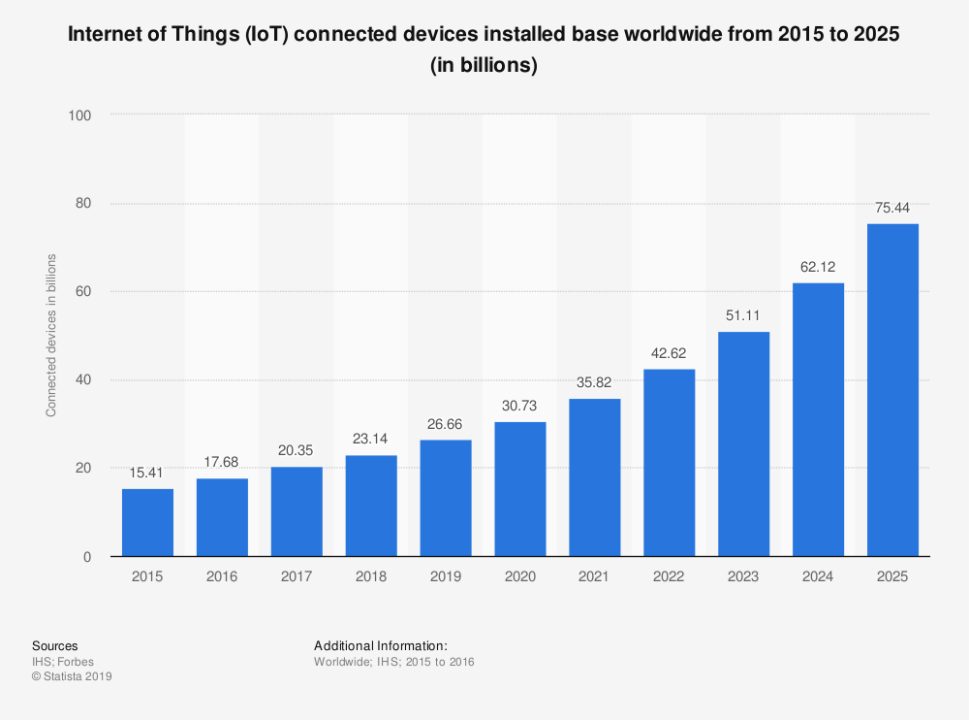


Figura 1.1: Previsão de dispositivos IoT conectados mundialmente, entre 2015 e 2025. Fonte: Statista (2016).

vos, *cloudlets* (*clusters* de computadores ou um computador rico em recursos, conectado(s) à Internet) disponibilizam recursos para atender as demandas das aplicações móveis na borda da rede, oferecendo as capacidades da nuvem para os dispositivos móveis (Satyanarayanan et al., 2009). Existem duas principais abordagens em MCC: arquitetura agente-cliente, onde apenas um *data-center* central fornece recursos e os dispositivos móveis não contribuem para execução do serviço (Nishio et al., 2013) e a arquitetura baseada em cooperação, em que os dispositivos são considerados parte da nuvem, compartilhando seus recursos para suporte do serviço através de comunicação sem fio (Ahmad et al., 2017).

A utilização intensiva da infraestrutura da nuvem é uma estratégia que pode não atender as aplicações de restrição de tempo, como aplicações de tempo real, pois os dispositivos normalmente se encontram geograficamente distantes da infraestrutura da nuvem e este fator é potencialmente um gerador de grandes latências. Para se ter uma ideia, aplicações como *backup offline* de dados, automação doméstica e videovigilância são compatíveis com as latências experienciadas na nuvem. Ao passo que, aplicações que envolvam realidade aumentada, jogos e manufatura em tempo real, têm restrições de tempo que as tornam incompatíveis com a utilização da nuvem, exclusivamente (Byers, 2017).

Uma estratégia adotada, alinhada à necessidade da redução do tráfego entre os dispositivos e a nuvem, é aproveitar o potencial dos dispositivos da borda. *Edge Computing* é um paradigma de computação distribuída, que traz o processamento dos dados mais próximo do usuário ou da fonte geradora deles. Em Bonomi et al. (2012), os autores propõem o novo paradigma de computação distribuída, chamada computação em névoa, que visa cooperar com a nuvem, provendo baixa latência, alta conectividade, segurança, suporte a mobilidade, escalabilidade e heterogeneidade. Os

dispositivos da borda podem ser utilizados para pré-processar e armazenar dados das aplicações e utilizar a infraestrutura da nuvem de forma otimizada. Assim sendo, o principal objetivo da névoa é estender a nuvem e não deixar de utilizá-la por completo. Essencialmente, a névoa é um padrão e a computação de borda é um conceito (Linthicum, 2018).

Aplicações de tempo real, como *streaming* de vídeo, sistemas de monitoramento de saúde, realidade aumentada, *caching* e pré-processamento, trânsito inteligente de veículos, jogos, dentre outras, podem se beneficiar dos benefícios provindos pela computação em névoa ((Dastjerdi et al., 2016), (Anawar et al., 2018)). Como exemplo prático, uma câmera IP de vigilância de 5MP (*mega-pixels*) requer uma largura de banda entre 12 a 16 Mb/s, a 30 FPS (*frames* por segundo) e a contínua transmissão requer 4.5TB de dados, por mês, trafegados até a nuvem. Pelo custo incremental típico de US$ 10/GB praticado pelos planos de companhia de celular, isso significa uma cobrança de US$ 45.000,00 ao mês. Dispositivos da névoa poderiam realizar análises locais e armazenamento de imagem, diminuindo a quantidade de dados trafegados até a nuvem e consequentemente, o preço cobrado (Byers, 2017).

1.1 O problema e a sua importância

Aplicações modernas comumente têm requisitos que demandam transferência de parte dos esforços computacionais para outras estruturas. O *offloading* computacional é uma técnica que permite com que dispositivos possam descarregar suas tarefas em servidores com mais recursos localizados remotamente ou nas proximidades (Liang et al., 2016). Essa estratégia traz algumas vantagens, como economia de energia, por poupar os dispositivos de realizar tarefas intensas que normalmente esgotam as baterias deles (Wang et al., 2013) e a equilibrar o consumo de energia entre os dispositivos (Mtibaa et al., 2013a). Outro fator positivo é a obtenção de baixas latências, por transferir parte do processamento para dispositivos que estejam próximos (Mtibaa et al., 2013b). O potencial de computação dos dispositivos modernos também apoia esta técnica. Aazam et al. (2018) apresentam alguns casos: *smartphones* podem traduzir áudio capturado por óculos inteligentes. Da mesma forma, são capazes de realizar identificação facial de imagens capturadas pelo mesmo dispositivo. Além disso, podem sugerir atividades físicas a usuários, baseadas nas condições físicas individuais deles, ao processar dados capturados por relógios inteligentes. Em uma casa inteligente, um dispositivo inteligente de borda pode guiar um aspirador de pó autônomo caso exista algum resquício de sujeira no ambiente, através de reconhecimento de imagens.

A descoberta de recursos dinâmicos em ambientes de computação distribuída, como a névoa, é um assunto recorrente em pesquisas. Mediante a alta dinamicidade dos dispositivos da borda (mobilidade, carga disponível, etc.) e a heterogeneidade deles (capacidade de processamento, armazenamento, etc.), um plano de controle na névoa deve considerar a descoberta dos recursos ideais para realizar *offloading* de tarefas.

O emprego de controladores de borda, dispositivos responsáveis por mapear os recursos altamente dinâmicos presentes na borda, é uma estratégia que tem sido adotada em alguns trabalhos, em distintos cenários. A utilização deles propicia menor atraso na realização de tarefas de controle, devido a proximidade dos outros dispositivos, ainda que a carga imposta a eles pelo rápido crescimento dos dispositivos IoT

seja uma tarefa desafiadora e possa comprometer a requisitos de qualidade de serviço (*quality-of-service* - QoS) (Jiang et al., 2018). Alinhado a isto, Souza et al. (2017) propuseram um modelo colaborativo e distribuído, chamado de controle-como-um-serviço (*Control-as-a-Service* - CaaS). O modelo prevê um plano de controle dividido em camadas, em que controladores da borda dispostos nessas camadas de forma proporcional aos recursos disponíveis, devem oferecer processamento e armazenamento para manter informações de controle referente aos recursos mais adequados no atendimento das tarefas requisitadas pelos usuários da névoa. O controlador não participa da execução do serviço em si, mas deve conhecer e prover os recursos mais adequados para esta tarefa. No entanto, os autores não propuseram uma estratégia para selecionar os melhores dispositivos para exercer o papel de controlador.

A seleção de controladores na névoa é um desafio, pois existem características dinâmicas na borda a serem analisadas para determinar este papel. Um controlador que tenha alto poder de processamento pode prejudicar o *offloading* de tarefas na borda, devido a fato de que este poderia ser utilizado como um potencial dispositivo para executar rotinas do serviço. Da mesma forma, um nó que tenha baixa memória disponível pode não ter capacidade de manter grandes bancos de dados dos recursos dinâmicos da borda, se controlador. Controladores que tenham baixo nível de bateria ou que estejam em movimentação têm maior probabilidade de indisponibilidade. Assim sendo, é necessária uma estratégia para selecionar controladores eficientes para atuar em cenários dinâmicos.

1.2 Hipótese

Uma abordagem que não considera o dinamismo presente na névoa para realizar trocas de controladores, não atende a requisitos de QoS, como desempenho e disponibilidade. Assim sendo, uma estratégia que seleciona controladores e avalia o cenário, possibilitando o rodízio deles sob demanda, pode evitar possíveis falhas no provimento de CaaS. Além disso, esta estratégia deve empregar controladores que têm mais capacidade de criar maiores bancos de dados dos recursos da borda e menor possibilidade de prejudicar tarefas de *offloading* do serviço, para se adequar melhor as aplicações da névoa.

1.3 Objetivos

O objetivo geral deste trabalho é apresentar uma estratégia de seleção em tempo real de potenciais nós controladores para provimento de CaaS em ambiente de computação dinâmico e heterogêneo de névoa.

1.3.1 Objetivos específicos

- Identificar e classificar as características de interesse (*characteristics of interest* - CI) que são determinantes para avaliar um nó para exercer o papel de controlador.
- Propor um modelo eficaz baseado em classificação que pondera as CI dos nós.
- Avaliar o comportamento do modelo em relação as CI.

- Minimizar o número de trocas desnecessárias dos nós, devido a alta dinamicidade das CI.
- Avaliar a eficiência das técnicas para redução de trocas desnecessárias entre os controladores.

1.4 Estrutura da dissertação

Esta dissertação foi elaborada e escrita de acordo com um dos formatos recomendados pela Comissão do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Viçosa. A mesma se encontra em formato de coletânea dos artigos resultantes do desenvolvimento desta pesquisa. O restante da dissertação está organizada como segue:

O Capítulo 2 consiste de uma versão modificada do artigo Costa et al. (2019), publicado nos anais da *27th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2019)*. Neste trabalho, uma estratégia para seleção de controladores em tempo real para provimento de CaaS em ambiente de computação em névoa é apresentada. O artigo introduz as CI de cada nó para avaliar a viabilidade deles para exercer o papel controlador, bem como os critérios necessários para a troca de controladores.

Os Capítulos 3 e 4 são extensões do artigo anterior. O artigo do Capítulo 3 foi submetido para avaliação em uma conferência nacional e o artigo contido no Capítulo 4 poderá ser submetido em uma conferência ou revista científica da área. Os esforços empenhados neles, em relação ao artigo do Capítulo 2, são para redução do número de trocas desnecessárias entre controladores. O artigo contido no Capítulo 3 propõe uma abordagem de variação dos pesos adotados para ponderar as CI dos nós, considerando a dinamicidade do cenário e se adaptando melhor a ele, em relação a mesma abordagem com o emprego de ponderação fixa. Já o artigo contido no Capítulo 4 propõe uma estratégia simples de predição baseada em previsão dinâmica de desvios para prever nós que variam seus valores de classificação muitas vezes, e em pouco tempo, contribuindo para prevenir a seleção deles para controlador e assim, diminuir o número de trocas desnecessárias. Em ambos os trabalhos, a classificação de algumas CI como memória, processamento e mobilidade também foram aprimoradas em relação ao artigo do Capítulo 2.

Por fim, o Capítulo 5 comenta os resultados e aponta possíveis extensões desta pesquisa.

Capítulo 2

Dynamic Control-as-a-Service provisioning in Fog computing

COSTA, Marcus Vinícius Souza; SOUZA, Vitor Barbosa; JÚNIOR, Sócrates S. Araújo. Dynamic Control-as-a-Service provisioning in Fog computing. Em: International Conference on Software, Telecommunications and Computer Networks (SoftCOM). **Proceedings of the 27th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)**. Split: Croacia, 2019. p. 1-6.

Abstract

The Internet of Things is a technological revolution that is transforming several application domains. It has brought a number of solutions and alternatives and is one of the pillars of Industry 4.0. Since several current and future IoT applications will require near-zero communication latency, Fog computing was recently proposed as an extension of the Cloud paradigm to the edge of the network. In the collaborative scenario envisioned by Fog computing, a fog controller is an entity responsible for mapping end-user service requests into the most suitable resources at the edge of the network. However, given the high dinamicity and heterogeneity perceived in this scenario, we argue for the benefits of the dynamic deployment of fog controllers (Control-as-a-Service) according to, for instance, the amount of underlying resources and their characteristics. This work introduces a rank-based weighted distributed algorithm that considers distinct characteristics of interest to select nodes capable of playing the controller's role at the edge of the network. In the performed experiments, we analyze the convergence time of the proposed algorithm and compare the frequency on controller changes in our approach with an adapted version of the well-known LEACH protocol, which was tailored to fulfill our scenarios' demands. In four distinct scenarios, the results show a reduction range varying from 68% to 95% on the frequency for selecting new controllers when comparing our proposal to the comparative method. Furthermore, the average latency for selecting the controller is low even in scenarios with increased number of candidates.

2.1 Introduction

Since the last decade, the evolution of the Internet of Things (IoT) concept has fueled the deployment of a plethora of connected devices being able to both investigate and act over an environment in different ways. It is predicted that over 35 billion things

will be connected by 2020 (Plummer, 2015). As a consequence of employing IoT devices in large scale, large amounts of data are generated requiring huge processing capacity that usually cannot be found at the edge of the network. To cope with this issue, the centralized structure provided by cloud Data Centers (DCs) have been often employed, producing huge data traffic at the core network whilst impacting negatively on communication latency, specially for real-time applications, due to the large distance between edge and core.

Fog Computing has been recently coined aiming at extending the Cloud Computing paradigm to the edge of the network. Some of its main goals are to decrease communication latency, provide mobility, support scalability and heterogeneity (Bonomi et al., 2012). Fog Computing can be used in distinct scenarios such as SmartGrids, Smart Traffic Lights and Vehicles, Sensor and Actuators Networks, Decentralized Smart Building Control, IoT and Software Defined Networks (SDN) (Stojmenovic and Wen, 2014). Moreover, localized fogs can provide low network latency while enabling real-time processing of IoT applications' data, especially in critical applications such as health-related services (Aazam and Huh, 2014).

In a layered Fog Computing architecture, the lowest level is formed by devices located closer to the end user, as shown in Figure 2.1, possibly reaching billions of units and presenting broad geographical distribution. Devices at this level may assume a vast range of roles, such as, data sensing, user data acquiring, short/medium-term data storage, local processing of low-demanding tasks, acting over the environment, among others. They are the so-called IoT devices, which may or may not present mobility. The middle layer represents the fog layer, which is possibly constituted by several fog domains. Each fog domain embraces devices such as routers, switches, gateways, fog servers, as well as underlying edge resources such as PC's, smartphones, vehicles, among others (Mouradian et al., 2018). Communication between devices at the edge of the network and fog occurs through a Local Area Network (LAN), enabling a near-zero delay communication. On the other hand, the cloud, endowed with large data-centers, is located at the top of the hierarchy providing powerful computing and storage resources at the cost of a higher communication delay due to the need for communicating with lower layers over a Wide Area Network (WAN).

It is worth noting that, albeit several scenarios may benefit from Fog Computing, the limitations inherent to fog resources (e.g., computing capacity and energy availability) make it unable to allocate highly demanding services. Hence, Fog and Cloud cooperation is often the best strategy for supporting heterogeneous service demands. A coordinated management of cloud and fog resources has been proposed recently by Masip-Bruin et al. (2016). In that work, authors propose the so-called Fog-to-Cloud architecture (F2C), a highly distributed architecture aiming at making the most of the low delay provided by fog resources at the edge and the high computing capacity provided by cloud data-centers.

Given the distributed nature of F2C, and consequently the need for a distributed control topology, the work proposed in Souza et al. (2017) unveils the benefits in terms of QoS provisioning arisen from the employment of edge resources for reducing control decisions delay. In the proposed approach, edge resources may play the role of controllers to deliver Control-as-a-Service (CaaS). Thus, by categorizing the underlying infrastructure, one controller can map service requests from fog users into the most suitable resources. Indeed, in a highly dynamic and heterogeneous environment, where the amount of resources may present a big variation in a short term basis,

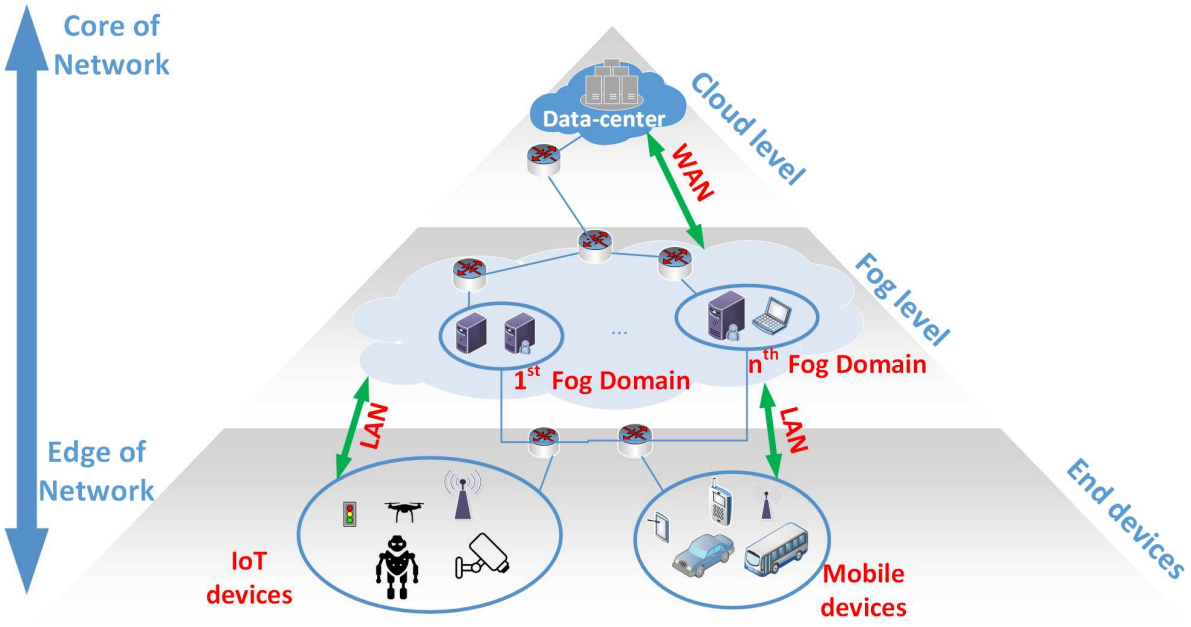


Figura 2.1: Fog Computing Architecture.

the management of all resources by a limited and static number of controllers may not be feasible. Thus, the deployment of controllers close to the edge in a dynamic fashion can have a positive impact on QoS by decreasing control decisions delay.

Nevertheless, designing strategies for selecting the best resource (or resources) to play the controller role in the fog is not a straightforward task. Albeit a number of researches have proposed mechanisms for selecting leaders within groups of devices, most proposed solutions are not suitable for the scenario considered in this work since it presents particularities that make it considerably different from the ones presented by previous works in the literature. For instance, the controller role in this scenario is considerably different from the clusterheads (CH) role in most cluster-based approaches. On one hand, a CH must receive data incoming from other cluster members, then, perform processing and aggregation before sending them to a sink node. Furthermore, in scenarios such as wireless sensor networks (WSN), clusters are formed by sensor nodes, which are usually constrained in terms of both processing and storage capacity. On the other hand, devices in a fog domain are possibly endowed of more powerful resources when compared to sensor nodes. In a collaborative model, enabled by fog computing, these devices are also employed for both processing and storing data rather than being limited to the role of producers. The need for dynamic service allocation in a such volatile scenario enforces controllers to keep updated information regarding underlying resources. However, in traditional cluster-based applications, it is commonly not required from the CH to keep characteristics of each cluster member. The other way around, it is sufficient that cluster members are aware of the CH for task/data offloading purposes.

Finally, since the fog controller must collect information from each underlying resource, the overhead imposed by exchanging controllers due to volatility or resources limitation cannot be neglected. Therefore, a key characteristic of this work is related to the need for a strategy for selecting the best resources for the provisioning of the CaaS concept whilst minimizing the change of controllers, considering the volatility

and heterogeneity inherent of the envisioned scenario. The main contributions of this work are:

- Proposing an effective rank-based model in order to dynamically select the best resources to play the controllers role in dynamic and heterogeneous environments.
- Minimizing the frequency of controller exchanging whilst enabling efficient CaaS provisioning in real-time applications.

This chapter is organized as follows. Section 2.2 reviews some methods based on classification in distributed computing scenarios. Section 2.3 presents a rank-based approach for selecting controllers in dynamic and heterogeneous environments, such as Fog. In Section 2.4, we evaluate the proposed approach with experimental results. Section 2.5 concludes the chapter and suggests possible extensions of this work.

2.2 Background

Research involving the definition of centralizing nodes in data networks is extensive in the literature. Several works are geared towards solving existing problems in WSN, for instance, extending the network life and saving battery power. Towards this direction, Kumar et al. (2008) extends the LEACH protocol to better serve a scenario where the nodes are moving, and Wang et al. (2009) also extends the LEACH protocol in order to tailor the generated CH control.

In Athwani and Vidyarthi (2015), a clustering based approach for Mobile Cloud Computing (MCC) is proposed for selecting the best available resources among mobile devices in a local network. The proposed method uses parameters such as distance, signal strength, mobility, battery power and coverage to create a CH definition function, which is employed by each node. Authors do not consider processing and memory capacity of each node to become CH. In Ettikyala and Latha (2016), authors propose a rank-based scheduler for Cloud Computing that sort the VMs based on their MIPS to adapt the tasks according to the characteristics of each VM that will process them. In Abreu et al. (2018), authors propose a scheduling algorithm for classifying *cloudlets* in federated Fog environments. The classification proposed by authors is based on properties of interest such as latency.

Works that explore the heterogeneity of devices for topology formation in mesh networks can be found in Agnihotri et al. (2016) and Ok et al. (2017). These works classify the devices according to predetermined resources of interest in order to create a metric and assign roles to each device. Both works relate an increase in network lifetime when creating the topology according to devices' characteristics. The approach may be useful in short-range connectivity Machine-to-Machine (M2M) networks.

A number of works in the literature cope with the classification of resources within distinct scenarios. As the reader may notice, in deployments involving heterogeneous devices, the selection of the most suitable resources according to the requirements of each service is mandatory. It is the main factor for obtaining lower latency and better use of the available capacity at the edge of the network.

2.3 An approach to rank-based controller selection in the Fog

In this section, we present a ranking mechanism for selecting a controller in a Fog environment inspired on the so-called "Role Suitability Metric" (RSM), presented by Agnihotri et al. (2016) to cope with topology formation in mesh networks. Since the scenario considered by authors on that paper is distinct from the one considered by us, let us revisit the main duties of a controller in the proposed scenario. A controller is responsible for centralizing the management, selection and allocation of the most suitable resources for the execution of incoming service requests. Notice that the controller is not responsible for executing the service itself. Having this in mind, we believe that if controllers are the ones with maximum computing capacity, offloading highly demanding tasks can be adversely affected due to the lack of resources suitable for services execution. Likewise, if a controller is on the move or has low power status, it should not be an effective controller, as it may become unavailable in a short-term basis. Designing an effective method for controllers election must consider several candidates' characteristics.

In this work, we propose a weight-based metric where distinct resource's characteristics are considered for the rank calculation. In fact, real world deployments embrace devices presenting very high heterogeneity in terms of available memory, processing and mobility, just to name a few. Certainly, these idiosyncrasies must be considered in a highly dynamic and distributed scenario such as Fog.

In the following subsection, we discuss on the rank calculation procedure. Later in this section, the complete flowchart for controller selection is presented.

2.3.1 Rank calculation

Let us assume each device has a set of characteristics $C \subseteq \{c_1, c_2, c_3, ..., c_n\}$ respectively related to a set of weights $W \subseteq \{w_1, w_2, w_3, ..., w_n\}$, where n is the number of considered characteristics and $0 \leq w_i \leq 1$, where $1 \leq i \leq n$.

The rank value (R) for each node is calculated as:

$$R = \sum_{i=1}^{n} c_i * w_i \tag{2.1}$$

The characteristics considered in this work are mobility, processing power, available memory, battery power and signal strength. Each one is comprehensively described in the following lines:

- The mobility (D) metric describes whether a device is moving or not in a given instant. Since rating movement patterns is left for future work, D is given by:

$$D = \begin{cases} 0, & \text{if moving} \\ 100, & \text{if stopped} \end{cases}$$

- Available memory (M) is the amount of memory a node can share while playing the controller's role. In order to be a controller, a node must have enough memory to store underlying resources information, including processing, storage

and battery capacity, as well as mobility. We normalize the values for memory capacity of each node and organize them into five levels. Thus, M is given by:

$$M = \begin{cases} 20, & \text{if } \mathrm{T}_{level1} \\ 40, & \text{if } \mathrm{T}_{level2} \\ 60, & \text{if } \mathrm{T}_{level3} \\ 80, & \text{if } \mathrm{T}_{level4} \\ 100, & \text{if } \mathrm{T}_{level5} \end{cases}$$

- The battery level (B) can be determined by each edge device in a range of 1% to 100%. To prevent the controller from running out of battery, when its level is lower than a pre-established threshold b_{min}, a new controller must be selected. Likewise, a node that has its battery level lower than this threshold cannot be a controller candidate.

- The Signal-to-Noise ratio (SNR) is used for evaluating the signal strength (S) of each node. The SNR value is normalized according to a predefined base value s_{max}.

- The processing capacity (P) of each node is measured in MIPS. Analogously to the available memory, the value for processing performance is normalized and organized into five levels. However, the maximum value (P=100) is attributed to P_{level1}, which is subject to $P_{level1} > p_{min}$, where p_{min} is the lower threshold for P. This approach aims at employing resources that comply with the minimum processing requirements for controllers whilst preserving the most powerful resources for services execution. Hence, P is given by:

$$P = \begin{cases} 100, & \text{if } \mathrm{P}_{level1} \\ 80, & \text{if } \mathrm{P}_{level2} \\ 60, & \text{if } \mathrm{P}_{level3} \\ 40, & \text{if } \mathrm{P}_{level4} \\ 20, & \text{if } \mathrm{P}_{level5} \end{cases}$$

Therefore, the complete equation for obtaining the rank value of each candidate is given by:

$$R = (D * w_1) + (B * w_2) + (M * w_3) + (S * w_4) + (P * w_5) \tag{2.2}$$

2.3.2 Controller selection

In this subsection, we detail the workflow for selecting controllers in a decentralized fashion. The definition of the first controller begins by each candidate node computing its own rank (as previously detailed). It is the first step of the flowchart for controller definition shown in Figure 2.2.

In the next step, each node broadcasts the calculated rank and the received ranks are compared with its own rank value. The candidate with the highest rank will be the controller. Since a candidate must be sure it has the highest rank before declaring itself a controller, after broadcasting its own rank, it sets up a timeout and waits in order to receive other nodes' ranks. After receiving a node's rank, the receiving node compares both ranks. On one hand, if its own rank value is higher than the received one, it resets the timeout, restarting the wait time. On the other hand, if the received rank is higher than its own, the receiving node gives up being a controller by cancelling the

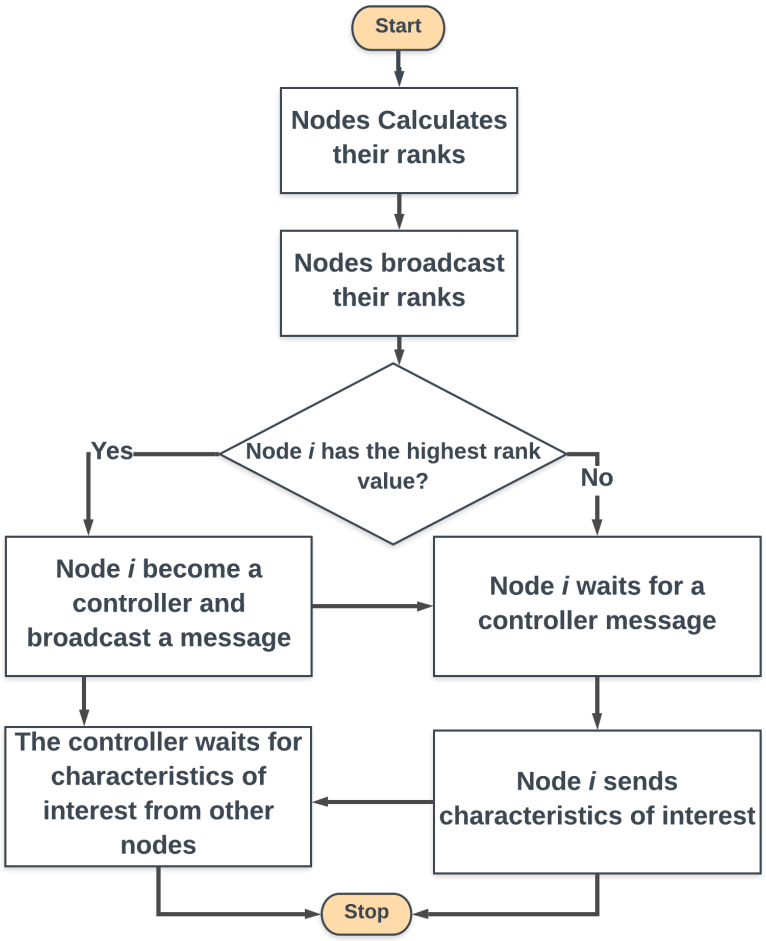


Figura 2.2: Flowchart for controller selection.

scheduled timeout event. All the nodes receive the ranks from other candidates and, when one candidate does not receive any other rank before the timeout, it declares itself a controller. Therefore, before becoming a controller, a candidate resets its count at most *n-1* times, where n is the total number of candidates. Each candidate counts its own timeout, which is restarted each time a received rank is lower than its own rank value. If a tie occurs, the controller will be the one with highest battery level. If the tie persists, random numbers between 0 and 65535, generated by each tied candidate, are compared. Therefore, the tied nodes must exchange one extra message containing those values. As previously mentioned, a node cannot be candidate if $B < b_{min}$ or if $P \leq p_{min}$.

Once the controller is defined, it broadcasts a message to all remaining nodes informing its status. When all nodes know the controller, the investigation phase of the computational capacities located at the edge nodes begins. Nodes must send the controller their characteristics so that the controller can manage resources for end-users service execution. Defining the characteristics of interest of nodes is service dependent and is outside the scope of this work.

Periodically, the controller recalculates and broadcasts its current rank to others, allowing other nodes to compare their current rank value against the controller's one. If any node has a higher rank, it asks the controller for a new election. The controller discovers that there are one or more nodes with better rank than its own and broadcasts a new election message including its current rank. All nodes having a rank higher than the controller will be a candidate for the new election.

2.4 Performance analysis and comparative study

In order to evaluate the proposed model for controller selection, distinct experiments were designed aiming at measuring and comparing its performance. The evaluations were done in three aspects: total delay for selecting the controller, convergence time and the number of resources employed as controllers. Moreover, the model was executed in distinct scenarios. For the sake of comparison, we have tailored the LEACH protocol so that it can be employed in the proposed scenario. The following subsections are devoted to detailing the performed experiments, as well as the adapted version of LEACH.

2.4.1 Nodes' characteristics

The emulated Fog is constituted of devices classified into three distinct types, according to their rank variation rate.

- Type 1 nodes have a higher displacement probably, consequently decreasing its rank due to mobility. Furthermore, it may suffer a higher rank variation due to a varying signal strength. These nodes can be represented, for instance, by vehicles.
- Type 2 nodes are either unlikely to be in motion or present lower displacement, such as embedded devices or smartphones.
- PCs and micro-servers are examples of devices that are not in motion and are categorized as type 3 nodes. Type 3 nodes have less possibility of large variations in their ranks compared to nodes of type 1 and 2, since mobility does not affect the final rank value of nodes of this type.

2.4.2 Calculation of convergence time

The convergence time is the time it takes for a controller to be ready to receive service requests from end-users. In order to declare itself a controller, a node broadcasts a message informing all other nodes about its status. The time for a controller to be discovered by all non-controller nodes (NCN) is given by:

$$T_{ctrl\_discovery} = max\{time(Controller \rightarrow NCN)\} \tag{2.3}$$

where *NCN={x: x is a non-controller node}* and *time* is a generic function that computes the time it takes for a controller to communicate with each node contained in the *NCN* set.

As soon as each node in NCN receives the controller message, it immediately builds a message containing their characteristics of interest and send it to the controller.

The time required for the controller to gather the characteristics of all NCN is given by:

$$T_{ctrl\_buildDB} = \sum_{i=1}^{|NCN|} time(NCN_i \rightarrow Controller) \tag{2.4}$$

where NCN_i is a i-th node element of the set NCN and *time* is a generic function that computes the time it takes to deliver to the controller a message sent by node NCN_i, containing its characteristics. Thus, the convergence time is given by:

$$T_{Convergence} = T_{ctrl\_discovery} + T_{ctrl\_buildDB} \tag{2.5}$$

2.4.3 Adapted LEACH

LEACH (Low-Energy Adaptive Cluster Hierarchy) (Heinzelman et al., 2002) is a protocol that aims at minimizing energy consumption in WSN. Some of its main tasks are the definition of clusters and the selection of respective CHs, by means of a stochastic method. The protocol has two phases: setup phase, where definition of clusters and CHs are performed, and steady phase, where non-CH nodes at each cluster (so-called cluster members) send their data to the respective CH.

In the LEACH protocol, the CH definition is done in rounds. Therefore, given the round number r, the desired probability of a node becomes CH p and the set of nodes that have not played the CH role in the previous rounds G, a threshold $T(n)$ given by (2.6) is computed. In each round, each node generates a random number between 0 and 1 and, if it is smaller than the threshold, the node declares itself a CH and clusters are redefined if necessary. This procedure is repeated at each round and, once a new CH is selected, the previous one must wait $1/p$ rounds to be a CH candidate again.

$$T(n) = \begin{cases} \frac{p}{1-p*(r \bmod p^{-1})} & \text{if } n \in G \\ \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

LEACH is a cluster-based protocol used in ad-hoc networks. However, for the sake of comparison, we have adapted the LEACH protocol so that it can be deployed in our scenario. We employ the same criteria for selecting CHs used in the traditional LEACH protocol, but, since the proposed method deals with an infrastructured network, the nodes are just two hops away from each other. It is left for future work the simulation of larger networks embracing several access points and the need for deploying more than one controller in a single moment. Therefore, LEACH is tailored to operate within one single cluster and only one CH. In order to avoid a tie among nodes declaring as CHs, the node with the lowest random number is wins the content. Similar to proposed method, if a tie persists, battery level and additional random numbers are used for tie braking.

Distinct from LEACH, the rank-based approach proposed in this work does not employ the concept of rounds, in which the exchange of a controller can occur only at the end of a round. On one hand, it is not always easy to predict when a controller may change in the proposed approach. On the other hand, in a highly dynamic scenario, it is important to be able to change the controller as soon as its capabilities turn to be not suitable to play the controller role.

2.4.4 Scenario description

In the conducted experiment, the deployed in-lab testbed emulates a fog environment, where the proposed model is used to dynamically select controllers amongst available devices, allowing CaaS provisioning. In this scenario, devices are connected through

wireless LAN IEEE 802.11 b/g network.

The hardware features in nodes include processing capacity ranging from 1500 to 35000 MIPS and available memory ranging from 150 MB to 3 GB. Features such as SNR, mobility, memory and battery availability can vary throughout the simulation time. Therefore, each node may recalculate its own rank value periodically, for instance, after receiving a beacon message from the current controller. For comparison, the received beacon may include the controller's current rank value.

Table 2.1 shows the general parametric values used in the performed experiment. Table 2.2 shows how we classify available memory and processing capacity of nodes.

Tabela 2.1: Experiment parameters

Number of nodes	10
Experiment time	600 seconds
w1 (weight for mobility)	0.20
w2 (weight for battery level)	0.25
w3 (weight for memory available)	0.25
w4 (weight for signal strength)	0.1
w5 (weight for processing capacity)	0.2
b_{min}	10%
p_{min}	1499
s_{max}	20db

Tabela 2.2: Parameters for available memory (AM) and processing capacity (P)

AM Level	Memory (MB)	P Level	MIPS
T_{level1}	less than 200	P_{level1}	less than 20000
T_{level2}	200 to 400	P_{level2}	20000 to 25000
T_{level3}	401 to 800	P_{level3}	25001 to 30000
T_{level4}	801 to 1024	P_{level4}	30001 to 35000
T_{level5}	more than 1024	P_{level5}	more than 35000

Furthermore, we implement the proposed model in OMNeT++ Simulator in order to assess scalability. The parameters employed in the simulations are analogous to the ones described above, however, the number of nodes are increased so that we can evaluate its impact on the overall latency.

2.4.5 Results

In order to evaluate the amount of controllers exchanges during the experiment time for both rank-based (proposed method) and adapted LEACH approaches, we make use of the in-lab testbed described in the previous section. We consider four distinct scenarios for the rank-based approach: in scenario 1, 100% of the nodes are type 1; in scenario 2, 100% of the nodes are type 2; in scenario 3, 100% of the nodes are type 3; in scenario 4, we assess a merge of the other three scenarios. Hence, in the latter, 30% of type 1 nodes, 30% of type 2 nodes and 40% of type 3 nodes are deployed. As shown in Figure 2.3, the rank-based approach results in a decreased amount of controller changes in all four scenarios when compared to adapted LEACH.

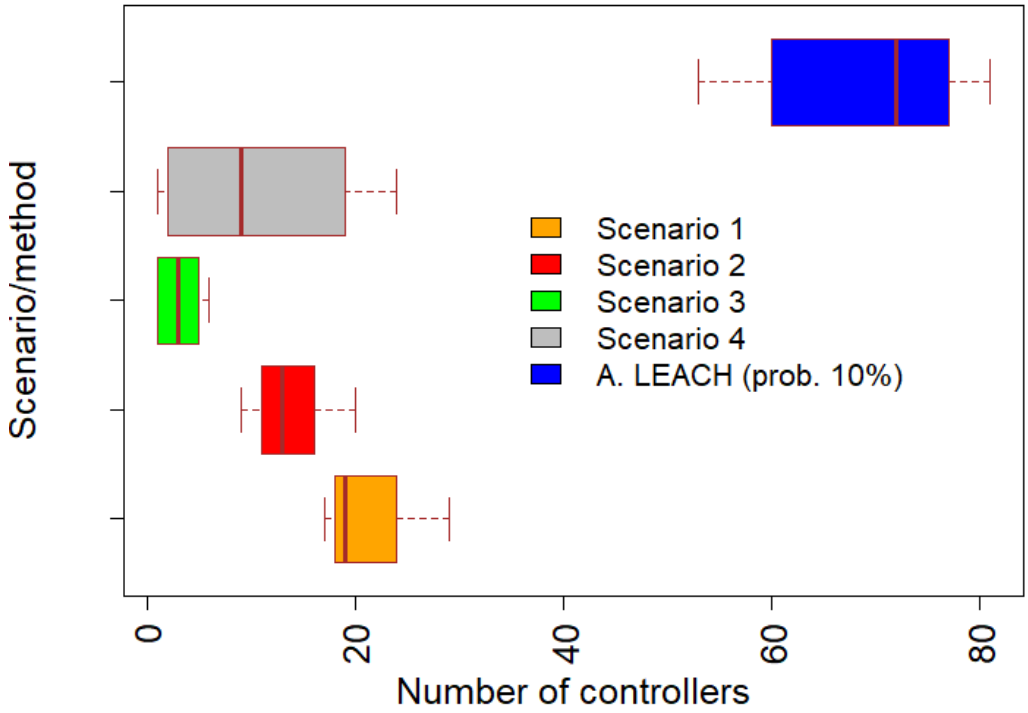


Figura 2.3: Controller changes in different scenarios/methods

It is worth mentioning that one policy for diminishing the amount of changes in the adapted LEACH is to reduce the probability p to a value lower than *10%*. However, it would impact negatively on the first controller selection delay, as well as on the delay for selecting a new controller due to failures on the current one. For sure, real-time applications in Fog require low delay for setting up controllers as well as the reduction on the number of controller changes. By achieving that, we can diminish the frequency for resources discovery by controllers, avoid service disruptions and dropping service requests.

The scalability assessing of the proposed model is twofold: first, the latency variation for selecting a controller when none is active is evaluated according to the increasing number of nodes in the access network; second, we evaluate the impact of number of nodes on the convergence time, which is the time until the controller becomes known by all nodes plus the time until the controller receives the characteristics of interest of each underlying node. The presented results were achieved by carrying out a simulation configured within a 500x500 meters area where the number of nodes vary from from 10 to 160. As shown in Figure 2.4, the average time for selecting the first controller is linear with a variation of approximately 100ms when the number of nodes is increased. It is worth mentioning that, the presented selection time is the total time until the selected node declares itself as controller. Thus, it includes the minimum time it must wait before making sure it has the highest rank value, as described in the proposed model. This timeout is set to 100ms in the performed simulations.

The average time until the controller gets known by underlying nodes and the time until controller builds a database containing the nodes' characteristics of interest is also presented in Figure 2.4. As one may notice, albeit the convergence time is relatively low, it is proportional to the number of underlying nodes and presents a higher increase rate in comparison to the selection time. Indeed, this is expected since the messages containing detailed characteristics of edge resources require higher

transmission time in comparison to the messages containing the rank value, which aggregates resources information. Thus, it evidences the importance of minimizing controller exchanges in order to minimize the impact of convergence time on QoS.

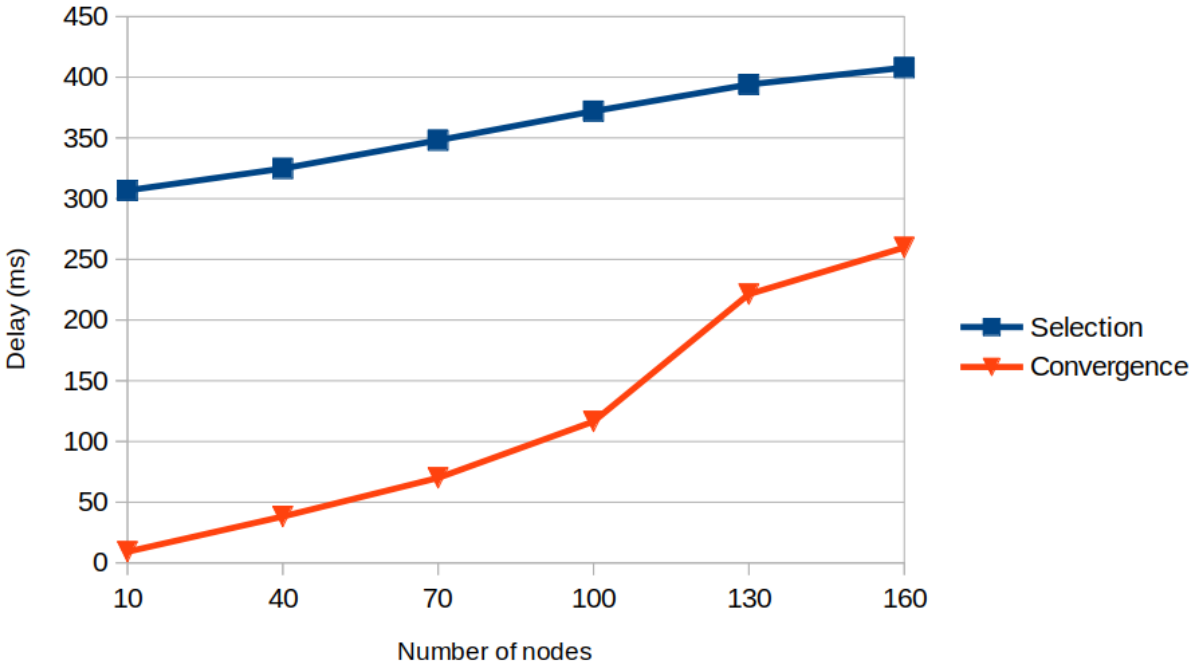


Figura 2.4: Average delay for selection and convergence.

2.5 Conclusion and future work

In this chapter, was presented a distributed rank-based model for on-demand selection of suitable resources for playing the controller role in a fog environment. According to the performed experiments, the rank-based approach produces a significantly lower number of controller changes when compared to a adapted version of LEACH. As a consequence, it reduces the negative impact on QoS produced by the selection and setup of new controllers. As future work, we plan to extend this approach by tuning the weights to best fit distinct scenarios according to its characteristics. Finally, we will work on resiliency techniques, such as the deployment of backup controllers, making the scenario operational when the current controller becomes unavailable.

Capítulo 3

An adaptive rank-based approach for dynamic controller selection in Fog Computing

Artigo submetido
COSTA, Marcus Vinícius Souza; SOUZA, Vitor Barbosa. An adaptive rank-based approach for dynamic controller selection in Fog Computing. Em: XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC).

Abstract

The deployment of a dynamic and cooperative fog control plane, where controllers are selected on-demand among the most suitable underlying resources, has been recently proposed as a Control-as-a-Service (CaaS) model. Albeit it is expected that real-time applications shall benefit from this concept, mechanisms for QoS-aware controllers election is yet an open issue. In this work, we propose an adaptive rank-based controller selection strategy that is capable of tuning the weights employed for each characteristic of interest in order to cope with the environment dynamism. Results have showed an average controller exchange reduction of 37% when compared with a preliminary approach employing fixed weights in a dynamic scenario, as well as its efficiency in battery and memory usage by controllers.

3.1 Introduction

In recent years, smart devices have significantly increased their ability to produce, process and store data, piquing the interest of governments, academia and industry for the development of innovative services, such as smart homes, smart cities, smart agriculture, green energy, e-health, disaster management, among others (Aazam et al., 2018). Aligned to this evolution, the Internet of Things (IoT) brings a number of concepts, strategies and solutions related to devices located at the edge of the network. The number of deployed devices has reached impressive numbers in recent years and is expected to reach 29 billion connected devices by 2022 (Ericsson, 2017). Since these devices may have the ability to both perform remote sensing and operate over environments, data traffic in the network core is facing a huge increase due to the fact that data is traditionally stored and processed in large Data-Centers (DC) infrastructure offered by cloud service providers.

Transferring most of the computational effort required to support applications to the cloud may not be the most appropriate strategy in several cases. Applications requiring low service response time (SRT), such as real time applications, may experience significant delays if the computational capacity is limited to the potential available at the core of the network. Indeed, the exclusive employment of cloud-based architectures may hinder the deployment of several IoT applications presenting strict QoS demands (Byers, 2017).

Recent works have explored edge resource capabilities in order to deploy new approaches for task offloading. The rationale behind these approaches is that several applications may benefit from edge resources as an active part of data processing, reducing communication latency due to the physical proximity among end-points. Fog computing is a novel edge computing paradigm that aims at extending the cloud to the edge of the network. Its main goal is to reduce communication delay whilst supporting system heterogeneity, mobility and scalability (Bonomi et al., 2012). Fog nodes process tasks collaboratively in a hierarchical architecture constituted by fog servers, clients, IoT devices, and users in close proximity, providing computational flexibility, storage, better communication, among others (Anawar et al., 2018).

The burden imposed on edge controllers is a challenging issue that is preventing fog providers to meet QoS requirements (Jiang et al., 2018). Aligned to this, Souza et al. (2017) proposed a distributed and collaborative control model, so-called Control-as-a-Service (CaaS), in order to dynamically deploy a QoS-aware control plane at the edge. Albeit the benefits coming from the deployment of on-demand controllers is shown in that work, authors do not propose a strategy for electing the best resources to play the controller role. In Costa et al. (2019), authors propose an approach for selecting, in real time, the most suitable nodes to play as controllers in dynamic and heterogeneous fog domains. The rank-based approach weighs some key characteristics to measure a node's ability to become a controller while not affecting service performance and minimizing the overhead imposed by the need for controller exchanges. Although that work has presented satisfactory results regarding the convergence time and number of controller exchanges, there is still room for significant improvement. Notice that fog dynamism and heterogeneity can generate excessive number of controller exchanges in a short-term basis. Indeed, a node can have the highest rank value (RV) at one instant, hence, becoming controller, and have its RV decreased in a short time frame, being surpassed by another node, which becomes the new controller. On the other way around, if it subsequently suffers a new RV increase, it shall assume the controller role one more time. Excessive controller exchanges can increase overhead and degrade QoS. In addition, the original weight-based ranking scheme employs fixed weight for each node characteristic, not considering the intrinsic dynamism due to, for instance, the rate of moving nodes or with low battery. As an example, let us consider a scenario where most nodes are significantly constrained in terms of battery capacity. In such case, the weight applied to this characteristic should be increased over others in order to prioritize the selection of candidates presenting higher battery levels.

In order to address this issue, this chapter proposes an adaptive scheme where RV weights are tailored according to the fog scenario demands. Therefore, we propose an approach to select controllers on-demand in fog environments for CaaS provisioning, taking into account the environment characteristics to minimize the number of controller exchanges, decreasing overhead and better meeting QoS requirements. The

main contributions of this work are:

- Extending the rank-based model proposed in Costa et al. (2019) to dynamically tune the employed weights aiming at selecting the best resources to play the controller role according to environment constraints.
- Minimizing the frequency on controller exchanging in order to enable efficient CaaS provisioning whilst meeting QoS requirements in real-time applications.

This chapter is organized as follows. Section 3.2 reviews resource selection proposals in distinct scenarios. Section 3.3 presents a enhanced rank-based approach for selecting controllers in dynamic and heterogeneous environments, such as in fog. In Section 3.4, we evaluate the proposed approach with experimental results. Section 3.5 concludes the chapter and suggests avenues for future work.

3.2 Related work

A number of recent works has been devoted to solutions for the underlying resource selection problem. A widespread approach in the literature is to employ centralized controllers in distributed scenarios seeking to solve different objectives, for instance, optimizing the consumption of computational resources, reducing latency, network mapping, energy efficiency, just to name a few.

In Software Defined Networks (SDN) scenario, Jiménez et al. (2015) propose an resource discovery protocol, so-called SDN-RDP, in order to allow controller nodes to discover the network topology by creating a control layer based on a tree topology. An approach for controller selection in a wireless mesh SDN (wmSDN) architecture is proposed by Salsano et al. (2014). The proposal allows Wireless Mesh Routers (WMR) to select controllers through a priority list.

Works that relate controllers to switches in SDN are found to solve the problem of QoS-guaranteed controller placement problem (Cheng et al., 2015) and minimize overall flow setup time in the network through a switch-controller mapping scheme (Sridharan et al., 2017). The focus of both works is traditional SDN architectures, hence, switch and controller nodes are well defined.

In Lee et al. (2017), authors proposed a framework in which a fog node selects potential neighboring nodes in order to form a dynamic fog network for service offloading. Unfortunately, no strategy for selecting the first node is presented. A dynamic market game for fog environments was formulated by Kim et al. (2018). Authors evaluate economic aspects among end service-users (SU), edge resource owners (ERO) and ISP. The approach considers the ISP as a static controller leveraging dynamic edge resources for services allocation.

A cluster-based approach for resource discovery in Mobile Cloud Computing (MCC) is proposed in Athwani and Vidyarthi (2015). Battery power, signal strength and mobility are parameters used to create a cluster-head (CH) selection function. Each CH manages resources in its cluster in order to select potential nodes for service execution. Authors present no policy to update underlying cluster resources. Arkian et al. (2014) also address the resource selection problem in a vehicular cloud architecture. The cluster-based approach selects vehicles to perform the role of CH using fuzzy logic and reinforcement learning. Nodes' memory and processing capacity are not evaluated for CH selection in both works.

3.3 Enhanced model for controller selection

In this section, a rank-based strategy for selecting on-demand controllers in dynamic and heterogeneous distributed computing scenarios is presented. This strategy is an extension of the one presented in Costa et al. (2019). The following subsection details each characteristic of interest (CI) considered for selecting potential controllers, then, it is presented how CIs are dynamically weighed in order to minimize controller exchanges and, finally, it is discussed on the process of selecting and exchanging controllers.

3.3.1 Identifying controller's requirements

In a distributed computing scenario, one of the main controller duties is to map service requests into the most suitable resources. The controller is not responsible for performing the service itself, but, in a highly dynamic and heterogeneous scenario, it must be able to keep an updated view of the underlying resource characteristics. Moreover, the selection of controllers should occur optimally so that it shall not impair the performance of the service. On one hand, devices with powerful processors and larger memory are better suited for process offloading since service execution may be highly demanding. On the other hand, nodes endowed with low energy resources or presenting high displacement probabilities are more susceptible to unavailability and should not be employed as controllers, preventing QoS degrading.

Therefore, a strategy for selecting controllers has to consider the minimum requirements candidates should present to make them capable of processing and storing data referring to other nodes of the network, without compromising the execution of the service, prioritizing the control function. The key CIs considered in this work for selecting potential controllers are as follows.

- mobility: nodes with higher probability of displacement are more likely to become unavailable, thus, they are less suitable to play the controller role;
- energy capacity: low battery power increases nodes volatility, reducing potential controllers availability;
- signal strength: low signal strength can increase packet loss, QoS degradation and even connection disruption;
- processing and memory capacity: employing the most powerful devices as controllers may reduce the availability of powerful resources for service execution, increasing SRT.

Current edge devices can differ greatly in terms of memory, processing, power capacity and mobility. Therefore, we propose a rank-based metric that weighs the heterogeneous CI mentioned above to select the best nodes to perform the controller role in a dynamic computing scenario.

3.3.2 RV calculation

Each node j has a set of CIs, denoted by $C_j = \{c_1, c_2, c_3, ..., c_n\}$, related to a set of weights $W = \{w_1, w_2, w_3, ..., w_n\}$, where n is the number of considered CIs, $0 \leq c_i \leq$

100 and $0 \leq w_i \leq 1$, where $1 \leq i \leq n$. Therefore, the RV of a candidate node j is given by

$$RV_j = \sum_{i=1}^{n}(c_i \times w_i) - p_j \tag{3.1}$$

where p_j is a punishment factor applied to a node j in order to minimize exchanges and is given by

$$p_j = \begin{cases} 0\,, & \text{if } j \text{ is the current controller} \\ max\{0, (\overline{RV_{diff}})^{-1} \times 100 - (R_c + R_n)\}\,, & \text{if } j \text{ has been a controller earlier} \\ 0\,, & \text{otherwise} \end{cases} \tag{3.2}$$

where $\overline{RV_{diff}}$ is the average difference between RV_j and $RV_{new\_controller}$ considering each time node j releases the controller role in favor of *new_controller*, R_c is the number of consecutive rounds in which node j remained as controller within the last time it assumed that role, and R_n is the number of elapsed rounds since the last time node j released the controller role. As shown in (3.2), the current controller is not affected by the punishment factor. The strategy is to apply greater punishments to nodes that left the controller role with smaller $\overline{RV_{diff}}$ values or assumed as controller only for a few rounds. These nodes are more likely to take over and lose the controller function many times in a few rounds and increase the number of exchanges.

As previously mentioned, this work considers processing, available memory, battery level, mobility, and signal strength as CIs for classifying nodes and selecting the best suited to perform the controller function. In the following lines, a comprehensive discussion on each CI is presented as well as how each one is quantified.

- Available memory (M) is the memory a candidate can share for executing controller duties. To play the controller role, a node must have enough memory to store information regarding resources shared by underlying nodes. The available memory value is classified into six levels. Therefore, M is given by

$$M = \begin{cases} 0, & \text{if } \mathrm{T}_{level1} \\ 20, & \text{if } \mathrm{T}_{level2} \\ 40, & \text{if } \mathrm{T}_{level3} \\ 60, & \text{if } \mathrm{T}_{level4} \\ 80, & \text{if } \mathrm{T}_{level5} \\ 100, & \text{if } \mathrm{T}_{level6} \end{cases} \tag{3.3}$$

- The mobility metric (D) sets higher scores to devices with reduced displacements. It is given by

$$D = 100 - F_d \tag{3.4}$$

  where F_d is a displacement factor of a node, which is a normalized variation of its coordinates – obtained by means of positioning techniques, such as GPS or triangulation – within a time interval $\Delta t = t_2 - t_1$. Therefore, F_d is given by

$$F_d = min\{100, \frac{100}{d_{max}}\sqrt{(x_{t2} - x_{t1})^2 + (y_{t2} - y_{t1})^2}\} \tag{3.5}$$

  where (x_{t1},y_{t1}) and (x_{t2},y_{t2}) are the location coordinates at the instant t_1 and t_2, respectively, while d_{max} is a predefined constraint for normalizing the computed

distance according to the maximum displacement expected or allowed for a device playing the controller role within the interval Δt.

- Battery level (B) is computed by the percentage of total battery b currently available for a candidate – ranging from 1% to 100%. In addition, a controller candidate must have b greater than a predefined threshold b_{min}. Similarly, when a controller has b lower than b_{min}, a new controller must be selected to prevent the current one from running out of battery. Therefore, B is given by

$$B = \begin{cases} 100 \times b\ , & \text{if } b \geq b_{min} \\ 0\ , & \text{otherwise} \end{cases} \tag{3.6}$$

- To measure the signal strength (S) for each node, the Signal-to-Noise Ratio (SNR), normalized according to a predefined base value s_{max}, is used. Hence, S is given by

$$S = min\{100, \frac{100}{s_{max}} \times SNR\} \tag{3.7}$$

- The processing capacity (P) of each candidate is measured in MIPS. Similar to M, the value for processing performance is also classified into levels. However, for processing, eight distinct levels are proposed where the maximum value (P=100) is attributed to P_{level4} whilst the attributed value is reduced as the processing capacity is either increased or decreased. Hence, P is given by

$$P = \begin{cases} 0, & \text{if } \mathrm{P}_{level1} \\ 50, & \text{if } \mathrm{P}_{level2} \\ 75, & \text{if } \mathrm{P}_{level3} \\ 100, & \text{if } \mathrm{P}_{level4} \\ 90, & \text{if } \mathrm{P}_{level5} \\ 60, & \text{if } \mathrm{P}_{level6} \\ 20, & \text{if } \mathrm{P}_{level7} \\ 5, & \text{if } \mathrm{P}_{level8} \end{cases} \tag{3.8}$$

On one hand, P_{level2} is considered the minimum processing capacity required for a candidate to be used as a controller. On the other hand, P_{level4} is the desired processing capacity for the proper execution of controller tasks without loss of QoS due to processing overhead. Our goal is to allocate resources that comply with the minimum processing requirements for controllers while preserving the most powerful resources for services execution.

Therefore, the complete equation for obtaining the RV of each node j is given by

$$RV_j = ((D_j \times w_D) + (M_j \times w_M) + (B_j \times w_B) + (S_j \times w_S) + (P_j \times w_P)) - p_j \tag{3.9}$$

where w_D, w_M, w_B, w_S and w_P are the respective weights applied to D_j, M_j, B_j, S_j and P_j.

3.3.3 Computing RV weights dynamically

As described in the previous subsections, calculating the RV consists in obtaining and classifying nodes' CIs followed by each weight assignment. This is performed after

every time period (hereinafter referred to as rounds) when underlying nodes send their CIs to the current controller. In this work, we propose to define weights dynamically, after each round, according to scenario's demands. This is a challenging task because, in very dynamic environments, CIs can switch quickly. Furthermore, acknowledged the heterogeneity of underlying devices, an improper strategy for weighing CIs shall, for sure, lead to under-utilization of available resources due to the selection of bad candidates to play as controllers, violating QoS requirements.

In Figure 3.1, it is shown an analysis of some CIs variation of five random nodes for each CI in two hundred rounds. The observed nodes have high displacement probability, impacting on large variation of SNR and the employed mobility metric (see (3.4) and (3.5)), as shown respectively in Figure 3.1(c) and Figure 3.1(d). On the other hand, albeit available memory (see Figure 3.1(b)) has had a high variation in some nodes, its contribution to large variations in RV tends to be considerably smaller. Therefore, to minimize unnecessary exchanges due to RV fluctuation, the weights assigned for D and S must be lower. Nevertheless, these CIs cannot be neglected as they are important to assess a node availability at a given time, hence, the respective weights should not be underestimated.

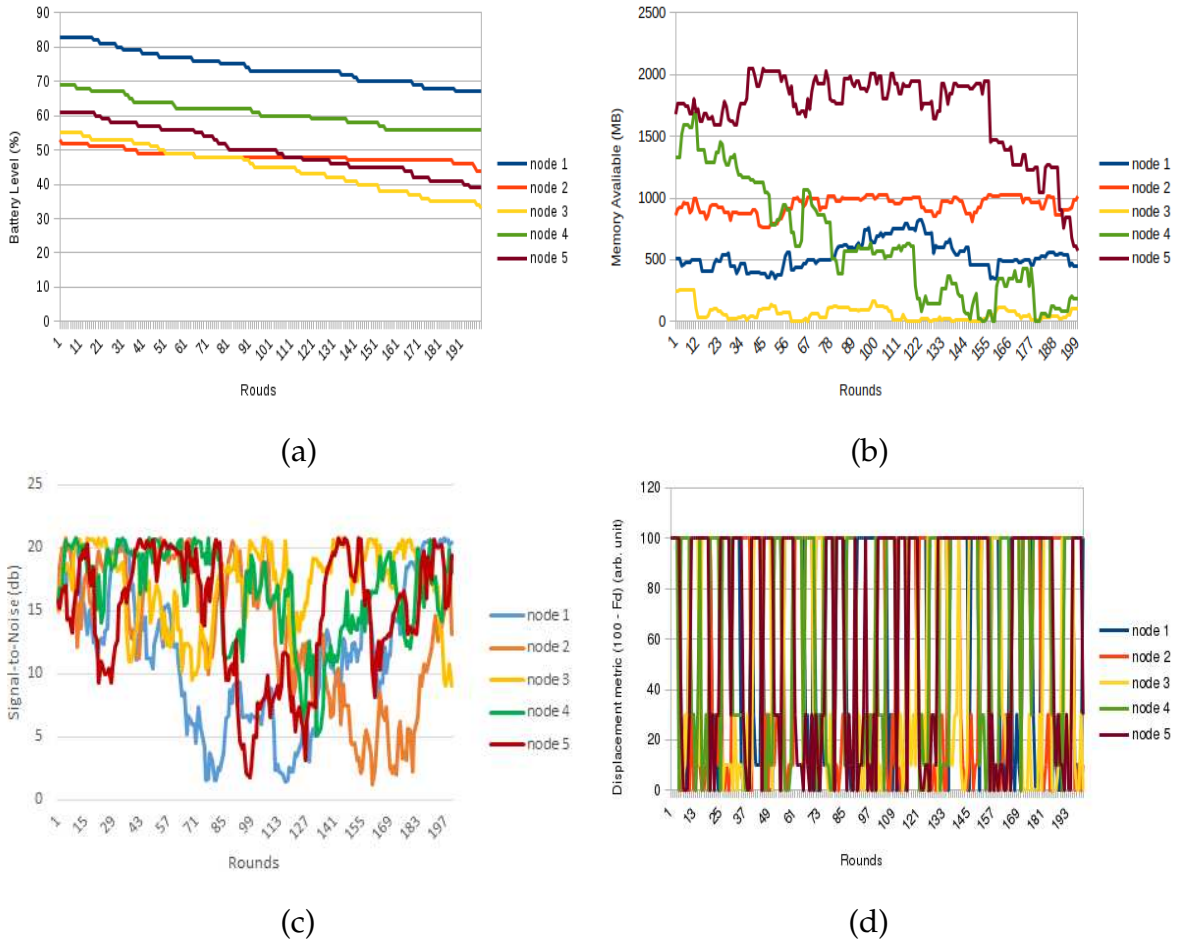


Figura 3.1: Variation of CIs in arbitrary nodes: (a) Battery level, (b) Available memory,(c) SNR and (d) Mobility.

In the deployed strategy, the weight applied to node's mobility (w_D) is scenario-dependent. Hence, the higher the number of nodes with low signal strength, the

higher the weight attributed to this CI. The rationale behind this approach is to prioritize nodes with low mobility since there is higher chance of increasing packet loss due to mobility. Moreover, this strategy reflects the need for prioritizing controllers offering both high availability and, as far as possible, large available memory. The weights dynamically applied to RV computation are given by

$$\begin{cases} w_D \text{ (mobility)} & = & 0.10 + (R_{smin} \times 0.05) \\ w_B \text{ (battery level)} & = & 0.35 + (R_{bmin} \times 0.05) \\ w_M \text{ (memory available)} & = & 0.35 - w_D \\ w_S \text{ (signal strength)} & = & 0.50 - w_B \\ w_P \text{ (processing capacity)} & = & 1 - (w_D + w_B + w_M + w_S) \end{cases} \tag{3.10}$$

where R_{smin} is the ratio of nodes presenting S less than a *low_signal_strength* threshold previously defined and R_{bmin} is the ratio of nodes presenting B less than a predefined *low_battery_level* threshold. Notice that a tradeoff comes to light according to the number of nodes above R_{smin} and R_{bmin} thresholds. In fact, the weight applied for mobility and available memory are inversely proportional. The same occurs between battery and signal strength. In this particular case, a balance is proposed: the battery may increase or maintain its priority over the a node's indicators of displacement (signal strength and mobility) proportionally to the number of nodes below the threshold R_{bmin}.

3.3.4 Controllers selection

This subsection details the decentralized controller selection process for CaaS provisioning as well as the controller exchanging process. Initially, each node calculates its own RV. Since, originally, candidates do not know each other, it is not possible to tailor weight values according to current scenario prior to the first controller selection. Therefore, all nodes assume predefined weight values to compute their RV for the first time. Every node presenting one or more CIs equal to zero give up the selection process. Additionally, since mobility computing takes a time Δt, it is not considered in the first controller election, preventing latency increase. Hence, the mobility weight is initially set to 0 (zero).

Each candidate node broadcasts its calculated RV and waits for the reception of other candidates RVs. Due to the large variation expected on the number of devices, each controller candidate sets a timeout interval in which it listens to other candidates messages. Upon receiving a RV from another node, the candidate compares it with its own RV. If it is greater than or equal to the received one, it resets the timeout timer and continues to wait for messages from other candidates. Otherwise, it gives up being a controller by canceling the scheduled timeout event.

The candidate with the highest RV shall be the only node persisting in the selection after message exchanges. As soon as the timeout event is triggered, the node broadcasts its controller status. With this approach, a candidate must reset its timeout at most *n-1* times before becoming a controller, where n is the number of candidates. In the hypothesis of candidates presenting same RV, the contention is resolved through the highest battery level. Involved nodes exchange extra messages containing the battery level and an extra 2 bytes random number to be used in case of equal battery levels. In this case, the node with largest random number wins the contention. If two

or more nodes announce themselves as controllers at the end of the selection process, the same tie breaking process is employed.

After the elected controller broadcasts its status, the resource discovery phase starts. Within one round, each non-controller node unicasts a message to the controller informing its characteristics, including the CIs used for the controller selection and its punishment factor (calculated by the nodes). Thus, the controller builds its local database so it can map received requests into the most suitable resources according to the service requirements. It is worth mentioning that extra characteristics may be included apart from the characteristics used for controller selection, but, since they are service specific, they are out of the scope of this work.

In addition, non-controller nodes send their CI and punishment factor at each round enabling the controller to keep an up-to-date database. At the end of each round, the controller computes both the amount of nodes with low battery level and the amount of nodes with signal strength bellow the defined threshold and determines the weights for each CI. Then, the weights are employed to calculate the RV of each node, including its own. If one node has a RV higher than the controller's one, the controller unicasts a message informing it shall be the next controller. After acknowledging the message, it announces itself as the new controller. The message sent by the previous controller informs which weights it used in the last round so that the new controller updates its weight values.

In order to prevent $(\overline{RV_{diff}})^{-1}$ from assuming high values, making it impossible for a node to be a controller candidate for hundreds of rounds, controller exchanging takes place only if the node's RV is higher than the controller's RV in at least one unit. The controller exchange is detailed by Algorithm 1. If a tie occurs when selecting the new controller, the same criteria for solving a contention is employed. However, if nodes have same battery level, the current controller randomly chooses its successor.

Algorithm 1 Controller Exchange

```
1:  l: controller node
2:  N: set of non-controller nodes
3:  t: predetermined round time
4:  h: arbitrary node
5:  procedure ControllerExchange( )
6:      while True do
7:          for each node j in N do
8:              l collects j's CI and j's pf within a time t        ▷ performs concurrently
9:          l determines the weights that will be applied to the RV
10:         l calculates all nodes' RV including its own
11:         h ← node with highest RV in l's database
12:         if h's RV > l's RV then                          ▷ h can be a new controller
13:             l informs h that it is the new controller and the weights used in the last round
14:             break
15:     h broadcasts its status to N
16: end
```

3.4 Performance analysis and comparative study

This section presents the experiments deployed for evaluating the proposed strategy. The experiments assess the efficiency in reducing controller exchanges and aspects involving QoS requirements. For instance, controllers are evaluated in terms of both availability and capacity to store underlying resources information. In addition, the controller capability to keep an updated view of those resources CI is further evaluated. For each experiment, we compare, through obtained results, the proposed approach with a tailored version of a resource selection strategy available in the literature.

3.4.1 Comparative resource selection strategy

For the sake of comparison, a CH selection strategy devoted to MCC scenarios, described by Athwani and Vidyarthi (2015), was tailored in order to fulfill the scenario considered in this work. In the presented cluster-based approach, each selected CH is responsible for acquiring and sharing knowledge about the resources available in the respective cluster. For comparison, the proposed scheme is adapted from ad-hoc to an infra-structured communication model. For instance, in the original model, a relative mobility function is computed through the variation of the signal strength between each pair of neighboring nodes. In the infra-structured communication model, the mobility is calculated individually by each node by means of the variation on the strength of beacons received from the access point (AP). Notice that, since the interval between beacon messages is too short, T_s is employed as the minimum time between two beacons considered for computing signal variation. The adapted version is hereinafter referred to as cluster-based model.

Nodes are elected by a cluster function given by

$$Clus_{func} = w * Mob + (1 - w) * (\frac{1}{B_{power}}) \tag{3.11}$$

where Mob is given by the node mobility function, B_{power} is the battery power of a node, w is a weight factor, subject to $0 \leq w \leq 1$. Each node broadcasts its $Clus_{func}$ value and the one with lower value is selected as CH.

Since the focus of this work is not on cluster formation, only one single cluster is considered, with one single CH at a time. Another important difference is that the former work does not consider the need for reelection when nodes' parameters change over time. Rather than that, a new CH takes place only after a node failure or if a node with better $Clus_{func}$ value gets in the cluster. As a result, it reduces the amount of controller exchanges, however QoS degradation is expected since, before a failure, a controller may suffer from low throughput and memory overload for some period, depending on the cause of failure. Consequently, the controller may relinquish service requests.

3.4.2 Scenario Description

The conducted experiments emulate a fog environment with dynamic and heterogeneous devices. Each edge node is emulated by a virtual machine whilst the IEEE 802.11b/g connection among them is emulated by means of NetEm Foundation (2018).

In this work, the same criteria adopted in Costa et al. (2019) is used to classify the nodes regarding their displacement profile, as shown in table 3.1. In this classification, type 1 nodes are more likely to suffer large variations in RV due to mobility and varying signal strength.

Tabela 3.1: Nodes Classification

Node Type	Displacement Probability	Examples
Type 1	High	Vehicles, drones
Type 2	Moderate	Smartphones, embedded devices
Type 3	Low	PCs, micro-servers

Regarding the heterogeneity of nodes characteristics, each node is endowed with processing capacity ranging from 1500 to 35000 MIPS and available memory ranging from 150MB to 3GB. Available memory, signal strength, mobility and battery level are features that can vary throughout simulation time. The memory variation is simulated through a probability density function (PDF) in inverse gamma distribution with shape parameter $\alpha = 0.5$ and scale parameter $\beta = 10$. A random mobility model is used while SNR is calculated according to the current location of the node. For battery consumption, a linear variation is employed. Table 3.2 shows the main parameters employed. It is worth mentioning that the weights shown in the table are used for setting up the emulation start and shall vary throughout the experiment time. The initial weights are inferred from Costa et al. (2019). Table 3.3 shows the adopted classification for memory and processing capabilities of nodes.

Tabela 3.2: Experiment parameters

Number of nodes	100
Experiment time	900 rounds
w_D (weight for mobility)	0.2
w_B (weight for battery level)	0.25
w_M (weight for memory available)	0.25
w_S (weight for signal strength)	0.1
w_P (weight for processing capacity)	0.2
b_{min}	10%
$\Delta t = T_s$	5 rounds
d_{max}	100m
s_{max}	20db
$low\_signal\_strength$ (relative to S metric)	35
$low\_battery\_level$	30%
w (weight factor for cluster-based model)	0.5

3.4.3 Results

In this section, the proposed approach is assessed in three different aspects: number of controller exchanges, accuracy of controller knowledge regarding dynamic node resources, and resources availability on employed controllers. Three distinct scenarios are considered for the proposed model.

Tabela 3.3: Available memory (M) and processing capacity (P) levels

Memory		Processing	
M Level	Memory (MB)	P level	MIPS
T_{level1}	less than 128	P_{level1}	less than 4375
T_{level2}	from 128 to 256	P_{level2}	from 4375 to 8750
T_{level3}	from 257 to 512	P_{level3}	from 8751 to 13125
T_{level4}	from 513 to 768	P_{level4}	from 13126 to 17500
T_{level5}	from 769 to 1024	P_{level5}	from 17501 to 21875
T_{level6}	more than 1024	P_{level6}	from 21876 to 26250
		P_{level7}	from 26251 to 30625
		P_{level8}	more than 30625

- Scenario 1 is constituted by nodes with high displacement probability presenting a distribution of 50% type 1 nodes and 50% type 2 nodes.
- Scenario 2 is constituted by 50% of type 3 nodes, 25% of type 1 nodes, and of 25% type 2 nodes.
- Scenario 3 is fully constituted by devices with low probability of movement, i.e., 100% of type 3 nodes.

In the cluster-based model emulation, all nodes are considered to have high probability of movement since the work in which it is based is focused in scenarios with high mobility. Therefore, the nodes configuration are similar to Scenario 1, i.e., 50% are type 1 nodes while 50% are type 2 nodes.

In the first experiment, the amount of controller exchanges is evaluated. For comparison purposes, both versions of the proposed model are considered. The original version Costa et al. (2019), hereinafter referred to as former model, makes use of fixed weights and no punishment factor, whilst the one proposed in this work implements the punishment factor and adaptive weights. The amount of controller exchanges in the cluster-based model, the former model and the proposed adaptive model is shown in Figure 3.2. For the former and adaptive models three distinct scenarios are considered.

As expected, fewer changes are observed in the cluster-based model since changes only occur when connection fails or battery discharges. The adaptive weights strategy for computing the RV, in collaboration with the punishment factor, led to a decrease of approximately 37% on the amount of controller exchanges in scenario 1, which is more conducive to exchanges, compared to the use of fixed weights. Some examples of the employed weights variation within the experiment time are shown by Figure 3.3.

A mandatory requirement for QoS-aware controllers is to keep up-to-date information regarding highly dynamic resources so that the most suitable ones can be employed for supporting service requests. In the second experiment, the adaptive model is compared with the cluster-based model on this behalf. Distinct characteristics collected by controllers from one non-controller node are shown in Figure 3.4, where a blue line shows the actual value for each characteristic. Within 500 rounds, 70 samples of each CI where randomly collected for both cluster-based and adaptive approaches. The proposed adaptive approach presents better accuracy regarding the

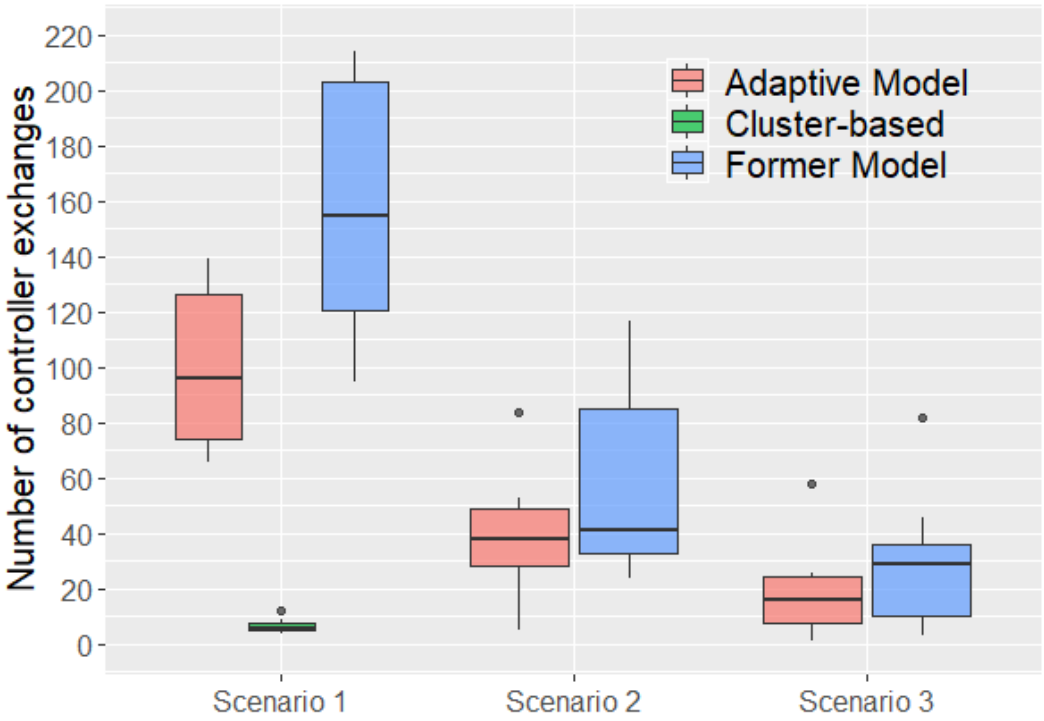
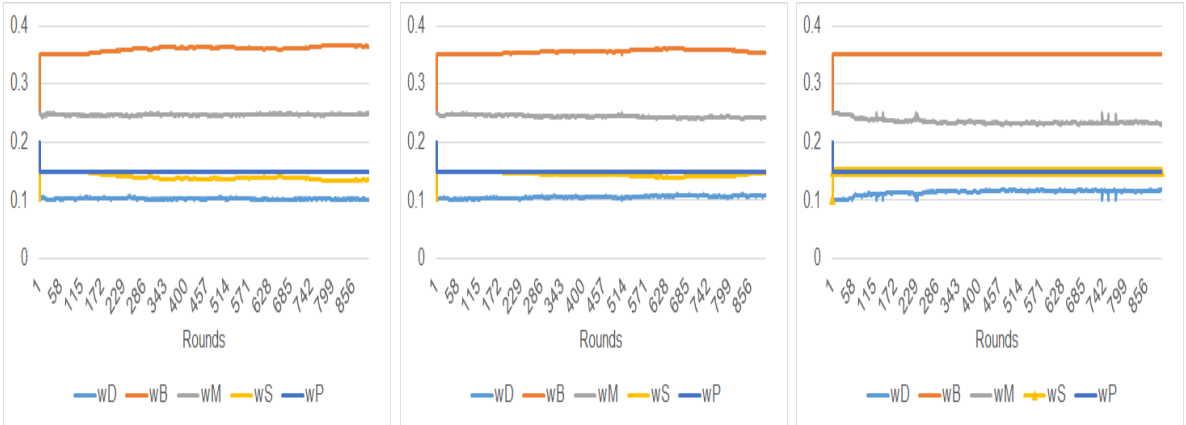


Figura 3.2: Controller exchanges

variations of node characteristics than the comparative method. Indeed, the cluster-based approach expects edge nodes to send messages about their available resources only when a new CH takes place.

A critical requirement is that resources selected to play the controller role do not adversely affect service performance. In the third experiment, characteristics of all deployed controllers, collected at each round, are compared. Figure 3.5 shows battery and memory availability in controllers. As can be seen from the results, the proposed approach is more efficient in selecting potential controllers for all scenarios evaluated than the comparative method. Selecting controllers with higher battery levels enables them to provide CaaS longer. In addition, controllers with more available memory are more capable of managing large databases of edge resources.



(a) (b) (c)

Figura 3.3: Weight variation. (a) Scenario 1, (b) Scenario 2 and (c) Scenario 3

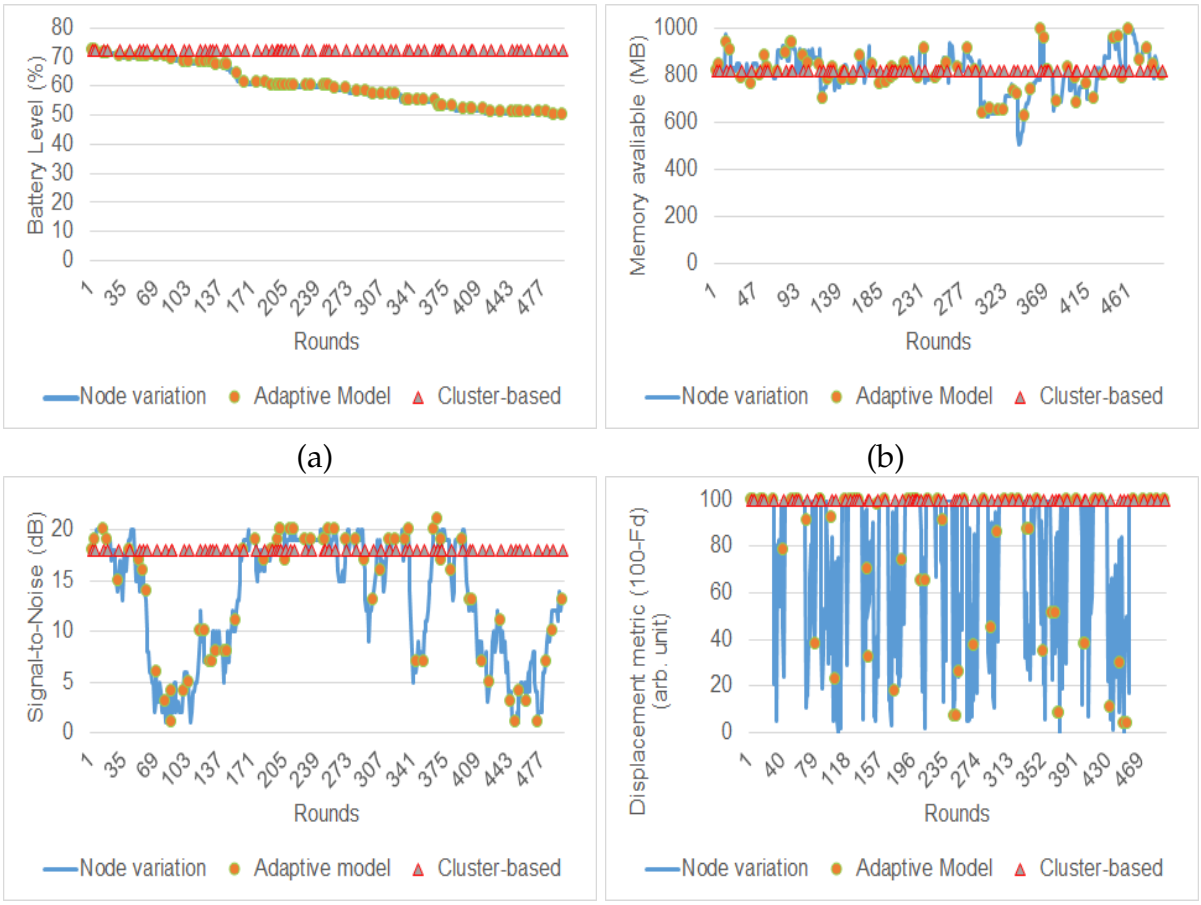


Figura 3.4: Accuracy of controller knowledge of a node's dynamic resources (a) Battery level, (b) Available memory,(c) Signal-to-Noise and (d) Mobility

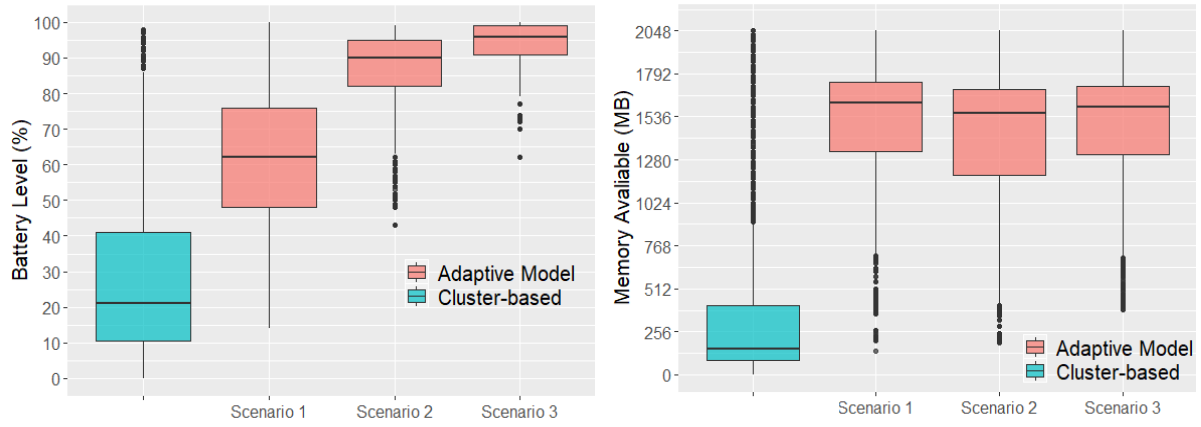


Figura 3.5: (a) Battery level and (b) Memory availability on controllers

3.5 Conclusion

This work proposed an adaptive weighting strategy for rank-based on-demand controller selection aiming at CaaS provisioning in dynamic and heterogeneous scenarios,

such as fog. The proposed model makes use of dynamic weights for the characteristics considered for selection in order to elect QoS-aware controllers whilst minimizing the number of unnecessary controller exchanges. In the performed experiments, this strategy has proved its efficiency when compared with a static weight approach. When compared to a cluster-based model present in the literature, albeit the amount of controller exchanges has shown to be higher, the model proposed by this work has overcome the cluster-based approach both in terms of accuracy of underlying resources information, and efficiency on selecting resources for playing the controller role, offering QoS-aware service allocation. As future work, we aim at addressing the increasing overhead inherent to the need for keeping an updated view of the available resources. We also aim at providing resilience to cope with unexpected controller failures.

Capítulo 4

A simple rank-based prediction strategy for dynamic controller selection in Fog computing

Abstract

Fog computing is a novel paradigm leveraging available resources at the edge of the network for extending the Cloud, promoting low-latency, high scalability, and reducing core network traffic produced by IoT applications. Recognized the unstoppable growth of highly dynamic and heterogeneous connected edge devices, as well as the pop up of diverse services, the provision of some control functionalities intended to select the edge resources best meeting service requirements while also matching the expected QoS guarantees is, with no doubt, a challenging task. This chapter presents a rank-based model aimed at both evaluating edge nodes' characteristics and selecting nodes best performing the controller role, whilst simultaneously satisfying the required QoS constraints, coining the so-called Control-as-a-Service concept. To that end, a yet simple prediction strategy, based on Dynamic Branch Prediction is introduced to avoid unnecessary controller exchanges and QoS degradation. In the performed experiments, the proposed method showed a reduction in the number of exchanges, for a different number of evaluated nodes when compared to the same method with no prediction. When comparing to distinct selection strategies, the proposed model presented an improvement on controller availability as well as on the average controllers battery and memory capacity.

4.1 Introduction

The enormous growth on the number of connected devices observed in the recent years has created several opportunities for innovative services, accommodating people, industry and academia needs and demands. Aligned to this evolution, the Internet of Things (IoT) plays a key role in this scenario by putting together the set of concepts, strategies, services, technologies and solutions linked to the deployment of a huge variety of devices at the edge of the network. Moreover, since a large amount of these devices are employed for effective and low-cost remote sensing over heterogeneous environments, a massive volume of data is produced, resulting in large data traffic that is often processed and also stored when needed in large data-centers (DCs) located at cloud. However, due to the latency imposed by reaching out to cloud premises, some applications requiring low service response time (SRT), such as

real-time applications, may experience undesired issues when relying exclusively on Cloud computing resources.

To address this issue, approaches aimed at making the most out of the potential of edge devices are gaining momentum leveraging the utilization of resources available at the edge to increase scalability while reducing the data load forwarded to cloud (Satyanarayanan, 2017). Fog computing is designed as an edge computing paradigm intended to extend cloud towards the edge of the network, thus decreasing communication latency, supporting mobility, systems heterogeneity and improving scalability (Bonomi et al., 2012). Fog computing applications may fit diverse scenarios, particularly those with strict demands on real-time data processing, such as dependable services in e-health (Aazam and Huh, 2014), or other applications in a diverse set of scenarios, including smart cities, smart vehicles, sensor and actuator networks, SmartGrids, IoT and Cyber-Physical Systems, decentralized control of smart buildings, and software-defined networks (SDN) (Stojmenovic and Wen, 2014).

A key challenge in Fog computing refers to QoS provisioning and is mainly related to the burden imposed on controllers in centralized fog control planes (Jiang et al., 2018). Aligned to the need for real-time resource selection in highly dynamic and volatile fog domains, Souza et al. (2017) introduces a distributed control model, so-called Control-as-a-Service (CaaS), intended to bring the control decisions to the edge. Considering this distributed control model, edge resources may be allocated on-demand to play the controller role, thus, being responsible for both gathering underlying edge resources information and mapping service requests into those resources best matching the service requirements. In a recent work, Costa et al. (2019) presents a rank-based strategy to select the best resources to play the controller role. Albeit the authors show satisfactory results, specially in terms of model convergence time, the frequency observed on controller exchanges (i.e., changes in the controller nodes allocation) may still be relatively high, mainly motivated by the dynamic behavior inherent to the assessed scenario, what undoubtedly turns into a large variation on the computed rank values (RV). Since a controller exchange takes place each time a candidate node's RV is higher than the current controller's RV, a continuous and non-controlled RV variation may result on two or more devices interchanging controller roles repeatedly in a short-term basis. Consequently, albeit the originally proposed controller selection model states that the best controller is the one with highest RV, frequent controller exchanges may degrade QoS and produce an undesired network overhead, for instance, because newly elected controllers are responsible for collecting underlying resources information.

Assuming this behavior not to be optimal, this chapter proposes a RV forecast strategy, based on predicting the RV variation in order to prevent unnecessary controller exchanges. Therefore, the objective of this work is to propose an approach to select potential controllers for CaaS provisioning, considering devices and environments particularities whilst minimizing overhead and QoS degradation by predicting RV variation. The main contributions of this work are:

- Extending the rank-based model proposed in Costa et al. (2019) in order to dynamically select the best resources to play the controller role in dynamic and heterogeneous environments.

- Minimizing the frequency on controller exchanging in order to enable efficient CaaS provisioning whilst meeting QoS requirements in real-time applications.

This chapter is organized as follows. Section 4.2 reviews resource selection proposals in distinct scenarios. Section 4.3 presents an enhanced rank-based approach for selecting controllers in dynamic and heterogeneous environments, such as Fog. In Section 4.4, the proposed approach is evaluated with experimental results. Section 4.5 concludes the chapter and gives some directions for possible future work.

4.2 Related work

The selection of resources to play as centralized controllers in distributed scenarios is a challenging topic in current research. Existing approaches seek to solve different objectives, like energy efficiency, computational resource consumption, reduce latency, and/or network mapping, just to name a few.

Albeit the final purpose of the work presented by Athwani and Vidyarthi (2015) is resource discovery in Mobile Cloud Computing (MCC), an appealing clustering based mechanism is also proposed for clusterhead (CH) selection. Authors use parameters such as signal strength, battery power and mobility to create a CH definition function, which is utilized by each node. Similar to the work proposed in this chapter, the CH is used as a resource manager for discovering and selecting resources for service execution. Arkian et al. (2014) also present a cluster-based approach for resource selection in a vehicular cloud architecture. The strategy presented selects vehicles to perform the role of CH using fuzzy logic and reinforcement learning. In both works, authors do not consider processing and memory capacity of each node to select CH.

Cheng et al. (2015) address the controller placement problem in SDN by mapping controllers under several QoS requirements, where controller candidates are selected according to the number of switches they can serve. Authors in Sridharan et al. (2017) propose a switch-controller mapping scheme, mapping a switch into multiple controllers to distribute flow configuration requests among them and minimize flow configuration time. Since both works focus is traditional SDN architectures, controller and switch nodes are well defined.

In Lee et al. (2017), authors propose an online secretary framework for dynamic fog network formation in which a given fog node selects the most suitable set of neighboring fog nodes for service offloading. However, no policy for selecting the first node is presented. Kim et al. (2018) formulate a dynamic market game that performs an economic analysis among ISP, end service-users (SU), and edge resource owners (ERO) in a fog environment. In the proposed approach, the ISP is a sort of static controller that mediates service requests through the dynamical employment of edge resources.

4.3 Enhanced model for controller selection

This section presents an enhanced rank-based model aimed at selecting a set of suitable controllers to provide CaaS in dynamic and heterogeneous environments, such as Fog. This model is an extension of the preliminary version presented in Costa et al. (2019).

Next subsection briefly discusses about controllers particularities in the envisioned distributed scenario. Subsequently, a strategy for computing candidates' RVs is

presented, as well as the criteria deployed to select the first controller. The section ends discussing when controller exchanges should occur.

4.3.1 Highlighting controller's particularities

Since the controller is responsible for selecting resources for service execution, rather than executing the service itself, it must gather information regarding resources for service execution on the underlying nodes. In addition, the controller election should not consider resources related to specific service execution, such as sensors and actuators. Taking this into consideration, it must be clear that any controller selection mechanism must evaluate service-agnostic characteristics of interest (CI) of each candidate node. In this work, the following CIs are considered, all related to QoS provisioning in terms of either controller availability or latency.

- mobility: nodes with high mobility are more likely to migrate to distinct fog domains and relinquish the controller role;
- processing and memory capacity: the utilization of powerful devices for low latency control decisions is a trade-off for allocating them for low delay service execution;
- battery power: low battery may result in controller unavailability due to discharging;
- signal strength: low signal strength can increase connection disruption probability as well as packet loss, thus compromising the QoS.

This chapter proposes a weight-based metric that makes use of distinct nodes' CIs and computes their RVs in order to discover potential controllers. In real-world deployments, devices may present high heterogeneity in terms of processing, available memory and mobility, among others. It is with no doubt that these idiosyncrasies must be analyzed when selecting the most suitable resources to play the controller role.

4.3.2 RV calculation

Each node j is specified by a set of CIs, denoted by $C_j = \{c_1, c_2, c_3, ..., c_n\}$ respectively related to a set of weights $W = \{w_1, w_2, w_3, ..., w_n\}$, where n is the number of considered CIs, $0 \leq c_i \leq 100$ and $0 \leq w_i \leq 1$, where $1 \leq i \leq n$. Therefore, the RV for a node j is calculated as:

$$RV_j = \sum_{i=1}^{n} c_i \times w_i \tag{4.1}$$

As previously mentioned, the CIs considered in this work for RV calculation are mobility, available memory, battery, signal strength and processing power. The following lines introduces the proposed strategy to compute the value for each CI. It is worth mentioning that a candidate node gives up the election process if any of the CI is equal to 0 (zero).

- The mobility metric (D) contributes so that nodes with a low probability of displacement have higher score values in this evaluation. Therefore, D is given by:

$$D = 100 - F_d \tag{4.2}$$

where F_d is a displacement factor of a node which is a normalized variation of its coordinates. Positioning techniques can be employed, such as triangulation or GPS, within a time interval of $\Delta t = t_2 - t_1$. Hence, F_d is given by:

$$F_d = min\{100, \frac{100}{d_{max}}\sqrt{(x_{t2} - x_{t1})^2 + (y_{t2} - y_{t1})^2}\} \tag{4.3}$$

where (x_{t1},y_{t1}) and (x_{t2},y_{t2}) are the location coordinates at the instants t_1 and t_2, respectively, while d_{max} is a predefined constraint for normalizing the computed distance according to the maximum displacement expected or allowed for a controller within the interval Δt.

- A controller candidate must have enough memory to store information about the underlying edge resources. To classify the available memory (M), six levels are used. Therefore, M is given by:

$$M = \begin{cases} 0, & \text{if } \mathrm{T}_{level1} \\ 20, & \text{if } \mathrm{T}_{level2} \\ 40, & \text{if } \mathrm{T}_{level3} \\ 60, & \text{if } \mathrm{T}_{level4} \\ 80, & \text{if } \mathrm{T}_{level5} \\ 100, & \text{if } \mathrm{T}_{level6} \end{cases} \tag{4.4}$$

- Battery level (B) indicates the battery capacity that a node has in an instant. B is given by the percentage of battery (b) and can vary between 1% to 100%. A node must have b greater than a predefined threshold b_{min} to be able to be a controller candidate. To avoid the unavailability of the controllers, if any of them has b less than b_{min}, a new selection of controllers must be made. Therefore, B is given by:

$$B = \begin{cases} 100 \times b, & \text{if } b \geq b_{min} \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

- To assess signal strength (S) for each node, the signal-to-noise ratio (SNR) is normalized according to s_{max}, a predefined base value. Therefore, S is given by:

$$S = min\{100, \frac{100}{s_{max}} \times SNR\} \tag{4.6}$$

- MIPS is used to classify a candidate's processing capacity (P). As in M, levels are used to classify P. But in this case, eight different levels are proposed, where the maximum value (P = 100) is assigned to P_{level4}, while the assigned value is reduced as the processing capacity is either increased or decreased. Therefore,

P is given by:

$$P = \begin{cases} 0, & \text{if } \mathrm{P}_{level1} \\ 50, & \text{if } \mathrm{P}_{level2} \\ 75, & \text{if } \mathrm{P}_{level3} \\ 100, & \text{if } \mathrm{P}_{level4} \\ 90, & \text{if } \mathrm{P}_{level5} \\ 60, & \text{if } \mathrm{P}_{level6} \\ 20, & \text{if } \mathrm{P}_{level7} \\ 5, & \text{if } \mathrm{P}_{level8} \end{cases} \tag{4.7}$$

In this approach, P_{level2} is considered the minimum processing capacity required for a candidate to be utilized as controller, while P_{level4} is the desired processing capacity for proper execution of controller duties with no QoS loss due to processing overhead. The rationale behind this approach is to employ resources that comply with the minimum processing requirements for controllers whilst preserving the most powerful resources for services execution. For this reason, the processing classification requires more levels than memory, for example, due to the need for a balance in the levels for nodes that have processors unable to perform controller duties and those that have potential processors to perform task offloading.

Therefore, the complete equation for obtaining the RV of each candidate node j is given by:

$$RV_j = (D_j \times w_1) + (M_j \times w_2) + (B_j \times w_3) + (S_j \times w_4) + (P_j \times w_5) \tag{4.8}$$

4.3.3 Controller selection process

This subsection details the steps for selecting and eventually exchanging controllers for dynamic CaaS provisioning. The selection of the first node to play as controller takes place in a decentralized fashion. The CI calculation process starts by all candidate nodes calculating their CIs values separately, and pruning all candidate nodes with one or more CI values are equal to 0. Moreover, in order to avoid increasing the selection delay, the mobility is not considered in the first controller election, since its calculation takes Δt seconds. Hence, the mobility weight is set to 0 (zero).

Then, each remaining node broadcasts its calculated RV and waits for the reception of other candidates' RV. Since this approach deals with dynamic scenarios with varying amount of nodes, after broadcasting its RV, each candidate must set up a timeout interval. When the candidate node receives a message containing a RV, it compares the received RV with its own RV. If its own RV is greater than or equal to the received one, it restarts its timeout timer and keeps waiting for messages from other candidates. In case its RV is lower than the received one, it gives up on being a controller by cancelling the scheduled timeout event.

After messages exchange, all but the candidate with the greatest RV give up, canceling their timeout event. Finally, when the timeout event is triggered in the remaining candidate, it becomes the controller and broadcasts its status. The node with greatest RV resets its timeout timer at most *n-1* times before becoming a controller, where n is the number of candidates. If a tie occurs among two or more candidates, the battery level is used as tiebreaker. Thus, after timeout, involved candidates exchange messages containing battery scores followed by a 2 bytes random number, which is used in case a tie persists after comparing battery values. If two or more nodes announce

themselves as controllers at the end of the selection process, the same tie breaking process is employed.

Once the elected controller broadcasts its status, the investigation phase starts. Each non-controller node unicasts a message to the controller informing about its characteristics, including the CIs used for the controller selection. Thus, the controller builds an internal database so it can map received requests into the most suitable resources according to the service requirements. It is worth mentioning that extra characteristics may be included apart from the characteristics used for controller selection, but, since they are service specific, they are out of the scope of this work. In addition, non-controller nodes send their CI periodically enabling the controller to keep an up-to-date database. The whole process for selecting the first controller is described by Algorithm 2.

Algorithm 2 First controller selection algorithm

```
1: N: set of all nodes
2: C_j: set of all CIs of node j
3: procedure FirstControllerSelection( )
4:     for each node j in N do
5:         j calculates its RV
6:         j broadcasts its RV
7:     nodeId ← node with highest RV in N
8:     for each node j in N do
9:         if j = nodeId then                          ▷ j becomes controller
10:            j broadcasts its controller status
11:            while not received all nodes' CIs do
12:                j listen for node CI
13:                j builds characteristics database
14:        else
15:            j waits for controller announcement
16:            j unicasts C_j to controller
17:
    end
```

Each set of CIs received by a controller within one round is then used to update the underlying nodes information. Furthermore, at the end of each round, the controller calculates the RV of each node, including its own. Since nodes characteristics can change rapidly over time, the RV of each node may vary widely, which might lead to several controller exchanges. Unfortunately, frequent controller exchanges would certainly result in QoS degradation due to frequent messages exchange for the selection itself, followed by the need for building underlying resources database. Consequently, it is a must to design a strategy to reduce such undesired controller exchanges, thus minimizing the impact on the perceived QoS.

Predictive mechanisms can help identify nodes that are rapidly changing RVs, thus preventing them from becoming controllers. Machine learning techniques may be effective for this purpose, but as nodes may have limited memory and processing capacity, other alternatives may be applied to solve this problem. In addition, using complex mechanisms for predicting RV variation in hundreds of candidates simultaneously would certainly introduce a burden that controllers cannot cope with. For this reason, this work proposes to adapt a simple prediction strategy widely used in microprocessors, so-called Dynamic Branch Prediction (DBP). In the proposed approach, the controller keeps a Rank Variation History Table (RVHT) with 2-bit saturating

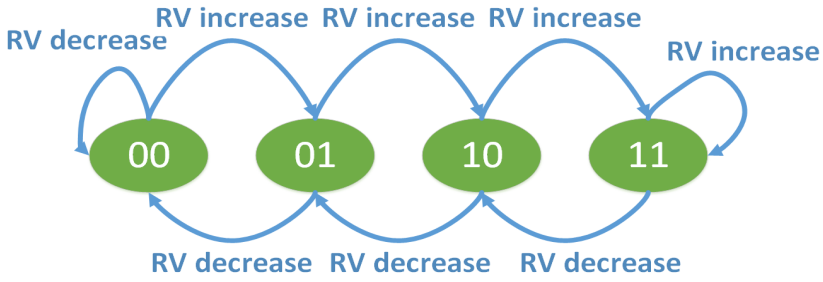


Figura 4.1: RVHT with 2-bit saturating counter for predicting RV variation.

counter for each node, representing 4 distinct levels. Therefore, once the controller receives the CIs of one node and computes its respective RV, it compares the previous RV with the newly calculated one so it can determine the RVHT level of the node. Each RV increase makes the node advance one step closer to the forth level, whilst, each RV decrease results in one step closer to the first level. This is illustrated by Fig. 4.1. States 00, 01, 10 and 11 are related to levels one, two, three and four, respectively.

As the controller calculates each node's RV and updates its internal database, it checks if any RV is greater than its own. Additionally, it checks if the node level in RVHT is equal to level four. Otherwise, it might indicate that if any of these nodes become a controller, it is more likely to be replaced quickly. Therefore, if both comparisons (RV and RVHT level) are satisfied, an exchange may occur. In such case, the current controller notifies the node with greatest RV amongst the ones satisfying both conditions. After receiving the notification, the new controller broadcasts its controller status and the investigation phase restarts. The controller exchange process is described in algorithm 3. Similarly to the first controller selection, if a tie occurs, the current controller makes use of battery level as tiebreaker and, if a tie persist, it randomly chooses the new controller among them.

Algorithm 3 Controller Exchange

```
1: l: controller node
2: N: set of all nodes
3: t: predetermined round time
4: h: arbitrary node
5: procedure ControllerExchange( )
6:     while True do
7:         l collects nodes' CI for t seconds
8:         l calculates all nodes' RV including its own
9:         h ← node with highest RV in l's database
10:        if (h's RV > l's RV and h's RVHT level = 4) or (l's battery level <= b_min) then
11:            l informs h that it is the new controller
12:            break
13:    h broadcasts its status to S
14: end
```

4.4 Performance analysis and comparative study

In order to evaluate the proposed selection strategy, some experiments are carried out to verify the prediction efficiency in reducing unnecessary controller exchanges. Furthermore, the proposed method is also compared with a tailored version of a

distinct election strategy found in the literature. In the following subsection, the adapted version used for comparison is described. Later, the performed experiments and the attained results are presented

4.4.1 Comparative election strategy

The election strategy tailored for comparison purposes is fully described in Athwani and Vidyarthi (2015), where authors present a cluster formation and CH selection strategy for MCC. That work is properly enriched and adapted in this chapter to fulfill the requirements posed by the scenario assessed in this work, shifting from an ad-hoc to an infra-structured communication model. For instance, rather than computing the relative mobility through the signal strength variation of each node according to all other nodes, as proposed in the original work, in an infra-structured scenario, each node computes the signal strength variation regarding the access point (AP) it is connected to through received beacon messages.

Besides node mobility (Mob), the battery power (B_{power}) is also used to calculate its cluster function ($Clus_{func}$). Hence, $Clus_{func}$ is given by:

$$Clus_{func} = w * Mob + (1 - w) * (B_{power})^{-1} \tag{4.9}$$

where w is a weight factor, subject to $0 \leq w \leq 1$. In addition, each node broadcasts its $Clus_{func}$ value and the one with lower value is selected as CH.

Since the usual interval between conventional APs beacon messages is relatively short, the interval between beacons considered for computing signal variation is, at minimum, T_s. Therefore, after listening to the first beacon message, one node must wait at least T_s to consider a new beacon message and compute the signal variation.

It is worth noting that this work considers one fog domain represented as one single cluster with one CH at a time. Another difference is that the original work Athwani and Vidyarthi (2015) does not consider the need for reelection when nodes' parameters change over time. Rather than that, a new CH takes place only after a node failure or if a node with better $Clus_{func}$ value gets in the cluster. On one hand, this approach reduces the number of controller exchanges. On the other hand, QoS degradation may be expected since, before a failure, a controller may present low throughput and memory overload for several seconds. As a consequence, controller may relinquish service requests.

4.4.2 Scenario description

In the conducted simulation experiment, the scenario includes a fog environment where the proposed approach is used to dynamically select controllers amongst available devices as well as the adapted version of CH selection in MCC. The simulations were performed in OMNeT++ simulator using INET framework, where devices are connected through wireless LAN IEEE 802.11 b/g network.

The hardware features in nodes include processing capacity simulation ranging from 1500 to 35000 MIPS and available memory ranging from 150 MB to 3 GB. Features such as SNR, mobility, available memory and battery availability vary throughout the simulation time. Probability density function (PDF) in inverse gamma distribution with shape parameter $\alpha = 0.5$ and scale parameter $\beta = 10$ is used to simulate memory variation. A random mobility model by means of OMNet++ MassMobility module

is utilized while signal power is computed by means of OMNet++ simulator. For battery consumption, a linear variation is employed. Table 4.1 shows the parameter values used for all experiments.

Tabela 4.1: Experiment parameters

Number of nodes	100
Experiment time	900 rounds
w1 (weight for mobility)	0.2
w2 (weight for battery level)	0.25
w3 (weight for memory available)	0.25
w4 (weight for signal strength)	0.1
w5 (weight for processing capacity)	0.2
b_{min}	10%
$\Delta t = T_s$	5 rounds
d_{max}	100m
s_{max}	20db
w (weight factor for comparative model)	0.5

In the parameters, s_{max} is defined as being 20db because it is the moderate quality average of a channel (Lai et al., 2003) and is recommended as being a good value for data networks (Meraki, 2015). As the battery of the nodes has different capacities and the consumption pattern can vary a lot, b_{min} uses 10% as a safe prevention margin for possible controller discharges. Table 4.2 shows nodes' memory and processing capacity classification. Values for weight and Table 4.2 are inferred from Costa et al. (2019).

Tabela 4.2: Available memory (M) and processing capacity (P) levels

Memory		Processing	
M Level	Memory (MB)	P level	MIPS
T_{level1}	less than 128	P_{level1}	less than 4375
T_{level2}	from 128 to 256	P_{level2}	from 4375 to 8750
T_{level3}	from 257 to 512	P_{level3}	from 8751 to 13125
T_{level4}	from 513 to 768	P_{level4}	from 13126 to 17500
T_{level5}	from 769 to 1024	P_{level5}	from 17501 to 21875
T_{level6}	more than 1024	P_{level6}	from 21876 to 26250
		P_{level7}	from 26251 to 30625
		P_{level8}	more than 30625

4.4.3 Results

In order to assess the proposed approach, three distinct aspects are analyzed: number of controller exchanges, controller availability, and characteristics of the used controllers.

The first experiment evaluates the amount of controller exchanges within the simulation time. For the sake of comparison, the proposed method is shown both with and without RVHT prediction. As shown in Fig. 4.2, the prediction strategy showed a reduction in the number of exchanges, for a different number of evaluated nodes.

The number of controller exchanges is compared in Fig. 4.3, where an execution slice between rounds 539 and 571 shows RV variation for nodes 44, 55, 71 and 89, which are the ones that assumed the controller role within this time. The triangle

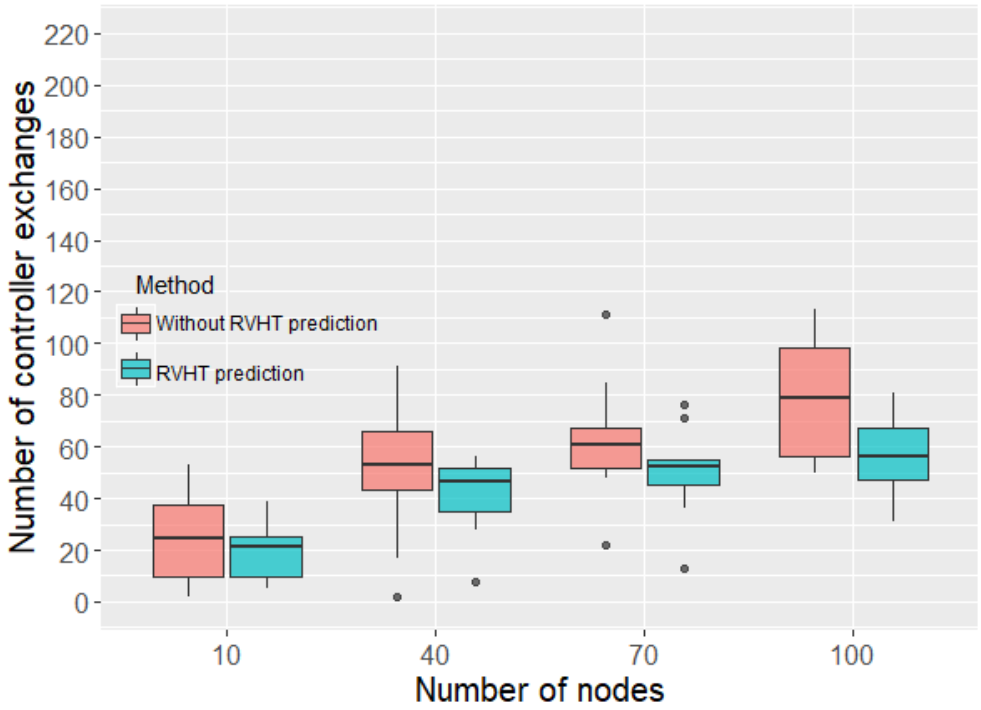


Figura 4.2: Controller exchanges.

marks show the controller in each round. Using the proposed prediction mechanism, one single exchange – from node 55 to 89 – occurs in round 563. On the other hand, when prediction is not employed, 10 exchanges occur in rounds 541, 542, 550, 551, 555, 556, 561, 566, 567 and 569. When considering the complete execution of this experiment, the chance for exchanging controllers after each round without prediction was 11% and approximately 5% when prediction was employed.

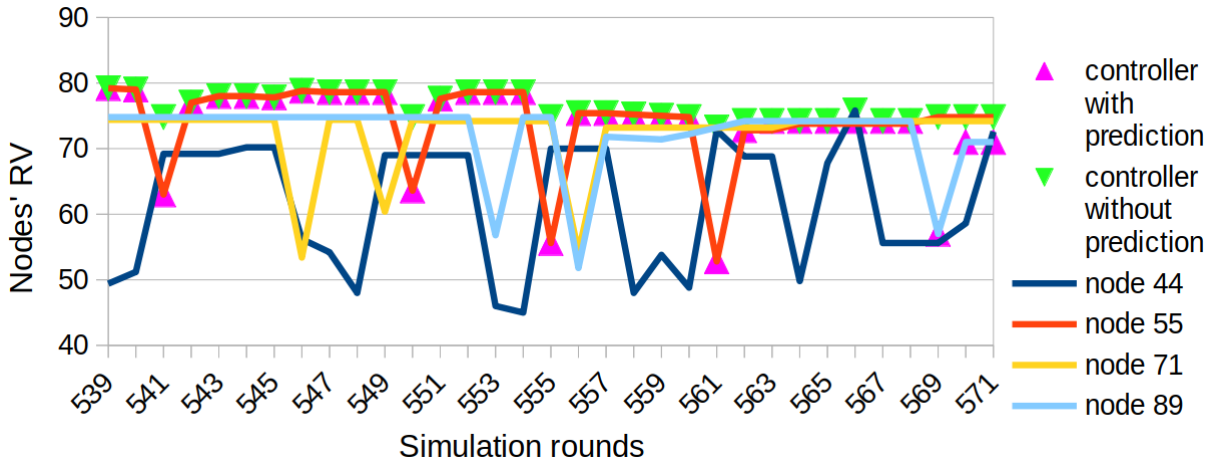


Figura 4.3: Nodes' RV variation over time in the proposed strategy.

The second experiment, intended to check the availability of controllers, evaluates the average number of rounds in which a controller was not available due to lack of battery or connection disruption due to node mobility. As shown in Fig. 4.4, the mechanism used by controllers to keep up-to-date nodes information makes possible to exchange controller before failures occurrence.

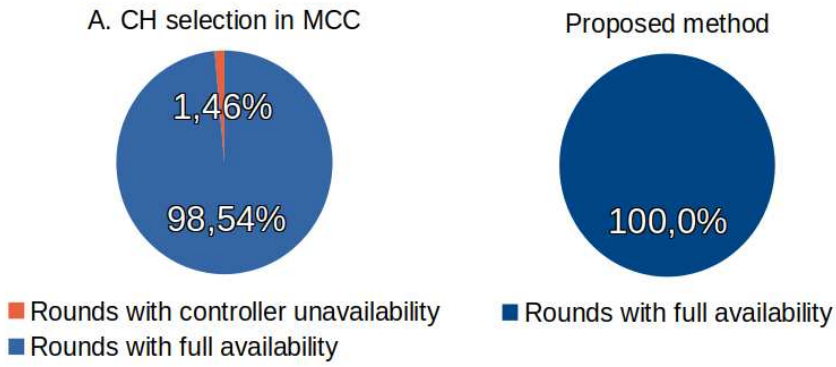
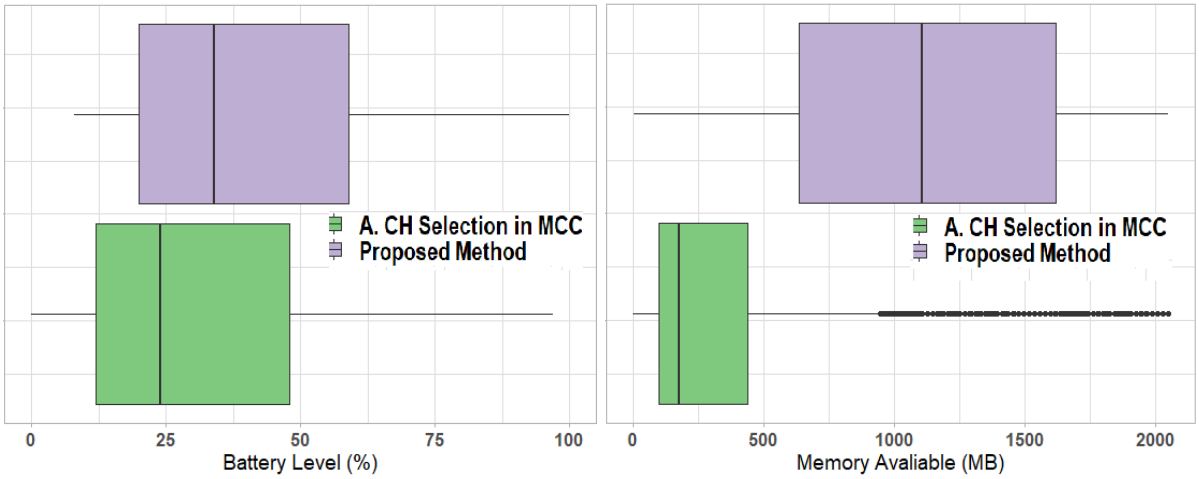


Figura 4.4: Average controller availability.

When selecting resources to play the controller role, a critical requirement is that on-demand selected controllers do not adversely affect service performance. The third experiment compares currently used controllers' characteristics by measuring them in each round. Fig. 4.5(a) and Fig. 4.5(b) use box plots to show, respectively, available battery and memory on controllers.



(a) (b)

Figura 4.5: (a) Battery and (b) Memory availability on controllers.

Notice that the proposed method has selected controllers with higher battery availability over time, enabling long-term CaaS provisioning. It also shows greater effectiveness than the comparative method for selecting controllers that have more available memory. This means that the controllers selected in the proposed approach are more suitable to cope with large edge resource databases.

4.5 Conclusion

In this chapter, an enhanced rank-based strategy with RV variation prediction to select nodes as controllers on-demand was proposed. The performed experiments highlight that the proposed approach shows an effective reduction of controllers exchanges. Albeit an adapted approach for CH selection, used as comparative method, may present fewer controller exchanges, it comes up to present drawbacks such as a higher controller unavailability rate and a notable QoS degradation due to the employment

of less suitable resources as controllers. As a future work, two main directions are proposed. First, is aimed at addressing the effects on the overhead growth motivated by the process of discovering and maintaining dynamic resources in heterogeneous environments. Second direction focuses on providing resilience, coping with unpredictable controllers failures.

Capítulo 5

Conclusões Gerais e Trabalhos Futuros

Nesta dissertação, uma solução para o problema da descoberta de recursos dinâmicos e heterogêneos em cenários de computação em névoa é discutida com base em três artigos resultantes da pesquisa realizada. O modelo apresentado visa mensurar e ponderar determinadas CIs e selecionar nós sob demanda para se tornarem controladores da névoa e mapear os recursos subjacentes, provendo CaaS. Os resultados foram submetidos a experimentos, analisados e apresentados.

O artigo do Capítulo 2 introduz o modelo de classificação discutido nesta dissertação, bem como a forma com que o mapeamento dos recursos da borda ocorre. Em relação ao número de trocas entre controladores, ao ser comparado com a versão adaptada do protocolo LEACH, originalmente utilizado em RSSF, para um cenário infraestruturado, o método proposto obteve uma eficiência de até 95% em um dos cenários avaliados. A estratégia adotada mostrou ser mais eficiente do que a adoção de um modelo estocástico para seleção de nós controladores, haja visto que existe um aumento no tempo de convergência em relação ao número de trocas.

Uma característica observada no modelo apresentado é que, devido a alta dinamicidade que algumas CI podem apresentar, o número de trocas pode aumentar de forma significativa e degradar métricas de QoS, o que motivou os trabalhos referentes aos artigos dos Capítulos 3 e 4. A seleção de novos controladores da névoa, feita pelo controlador corrente, possibilita que ele tenha autonomia para implementar regras mais eficazes para diminuir o número de trocas. Ao adaptar a ponderação aplicada as CI em decorrência da dinamicidade do cenário, com a estratégia adotada no Capítulo 3, foi possível uma maior eficiência na seleção de controladores que têm mais memória e bateria disponíveis. Em relação ao experimento que avalia o número de trocas, a eficácia da adaptação da ponderação aplicada as CI, adicionada do fator de punição adotado na intenção de diminuir a incidência de trocas repentinas, foi comprovada, para o dinamismo de distintos cenários avaliados. Outro experimento realizado neste artigo avalia a acurácia dos controladores selecionados pelo método apresentado na dissertação, em relação ao conhecimento de possíveis recursos subjacentes para execução do serviço. O método comparativo mostrou uma pior acurácia, devido ao fato de que o mesmo apresenta uma política de atualização mais estática, onde um CH apenas atualiza sua base de dados de recursos da borda no momento em que assume este papel. Apesar do fato de que esta estratégia diminui o *overhead*, em cenários muito dinâmicos, o conhecimento do controlador sobre um dispositivo pode diferir em relação ao atual estado deste último, ocasionando a seleção de recursos inapropriados para seleção do serviço e degradar requisitos de QoS.

A estratégia adotada no artigo do Capítulo 4, que conta com uma adaptação da predição dinâmica de desvios utilizando dois bits de saturação, se mostrou mais eficiente do que a estratégia de selecionar sempre o nó que tem o maior valor de classificação, dentre a quantidade de dispositivos avaliados. Como os dispositivos da borda

podem diferir bastante em termos de memória e processamento, estratégias baseadas em *machine learning* podem não ser suportadas e não serem as melhores neste caso, mesmo tendo maior probabilidade de eficiência na diminuição do número de trocas, prevendo e prevenindo que nós indesejáveis se tornem controladores.

Ainda que as duas estratégias utilizadas na intenção de diminuir o número de trocas entre controladores não são melhores do que o método comparativo adaptado de Athwani and Vidyarthi (2015) (Figura 3.2), nos experimentos, foi observado total disponibilidade de controladores, bem como seleção daqueles que têm mais capacidade de armazenamento e bateria. Os resultados mostraram que as trocas, apesar de aumentar o tempo de convergência, devem ocorrer em momentos específicos, para selecionar controladores mais capazes no momento e evitar indisponibilidade do serviço. O método comparativo não implementa uma política de rodízio de CH após a seleção dos primeiros e as trocas ocorrem apenas quando existe falha do CH ou na conexão. Isso demonstra que, para um cenário dinâmico, uma abordagem que identifica e avalia corretamente as CI em tempo real para tomada de decisão é mais eficiente do que a adoção de controladores fixos ou alguma estratégia que não considera a dinamicidade do cenário para prevenir possíveis falhas no serviço.

Como trabalhos futuros, identificamos possíveis aprimoramentos. Um deles é unificar as estratégias adotadas nos artigos dos Capítulos 3 e 4. É possível também desenvolver técnicas de resiliência e permitir com que o mecanismo de seleção de controladores possa lidar com possíveis falhas de controladores. A atualização da base de dados dos recursos subjacentes da borda é uma tarefa computacionalmente cara e uma solução para este problema merece destaque. Nas simulações, o modelo proposto foi analisado para selecionar apenas um controlador por vez, mas em um plano de controle distribuído, mais de um controlador pode ser empregado simultaneamente. Dessa forma, a avaliação da solução integral, considerando a interação entre distintos controladores deverá ser contemplada em trabalhos futuros. A segurança na névoa é tema de interesse em pesquisas, sendo importante apresentar estratégias que impeçam o sucesso de possíveis ataques realizados por agentes maliciosos da borda, que objetivam assumir a identidade de dispositivos controladores ou de potenciais dispositivos que desempenhem *offloading* de tarefas. Por fim, este trabalho pode ser estendido ao nível de serviço, com um protocolo de comunicação a fim de regulamentar todo o processo necessário para seleção e troca de controladores, bem como o mapeamento dos recursos.

Referências Bibliográficas

Aazam, M. and Huh, E. (2014). 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.

Aazam, M., Zeadally, S., and Harras, K. A. (2018). Offloading in fog computing for iot: Review, enabling technologies, and research opportunities. *Future Generation Computer Systems*, 87:278 – 289.

Abreu, D. P., Velasquez, K., Miranda Assis, M. R., Bittencourt, L. F., Curado, M., Monteiro, E., and Madeira, E. (2018). A rank scheduling mechanism for fog environments. In *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 363–369.

Agnihotri, M., Chirikov, R., Militano, F., and Cavdar, C. (2016). Topology formation in mesh networks considering role suitability. In *2016 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 421–427.

Ahmad, A., Paul, A., Khan, M., Jabbar, S., Rathore, M. M. U., Chilamkurti, N., and Min-Allah, N. (2017). Energy efficient hierarchical resource management for mobile cloud computing. *IEEE Transactions on Sustainable Computing*, 2(2):100–112.

Alliance, T. S. (2015). What's the big deal with data? Disponível em: `https://data.bsa.org/wp-content/uploads/2015/12/bsadatastudy_en.pdf`. Acesso: 21 de dezembro de 2019.

Anawar, M. R., Wang, S., Zia, M. A., Jadoon, A. K., Akram, U., and Raza, S. (2018). Fog computing: An overview of big iot data analytics. *Wireless Communications and Mobile Computing*, 2018:7157192:1–7157192:22.

Arkian, H. R., Atani, R. E., and Pourkhalili, A. (2014). A cluster-based vehicular cloud architecture with learning-based resource management. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 162–167.

Ashton, K. (2009). That 'internet of things' thing. *RFID journal*, 22(7):97–114.

Athwani, P. and Vidyarthi, D. P. (2015). Resource discovery in mobile cloud computing: A clustering based approach. In *2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON)*, pages 1–6.

Baktir, A. C., Ozgovde, A., and Ersoy, C. (2017). How can edge computing benefit from software-defined networking: A survey, use cases, and future directions. *IEEE Communications Surveys Tutorials*, 19(4):2359–2391.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). 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. ACM.

Byers, C. C. (2017). Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled iot networks. *IEEE Communications Magazine*, 55(8):14–20.

Cheng, T. Y., Wang, M., and Jia, X. (2015). QoS-guaranteed controller placement in SDN. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Costa, M. V. S., Souza, V. B., and Júnior, S. S. A. (2019). Dynamic control-as-a-service provisioning in fog computing. In *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6.

Dastjerdi, A., Gupta, H., Calheiros, R., Ghosh, S., and Buyya, R. (2016). Chapter 4 - fog computing: principles, architectures, and applications. In Buyya, R. and Dastjerdi, A. V., editors, *Internet of Things*, pages 61 – 75. Morgan Kaufmann.

Dinh, H. T., Lee, C., Niyato, D., and Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611.

Ericsson (2017). Ericsson mobility report. Disponível em: `https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf`. Acesso: 8 de novembro de 2019.

Ettikyala, K. and Latha, Y. V. (2016). Rank based efficient task scheduler for cloud computing. In *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, pages 343–346.

Foundation, L. (2018). Linux foundation wiki: netem. Disponível em: `https://wiki.linuxfoundation.org/networking/netem`. Acesso: 14 de outubro de 2019.

Heinzelman, W. B., Chandrakasan, A. P., and Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4):660–670.

Jiang, Y., Huang, Z., and Tsang, D. H. K. (2018). Challenges and solutions in fog computing orchestration. *IEEE Network*, 32(3):122–129.

Jiménez, Y., Cervelló-Pastor, C., and García, A. (2015). Dynamic resource discovery protocol for software defined networks. *IEEE Communications Letters*, 19(5):743–746.

Kim, D., Lee, H., Song, H., Choi, N., and Yi, Y. (2018). On the economics of fog computing: Inter-play among infrastructure and service providers, users, and edge resource owners. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6.

Kumar, G. S., Vinu, P. M. V., and Jacob, K. P. (2008). Mobility metric based leach-mobile protocol. In *2008 16th International Conference on Advanced Computing and Communications*, pages 248–253.

Lai, D., Manjeshwar, A., Herrmann, F., Uysal-Biyikoglu, E., and Keshavarzian, A. (2003). Measurement and characterization of link quality metrics in energy constrained wireless sensor networks. In *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, volume 1, pages 446–452. IEEE.

Lee, G., Saad, W., and Bennis, M. (2017). An online secretary framework for fog network formation with minimal latency. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6.

Liang, K., Zhao, L., Zhao, X., Wang, Y., and Ou, S. (2016). Joint resource allocation and coordinated computation offloading for fog radio access networks. *China Communications*, 13(2):131–139.

Linthicum, D. (2018). Edge computing vs. fog computing: Definitions and enterprise uses. Disponível em: `https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html`. Acesso: 12 de janeiro de 2020.

Masip-Bruin, X., Marín-Tordera, E., Tashakor, G., Jukan, A., and Ren, G. (2016). Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Communications*, 23(5):120–128.

Meraki, C. (2015). Signal-to-noise ratio (snr) and wireless signal strength - cisco meraki. Disponível em: `https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/Signal-to-Noise_Ratio_(SNR)_and_Wireless_Signal_Strength`. Acesso: 31 de janeiro de 2020.

Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J., and Polakos, P. A. (2018). A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 20(1):416–464.

Mtibaa, A., Fahim, A., Harras, K. A., and Ammar, M. H. (2013a). Towards resource sharing in mobile device clouds: Power balancing across mobile devices. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 51–56. ACM.

Mtibaa, A., Harras, K. A., and Fahim, A. (2013b). Towards computational offloading in mobile device clouds. In *2013 IEEE 5th international conference on cloud computing technology and science*, volume 1, pages 331–338. IEEE.

Nishio, T., Shinkuma, R., Takahashi, T., and Mandayam, N. B. (2013). Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. In *Proceedings of the first international workshop on Mobile cloud computing & networking*, pages 19–26.

Ok, D., Ahmed, F., Agnihotri, M., and Cavdar, C. (2017). Self-organizing mesh topology formation in internet of things with heterogeneous devices. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–5.

Plummer, D. C. (2015). Top strategic predictions for 2016 and beyond:the future is a digital thing. Technical report, Gartner.

Salsano, S., Siracusano, G., Detti, A., Pisa, C., Ventre, P. L., and Blefari-Melazzi, N. (2014). Controller selection in a wireless mesh SDN under network partitioning and merging scenarios. *CoRR*, abs/1406.2470.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1):30–39.

Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23.

Souza, V. B., Gomez, A., Masip-Bruin, X., Marin-Tordera, E., and Garcia, J. (2017). Towards a fog-to-cloud control topology for qos-aware end-to-end communication. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–5.

Sridharan, V., Gurusamy, M., and Truong-Huu, T. (2017). On multiple controller mapping in software defined networks with resilience constraints. *IEEE Communications Letters*, 21(8):1763–1766.

Statista (2016). Iot: number of connected devices worldwide 2012-2025. Disponível em: `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`. Acesso: 21 de novembro de 2019.

Stojmenovic, I. and Wen, S. (2014). The fog computing paradigm: Scenarios and security issues. In *2014 Federated Conference on Computer Science and Information Systems*, pages 1–8.

Wang, W., Wang, Q., Luo, W., Sheng, M., Wu, W., and Hao, L. (2009). Leach-h: An improved routing protocol for collaborative sensing networks. In *2009 International Conference on Wireless Communications Signal Processing*, pages 1–5.

Wang, Y., Lin, X., and Pedram, M. (2013). A bayesian game formulation of power dissipation and response time minimization in a mobile cloud computing system. In *Proceedings of the 2013 IEEE Second International Conference on Mobile Services*, MS '13, page 7–14, USA. IEEE Computer Society.