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

HÉLIO CARLOS BRAUNER FILHO

A Arquitetura RD para Seleção de Sensores IoT Baseada em Qualidade de Contexto no paradigma Edge Computing

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

Orientador: Prof. Dr. Claudio Fernando Resin Geyer

Porto Alegre
2020

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Brauner Filho, Hélio Carlos

A Arquitetura RD para Seleção de Sensores IoT Baseada em Qualidade de Contexto no paradigma Edge Computing / Hélio Carlos Brauner Filho. – Porto Alegre: PPGC da UFRGS, 2020.

130 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Orientador: Claudio Fernando Resin Geyer.

1. Quality of Context. 2. Edge Computing. 3. Sensor Selection. 4. Context-Awareness. 5. Internet of Things. 6. Ubiquitous Computing. I. Geyer, Claudio Fernando Resin. II. Título.

“Habe nun, ach! Philosophie,
Juristerei und Medizin,
Und leider auch Theologie
Durchaus studiert, mit heißem Bemühn.
Da steh ich nun, ich armer Tor!
Und bin so klug als wie zuvor [...]”

— JOHANN WOLFGANG VON GOETHE

AGRADECIMENTOS

Agradeço a minha família, a minha namorada, a meu orientador, Claudio Geyer; a meu primo, Vítor; a meus colegas de GPPD, ao CNPQ e a todos que contribuíram de alguma forma para a elaboração deste trabalho.

RESUMO

Com a adoção em massa da *Internet das Coisas*, alcançando milhões de dispositivos e com crescimento previsto para alcançar vários bilhões de dispositivos até 2025, as possibilidades de uso para as mais variadas aplicações que utilizem os dados gerados e sensoriados por estes dispositivos são cada vez maiores e mais diversificadas. Neste contexto, surge a necessidade para os programadores de aplicação de encontrar e selecionar os sensores que melhor se adequem aos requisitos da aplicação que pretendem implementar, promovendo um grande aumento no tráfego de dados e uso da infraestrutura de rede, gerando preocupações mesmo para as grandes empresas do ramo de telecomunicações.

É neste contexto que é proposta a *Arquitetura RD* para Seleção de Sensores *IoT* Baseada em *Qualidade de Contexto*, a qual permite selecionar diferentes tipos de sensores disponíveis na rede de acordo com as diferentes propriedades e condições dos dispositivos e da informação contextual gerada por estes dispositivos.

O objetivo é reduzir o tráfego de rede resultante das operações envolvidas nesta seleção seguindo o paradigma *Edge Computing*, e facilitando o trabalho dos programadores de aplicação em informar seus requisitos pelo uso do sistema de *Perfis de QoC* (*Quality of Context*), buscando ser uma ferramenta completa para a solução dos problemas envolvidos nestas atividades.

Palavras-chave: Quality of Context. Edge Computing. Sensor Selection. Context-Awareness. Internet of Things. Ubiquitous Computing.

The *RD Architecture* for Quality of Context based IoT Sensor Selection

ABSTRACT

The mass adoption of the Internet of Things, with millions of connected devices, and several billion predicted to be connected by 2025, enables greater and more diversified possibilities of use in a wide range of applications that might use data sensored and generated by IoT-capable devices. In this context the need felt by application programmers for finding and selecting sensors that best fit the requirements of an intended application arises, dramatically increasing network data traffic and burdening network infrastructure, to the point of raising concern even among large telecommunication companies.
This scenario led to the proposal of the *RD Architecture* for Quality of Context (QoC) based IoT Sensor Selection, which allows for the selection of different sensor types that are available throughout the Internet, according to different properties, device conditions and context information generated by such devices, while looking to reduce the network traffic that ensues from the operations involved in the selection process through adoption of the Edge Computing paradigm. Another goal of the architecture is to make it easier for application programmers to translate requirements that must be met in applications they develop. This is achieved through the *QoC Profile* system, thus allowing the RD architecture to strive for being a complete tool in the solution of shortcomings related to sensor selection activities.

Keywords: Quality of Context, Edge Computing, Sensor Selection, Context-Awareness, Internet of Things, Ubiquitous Computing.

LISTA DE ABREVIATURAS E SIGLAS

AHP	Analytic Hierarchical Process
ART	Android Runtime
BAN	Body Area Network
CA4IoT	Context-Awareness for Internet of Things
CASSARAM	Context-aware sensor search, selection and ranking model
CDN	Content Delivery Network
CEO	Chief Executive Officer
CPU	Central Processing Unit
CPWI	Comparative Priority-based Weighted Index
CUIDA	Contexto Ubíquo Interno em Domicílios Assistidos
EDB	Euclidean-Distance Based
GHz	Giga-Hertz
GPS	Global Positioning System
GPU	Graphics Processing Unit
GraphQL	Graph Query Language
GSM	Global System for Mobile Communications
HCI	Human-Computer Interaction
IDE	Integrated Development Environment
IoT	Internet of Things
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JVM	Java Virtual Machine
LAN	Local Area Network
NIST	National Institute of Standards and Technology

OpenCOPI	Open COntext Platform Integration
PriEsT	Priority Estimation Tool
QoC	Quality of Context
QoCIM	Quality of Context Information Model
QoD	Quality of Device
QoS	Quality of Service
RACS	Radio Applications Cloud Server
RAM	Random Access Memory
RFID	Radio-Frequency IDentification
RMSE	Root Mean Squared Error
ROM	Read-Only Memory
SPARQL	SPARQL Protocol and RDF Query Language (acrônimo recursivo)
TAM	Technology Acceptance Model
UML	Unified Modeling Language
UPC	User Preference Capture
VANET	Vehicular Ad-hoc Network
W3C	World Wide Web Consortium
WAVG	Weighted Average
XML	eXtended Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Estimativas relativas à quantidade de dispositivos com capacidade *IoT* (*Internet of Things*) em 2015 indicavam a existência de cerca de 15,4 bilhões destes dispositivos, com crescimento esperado para uma marca superior a 30 bilhões de dispositivos em 2020, e até a marca de 75 bilhões em 2025 (IHS TECHNOLOGY, 2018). Outras fontes citam a marca de 7 bilhões de dispositivos em 2018, sem incluir smartphones, tablets, laptops e telefones de linha fixa (IOT ANALYTICS, 2018).

Mesmo com divergência nos números, é possível perceber convergência na visão das diferentes fontes indicando aumento expressivo do número de dispositivos, bem como um já grande número de dispositivos deste tipo em atividade. À medida que o número de sensores for crescendo, a funcionalidade de busca de sensores se tornará cada vez mais importante (PERERA et al., 2013).

Para realizar estas buscas, é necessário que usuários interessados em determinados tipos de sensores selecionem características de interesse para que sejam filtrados e encontrados sensores apropriados (PERERA et al., 2013). Este conceito pode ser associado a outro conceito já existente, *QoC* (*Quality of Context*), que, embora originalmente tenha sido definido como "*qualquer informação que descreva a qualidade da informação usada como informação de contexto*" (BUCHHOLZ; SCHIFFERS, 2003) e que "*se refere à informação, e não ao processo ou componente de hardware que venha a fornecer a informação*" (BUCHHOLZ; SCHIFFERS, 2003), evoluiu para permitir avaliação de diferentes propriedades de contexto utilizando fórmulas matemáticas (KIM; LEE, 2006), o que permite atribuir um valor geral para a qualidade de informação de um sensor (NAZÁRIO et al., 2015).

Com base na qualidade do sensor de acordo com os critérios desejados, é possível realizar um ranqueamento de sensores utilizando uma função de agregação e um algoritmo de ranqueamento (PERERA et al., 2013). No entanto, para indexar e ranquear todos os sensores possíveis, é necessário que sejam aplicados muitos recursos computacionais (PERERA et al., 2013), e há também geração de um intenso fluxo de dados trafegando nas redes computacionais (CISCO, 2018).

Para lidar com a questão do uso de recursos nos servidores, bem como tentar diminuir o tráfego de dados na rede, é proposto o uso de *Edge Computing*, considerado o modelo ideal para *IoT* quando os dados são coletados na borda (*edge*) (CISCO, 2018), quando há milhares ou milhões de dispositivos espalhados em uma larga área geográfica

gerando estes dados (CISCO, 2018) e/ou os dados precisam ser analisados e uma atuação precisa ser feita, por exemplo, em menos de um segundo (CISCO, 2018).

Levando em consideração esta proposta e as demais características, o principal problema de pesquisa pode ser definido como:

- ***É possível reduzir o tráfego de dados na rede para aplicações que dependam de seleção de sensores utilizando QoC e Edge Computing, sem que haja um aumento proibitivo, a ponto de atrasar computações por esgotamento, do uso dos outros recursos computacionais do dispositivo Edge?***

Além deste, como problemas secundários temos:

- *É viável utilizar dispositivos de borda para realizar parte das funções de cálculo QoC, considerando seus recursos limitados em comparação a sistemas computacionais tradicionais e Cloud?*
- *É possível facilitar a tarefa do usuário que definirá os critérios de avaliação QoC através de padronização de definição de critérios?*
- *Qual o desempenho computacional das funções de agregação utilizadas em trabalhos diversos? Alguma destas funções é superior às outras, em termos de qualidade do ranqueamento e tempo computacional, podendo ser adotada como padrão?*

Procurando responder ao maior número de questões dentre as apresentadas, foi criada a arquitetura *RD*, que permite a distribuição de seus diferentes módulos entre os dispositivos centrais e os *Edge Devices*, permitindo validar diferentes configurações para a avaliação de *QoC*.

Além disso, a RD conta com um sistema de *Perfis de QoC* que permite realizar sugestões de definições de *QoC* aos programadores de aplicações, visando facilitar o processo de definição das consultas de seleção e atribuição de pesos e preferências para critérios de *QoC*.

Este trabalho está dividido em 7 capítulos além desta introdução. O capítulo 2 apresenta o embasamento teórico com definições sobre *Contexto*, *Computação Sensível ao Contexto*, *Qualidade de Contexto*, *Internet das Coisas*, *Cloud*, *Edge* e *Fog Computing*.

O capítulo 3 apresenta diferentes métodos de cálculo e de agregação de *QoC* encontrados na literatura, bem como o método *AHP*, largamente utilizado em processos de decisão, e que serve como base de comparação para outros métodos de agregação.

O capítulo 4 apresenta a já citada arquitetura RD, desenvolvida para juntar os conceitos de *QoC* a *IoT* e *Edge Computing* e inspirada na literatura encontrada, com a

descrição de seus módulos e submódulos.

O capítulo 5 apresenta detalhes da implementação de um protótipo da arquitetura RD, indicando algumas das escolhas em termos de ferramentas utilizadas e configurações de distribuição de módulos que foram definidas.

O capítulo 6 apresenta os experimentos e os resultados obtidos, detalhando seus objetivos e buscando justificar por que e de que forma cada um deles foi realizado.

O capítulo 7 apresenta trabalhos relacionados considerados especialmente importantes para comparação à arquitetura RD, identificando seus objetivos e aspectos definidores, buscando justificar a inclusão destes no texto e demonstrar a influência que tiveram em algumas das decisões tomadas durante o desenvolvimento do trabalho.

Por fim, o capítulo 8 apresenta a conclusão e os trabalhos futuros, destacando como o trabalho resultou em respostas satisfatórias para os problemas de pesquisa principal e secundários, ainda que outras perguntas permaneçam e tenham surgido no decorrer da pesquisa, e não seja possível dar respostas definitivas devido à subjetividade inerente à própria natureza dos problemas sendo abordados, das soluções e dos respectivos métodos de avaliação possíveis.

Além destes capítulos, ao final do texto é possível encontrar apêndices que incluem gráficos, questionários etc. que visam auxiliar na compreensão do texto e também explicitar um pouco do esforço feito, de forma a não tirar o foco dos aspectos principais abordados no texto desta dissertação.

2 BASE CONCEITUAL

Neste capítulo são apresentados os principais conceitos que fundamentam o trabalho a ser apresentado: *Contexto*, *Computação Sensível ao Contexto*, *Qualidade de Contexto*, *Internet das Coisas*, *Cloud*, *Fog* e *Edge Computing*, com parte da história da evolução destes conceitos e algumas das escolhas feitas para o desenvolvimento do projeto com base nos conceitos estudados.

2.1 Contexto

Contexto é um conceito rico e difícil de definir, e tem sido estudado por filósofos, linguistas, psicólogos e, mais recentemente, por cientistas da computação (WAN, 2009). O *Oxford English Dictionary* indica que contexto denota "*as circunstâncias que formam o cenário para um evento*" (OXFORD ENGLISH DICTIONARY, 2009). O termo "contexto" vem das palavras em latim *con* ("com" ou "junto") e *texere* ("tecer" ou "tramar") (WAN, 2009).

Contexto, na visão de Schilit, Adams and Want (1994), se refere a localização, identidades de pessoas próximas, objetos e mudanças nestes objetos. No entanto, este tipo de definição por exemplos é difícil de aplicar. Quando queremos determinar se um tipo de informação não listada na definição é ou não contexto, não fica claro como podemos usar a definição para resolver o dilema (DEY, 2001).

Retomando a ideia de Wan, podemos pensar em um contexto como uma circustância em que são "tramados juntos" diversos tipos de entidades (WAN, 2009). Um exemplo de cenário, "seminário", é a combinação de diversas entidades, tais como *apresentador*, *tópico*, *audiência*, *tempo* e *localização*, entre outros (WAN, 2009).

Tomando a definição dada em *Understanding and Using Context* (DEY, 2001), que diz que contexto "*é qualquer informação que possa ser usada para caracterizar a situação de uma entidade, sendo que uma entidade é uma pessoa, local ou objeto considerados relevantes à interação entre um usuário e uma aplicação, incluindo os próprios usuário e aplicação*", temos um aparente paradoxo, já que a definição dada em (WAN, 2009) define o contexto em razão das entidades, e a apresentada por Dey define a situação das entidades em razão do contexto.

No entanto, ao analisarmos mais cuidadosamente as duas definições, e considerando o afirmado por Wan sobre estarem "tramados juntos" diversos tipos de entidades,

percebemos que a relação entre entidade e contexto pode ser conciliada, já que a presença do conjunto de entidades envolvidas em um cenário também faz parte do contexto de uma outra determinada entidade dentro do mesmo cenário.

Assim, é possível unir estas duas definições e enunciar uma nova, afirmando que "contexto é qualquer informação que possa ser usada para caracterizar a situação de uma entidade, incluindo o contexto das outras entidades envolvidas na formação de um cenário", sendo o conceito de entidade conforme descrito por Dey. Esta definição será adotada e referida ao longo do texto, com destaque após a definição de *dimensões de QoC*, com a apresentação de dimensões dependentes de outras dimensões.

2.2 Computação Sensível ao Contexto

O conceito de "*Sensibilidade ao Contexto*" ou "*Ciência de Contexto*" (em inglês, *Context Awareness*), no âmbito da Computação Ubíqua, conforme descrita por Weiser em "*The Computer for the 21st Century*" (WEISER, 1991), foi apresentado e definido pela primeira vez em "*Disseminating Active Map Information to mobile Hosts*" (SCHILIT; ADAMS; WANT, 1994) como "*a capacidade das aplicações para um usuário [de computação] móvel em descobrir e reagir às mudanças no ambiente onde estão situadas*". O foco do trabalho era em como disseminar informação contextual de um servidor para uma gama de dispositivos móveis (BISGAARD; HEISE; STEFFENSEN, 2004), sendo primariamente técnico.

Já em "*Context-aware computing applications*" (SCHILIT; ADAMS; WANT, 1994), o termo é utilizado para se referir a *Software Sensível ao Contexto*, que é definido como *software* capaz de examinar o ambiente computacional e reagir às mudanças de ambiente, de acordo com a localização onde se dá o uso do sistema, o conjunto de pessoas próximas, servidores e dispositivos que podem ser acessados, bem como as mudanças que podem ocorrer com os mesmos (SCHILIT; ADAMS; WANT, 1994). Isto significa que o software é capaz de perceber o ambiente que o cerca e reagir de acordo com mudanças constantes. Baseados nestas definições, descrevem então uma matriz com 4 diferentes categorias de sistemas sensíveis ao contexto (BISGAARD; HEISE; STEFFENSEN, 2004) (Tabela 2.1).

A matriz é um produto da dimensão ortogonal, seja a tarefa obter informação ou executar um comando, e a dimensão vertical, seja ela executada de forma manual ou automática. *Seleção Próxima* é quando a informação solicitada pelo usuário é apresentada de acordo com o contexto atual. Quando um sistema emprega comandos contextuais,

Tabela 2.1: Matriz de categorias de sistemas Sensíveis ao Contexto

	Manual	**Automática**
Informação	Seleção Próxima e Informação Contextual	Reconfiguração Contextual Automática
Comando	Comandos Contextuais	Ações Disparadas por Contexto

Fonte: (SCHILIT; ADAMS; WANT, 1994)

apresenta apenas elementos de interface que são relevantes no contexto atual (BISGAARD; HEISE; STEFFENSEN, 2004). Sistemas que aplicam automaticamente uma nova configuração ao sistema são classificados como *Reconfiguração Contextual Automática*, e a última célula se refere às ações disparadas automaticamente quando o contexto do software muda (BISGAARD; HEISE; STEFFENSEN, 2004).

Em "*Using while moving: HCI issues in fieldwork environments*" (PASCOE; RYAN; MORSE, 2000), sensibilidade ao contexto é definida como a habilidade dos dispositivos computacionais em detectar, sentir, interpretar e responder a aspectos do ambiente local do usuário e aos próprios dispositivos computacionais (BISGAARD; HEISE; STEFFENSEN, 2004). Esta definição de Pascoe, conforme descrita por Bisgaard, leva o conceito de computação sensível ao contexto a apresentar características compatíveis com a definição de contexto adotada na seção anterior, ao apresentar a preocupação dos dispositivos computacionais com as informações relativas a outros dispositivos computacionais.

De acordo com "*Towards a Better Understanding of Context-Awareness*" (ABOWD et al., 1999), "*Um sistema é sensível ao contexto se usa contexto para prover informação relevante e/ou serviços ao usuário, onde relevância depende da tarefa do usuário*". De acordo com os autores, esta definição se opõe à compreensão comum de contexto como sendo apenas informação explícita, e o fato de a informação ser percebida pelo sistema ou de um usuário ser instado a apresentar a informação é considerado por eles como imaterial (BISGAARD; HEISE; STEFFENSEN, 2004).

Ainda, de acordo com Schmidt em "*There is more to Context than Location*" (SCHMIDT; BEIGL; GELLERSEN, 1999), *fatores humanos* e *ambiente físico* são definidos como dois aspectos importantes relacionados a sensibilidade ao contexto e Ciência da Computação.

A partir das definições apresentadas, neste trabalho será considerado o conceito de computação sensível ao contexto como um amálgama destas, sendo enunciado como "a capacidade de um sistema computacional em detectar, sentir, interpretar e responder a

fatores humanos, do ambiente físico, e possivelmente de outros sitemas computacionais, de forma a prover informação relevante ou serviços a um usuário ou aplicação".

2.3 Qualidade de Contexto (*Quality of Context* ou *QoC*)

Qualidade de Contexto é um conceito complexo, com diferentes definições e interpretações do que envolve e qual o seu propósito. As principais definições encontradas, e aquela adotada para o trabalho, são apresentadas nesta seção.

2.3.1 Origem do termo e diferença entre *QoC*, *QoD* e *QoS*

Quality of Context ou *Qualidade de Contexto* foi introduzido como conceito no artigo "*Quality of Context: What it is and why we need it*" (BUCHHOLZ; SCHIFFERS, 2003), que o define como "*qualquer informação que descreve a qualidade da informação utilizada como informação de contexto*", e afirma que "*se refere à informação, e não ao processo ou componente de hardware que possivelmente provê a informação*" (BUCHHOLZ; SCHIFFERS, 2003). Ainda neste artigo, os autores apontam que, em sua experiência, os *parâmetros de QoC* mais importantes, os quais são usados em literatura posterior, e suas respectivas descrições, são:

1. *Precisão* (*Precision*): Descreve o quanto a informação de contexto provida reflete a realidade. É especificada com limites. Um receptor de GPS, por exemplo, permite precisão de cerca de 4 metros, enquanto que o posicionamento via GSM atinge precisões de até 500 metros em áreas urbanas. A precisão pode ser especificada na mesma escala em que a informação de contexto, ou um percentual pode ser usado;
2. *Probabilidade de Corretude* (*Probability of Correctness*): Denota a probabilidade de uma determinada informação de contexto estar correta. Como exemplo, os autores citam uma rede de sensores de temperatura, onde os sensores podem falhar e começar a prover informação errada, medindo 10 graus Celsius quando a informação correta seria 20 graus Celsius. Com o parâmetro Probabilidade de Corretude, a fonte original da informação de contexto estima a frequência com que a informação de contexto provida estará errada por causa de problemas internos;
3. *Confiabilidade* (*Trust-worthiness*): Também descreve a probabilidade de uma informação provida estar correta. No entanto, confiabilidade é usada pelo provedor

de contexto para avaliar a qualidade do ator do qual o provedor recebeu a informação de contexto originalmente. Assim, por exemplo, propõe-se que um provedor de contexto A envie informações sobre o saldo de uma conta a um provedor de contexto B, e esta tenha probabilidade de corretude de 100% de acordo com A, mas, no passado, B tenha recebido informação incorreta de A. Neste caso, o provedor de contexto repassa a informação sobre o saldo da conta, mas indicando que o provedor de contexto A é não-confiável;

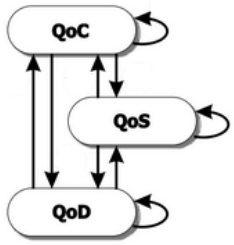
4. *Resolução* (*Resolution*): denota a granularidade da informação. Como exemplo, os autores sugerem que seja considerado um provedor de contexto anunciando que a temperatura em um certo cômodo é de 17 graus Celsius. Embora em média isto seja verdade, há uma torradeira quente no canto de um dos cômodos. Entretanto, o provedor de contexto é incapaz de oferecer uma informação com melhor granularidade, devido a um número restrito de termômetros;
5. *Atualidade* (*Up-to-Dateness*): descreve a idade da informação contextual. Em geral, atualidade é especificada somando um *time-stamp* à informação contextual. Assim, a sincronização de relógios entre a fonte do contexto e o local onde o contexto será consumido é necessária. Frequentemente, será mais interessante saber o quão bem uma informação contextual formalmente provida ainda descreve de forma acurada a situação atual. Isto pode ser determinado requisitando-se a informação contextual atual ou instalando-se um serviço de evento na fonte do contexto, confiavelmente atualizando a informação de contexto se o valor de contexto atual difere de forma significativa da informação contextual provida anteriormente.

Buchholz também descreve a diferença entre *QoC*, *QoD* (*Quality of Device* ou *Qualidade de Dispositivo*) e *QoS* (*Quality of Service* ou *Qualidade de Serviço*). Qualidade de Dispositivo é definida como qualquer informação sobre as propriedades e capacidades técnicas de um dispositivo, enquanto Qualidade de Serviço é qualquer informação que define o quão bem um serviço funciona. Qualidade de Contexto é diferenciada dos outros dois conceitos por descrever a qualidade da informação, inclusive quando não há um dispositivo ou um serviço envolvido (BUCHHOLZ; SCHIFFERS, 2003).

Os autores ainda afirmam que, apesar de serem diferentes, estas noções de qualidade influenciam umas às outras, podendo estas noções afetar as outras de duas maneiras, seguindo o modelo encontrado em "*A Framework for IT Service Management*" (RODOŠEK, 2002): *bottom-up* ou *top-down*. A relação *bottom-up* indica que uma das noções de qualidade afeta tecnicamente a outra, e.g. a distância entre sensores *GSM*

(*QoD*) afeta a precisão da informação de localização (*QoC*). Esta abordagem é representada pelas setas direcionadas para cima na Figura 2.1 (BUCHHOLZ; SCHIFFERS, 2003). Já a relação *top-down* indica que uma noção afeta outra pelos requisitos que impõe e.g. se é necessário que seja provida informação de temperatura de alta precisão (*QoC*), dispositivos com grande exatidão precisam ser utilizados (*QoD*). Esta relação é indicada pelas setas apontando para baixo na Figura 2.1 (BUCHHOLZ; SCHIFFERS, 2003).

Figura 2.1: Relação de influência entre noções de contexto.



Fonte: (BUCHHOLZ; SCHIFFERS, 2003), adaptada

Em "*Context-aware Web Engineering: Modeling and Applications*" (KALTZ; ZIEGLER; LOHMANN, 2005), "*Method and apparatus for establishing context among events and optimizing implanted medical device performance*" (KENKNIGHT et al., 2006) e "*Context-aware Sensor Search, Selection and Ranking Model for Internet of Things Middleware*" (PERERA et al., 2013), para citar alguns, os autores utilizam aspectos de *hardware*/dispositivo (e.g. *integridade física*, *qualidade da bateria* etc.) além de outros aspectos considerados pela literatura como sendo *QoC*, para definir um determinado contexto. Desta forma, para os propósitos deste trabalho, aspectos de *QoD*, como saúde e carga da bateria, serão incorporados e tratados como sendo *QoC*.

2.3.2 Evolução do conceito de *QoC* e *Parâmetros QoC*

Um dos primeiros artigos a considerar a necessidade de mensurar *QoC* foi "*A Quality Measurement Method of Context Information in Ubiquitous Environments*" (KIM; LEE, 2006). Nele, os autores afirmam que "*A Informação Contextual muda constantemente, e se origina de fontes distribuídas, incompletas e incertas. Assim, a informação contextual pode não ser confiável, nem útil, o que transparece como um problema de qualidade de informação contextual*".

Em outro trecho, é feita a associação de *qualidade da informação contextual* ao

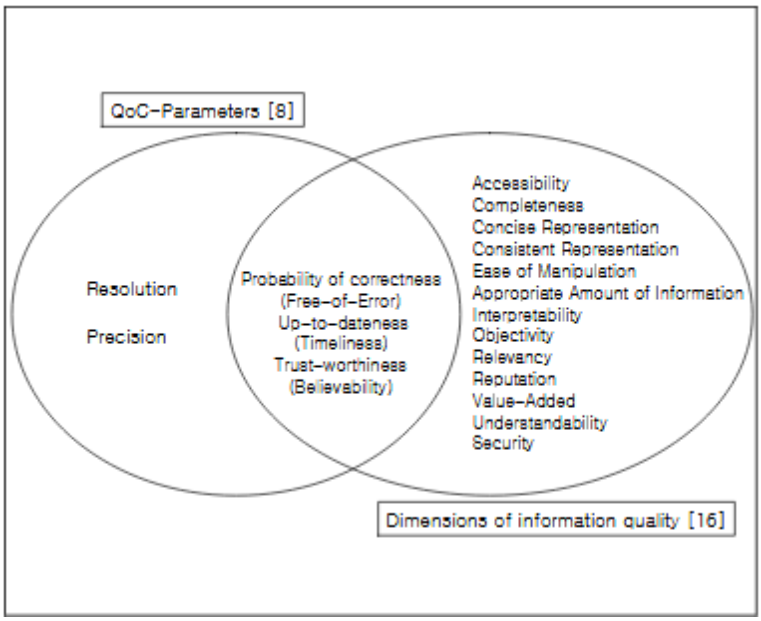
conceito de *QoC* apresentado por Buchholz, bem como são citados os parâmetros *QoC* tidos por Buchholz como principais. Os autores então afirmam que, apesar de muitos estudos sobre estes e outros parâmetros realizados por diversos pesquisadores, pouco havia sido estudado sobre como mensurar estes parâmetros (ou *dimensões*, como também são chamados no artigo).

Kim e Lee então apresentam quatro aspectos que consideram características fundamentais para a informação contextual, e descrevem algumas de suas implicações da seguinte maneira:

1. *Dinamismo* (*Dynamic*): Informação contextual em ambientes ubíquos muda de maneira constante, com base nas proximidades do usuário. Implica então em situação incerta, incompleta ou incorreta. Além disso, como a própria mudança implica em temporalidade da informação, ter a informação mais recente é um requisito;
2. *Distribuição* (*Distribution*): Informação contextual se origina nas mais diversas fontes. Assim, a representação desta informação pode ser diferente, e ocasiona uma contradição lógica quando representações mutuamente diferentes são usadas em aplicações;
3. *Transformação* (*Transformation*): Aplicações sensíveis ao contexto transformam informação de baixo nível gerada por sensores em informação de alto nível, ou geram informação contextual que realiza uma nova ação agregando informação. Assim, se a informação de baixo nível apresentar erros, a informação de alto nível terá também, naturalmente, um erro, e mecanismos extremamente simplificados de raciocínio podem a partir daí também originar um erro;
4. *Confidencialidade* (*Confidentiality*): Informação contextual contém muitos dados sobre os indivíduos, de forma que esta informação deve ser mantida de maneira estritamente confidencial quando coletada ou utilizada.

Os autores seguem descrevendo a relação entre dimensões de qualidade da informação, conforme "*Information quality benchmarks: product and service performance*" (KAHN; STRONG; WANG, 2002), e parâmetros *QoC*, conforme (BUCHHOLZ; SCHIFFERS, 2003), selecionando a partir daí as dimensões de qualidade de informação contextual a serem abordadas, justificando a escolha dos dois artigos utilizados por considerarem os dois estudos de alta qualidade em termos de condução da pesquisa. Esta relação está representada pela Figura 2.2. As dimensões escolhidas e o que chamam de *preocupação do usuário* (*user's concern*) em relação a estas dimensões são apresentados na Tabela 2.2.

Figura 2.2: Relação entre Dimensões de Qualidade de Informação e Parâmetros *QoC*.



Fonte: (KIM; LEE, 2006)

Tendo escolhido as dimensões consideradas relevantes, os autores então implementaram um cenário de teste com uma aplicação destinada a auxiliar no tratamento de hipertensos, e introduziram fórmulas matemáticas que permitem mensurar duas dimensões: *acurácia* e *completude*. Acurácia é definida como "a medida em que a informação de contexto é correta e confiável", e mensurada usando um modelo probabilístico em que os valores devem estar dentro de um determinado intervalo de confiança. Já a completude é definida como sendo o quanto da informação de contexto disponível está presente nos dados utilizados, tendo seu valor entre 0 e 1, e é calculada sob duas perspectivas: *individual*, em que o valor de completude depende da razão entre o número de leituras presente nos dados e o número de leituras efetivamente realizadas por um sensor, e *agregado*, em que o valor é dado pela razão entre a quantidade de dimensões de qualidade encontradas nos dados e quantidade total de dimensões que se deseja considerar para criar uma nova informação contextual.

Em "*Middleware support for quality of context in pervasive context-aware systems*" (SHEIKH; WEGDAM; SINDEREN, 2007), os autores buscam criar um sistema de tele-monitoramento de pacientes epiléticos, e identificam algumas responsabilidades que o *middleware* deve ter, como aquisição de contexto, agregação, raciocínio/inferência e distribuição, e, em particular, a necessidade de considerar explicitamente *QoC*.

Os autores então explicam o processo realizado pela aplicação de monitoramento, em que os sinais vitais do paciente são monitorados constantemente utilizando sensores

Tabela 2.2: Relação entre Preocupações do Usuário e Dimensões de Qualidade

Preocupação do Usuário	Dimensão de Qualidade
Até onde os dados são corretos e confiáveis? Qual a probabilidade de uma determinada informação contextual estar correta?	Acurácia (*Accuracy*)
Até onde os dados são completos e suficientes para a tarefa? Quanta informação está disponível?	Completude (*Completeness*)
Até que ponto os dados são representados no mesmo formato? As representações de informação contextual são consistentes?	Consistência de Representação (*Representation Consistency*)
O quão apropriadamente restrito é o acesso aos dados, de forma a manter sua segurança?	Segurança de Acesso (Access *Security*)
Até que ponto os dados são suficientemente atuais para a tarefa? Qual a idade do contexto?	Atualidade (*Up-to-Dateness*)

Fonte: (KIM; LEE, 2006), adaptada

e uma *body area network* (*BAN*), e são enviados através de redes móveis ou *wireless* para um *back-end* que analisa estes sinais, e a aplicação notifica um cuidador se houver iminência de um ataque. A escolha do cuidador é feita de acordo com a localização do paciente, bem como a localização e disponibilidade dos cuidadores.

A partir deste exemplo, os autores afirmam ter encontrado três razões pelas quais *QoC* deve ser explicitamente considerada:

1. *Adaptação de aplicação orientada a QoC*, pois a informação contextual é inerentemente imperfeita (CASTRO; MUNZ, 2000), e como a adaptação depende fortemente da aplicação, a informação de *QoC* deve ser passada junto com os dados contextuais específicos para a aplicação. Em relação ao exemplo, se dois cuidadores estivessem a uma mesma distância do paciente, a informação de *QoC* relativa à probabilidade de disponibilidade serviria para selecionar um dos cuidadores antes dos outros;

2. *Eficiência do middleware*, pois existem diversas fontes de contexto disponíveis que fornecem o contexto específico necessário para a aplicação, e parte deste contexto gerado pode ter sido armazenado em cache. Ao comparar este contexto armazenado com o contexto das fontes disponíveis, o *middleware* pode otimizar a seleção das fontes de contexto. No exemplo, a localização de um cuidador não precisa ser tão precisa quanto a do paciente, e usar um contexto armazenado pode ser suficiente;
3. *Garantia de privacidade do usuário*, pois existe uma relação entre a sensibilidade da privacidade da informação contextual e *QoC*. O *middleware* deve ser capaz de limitar a qualidade do contexto de acordo com quem solicita a informação. Um exemplo é a informação de localização, em que informar o endereço completo onde um usuário está presente é mais sensível do que prover apenas o nome da cidade. No exemplo, o nível de restrição de *QoC* permite apenas determinar a localização até o nível "rua".

Com base nestas três razões, os autores elencam a escolha de critérios *QoC* a serem utilizados em sua aplicação, sendo eles *precisão*, *atualidade* (*freshness*)[1], *resolução temporal*, *resolução espacial* e *probabilidade de corretude*, apresentando assim os mesmos critérios presentes nos trabalhos já referenciados.

Em "*On the Evaluation of Quality of Context*" (MANZOOR; TRUONG; DUSTDAR, 2008), motivado pelo objetivo de criar uma aplicação de controle de desastres, em que a qualidade da informação contextual provida por trabalhadores no local de desastre, bem como informações geográficas, devem ser avaliadas com base em diferentes fontes, com riscos muito grandes em caso de uma análise errada, foi feito um estudo sobre parâmetros *QoC* e como avaliá-los objetivamente com fórmulas matemáticas.

Os autores afirmam a intenção de utilizar os critérios de contexto apresentados por (BUCHHOLZ; SCHIFFERS, 2003) e (KIM; LEE, 2006). No entanto, implementam *métodos de cálculo* para apenas três deles: Atualidade, confiabilidade e completude. Estes métodos retornam valores entre 0 e 1, decisão que os autores fundamentam com base no fato de que *QoC* indica o nível de conformidade com os requisitos especificados para uma aplicação, sendo que 1 representa conformidade total com os requisitos, e 0, desconformidade total.

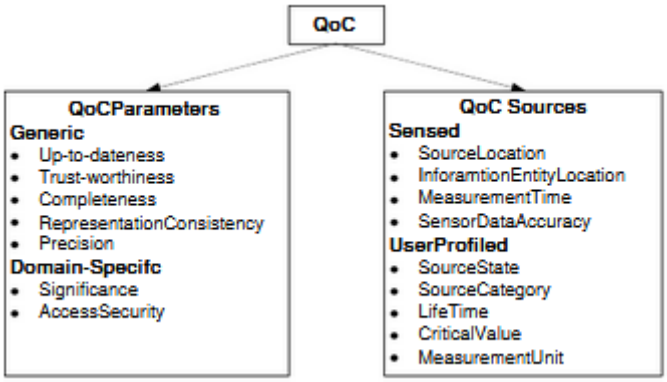
Os autores também criam uma classificação de *QoC* em dois grupos: *fontes* (*QoC Sources*) e *parâmetros* (*QoC Parameters*). Fontes são definidas como as quantidades

[1] o conceito foi traduzido como "atualidade" pois é equivalente ao conceito de "up-to-dateness" apresentado nos trabalhos anteriores

medidas no ambiente ou caracterizadas como configurações do sistema, e não podem ser usadas diretamente por sistemas computacionais para medir *QoC*. Parâmetros são a transformação em informação de alto nível a ser usada em fórmulas para calcular *QoC*.

A divisão entre fontes e parâmetros adotada é apresentada na Figura 2.3, em que alguns exemplos de fontes incluem a *localização* (*sourceLocation*) e o *tempo da medição* (*measurementTime*), apresentados de maneira bruta (e.g. "Rua da Praia, 2923", "23:54, 22/12/2017"), que podem ser relacionados aos parâmetros precisão (*precision*) e atualidade (*up-to-dateness*), entre outros.

Figura 2.3: Classificação de *QoC* feita por Manzoor.



Fonte: (MANZOOR; TRUONG; DUSTDAR, 2008)

O modelo apresentado é implementado utilizando alguns componentes, tais como o *avaliador de QoC* (*QoC Evaluator*), que aplica as fórmulas sobre os valores de parâmetros *QoC*, o *anotador de QoC* (*QoC Annotator*), que anota os valores obtidos junto a outras informações de contexto relevantes utilizando o formato *XML*; e o *provedor de contexto* (*QoC Provider*), que disponibiliza este contexto para as aplicações. Além disso, também implementam um sistema de gerenciamento de contexto que representa os sensores, publicando suas informações numa rede, para que sejam obtidas, utilizando o modelo *publish/subscribe* (BIRMAN; JOSEPH, 1987).

Em "*A Quality-Aware Approach for Resolving Context Conflicts in Context-Aware Systems*" (FILHO; AGOULMINE, 2011), uma divisão entre *parâmetros* e *indicadores* de *QoC* é apresentada, em que parâmetros são os valores obtidos diretamente a partir dos sensores, e os indicadores são valores obtidos utilizando parâmetros aplicados em fórmulas matemáticas, seguindo a mesma lógica encontrada em (MANZOOR; TRUONG; DUSTDAR, 2008), mas com nomes diferentes, já que o conceito de parâmetros de Bringel

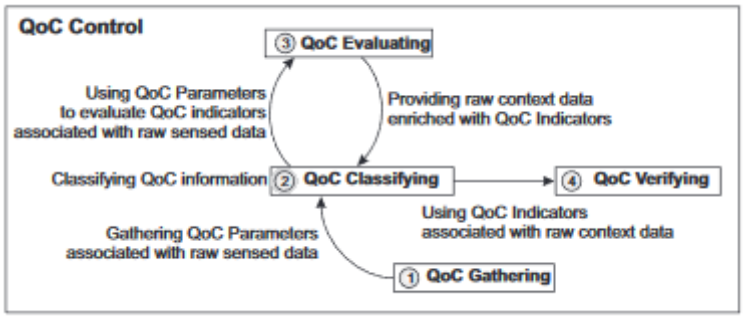
Filho é equivalente ao conceito de fontes de Manzoor, e o conceito de indicadores do primeiro é equivalente ao conceito de parâmetros do segundo.

Ainda no mesmo artigo, os autores definem *QoC* como sendo "*um conjunto de operações de gerenciamento de qualidade realizado por sistemas sensíveis ao contexto de maneira a prover meios de avaliar e verificar a qualidade de informação contextual utilizada no apoio à tomada de decisão sensível ao contexto*" (FILHO; AGOULMINE, 2011), evidenciando a mudança no foco de *QoC* para um aspecto prático e objetivo, com finalidade determinada.

São também apresentadas quatro operações consideradas pelos autores como fundamentais para o controle de contexto:

1. *Aquisição de QoC* (*QoC Gathering*): sensoriamento e caracterização (*profiling*) de parâmetros de *QoC*;
2. *Classificação de QoC* (*QoC Classifying*): representando parâmetros e indicadores *QoC* em formato intelígivel por arcabouços de gerenciamento de contexto e sistemas sensíveis ao contexto;
3. *Avaliação de QoC* (*QoC Evaluating*): avaliação de indicadores *QoC* utilizando métodos de medida aplicados a parâmetros *QoC*;
4. *Verificação de QoC* (*QoC Verfying*): verificação de valores de indicadores *QoC* associados à informação contextual para um propósito bem definido, tal como resolução de conflitos de contexto.

Figura 2.4: Operações principais de controle de *QoC* segundo Bringel Filho.



Fonte: (FILHO; AGOULMINE, 2011)

Além de usar alguns indicadores de contexto encontrados em artigos já referenciados nesta seção, o artigo também apresenta novos métodos para cálculo de *confiabilidade* (*trustworthiness*) e *probabilidade de corretude* (*probability of correctness*), bem como

outros critérios, chamados de *conflito espacial* (*spatial conflict*), *conflito temporal* (*temporal conflict*) e *probabilidade de ocorrência* (*probability of occurence*).

Os autores também apontam uma contradição na definição dada em (BUCHHOLZ; SCHIFFERS, 2003), em relação ao critério confiabiliade (*trustworthiness*), já que esta enunciava que "*confiabilidade é usada pelo provedor de contexto para avaliar a qualidade do ator do qual o provedor recebeu a informação de contexto originalmente*" (BUCHHOLZ; SCHIFFERS, 2003), indo contra a própria definição de *QoC*, que afirma que *QoC* diz respeito diretamente à informação contextual, e não ao processo ou componente de hardware que provê dados contextuais.

Desta maneira, confiabilidade é definida por (FILHO; AGOULMINE, 2011) como sendo a crença que um arcabouço de gerenciamento de contexto tem na corretude da informação de contexto, podendo ser abordada de duas formas:

1. A avaliação do grau de crença que o arcabouço deposita quanto à informação contextual de forma direta;
2. A avaliação de quanto o arcabouço acredita ser verdade o que foi informado pela entidade que providenciou o contexto.

Estas duas formas, de acordo com autores, são utilizadas de maneira mesclada em sua solução, "*verificando a corretude associada a um elemento de contexto previamente provido ao arcabouço*" (FILHO; AGOULMINE, 2011). No entanto, a afirmação não deixa transparecer esta mescla. Além disso, embora a contradição em (BUCHHOLZ; SCHIFFERS, 2003) realmente exista, para os propósitos desta dissertação, conforme o exposto na seção 2.3.1, não ocasiona problema algum, já que foi utilizada outra definição de *QoC*, que considera aspectos de hardware como parte de *QoC*, podendo a definição de confiabiliade dada por (BUCHHOLZ; SCHIFFERS, 2003) ser utilizada.

Em "*Trust and Privacy Management Support for Context-Aware Service Platforms*" (NEISSE, 2012), o autor define atributos *QoC* e relaciona a terminologia de acordo com o padrão de metrologia encontrado em "*International vocabulary of basic and general terms in metrology*" (INTERNATIONAL BUREAU OF WEIGHTS AND MEASURES, 1984).

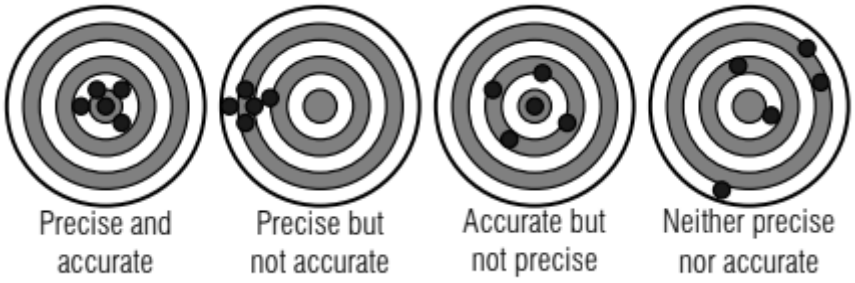
Os atributos considerados incluem acurácia, precisão e resolução, de acordo com o padrão de metrologia utilizado, que são então comparados com atributos *QoC* encontrados em (BUCHHOLZ; SCHIFFERS, 2003) e outros, a saber: acurácia, precisão, probabilidade, resolução espacial, resolução temporal e atualidade (*freshness*). Nota-se que a

definição de *atributos* corresponde às *dimensões* e *critérios* encontrados em outros textos.

O autor decide ainda não incluir confiabilidade (*trustworthiness*) neste conjunto de atributos, argumentando que os modelos existentes à época não apresentavam um consenso que respaldasse a inclusão de confiabilidade como um aspecto de *QoC*. Argumenta também que confiabilidade não seria um critério para a qualidade da informação contextual por si só, mas que depende da visão do consumidor de contexto.

Neisse ainda apresenta uma diferenciação entre acurácia e precisão, com exemplos. Pelo padrão *ISO* em (INTERNATIONAL BUREAU OF WEIGHTS AND MEASURES, 1984), acurácia mede o quanto uma determinada medição está próxima do valor real, enquanto que precisão mede o quão próximos ficam ou quanto é possível repetir os resultados das medições. A partir daí, apresenta um exemplo gráfico (Figura 2.5) fazendo analogia com uma prova de tiro. Estas definições para acurácia e precisão são adotadas para os cálculos de *QoC* descritos no capítulo 6.

Figura 2.5: Representação gráfica de analogia entre prova de tiro e as definições ISO para acurácia e precisão.



Fonte: (NEISSE, 2012)

A questão de nomenclaturas diferentes, conforme pode ser notado nesta seção, é recorrente na literatura. Para os propósitos do trabalho, convencionou-se utilizar os termos “indicadores”, “dimensões” e “parâmetros” para se referir ao que Manzoor define como parâmetros *QoC*, e os dados obtidos diretamente pelos sensores, sem a aplicação de fórmulas, como “dados brutos”.

2.3.3 Agregação *QoC*

Em “*When Efficiency Matters: Towards Quality of Context-Aware Peers for Adaptive Communication in VANETs*” (YASAR et al., 2011), o autor busca realizar a seleção

de possíveis rotas para motoristas e coordenação no trânsito utilizando redes veiculares ad-hoc (*Vehicular Ad-Hoc Networks* ou *VANETs*), em que as recomendações são feitas consultando os hábitos e informações de outros motoristas ligados a este tipo de rede.

Três perguntas são então apresentadas por Yasar:

1. Qual é a qualidade da informação recebida?
2. Qual é a reputação do provedor da informação?
3. No caso de múltiplos provedores, qual deve ser selecionado?

Para respondê-las, define então que é necessário avaliar a qualidade do contexto e a reputação dos provedores. A avaliação de reputação é dependente da qualidade do contexto que é compartilhado pelos diferentes provedores de contexto. Dados com alta qualidade de contexto aumentam a reputação do provedor, enquanto dados com baixa qualidade diminuem.

A qualidade de contexto, segundo o autor, é afetada por diversos fatores, tais como pontualidade e completude das informações, os quais podem ser medidos através de alguns critérios já apresentados na subseção anterior, como atualidade (*freshness*) e completude. No entanto, Yasar vai além, e define que Qualidade de Contexto (*QoC*) é um valor agregado, dado por média ponderada, com os pesos definindo a importância de cada critério, e os valores para cada critério obtidos utilizando fórmulas individuais, de forma que a média resulte em um valor entre 0 e 1.

Após calcular este valor de *QoC*, a reputação de um determinado par entre os provedores de informação contextual, que também varia entre 0 e 1, é utilizada em uma segunda etapa de agregação, juntamente com o valor de *QoC*, para informar a qualidade agregada da informação provida, a qual é utilizada para determinar um novo valor de reputação para o provedor (Figura 2.6).

Se um motorista receber informação contextual sobre um determinado tópico de uma determinada fonte, e logo após receber informação contextual relevante sobre o mesmo tópico de uma outra fonte, com uma reputação melhor, a informação recebida anteriormente é descartada, sendo substituída pela informação desta segunda fonte.

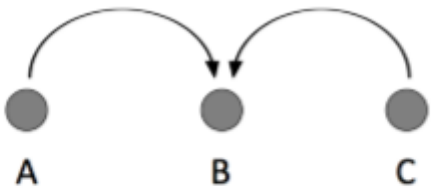
O conceito de agregação utilizando média ponderada apresentado em (YASAR et al., 2011) é utilizado posteriormente em "*Cuida: um Modelo de Conhecimento de Qualidade de Contexto Aplicado aos Ambientes Ubíquos Internos em Domicílios Assistidos*" (NAZÁRIO et al., 2015), porém utilizando critérios *QoC* diferentes, e sem considerar reputação de maneira separada, nem utilizando uma segunda etapa de agregação.

Em (PERERA et al., 2013), os autores apresentam um modelo de busca, seleção e ranqueamento de sensores com base em Sensibilidade ao Contexto, chamado *CASSARAM*, o qual foi feito para ser integrado a sua arquitetura de *IoT* chamada *CA4IoT* (PERERA et al., 2012). O propósito do modelo é permitir aos usuários expressar os requerimentos necessários para uma determinada aplicação que faça uso dos dados obtidos por sensores *IoT*.

Este modelo prevê o uso de diversas *propriedades* ou *dimensões* de contexto, tais como bateria, acurácia, precisão etc. que são calculadas através de fórmulas. Estes valores são agregados através de um método matemático para gerar um valor entre 0 e 1, que é a nota final dada a um determinado sensor. Os valores para cada sensor são utilizados para gerar um *ranking*, permitindo realizar a seleção do número de sensores desejado pelo usuário. Mais detalhes sobre este modelo são apresentados no capítulo 7 de trabalhos relacionados.

É possível notar que, apesar de em nenhum momento os autores se referirem a *QoC* no modelo, todas as características que definem *QoC*, conforme os outros trabalhos apresentados, estão presentes, e por isso é aqui tratado como sendo uma solução utilizando *QoC*.

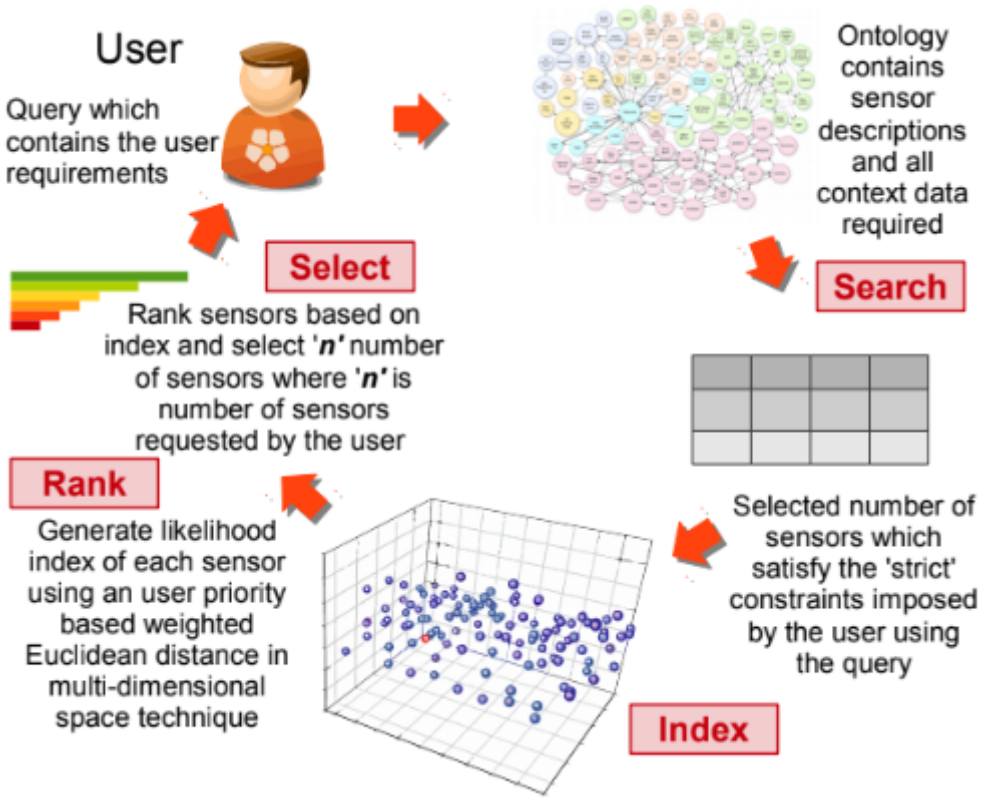
Figura 2.6: Gráfico demonstrativo do sistema de reputação idealizado por Yasar, em que nodo B recebe informação de dois outros nodos e opta por utilizar a informação do nodo C, de maior reputação, após agregação



Message	PR	QoC	Aggregated Quality	New PR	Accept
A -> B	0.40	0.80	0.60	0.56	Yes
C -> B	0.60	0.90	0.75	0.90	Yes
A -> B	0.56	0.90	0.73	0.84	No

Fonte: (YASAR et al., 2011)

Figura 2.7: Visão geral de alto nível do modelo CASSARAM.



Fonte: (PERERA et al., 2013), adaptada

2.4 Internet das Coisas

Atribui-se a criação do conceito "*Internet das Coisas*" (*IoT*, ou *Internet of Things*) a Kevin Ashton, que teria dado a uma apresentação feita para a P&G em 1999 o título "*Internet of Things*" (ASHTON et al., 2009). Em um artigo apresentado no mesmo ano, escreveu que "*Se tivéssemos computadores que soubessem tudo sobre as coisas em geral – usando dados que coletassem sem a nossa ajuda – seríamos capazes de rastrear e contar tudo, e reduzir bastante o desperdício, a perda e os custos. Nós saberíamos quando é necessário substituir, reparar ou fazer um recall de um produto, e se estão novos ou ultrapassados. Precisamos capacitar os computadores com seus próprios meios de coletar informações, para que possam ver, ouvir e cheirar o mundo sozinhos, com toda a sua glória aleatória. O RFID e a tecnologia de sensores capacitam os computadores a observar, identificar e entender o mundo sem as limitações dos dados inseridos pelos humanos*" (LOPEZ RESEARCH LLC, 2018) (ASHTON et al., 2009).

Para que esta visão pudesse se concretizar, campos tradicionais de sistemas embarcados, redes de sensores sem fio (*wireless*), sistemas de controle, automação e outros contribuem todos para permitir a Internet das Coisas (GATEWAY TECH, 2018).

Uma definição para o que é *IoT* é apresentada por Jacob Morgan em "*A simple explanation of 'The Internet of Things' "* (MORGAN, 2014) como sendo "*basicamente conectar cada dispositivo[...] com a Internet (e/ou um ao outro). Isto inclui tudo, desde telefones celulares, cafeteiras, máquinas de lavar, fones de ouvido, lâmpadas, dispositivos vestíveis [. . .] e também se aplica a componentes de máquinas, como, por exemplo, um motor a jato de um avião ou uma perfuradora de petróleo*" (MORGAN, 2014).

A *ITU* (*International Telecommunication Union*) (ITU - INTERNATIONAL TELECOMMUNICATION UNION, 2018) define *IoT* como "*uma infraestrutura global para a sociedade da informação, permitindo a existência de serviços avançados interconectando coisas físicas e virtuais, com base em tecnologias de informação e comunicação existentes e em evolução*" e afirma que "*através da exploração da identificação, captura de dados, capacidades de processamento e comunicação, IoT faz um uso completo das coisas para oferecer serviços a todos tipos de aplicações, enquanto garante que os requisitos de segurança (security) e privacidade sejam cumpridos*" (ITU - INTERNATIONAL TELECOMMUNICATION UNION, 2018).

Desta maneira, o número de dispositivos *IoT* pode escalar facilmente, com diversas previsões indicando crescimento exponencial, com bilhões de dispositivos conectados, com empresas como a Gartner afirmando que em 2020 haverá mais de 26 bilhões de dispositivos *IoT* (MORGAN, 2014), ou ainda 500 bilhões em 2030 (CISCO, 2018), sendo o número em 2018 já estimado em 7 bilhões (IOT ANALYTICS, 2018).

Imaginando possíveis aplicações para os dados obtidos de dispositivos *IoT*, pode-se citar como exemplo o controle de poluição, que pode utilizar diferentes sensores para avaliar os níveis de poluição ambiental (PERERA et al., 2013). Esta aplicação exigiria selecionar dentre possivelmente milhões de dispositivos aqueles que fossem capazes de medir os índices de poluição de acordo com a realidade (PERERA et al., 2013).

Realizar seleções, além de apresentar dificuldades para programadores de aplicação em definir quais sensores devem ser utilizados, também representa alto custo computacional (PERERA et al., 2013), bem como oneração das redes com o aumento do tráfego de dados (SATYANARAYANAN, 2017).

Portanto, selecionar os melhores sensores entre tantas possibilidades exige algum método que permita aos programadores de aplicação e às próprias aplicações fazê-lo facil-

mente, sem onerar excessivamente a infraestrutura de rede. *QoC* se apresenta como uma possibilidade para a solução do primeiro problema, enquanto *Edge* ou *Fog computing* são soluções possíveis para o segundo problema.

2.5 *Cloud, Edge* e *Fog Computing*

Os conceitos de *Cloud*, *Edge* e *Fog Computing* são frequentemente relacionados, e até mesmo confundidos, em diversos dos textos que afirmam utilizá-los. É necessário, portanto, que seja feita distinção clara entre estes três conceitos, a fim de definir a visão de *Edge Computing* adotada na elaboração da arquitetura RD. Esta distinção é obtida a partir do histórico e da argumentação apresentados a seguir.

2.5.1 *Cloud Computing*

Para entender os conceitos de *Edge/Fog computing*, é necessário primeiro conhecer o conceito básico de *cloud computing*. A expressão *cloud computing*, ou *computação em nuvem*, é geralmente atribuída a Eric Schmidt, então *CEO* (*Chief Executive Officer*, ou diretor executivo, em português) da Google, após ter introduzido o termo em uma conferência em 2006 (REGALADO, 2018). No entanto, o termo existe pelo menos desde 1996, sendo utilizado nos escritórios da Compaq para se referir a aplicações como, por exemplo, armazenamento de dados pela Internet (REGALADO, 2018).

Para os propósitos deste trabalho, será utilizada a definição de *cloud computing* dada por Peter Mell do *NIST* (*National Institute of Standards and Technology*) dos Estados Unidos, que diz que *Cloud computing* "*é um modelo para permitir acesso conveniente, ubíquo e sob demanda através da rede a um conjunto compartilhado de recursos computacionais configuráveis (tais como redes, servidores, armazenamento, aplicações e serviços) que possam ser provisionados rapidamente e liberados com o mínimo de esforço para gerenciar ou com o mínimo de interação por parte do provedor de serviço*" (MELL; GRANCE et al., 2011).

Além disso, ainda segundo Mell, é preciso considerar cinco características essenciais de *cloud computing*:

1. *Auto-serviço sob demanda* (*On-demand Self-Service*): Um consumidor pode provisionar unilateralmente capacidade computacional, tal como tempo de servidor ou

armazenamento de rede, conforme a necessidade e de maneira automática, sem necessitar de interação humana com cada provedor de serviço;

2. *Acesso amplo à rede* (*Broad Network Access*): Os recursos (no original, *capabilites*) estão disponíveis através da rede e são acessados através de mecanismos padrão que promovem o uso heterogêneo por parte de plataformas cliente *thick* (com muito recurso computacional (BEAL, 2018)) ou *thin* (com pouco recurso computacional (BEAL, 2018)) (e.g. Telefones móveis, *tablets*, *laptops* e estações de trabalho);
3. *Agrupamento de recursos* (*Resource pooling*): os recursos computacionais do provedor são agrupados para servir a diversos consumidores utilizando um modelo *multi-inquilino* (*multi-tenant*). Um inquilino é um usuário ou grupo de usuários com determinados privilégios dentro da estrutura que utiliza o modelo encontrado em (KREBS; MOMM; KOUNEV, 2012)), com diferentes recursos físicos e virtuais atribuídos e reatribuídos dinamicamente de acordo com a demanda do consumidor. Há um senso de independência de localidade à medida que o usuário geralmente não tem controle ou conhecimento sobre a localização exata dos recursos providos; no entanto, pode especificar uma localização em um nível mais alto de abstração (e.g. País, estado ou *datacenter*). Exemplos de recursos incluem armazenamento, memória e largura de banda (*bandwidth*) de rede;
4. *Elasticidade rápida* (*Rapid elasticity*): Recursos podem ser provisionados e liberados de maneira elástica, em alguns casos de forma automática, para escalar rapidamente em expansão ou contração de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento frequentemente parecem ilimitados e podem ser apropriados em quantidade a qualquer momento;
5. *Serviço mensurado* (*Measured service*): Sistemas *Cloud* controlam automaticamente e otimizam o uso de recursos aproveitando alguma capacidade de medição (geralmente através de um sistema de pagamento ou cobrança por uso) em algum nível de abstração adequado ao tipo de serviço (e.g. armazenamento, processamento, largura de banda ou contas de usuário ativas). O uso de recursos pode ser monitorado, controlado e relatado, provendo transparência tanto para o usuário quanto para o provedor do serviço utilizado.

Considerando estas características, alguns aspectos importantes em relação ao uso de *cloud computing* para uma possível aplicação de *IoT* utilizando *QoC* são a distância dos recursos em relação aos sensores utilizados, bem como o uso de banda, memória

e processamento para transferir dados, aplicar métodos de cálculo, agregar e retornar resultados a partir dos dispositivos *IoT*.

2.5.2 *Edge/Fog Computing*

Embora o termo "*Edge Computing*" apareça na literatura pelo menos desde 2004 (PANG; TAN, 2004), as raízes de *Edge Computing* se estendem no passado até o final dos anos 90, quando a Akamai introduziu as *redes de entrega de conteúdo* (*Content Delivery Networks*, ou *CDNs*) para acelerar a performance da *web* (SATYANARAYANAN, 2017). Uma *CDN* usa nodos na borda (*edge*) da rede, próximo aos usuários, para pré-obter e fazer *cache* de conteúdo *web*. Estes nodos de borda também podem realizar customizações, como, por exemplo, adicionar propaganda relevante para o usuário (SATYANARAYANAN, 2017). *CDNs* são especialmente valiosas na distribuição de conteúdo em vídeo, pois a economia de largura de banda de rede utilizando *cache* pode ser substancial.

Edge Computing generaliza e estende o conceito de *CDN* fazendo uso de infraestrutura de *cloud*. Assim como as *CDNs*, a proximidade de *cloudlets*/dispositivos *edge* (recursos computacionais e armazenamento fisicamente mais próximos do usuário do que a infraestrutura de *cloud computing*) (SATYANARAYANAN et al., 2009) em relação ao usuário final é crucial. No entanto, diferentemente de uma *CDN*, um *cloudlet* pode executar código arbitrário, como em *cloud computing*. Este código é tipicamente encapsulado em uma *VM* ou em alguma estrutura de *container* mais leve, para fins de isolamento, segurança (*safety*[2]), gerenciamento de recursos e medição (SATYANARAYANAN, 2017).

Satyanarayanan defende o uso de uma arquitetura de *edge computing* de dois níveis, inicialmente, mas que pode ser expandida para ter mais níveis. O primeiro nível é a arquitetura de *cloud* sem modificações, enquanto o segundo nível é composto por elementos dispersos, *cloudlets*, com estados do primeiro nível em *cache* (SATYANARAYANAN, 2017). Esta arquitetura teria como objetivo reduzir a latência e o uso de largura de banda, principalmente em redes *wireless*, utilizando os *cloudlets* próximos à fonte dos dados e aos usuários, tendo como foco principalmente as aplicações *IoT*.

As explicações e objetivos apresentados por Satyanarayanan estão de acordo com a definição apresentada por Shi em "*Edge Computing: Vision and Challenges*", que diz

[2] Aqui, o conceito buscado por Satyanarayanan provavelmente é *security*, que é o termo mais comum na literatura para tratar de segurança de acesso a dados e sistemas de software, como explicado em (LINE et al., 2006)

que "*edge computing se refere às tecnologias facilitadoras que permitem que computação seja realizada na borda da rede, sobre dados enviados a partir dos serviços em cloud e também enviados no sentido inverso pelos serviços IoT*" (SHI et al., 2016).

Ainda, sob o ponto de vista de Shi, *edge computing* é um conceito intercambiável com o conceito de *Fog Computing*, com a diferença entre os dois sendo o foco de *edge computing* em coisas, ou seja, os dispositivos que geram dados de interesse, enquanto *fog computing* foca na infraestrutura necessária para tornar o acesso aos dados gerados para usuários, e aplicações que os utilizem, possível.

De acordo com Satyanarayanan, Bonomi introduziu o conceito de *Fog Computing* em seu artigo "*Fog computing and its role in the internet of things*" (SATYANARAYANAN, 2017) (BONOMI et al., 2012). Nele, o autor afirma que *Fog Computing* "*estende o paradigma de cloud computing para a borda da rede, assim permitindo uma nova gama de aplicações e serviços*" (BONOMI et al., 2012).

Bonomi define *fog computing* como sendo "*uma plataforma altamente virtualizada que provê serviços de computação, armazenamento e rede entre dispositivos na ponta [da rede] e datacenters de cloud computing tradicionais, tipicamente, mas não exclusivamente, localizados na borda (edge) da rede*". Esta definição vai contra a afirmação de Shi sobre *edge* e *fog computing* serem intercambiáveis, por levar em conta dispositivos que não estão na borda da rede como sendo um nó *fog* em potencial.

Bonomi ainda cita sete características para definir o conceito de *fog computing*:

1. Baixa latência e sensibilidade/consciência de localização;
2. Distribuição geográfica amplamente distribuída;
3. Mobilidade;
4. Número muito grande de nodos;
5. Papel predominante do acesso sem fio (*wireless*);
6. Forte presença de *streaming* e aplicações de tempo real;
7. Heterogeneidade.

A definição das características de *fog computing* apresentada por Bonomi vai na direção dos objetivos e características definidos por Satyanarayanan, bem como dos objetivos apresentados em um trabalho da IBM em parceria com a Nokia e a Siemens: o *RACS* (*Radio Applications Cloud Server*) (IBM, 2013), apresentado por estas empresas como sendo uma solução para *edge computing*. Estes objetivos incluem a *redução da latência de rede* e *otimização do tráfego de rede* em redes móveis e sem fio.

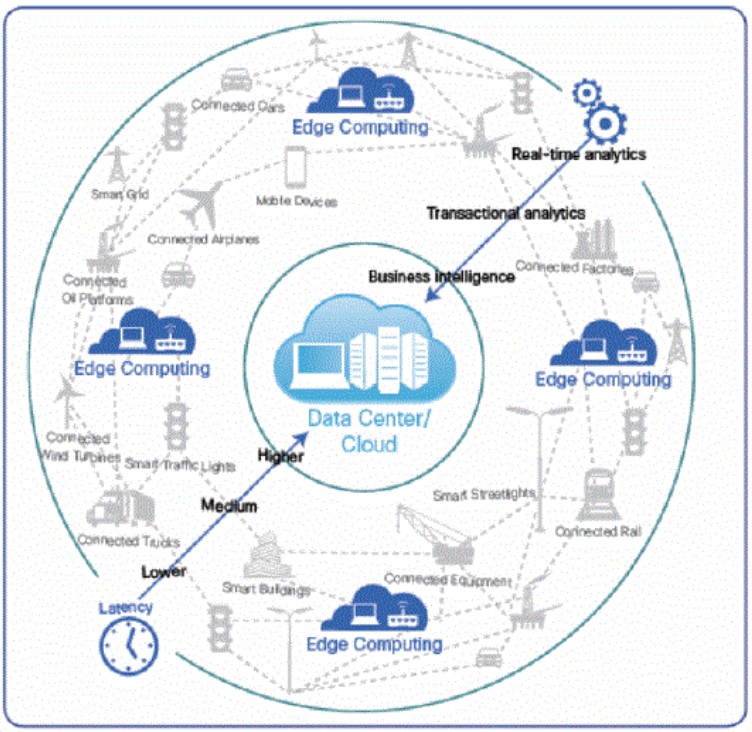
Shi ainda afirma, explicando porque *edge computing* é necessário, que, em *IoT*, os dados são produzidos e consumidos na borda da rede e, portanto, os dados, e também as operações aplicadas aos dados, devem ter tratamento na borda (SHI et al., 2016).

Considerando os conceitos e definições de *edge* e *fog computing* encontrados na literatura, é possível perceber que, para aplicações *IoT*, é preciso considerar o uso de *edge* ou *fog computing*, visando otimizar o uso de recursos computacionais e o tempo de resposta.

A partir das definições e conceitos encontrados na literatura, será considerado que *fog computing* é uma extensão do conceito de *edge computing*, seguindo a definição de Bonomi, em que podem haver níveis intermediários de nós *fog*, e *edge computing* considerando apenas a provisão de serviços para dispositivos de borda efetivamente, sem níveis intermediários.

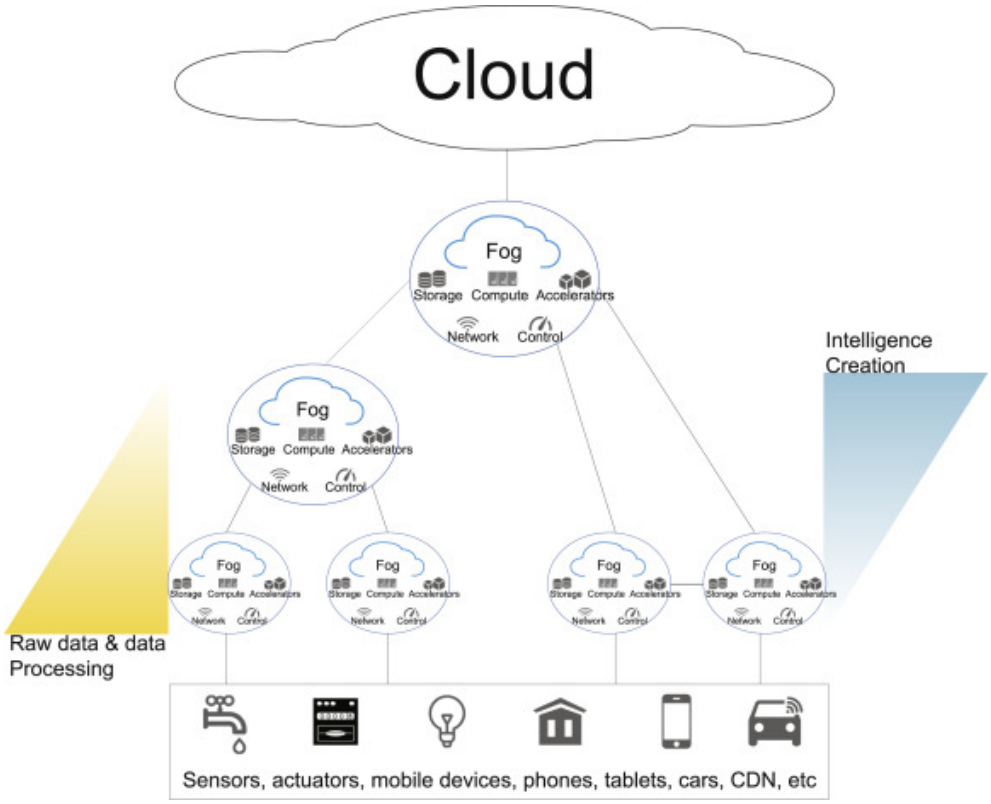
Desta maneira, *fog computing* e *edge computing* podem ser intercambiáveis, como considera Shi, nos casos em que a estrutura *fog* não tem mais níveis intermediários entre a primeira estrutura, na borda, e o *cloud datacenter*.

Figura 2.8: Estrutura típica de sistemas que utilizam *Edge Computing*.



Fonte: (BYERS; WETTERWALD, 2015), adaptada

Figura 2.9: Estrutura típica de sistemas que utilizam *Fog Computing*.



Fonte: (AI; PENG; ZHANG, 2018), adaptada

2.6 Considerações Finais

Neste capítulo, foram apresentados conceitos e definições sobre *contexto*, *QoC*, *IoT*, *cloud*, *edge* e *fog computing*. O estudo permitiu identificar aspectos importantes para uma arquitetura de seleção de sensores *IoT* que proveja dados utilizando critérios de *QoC*. Entre estes aspectos, está o uso de *métodos de cálculo* para avaliar cada um dos critérios, o uso de um *método de agregação* que permita dar uma nota única para cada sensor considerando todos os critérios, e o uso de um *intervalo padronizado* que permita comparar sensores.

O estudo sobre *IoT* permitiu perceber que o número de sensores de *IoT*, e a previsão de crescimento deste número, implica em problemas devidos à escala para a criação de aplicações, à necessidade de selecionar os melhores dispositivos *IoT* para os diferentes requisitos, e ao aumento do tráfego de dados pelas redes, sobrecarregando a infraestrutura subjacente.

Além disso, o estudo de *cloud*, *edge* e *fog computing* permitiu perceber que, para aplicações que utilizem informação de dispositivos *IoT*, é necessário levar em conta a

distância das estruturas de *cloud computing* em relação aos dispositivos e às aplicações, e que os principais objetivos ao fazer uso de *fog* e *edge computing* são incrementar a eficiência das redes de transmissão, reduzindo a latência e otimizando o tráfego, bem como permitir aos sistemas computacionais obter um melhor tempo de resposta.

3 MÉTODOS DE CÁLCULO E AGREGAÇÃO

Durante o estudo da literatura, foram encontrados diversos métodos de cálculo para os critérios de *QoC*, bem como métodos de agregação que permitem gerar os valores utilizados para fazer ranqueamento de dispositivos. Estes métodos foram implementados no protótipo da arquitetura RD, e por isso foram compilados e apresentados neste capítulo.

3.1 Métodos de Cálculo

Os métodos de cálculo servem para atribuir nota aos critérios individuais de QoC, tais como acurácia, distância etc. Os métodos encontrados na literatura são apresentados a seguir.

3.1.1 Métodos de Kim & Lee

Os seguintes métodos, para acurácia e completude, respectivamente, são encontrados em (KIM; LEE, 2006):

$$RMSE(S_i) = \sqrt{\frac{1}{N} * [\sum_{i=1}^{N}()x_i - \bar{x})^2]} \tag{3.1}$$

Onde N é o número de entradas observadas, x_i é o valor observado para o dado, $\bar{x}$ é a média dos valores observados e S_i é o sensor i. $RMSE$ significa "*Root Mean Squared Error*" ou "*Raíz do erro quadrático médio*".

$$CS_i = \frac{AD}{TD} \tag{3.2}$$

Onde CS_i é a completude para o Sensor i, AD é o número de critérios *QoC* disponíveis e TD é o total de critérios *QoC* que se deseja considerar.

3.1.2 Métodos usados no *Cuida*

Os seguintes métodos de cálculo para critérios *QoC* são apresentados e utilizados por (NAZÁRIO et al., 2015) para encontrar problemas em sensores:

$$\begin{aligned} idade &= tempo\ atual - tempo\ da\ informacao \\ Atualidade &= 1 - \frac{idade}{tempo\_de\_vida}, se\ idade < tempo\_de\_vida \\ Atualidade &= 0,\ nos\ demais\ casos \end{aligned} \tag{3.3}$$

$$\begin{aligned} Cobertura &= \forall valor(sensor) \in [min, max], 1 \\ Cobertura &= 0,\ nos\ demais\ casos \end{aligned} \tag{3.4}$$

$$Significância = \frac{VC}{VC_{Max}} \tag{3.5}$$

Onde VC é o valor crítico, e VC_{Max} é o máximo valor crítico. Cabe notar que, embora Nazário descreva este último método, opta por não utilizar efetivamente no Cuida.

3.1.3 Métodos de Yasar

Os seguintes métodos são encontrados em (YASAR et al., 2011), e apresentados em sua forma adaptada aqui:

$$P(m)_{Relative} = (P(m)_{ABS}/P_{MAX}) \tag{3.6}$$

Onde m é uma informação contextual, $P(m)_{Relative}$ é a prioridade relativa de uma informação contextual, $P(m)_{ABS}$ é a prioridade absoluta da informação contextual e P_{MAX} é a maior prioridade possível. As prioridades entre informações podem ser definidas arbitrariamente, ou através de outros métodos não apresentados por Yasar.

$$\begin{aligned} SR(m) = 1 - (\frac{dist(Src, Dst)}{dist_{max}})\ &se\ (OccurTime < ObsTime) \\ 0, &se\ (OccurTime > ObsTime) \end{aligned} \tag{3.7}$$

Onde $SR(m)$ é a resolução espacial para uma informação contextual m, $dist(Src, Dst)$ é a distância da origem da informação até o destino, $dist_{max}$ é a maior distância a ser considerada válida para receber a informação, $OccurTime$ é o tempo da ocorrência de um fato e $ObsTime$ é o tempo da observação do fato. Distância é dada pela equação:

$$dist(Src, Dst) = \sqrt{(x(Src) - x(Dst))^2 + (y(Src) - y(Dst))^2} \tag{3.8}$$

$$pf = 1 - \frac{\log_2 i + 1}{i} \tag{3.9}$$

Onde pf é o fator de participação de um provedor de contexto e i é o número de mensagens enviadas pelo provedor.

$$PR(n)_i = PR(n)_{i-1} + (QoC(m) - T) \times \frac{PR(n)_{i-1} \times pf}{HopCount} \tag{3.10}$$

Onde $PR(n)$ é a reputação entre os pares (***Peer Reputation***) de um determinado nodo n, provedor de contexto; i é a quantidade de mensagens recebidas pelo provedor; T é o limiar de reputação e $QoC(m)$ é o valor agregado de ***QoC***, dado por uma média ponderada dos critérios de ***QoC*** considerados, e $HopCount$ é a quantidade de vezes em que as mensagens passaram por nodos intermediários.

3.1.4 Métodos Próprios

Os métodos a seguir foram utilizados para a implementação apresentada no capítulo 5. São métodos próprios, necessários para os testes apresentados no capítulo 6, sem equivalentes expressos por fórmulas na literatura. Não foi feita verificação formal destes métodos, sendo considerados suficientes os testes de mesa realizados. São eles:

$$\varsigma = \frac{t}{\hat{t}} \tag{3.11}$$

Onde ς é a função saúde da bateria, t é o tempo de duração da bateria desde a última recarga e $\hat{t}$ é o tempo de duração da bateria estimado com base na capacidade total.

$$\sigma = \frac{b}{\hat{b}} * \varsigma \tag{3.12}$$

Onde σ é a função duração da bateria, b é a capacidade atual da bateria e $\hat{b}$ é a capacidade total da bateria.

$$\begin{aligned} \rho &= 1,\ se\ d < \hat{d}, \\ \rho &= 0,\ se\ d \geq \hat{d} \end{aligned} \tag{3.13}$$

Onde ρ é a proximidade, d é a distância do sensor em relação ao ponto origem e $\hat{d}$ é a distância máxima definida pelo usuário.

$$\kappa_i = min(\varphi)/\varphi_i \tag{3.14}$$

Onde κ_i é a nota para o critério preço/custo de um sensor i, φ_i é o valor do preço/custo do sensor i e $min(\varphi)$ é o menor preço/custo do conjunto de sensores avaliado.

$$\omega = (\sum_{i=0}^{n} c_i) * \kappa \tag{3.15}$$

Onde ω é o custo-benefício, c_i é o critério i, n é o número de critérios usados na avaliação do sensor sem incluir o critério preço/custo e κ é o critério preço/custo associado ao sensor avaliado.

3.2 Métodos de Agregação

Os métodos de agregação permitem atribuir a nota agregada de *QoC*, a partir das notas de critérios individuais, permitindo realizar ranqueamento de dispositivos.

3.2.1 Média Ponderada

Embora o uso de média ponderada (*weighted average*, ou *WAvg*) seja encontrado em diversos textos na literatura, aqui optou-se por utilizar a versão genérica do método

encontrado em (NAZÁRIO et al., 2015), que é descrita pela fórmula:

$$WAvg = \frac{\sum_{i=1}^{n} c_i.w_i}{\sum_{i=1}^{n} w_i} \tag{3.16}$$

Onde c_i é o valor de *QoC* para o critério *QoC i*, w_i é o peso atribuído ao critério *i* e *n* é o número de critérios *QoC* considerados.

3.2.2 Distância Euclidiana

Este método é apresentado em (PERERA et al., 2013), sendo descrito pela fórmula:

$$EDB = \sqrt{\sum_{i=1}^{n} [W_i.(U_i^d - S_i^\alpha)^2]} \tag{3.17}$$

Onde W_i é o peso normalizado para o critério *i*, dado pela razão entre o peso atribuído e a soma de todos os pesos atribuídos; U_i^d é o valor definido por um usuário para o critério i em um sensor considerado ideal e S_i^α é o valor atribuído ao sensor α, com α sendo qualquer sensor em um conjunto de sensores avaliados, para o critério i.

O sensor ideal por padrão recebe todos os valores em 1, em que o ideal é sempre o melhor sensor. A alteração na configuração de sensor ideal permite, por exemplo, encontrar sensores com problemas, pois se forem atribuídos valores baixos para o sensor ideal, um ranqueamento resultante colocará os sensores com baixo valor de *QoC* no topo.

3.2.3 Métodos de agregação encontrados no *Exehda-QoC*

Os métodos compilados nesta seção são todos apresentados em (XAVIER et al., 2017), mas aqui foram simplificados para abstrair algumas das características do *Exehda-QoC*. Assim, para todos eles, se considera que $\widehat{w_k}$ é o peso normalizado para o conjunto de critérios k considerado e μ_k é o valor atribuído após avaliação de *QoC* do critério k:

$$fminimo = min(\mu_k) \tag{3.18}$$

$$fgeometrica = \prod \mu_k^{\widehat{w_k}} \tag{3.19}$$

$$fharmonica = (\sum \frac{\widehat{w_k}}{\mu_k})^{-}1 \tag{3.20}$$

$$flimiar = se\,\forall k(\mu_k \geq \widehat{w_k}),\, entao\, 1,\, senao\, 0), \tag{3.21}$$

Para a equação de *CES* a seguir, existem dois parâmetros adicionais (ρ e κ), que controlam, respectivamente, o grau de homogeneidade e a elasticidade da função:

$$fCES = [\widehat{w_k}\mu_k^{\rho}]^{\frac{\kappa}{\rho}} \tag{3.22}$$

3.3 Método *AHP*

O método *AHP* (*Analytic Hierarchy Process*) (SAATY, 2008) visa permitir tomar decisões a partir da atribuição de prioridades para diferentes critérios e para os elementos que contenham estes critérios. Por exemplo, para decidir qual é o melhor candidato para um determinada vaga de emprego, são estabelecidos critérios como liderança, escolaridade e idade (Figura 3.1).

Primeiramente são estabelecidas as prioridades entre um critério e outro. Depois disso, são feitas comparações, dois a dois, para cada um dos critérios, entre os candidatos. A partir destas comparações, são criadas matrizes de prioridade (Figura 3.2), que permitem, através de um método matemático que utilize matrizes, gerar uma nota final para cada um dos candidatos, auxiliando na tomada de decisão por parte de quem está contratando para a vaga.

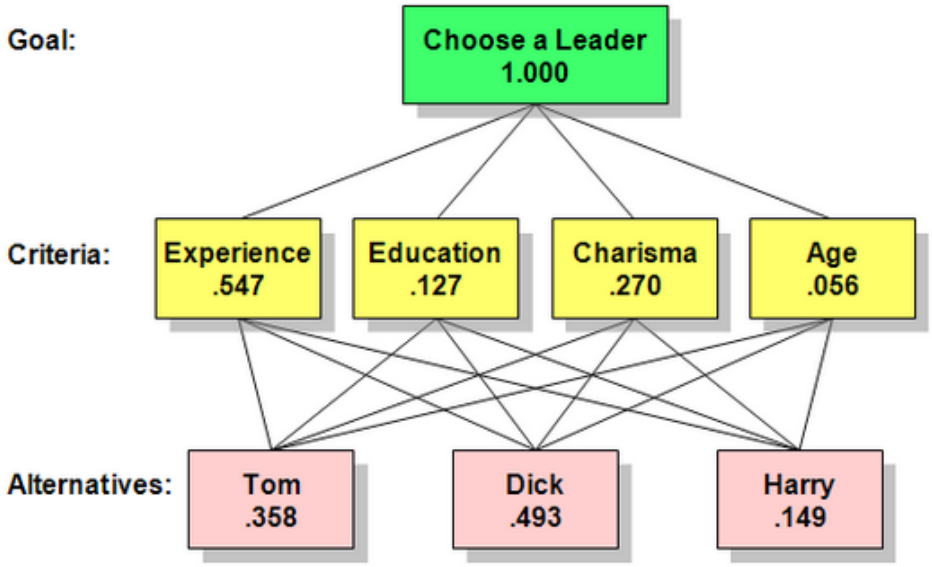
Saaty recomenda o uso do método de *eigenvectors* (SAATY; HU, 1998) para realizar os cálculos e obter as prioridades entre as opções.

Um aspecto importante a ser considerado é o fator de consistência, entre 0 e 1, que indica o quanto um determinado resultado de prioridades é confiável. Quanto mais próximo de 0 é o fator, mais consistente é o resultado, e, portanto, uma ordem de prioridades originada de uma aplicação do método com fator próximo de 0 é considerada como sendo o mais próximo possível de um ranqueamento ideal (SAATY, 2008).

Desta forma, para verificar a qualidade de ranqueamentos feitos através de outros métodos, é possível utilizar o resultado de um ranqueamento realizado utilizando *AHP* com fator de consistência próximo de 0, estratégia que será adotada para comparar alguns dos métodos de agregação apresentados na seção anterior.

Para obter a ordem de prioridades através de *AHP*, será utilizado um software especializado, descrito na seção 6.3. Recomenda-se a leitura dos artigos citados para entendimento aprofundado sobre o método.

Figura 3.1: Exemplo de estruturação de um modelo *AHP*



Fonte: (Lou Sands, 2019), adaptada

Figura 3.2: Matrizes de prioridade. 1) é a matriz de comparação entre critérios e 2) é a matriz de comparação para o critério "Experiência"

1)

Criteria	Experience	Education	Charisma	Age	Priority
Experience	1	4	3	7	0.547
Education	1/4	1	1/3	3	0.127
Charisma	1/3	3	1	5	0.270
Age	1/7	1/3	1/5	1	0.056
			Sum of Priorities		1.000
			Inconsistency		0.044

2)

Experience	Tom	Dick	Harry	Priority
Tom	1	1/4	4	0.217
Dick	4	1	9	0.717
Harry	1/4	1/9	1	0.066
		Sum of Priorities		1.000
		Inconsistency		0.035

Fonte: (Lou Sands, 2019), adaptada

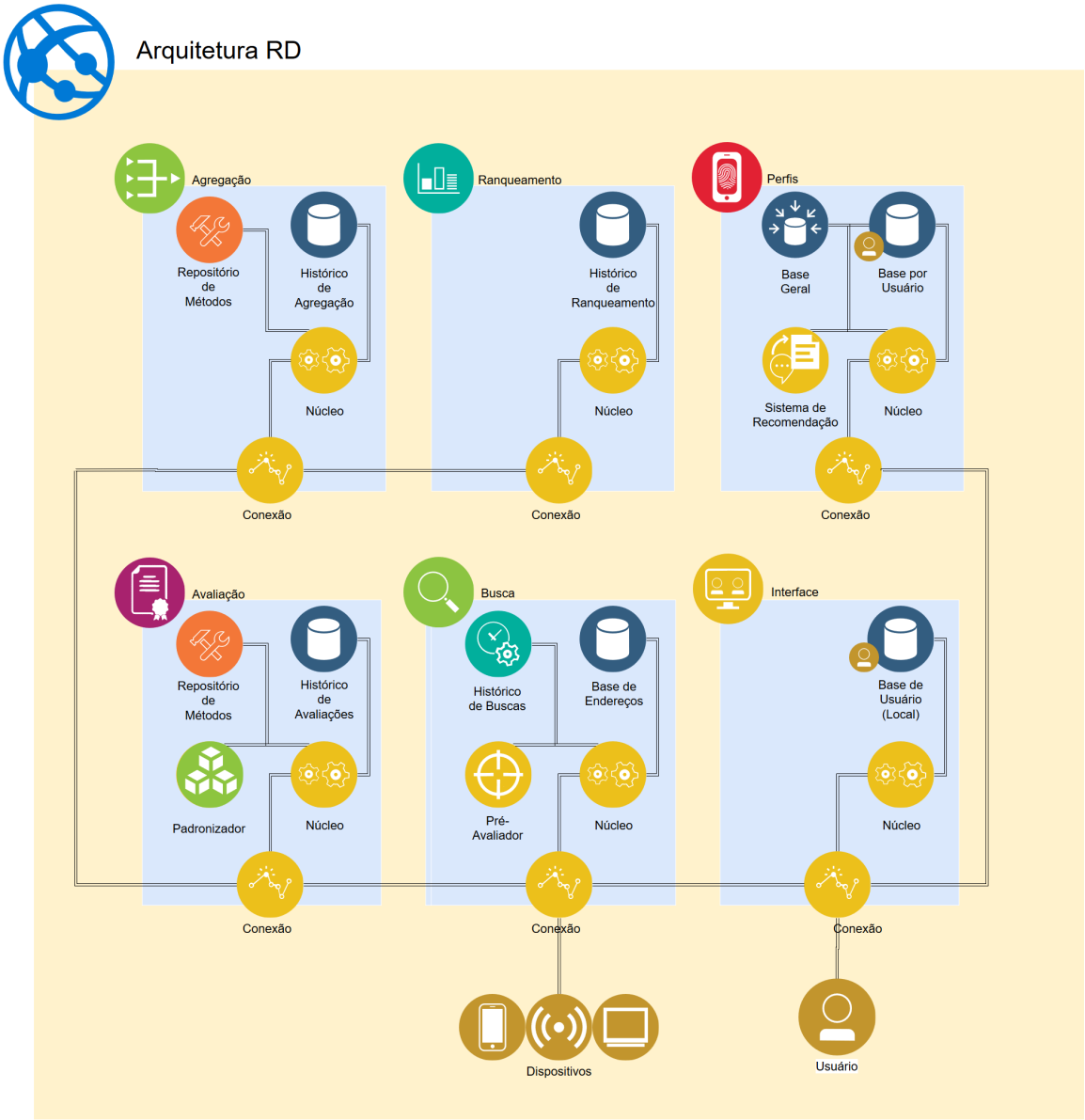
4 ARQUITETURA *RD*

Neste capítulo será abordada a modelagem e construção da Arquitetura *RD*, concebida para seleção e ranqueamento de dispositivos *IoT* por meio de critérios de *QoC* e agregação de critérios para gerar um valor único sobre o qual é feito o ranqueamento destes dispositivos, de acordo com a qualidade do contexto gerado e dos requisitos apresentados pelo usuário, que deseja obter dados dos dispositivos mais adequados para a aplicação que busca desenvolver.

A Arquitetura RD busca satisfazer os requisitos encontrados na literatura para seleção de dispositivos *IoT* e na avaliação dos valores de *QoC* dos diversos parâmetros e para diferentes dispositivos, de forma a permitir utilizar recursos computacionais distribuídos e *Edge Computing*, fazendo uso da capacidade computacional de dispositivos móveis. A partir destes requisitos, foi feito o estudo das capacidades a serem incluídas na arquitetura, dando origem ao *modelo RD* (Figura 4.1).

A estrutura modular adotada habilita a realização das diversas etapas do processo de avaliação de *QoC* e seleção de dispositivos *IoT* em diferentes tipos de dispositivo, permitindo a posterior avaliação das melhores configurações em termos de uso de recursos computacionais e tempo de resposta para os usuários.

Figura 4.1: Visão Geral da Arquitetura RD e seus componentes. Todas as linhas representam comunicação bidirecional entre os submódulos, dispositivos e usuários



Fonte: Autor, usando a ferramenta draw.io (JGRAPH LTD., 2018)

4.1 Estrutura de Módulos

A estrutura da RD (Figura 4.1) é composta por 6 módulos distintos: *Pesquisa/-Busca*, *Avaliação de Critérios de QoC*, *Agregador*, *Ranqueamento e Seleção*, *Perfis QoC* e *Interface*. A funcionalidade de cada um dos módulos é apresentada a seguir. Por simplicidade, as definições serão apresentadas levando em conta a interação com um usuário, mas a interação também poderia ser realizada com uma aplicação.

4.1.1 Pesquisa/Busca

O módulo de Pesquisa/Busca (Figura 4.2) de dispositivos *IoT* utiliza parâmetros informados pelo usuário por via do módulo de Interface, tais como tipo de sensor (e.g. acelerômetro, termômetro, medidor de tráfego de rede, etc.), localização e alcance da busca (e.g. buscar em um raio de 1000 metros a partir do ponto onde o usuário está), e então seleciona os dispositivos que satisfazem os parâmetros e busca seus dados brutos sobre todas as informações de contexto que estão disponíveis.

Após o recebimento dos dados, o módulo seleciona apenas os critérios que o usuário deseja que sejam considerados na avaliação final, de acordo com a aplicação a ser desenvolvida, e repassa os dados pertinentes ao módulo de Avaliação de Critérios *QoC*.

É possível também selecionar dispositivos que não sejam do tipo informado pelo usuário, mas que possam prover a mesma informação por aproximação, desde que o usuário indique por meio da Interface que deseja a inclusão destes dispositivos na busca. Neste caso, os dispositivos de tipos similares serão marcados como sendo de um tipo diferente do desejado, o que será informado ao módulo de Avaliação de Critérios *QoC*, e influenciará na nota final atribuída pelo módulo Agregador.

Para consultar os dispositivos, é possível utilizar diferentes métodos de obtenção dos dados referentes a estes, incluindo o uso de linguagens de consulta *RDF*, tais como a linguagem considerada padrão na área, *SPARQL* (PRUD; SEABORNE et al., 2006), *GraphQL* (Facebook Inc., 2018), entre outras, além de soluções ad hoc ou implementações de métodos de consulta próprios implementados pelo usuário, usando geralmente o padrão *publish-subscribe* (BIRMAN; JOSEPH, 1987).

O módulo de busca inclui ainda uma base de dados com os endereços de dispositivos com os quais se conectou anteriormente, bem como um histórico dos resultados das últimas buscas, visando ganho de performance no caso de requisições que já tenham sido realizadas anteriormente pelo usuário.

Além disso, conta com um submódulo para realizar pré-avaliações e retirar sensores que não satisfaçam os critérios básicos de distância e tipo, permitindo que a escolha de sensores seja mais flexível, de acordo com a necessidade do usuário.

A estrutura do módulo de Pesquisa/Busca conta com cinco submódulos: *Base de endereços*, *Histórico*, *Pré-avaliador*, *Comunicação* e *Núcleo*. O Núcleo é o responsável por realizar as principais tarefas do módulo, recebendo as informações do usuário e coordenando os outros submódulos.

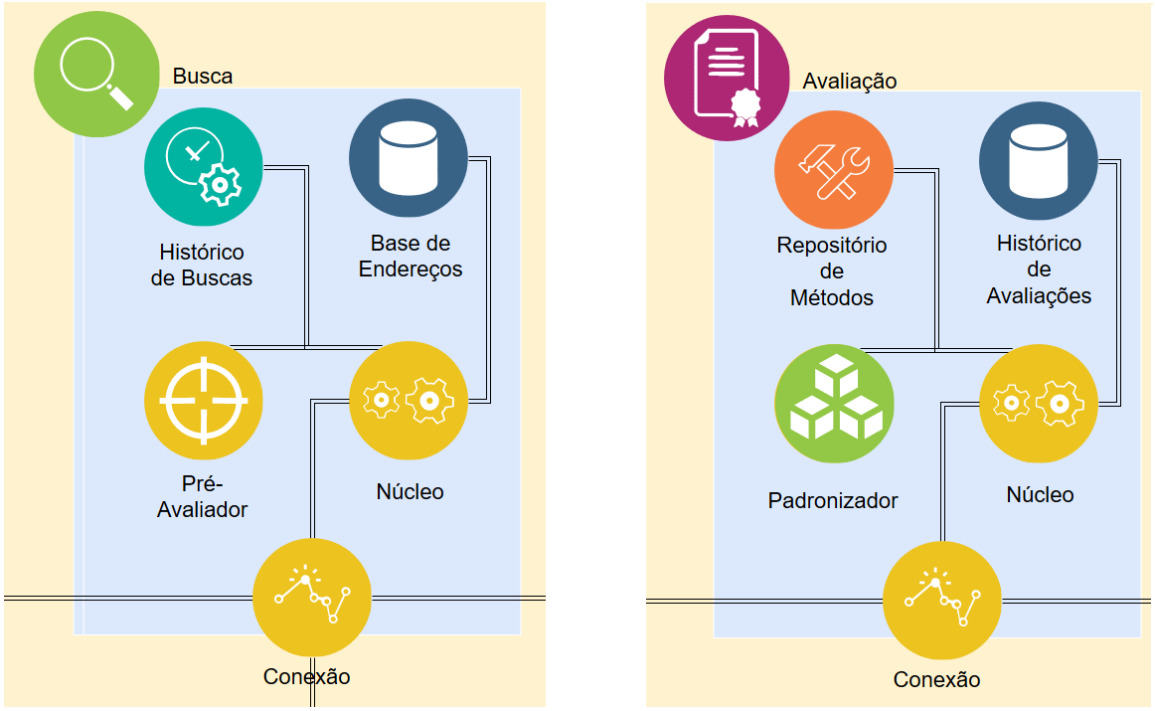
O Pré-Avaliador é responsável por descartar os sensores que não se enquadram nas características básicas de tipo e distância, gerando economia de recursos ao não obter e enviar aos outros módulos da arquitetura dados que não serão úteis.

O Histórico armazena os resultados de pesquisas realizadas anteriormente, permitindo que, ao ser realizada uma requisição de usuário que seja igual a uma requisição realizada anteriormente, seja tentada a comunicação com os mesmos sensores que satisfizeram esta requisição anterior, e, em casos em que a *atualidade* dos dados não seja considerada importante para o usuário, será evitada a consulta aos sensores, com os dados armazenados sendo diretamente enviados para os outros módulos.

A Base de endereços armazena os endereços de rede dos dispositivos que foram encontrados em buscas anteriores, poupando tempo nas comunicações ao direcionar as mensagens diretamente para estes endereços.

Por fim, o submódulo de Comunicação é o responsável por transmitir e receber os pacotes com as requisições e dados contextuais dos sensores, bem como retransmiti-los aos outros módulos da arquitetura.

Figura 4.2: Detalhe dos módulos de Busca e Avaliação. "Conexão" é o submódulo de Comunicação



Fonte: Autor

4.1.2 Avaliação de Critérios de *QoC*

O módulo de Avaliação de Critérios *QoC* (Figura 4.2) é responsável por aplicar fórmulas definidas para cada um dos critérios/parâmetros de *QoC* considerados de interesse pelo usuário e que foram enviados pelo modo de Pesquisa/Busca para medir o valor correspondente de *QoC* individualmente. Este valor é sempre medido entre 0 e 1, seguindo o padrão estabelecido na literatura. Estas fórmulas têm a definição armazenada em um Repositório de Métodos.

Cada critério pode ter diferentes fórmulas para o seu cálculo. Cabe aos usuários especialistas de domínio de aplicação definir qual fórmula deve ser utilizada através da Interface, de acordo com suas necessidades em termos de precisão, disponibilidade de tempo etc. Diferentes conjuntos de fórmulas pré-definidos podem ser salvos e selecionados posteriormente graças ao sistema de Perfis *QoC*, explicado na seção 4.1.5.

Ao terminar a avaliação dos critérios/parâmetros, o módulo deve repassar os dados avaliados, agrupados por origem, ao módulo Agregador.

O módulo de Avaliação de Critérios de *QoC* tem 5 submódulos: *Repositório de métodos*, *Histórico*, *Padronizador*, *Comunicação* e *Núcleo*.

O Núcleo é o responsável por coordenar os outros submódulos e aplicar os métodos de cálculo para atribuir os valores de *QoC* aos parâmetros indicados pelo usuário.

O Repositório de métodos armazena diferentes métodos de avaliação de critérios, que são escolhidos de acordo com a requisição do usuário e aplicados pelo Núcleo aos diferentes critérios.

O Padronizador é responsável por transformar os dados brutos em um formato padronizado que possa ser utilizado pelos métodos de cálculo presentes no repositório. Por exemplo, ao receber dados sobre o tempo de obtenção dos dados para o cálculo do critério *atualidade*, é possível que cada sensor utilize um formato de data/hora diferente, sendo necessário adequar estas informações para o método de cálculo escolhido pelo usuário.

O Histórico do módulo de Avaliação armazena operações e resultados de requisições que já tenham sido feitas no passado, permitindo ganho de performance ao obter os valores de critérios que já tenham sido utilizados no passado, bem como o uso dos dados para fins estatísticos.

O módulo de Comunicação é responsável por receber os dados do módulo de Pesquisa/Busca e enviar os resultados obtidos pela aplicação dos métodos de cálculo para o módulo Agregador.

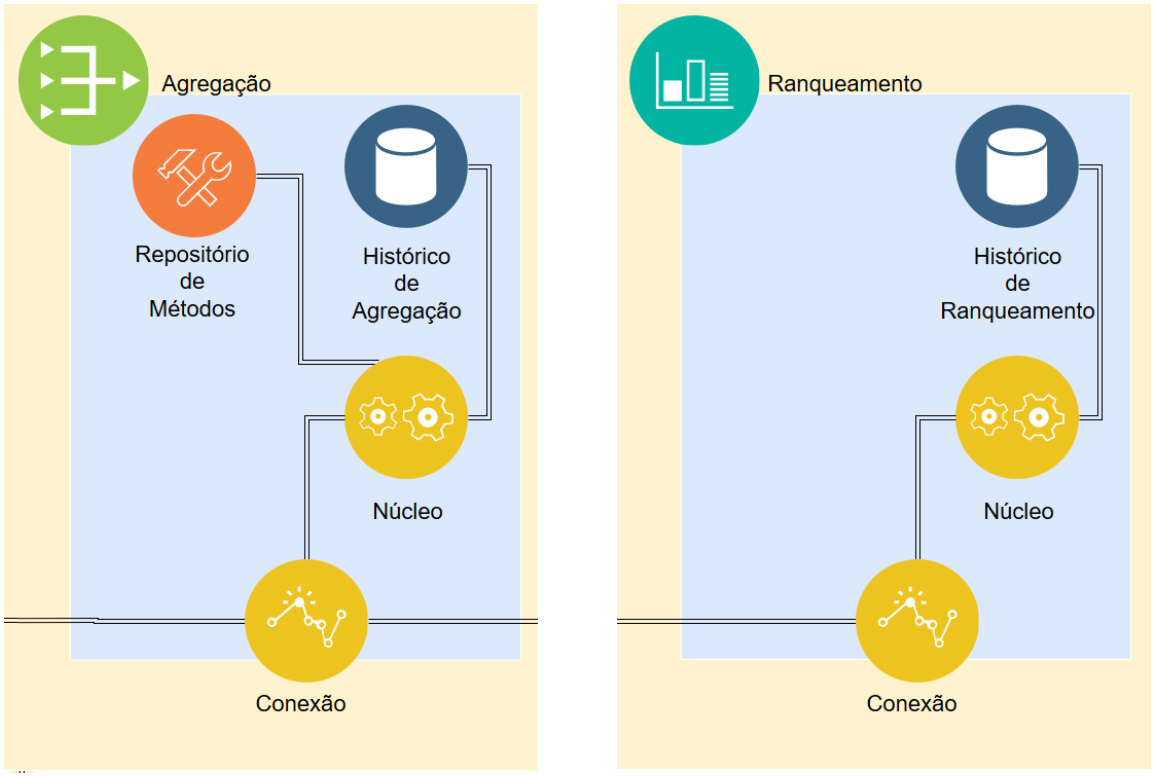
4.1.3 Agregador

O módulo Agregador (Figura 4.3) recebe os valores de *QoC* dos critérios individuais, agrupados por dispositivo de origem, do módulo Avaliador de *QoC*, bem como os pesos e outros requerimentos do usuário enviados pelo módulo de Interface, e aplica uma fórmula de agregação para gerar um valor único de *QoC* para cada dispositivo. Se o dispositivo tiver sido marcado pelo módulo de Pesquisa/Busca como sendo de um tipo diferente do desejado pelo usuário, um fator de redução da nota será aplicado ao valor agregado.

O módulo Agregador permite o uso de diferentes formas de agregação, sendo que a fórmula desejada é escolhida pelo usuário, e pode fazer parte de um Perfil *QoC*. Isto dá ao usuário liberdade para avaliar os dispositivos sob diferentes perspectivas, de acordo com a aplicação que usará os dados destes dispositivos.

O resultado da operação de agregação é então passado ao módulo de Seleção e Ranqueamento. O módulo de Agregação é composto por 4 submódulos: *Repositório de Métodos de Agregação*, *Histórico*, *Núcleo* e *Comunicação*.

Figura 4.3: Detalhe dos módulos de Agregação e Ranqueamento



Fonte: Autor

O Núcleo, como nos módulos anteriores, coordena os outros submódulos, e é o responsável por aplicar os métodos de agregação sobre os valores de *QoC* recebidos do módulo de Avaliação de Critérios de *QoC* e utilizando os pesos e preferências passadas pelo usuário através da Interface.

O Repositório de métodos de agregação armazena diferentes métodos, da mesma forma como é feito com os métodos de cálculo de critérios individuais no Avaliador de Critérios *QoC*.

O Histórico funciona de maneira similar ao módulo Avaliador de Critérios *QoC*, armazenando o resultado de agregações feitas para requisições anteriores, para que possa ser utilizado quando o usuário fizer uma nova requisição que utilize os mesmos dados e para fins estatísticos.

O submódulo de Comunicação é o responsável por receber os dados do Avaliador de Critérios *QoC* e enviar os resultados agregados para o módulo de Seleção e Ranqueamento.

4.1.4 Seleção e Ranqueamento

O módulo de Seleção e Ranqueamento (Figura 4.3) é responsável por selecionar os sensores que mais se adequem às exigências do usuário. Para tanto, a lista de valores passada pelo módulo de agregação é ranqueada em ordem crescente ou decrescente de valor. A etapa de ranqueamento também pode ser pulada se o usuário assim desejar. É então feita a seleção dos dispositivos e seus respectivos valores de *QoC* agregado, de acordo com o número necessário de dispositivos que o usuário informou por meio da Interface. Esta seleção é retornada para a Interface e exibida ao usuário.

O módulo de Seleção e Ranqueamento tem 3 submódulos: *Núcleo*, *Histórico* e *Comunicação*.

O Núcleo é o responsável por ordenar e gerar as listas de ranqueamento, bem como selecionar os sensores, de acordo com o número de sensores desejado pelo usuário, além de coordenar os outros dois submódulos.

O Histórico, similarmente ao que acontece com os outros módulos, armazena as últimas operações realizadas, permitindo retornar rapidamente listas que foram passadas para consultas do usuário que tenham a mesma forma de consultas anteriores. Ao verificar o histórico de atividades, este submódulo deve garantir que os resultados anteriores ainda são válidos. Para tanto, deve verificar se o usuário atribuiu importância ao critério

atualidade. Se alguma importância for atribuída, o histórico é ignorado, da mesma forma como ocorre nos módulos descritos anteriormente.

Se não for atribuída importância pelo usuário ao critério atualidade, o módulo de Seleção e Ranqueamento se comunicará com o módulo de Pesquisa/Busca para garantir que a comunicação com os sensores ainda é possível, e utilizará a mesma lista de sensores.

Por fim, o submódulo de Comunicação recebe os valores agregados pelo módulo Agregador, e envia as listas com os sensores ranqueados e selecionados para a Interface.

4.1.5 Perfis *QoC*

O módulo de Perfis *QoC* (Figura 4.4) é responsável por salvar escolhas anteriormente informadas pelos usuários em relação a parâmetros e métodos de acordo com a descrição de uma aplicação. Por exemplo, se um usuário definiu suas necessidades através da Interface para uma aplicação de controle do clima, pode salvar estas definições informando à Interface o nome "Seleção de Medidores de Temperatura/Clima". Estas definições são passadas ao módulo de Perfis *QoC*, que as armazena em sua base de dados.

Assim, o próprio usuário ou algum outro usuário que deseje obter informações para uma aplicação de controle do clima pode escolher esta opção na Interface. O módulo de Perfis de *QoC* será então informado pela Interface sobre a escolha do usuário, buscando as configurações salvas em sua base de dados e permitindo ao usuário poupar tempo.

No entanto, é possível que as definições salvas para uma determinada aplicação não correspondam às necessidades do usuário. Neste caso, o usuário pode informar suas próprias definições e salvá-las. Estas novas definições serão mantidas associadas ao usuário na base de dados do módulo de Perfis *QoC*.

No exemplo, temos então duas definições de parâmetros e métodos diferentes para a mesma aplicação. Para que um terceiro usuário pudesse utilizar definições prontas, seria necessário escolher entre uma delas. Com apenas duas definições, realizar esta escolha é viável. Entretanto, o sistema poderá registrar muitas definições diferentes para a mesma descrição de aplicação, aumentando muito o tempo necessário para que o usuário possa avaliar as opções e selecionar aquela que satisfaça seus requisitos.

Tendo isto em mente, o módulo de Perfis *QoC* deve contar com um algoritmo que permita definir um padrão para a descrição, gerando apenas uma opção para usuários que não tenham um padrão associado na base de dados.

Outro problema que deve ser considerado diz respeito a aplicações com descrições

diferentes, mas com a mesma definição de parâmetros e métodos. Para tratar este caso, o módulo de Perfis *QoC* precisa de um algoritmo que identifique estes casos e permita juntá-los, tanto de maneira manual como de maneira automática.

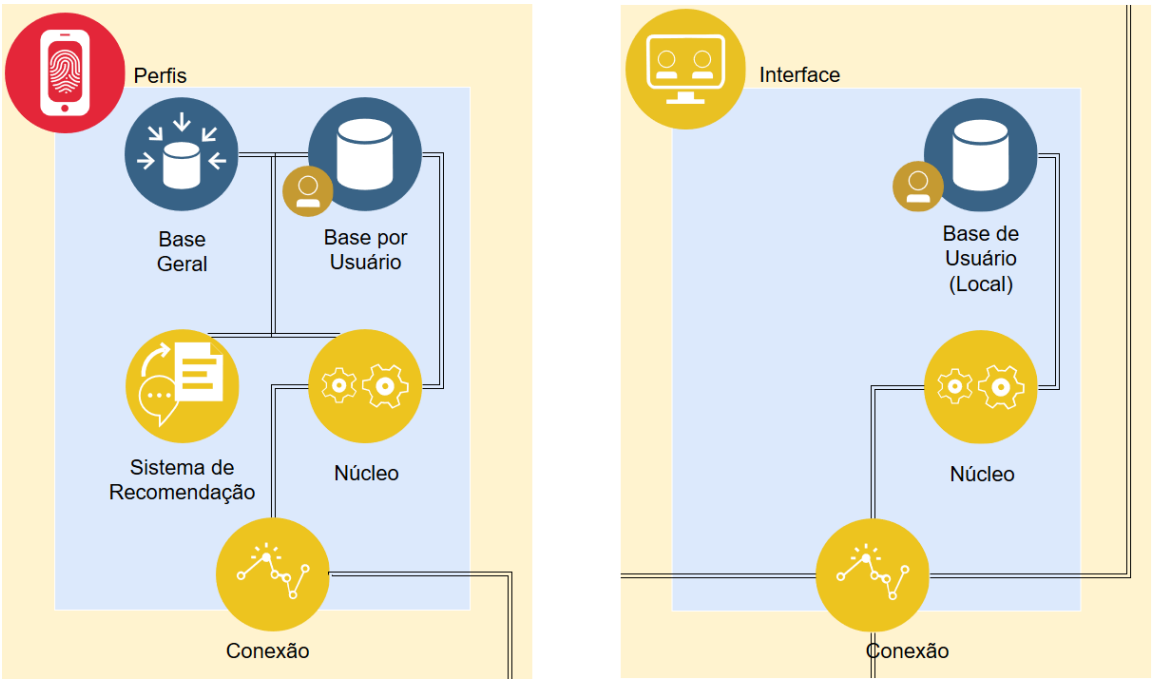
Quanto à estrutura do módulo de Perfis *QoC*, cinco submódulos a compõem: *Base Geral*, *Base por Usuário*, *Sistema de Recomendação*, *Núcleo* e *Comunicação*.

O Núcleo é o responsável por gerenciar a seleção de perfis, definindo de qual base será obtido o perfil *QoC* recomendado para o usuário e coordenando os outros submódulos.

A Base Geral armazena os dados de preferências de todos os usuários, permitindo a aplicação de algoritmos para a seleção de um padrão que possa ser compartilhado com todos os usuários que utilizem a arquitetura. O objetivo de um padrão é permitir selecionar dispositivos sem que seja necessário fazer ajuste fino dos critérios, poupando tempo aos usuários, bem como possibilitando aos usuários com menor conhecimento sobre o domínio de aplicação realizar seleções de dispositivos.

A Base por Usuário armazena de forma agrupada as preferências dos usuários, permitindo que seja feita uma seleção rápida de um perfil de forma a ignorar um perfil padrão definido a partir dos dados da Base Geral.

Figura 4.4: Detalhe dos módulos de Perfis e Interface



Fonte: Autor

No entanto, estes mesmos dados são armazenados também na Base Geral, sem a identificação do usuário, para permitir eventualmente melhorar os perfis padrão a serem apresentados aos outros usuários.

O Sistema de Recomendação é o responsável por gerenciar os padrões para as descrições e aplicar o algoritmo definido para obter o padrão, utilizando os dados da Base Geral para este fim.

O envio dos dados de perfis e a recepção das definições do usuário fica sob responsabilidade do submódulo de Comunicação.

4.1.6 Interface

O módulo de Interface (Figura 4.4) é responsável por receber todas as definições e atribuições feitas pelo usuário e transmiti-las aos outros módulos, bem como receber os valores retornados por estes módulos e exibi-los ao usuário.

Este módulo é definido pelas interações entre os outros módulos do sistema, sendo seus próprios requisitos dependentes dos requisitos funcionais dos outros módulos.

Assim, o mínimo a ser apresentado na Interface deve incluir a definição dos parâmetros *QoC* a serem considerados, os pesos atribuidos pelo usuário, os métodos a serem utilizados para o cálculo destes parâmetros, o método de agregação a ser utilizado, a quantidade desejada de dispositivos a serem selecionados, o método de ordenamento do resultado e os controles para o sistema de Perfis *QoC*.

A Interface deve também informar o sucesso ou a falha das operações intermediárias entre os módulos. É ainda responsável por armazenar uma cópia local das definições feitas pelo usuário, evitando operações desnecessárias de consulta ao perfil em uma base remota enquanto as definições de perfil do usuário não forem mudadas.

A Interface possui 3 submódulos: *Núcleo*, *Base Local por Usuário* e *Comunicação*.

O Núcleo realiza a integração de todos os aspectos da Interface, incluindo a exibição da Interface Gráfica, *I/O* e consultas à Base Local, e também coordena os outros módulos.

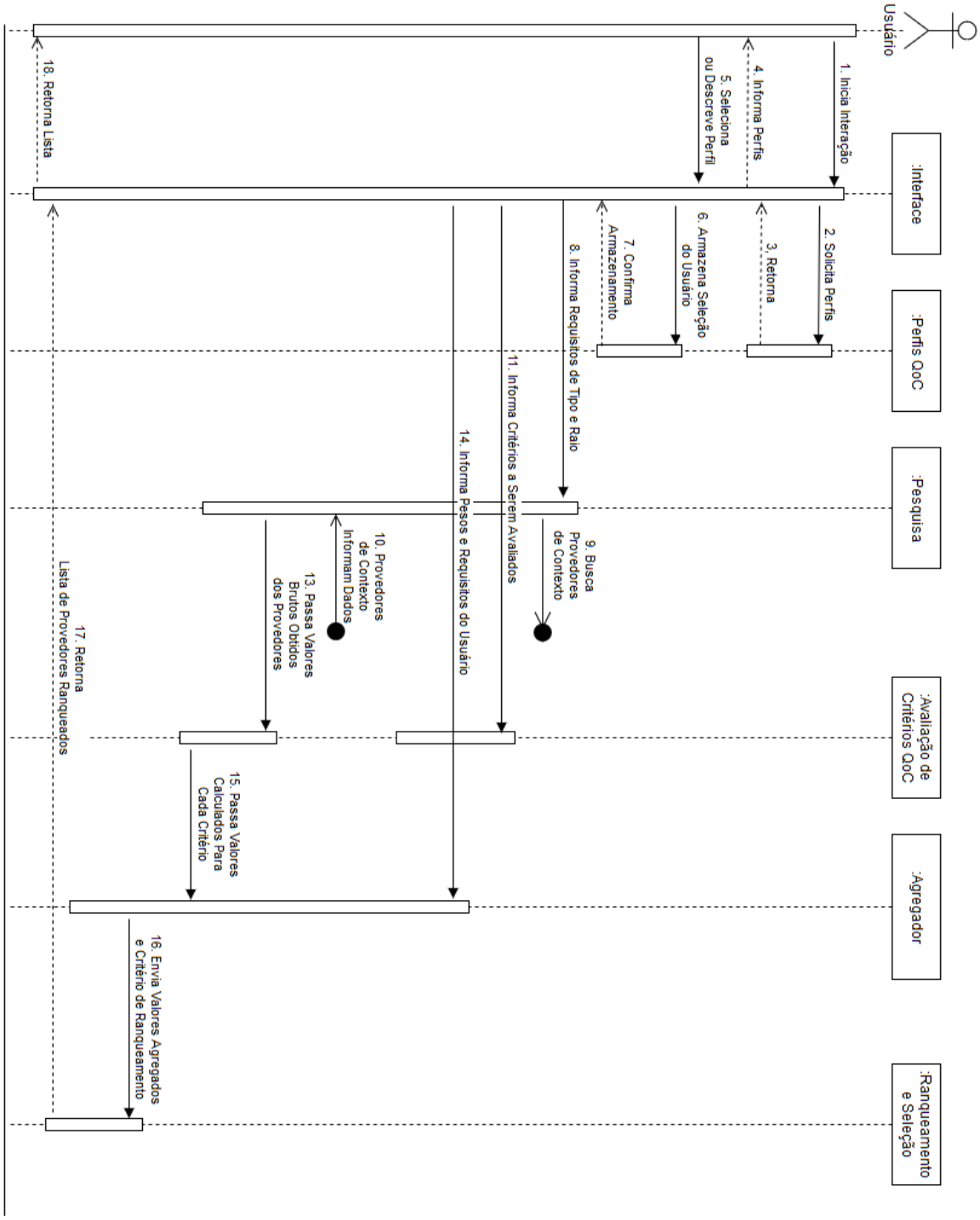
A Base Local por Usuário armazena as definições de perfis informadas pelo usuário, da mesma forma como o submódulo de Base por Usuário no módulo de Perfis *QoC*, permitindo evitar consultas desnecessárias à base remota, que é consultada apenas quando a Base Local não está disponível.

O submódulo de Comunicação deve realizar a comunicação com todos os outros módulos, recebendo os dados enviados por estes e distribuindo os aspectos relevantes obtidos de uma requisição do usuário para estes outros módulos.

4.2 Interação entre Módulos

As principais interações entre os módulos são apresentadas como um diagrama de interação *UML* (OBJECT MANAGEMENT GROUP INC., 2018). Especificamente, um diagrama de sequência (Figura 4.5). As interações são explicadas através da numeração exibida nas descrições das mensagens no diagrama. Algumas mensagens de confirmação passadas para a Interface e o usuário foram omitidas no intuito de simplificar a representação no diagrama.

Figura 4.5: Principais Interações da Arquitetura RD. Algumas das interações de retorno foram omitidas, por simplicidade.



Fonte: Autor

4.3 Considerações Finais

Esta seção apresentou a estrutura com os módulos da arquitetura *RD*, realizada de acordo com os requisitos de seleção de dispositivos *IoT* utilizando *QoC*, e também com os requisitos de *Edge Computing*. Os módulos de *Avaliação de Critérios de QoC*, *Agregador* e *Seleção e Ranqueamento* formam a base da aplicação dos conceitos de *QoC*, sendo o núcleo da arquitetura. A divisão das atribuições destes módulos foi feita pensando em utilizar e avaliar as possíveis vantagens do uso de *Edge Computing*, pois permite diferentes configurações e distribuições entre a nuvem e os dispositivos *Edge*, conforme será visto no capítulo 5.

O módulo de *Pesquisa/Busca* é o principal responsável por realizar a comunicação com os dispositivos *IoT*, e por isso deve abarcar os aspectos referentes à capacidade de localizar e estabelecer a troca de dados, de acordo com os protocolos de comunicação mais adequados.

O módulo de *Perfis QoC*, no âmbito desta pesquisa, tem a principal função de auxiliar na definição de possíveis padrões de critérios de acordo com a aplicação à qual se destinará a lista de dispositivos obtida utilizando estas definições.

5 PROTOTIPAÇÃO

Para avaliar a viabilidade e as capacidades da arquitetura RD, bem como a possível facilidade provida ao usuário utilizando o sistema de perfis, foram implementados protótipos para os módulos descritos no capítulo 4. Neste capítulo são apresentados estes protótipos, o ambiente de desenvolvimento utilizado e demais considerações feitas a partir da implementação.

5.1 Ambiente de Prototipação

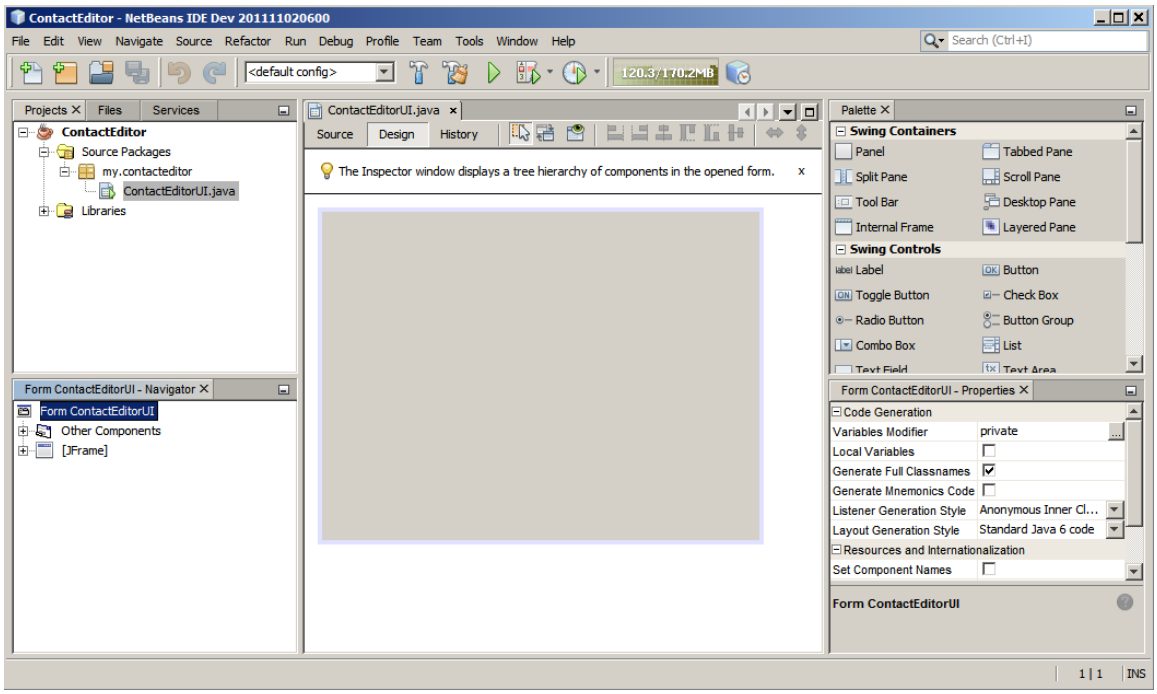
Diversas linguagens de programação, IDEs e Bibliotecas foram utilizadas para construir o protótipo, sendo apresentadas a seguir.

5.1.1 Linguagem Java e *IDE* NetBeans

Devido às características necessárias para realizar os testes quanto às possíveis vantagens no uso de capacidades de *Edge Computing*, foi definido que uma implementação da RD deveria permitir distribuir módulos entre os dispositivos próximos aos sensores e os dispositivos centrais da nuvem. Assim, para a implementação, optou-se por utilizar a linguagem Java (ORACLE CORPORATION, 2018b), na versão 1.8.0.77, pois além de permitir executar código em diversas plataformas sem necessidade de adaptações, também permite que o código seja utilizado em aplicações para *Android* (GOOGLE LLC, 2018a) facilmente.

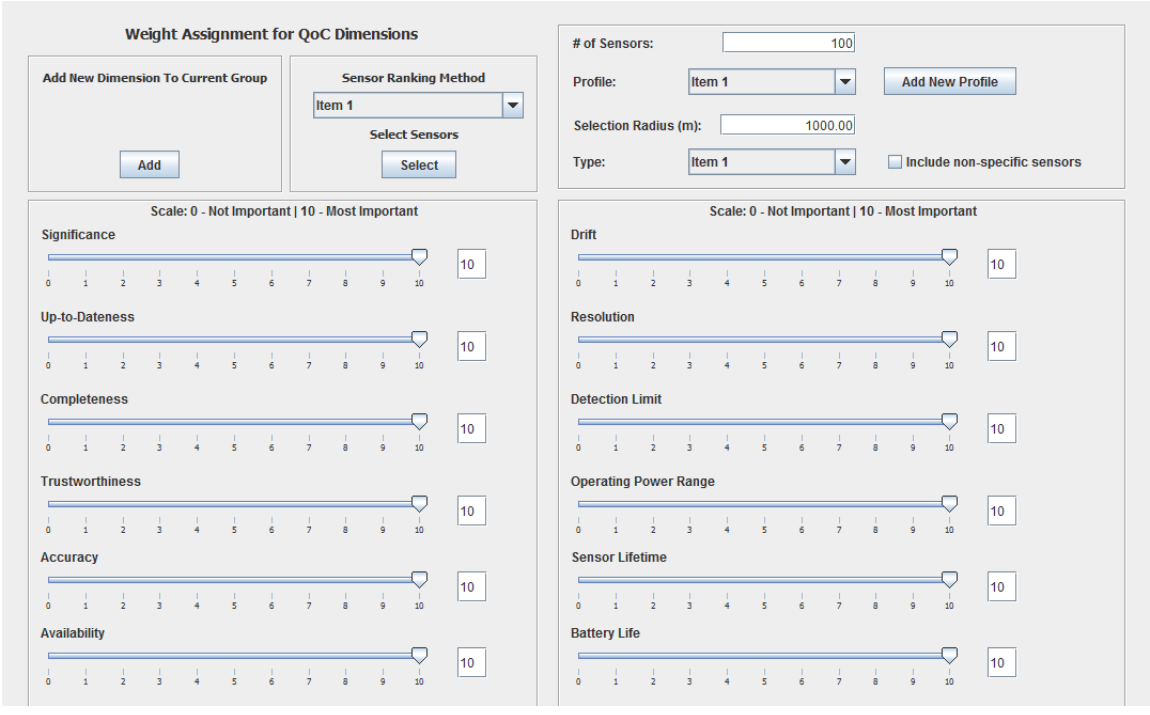
O código da aplicação em Java foi desenvolvido utilizando a *IDE* Netbeans (THE APACHE SOFTWARE FOUNDATION, 2018), em diversas versões, sendo a versão mais antiga utilizada a 7.0. Esta *IDE* foi escolhida pois além de permitir configurar facilmente o ambiente de execução, também permite construir interfaces gráficas de maneira direta e intuitiva, o que veio a ser útil para o desenvolvimento do módulo de Interface, na versão voltada aos usuários.

Figura 5.1: Interface da *IDE* Netbeans em modo de construção de interface gráfica.



Fonte: (THE APACHE SOFTWARE FOUNDATION, 2018)

Figura 5.2: Interface Gráfica do Módulo de Interface.



Fonte: Autor

5.1.2 *IDE Android Studio* e ferramenta *Profiler*

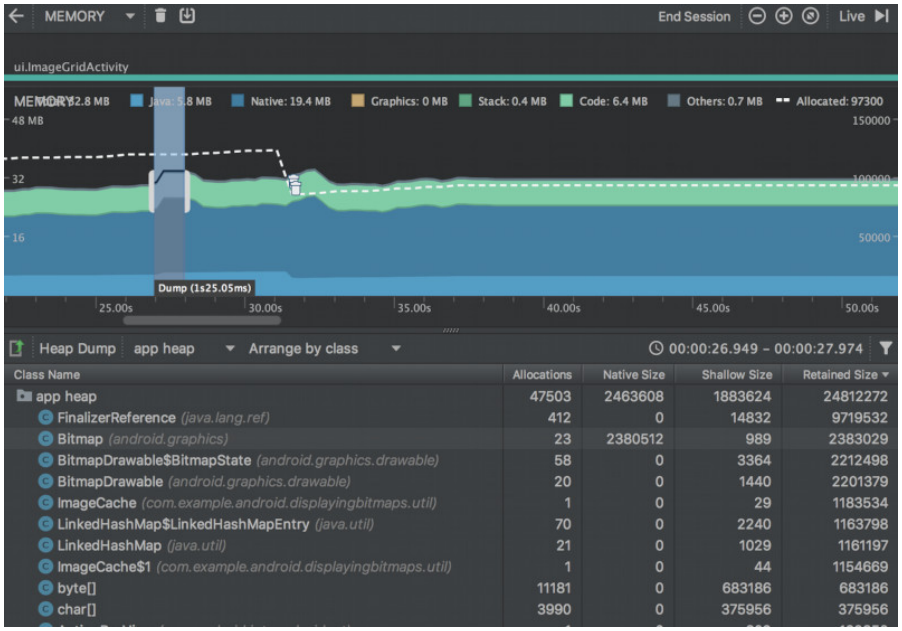
Para criar aplicações para a plataforma *Android*, foi utilizada a *IDE Android Studio* (GOOGLE LLC, 2018b) , *IDE* oficial do *Android*, que conta com diversas ferramentas para emulação, testes e medição de performance de aplicações de forma integrada à plataforma *Android*. As versões da *IDE* utilizadas variaram entre 2.6 e 3.3.

Uma das principais vantagens desta *IDE* é permitir o uso de ferramentas como o *Profiler* (GOOGLE LLC, 2018c), que permite acompanhar o uso de espaço em disco, memória e processamento, entre outros, de uma aplicação, que foi utilizada para medir o desempenho das diferentes configurações nas avaliações.

5.1.3 Linguagem C

Antes de implementar as versões em Java e *Android* dos métodos de cálculo de agregação, foi utilizada a linguagem C com suporte ao padrão 11 (OPEN-STD, 2018), através de uma versão do *GCC* para o *Windows*, o *Mingw-w64* (THE MINGW COMMUNITY, 2018), para fins de avaliação de viabilidade do uso dos métodos em razão do tempo computacional.

Figura 5.3: Ferramenta Profiler do Android Studio para análise de desempenho.



Fonte: (GOOGLE LLC, 2018c)

5.1.4 Bibliotecas

Para todas as implementações foram utilizadas apenas bibliotecas padrão das linguagens, exceto para a implementação em Java, onde foi necessário empregar a biblioteca *Java Microbenchmark Harness* (ORACLE CORPORATION, 2018a), voltada para realização de *benchmarks* de métodos de agregação.

5.2 Implementação

Os módulos e submódulos, bem como as diferentes configurações e uma representação algorítmica em pseudocódigo das principais tarefas executadas pela versão implementada da arquitetura RD são descritos nesta seção.

5.2.1 Módulos

Todos os módulos previstos para a arquitetura RD foram implementados sem, no entanto, terem sido implementados os submódulos com as bases de dados históricos. Apenas o módulo de Perfis QoC contou com uma base de dados implementada de maneira ad hoc para permitir testar a sugestão de parâmetros.

O algoritmo 1 mostra de forma simplificada todo o ciclo realizado em uma execução típica dos módulos da arquitetura para um dispositivo.

Algoritmo 1 Ciclo RD

1: **procedure** RD($dadosCrit[n], peso[n], criterio[n], metodoCalc[n], metodoAgrega$)
▷ n é o número de entradas
2: $r \leftarrow 0$
3: $Declara(resultado[n])$
4: **for** $i = 0; i < len(resultado); i++$ **do**
5: $resultado[i] \leftarrow 0$
6: **end for**
7: **while** $r < Comprimento(criterios)$ **do** ▷ enquanto houver critérios
8: $resultado[r] \leftarrow CalculoQoCIndividual(metodoCalculo[r], criterio[r],$
9: $dadosCriterio[r], peso[r])$
10: $r \leftarrow r + 1$
11: **end while**
12: **return** $Agregacao(metodoAgregacao, resultado[n])$
13: **end procedure**

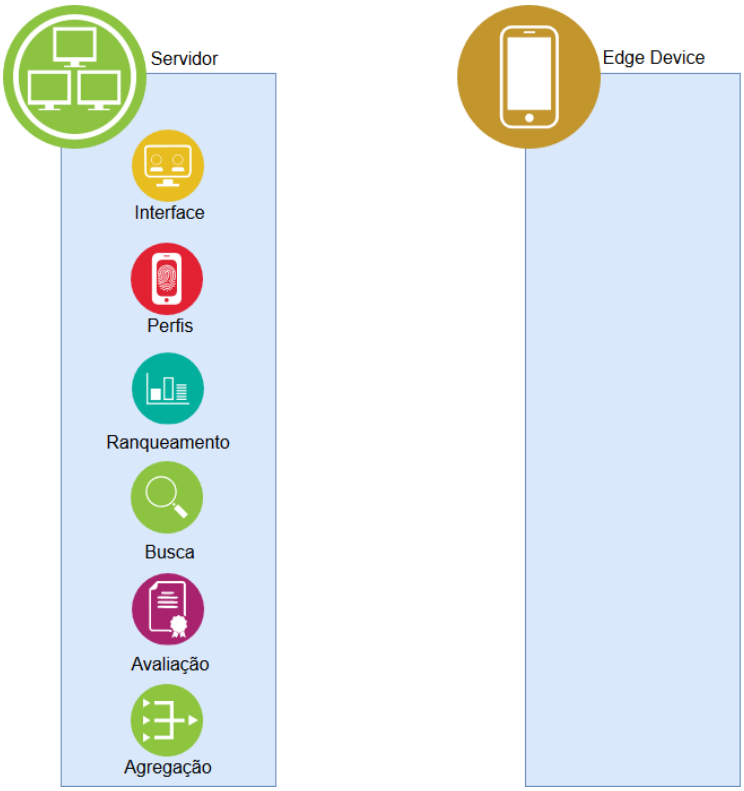
5.2.2 Configurações

Para realizar os testes, foi considerada uma configuração típica de *Edge Computing*, em que há uma nuvem, afastada dos dispositivos *IoT*, e um dispositivo próximo aos dispositivos *IoT*, o dispositivo *Edge*. Assim, foram implementadas três versões com configurações diferentes, distribuindo os módulos entre um servidor central, representando a nuvem, e o dispositivo *Edge* próximo aos dispositivos *IoT*:

1. Todos os módulos no servidor central (Figura 5.4);
2. Módulo de Avaliação no dispositivo *Edge* (Figura 5.5);
3. Módulos de Avaliação e de Agregação no dispositivo *Edge* (Figura 5.5).

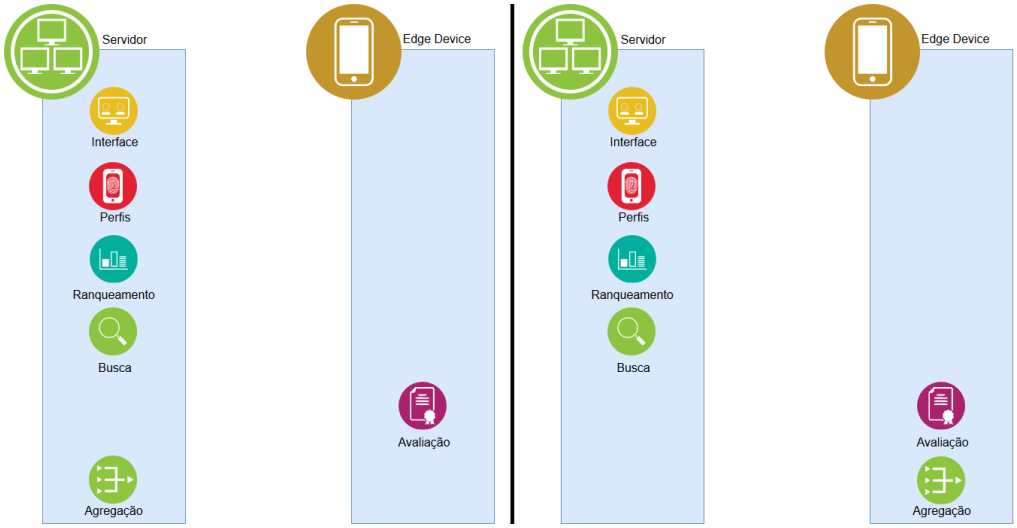
No caso 1, todo o código foi implementado em Java. Nos casos 2 e 3, os módulos que executam no dispositivo *Edge* foram adaptados para que executassem no *Android*.

Figura 5.4: Configuração 1: Todos os módulos no Servidor Central.



Fonte: Autor

Figura 5.5: Configuração 2 (à esquerda), apenas Avaliação de Critérios no *Edge Device*; e configuração 3 (à direita), Avaliação e Agregação no *Edge Device*.



Fonte: Autor

5.3 Métodos de Cálculo e Agregação

Foram implementados todos os métodos de agregação apresentados na seção 3.2, bem como todos os métodos de cálculo apresentados na seção 3.1. O método *AHP* não foi implementado devido ao alto custo computacional e complexidade de utilização, que o tornam não-recomendável quando há um grande número de elementos a serem avaliados, sendo usado apenas em ferramenta externa para o teste descrito na seção 6.3.

5.4 Considerações Finais

Neste capítulo foram apresentados alguns aspectos de implementação e escolha de ferramentas e linguagens de programação, de acordo com os requisitos apresentados pela arquitetura RD no capítulo 4. Foi apresentado também um pseudo-código, que permite entender facilmente os passos envolvidos em um ciclo de execução da implementação da arquitetura para avaliar e selecionar sensores.

As diferentes configurações de distribuição de módulos de acordo com a idéia de utilizar *Edge Computing* foram também apresentadas. O capítulo 6 a seguir apresenta a avaliação da implementação, melhor justificando as decisões de implementação tomadas.

6 AVALIAÇÃO DA ARQUITETURA

Para avaliar diferentes aspectos da arquitetura RD e sua implementação, bem como alguns de seus usos, foram feitos diversos testes e avaliações, tanto de caráter qualitativo como quantitativo.

Entre as avaliações realizadas está o uso das diferentes configurações implementadas no protótipo e apresentadas na seção 5.2.2, para avaliar os impactos do uso de *Edge Computing* em relação ao tempo computacional e ao uso de recursos; a comparação de dois métodos de agregação apresentados na literatura, no intuito de determinar a performance do método *EDB* apresentado na seção 3.2, e também se efetivamente pode ser usado como padrão conforme preconizam seus autores; a avaliação do sistema de Perfis *QoC*, buscando determinar sua utilidade para os usuários; e a realização de dois estudos de caso para apresentar o potencial da arquitetura na resolução de problemas reais.

6.1 Distribuição dos módulos para uso de *Edge Computing*

Para avaliar o impacto do uso de *Edge Computing* em um sistema baseado na arquitetura RD, e se o uso deste paradigma realmente reduz o tráfego de rede sem incorrer em uso proibitivo dos recursos computacionais em dispositivos móveis, foram executados testes com as três diferentes versões de implementação do protótipo apresentadas na seção 5.2.2. Para simular um ambiente de *cloud* com grandes recursos computacionais, foi utilizada uma *workstation Dell* com *Windows 10 Pro 64 bits*, utilizando processador *Intel Xeon CPU E3-1270 v5 3.6 GHz* e *RAM* de 32*GB*.

Foi utilizado também um *smartphone Motorola G4 Play*, por ser um modelo de entrada com especificações (Tabela 6.1) suficientes para a realização do experimento, similar aos telefones mais vendidos [1].

A avaliação, utilizando a ferramenta *Android Studio Profiler*, levou em conta o uso da rede no envio dos dados, o uso de memória, o uso de *CPU* e o uso de bateria. Para avaliar este último, no entanto, não foi possível utilizar o *Profiler*, pois a funcionalidade de medição de uso da bateria nesta ferramenta só está disponível a partir do *Android 7.0 Nougat*. Desta maneira, para avaliar o uso da bateria, foi registrado o percentual de uso

[1] Segundo a lista dos Top 20 smartphones mais vendidos em https://www.gsmarena.com/the_20_most_popular_phones_of_2017-news-28892.php (GSM ARENA, 2017)

Tabela 6.1: Especificações *Motorola G4 Play.*

OS	Android(TM) 6.0.1, Marshmallow
Arquitetura/Processador	Motorola Mobile Computing System, com processador quad-core 1.2 GHz Qualcomm® Snapdragon™ e GPU Adreno 306 450 Mhz
Memória (RAM)	2GB
Armazenamento (ROM)	Interna de 16GB, com suporte a microSD CARD 128GB
Peso	137g
Tela	5.0" 720p HD (1280x720) 294 ppi
Bateria	Bateria para uso diário (2800 mAh)
Redes	4G LTE (Cat 4) CDMA / EVDO Rev A UMTS / HSPA+ GSM / EDGE
Bandas	Moto G Play - XT1607 CDMA (850, 1900 MHz) GSM / GPRS / EDGE (850, 900, 1800, 1900 MHz) UMTS / HSPA+ (850, 900, 1700, 1900, 2100 MHz) 4G LTE (B1, 2, 3, 4, 5, 7, 8, 12, 13, 25, 26, 41) Moto G Play - XT1609 CDMA (850, 1900 MHz) 4G LTE (B2, 4, 5, 13)
Bluetooth®	Bluetooth versão 4.1 LE
Wi-Fi	802.11 b/g/n (2.4 GHz)

Fonte: (LENOVO, 2018), adaptada

indicado pelo sistema *Android* na tela de uso da bateria.

Desta maneira, o que se buscou foi equilíbrio entre redução de envio de dados na rede e aumento do uso de recursos computacionais disponíveis. Para realizar os testes, foram sempre considerados 30 critérios *QoC*, com base na quantidade de critérios utilizados no trabalho de (PERERA et al., 2013). No entanto, como não foram implementados 30 métodos de cálculo diferentes, foi realizada uma replicação dos métodos para os critérios, tendo sido utilizados para este fim os métodos de cálculo para *duração da bateria*, *saúde da bateria*, *proximidade*, *atualidade dos dados* e *completude*. Para evitar otimização por parte do *Android Runtime* (*ART*) ao utilizar repetição de métodos, os dados duplicados sofreram pequenas perturbações aleatórias.

Os dados utilizados para o envio de informações não-tratadas e para aplicação dos métodos foram todos obtidos pelos próprios sensores do telefone (que é também um dispositivo *IoT*), contendo informações sobre o acelerômetro, relógio, GPS, dados da bateria, demais dados de localidade e dados de descrição do telefone.

Todas as observações foram feitas durante o período de duas horas para cada configuração, excetuando-se a observação feita para a avaliação do uso de bateria, que foi realizada em um período de 12 horas para cada configuração. O módulo Agregador foi

configurado para executar apenas o método de média ponderada, mas foi carregado com todos os métodos implementados.

6.1.1 Avaliação de Tráfego de dados

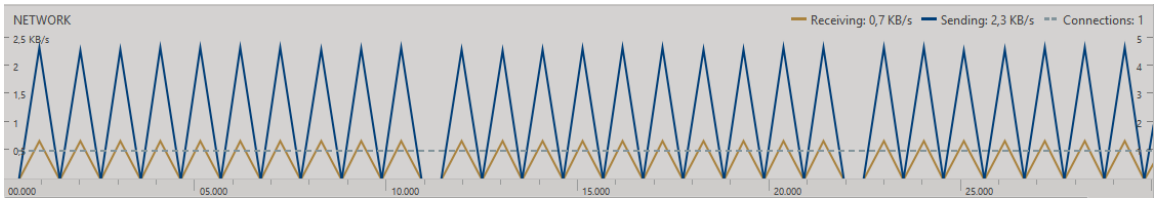
Para gerar os gráficos das Figuras 6.1, 6.2 e 6.3 foi utilizado um intervalo de 1000 milissegundos entre um envio e outro através da função *sleep* do *Android*. Analisando estes gráficos, é possível perceber que a quantidade de dados enviados pelo *Edge Device* para o Servidor Central, tanto na configuração 2 (Figura 6.2), com o Avaliador de Critérios *QoC* no *Edge Device*, quanto na configuração 3 (Figura 6.3), com o Avaliador e o Agregador no *Edge Device*, apresentam redução no envio de dados.

Isto se deve ao fato de que, enquanto na primeira configuração são sempre enviados dados não-tratados obtidos por meio dos sensores, na configuração 2 são enviadas apenas as notas obtidas para cada critério, e no caso da configuração 3, que apresentou o menor volume de dados, é enviado apenas um valor, que é a nota agregada atribuída ao dispositivo.

Assim, ao analisarmos apenas o aspecto de redução no tráfego de dados, o uso do paradigma *Edge Computing* permitiu reduzir a transmissão de dados, o que é vantajoso se considerarmos que *IoT* prevê o uso de inúmeros dispositivos enviando seus dados na rede. Desta maneira, uma redução de quase 83% no volume de dados trafegados utilizando a Configuração 3 (conforme a Tabela 6.2, sem considerar os dados enviados pelo Servidor ao *Edge Device*), ou mesmo uma redução de 26%, utilizando a Configuração 2, já representam economia suficiente para recomendar o uso de *Edge Computing*.

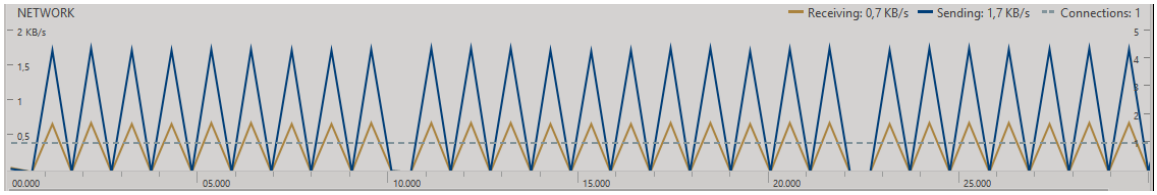
No entanto, é necessário analisar juntamente com o aspecto de redução de tráfego o aumento do uso de recursos computacionais no *Edge Device* decorrente dos cálculos efetuados pelos módulos de Avaliação e Agregação, buscando determinar se o *tradeoff* entre estas diferentes configurações é compensador, o que é apresentado nas próximas seções.

Figura 6.1: Tráfego de dados utilizando a Configuração de módulos 1: todos os módulos no Servidor Central.



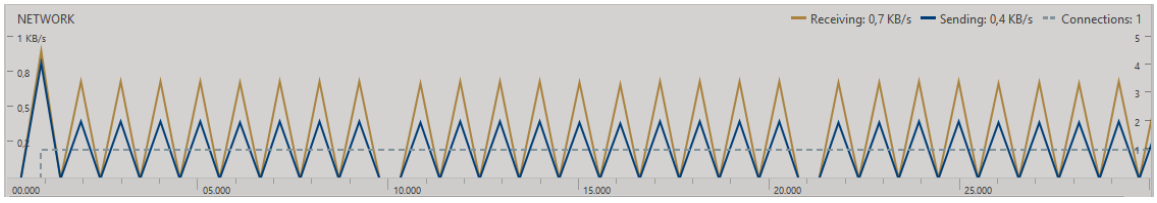
Fonte: Autor

Figura 6.2: Tráfego de dados utilizando a Configuração de módulos 2: apenas módulo de Avaliação de Critérios *QoC* no *Edge Device*.



Fonte: Autor

Figura 6.3: Tráfego de dados utilizando a Configuração de módulos 3: Avaliador e Agregador no *Edge Device*.



Fonte: Autor

6.1.2 Avaliação de uso de memória

Através da observação dos gráficos nas Figuras 6.4, 6.5 e 6.6, é possível perceber que não há diferença de comportamento entre as configurações, exceto pelas quedas em intervalos regulares no uso, atribuídas à liberação de memória ocasionada pelas diferentes estruturas de dados sendo destruídas e reconstruídas nas iterações para cada configuração.

Tabela 6.2: Comparação de redução no envio de dados pela rede utilizando as diferentes configurações de distribuição dos módulos.

Configuração	Dados Enviados Servidor	Dados Enviados Edge Device	Redução Envio Servidor (%)	Redução Envio Edge Device (%)
1. Todos módulos no Servidor Central	0,7kB/s	2,3kB/s	Base	Base
2. Apenas Avaliador no Edge Device	0,7kB/s	1,7kB/s	0	26,09
3. Avaliador e Agregador no Edge Device	0,7kB/s	0,4kB/s	0	82,61

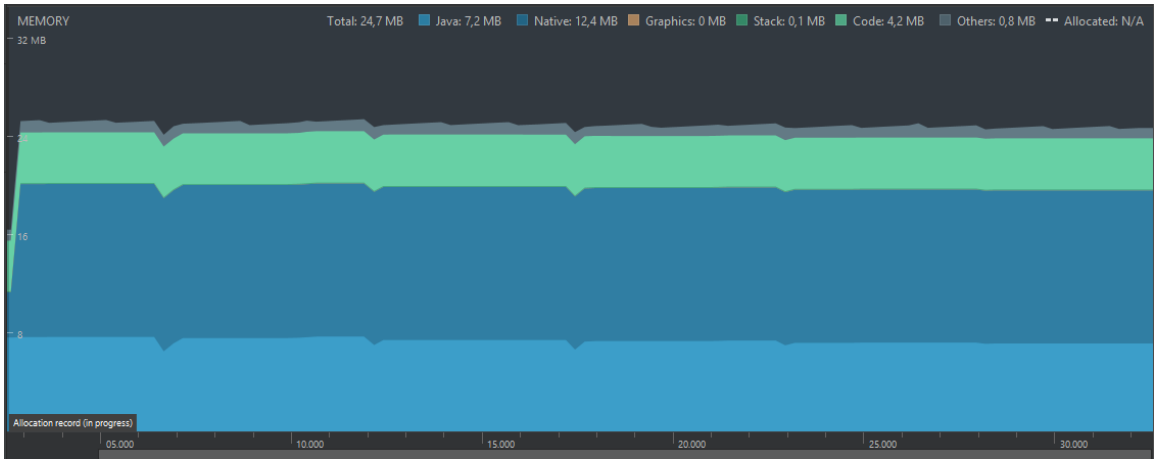
Fonte: Autor

Além disso, embora haja aumento no uso de memória[2], tanto na configuração 2 quanto na configuração 3, conforme demonstrado na Tabela 6.3, este aumento, de 8% para a configuração 2 em relação à configuração 1 e de 12% para a configuração 3 em relação à configuração 1, indica compensação em relação à economia no envio de dados pela rede.

Esta indicação é confirmada através da análise de *tradeoff*, descrita na seção 6.1.5. É preciso ressaltar também que, comparado com o total de memória fornecido pelo dispositivo, o aumento de 25*MB* para 27*MB* ou 28*MB* se evidencia como aceitável, já que, no pior caso, o percentual da memória total sendo utilizada é inferior a 2% da memória disponível.

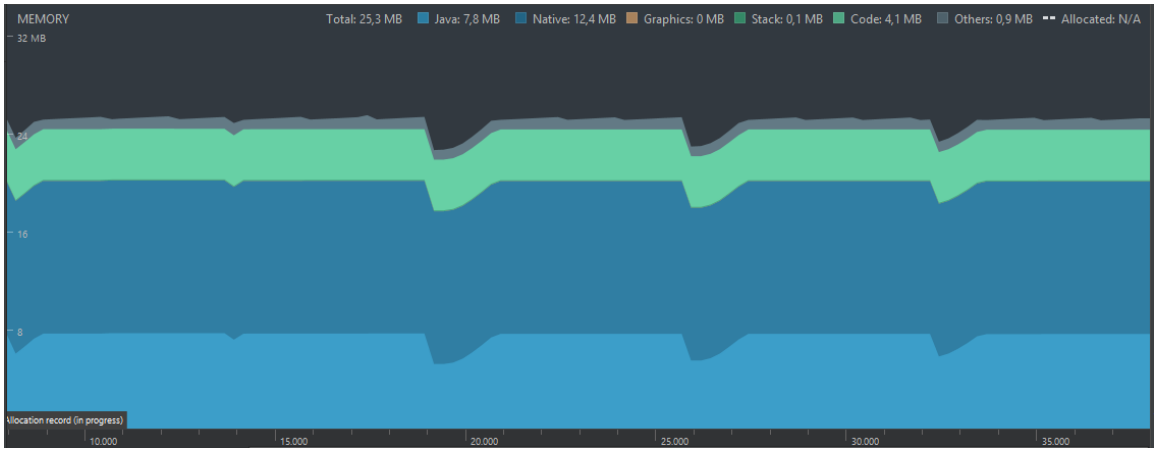
[2]O uso médio de memória e CPU é obtido através da análise do arquivo de trace gerado pelo *Android Profiler*, não incluído no texto devido ao seu tamanho

Figura 6.4: Gráfico de uso de memória *RAM* para a configuração de módulos 1: todos os módulos no Servidor Central.



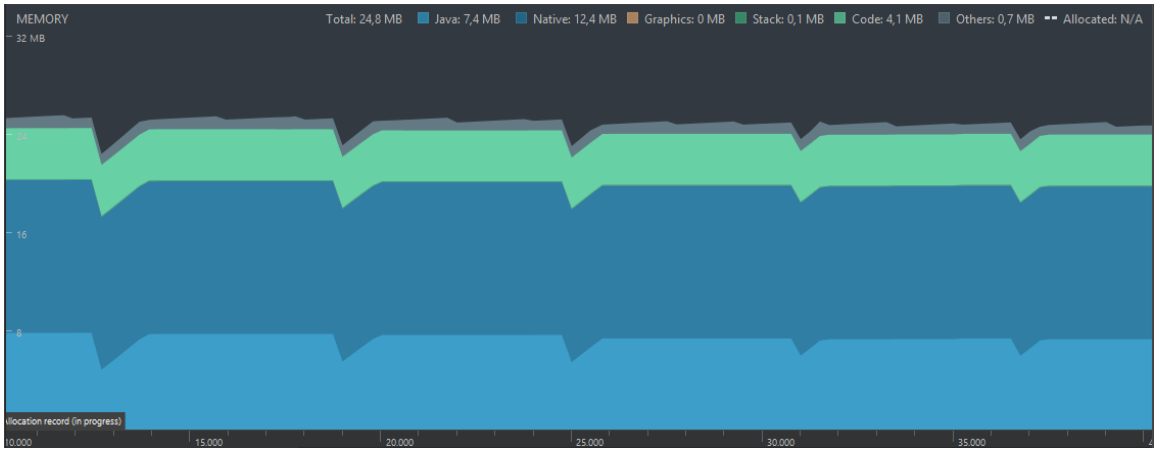
Fonte: Autor

Figura 6.5: Gráfico de uso de memória *RAM* para a configuração de módulos 2: apenas módulo de Avaliação de Critérios *QoC* no *Edge Device*.



Fonte: Autor

Figura 6.6: Gráfico de uso de memória *RAM* para a Configuração de módulos 3: Avaliador e Agregador no *Edge Device*.



Fonte: Autor

Tabela 6.3: Comparação de aumento no uso de memória utilizando as diferentes configurações de distribuição dos módulos.

Configuração	Uso de memória médio	Aumento Uso Memória (%)
1. Todos módulos no Servidor Central	25MB	Base
2. Apenas Avaliador no Edge Device	27MB	8
3. Avaliador e Agregador no Edge Device	28MB	12

Fonte: Autor

6.1.3 Avaliação de uso de *CPU*

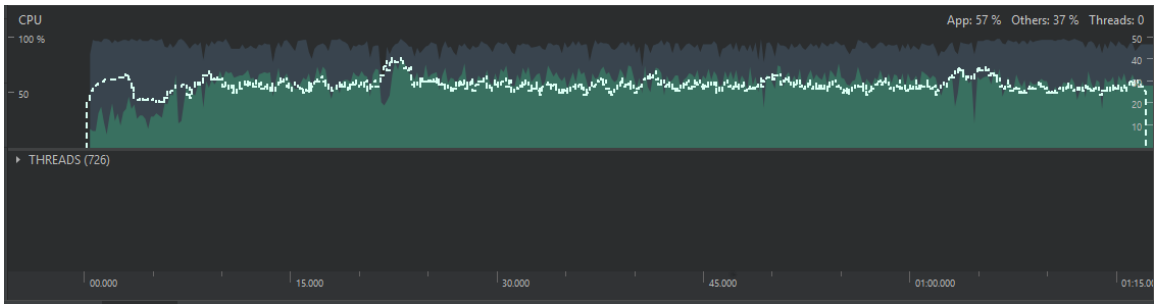
Analisando os gráficos das Figuras 6.7, 6.8 e 6.9 é possível perceber uma diferença significativa no comportamento de uso de *CPU* apresentado na configuração 1 em relação às configurações 2 e 3. Enquanto a configuração 1 apresentou um comportamento regular e estável ao longo da execução, o que se atribui ao fato de que não são aplicados métodos sobre os dados gerados pelo dispositivo, as configurações 2 e 3 apresentam um

comportamento menos regular ao longo do tempo, com grande variação e picos de uso destacados.

Cabe notar que o comportamento apresentado para a configuração 2 é similar ao apresentado para a configuração 3, o que se deve ao fato de a operação de agregação, para apenas um dispositivo e utilizando média ponderada, não ocupar muito tempo de *CPU*, sendo o comportamento nas duas configurações definido pelo módulo Avaliador.

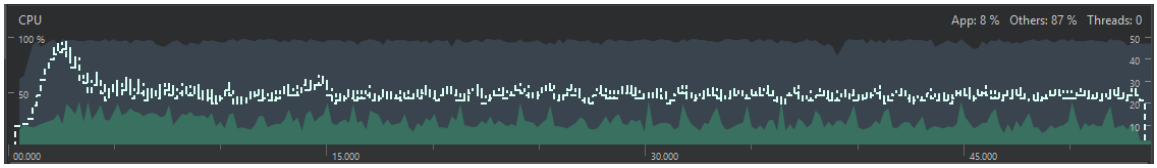
Conforme demonstrado na Tabela 6.4, o aumento percentual do uso de *CPU*, tanto para a configuração 2 quanto para a configuração 3, em relação à configuração 1, é, respectivamente, de 20,69% e 27,59%. Pode-se argumentar que o aumento do uso da *CPU* desmotiva o uso das configurações 2 e 3, sendo necessário ter maior cautela ao optar por uma configuração ou outra. No entanto, embora o aumento relativo seja alto, é preciso ressaltar que o aumento total não foi, e o *tradeoff* entre uso de *CPU* e rede ainda permite cumprir o objetivo de reduzir uso de rede sem sobrecarregar os recursos dos dispositivos. Uma análise de *tradeoff* detalhada, como a da avaliação de uso de memória, é apresentada em 6.1.5.

Figura 6.7: Gráfico de uso da *CPU* para a configuração de módulos 1: todos os módulos no Servidor Central.



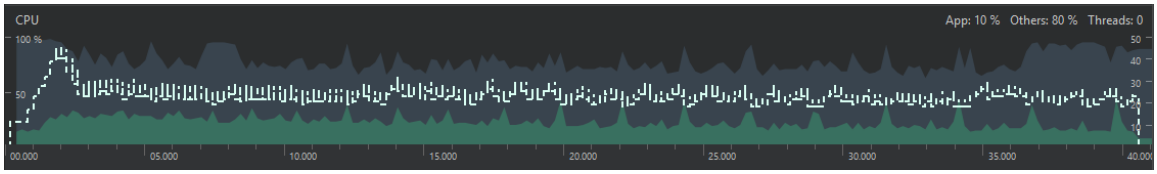
Fonte: Autor

Figura 6.8: Gráfico de uso da *CPU* para a configuração de módulos 2: apenas módulo de Avaliação de Critérios *QoC* no *Edge Device*.



Fonte: Autor

Figura 6.9: Gráfico de uso da *CPU* para a Configuração de módulos 3: Avaliador e Agregador no *Edge Device*.



Fonte: Autor

Tabela 6.4: Comparação de aumento no uso de CPU utilizando as diferentes configurações de distribuição dos módulos.

Configuração	Uso de CPU médio (%)	Aumento Uso CPU (%)
1. Todos módulos no Servidor Central	29	Base
2. Apenas Avaliador no Edge Device	35	20,69
3. Avaliador e Agregador no Edge Device	37	27,59

Fonte: Autor

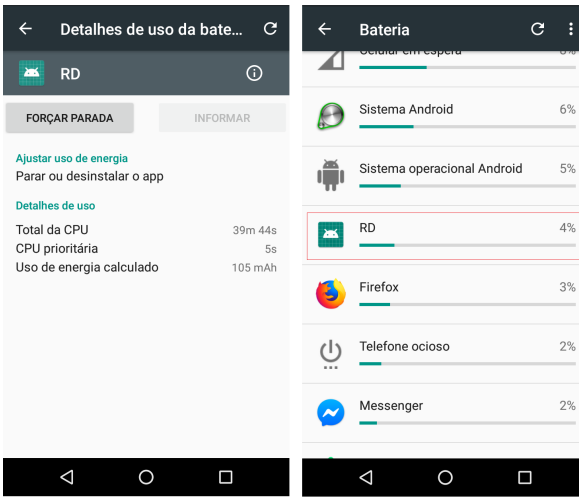
6.1.4 Avaliação de uso de Bateria

Antes de apresentar qualquer análise de uso de bateria do *Android*, é preciso ressaltar que os valores de uso de bateria obtidos pelo próprio *Android* através de seu gerenciador de bateria (Figura 6.1.4) são apenas uma estimativa de uso feita pelo sistema a partir da análise de uso de recursos como *CPU*, memória e rede, podendo divergir da realidade. No entanto, sem dispor de equipamentos apropriados para realizar medição real, esta foi a alternativa adotada para a avaliação.

A partir da análise da Tabela 6.5, é possível perceber que o aumento percentual do uso de bateria nas configurações 2 e 3, comparado ao uso na configuração 1, parece significativo. Entretanto, em termos absolutos e levando em conta as características do dispositivo, este aumento não tem impacto, representando apenas 1 ponto percentual no uso total de bateria.

Assim, da mesma forma como ocorreu com os outros aspectos avaliados, pode-se

Figura 6.10: Tela de Gerenciamento de Bateria do *Android*.



Fonte: Autor, a partir de tela do sistema *Android 6.0 Marshmallow* (GOOGLE LLC, 2018a)

Tabela 6.5: Comparação de aumento no uso de bateria utilizando as diferentes configurações de distribuição dos módulos.

Configuração	Uso de Bateria (%)	Uso de Bateria (mAh)	Aumento Uso Bateria mAh (%)
1. Todos módulos no Servidor Central	4	105	Base
2. Apenas Avaliador no Edge Device	5	131	24,78
3. Avaliador e Agregador no Edge Device	5	138	31,43

Fonte: Autor

argumentar, com base nestes resultados, tanto a favor quanto contra o uso das configurações 2 e 3. Uma análise de *tradeoff* também é apresentada para a avaliação do uso de bateria em 6.1.5.

6.1.5 Análise de *Tradeoff*

Para realizar uma análise simples de *tradeoff*, optou-se por utilizar o método de produtório apresentado por Daniels em seu artigo "*Quantitative Method for Tradeoff*

Analysis" (DANIELS; WERNER; BAHILL, 2001), cuja fórmula é apresentada a seguir:

$$f = \prod_{i=1}^{n} x_i.w_i \tag{6.1}$$

Onde *n* é o número de *figures of merit* (*valores de mérito*), ou seja, a quantidade de aspectos a serem avaliados; x_i representa o escore do valor de mérito atribuído através de fórmula matemática; e w_i representa o peso atribuído ao valor de mérito correspondente.

Este método foi escolhido pois, além de ser simples, é frequentemente usado em situações em que um valor de mérito com escore muito baixo não deve ser compensado por outros valores de mérito com escore alto (DANIELS; WERNER; BAHILL, 2001), o que é apropriado para o objetivo do estudo, já que se deseja obter economia no uso de rede sem comprometer demais os recursos computacionais do *Edge Device*.

Assim sendo, para o cálculo de *tradeoff* são considerados os seguintes valores de mérito:

1. *1- (Uso médio de memória / Total de memória)*
2. *1- (Percentual de Uso de CPU / 100)*
3. *1- (Uso de Bateria / Capacidade Total da Bateria * Saúde da Bateria)*
4. *1- Se (Uso de Rede = Max(Uso de Rede)), x = 1, Senão, x = 1 + (1- Uso de Rede/Max(Uso de Rede))*

A escolha das fórmulas de cálculo de valores de mérito foi feita de maneira a atenuar as diferenças de valores entre os valores de mérito que seriam obtidas utilizando todos os métodos com base em valores relativos, sem necessidade de atribuir pesos diferentes. Desta forma, para todos os valores de mérito é atribuído peso correspondente igual a 1.

Tabela 6.6: Valores de *tradeoff* obtidos considerando os valores para cada configuração, saúde de bateria de 97%, capacidade de bateria de 2800mAh e memória RAM de 2GB.

Config/*FoM*	1.Memória	2.CPU	3.Bateria	4.Uso de Rede	Tradeoff
Config 1	0,99	0,71	0,96	1,00	0,67
Config 2	0,99	0,65	0,95	1,27	0,77
Config 3	0,99	0,63	0,95	1,83	1,08

Fonte: Autor

Aplicando os valores de mérito com os pesos iguais, foi obtida a Tabela 6.6, a partir da qual é possível verificar que o melhor *tradeoff* é dado pela configuração 3. No entanto, é preciso ressaltar que, além de valores de mérito serem dotados de subjetividade,

e considerando os valores de mérito utilizados, os valores de *tradeoff* para os resultados obtidos dependem das especificações de hardware do dispositivo utilizado.

Desta maneira, com base na afirmação feita anteriormente na seção 6 sobre o dispositivo usado, de que este é similar aos dispositivos mais vendidos e, espera-se, mais comuns de encontrar, é possível afirmar que esta análise de *tradeoff* teria resultado similar na maioria dos dispositivos, e isto indica que o uso de *Edge Computing*, através da configuração 3, é justificado e preferível ao uso das outras configurações.

6.2 Avaliação comparativa dos métodos de agregação *Distância Euclidiana* e *Média Ponderada*

Buscando definir um possível método de agregação padrão para as aplicações, quando não for explicitado pelo usuário qual método deseja usar, e também procurando analisar a afirmação em (PERERA et al., 2013) de que seu método de *Distância Euclidiana* pode ser utilizado como padrão para aplicações que realizem seleção de sensores *IoT*, foram realizados dois experimentos.

O primeiro busca comparar o tempo de execução do método de *Distância Euclidiana* (*EDB*) com o método de *Média Ponderada* (*WAVG*), através da execução da agregação com diferentes configurações de número de critérios *QoC* e número de sensores.

O segundo busca comparar a qualidade do ranqueamento obtido por meio dos valores agregados gerados por cada método. Para tanto, foi utilizado um cenário de escolha de telefones celulares para compra, com o ranqueamento final dado pelo método *AHP*. A partir deste método, uma técnica de cálculo de distância entre ranqueamentos, o cálculo de correlação de Spearman (SPEARMAN, 1910), é aplicada.

A escolha pela comparação do método de Distância Euclidiana com o método de Média Ponderada foi feita devido ao fato de que o método de Média Ponderada é o mais encontrado na literatura, e é considerado como de uso geral para diferentes tipos de aplicação.

6.2.1 Avaliação de tempo de execução dos métodos *WAVG* e *EDB*

Para avaliar o tempo de execução destes dois métodos, foi utilizada a implementação em Java que havia sido feita, sendo adaptada para utilizar o *Java Microbenchmark*

Harness (*JMH*) (ORACLE CORPORATION, 2018a), o qual permite evitar otimizações feitas pela *JVM* que poderiam distorcer os resultados obtidos.

Para cada configuração de número de critérios e de dispositivos, foram feitas 500 execuções, com o intuito de reduzir o erro devido às diferenças entre os tempos de execução determinados pelo *JMH*. Considerando que para esta avaliação os valores e os tipos de critérios *QoC* e pesos atribuídos não causam alterações significativas no comportamento dos algoritmos, foram utilizados conjuntos de dados gerados aleatoriamente.

Para cada um dos métodos, foram utilizadas combinações de 10, 20 e 30 critérios, com 1, 1.000, 10 mil, 100 mil, 1 milhão, 10 milhões e 100 milhões de dispositivos. Cada dispositivo foi representado por uma linha em um arquivo contendo os valores de *QoC* para cada um dos critérios.

Analisando as Tabelas 6.7, 6.8 e 6.9, é possível perceber que, mesmo considerando o erro[3] para cada um dos testes realizados, o tempo total de execução do método *EDB* supera o do *WAVG* em quase todos os casos, excetuando-se a configuração com 1 dispositivo e 20 critérios. Embora seja difícil atribuir causa para o comportamento apenas neste caso, pode-se especular sobre algum tipo de anomalia na inicialização feita pela *JVM* na chamada aos métodos, que influenciaria esta configuração específica.

Desconsiderada a exceção, o que se apresenta é uma diferença nos tempos de execução entre os métodos *EDB* e *WAVG*, em que o tempo de execução de *EDB* pode chegar a mais de duas vezes e meia o de *WAVG*.

Ainda analisando os dados das tabelas, pode-se notar outra anomalia, que é um aumento desproporcional do erro para os 10 milhões de dispositivos, tanto para 20 quanto para 30 critérios. Novamente, é difícil atribuir uma causa para este comportamento sem uma análise mais profunda do funcionamento da *JVM*, mas é possível que isto se deva ao tamanho do *Heap* utilizado pela *JVM*, que é de no máximo 2*GB*.

O que aponta nesta direção é o fato de, durante os testes, terem sido gerados arquivos com os valores utilizados, e especificamente no caso de 20 critérios e 10 milhões de dispositivos, o arquivo resultante ocupou 2*GB* de espaço em disco. Desta forma, o tempo utilizado para realizar *garbage collection* pode ter influenciado o resultado final.

Para o caso dos 100 milhões de dispositivos, *garbage collection* deve ter executado

[3] A razão mínima nas tabelas é dada pelo menor tempo médio de EDB considerando o erro, dividida pelo maior tempo médio de WAVG considerando o erro. A razão máxima é dada de maneira análoga, mas com o maior tempo médio de EDB e o menor tempo médio de WAVG. A razão média é obtida a partir dos tempos médios de cada método sem o erro.

diversas vezes, e uma média dos tempos acabaria por atenuar o erro, o que explicaria o fato de o erro neste caso ser menor do que para 10 milhões de dispositivos.

Tabela 6.7: Tempos de execução para os métodos *WAVG* e *EDB*, considerando 10 critérios *QoC*.

Nº Dispositivos/Método	WAVG			EDB			Razão Tempo EDB/WAVG		
-	Tempo(µs)	Erro(µs)	% Erro	Tempo(µs)	Erro(µs)	% Erro	Mínima	Média	Máxima
1	0,1	0,001	1,000	0,105	0,001	0,952	1,02970	1,05000	1,07071
1.000	43,242	0,254	0,587	76,764	0,491	0,640	1,75356	1,77522	1,79713
10.000	438,413	4,072	0,929	812,099	6,577	0,810	1,82045	1,85236	1,88487
100.000	4027,954	23,201	0,576	8532,105	34,694	0,407	2,09753	2,11822	2,13916
1.000.000	45033,368	284,237	0,631	86428,956	397,191	0,460	1,89842	1,91922	1,94029
10.000.000	505647,085	4442,742	0,879	913008,035	7994,081	0,876	1,77422	1,80562	1,83758
100.000.000	5079279,961	42882,805	0,844	9113014,026	34666,594	0,380	1,77237	1,79415	1,81631

Fonte: Autor

Tabela 6.8: Tempos de execução para os métodos *WAVG* e *EDB*, considerando 20 critérios *QoC*.

Nº Dispositivos/Método	WAVG			EDB			Razão Tempo EDB/WAVG		
-	Tempo(µs)	Erro(µs)	% Erro	Tempo(µs)	Erro(µs)	% Erro	Mínima	Média	Máxima
1	0,111	0,002	1,802	0,1	0,001	1,000	0,87611	0,90090	0,92661
1.000	51,74	0,355	0,686	120,179	0,407	0,339	2,29911	2,32275	2,34672
10.000	538,882	4,237	0,786	1303,579	4,99	0,383	2,39098	2,41904	2,44755
100.000	5660,632	34,891	0,616	13256,348	53,104	0,401	2,31818	2,34185	2,36581
1.000.000	56742,285	353,257	0,623	132252,184	762,442	0,577	2,30298	2,33075	2,35887
10.000.000	787652,475	100865,601	12,806	1492105,95	77205,8	5,174	1,59243	1,89437	2,28501
100.000.000	7575604,237	164480,955	2,171	14113126,6	169466,29	1,201	1,80149	1,86297	1,92718

Fonte: Autor

Tabela 6.9: Tempos de execução para os métodos *WAVG* e *EDB*, considerando 30 critérios *QoC*.

Nº Dispositivos/Método	WAVG			EDB			Razão Tempo EDB/WAVG		
-	Tempo(µs)	Erro(µs)	% Erro	Tempo(µs)	Erro(µs)	% Erro	Mínima	Média	Máxima
1	0,111	0,001	0,901	0,113	0,001	0,885	1,00000	1,01802	1,03636
1.000	62,205	0,956	1,537	170,752	1,269	0,743	2,68335	2,74499	2,80855
10.000	681,183	6,219	0,913	1806,285	4,79	0,265	2,62073	2,65169	2,68322
100.000	7293,154	48,945	0,671	18455,128	43,602	0,236	2,50767	2,53047	2,55359
1.000.000	73153,253	438,261	0,599	183524,83	532,207	0,290	2,48660	2,50877	2,53121
10.000.000	1592730,066	86903,31	5,456	2648144,513	86530,261	3,268	1,52510	1,66264	1,81606
100.000.000	15735358,39	128393,265	0,816	26649162,69	124072,927	0,466	1,67206	1,69358	1,71547

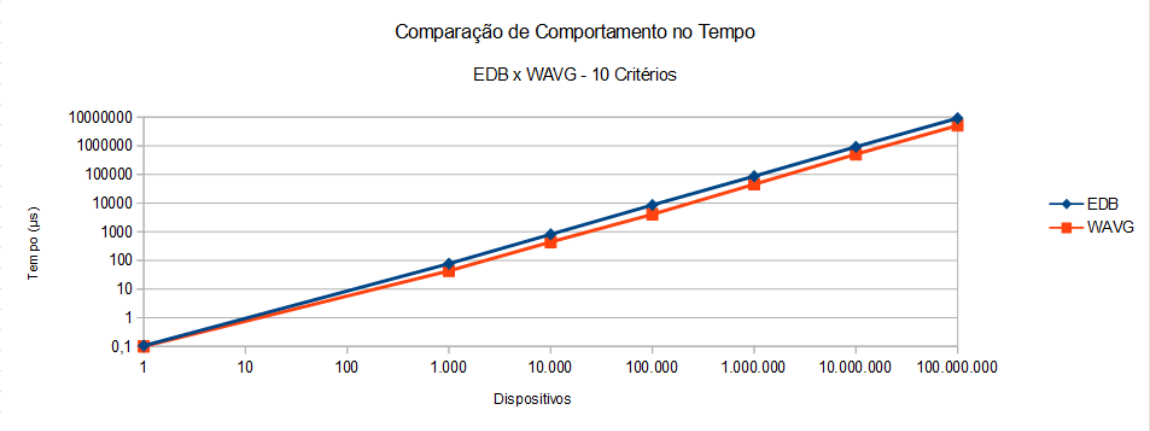
Fonte: Autor

Passando à análise dos gráficos de comportamento no tempo (Figuras 6.11, 6.12 e 6.13), percebemos que, embora as duas funções tenham comportamento similar, com crescimento quase linear do tempo de execução em relação ao número de dispositivos, há um descolamento entre as duas, facilitando perceber a diferença de crescimento nos tempos de execução para cada configuração testada.

Desta forma, considerando apenas o tempo de execução, com as diferenças significativas encontradas, conclui-se que o uso do método *WAVG* é recomendado face ao

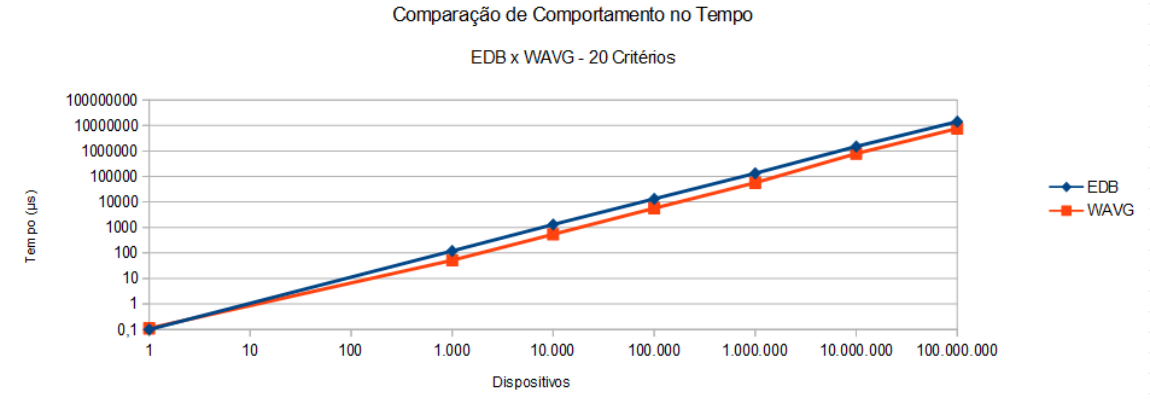
método *EDB*.

Figura 6.11: Gráfico em escala logarítmica dos tempos de execução dos métodos *EDB* e *WAVG* versus o número de dispositivos, considerando 10 critérios *QoC*.



Fonte: Autor

Figura 6.12: Gráfico em escala logarítmica dos tempos de execução dos métodos *EDB* e *WAVG* versus o número de dispositivos, considerando 20 critérios *QoC*.

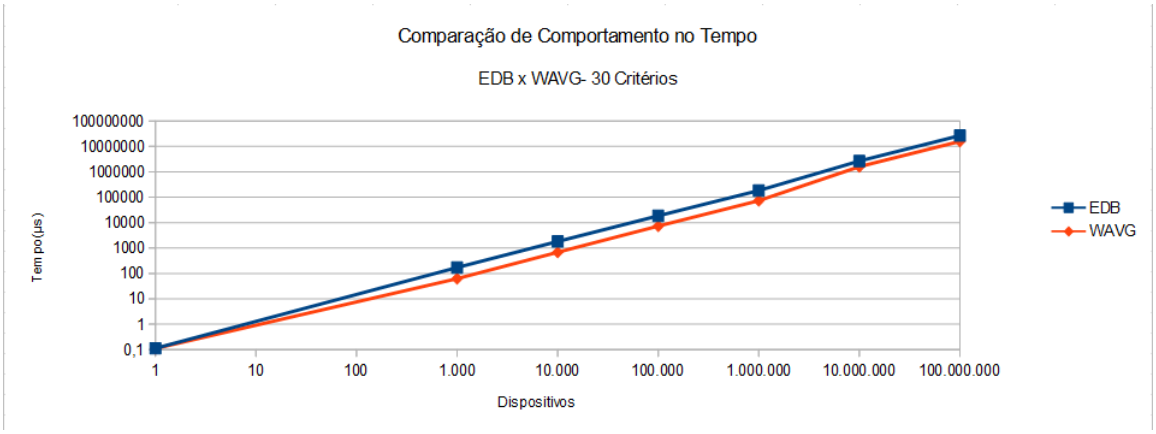


Fonte: Autor

6.3 Avaliação comparativa da qualidade do ranqueamento obtido com os métodos *EDB* e *WAVG* e o método *AHP*

Para realizar esta avaliação, foi utilizado um cenário não relacionado à seleção de sensores *IoT*, mas que envolve o uso de critérios e preferências de usuários para realizar ranqueamento e seleção de telefones celulares para compra. A escolha deste cenário para

Figura 6.13: Gráfico em escala logarítmica dos tempos de execução dos métodos *EDB* e *WAVG* versus o número de dispositivos, considerando 30 critérios *QoC*.



Fonte: Autor

a avaliação foi feita pois os dados utilizados para realizar esta escolha podem ser facilmente obtidos, e a definição dos critérios de qualidade a serem considerados é simples.

Além disso, o que se quer comparar é a qualidade dos ranqueamentos gerados pelos métodos, o que independe da aplicação para a qual se quer empregá-los. Desta forma, basta que haja critérios de qualidade, preferências do usuário e atribuição de peso para que os métodos possam ser utilizados e seus resultados avaliados.

Neste cenário hipotético, um comprador deseja saber dentre 16 modelos de telefone qual é aquele que melhor corresponde a suas necessidades, levando em conta o quanto está disposto a pagar.

Para este fim, responde a um questionário em que informa as características que deseja em termos de memória, bateria, tela etc. Além disso, para cada um dos critérios, atribui uma nota de importância, variando entre *1 – Não importa* e *10 – Muito Importante*. Uma tabela com os telefones, critérios considerados e especificações, bem como as especificações consideradas ideais pelo comprador, é apresentada na Tabela 6.10.

A Tabela 6.11 apresenta as especificações de hardware de cada um dos telefones considerados, já tratadas e normalizadas. É a partir desta tabela, juntamente com a tabela de pesos normalizados (Tabela 6.12) que são montadas matrizes de comparação par a par entre os critérios e entre os telefones, utilizando o método *AHP*. Uma ferramenta que permite montar estas matrizes e calcular o resultado final de preferência é o *PriEsT* (*Priority Estimation Tool*) (SIRAJ; MIKHAILOV; KEANE, 2015), utilizando o método de *Eigenvectors* (*autovetores*) preconizado por Saaty (SAATY, 2003). Utilizamos também os métodos *WAVG* e *EDB* com as mesmas preferências e valores, e então obtemos o

ranqueamento dado para cada um (Tabela 6.13). O método *EDB* foi aplicado utilizando duas configurações: uma com o telefone ideal indicado pelo comprador, e outra utilizando o ideal padrão (todos os valores normalizados iguais a 1).

Tabela 6.10: Modelos de telefones e especificações técnicas, juntamente com as especificações consideradas ideais pelo usuário.

Celulares	Bateria	Memória	Peso	Tela	Desempenho	Armazenamento	Câmera	Preço R$	Visual	Custo-Benefício
Moto G4 Play	17h	2GB	137g	5"	5,5	16GB	8Mp	825	0,6	?
Moto G4 Plus	22h	2GB	157g	5,5"	6,7	32GB	16Mp	1043	0,6	?
K4 Novo	11h	1GB	145g	5"	1,9	8GB	8Mp	426	0,6	?
Zuk Edge	19h	6GB	160g	5,5"	8,3	64GB	13Mp	2599	0,5	?
Zenfone Go ZB690KG	20h	1GB	270g	6,9"	3,9	8GB	8Mp	680	0,5	?
10 Evo	23h	3GB	174g	5,5"	7,5	32GB	16Mp	648	0,4	?
Galaxy J2 Prime	14h	1,4GB	160g	5"	2,4	16GB	8Mp	800	0,5	?
Redmi 4 Pro	35h	3GB	156g	5"	7,9	32GB	13Mp	950	0,5	?
Zenfone 3	22,5h	2GB	144g	5,2"	7,8	16GB	16Mp	1500	0,7	?
Shine Lite	4h	2GB	156g	5"	2,4	16GB	13Mp	700	0,4	?
Galaxy On 7 2016	20h	3GB	167g	5,5"	7,9	32GB	13Mp	1000	0,5	?
IPhone 7 Plus	14h	3GB	188g	5,5"	9,5	256GB	12Mp	3500	0,5	?
Galaxy J5 Prime	16h	2GB	143g	5"	5,3	32GB	13Mp	1000	0,5	?
Vibe K6 Plus	16h	3GB	169g	5,5"	6,6	32GB	16Mp	1300	0,5	?
Xperia XZ	10h	3GB	161g	5,2"	7,4	64GB	23Mp	4000	0,8	?
Nokia 6	18h	4GB	169g	5,5"	6,6	64GB	16Mp	1500	0,8	?
IDEAL	14h	2GB	200g	6"	7	16GB	10Mp	700	0,7	?
Importância (0-10) Critérios	8	8	5	6	9	8	4	8	2	8

Fonte: Autor, utilizando dados de (TUDO CELULAR TECNOLOGIA LTDA., 2018)

O fator de inconsistência encontrado pelo *PriEsT* neste cenário é de 0,001, o que indica que o ranqueamento gerado por ele com base nos dados fornecidos é consistente e uma decisão com base na ordem de preferência gerada é fortemente indicada. Assim, é possível afirmar que este ranqueamento corresponde ao melhor ranqueamento possível considerando os dados e preferências informadas, e podemos comparar os ranqueamentos obtidos utilizando os valores agregados fornecidos pelos métodos *WAVG* e *EDB* com o ranqueamento gerado pelo método *AHP*.

É preciso então utilizar algum método que permita comparar o grau de diferença entre os ranqueamentos gerados. Para este fim, foi escolhido o método de correlação de Spearman (SPEARMAN, 1910). Esta escolha é justificada por ser um método frequentemente utilizado na literatura, e também pelo fato de que, para o caso em que não há empate entre os integrantes do ranqueamento, é possível utilizar sua forma simplificada, apresentada na fórmula 6.2.

$$r_s = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n * (n^2 - 1)} \tag{6.2}$$

Onde d_i = $rg(X_i)$ - $rg(Y_i)$, é a diferença entre os dois rankings das duas observações; n é o número de observações.

Tabela 6.11: Normalização de critérios de seleção. Para cada critério, o telefone com melhor valor recebe o valor normalizado 1; os outros telefones recebem como valor normalizado a razão entre seu valor e o valor do melhor telefone. Cabe notar que os melhores valores não são necessariamente os valores do telefone ideal construído a partir das preferências do usuário.

AHP Celulares	Bateria	Memória	Peso	Tela	Desempenho	Armazenamento	Câmera	Preço	Visual	Custo-Benefício
Moto G4 Play	0,48571	0,33333	0,50741	0,72464	0,57895	0,06250	0,34783	0,51636	0,75000	0,63225
Moto G4 Plus	0,62857	0,33333	0,58148	0,79710	0,70526	0,12500	0,69565	0,40844	0,75000	0,60909
K4 Novo	0,31429	0,16667	0,53704	0,72464	0,20000	0,03125	0,34783	1,00000	0,75000	0,99228
Zuk Edge	0,54286	1,00000	0,59259	0,79710	0,87368	0,25000	0,56522	0,16391	0,62500	0,27779
Zenfone Go ZB690KG	0,57143	0,16667	1,00000	1,00000	0,41053	0,03125	0,34783	0,62647	0,62500	0,84040
10 Evo	0,65714	0,50000	0,64444	0,79710	0,78947	0,12500	0,69565	0,65741	0,50000	1,00000
Galaxy J2 Prime	0,40000	0,23333	0,59259	0,72464	0,25263	0,06250	0,34783	0,53250	0,62500	0,55708
Redmi 4 Pro	1,00000	0,50000	0,57778	0,72464	0,83158	0,12500	0,56522	0,44842	0,62500	0,71693
Zenfone 3	0,64286	0,33333	0,53333	0,75362	0,82105	0,06250	0,69565	0,28400	0,87500	0,43278
Shine Lite	0,11429	0,33333	0,57778	0,72464	0,25263	0,06250	0,56522	0,60857	0,50000	0,61541
Galaxy On 7 2016	0,57143	0,50000	0,61852	0,79710	0,83158	0,12500	0,56522	0,42600	0,62500	0,63768
IPhone 7 Plus	0,40000	0,50000	0,69630	0,79710	1,00000	1,00000	0,52174	0,12171	0,62500	0,21783
Galaxy J5 Prime	0,45714	0,33333	0,52963	0,72464	0,55789	0,12500	0,56522	0,42600	0,62500	0,53915
Vibe K6 Plus	0,45714	0,50000	0,62593	0,79710	0,69474	0,12500	0,69565	0,32769	0,62500	0,47853
Xperia XZ	0,28571	0,50000	0,59630	0,75362	0,77895	0,25000	1,00000	0,10650	1,00000	0,17768
Nokia 6	0,51429	0,66667	0,62593	0,79710	0,69474	0,25000	0,69565	0,28400	1,00000	0,48113
IDEAL	0,40000	0,33333	0,74074	0,86957	0,73684	0,06250	0,43478	0,60857	0,87500	0,8753768396

Fonte: Autor

Tabela 6.12: Normalização dos pesos atribuídos pelo usuário para cada critério.

Importância Critérios (0-10)	8	8	5	6	9	8	4	8	2	8
Peso normalizado	0,88889	0,88889	0,55556	0,66667	1,00000	0,88889	0,44444	0,88889	0,22222	0,88889

Fonte: Autor

Ao analisarmos os resultados obtidos pelo método de Spearman (Tabela 6.14), tanto o método *WAVG* quanto o *EDB* apresentam um nível semelhante de qualidade no ranqueamento gerado pelos valores retornados por eles, com o resultado de *WAVG* ainda sendo o mais fortemente correlacionado ao obtido pelo método *AHP*.

É possível perceber, também, que o menor nível de correlação foi apresentado pela configuração de *EDB* considerando as preferências atribuídas pelo usuário como sendo a de um telefone ideal, o que se explica pelo ranqueamento gerado com base nestas preferências não buscar a melhor opção, e sim aquela que mais se aproxima do que o usuário especificou. Este tipo de atribuição deve ser feita para casos específicos, não sendo recomendado para um método padrão de seleção.

Também considerando o resultado obtido na avaliação de tempo de execução, conclui-se que é melhor utilizar o método de Média Ponderada *WAVG* como padrão. No entanto, não é possível descartar totalmente o método *EDB* utilizando a configuração de sensor ideal como possível padrão a partir dos experimentos realizados, sendo assim mantida a afirmação feita por Perera de que *EDB* pode ser utilizado como método padrão para

Tabela 6.13: Ranqueamento resultante para cada um dos métodos considerados.

	AHP	WAVG	EDB (Usuário)	EDB (Padrão)
Iphone 7 Plus	1	3	1	2
10Evo	2	1	2	1
Redmi 4 Pro	3	2	3	3
Zuk Edge	4	4	4	7
Nokia 6	5	6	8	6
Galaxy On 7 2016	6	5	7	10
Moto G4 Plus	7	8	9	11
Zenfone Go ZB	8	7	5	5
Vibe K6 Plus	9	9	12	12
Xperia XZ	10	13	10	4
Zenfone 3	11	10	11	9
K4 Novo	12	11	6	8
Galaxy J5 Prime	13	14	14	14
Moto G4 Play	14	12	13	13
Shine Lite	15	15	15	15
Galaxy J2 Prime	16	16	16	16

Fonte: Autor

Tabela 6.14: Índice de correlação de Spearman em relação ao ranqueamento gerado pelo método *AHP* com *eigenvectors*.

Método	r_s
WAVG	0,96176
EDB (preferências do usuário)	0,89706
EDB (padrão)	0,82353

Fonte: Autor

seleção de sensores *IoT*.

6.4 Avaliação dos Perfis *QoC*

Para avaliar o quanto o uso de Perfis *QoC* pode ser útil e facilitar aos usuários a seleção de critérios, métodos e atribuição de pesos, optou-se por realizar análise qualitativa, apresentando o sistema aos usuários, pedindo que fizessem algumas definições e então apresentando questionários.

Inicialmente, foram realizadas entrevistas com 6 especialistas de domínio de aplicação, pesquisadores nas áreas de *IoT* e *Big Data*, apresentando algumas configurações

de perfis e pedindo aos usuários que avaliassem se a interface facilitava na seleção das definições de importância das dimensões *QoC* apresentadas, utilizando para isso algumas configurações iniciais de perfis e pedindo aos usuários que tentassem ajustar o cenário inicial a uma determinada configuração, que podia ou não ser diferente da apresentada inicialmente.

Esta abordagem serviu para direcionar a pesquisa para a elaboração de testes incluindo questionários, e apresentando todas as configurações de perfil criadas para cada um dos especialistas, em vez de apresentar configurações selecionadas para cada um como havia sido feito neste teste preliminar.

A partir deste esboço foram realizados testes com especialistas de domínio e apresentado um questionário com 9 questões utilizando opções valoradas entre "1 – Discordo Fortemente" e "5 – Concordo Fortemente", usando a escala de Likert (LIKERT, 1932), e tendo as perguntas baseadas no exposto em "*Convenience and TAM in a ubiquitous computing environment: The case of wireless LAN*" (YOON; KIM, 2007), que define a avaliação de uma tecnologia em termos de *facilidade de uso* (*ease-of-use*) e *utilidade* (*usefulness*), sendo 5 perguntas em relação à facilidade de uso e 4 em relação à utilidade.

Durante os testes, diversos cenários foram apresentados utilizando número variável de critérios, com orientações dadas sobre o que deveria ser feito (ver apêndice A). Os questionários foram apresentados antes de iniciar o teste com cada um dos especialistas, com as questões podendo ser respondidas a qualquer momento e em qualquer ordem desejada.

Desta segunda iteração fizeram parte 8 especialistas. Nenhum integrava o grupo original entrevistado, para evitar avaliação tendenciosa. Cada especialista recebeu 10 conjuntos (Tabela 6.15) de objetivos para os critérios de *QoC*, com uma sugestão de perfil respectiva para cada conjunto. A partir da sugestão, deveria modificar as configurações para chegar ao objetivo, e responder os questionários.

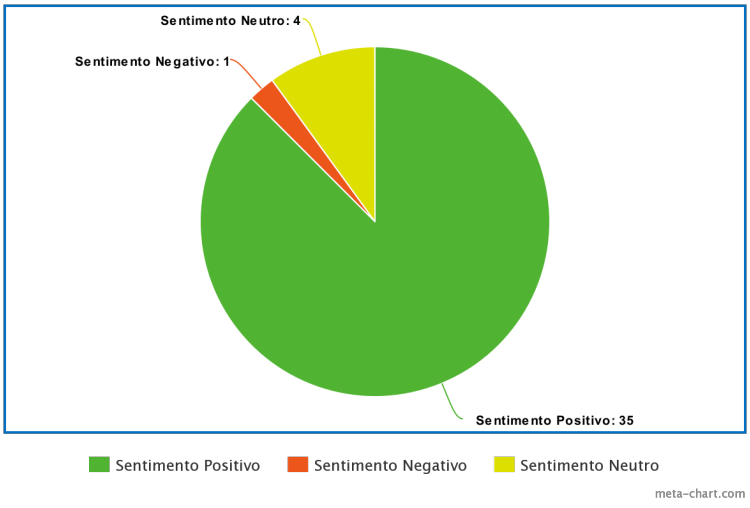
A Tabela 6.16 mostra um resumo com as questões obrigatórias e as respostas dadas pelos especialistas. Os gráficos das Figuras 6.14 e 6.15 mostram a distribuição de sentimento, com base nas respostas dos usuários. É possível perceber que o grau de aceitação da interface foi maior do que o do sistema de Perfis *QoC* em si, e o nível de indiferença foi menor, indicando a necessidade de empenhar mais esforços em melhorar o sistema de Perfis.

Tabela 6.15: Resumo mostrando as configurações iniciais apresentadas pela Interface e o objetivo final de configuração.

	Tipo	Perfil 1	Perfil 2	Perfil 3	Perfil 4	Perfil 5	Perfil 6	Perfil 7	Perfil 8	Perfil 9	Perfil 10
Custo	Sugestão	10	7	7	7	7	7	9	2	3	9
	Objetivo	5	7	6	6	7	7	7	4	3	7
Precisão	Sugestão	10	8	5	5	5	5	4	6	4	4
	Objetivo	7	8	7	7	7	7	7	8	6	7
Acurácia	Sugestão	10	8	4	4	5	5	3	8	7	3
	Objetivo	8	8	5	5	5	5	5	6	9	5
Atualidade	Sugestão	-	4	7	7	4	4	8	5	10	8
	Objetivo	-	4	8	8	8	8	8	7	9	8
Resiliência	Sugestão	-	3	10	5	0	10	4	3	6	5
	Objetivo	-	3	10	7	10	0	6	3	10	4
Eficiência	Sugestão	-	6	9	9	9	9	3	4	3	3
	Objetivo	-	6	7	7	9	9	6	7	10	6
Completude	Sugestão	-	9	9	9	7	7	7	9	1	7
	Objetivo	-	9	9	9	9	9	9	9	10	9
Tempo de Resposta	Sugestão	-			6	6	6	6	4	10	6
	Objetivo	-			10	10	10	6	7	3	6
Latência	Sugestão	-			10	10	10	5	6	7	5
	Objetivo	-			8	8	8	8	6	1	8
Processamento	Sugestão	-					6	3	5	2	3
	Objetivo	-					7	7	7	6	7
Memória	Sugestão	-					4	7	6	7	7
	Objetivo	-					7	4	8	9	4
Umidade	Sugestão	-								3	8
	Objetivo	-								6	6
Distância	Sugestão	0m	1000m	400m	1200m	100000m	5000m	100m	100m	50m	10m
	Objetivo	0m	1000m	500m	400m	85000m	3250m	110m	100m	2500m	8m
Restrição Tipo	Sugestão	Sim	Sim	Não	Não	Sim	Sim	Sim	Sim	Não	Sim
	Objetivo	Sim	Sim	Sim	Não	Não	Sim	Sim	Não	Sim	Sim

Fonte: Autor

Figura 6.14: Distribuição de sentimento dos usuários em relação à Interface.



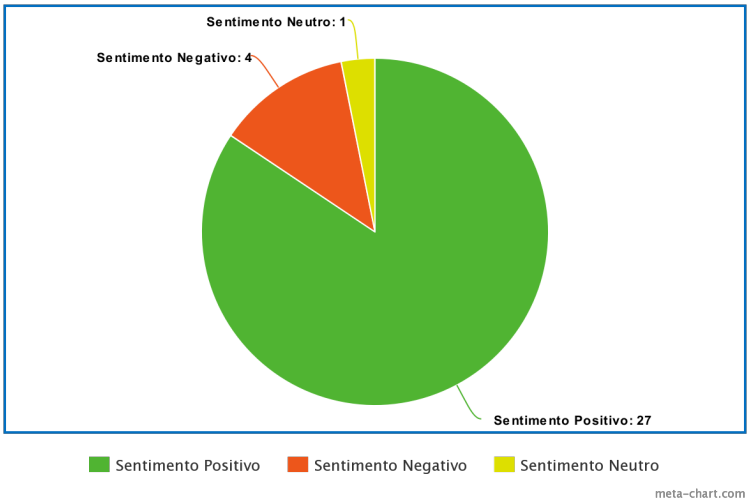
Fonte: Autor, utilizando a ferramenta Meta-Chart (META-CHART, 2018)

Tabela 6.16: Resumo com perguntas e respostas do questionário obrigatório sobre perfis *QoC*.

		Respostas				
Questão	Formato	Concordo Fortemente	Concordo	Nem Concordo nem Discordo	Discordo	Discordo Fortemente
1.É fácil se orientar na interface do sistema de perfis	Número	4	3	1	0	0
	Percentual	50,00%	37,50%	12,50%	0,00%	0,00%
2. O posicionamento dos objetos na interface do sistema de perfis segue uma linha lógica de acordo com o processo modelado	Número	3	4	1	0	0
	Percentual	37,50%	50,00%	12,50%	0,00%	0,00%
3. A interface do sistema de perfis tem um bom tempo de resposta aos comandos	Número	4	4	0	0	0
	Percentual	50,00%	50,00%	0,00%	0,00%	0,00%
4. Consigo realizar as alterações que desejo com facilidade na interface do sistema de perfis	Número	1	5	1	0	1
	Percentual	12,50%	62,50%	12,50%	0,00%	12,50%
5. A interface do sistema de perfis apresenta descrições corretas para suas funções e elementos, que se comportam de acordo com estas descrições	Número	1	6	1	0	0
	Percentual	12,50%	75,00%	12,50%	0,00%	0,00%
6. As funcionalidades do sistema de perfis auxiliam na tarefa de definir critérios de qualidade de contexto para escolha de sensores	Número	1	6	0	1	0
	Percentual	12,50%	75,00%	0,00%	12,50%	0,00%
7. O sistema de perfis pode auxiliar a realizar o meu trabalho	Número	1	6	0	0	1
	Percentual	12,50%	75,00%	12,50%	0,00%	12,50%
8. Os valores retornados pelo sistema de perfis podem ser usados para construir aplicações típicas com seleção de sensores	Número	4	2	1	0	1
	Percentual	50,00%	25,00%	12,50%	0,00%	12,50%
9. Usar o sistema de perfis para definir critérios e seus pesos me fará poupar tempo ao criar aplicações onde tais definições são necessárias	Número	2	5	0	1	0
	Percentual	25,00%	62,50%	12,50%	12,50%	0,00%

Fonte: Autor

Figura 6.15: Distribuição de sentimento dos usuários em relação ao sistema de Perfis *QoC*.



Fonte: Autor

6.5 Estudos de Caso

Para avaliar o potencial da arquitetura em casos práticos, elaborou-se dois estudos de caso com situações distintas, que foram então simuladas. Embora tenham sido testados diferentes simuladores de uso de contexto e *IoT*, como o *Siafu* (MARTIN; NURMI, 2006), o *Bevywise IoT* (BEVYWISE INC., 2018), e o *One Simulator* (KERÄNEN; OTT; KÄRKKÄINEN, 2009), entre outros, optou-se por utilizar o *Cupcarbon* (CUPCARBON, 2018), por apresentar facilidade de uso e suporte à simulação de vários dispositivos.

Estes estudos de caso, bem como as observações e os resultados obtidos são apresentados nas próximas seções.

6.5.1 Estudo de Caso 1: Redes Oportunistas

Para este estudo de caso, suponhamos que um usuário de *smartphone*, chamado A, deseje enviar uma mensagem para sua mãe, B, que mora do outro lado de uma cidade pequena. Nesta cidade, frequentemente o acesso à Internet é interrompido devido a problemas de infraestrutura da rede. Assim, várias pessoas utilizam um aplicativo que permite transmitir mensagens, de forma segura, utilizando a conexão *Wi-Fi* de outros usuários que tenham o mesmo aplicativo.

Desta forma, o usuário A compõe sua mensagem e envia pelo aplicativo. Este aplicativo então utiliza os dados contextuais de A e dos usuários mais próximos para definir qual usuário terá maior probabilidade de levar a mensagem adiante, num modelo conhecido como *Delay Tolerant Network* ou *DTN*, (*rede tolerante a atraso* ou *rede oportunista*) [4]. Para este fim, o aplicativo utiliza uma implementação da Arquitetura RD, que avalia os critérios de *QoC* para cada um dos dispositivos, agrega, ranqueia e envia a mensagem para os mais bem avaliados.

Os critérios utilizados pelo aplicativo para gerar o ranqueamento são: *confiabilidade*, *taxa de entrega* (quantas vezes a mensagem foi recebida pelo destinatário ao ser enviada pelo transmissor ranqueado), o *nível de bateria*, a *saúde da bateria*, a *intensidade do sinal* e a *distância* a partir do transmissor.

A tabela com os últimos valores conhecidos pelo telefone de A (Tabela 6.17) mostra que apenas um telefone (*Sensor s02*) além dele mesmo (*Sensor s01*) tem condições de

[4] Para uma visão aprofundada sobre *DTNs*, recomenda-se a leitura do artigo "*Delay-tolerant networking: an approach to interplanetary internet*" (BURLEIGH et al., 2003)

receber a mensagem, pois é o único que tem 10 como valor na distância (a função de distância utilizada pela aplicação atribui 0 quando o transmissor não está no raio de alcance do *Wi-Fi* e 10 quando está).

No entanto, avaliações e agregações (Tabela 6.18) são feitas para todos os sensores, pois a mensagem pode ser enviada para um dos sensores que não estão atualmente no raio de alcance, caso o tempo de *timeout* (definido no aplicativo de A como sendo 10 minutos) seja atingido e um deles passe a estar neste raio.

Antes que sejam agregados, os valores para cada critério são normalizados. A agregação se dá utilizando Média Ponderada, com pesos iguais para todos os critérios.

Figura 6.16: Mapa de modelagem do Estudo de Caso 1, de redes oportunistas.

Fonte: Autor

O telefone de A então gera o ranqueamento ordenado (Tabela 6.19), a partir do qual envia a mensagem para o telefone no topo da lista (excluindo ele próprio) e armazena este ranqueamento para uso futuro.

Em cada telefone o aplicativo armazena um ranqueamento utilizado para passar a mensagem adiante. Ao final do processo, A recebeu confirmação de recebimento originada do telefone de B antes do prazo de *timeout*. Desta forma, o aplicativo realizou a entrega de A até B, indicando ser viável o uso da RD em redes oportunistas.

Tabela 6.17: Valores conhecidos pelo telefone de A (*Sensor s01*) sobre os outros telefones que utilizam o aplicativo.

Sensor	Confiabilidade	Taxa de Entrega	Nível de Bateria	Saúde da Bateria	Intensidade do Sinal	Distância
s01	7	10	7	9	9	10
s02	8	8	5	8	9	10
s03	7	8	8	10	9	0
s04	8	1	7	8	8	0
s05	9	7	9	9	8	0
s06	7	7	8	9	8	0
s07	9	?	3	9	8	0
s08	7	8	7	8	8	0
s09	6	6	7	9	8	0
s10	8	8	4	9	8	0
s11	7	4	9	6	8	0
s12	8	5	10	8	7	0
s13	9	9	6	7	7	0
s14	7	6	5	9	7	0
s15	7	7	6	6	8	0
s16	9	8	7	6	7	0
s17	8	?	8	7	8	0
s18	7	7	9	8	8	0
s19	8	7	4	9	8	0
s20	4	6	9	8	9	0
s21	7	8	3	9	9	0
s22	8	10	6	8	9	0
s23	7	6	2	7	9	0

Fonte: Autor

6.5.2 Estudo de Caso 2: Seleção de Sensores Meteorológicos

Neste estudo de caso, um cientista deseja criar um modelo de predição com base em medições do nível de umidade que permita aumentar a eficiência dos sistemas agrícolas de irrigação. Para tanto, deseja analisar as condições de diferentes plantações em uma determinada região, utilizando dados obtidos a partir de sensores que satisfaçam alguns critérios de acurácia e precisão das medições, bem como tenham o maior número possível de informações (completude de informação), e que tenham boa autonomia de bateria, no caso de sensores que dependam deste tipo de fonte de energia.

Desta forma, o cientista utiliza um sistema baseado na arquitetura RD em sua estação de base para informar os critérios de seu interesse, a importância que atribui a cada um e o raio de distância onde se encontram os sensores cuja qualidade deseja conhecer.

Neste cenário, a avaliação e agregação dos critérios de *QoC* é realizada nos próprios sensores, com o resto das operações previstas pela arquitetura RD sendo feito na estação de base.

Para simular 200 sensores e poder realizar a análise dos resultados obtidos de

Tabela 6.18: Valores normalizados e ponderados por critério, juntamente com a nota agregada atribuída a cada telefone/sensor pelo telefone de A.

Sensor	Confiabilidade	Taxa de Entrega	Nível de Bateria	Saúde da Bateria	Intensidade do Sinal	Distância	Nota Agregada
s01	0,11667	0,16667	0,11667	0,15000	0,15000	0,16667	0,86667
s02	0,13333	0,13333	0,08333	0,13333	0,15000	0,16667	0,80000
s03	0,11667	0,13333	0,13333	0,16667	0,15000	0,00000	0,70000
s04	0,13333	0,01667	0,11667	0,13333	0,13333	0,00000	0,53333
s05	0,15000	0,11667	0,15000	0,15000	0,13333	0,00000	0,70000
s06	0,11667	0,11667	0,13333	0,15000	0,13333	0,00000	0,65000
s07	0,15000	0,01667	0,05000	0,15000	0,13333	0,00000	0,50000
s08	0,11667	0,13333	0,11667	0,13333	0,13333	0,00000	0,63333
s09	0,10000	0,10000	0,11667	0,15000	0,13333	0,00000	0,60000
s10	0,13333	0,13333	0,06667	0,15000	0,13333	0,00000	0,61667
s11	0,11667	0,06667	0,15000	0,10000	0,13333	0,00000	0,56667
s12	0,13333	0,08333	0,16667	0,13333	0,11667	0,00000	0,63333
s13	0,15000	0,15000	0,10000	0,11667	0,11667	0,00000	0,63333
s14	0,11667	0,10000	0,08333	0,15000	0,11667	0,00000	0,56667
s15	0,11667	0,11667	0,10000	0,10000	0,13333	0,00000	0,56667
s16	0,15000	0,13333	0,11667	0,10000	0,11667	0,00000	0,61667
s17	0,13333	0,01667	0,13333	0,11667	0,13333	0,00000	0,53333
s18	0,11667	0,11667	0,15000	0,13333	0,13333	0,00000	0,65000
s19	0,13333	0,11667	0,06667	0,15000	0,13333	0,00000	0,60000
s20	0,06667	0,10000	0,15000	0,13333	0,15000	0,00000	0,60000
s21	0,11667	0,13333	0,05000	0,15000	0,15000	0,00000	0,60000
s22	0,13333	0,16667	0,10000	0,13333	0,15000	0,00000	0,68333
s23	0,11667	0,10000	0,03333	0,11667	0,15000	0,00000	0,51667

Fonte: Autor

maneira simples, foram criados 10 tipos de sensores distintos com atribuição de notas para cada um dos critérios utilizados (Tabela 6.20), os quais foram replicados com as mesmas notas atribuídas aos critérios.

Assim, o cientista declara desejar informações dos 70 melhores sensores num raio de 5km a partir da estação de base. Além disso, atribui os pesos de *1 – Não importa* a *10 – Muito importante* para cada um dos critérios (Tabela 6.21) considerados relevantes para sua pesquisa.

A partir das informações enviadas pelo cientista, cada um dos sensores calculou seus valores para os critérios *QoC* escolhidos (Tabela 6.22), normalizou-os e agregou utilizando o método de Média Ponderada (Tabela 6.23). Ao final, uma lista com os 70 melhores sensores (Tabela 6.24), ordenados por nota, foi gerada e exibida ao cientista, que agora pode utilizá-la para desenvolver seu trabalho. Isto indica, da mesma forma como ocorreu no Estudo de Caso 1, que o uso da arquitetura RD é viável para aplicações que utilizem sensores em ambientes agrícolas.

Tabela 6.19: Ranqueamento armazenado pelo telefone de A.

Sensor	Nota	Ranking
s01	0,86667	1
s02	0,80000	2
s03	0,70000	3
s05	0,70000	4
s22	0,68333	5
s06	0,65000	6
s18	0,65000	7
s08	0,63333	8
s12	0,63333	9
s13	0,63333	10
s10	0,61667	11
s16	0,61667	12
s09	0,60000	13
s19	0,60000	14
s20	0,60000	15
s21	0,60000	16
s11	0,56667	17
s14	0,56667	18
s15	0,56667	19
s04	0,53333	20
s17	0,53333	21
s23	0,51667	22
s07	0,50000	23

Fonte: Autor

6.6 Considerações Finais

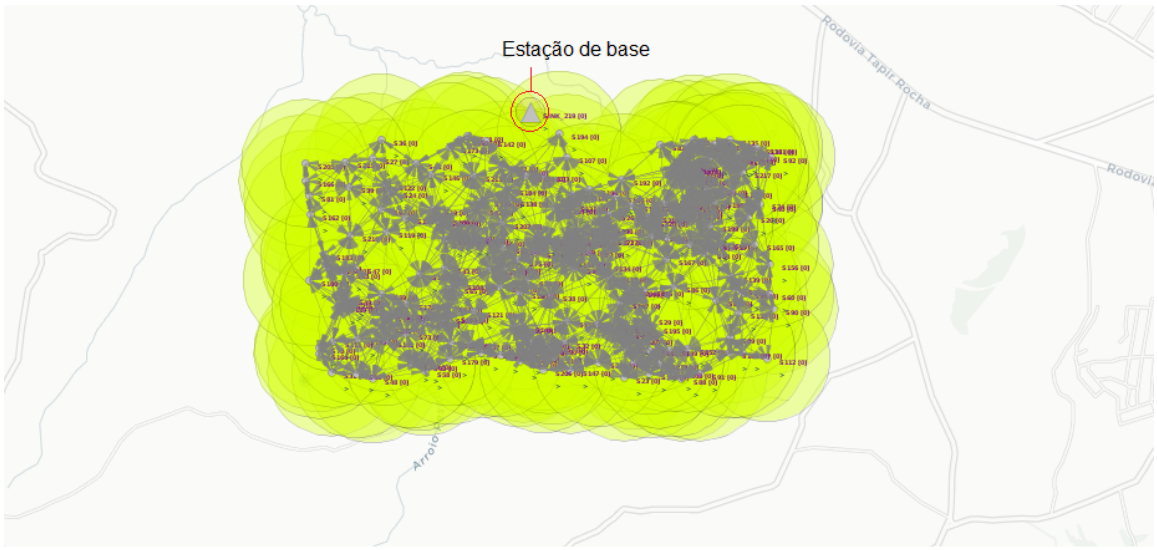
Nesta seção foram apresentadas diversas avaliações realizadas no intuito de testar diferentes aspectos da arquitetura RD e de sua implementação.

Para demonstrar a vantagem do uso de *Edge Computing* ao realizar os processos de avaliação e agregação de *QoC*, foi realizada análise comparativa do uso de recursos computacionais para cada configuração de distribuição de módulos entre um Servidor Central e um *Edge Device*, com respectiva análise de *tradeoff*, indicando que a configuração mais apropriada é aquela que realiza tanto avaliação de critérios *QoC* quanto agregação no *Edge Device*.

Na busca por um método de agregação que pudesse ser utilizado como padrão, foram avaliados comparativamente, em relação a tempo de execução e qualidade do ranqueamento gerado através das notas dadas, os métodos de *Média Ponderada*, ou *WAVG*, e *Distância Euclidiana*, ou *EDB*. Através da execução dos métodos com diferentes configurações de número de dispositivos e critérios de *QoC*, ficou evidente que o uso de Média Ponderada tem vantagem em relação ao tempo de execução.

Na comparação de qualidade de ranqueamento, tendo como base ranqueamento obtido utilizando o método *AHP* com *eigenvectors* para calcular a correlação de Spear-

Figura 6.17: Mapa de modelagem do Estudo de Caso 2.



Fonte: Autor

Tabela 6.20: Notas para cada critério *QoC* adotado para os 10 tipos de sensores definidos e replicados para o Estudo de Caso 2.

Sensor	Acurácia	Precisão	Completude	Autonomia de Bateria
s01	9	10	9	9
s02	8	8	10	10
s03	5	7	5	10
s04	9	7	8	6
s05	9	9	7	8
s06	10	10	8	4
s07	8	8	8	8
s08	7	7	6	9
s09	7	8	7	8
s10	7	9	10	8

Fonte: Autor

man, descobriu-se similaridade entre os dois métodos, não sendo estabelecida uma preferência para este critério. Devido a isso, embora a preferência seja por adotar Média Ponderada como método padrão para agregação, não é possível descartar o método de Distância Euclidiana como possível padrão, sendo mantida a afirmação de Perera sobre a possibilidade de uso de Distância Euclidiana como padrão.

Em relação ao uso de Perfis *QoC*, foi feito um estudo com especialistas de domínio buscando determinar o quanto as sugestões podem ser úteis para aplicações que utilizem seleção de sensores. Através de questionário, foi possível determinar que a interface para definir os critérios *QoC* e sua importância, a fim de realizar a seleção dos sensores, é adequada, sendo necessários, no entanto, mais esforços para melhorar o sistema de Perfis.

Tabela 6.21: Pesos atribuídos para cada um dos critérios no Estudo de Caso 2.

Acurácia	Precisão	Completude	Autonomia de Bateria
8	8	10	5

Fonte: Autor

Tabela 6.22: Notas para cada critério *QoC* após normalização e ponderação, bem como nota agregada obtida pelo método *WAVG*, para o Estudo de Caso 2.

Sensor	Acurácia	Precisão	Completude	Autonomia de Bateria	Nota Agregada
s01	0,23226	0,25806	0,29032	0,14516	0,92581
s02	0,20645	0,20645	0,32258	0,16129	0,89677
s03	0,12903	0,18065	0,16129	0,16129	0,63226
s04	0,23226	0,18065	0,25806	0,09677	0,76774
s05	0,23226	0,23226	0,22581	0,12903	0,81935
s06	0,25806	0,25806	0,25806	0,06452	0,83871
s07	0,20645	0,20645	0,25806	0,12903	0,80000
s08	0,18065	0,18065	0,19355	0,14516	0,70000
s09	0,18065	0,20645	0,22581	0,12903	0,74194
s10	0,18065	0,23226	0,32258	0,12903	0,86452

Fonte: Autor

Além disso, ainda quanto ao sistema de Perfis *QoC*, é possível concluir que pode ser útil, e ter sugestões iniciais de acordo com o tipo de aplicação é uma característica desejável. Entretanto, é necessário realizar um estudo mais amplo, com uma amostragem da população grande o suficiente para garantir relevância estatística e utilizando exemplos de aplicações reais, a fim de afirmar que o uso de Perfis *QoC* é capaz de prover sugestões de configurações que atendam às expectativas e necessidades dos usuários.

Por fim, foram feitos dois estudos de caso buscando demonstrar a possibilidade de uso da arquitetura RD em aplicações reais, utilizando para isso o simulador *CupCarbon U-One* para contexto e *IoT*.

O Estudo de Caso 1 indicou a viabilidade do uso em redes oportunistas, realizando a entrega de uma mensagem de texto na rede simulada, enquanto o Estudo de Caso 2 indicou a viabilidade para aplicações que utilizem sensores no meio agrícola, realizando a seleção dos melhores sensores, de acordo com a avaliação, agregação e ranqueamento pelos módulos da arquitetura RD.

No entanto, seria ideal, para validar os resultados obtidos, utilizar a arquitetura em um ambiente real onde haja interações com elementos externos e demais fatores que não foram considerados na simulação.

Tabela 6.23: Ranqueamento para cada tipo de sensor no Estudo de Caso 2.

sensor	nota	ranking
s01	0,92581	1
s02	0,89677	2
s10	0,86452	3
s06	0,83871	4
s05	0,81935	5
s07	0,80000	6
s04	0,76774	7
s09	0,74194	8
s08	0,70000	9
s03	0,63226	10

Fonte: Autor

Tabela 6.24: Ranqueamento final dos 70 sensores selecionados no Estudo de Caso 2.

Sensor	Posição	Sensor	Posição	Sensor	Posição	Sensor	Posição
s01	1	s02	21	s10	41	s06	61
s11	2	s12	22	s20	42	s16	62
s21	3	s22	23	s30	43	s26	63
s31	4	s32	24	s40	44	s36	64
s41	5	s42	25	s50	45	s46	65
s51	6	s52	26	s60	46	s56	66
s61	7	s62	27	s70	47	s66	67
s71	8	s72	28	s80	48	s76	68
s81	9	s82	29	s90	49	s86	69
s91	10	s92	30	s100	50	s96	70
s101	11	s102	31	s110	51		
s111	12	s112	32	s120	52		
s121	13	s122	33	s130	53		
s131	14	s132	34	s140	54		
s141	15	s142	35	s150	55		
s151	16	s152	36	s160	56		
s161	17	s162	37	s170	57		
s171	18	s172	38	s180	58		
s181	19	s182	39	s190	59		
s191	20	s192	40	s200	60		

Fonte: Autor

7 TRABALHOS RELACIONADOS

Neste capítulo são apresentados diferentes trabalhos que utilizam conceitos de *QoC* envolvendo avaliação de critérios por meio de métodos de cálculo e realizando agregação dos valores de forma a permitir a comparação de sensores e/ou informação contextual.

A maioria dos trabalhos apresentados nesta seção já foram mencionados na seção 2.3.2 ou ao longo do capítulo 3, mas optou-se por recapitulá-los para permitir uma análise comparativa de suas características em relação ao trabalho desenvolvido, bem como em relação aos outros trabalhos.

7.1 Kim & Lee

Em (KIM; LEE, 2006), os autores propõem pela primeira vez na literatura o uso de métodos de cálculo para os critérios de *QoC* (MARIE et al., 2015), bem como um método de agregação, já apresentados no capítulo 3. O propósito dos autores era avaliar o contexto tendo em mente aplicações de *home care*, citando como exemplo o tratamento de pacientes hipertensos, utilizando *QoC* para definir quais sensores seriam mais adequados para indicar a situação dos pacientes.

Este trabalho foi incluído na comparação (Tabela 7.1) por lançar as bases do uso de *QoC* de forma objetiva com o uso de valores matemáticos, bem como apresentar uma etapa de avaliação e outra de agregação, servindo de inspiração para a arquitetura RD.

7.2 Yasar et al.

(YASAR et al., 2011) prevê o uso de critérios de *QoC* e um método de agregação para comparar sensores em redes veiculares Ad-hoc (*VANETs*), permitindo aos motoristas trocar informações sobre as rotas, e que a informação fosse atualizada apenas quando uma informação mais nova e confiável, de acordo com os critérios de *QoC*, fosse apresentada.

Além de definir formalmente o método de média ponderada para agregação, apresenta o princípio de utilização de *QoC* para seleção de informações em numerosos dispositivos móveis e com características heterogêneas, um dos principais requisitos para uma aplicação que realize seleção de sensores *IoT*.

7.3 Neisse

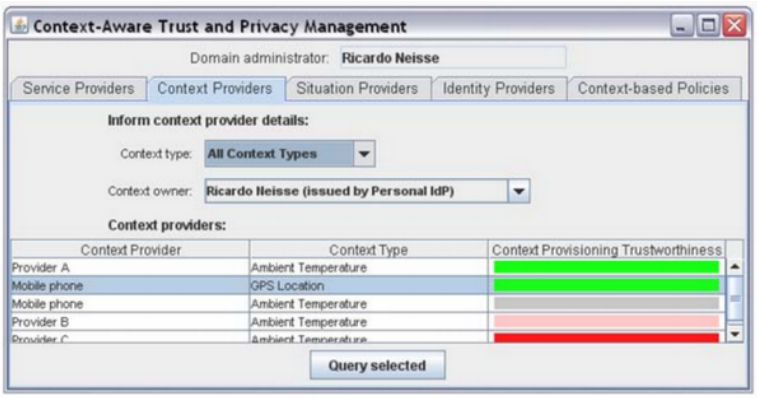
Neisse (NEISSE, 2012) tem como objetivo analisar o *tradeoff* entre privacidade e adaptação de serviços baseados em contexto, levando em conta tanto a seleção dos provedores de serviço por parte dos usuários, quanto a seleção dos provedores de contexto por parte dos provedores de serviço.

Neste intuito, Neisse apresenta como solução um sistema que permita gerenciar *privacidade* e *confiança* (*trust*), tanto para os usuários quanto para os provedores de serviço. Este sistema conta com um modelo dedicado a *QoC*, utilizando critérios de *QoC* recorrentes na literatura, tais como acurácia, precisão, resolução temporal e espacial, atualidade (*freshness*) e probabilidade.

Separa ainda o modelo de confiança do modelo *QoC*, o que o diferencia das outras soluções encontradas na literatura, que incluem confiança como critério de *QoC* chamado de *confiabilidade*.

Além disso, o gerenciamento das definições e preferências, bem como a seleção de provedores de contexto, é feito através de uma interface gráfica (Figura 7.1), servindo como uma das inspirações para a interface gráfica usada na implementação e prevista no modelo da arquitetura RD.

Figura 7.1: Interface de seleção de provedores de contexto do gerenciador proposto por Neisse.



Fonte: (NEISSE, 2012), adaptada

7.4 CASSARAM

O *CASSARAM* (*Context-aware sensor search, selection and ranking model*) (PERERA et al., 2013) foi concebido para permitir a seleção de sensores com capacidades *IoT* em meio a um grande conjunto de sensores, utilizando como motivador para sua idealização um cenário de agricultura, em que os sensores indicam os níveis de umidade para controlar os sistemas de irrigação das plantações.

Embora os autores não tenham mencionado *QoC* em seu trabalho, utilizam diferentes critérios de contexto, como acurácia, precisão, atualidade etc., e aplicam métodos de cálculo para definirem uma nota, a qual é utilizada por um método final, que agrega estes dados, e gera um valor final, o qual é utilizado para realizar um ranqueamento, permitindo a seleção dos sensores *IoT* que mais se adequem aos requisitos do usuário que pretende selecionar sensores para sua aplicação.

Para agregação, introduzem o *CPWI* (*Comparative Priority-based Weighted Index*), ou *EDB* (*Euclidean Distance-Based*), já apresentado na seção 3.2.

A captura de preferências dos usuários (*UPC*, *User Preference Capture*, como chamam os autores) é realizada em uma interface gráfica (Figura 7.2), utilizando *sliders*. A partir desta consulta, é construída uma consulta *SPARQL* (HEBELER et al., 2011), linguagem recomendada pela *W3C* (THE WORLD WIDE WEB CONSORTIUM (W3C), 2018), que busca os sensores na rede.

Ao final, uma lista com os sensores que mais se aproximem do ideal apresentado pelo usuário é retornada.

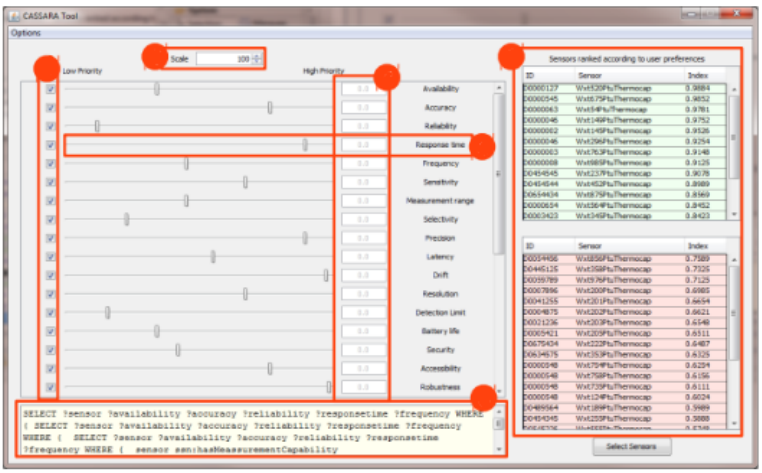
Dos trabalhos apresentados nesta seção, o *CASSARAM* é o que mais fortemente se relaciona à RD, tendo sido sua principal inspiração, principalmente em relação à captura das preferências do usuário utilizando uma interface gráfica.

7.5 OpenCOPI

O *OpenCOPI* (*Open COntext Platform Integration*) (LOPES et al., 2014a) é um *middleware* para provisionamento de contexto que utiliza *QoC* e *QoS* para selecionar planos de execução que dependem de combinações de provedores de serviço com base na nota de qualidade atribuída aos provedores de contexto.

É interessante notar que o conceito apresentado pelos autores como agregação difere do que normalmente se encontra na literatura. Como diferentes planos de execução

Figura 7.2: Interface gráfica do CASSARAM.



Fonte: (PERERA et al., 2013), adaptada

são construídos com base em diferentes combinações de serviços, primeiramente agrega-se as notas de cada critério. Por exemplo, para avaliar o valor de corretude de um plano de execução, os valores de corretude de todos os provedores de serviço são agregados através de um produtório de seus valores, de forma que combinações de serviços que tenham um provedor de serviço com valor baixo para o critério acabam sofrendo grande influência na nota agregada por causa deste provedor com nota baixa.

Só após agregar individualmente cada critério de *QoC* é aplicada uma média ponderada de todos os critérios, utilizando pesos e atribuições de acordo com as preferências do usuário, cujo resultado os autores chamam de *utilidade* do plano de execução.

Desta forma, por apresentar o uso de conceitos *QoC* e essa diferença conceitual em relação aos demais trabalhos, optou-se por incluir o *OpenCOPI* na lista de trabalhos relacionados.

7.6 Cuida

O Cuida (NAZÁRIO et al., 2015) é um ambiente de *homecare* ao qual foi aplicado um modelo de conhecimento de *QoC*, utilizando ontologias.

O objetivo deste modelo é facilitar o entendimento da semântica de *QoC*, permitindo relacionar conceitos, de forma a contornar a falta de padronização de nomenclatura e de modelagens específicas para o domínio, o que acaba por entregar capacidade de interoperabilidade e compartilhamento entre dispositivos e pessoas (NAZÁRIO et al., 2015).

Para criar o modelo, os autores apresentam uma vasta relação de critérios e métodos de cálculo para critérios *QoC*, apresentando também a equivalência de termos entre trabalhos distintos.

Buscando validar o modelo, apresenta ainda uma arquitetura (Figura 7.3), incluindo, no nível de processamento de contexto, módulos de coleta, avaliação e agregação de *QoC*, além de um módulo de política de segurança.

A partir desta arquitetura é feita uma implementação, em um caso real, em que os critérios de *QoC* são utilizados para analisar se a informação obtida por sensores tais como medidores de pressão, acelerômetros etc. indica a realidade de um paciente monitorado em sua casa.

São utilizados cinco critérios e agregação por média ponderada; no entanto, um dos critérios utilizados, *significância*, não altera o valor agregado de *QoC*, sendo utilizado como *flag* de alerta, quando seu valor for 1, de que os últimos valores obtidos com *QoC* suficiente indicam potenciais problemas com os pacientes monitorados.

Este trabalho se relaciona à RD principalmente por causa da arquitetura apresentada e dos métodos de cálculo e agregação utilizados. Além disso, a grande quantidade de definições e o estudo sobre o uso de ontologias já o tornam digno de menção, podendo motivar trabalhos futuros.

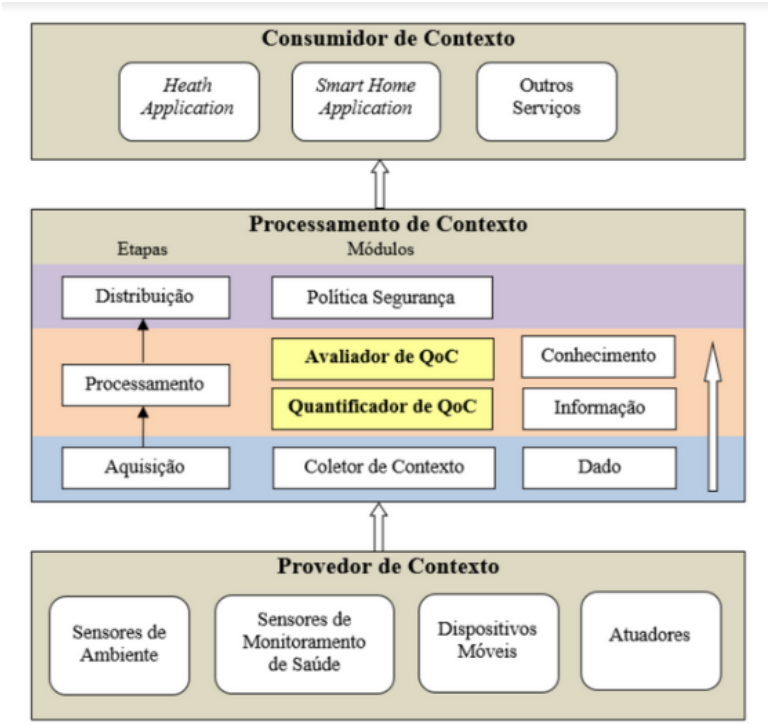
7.7 *QoCIM*

O *QoCIM* (MARIE et al., 2015) é um *framework* orientado a modelos baseado em *QoC*, visando auxiliar os desenvolvedores de aplicações durante todo o ciclo de vida de *QoC* em ambientes distribuídos, com o objetivo de adaptar o gerenciamento de *QoC* à realidade da adoção em larga escala de *IoT*.

Para motivar o trabalho, os autores apresentam um cenário de medição de poluição em ambientes urbanos, no qual sensores de diversos tipos são instalados em veículos, ruas etc. e suas informações contextuais são apresentadas aos usuários de maneira categorizada, de acordo com o nível de qualidade do ar. Como existem diferentes tipos de usuários, com riscos diferentes envolvidos, é necessário que haja garantia na qualidade do contexto provido, e.g. aplicações criadas para a área e profissionais da saúde em setores urbanos precisam de informação com alta qualidade, enquanto uma aplicação de uso geral para usuários comuns pode utilizar informação de média qualidade.

Os autores então explicam que o *QoCIM* permite aos usuários incluir os critérios

Figura 7.3: Arquitetura proposta no Cuida, utilizada para validar o modelo de conhecimento proposto pelos autores.



Fonte: (NAZÁRIO et al., 2015), adaptada

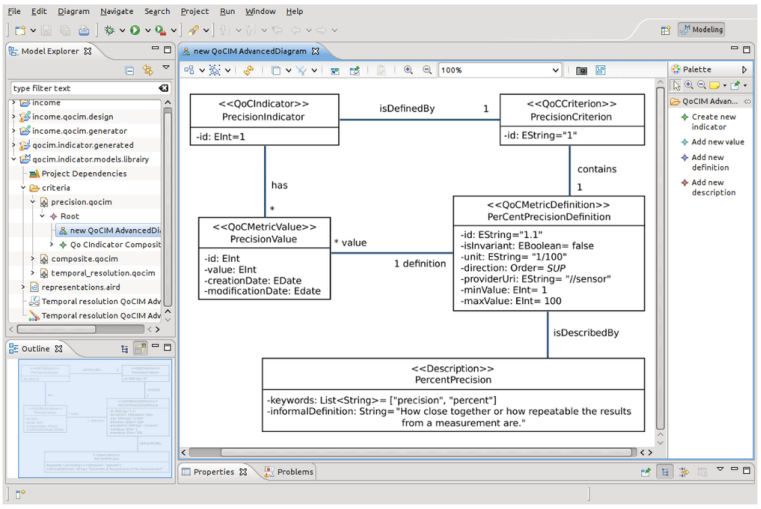
e métodos que desejarem, podendo ser estes critérios de 3 tipos: *primitivos*, que não dependem de outros critérios; *compostos*, que são calculados a partir de outros critérios; e *invariantes*, em que os valores que podem ser atribuídos para o critério seguem um conjunto de possibilidades bem definido e limitado.

A definição dos critérios, não importa de qual tipo, é então feita através de um editor gráfico para definições em formato *XML*, permitindo incluir quantos critérios e de quantas formas se queira, buscando sanar a falta de uma padronização quanto a nomenclatura e métodos de cálculo.

Além disso, os autores citam uma preocupação com o uso de dispositivos que possam tanto produzir quanto consumir contexto, como acontece com os telefones móveis, largamente distribuídos e próximos da fonte de contexto, para processar informações contextuais e enviá-las a outras camadas. Isto indica, embora os autores não utilizem diretamente a expressão *Edge Computing*, uma preocupação em adotar este paradigma.

Assim, as preocupações em comum, o uso de uma interface gráfica e a capacidade de permitir aos usuários a inclusão de critérios *QoC* relacionam este trabalho à RD.

Figura 7.4: Interface gráfica do *Framework QoCIM*.



Fonte: (MARIE et al., 2015)

7.8 EXEHDA-QoC

O *EXEHDA-QoC* (XAVIER et al., 2017) é um arcabouço (ou *framework*) para gerência de indicadores de *QoC* que estende as funcionalidades de manipulação de contexto do *middleware EXEHDA* (LOPES et al., 2014b)(YAMIN et al., 2005)(XAVIER et al., 2017), que implementa os aspectos básicos de tratamento de *QoC*, como a utilização de parâmetros e métodos de agregação, e também uma estratégia para reduzir o número de conflitos de contexto, utilizando para isso políticas baseadas em qualidade.

O modelo desenvolvido leva em conta publicações realizadas por provedores de contexto, de forma a caracterizar algumas especificidades dos processos através de parâmetros de *QoC*.

O cálculo de *QoC* para as publicações é realizado inicialmente através de métodos de cálculo, ligados aos indicadores de *QoC*, os quais são aplicados sobre os parâmetros de *QoC* disponíveis. Desta forma, um indicador de *QoC* pode ter diferentes métodos de cálculo para obter uma avaliação, a qual deve retornar um valor entre 0 e 1. Por fim, um avaliador de *QoC*, composto por um algoritmo e um conjunto de métodos de cálculo para os indicadores, é responsável por determinar o algoritmo a ser usado, quais os indicadores disponibilizados e seus métodos de cálculo correspondentes.

Após a obtenção dos indicadores de *QoC*, é aplicado um método de agregação, que é a nota final de *QoC* da publicação. A solução permite utilizar diferentes métodos de agregação, garantindo flexibilidade às aplicações que utilizem o modelo. Além disso,

o modelo prevê a possibilidade de incluir novos métodos de cálculo, indicadores etc. permitindo estender as funcionalidades do arcabouço.

A partir dos elementos apresentados, os autores definem as políticas de *QoC*, que incluem um avaliador, pesos e um limiar mínimo. Se o valor de *QoC* de uma publicação superar este limiar, a publicação é aceita; caso contrário, é descartada.

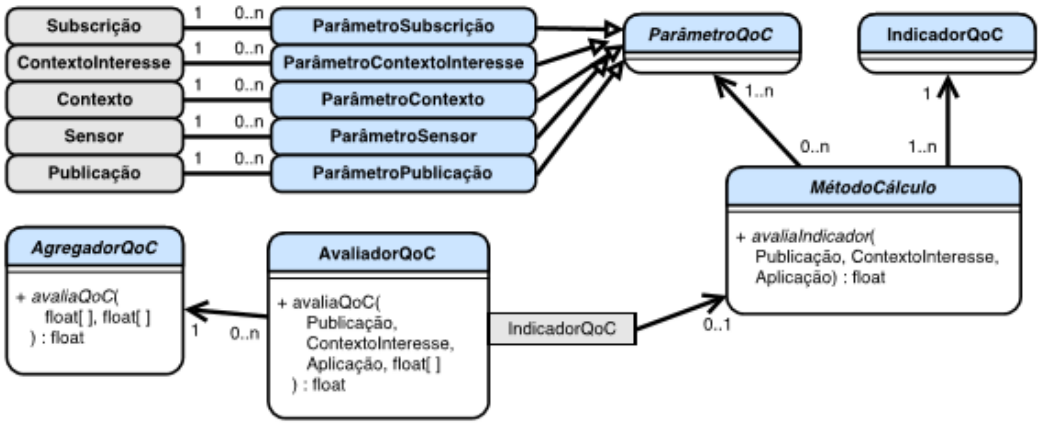
Os autores apresentam também um estudo de caso simulado em que é buscado determinar se usuários estão localizados em ambientes internos ou externos. Como trabalho futuro, citam a integração com servidores de borda para avaliação de qualidade mais próxima do local de aquisição de dados contextuais e consequente atuação, evidenciando a preocupação com adoção do paradigma *Edge Computing*.

É possível perceber que o nível de detalhamento das abstrações dos elementos utilizados para valorar *QoC* neste modelo é maior do que o apresentado pelas outras soluções apresentadas nos trabalhos relacionados. Para a RD, optou-se por adotar uma modelagem mais simples, mas suficiente para os objetivos da arquitetura. No entanto, é possível alterar alguns dos aspectos em trabalhos futuros para que estejam em conformidade com o modelo do *EXEHDA-QoC*, de forma a permitir integração entre as soluções.

Os autores apresentam ainda um histórico sobre qualidade de contexto e trabalhos relacionados que foram considerados especialmente relevantes, de forma que alguns deles foram citados em diferentes trechos do capítulo 2, e outros foram incluídos nesta seção.

Excetuando-se o *CASSARAM*, este trabalho é o que apresenta maiores preocupações e definições em comum, justificando-se sua inclusão no texto.

Figura 7.5: Diagrama de classes do modelo *EXEHDA-QoC* integrado ao *EXEHDA*.



Fonte: (XAVIER et al., 2017)

7.9 Considerações Finais

Esta seção apresentou os principais trabalhos relacionados à RD, resumindo seus objetivos e características, procurando justificar porque foram incluídos na comparação.

Alguns critérios de comparação são apresentados na Tabela 7.1. Os critérios inclusos nesta tabela levam em conta as principais características, segundo a literatura, que uma solução utilizando *QoC* deve apresentar, bem como os objetivos do trabalho em relação a *Edge Computing* e facilidade para os usuários.

Através da análise do texto e do comparativo da Tabela 7.1, é possível perceber que poucos trabalhos apresentaram formalmente novos métodos de cálculo para os critérios de *QoC*, sendo uma das áreas que pode ser explorada em trabalhos futuros.

Além disso, nenhum trabalho encontrado apresenta a ideia de um sistema de recomendação através de Perfis para que o usuário possa definir as preferências e critérios de acordo com a aplicação que deseja realizar, sendo um dos aspectos que diferenciam a RD, devendo ser um dos focos em trabalhos futuros.

Outro diferencial da RD é a preocupação com o uso de *Edge Computing*, apresentada em apenas dois dos trabalhos relacionados; e a análise de desempenho decorrente desta preocupação.

Todos os aspectos apresentados evidenciam a relevância dos temas de pesquisa abordados, bem como do estudo realizado.

Tabela 7.1: Comparativo de trabalhos relacionados de acordo com presença de critérios selecionados.

Trabalho/Critério	Utiliza Interface Gráfica	Utiliza métodos da literatura para cálculo de dimensões QoC	Introduz novos métodos de cálculo	Prevê uso de Perfis de QoC	Permite uso de diferentes métodos de agregação	Utiliza métodos de agregação encontrados na literatura	Introduz novos métodos de agregação	Faz análise de desempenho de métodos de agregação	Prevê Edge Computing	Permite estender a solução
CASSARAM [Perera et al., 2013]	X						X	X	X	
Cuida [Nazário et al., 2015]	X	X				X				
Exehda-QoC [Xavier et al. 2017]	X	X			X	X	X		X	X
[Kim & Lee, 2006]		X	X				X			
[Neisse, 2012]	X	X					X			
OpenCOPI [Lopes et al. 2014]		X				X				X
QoCIM [Marie et al. 2015]	X	X			X	X			X	X
[Yasar et al., 2011]			X				X			
RD	X	X	X	X	X	X		X	X	X

Fonte: Autor

8 CONCLUSÃO E TRABALHOS FUTUROS

Na Dissertação foi realizado estudo dos principais conceitos envolvidos na criação da arquitetura RD, tendo sido apresentado no capítulo 2, com um histórico que foi desde a primeira menção da expressão *Quality of Context* até a evolução dos paradigmas computacionais para *Cloud*, *Edge* e *Fog Computing*.

Além disso, no capítulo 3 foi apresentada uma compilação dos métodos de cálculo e agregação encontrados na literatura, os quais foram utilizados na implementação da arquitetura, bem como o método *AHP*, que serviu como base para comparação qualitativa entre métodos de agregação.

No capítulo 4, foi apresentada a modelagem da *arquitetura RD*, com seus módulos e a interação entre eles, bem como a descrição dos submódulos e suas interações. Dentre estes módulos, destaca-se o módulo de *Perfis QoC*, que visa auxiliar os programadores de aplicação a tomarem decisões quanto ao uso de critérios, métodos, pesos e demais definições necessárias para realizar a seleção de sensores, de acordo com a aplicação desejada, e com base em definições dadas por outros programadores de aplicação.

No capítulo 5, detalhes da implementação e das decisões envolvidas, como escolha das ferramentas e dos métodos a serem implementados, foram apresentados, juntamente com as diferentes configurações de distribuição de módulos entre servidores centrais e *Edge Devices*, que serviram para avaliar o *tradeoff* entre redução de tráfego de dados na rede e uso de recursos computacionais destes dispositivos.

Esta avaliação foi apresentada no capítulo 6, indicando um *tradeoff* compensador entre estes aspectos, tendo o volume de dados sido reduzido em mais de 80% ao se realizar tanto a avaliação quanto a agregação de *QoC* no *Edge Device*.

Além disso, ainda no capítulo 6, foi avaliado o sistema de Perfis *QoC*, através de entrevistas e questionários, os quais indicaram que este sistema é útil aos programadores de aplicação, mas que ainda deve ser melhor avaliado e aprimorado, pois a avaliação de sentimento dos usuários baseada nos questionários indicou a necessidade de refinamento do sistema de Perfis *QoC*.

Também no capítulo 6 foi realizada uma comparação entre os métodos de agregação *EDB* e *WAVG*, buscando definir um método padrão para sugestão inicial aos programadores de aplicação. Para isto, foram avaliados os tempos de execução de cada um em diferentes configurações de número de dispositivos e de critérios *QoC*, indicando vantagem para o método *WAVG*. Uma avaliação da qualidade dos ranqueamentos também

foi feita, utilizando um cenário de decisão que permitisse utilizar os resultados de um ranqueamento feito pelo método *AHP* como base de comparação.

Ficou demonstrada novamente vantagem para o método *WAVG*. Entretanto, concluiu-se que o método *EDB* não pode ser totalmente descartado como possível padrão, sendo necessário realizar novos estudos sobre este método a fim de determinar a possibilidade ou não de uso como padrão. Além disso, a possibilidade de utilizá-lo para comparar contra uma configuração ideal indicada pelo usuário, além dos pesos, pode ser compensadora. De qualquer forma, *WAVG* permanece sendo o método padrão indicado, seguindo a tendência apresentada na literatura.

Ainda no capítulo 6, por fim, foram apresentados dois estudos de caso, simulados através da ferramenta *CupCarbon*, para cenários de redes oportunistas e agricultura, em que se demonstrou ser possível utilizar a arquitetura RD para casos reais, sendo necessário, no entanto, novo estudo, saindo do campo da simulação para o uso em dispositivos reais.

No capítulo 7 foram apresentados trabalhos relacionados, os quais não só permitiram tomar decisões na modelagem e implementação da arquitetura RD, como também indicaram outras áreas de pesquisa que podem e devem receber especial atenção em trabalhos futuros, podendo ser citada como exemplo a utilização de ontologias para resolver problemas de nomenclatura, ou mesmo para auxiliar na definição dos Perfis de *QoC*.

Considerando o que foi exposto ao longo do trabalho, podemos retomar as questões apresentadas como problema de pesquisa na introdução, apresentando agora as respostas obtidas por meio de experimentos e literatura:

1. ***É possível reduzir o tráfego de dados na rede para aplicações que dependam de seleção de sensores utilizando* QoC *e* Edge Computing*, sem que haja um aumento proibitivo, a ponto de atrasar computações por esgotamento, do uso dos outros recursos computacionais do dispositivo* Edge*?*** Sim. Nos experimentos, houve redução de mais de 80% no volume de dados enviados pela rede, com análise de *tradeoff* utilizando método de produtório encontrado em (DANIELS; WERNER; BAHILL, 2001) indicando que o aumento no uso de recursos computacionais é aceitável e compensador considerando o objetivo de não sobrecarregar os dispositivos a ponto de prejudicar suas computações. No entanto, é necessário realizar mais estudos, já que a análise de *tradeoff* é subjetiva, e depende das configurações de hardware utilizadas nos testes.
2. *É viável utilizar dispositivos de borda para realizar parte das funções de cálculo*

QoC, considerando seus recursos limitados em comparação a sistemas computacionais tradicionais e Cloud? Sim, se considerarmos as configurações utilizadas para os testes feitos como representando adequadamente a média das capacidades dos dispositivos de borda mais utilizados, o que, no entanto, deve ser confirmado em estudo dedicado a este fim.

3. *É possível facilitar a tarefa do usuário que definirá os critérios de avaliação QoC através de padronização de definição de critérios?* Os indícios obtidos através da avaliação do sistema de Perfis *QoC* idealizado com este intuito, através dos questionários apresentados a especialistas de domínio de aplicação, apontam nesta direção. Contudo, é necessário realizar um estudo mais amplo, com amostragem que permita afirmar com segurança a capacidade do sistema de Perfis em facilitar a realização de definições pelo usuário, bem como, possivelmente, o uso de outros métodos com o mesmo propósito.
4. *Qual o desempenho computacional das funções de agregação utilizadas em trabalhos diversos? Alguma destas funções é superior às outras, em termos de qualidade do ranqueamento e tempo computacional, podendo ser adotada como padrão?* Embora não seja possível afirmar que uma função seja superior às outras, o estudo feito indica que o método mais encontrado na literatura, além de apresentar desempenho computacional aceitável, com crescimento praticamente linear do tempo de execução em relação ao número de sensores avaliados e critérios considerados, também retorna valores que, quando utilizados para criação de um ranqueamento, apresentam resultados que podem ser considerados fiéis à realidade do que está sendo avaliado. No entanto, mais uma vez, é preciso ressaltar que os meios utilizados para a avaliação dos métodos de agregação envolvem subjetividade significativa, sendo necessários mais estudos para que se possa estabelecer a resposta apresentada como sendo sempre verdadeira.

8.1 Trabalhos Futuros

Considerando as respostas aos problemas de pesquisa, bem como outras questões levantadas ao longo do texto, ficam estabelecidas como metas para trabalhos futuros:

- Aperfeiçoamento do sistema de Perfis de *QoC*, com posterior avaliação com amostragem suficiente entre os especialistas de domínio para que se possa afirmar a

utilidade do sistema inequivocamente;

- A realização de mais experimentos, utilizando variadas configurações de *hardware*, para corroborar os resultados obtidos em relação às vantagens conferidas pelo uso de *Edge Computing*;
- O uso de ontologias para sistemas de conhecimento que permitam contornar a falta de padronização dos métodos de cálculo, e também para melhorar o sistema de Perfis *QoC*;
- Implementar consultas utilizando a linguagem *SPARQL* para busca de sensores;
- Adaptar a modelagem, em especial dos módulos de avaliação e agregação de *QoC*, à modelagem apresentada no *EXEHDA-QoC*, no intuito de estudar a viabilidade de integração com outras soluções de *QoC*;
- Avaliar o desempenho dos outros métodos de agregação que foram implementados, no intuito de definir quais podem ser utilizados em *Edge Devices* ou não;
- Realizar experimentos utilizando dispositivos *IoT* reais em aplicações que tenham utilidade prática, visando confirmar a viabilidade de uso da arquitetura RD para resolver problemas.

8.2 Contribuições

As principais contribuições resultantes do trabalho realizado, com destaque para os itens 4 e 5, para o avanço em relação ao Estado da Arte, são resumidas como:

1. Uma nova definição, a partir das definições encontradas na literatura, para *Computação Sensível ao Contexto*, enunciada como sendo "a capacidade de um sistema computacional em detectar, sentir, interpretar e responder a fatores humanos, do ambiente físico, e possivelmente de outros sistemas computacionais, de forma a prover informação relevante ou serviços a um usuário ou aplicação";
2. Uma comparação entre algoritmos de agregação utilizados na literatura, em termos de tempo computacional e ranqueamento final gerado;
3. Uma arquitetura para seleção de dispositivos *IoT* com base em *QoC*, tendo em mente o uso das vantagens de *Edge Computing*;
4. Uma avaliação sobre a viabilidade do uso de *Edge Computing* para a seleção de dispositivos *IoT*;

5. O uso de um sistema de Perfis *QoC*, visando facilitar o trabalho de especialistas de domínio em definir critérios *QoC* e realizar a seleção de dispositivos *IoT*.;
6. Uma avaliação preliminar sobre o uso do sistema de Perfis *QoC*, a partir de questionários respondidos por especialistas de domínio;
7. Um histórico da evolução de *QoC* como conceito e suas aplicações;
8. Uma compilação de métodos de cálculo e de agregação para *QoC* encontrados na literatura;
9. Dois estudos de caso baseados em aplicações reais, com uma simulação feita na ferramenta *CupCarbon*.

REFERÊNCIAS

ABOWD, G. D. et al. Towards a better understanding of context and context-awareness. In: SPRINGER. **International symposium on handheld and ubiquitous computing**. [S.l.], 1999. p. 304–307.

AI, Y.; PENG, M.; ZHANG, K. Edge computing technologies for internet of things: a primer. **Digital Communications and Networks**, Elsevier, v. 4, n. 2, p. 77–86, 2018.

ASHTON, K. et al. That 'internet of things' thing. **RFID journal**, Jun, v. 22, n. 7, p. 97–114, 2009.

BEAL, V. **The Differences Between Thick, Thin & Smart Clients**. 2018. Disponível em: <https://www.webopedia.com/DidYouKnow/Hardware_Software/thin_client_applications.asp>. Acesso em 28 de maio de 2018.

BEVYWISE INC. **The Bevywise IoT Simulator**. 2018. Disponível em: <https://www.bevywise.com/iot-simulator/>. Acesso em 29 de novembro de 2018.

BIRMAN, K.; JOSEPH, T. **Exploiting virtual synchrony in distributed systems**. [S.l.]: ACM, 1987.

BISGAARD, J. J.; HEISE, M.; STEFFENSEN, C. How is context and context-awareness defined and applied? a survey of context-awareness. **Aalborg university**, 2004.

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

BUCHHOLZ, T.; SCHIFFERS, M. Quality of context: What it is and why we need it. In: **In Proceedings of the 10th Workshop of the OpenView University Association: OVUA'03**. [S.l.: s.n.], 2003.

BURLEIGH, S. et al. Delay-tolerant networking: an approach to interplanetary internet. **IEEE Communications Magazine**, IEEE, v. 41, n. 6, p. 128–136, 2003.

BYERS, C. C.; WETTERWALD, P. Fog computing distributing data and intelligence for resiliency and scale necessary for iot: The internet of things (ubiquity symposium). **Ubiquity**, ACM, v. 2015, n. November, p. 4, 2015.

CASTRO, P.; MUNZ, R. Managing context data for smart spaces. **IEEE Personal Communications**, IEEE, v. 7, n. 5, p. 44–46, 2000.

CISCO. **Internet of Things At a Glance**. 2018. Disponível em: <https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf>. Acesso em 29 de maio de 2018.

CUPCARBON. **CupCarbon U-One**. 2018. Disponível em: <http://www.cupcarbon.com/>. Acesso em 28 de novembro de 2018.

DANIELS, J.; WERNER, P. W.; BAHILL, A. T. Quantitative methods for tradeoff analyses. **Systems Engineering**, Wiley Online Library, v. 4, n. 3, p. 190–212, 2001.

DEY, A. K. Understanding and using context. **Personal and ubiquitous computing**, Springer-Verlag, v. 5, n. 1, p. 4–7, 2001.

Facebook Inc. **GraphQL**. 2018. Disponível em: <https://graphql.org/>. Acesso em 30 de maio de 2018.

FILHO, J. B.; AGOULMINE, N. A quality-aware approach for resolving context conflicts in context-aware systems. In: IEEE. **Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on**. [S.l.], 2011. p. 229–236.

GATEWAY TECH. **Internet of Things**. 2018. Disponível em: <http://www.gatewaytechnolabs.co.uk/internet-things>. Acesso em 28 de maio de 2018.

GOOGLE LLC. **Android**. 2018. Disponível em: <https://www.android.com/>. Acesso em 01 de junho de 2018.

GOOGLE LLC. **Android Studio**. 2018. Disponível em: <https://developer.android.com/studio/?hl=pt-br>. Acesso em 01 de junho de 2018.

GOOGLE LLC. **Android Studio Profiler**. 2018. Disponível em: <https://developer.android.com/studio/profile/memory-profiler?hl=pt-br>. Acesso em 01 de junho de 2018.

GSM ARENA. **The Top 20 most popular phones of 2017**. 2017. Disponível em: <https://www.gsmarena.com/the_20_most_popular_phones_of_2017-news-28892.php>. Acesso em 01 de junho de 2018.

HEBELER, J. et al. **Semantic web programming**. [S.l.]: John Wiley & Sons, 2011.

IBM. **IBM and Nokia Siemens Networks Announce World's First Mobile Edge Computing Platform**. 2013. Disponível em: <https://www-03.ibm.com/press/us/en/pressrelease/40490.wss>. Acesso em 27 de novembro de 2018.

IHS TECHNOLOGY. **The Evolving Internet of Things (IoT)**. 2018. Disponível em: <https://technology.ihs.com/577019/the-evolving-internet-of-things-iot>. Acesso em 01 de junho de 2018.

INTERNATIONAL BUREAU OF WEIGHTS AND MEASURES. **International vocabulary of basic and general terms in metrology**. [S.l.]: International organization for standardization, 1984.

IOT ANALYTICS. **State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating**. 2018. Disponível em: <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>. Acesso em 29 de maio de 2018.

ITU - INTERNATIONAL TELECOMMUNICATION UNION. **ITU-T Y.4000/Y.2060**. 2018. Disponível em: <https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=y.2060>. Acesso em 28 de maio de 2018.

JGRAPH LTD. **draw.io**. 2018. Disponível em: <https://www.draw.io/>. Acesso em 01 de junho de 2018.

KAHN, B. K.; STRONG, D. M.; WANG, R. Y. Information quality benchmarks: product and service performance. **Communications of the ACM**, ACM, v. 45, n. 4, p. 184–192, 2002.

KALTZ, J. W.; ZIEGLER, J.; LOHMANN, S. Context-aware web engineering: Modeling and applications. **Revue d'intelligence artificielle**, Lavoisier, v. 19, n. 3, p. 439–458, 2005.

KENKNIGHT, B. H. et al. **Method and apparatus for establishing context among events and optimizing implanted medical device performance**. [S.l.]: Google Patents, 2006. US Patent 7,043,305.

KERÄNEN, A.; OTT, J.; KÄRKKÄINEN, T. The one simulator for dtn protocol evaluation. In: ICST (INSTITUTE FOR COMPUTER SCIENCES, SOCIAL-INFORMATICS AND TECHNOLOGY. **Proceedings of the 2nd international conference on simulation tools and techniques**. [S.l.], 2009. p. 55.

KIM, Y.; LEE, K. A quality measurement method of context information in ubiquitous environments. In: IEEE. **Hybrid Information Technology, 2006. ICHIT'06. International Conference on**. [S.l.], 2006. v. 2, p. 576–581.

KREBS, R.; MOMM, C.; KOUNEV, S. Architectural concerns in multi-tenant saas applications. In: **Closer**. [S.l.: s.n.], 2012. v. 12, p. 426–431.

LENOVO. **Moto G4 Play**. 2018. Disponível em: <https://www.lenovo.com/za/en/smart-devices/moto/phones/Moto-G4-Play-Smartphone/p/WMD00000157>. Acesso em 01 de junho de 2018.

LIKERT, R. A technique for the measurement of attitudes. **Archives of psychology**, 1932.

LINE, M. B. et al. Safety vs security? In: **PSAM conference, New Orleans, USA**. [S.l.: s.n.], 2006.

LOPES, F. et al. Opencopi: middleware integration for ubiquitous computing. **International Journal of Parallel, Emergent and Distributed Systems**, Taylor & Francis, v. 29, n. 2, p. 178–212, 2014.

LOPES, J. L. et al. A middleware architecture for dynamic adaptation in ubiquitous computing. **J. UCS**, v. 20, n. 9, p. 1327–1351, 2014.

LOPEZ RESEARCH LLC. **Uma introdução à Internet das Coisas(IoT)**. 2018. Disponível em: <https://www.cisco.com/c/dam/global/pt_br/assets/brand/iot/iot/pdfs/lopez_research_an_introduction_to_iot_102413_final_portuguese.pdf>. Acesso em 28 de maio de 2018.

Lou Sands. **AHP Leader Example**. 2019. Disponível em: <https://en.wikipedia.org/wiki/Analytic_hierarchy_process_%E2%80%93_leader_example>. Acesso em 27 de julho de 2019.

MANZOOR, A.; TRUONG, H.-L.; DUSTDAR, S. On the evaluation of quality of context. In: SPRINGER. **European Conference on Smart Sensing and Context**. [S.l.], 2008. p. 140–153.

MARIE, P. et al. From ambient sensing to iot-based context computing: An open framework for end to end qoc management. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 15, n. 6, p. 14180–14206, 2015.

MARTIN, M.; NURMI, P. A generic large scale simulator for ubiquitous computing. In: IEEE. **2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services**. [S.l.], 2006. p. 1–3.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. National Computer Security Division, Information Technology Laboratory, 2011.

META-CHART. **Meta-Chart**. 2018. Disponível em: <https://www.meta-chart.com/pie>. Acesso em 29 de novembro de 2018.

MORGAN, J. A simple explanation of'the internet of things'. **Forbes/Leadership**, 2014.

NAZÁRIO, D. C. et al. Cuida: um modelo de conhecimento de qualidade de contexto aplicado aos ambientes ubíquos internos em domicílios assistidos. 2015.

NEISSE, R. **Trust and privacy management support for context-aware service platforms.** [S.l.]: University of Twente, Enschede, Netherlands, 2012.

OBJECT MANAGEMENT GROUP INC. **UML - Unified Modeling Language**. 2018. Disponível em: <http://www.uml.org/>. Acesso em 01 de junho de 2018.

OPEN-STD. **ISO/IEC 9899:TC3, C11**. 2018. Disponível em: <http://www.open-std.org/JTC1/SC22/WG14/www/docs/n1256.pdf>. Acesso em 28 de novembro de 2018.

ORACLE CORPORATION. **Code Tools: jmh**. 2018. Disponível em: <https://openjdk.java.net/projects/code-tools/jmh/>. Acesso em 28 de novembro de 2018.

ORACLE CORPORATION. **The Java Programming Language**. 2018. Disponível em: <https://www.java.com/pt_BR/download/faq/java8.xml>. Acesso em 01 de junho de 2018.

OXFORD ENGLISH DICTIONARY. Oed. **nd Learning. <http://dictionary.> oed. com**, 2009.

PANG, H.; TAN, K.-L. Authenticating query results in edge computing. In: IEEE. **Proceedings. 20th International Conference on Data Engineering**. [S.l.], 2004. p. 560–571.

PASCOE, J.; RYAN, N.; MORSE, D. Using while moving: Hci issues in fieldwork environments. **ACM Transactions on Computer-Human Interaction (TOCHI)**, ACM, v. 7, n. 3, p. 417–437, 2000.

PERERA, C. et al. Ca4iot: Context awareness for internet of things. In: IEEE. **2012 IEEE International Conference on Green Computing and Communications**. [S.l.], 2012. p. 775–782.

PERERA, C. et al. Context-aware sensor search, selection and ranking model for internet of things middleware. In: IEEE. **Mobile Data Management (MDM), 2013 IEEE 14th International Conference on**. [S.l.], 2013. v. 1, p. 314–322.

PRUD, E.; SEABORNE, A. et al. Sparql query language for rdf. 2006.

REGALADO, A. **Who Coined 'Cloud Computing'?** 2018. Disponível em: <https://www.technologyreview.com/s/425970/who-coined-cloud-computing/>. Acesso em 28 de maio de 2018.

RODOŠEK, G. D. **A Framework for IT Service Management**. [S.l.: s.n.], 2002.

SAATY, T. L. Decision-making with the ahp: Why is the principal eigenvector necessary. **European journal of operational research**, Elsevier, v. 145, n. 1, p. 85–91, 2003.

SAATY, T. L. Decision making with the analytic hierarchy process. **International journal of services sciences**, v. 1, n. 1, p. 83–98, 2008.

SAATY, T. L.; HU, G. Ranking by eigenvector versus other methods in the analytic hierarchy process. **Applied Mathematics Letters**, Elsevier, v. 11, n. 4, p. 121–125, 1998.

SATYANARAYANAN, M. The emergence of edge computing. **Computer**, IEEE, v. 50, n. 1, p. 30–39, 2017.

SATYANARAYANAN, M. et al. The case for vm-based cloudlets in mobile computing. **IEEE pervasive Computing**, IEEE, n. 4, p. 14–23, 2009.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-aware computing applications. In: IEEE. **wmcsa**. [S.l.], 1994. p. 85–90.

SCHMIDT, A.; BEIGL, M.; GELLERSEN, H.-W. There is more to context than location. **Computers & Graphics**, Elsevier, v. 23, n. 6, p. 893–901, 1999.

SHEIKH, K.; WEGDAM, M.; SINDEREN, M. V. Middleware support for quality of context in pervasive context-aware systems. In: IEEE. **null**. [S.l.], 2007. p. 461–466.

SHI, W. et al. Edge computing: Vision and challenges. **IEEE Internet of Things Journal**, IEEE, v. 3, n. 5, p. 637–646, 2016.

SIRAJ, S.; MIKHAILOV, L.; KEANE, J. A. Priest: an interactive decision support tool to estimate priorities from pairwise comparison judgments. **International Transactions in Operational Research**, Wiley Online Library, v. 22, n. 2, p. 217–235, 2015.

SPEARMAN, C. Correlation calculated from faulty data. **British Journal of Psychology, 1904-1920**, Wiley Online Library, v. 3, n. 3, p. 271–295, 1910.

THE APACHE SOFTWARE FOUNDATION. **Apache Netbeans**. 2018. Disponível em: <https://netbeans.org/>. Acesso em 20 de janeiro de 2018.

THE MINGW COMMUNITY. **Mingw-w64**. 2018. Disponível em: <http://mingw-w64.org/doku.php>. Acesso em 28 de novembro de 2018.

THE WORLD WIDE WEB CONSORTIUM (W3C). **SPARQL Overview**. 2018. Disponível em: <https://www.w3.org/TR/sparql11-overview/>. Acesso em agosto de 2017.

TUDO CELULAR TECNOLOGIA LTDA. **tudocelular.com**. 2018. Disponível em: <https://www.tudocelular.com/>. Acesso em 15 de agosto de 2018.

WAN, K. A brief history of context. **arXiv preprint arXiv:0912.1838**, 2009.

WEISER, M. The computer for the 21 st century. **Scientific american**, JSTOR, v. 265, n. 3, p. 94–105, 1991.

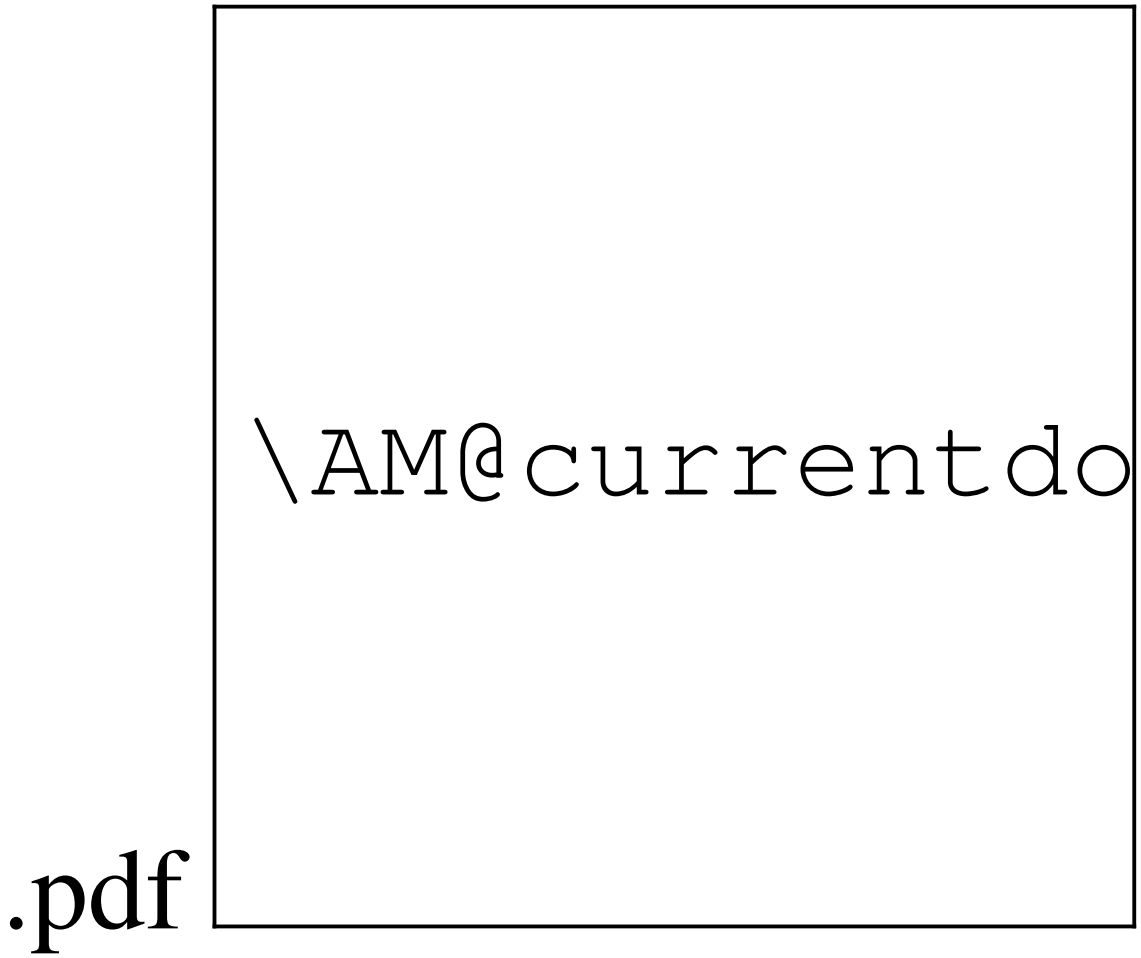
XAVIER, L. M. et al. Gerência de qoc: Uma abordagem direcionada ao tratamento de conflitos de contexto. In: SBC. **9º Simposio Brasileiro de Computacao Ubiqua e Pervasiva (SBCUP 2017)**. [S.l.], 2017. v. 9, n. 1/2017.

YAMIN, A. et al. Exehda: adaptive middleware for building a pervasive grid environment. In: IOS PRESS. **Proceedings of the 2005 conference on Self-Organization and Autonomic Informatics (I)**. [S.l.], 2005. p. 203–219.

YASAR, A.-U.-H. et al. When efficiency matters: Towards quality of context-aware peers for adaptive communication in vanets. In: IEEE. **2011 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.], 2011. p. 1006–1012.

YOON, C.; KIM, S. Convenience and tam in a ubiquitous computing environment: The case of wireless lan. **Electronic Commerce Research and Applications**, Elsevier, v. 6, n. 1, p. 102–112, 2007.

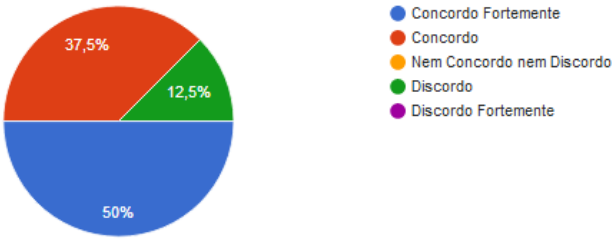
APÊNDICE A — ORIENTAÇÕES PARA EXPERIMENTO PERFIS QOC



APÊNDICE B — GRÁFICOS OBTIDOS ATRAVÉS DO GOOGLE FORMS SOBRE O QUESTIONÁRIO OBRIGATÓRIO DE PERFIS QOC

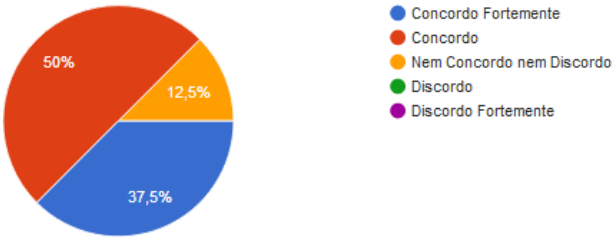
1. É fácil se orientar na interface do sistema de perfis

8 respostas



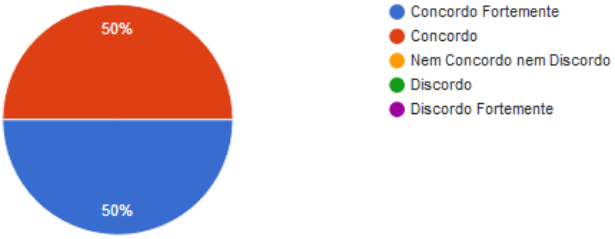
2. O posicionamento dos objetos na interface do sistema de perfis segue uma linha lógica de acordo com o processo modelado

8 respostas



3. A interface do sistema de perfis tem um bom tempo de resposta aos comandos

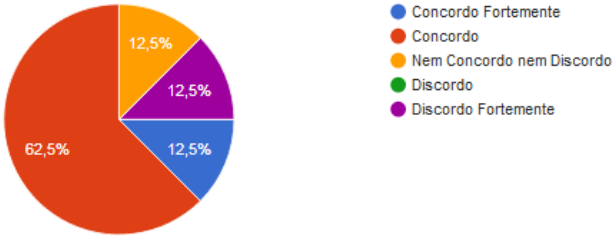
8 respostas



APÊNDICE C — GRÁFICOS OBTIDOS ATRAVÉS DO GOOGLE FORMS SOBRE O QUESTIONÁRIO OBRIGATÓRIO DE PERFIS QOC

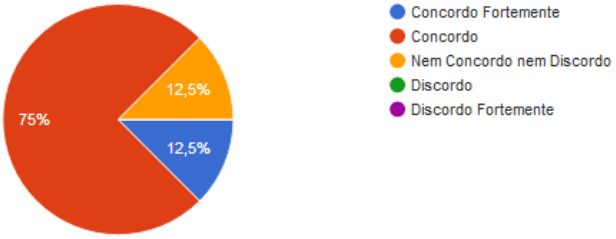
4. Consigo realizar as alterações que desejo com facilidade na interface do sistema de perfis

8 respostas



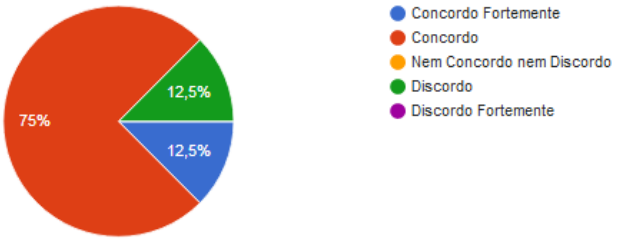
5. A interface do sistema de perfis apresenta descrições corretas para suas funções e elementos, que se comportam de acordo com estas descrições

8 respostas



6. As funcionalidades do sistema de perfis auxiliam na tarefa de definir critérios de qualidade de contexto para escolha de sensores

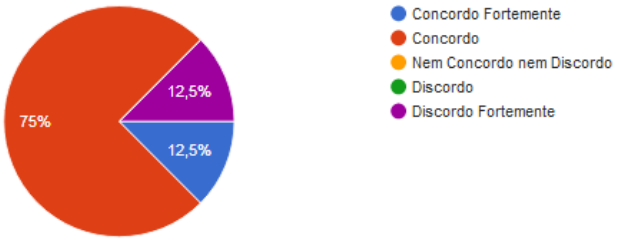
8 respostas



APÊNDICE D — GRÁFICOS OBTIDOS ATRAVÉS DO GOOGLE FORMS SOBRE O QUESTIONÁRIO OBRIGATÓRIO DE PERFIS QOC

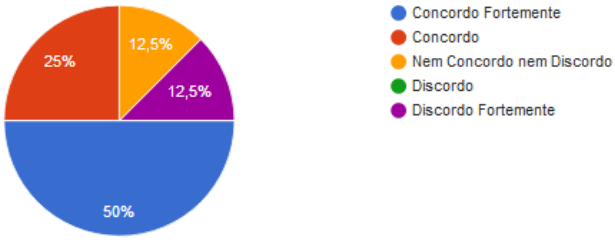
7. O sistema de perfis pode auxiliar a realizar meu trabalho

8 respostas



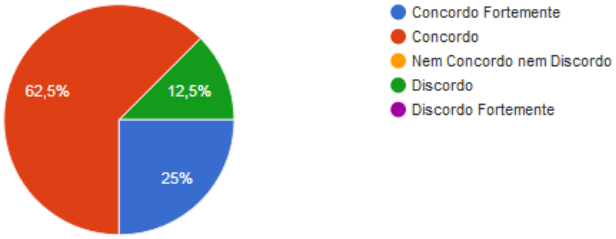
8. Os valores retornados pelo sistema de perfis podem ser usados para construir aplicações típicas com seleção de sensores

8 respostas



9. Usar o sistema de perfis para definir critérios e seus pesos me fará poupar tempo ao criar aplicações onde tais definições são necessárias

8 respostas



APÊNDICE E — EXEMPLO EXECUÇÃO JMH - WAVG 100 MIL ENTRADAS E 30 CRITÉRIOS

```
------------------------------------------------------------------------
Building samples 1.0-SNAPSHOT
------------------------------------------------------------------------
--- exec-maven-plugin:1.2.1:exec (default-cli) @ samples ---
# JMH 1.5.1 (released 1481 days ago, please consider updating!)
# VM invoker: C:\Program Files\Java\jdk1.8.0_162\jre\bin\java.exe
# VM options: <none>
# Warmup: 20 iterations, 1 s each
# Measurement: 500 iterations, 1 s each
# Timeout: 10 min per iteration
# Threads: 1 thread, will synchronize iterations
# Benchmark mode: Average time, time/op
# Benchmark: org.openjdk.jmh.samples.JMHSample_02_BenchmarkModes.measureAll

# Run progress: 0,00% complete, ETA 00:52:00
# Fork: 1 of 1
# Warmup Iteration   1: 7474,351 us/op
# Warmup Iteration   2: 7337,880 us/op
# Warmup Iteration   3: 7506,387 us/op
# Warmup Iteration   4: 6769,228 us/op
# Warmup Iteration   5: 6359,413 us/op
# Warmup Iteration   6: 6474,638 us/op
# Warmup Iteration   7: 6489,756 us/op
# Warmup Iteration   8: 6589,885 us/op
# Warmup Iteration   9: 6418,409 us/op
# Warmup Iteration  10: 6424,821 us/op
# Warmup Iteration  11: 6389,155 us/op
# Warmup Iteration  12: 6435,484 us/op
# Warmup Iteration  13: 6418,056 us/op
# Warmup Iteration  14: 6655,478 us/op
# Warmup Iteration  15: 6664,725 us/op
# Warmup Iteration  16: 6671,271 us/op
# Warmup Iteration  17: 6724,650 us/op
# Warmup Iteration  18: 6952,579 us/op
# Warmup Iteration  19: 6333,994 us/op
# Warmup Iteration  20: 6431,192 us/op
Iteration   1: 6451,374 us/op
Iteration   2: 6399,554 us/op
Iteration   3: 6368,504 us/op
Iteration   4: 6455,624 us/op
Iteration   5: 6414,064 us/op
Iteration   6: 6354,590 us/op
Iteration   7: 6433,380 us/op
Iteration   8: 6927,313 us/op
Iteration   9: 6449,316 us/op
Iteration  10: 6393,362 us/op
[…]
Iteration 490: 7369,669 us/op
Iteration 491: 7587,043 us/op
Iteration 492: 7324,883 us/op
Iteration 493: 7526,302 us/op
Iteration 494: 7265,544 us/op
Iteration 495: 7295,530 us/op
Iteration 496: 7686,414 us/op
Iteration 497: 7514,139 us/op
Iteration 498: 7534,339 us/op
Iteration 499: 7517,860 us/op
Iteration 500: 7564,481 us/op


Result: 7293,154 ñ(99.9%) 48,945 us/op [Average]
  Statistics: (min, avg, max) = (6354,590, 7293,154, 8831,214), stdev = 330,636
  Confidence interval (99.9%): [7244,209, 7342,099]

# Run complete. Total time: 00:08:44

Benchmark                                  Mode  Cnt     Score    Error  Units
JMHSample_02_BenchmarkModes.measureAll  avgt  500  7293,154 ± 48,945  us/op
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 8:45.359s
Finished at: Tue Feb 19 17:34:38 BRT 2019
Final Memory: 12M/491M
------------------------------------------------------------------------
```

APÊNDICE F — EXEMPLO CÓDIGO SENSSCRIPT CUPCARBON - ESTUDO DE CASO 2

```
//S01
wait
read peso1 peso2 peso3 peso4
mark $peso1 $peso2 $peso3 $peso4
set res1 = (9*$peso1)/10 res2 = (10*$peso2)/10 res3=(9*peso3) res4=(9*peso4)
send $res1 $res2 $res3 $res4
printfile $res1 $res2 $res3 $res4



//base
set peso1 = 8 peso2 = 8 peso3 = 10 peso4 = 5
send $peso1 $peso2 $peso3 $peso4
set array = ''

for i 0 200

wait
read res1 res2 res3 res4
mark $res1 $res2 $res3 $res4
set cursensor = $i
set wavg = $res1+$res2+$res3+$res4 $peso1+$peso2+$peso3+$peso4
set array = function storeaggr $wavg $cursensor $array

end

function rank $array
```

APÊNDICE G — EXEMPLO TABELAS INTERMEDIÁRIAS PARA *AHP* - CRITÉRIO ARMAZENAMENTO -3 DE UM TOTAL DE 15 TABELAS

Tabela G.1: Comparação Moto G4 Play em relação a outros telefone

	Valor	Comparado
Moto G4 Play	0,063	
Moto G4 Plus	0,125	2,000
K4 Novo	0,031	0,500
Zuk Edge	0,250	4,000
Zenfone Go ZB690KG	0,031	0,500
10 Evo	0,125	2,000
Galaxy J2 Prime	0,063	1,000
Redmi 4 Pro	0,125	2,000
Zenfone 3	0,063	1,000
Shine Lite	0,063	1,000
Galaxy On 7 2016	0,125	2,000
IPhone 7 Plus	1,000	16,000
Galaxy J5 Prime	0,125	2,000
Vibe K6 Plus	0,125	2,000
Xperia XZ	0,250	4,000
Nokia 6	0,250	4,000

Fonte: Autor

Tabela G.2: Comparação Moto G4 Plus em relação a outros telefone

	Valor	Comparado
Moto G4 Play	0,063	0,500
Moto G4 Plus	0,125	
K4 Novo	0,031	0,250
Zuk Edge	0,250	2,000
Zenfone Go ZB690KG	0,031	0,250
10 Evo	0,125	1,000
Galaxy J2 Prime	0,063	0,500
Redmi 4 Pro	0,125	1,000
Zenfone 3	0,063	0,500
Shine Lite	0,063	0,500
Galaxy On 7 2016	0,125	1,000
IPhone 7 Plus	1,000	8,000
Galaxy J5 Prime	0,125	1,000
Vibe K6 Plus	0,125	1,000
Xperia XZ	0,250	2,000
Nokia 6	0,250	2,000

Fonte: Autor

Tabela G.3: Comparação Xperia XZ em relação a outros telefone

	Valor	Comparado
Moto G4 Play	0,063	0,250
Moto G4 Plus	0,125	0,500
K4 Novo	0,031	0,125
Zuk Edge	0,250	1,000
Zenfone Go ZB690KG	0,031	0,125
10 Evo	0,125	0,500
Galaxy J2 Prime	0,063	0,250
Redmi 4 Pro	0,125	0,500
Zenfone 3	0,063	0,250
Shine Lite	0,063	0,250
Galaxy On 7 2016	0,125	0,500
IPhone 7 Plus	1,000	4,000
Galaxy J5 Prime	0,125	0,500
Vibe K6 Plus	0,125	0,500
Xperia XZ	0,250	
Nokia 6	0,250	1,000

Fonte: Autor

APÊNDICE H — TABELAS INTERMEDIÁRIAS PARA NORMALIZAÇÃO DOS CRITÉRIOS E CÁLCULO DE PREÇO E CUSTO BENEFÍCIO

Tabela H.1: Sem tratamento

Celulares	Bateria	Memória	Peso	Tela	Desempenho	Armazenamento	Câmera	Preço R$	Visual	Custo-Benefício	Marca?
Moto G4 Play	17h	2GB	137g	5''	5,5	16GB	8Mp	825	0,6	?	Motorola
Moto G4 Plus	22h	2GB	157g	5,5''	6,7	32GB	16Mp	1043	0,6	?	Motorola
K4 Novo	11h	1GB	145g	5''	1,9	8GB	8Mp	426	0,6	?	LG
Zuk Edge	19h	6GB	160g	5,5''	8,3	64GB	13Mp	2599	0,5	?	Lenovo
Zenfone Go ZB690KG	20h	1GB	270g	6,9''	3,9	8GB	8Mp	680	0,5	?	Asus
10 Evo	23h	3GB	174g	5,5''	7,5	32GB	16Mp	648	0,4	?	HTC
Galaxy J2 Prime	14h	1,4GB	160g	5''	2,4	16GB	8Mp	800	0,5	?	Samsung
Redmi 4 Pro	35h	3GB	156g	5''	7,9	32GB	13Mp	950	0,5	?	Xiaomi
Zenfone 3	22,5h	2GB	144g	5,2''	7,8	16GB	16Mp	1500	0,7	?	Asus
Shine Lite	4h	2GB	156g	5''	2,4	16GB	13Mp	700	0,4	?	Alcatel
Galaxy On 7 2016	20h	3GB	167g	5,5''	7,9	32GB	13Mp	1000	0,5	?	Samsung
IPhone 7 Plus	14h	3GB	188g	5,5''	9,5	256GB	12Mp	3500	0,5	?	Apple
Galaxy J5 Prime	16h	2GB	143g	5''	5,3	32GB	13Mp	1000	0,5	?	Samsung
Vibe K6 Plus	16h	3GB	169g	5,5''	6,6	32GB	16Mp	1300	0,5	?	Lenovo
Xperia XZ	10h	3GB	161g	5,2''	7,4	64GB	23Mp	4000	0,8	?	Sony
Nokia 6	18h	4GB	169g	5,5''	6,6	64GB	16Mp	1500	0,8	?	Nokia
IDEAL	14h	2GB	200g	6''	7	16GB	10Mp	700	0,7	?	IDEAL
Importância (0-10) Critérios	8	8	5	6	9	8	4	8	2	8	

Fonte: Autor

Tabela H.2: Com tratamento

AHP Celulares	Bateria	Memória	Peso	Tela	Desempenho	Armazenamento	Câmera	Preço	Visual	Custo-Benefício	Marca?
Moto G4 Play	17	2	137	5	5,5	16	8	825	0,6		Motorola
Moto G4 Plus	22	2	157	5,5	6,7	32	16	1043	0,6		Motorola
K4 Novo	11	1	145	5	1,9	8	8	426	0,6		LG
Zuk Edge	19	6	160	5,5	8,3	64	13	2599	0,5		Lenovo
Zenfone Go ZB690KG	20	1	270	6,9	3,9	8	8	680	0,5		Asus
10 Evo	23	3	174	5,5	7,5	32	16	648	0,4		HTC
Galaxy J2 Prime	14	1,4	160	5	2,4	16	8	800	0,5		Samsung
Redmi 4 Pro	35	3	156	5	7,9	32	13	950	0,5		Xiaomi
Zenfone 3	22,5	2	144	5,2	7,8	16	16	1500	0,7		Asus
Shine Lite	4	2	156	5	2,4	16	13	700	0,4		Alcatel
Galaxy On 7 2016	20	3	167	5,5	7,9	32	13	1000	0,5		Samsung
IPhone 7 Plus	14	3	188	5,5	9,5	256	12	3500	0,5		Apple
Galaxy J5 Prime	16	2	143	5	5,3	32	13	1000	0,5		Samsung
Vibe K6 Plus	16	3	169	5,5	6,6	32	16	1300	0,5		Lenovo
Xperia XZ	10	3	161	5,2	7,4	64	23	4000	0,8		Sony
Nokia 6	18	4	169	5,5	6,6	64	16	1500	0,8		Nokia
IDEAL	14	2	200	6	7	16	10	700	0,7		IDEAL
Importância (0-10) Critérios	8	8	5	6	9	8	4	8	2	8	

Fonte: Autor

Tabela H.3: Valores relativos por critério na comparação entre os telefones

AHP Celulares	Bateria (autonomia chamadas)	Memória	Peso	Tela	Desempenho (1-10)	Armazenamento	Câmera	Preço	Visual	Custo-Benefício	Marca?
Moto G4 Play	1,214285714	1	0,685	0,833333333	0,785714286	1	0,8	1,178571429	0,857142857		Motorola
Moto G4 Plus	1,571428571	1	0,785	0,916666667	0,957142857	2	1,6	1,49	0,857142857		Motorola
K4 Novo	0,785714286	0,5	0,725	0,833333333	0,271428571	0,5	0,8	0,608571429	0,857142857		LG
Zuk Edge	1,357142857	3	0,8	0,916666667	1,185714286	4	1,3	3,712857143	0,714285714		Lenovo
Zenfone Go ZB690KG	1,428571429	0,5	1,35	1,15	0,557142857	0,5	0,8	0,971428571	0,714285714		Asus
10 Evo	1,642857143	1,5	0,87	0,916666667	1,071428571	2	1,6	0,925714286	0,571428571		HTC
Galaxy J2 Prime	1	0,7	0,8	0,833333333	0,342857143	1	0,8	1,142857143	0,714285714		Samsung
Redmi 4 Pro	2,5	1,5	0,78	0,833333333	1,128571429	2	1,3	1,357142857	0,714285714		Xiaomi
Zenfone 3	1,607142857	1	0,72	0,866666667	1,114285714	1	1,6	2,142857143	1		Asus
Shine Lite	0,285714286	1	0,78	0,833333333	0,342857143	1	1,3	1	0,571428571		Alcatel
Galaxy On 7 2016	1,428571429	1,5	0,835	0,916666667	1,128571429	2	1,3	1,428571429	0,714285714		Samsung
IPhone 7 Plus	1	1,5	0,94	0,916666667	1,357142857	16	1,2	5	0,714285714		Apple
Galaxy J5 Prime	1,142857143	1	0,715	0,833333333	0,757142857	2	1,3	1,428571429	0,714285714		Samsung
Vibe K6 Plus	1,142857143	1,5	0,845	0,916666667	0,942857143	2	1,6	1,857142857	0,714285714		Lenovo
Xperia XZ	0,714285714	1,5	0,805	0,866666667	1,057142857	4	2,3	5,714285714	1,142857143		Sony
Nokia 6	1,285714286	2	0,845	0,916666667	0,942857143	4	1,6	2,142857143	1,142857143		Nokia
IDEAL	1	1	1	1	1	1	1	1	1		IDEAL

Fonte: Autor