Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Programa de Pós-Graduação em Engenharia de Eletricidade

Berto de Tácio Pereira Gomes

Uma Abordagem de Middleware com Suporte à Qualidade de Contexto voltada para Aplicações de Internet das Coisas

São Luís
2017

Berto de Tácio Pereira Gomes

Uma Abordagem de Middleware com Suporte à Qualidade de Contexto voltada para Aplicações de Internet das Coisas

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de DOUTOR em Engenharia de Eletricidade com área de concentração em Ciência da Computação.

Orientador: Francisco José da Silva e Silva

Doutor - UFMA

São Luís

2017

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Gomes, Berto de Tácio Pereira.
Uma Abordagem de Middleware com Suporte à Qualidade de Contexto voltada para Aplicações de Internet das Coisas / Berto de Tácio Pereira Gomes. - 2017.
218 f.

Orientador(a): Francisco José da Silva e Silva.
Tese (Doutorado) - Programa de Pós-graduação em Engenharia de Eletricidade/ccet, Universidade Federal do Maranhão, São Luis, 2017.

1. Internet das Coisas. 2. Middleware. 3. Qualidade de Contexto. I. Silva e Silva, Francisco José da. II. Título.

Berto de Tácio Pereira Gomes

Uma Abordagem de Middleware com Suporte à Qualidade de Contexto voltada para Aplicações de Internet das Coisas

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Berto de Tácio Pereira Gomes e aprovada pela comissão examinadora.

Aprovada em 08 de Novembro de 2017

BANCA EXAMINADORA

Francisco José da Silva e Silva (orientador)

Doutor - UFMA

María del Rosario Girardi Gutiérrez

Doutora - UFMA

Denivaldo Cícero Pavão Lopes

Doutor - UFMA

Fábio Moreira Costa

Doutor - UFG

Markus Endler

Doutor - PUC-Rio

(Suplente)

Doutor (a)

Em memória ao Prof. Zair Abdelouahab, com quem tive o privilégio de conviver e de aprender com seus exemplos.

Resumo

A qualidade de contexto em geral tem impacto direto e significativo sobre as experiências dos usuários de aplicações de Internet das Coisas, que podem ser positivas ou negativas, dependendo do tipo e nível de qualidade de contexto disponível para aplicação. Esta tese aborda como problemática de pesquisa o fato de que é muito complexo desenvolver aplicações de Internet das Coisas, principalmente aquelas que têm exigências de qualidade de contexto. Ainda que diversas abordagens de middleware tenham sido propostas na literatura, poucas delas oferecem suporte para aplicações que possuem requisitos de qualidade de contexto. De modo geral, os trabalhos relacionados ainda têm limitações em relação à alguns aspectos, tais como gerenciamento de sensores heterogêneos, comunicação distribuída, confiabilidade em ambientes móveis, provisionamento, avaliação e monitoramento de qualidade de contexto, principalmente em relação à qualidade de serviço, que sofre com a insuficiência de parâmetros. Com o objetivo superar algumas dessas limitações e facilitar o desenvolvimento de aplicações de Internet das Coisas que possuem requisitos de qualidade de contexto, esta tese propôs uma nova abordagem de middleware de contexto, denominada Mobile Hub / Context Data Distribution Layer. A solução proposta apresenta suporte ao gerenciamento e qualidade de contexto mais amplo e eficiente que as abordagens relacionadas em aspectos como gerenciamento de sensores, provisionamento, avaliação e monitoramento de qualidade de contexto. A avaliação quantitativa da solução proposta provou sua eficiência em relação ao tempo de entrega dos dados, confiabilidade em redes instáveis, tempo de monitoramento e consumo de memória e bateria. Já a prova de conceito mostrou que os mecanismos da solução proposta se aplicam a diversos domínios de aplicação, enquanto que os resultados do estudo de caso com usuários mostraram um elevado grau de satisfação dos mesmos em relação às abstrações de programação, serviços e suporte à qualidade de contexto providos pela solução proposta. Portanto, os resultados mostram a eficiência da solução proposta como abordagem facilitadora do desenvolvimento de aplicações de Internet das Coisas com requisitos de qualidade de contexto.

Palavras-Chave: Middleware, Qualidade de Contexto, Internet das Coisas.

Abstract

Quality of Context (QoC) in general, that is, both Quality of Information (QoI) and Quality of Service (QoS), has a direct and significant impact on IoT applications users' experiences, which can be positive or negative, depending on the type and level of QoC available for application. This doctoral thesis approaches as a research problem the fact that it is very complex to develop IoT applications, especially those having context quality requirements. Although several middleware approaches have been proposed in the literature, few of them offer support for applications that have QoC requirements. In general, related work still has limitations on some aspects, such as heterogeneous sensors management, distributed communication, reliability in mobile environments, provisioning, evaluation and monitoring of QoC, mainly in relation to quality of service, which suffers from the lack of parameters. In order to overcome some of these limitations and facilitate the development of IoT applications that have QoC requirements, this thesis proposed a new approach to context middleware, called M-hub/CDDL. The proposed solution presents mechanisms to support management and context quality that are broader and more efficient than the related approaches in several aspects, mainly in relation to management sensors, provisioning, evaluation, and monitoring of QoC. The quantitative evaluation of middleware has proven the efficiency of the M-Hub/CDDL in relation to data delivery time, reliability in unstable networks, monitoring time and resource consumption (memory and battery). The proof of concept has shown that the M-Hub/CDDL mechanisms apply to several application domains, while the results of the case study with users showed a high degree of satisfaction with the programming abstractions, services and QoC support provided by middleware. Therefore, the results show the efficiency of the M-Hub/CDDL as a facilitating approach to the IoT applications development with QoC requirements, showing that the objectives were achieved.

Keywords: Middleware, Quality of Context (QoC), Internet of Things (IoT).

Agradecimentos

Em primeiro lugar, a Deus, pelo dom da vida. Ele conhece bem as minhas limitações e me permitiu superá-las em diversos momentos da minha trajetória até aqui.

Ao meu orientador, Prof. Francisco Silva, pela confiança, disponibilidade e paciência.

Ao coordenador, colegiado, professores e técnicos administrativos do Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA, pelo trabalho sério e dedicado, que possibilita as condições para o desenvolvimento de nossas pesquisas.

Aos membros da banca examinadora, por aceitarem a missão de avaliar este trabalho.

A FAPEMA, pelo fomento a esta pesquisa.

Aos pesquisadores do Laboratório de Sistemas Distribuídos Inteligentes (LSDi) da UFMA, por compartilharem comigo momentos bons e ruins nesta pós-graduação.

Aos pesquisadores do *Laboratory for Advanced Collaboration* da Pontifícia Universidade Católica do Rio de Janeiro (PUC), parceiros com os quais pude buscar ajuda e co-orientação, trocar experiências valiosas, desenvolver projetos e publicar artigos.

A Direção Geral do Instituto Federal do Maranhão / Campus São Luís - Maracanã, que mesmo diante da impossibilidade de conceder-me afastamento, sempre ajustou meus horários para que eu pudesse me dedicar ao doutorado em meio período.

A minha família e meus amigos, pelo apoio que me deram nesta fase tão importante. Se fosse apenas por mim talvez eu até já tivesse desistido mas quando penso nessas pessoas e no quanto elas me ajudaram compreendo que desistir seria desapontá-las.

"Aquele que luta contra nós fortalece nossos nervos e aprimora nossas qualidades. Nosso antagonista trabalha por nós."

Edmund Burke

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AAL *Ambient Assisted Living*
ACM *Association for Computing Machinery*
ADSL *Asymmetric Digital Subscriber Line*
API *Application Program Interface*

BLE *Bluetooth Low Energy*
BN *Base Node*
BT *Bluetooth*

CAPES Coordenação de Aperfeicoamento de Pessoal de Nível Superior
CDDI *Context Data Distribution Infrastructure*
CDDL *Context Data Distribution Layer*
CDDLA *Context Data Distribution Level Agreement*
CEP *Complex Event Processing*
CN *Central Node*
CQL *Continuous Query Language*
CUN *Coordinator User Node*

DDS *Data Distribution Service*
DMS *Decision make Service*
DSL *Domain-Specific Language*

ECA *Event-Condition-Action*
ECG Eletrocardiograma
EDA *Event Driven Architecture*
EPA *Event Processing Agent*
EPL *Event Processing Language*

EPN *Event Processing Network*

GPS *Global Positioning System*
GSM *Global System for Mobile communication*

HARS *Human Activity Recognition Service*
HTTL *Horizontal Time to Live*
HUUFMA Hospital Universitário da UFMA

IBM *International Business Machines Corporation*
IDE *Integrated Development Environment*
IEEE *Institute of Electrical and Electronics Engineers*
IMS *Intensity Measurement Service*
IoMT *Internet of Mobile Things*
IoT *Internet of Things*
IP *Internet Protocol*

LAC *Laboratory for Advanced Collaboration*
LSDi Laboratório de Sistemas Distribuídos Inteligentes
LWT *Last Will and Testament*

M-Hub Mobile Hub
MHARS *Mobile Human Activity Recognition System*
MQTT *Message Queue Telemetry Transport*
MR-UDP *Mobile Reliable User Datagram Protocol*
MVMS *Mobile Vehicle Monitoring System*

NAT *Network Address Translation*
NFC *Near-Field Communication*

OO Orientação a Objetos

P2P *Peer to Peer*

PoA *Point of Attachment*

PPGCC Programa de Pós-Graduação em Ciência da Computação

PPGEE Programa de Pós-Graduação em Engenharia de Eletricidade

PUC Pontifícia Universidade Católica

QoC Quality of Context

QoD Quality of Device

QoI Quality of Information

QoS Quality of Service

QP *Query Priority*

QTL *Query Total Lifetime*

RAM *Random-Access Memory*

RDF *Resource Description Framework*

RDQL *RDF Data Query Language*

RFID *Radio-Frequency Ifentification*

RTPS *Real-Time Publish-Subscribe*

S2PA *Short-range Sensing, Presence & Actuation Service*

SAS *Situation-Aware Service*

SDDL *Scalable Data Distribution Layer*

SLA *Service Level Agreement*

SLP *Service Location Protocol*

SQL *Structured Query Language*

SSL *Secure Sockets Layer*

SUN *Simple User Node*

TCP *Transmission Control Protocol*

TLS *Transport Layer Security*

TTL *Time to Live*

UDP *User Datagram Protocol*

UFMA Universidade Federal do Maranhão

UPnP *Universal Plug and Play*

URI *Uniform Resource Identifier*

UTF *Unicode Transformation Format*

VTC *Vehicle Tracking Center*

WLAN *Wireless Local Area Network*

WPAN *Wireless Personal Area Network*

WSN *Wireless Sensor Network*

XML *eXtensible Markup Language*

1 Introdução

Este capítulo introdutório aborda o contexto em que esta pesquisa de doutorado está inserida, a caracterização do problema, a justificativa, os objetivos, a hipótese, a metodologia e as contribuições do trabalho. Ao final, mostra-se como os demais capítulos da tese estão organizados.

1.1 Contextualização

Sistemas sensíveis ao contexto são aqueles capazes de coletar, extrair e utilizar informações de contexto com a finalidade de adaptar o seu comportamento e executar ações em resposta às mudanças do ambiente, sem que intervenções sucessivas dos usuários sejam necessárias [7, 52, 95]. Em outra definição, um sistema é ciente de contexto se ele usa informações de contexto para prover serviços relevantes aos usuários, sendo que essa relevância vai depender das tarefas que eles realizam [2, 33].

Os avanços nas áreas de comunicação móvel, localização via sistema de posicionamento global (*Global Positioning System* - GPS) e tecnologias de redes de sensores sem-fio (*Wireless Sensor Networks* - WSN) são algumas das forças motrizes que empurraram a computação ciente de contexto em direção a sistemas ubíquos, motivando o desenvolvimento de novos serviços e aplicações. Enquanto a computação distribuída móvel oferece suporte ao acesso remoto de informações de contexto e aplicações adaptativas, a computação ubíqua estende este conceito para fornecer recursos de computação e comunicação que são tão discretamente integrados com os ambientes que chegam a parecer "invisíveis". Embora os usuários estejam conscientes das funcionalidades providas, eles não precisam ter conhecimento dos mecanismos subjacentes que as fornecem [25, 94].

Aplicações e serviços emergentes passaram então a reagir às mudanças de contexto do mundo físico, que são coletadas por sensores de baixo nível e distribuídas por meio de redes às quais essas aplicações também estão conectadas. Pode-se dizer

então que a computação ciente de contexto está sendo realizada na perspectiva de um novo paradigma, conhecido como a Internet das Coisas (*Internet of Things* - IoT).

A IoT é um campo de pesquisa que integra aspectos e tecnologias de diferentes áreas, tais como: computação ubíqua, ciência de contexto, protocolos e tecnologias de comunicação, redes e dispositivos com sensores embutidos. Essa integração visa a geração de sistemas nos quais milhares de dispositivos endereçáveis e heterogêneos, conhecidos como objetos inteligentes (*smart objects*), possuem uma representação na Internet e são capazes de compartilhar dados de contexto entre si e com aplicações diversas [13]. Às vezes, esses objetos tem recursos de processamento e conectividade tão limitados que precisam de um servidor local com acesso à Internet, conhecido como gateway de IoT, para que possam enviar dados para as aplicações remotas. Nessa perspectiva, a IoT funde os mundos físico e virtual em uma mesma rede. Grandes empresas têm investido no mercado de IoT[1], colocando a área em evidência. Estima-se que, por volta de 2020, mais de 20 bilhões de objetos inteligentes estejam conectados à rede [2].

Talavera et. al. [91] recentemente adotaram o termo Internet das Coisas Móveis (*Internet of Mobile Things* - IoMT) para designar uma extensão da IoT que envolve cenários nos quais gateways e objetos inteligentes possuem mobilidade irrestrita. Desse modo, é possível o desenvolvimento de aplicações móveis que descobrem e interagem com objetos inteligentes em todos os espaços, de maneira oportunista e ubíqua. Exemplos de objetos móveis incluem dispositivos vestíveis ou portáteis, robôs e veículos com sensores embutidos.

Conceitos, protocolos e padrões da IoT têm inspirado o desenvolvimento de aplicações e serviços em diversas áreas, tais como transporte, comércio, indústria e saúde. Especificamente na área da saúde, o termo *Ambient Assisted Living* (AAL) designa um campo de pesquisa multidisciplinar que emprega esforços no desenvolvimento de sistemas de monitoramento remoto de pacientes em ambientes inteligentes (e.g., *Smart Homes* e *Smart Cities*) [78]. Exemplos de objetos que fornecem dados de contexto em AAL incluem sensores vestíveis/portáteis, que coletam dados de sinais vitais (e.g., ECG, frequência cardíaca e volume de oxigênio consumido) e dados de movimentação do paciente (e.g., acelerômetro, giroscópio e gravidade), além

[1] http://www.computerworlduk.com/galleries/data/12-most-powerful-internet-of-things-
[2] http://www.gartner.com/newsroom/id/3165317

dos sensores embutidos no ambiente, que coletam dados que refletem as condições do local onde o paciente está (e.g., temperatura, luminosidade, umidade do ar e nível de gás carbônico). Aplicações de AAL despertam grande interesse da sociedade porque podem melhorar a qualidade de vida dos pacientes, especialmente aqueles que são idosos e/ou possuem algum tipo de doença crônica (e.g., arritmia, asma, diabetes).

Além dos requisitos convencionais de contexto, que expressam o tipo de informação que interessa para a cada aplicação consumidora, existem também requisitos de qualidade de contexto (*Quality of Context* - QoC)[3]. Em uma definição tradicional e restrita, a QoC é um conceito que expressa um conjunto de parâmetros que caracteriza apenas a qualidade da informação (QoI) em si e não o dispositivo ou serviço que a provê. Exemplos de parâmetros de QoC são: acurácia, atualidade e resolução [17]. Entretanto, uma noção mais recente e abrangente considera que o conceito de QoC, além de parâmetros de qualidade da informação, deve integrar também parâmetros que expressam ou que configuram a qualidade do serviço (*Quality of Service* - QoS) de distribuição dos dados de contexto, como forma de melhor atender os requisitos das aplicações consumidoras. Exemplos de parâmetros de QoS são: confiabilidade de entrega, prazo e taxa de atualização [10]. QoI e QoS são interdependentes e, portanto, exercem influência uma sobre a outra. Por exemplo, a atualidade da informação é afetada pelo atraso na comunicação, ou seja, o tempo decorrido entre a medição do dado e sua efetiva entrega à aplicação. Em outro exemplo, o prazo de validade das informações determina o tempo pelo qual o serviço de distribuição manterá essas informações armazenadas em seu histórico.

A qualidade de contexto pode ser utilizada por aplicações de IoT/IoMT com vários objetivos. Por exemplo, as aplicações podem usar a QoC como um critério adicional de consulta e/ou de seleção dos provedores de serviços de contexto. Desse modo, atende-se não apenas aos requisitos de contexto da aplicação, mas também aos requisito de QoC. Em alguns casos, a QoC pode ser uma cláusula presente em Acordos de Nível de Serviço (*Service Level Agreements* - SLAs). No contrato, os produtores declaram os tipos de informação de contexto e o nível de QoC que podem prover. Já os consumidores especificam as informações de seu interesse e o nível de qualidade com a qual exigem recebê-las [17].

[3]Na literatura, quando se trata de QoI, o termo parâmetro pode significar atributo ou metadado. Em relação à QoS, esse termo pode significar atributo ou variável de configuração da qualidade do serviço.

Em relação à QoI, as aplicações de IoT podem fazer uso dos metadados de qualidade a fim de decidir se devem considerar ou não a utilização da informação fornecida pelos sensores. Por exemplo, sistemas de monitoramento de atividades de pacientes exigem que os dados coletados tenham um mínimo de acurácia para que consigam inferir corretamente a atividade desenvolvida. Se essa QoI é fornecida, a aplicação pode filtrar um conjunto de amostras cuja acurácia garanta uma boa taxa de acerto. Se a QoI não está disponível, o sistema terá que partir do pressuposto de que todas as informações recebidas estão corretas, o que aumenta a probabilidade de erros no reconhecimento das atividades. Outro parâmetro de QoI relevante é a atualidade da informação, visto que algumas tomadas de decisão ou ações no âmbito de sistemas voltados ao monitoramento de pacientes não podem ser tomadas com base em informações antigas e/ou que não refletem a condição atual do paciente. Por exemplo, para uma aplicação de resgate, a ação de enviar um ambulância até o local do acidente deve levar em consideração a localização mais recentemente fornecida pelo dispositivo do paciente. O controle da QoI também pode ser utilizado para garantir a privacidade dos usuários das aplicações de cientes de contexto. Por exemplo, no caso de aplicações de AAL, a acurácia dos dados de localização de pacientes monitorados pode ser reduzida de propósito (processo de ofuscação), especialmente em situações nas quais eles não desejam divulgar com exatidão o local em que se encontram [99].

Em relação à QoS, pode ser útil para diversos tipos de aplicações ter a capacidade de configurar o serviço de distribuição de acordo com a maneira como o consumidor deseja receber os dados. Por exemplo, alguns sensores de movimentação, como acelerômetro e giroscópio, dentre outros, são capazes de gerar um grande volume de dados em curtos intervalos de tempo. Entretanto, nem sempre a capacidade dos gateways IoT e aplicações consumidoras permitem processar esses dados com a mesma velocidade com que são produzidos, principalmente quando se trata do uso de dispositivos móveis que geralmente possuem recursos de memória e processamento mais limitados. Nesses casos, é desejável para aplicações de IoT/IoMT que o serviço de distribuição possa ser configurado de modo a enviar e receber dados de contexto em conformidade com a capacidade de processamento dos dispositivos. Outro requisito de qualidade de serviço importante para algumas aplicações é a confiabilidade de entrega. Por exemplo, se uma determinada aplicação de monitoramento de pacientes envia notificações de alerta aos profissionais de saúde em situações de emergência,

ao invés de adotar a política do melhor esforço (*best-effort*), o serviço de distribuição deve confirmar a entrega e retransmitir a mensagem em caso de perda. Do contrário, o paciente não será atendido, pois a situação dele não foi comunicada a quem deveria.

A qualidade de contexto tem impacto significativo no comportamento das aplicações cientes de contexto e na eficiência dos serviços oferecidos. A experiência dos usuários dessas aplicações pode ser positiva ou negativa, dependendo do nível de QoC disponível [82]. Portanto, atender de forma satisfatória os requisitos de QoC das aplicações de IoT/IoMT é um passo muito importante para garantir o bom funcionamento delas e a satisfação dos seus usuários .

1.2 Caracterização do Problema

O problema central desta pesquisa consiste no fato de que o desenvolvimento de mecanismos de gerenciamento e qualidade de contexto aumenta consideravelmente a complexidade da implementação de aplicações de IoT/IoMT. Essa complexidade advém de diversos desafios que os desenvolvedores precisam superar, tais como:

- **Heterogeneidade dos sensores, protocolos e tecnologias de comunicação**: A aquisição de dados de contexto em IoT exige que os gateways estejam habilitados a interagir com sensores. A complexidade está em implementar o suporte às várias tecnologias e protocolos de comunicação, de maneira a permitir que esses gateways consigam gerenciar uma grande heterogeneidade de sensores [93] [86]. Em cenários de IoMT, essa tarefa é ainda mais complexa, pois não se conhece todos os tipos de sensores que os gateways encontrarão ao longo do trajeto.

- **Variedade de parâmetros de QoC exigidos pelas aplicações consumidoras**: Para algumas aplicações, pode ser suficiente aquele tipo de provisionamento de QoC que se restringe a parâmetros de qualidade da informação. Para outras, basta o provisionamento de parâmetros de qualidade de serviço para satisfazer os requisitos do sistema. Entretanto, a evolução dos serviços cientes de contexto tem mostrado que é cada vez maior o número de aplicações cujas necessidades de qualidade abrangem tanto parâmetros de QoI quanto parâmetros de QoS.

Quanto maior a variedade de parâmetros exigidos, mais complexa se torna a tarefa do desenvolvedor de atender a todos os requisitos de QoC da aplicação.

- **Variações significativas na qualidade de contexto em tempo de execução**: A qualidade de contexto nem sempre se mantém estável. Diversos fatores podem degradar a QoC, tais como falhas nos sensores, ausência ou má calibração, falhas do gateway IoT ou da aplicação, problemas na rede de comunicação, etc. [51] [77]. O funcionamento da aplicação é afetado negativamente quando a QoC cai a ponto de não mais atender suas exigências mínimas de qualidade. O problema da oscilação da QoC aumenta a complexidade no desenvolvimento de aplicações de IoT pelo fato de que exige a implementação de mecanismos que permitam monitorar a qualidade de contexto dinamicamente e executar ações adaptativas quando variações significativas na QoC são detectadas.

- **Conectividade intermitente e troca de endereço IP em cenários de mobilidade**: A intermitência das conexões de rede, principalmente em ambientes móveis, é altamente prejudicial para aplicações de IoT que exigem confiabilidade na entrega. Mesmo a adoção de protocolos de transporte confiáveis como TCP, que promovem retransmissão em caso de detecção de perda, por si só, pode não ser suficiente para garantir a confiabilidade em cenários de mobilidade, sobretudo em situações onde as desconexões com a rede são de longa duração e/ou quando o dispositivo troca de endereço IP. Isso porque os *buffers* de mensagens, assim como as tentativas de retransmissão do protocolo TCP em caso de falhas, são limitados. Isso exige do programador esforço adicional com a implementação de mecanismos de confiabilidade na camada de aplicação [93] [86] [69].

- **Especificação e descoberta de serviços baseada em qualidade de contexto**: No âmbito da IoT, onde existem milhares de dispositivos conectados à rede, há uma grande quantidade de serviços de contexto. Em cenários de IoMT, a mobilidade dos gateways e dos objetos inteligentes faz com que a oferta de serviços seja muito dinâmica. Os mecanismos de descoberta de serviços tradicionais como *Universal Plug and Play* (UPnP), *Apache River* (conhecido como Jini) e *Service Location Protocol* (SLP) abordam a descoberta apenas em âmbito de rede local e não foram projetados com foco em ambientes distribuídos e altamente dinâmicos como IoT/IoMT. Isso exige o desenvolvimento de mecanismos de descoberta

contínua e oportunista de serviços. Considerando que as aplicações, além de requisitos de contexto, possuem requisitos de qualidade de contexto, é necessário que a descoberta de serviços de contexto leve em consideração também a QoC.

1.3 Justificativa

A abordagem tradicional para o desenvolvimento de aplicações de IoT, resumidamente, consiste em fazer com que a aplicação seja responsável por todo o processo de gerenciamento de contexto (aquisição, processamento, distribuição e armazenamento). Entretanto, em situações onde as aplicações precisam interagir com uma grande quantidade de sensores, tecnologias,protocolos de comunicação heterogêneos, essa abordagem torna-se ineficiente [86]. Uma alternativa em relação à abordagem centrada na aplicação é adotar uma abordagem de middleware. A função do middleware é implementar todo o processo de gerenciamento de contexto de forma transparente para usuários e aplicações. Para isso, o middleware deve oferecer ao desenvolvedor um conjunto de abstrações, interfaces, linguagens, componentes e serviços de alto nível, que serão (re)utilizados pela camada de aplicação [8] [90] [11].

Uma vantagem da utilização da abordagem de middleware de contexto é que ela tende a reduzir o tempo e o esforço do desenvolvimento, pois permite que o programador se concentre mais na implementação dos requisitos inerentes à lógica de negócio da aplicação, ao invés de se preocupar com toda a complexidade da implementação dos mecanismos de gerenciamento de contexto. Outra vantagem da utilização da abordagem de middleware é a separação de responsabilidades, ou seja, o código-fonte responsável pelo gerenciamento de contexto não está entrelaçado ao código-fonte responsável pela lógica de negócio, pois eles estão isolados em camadas distintas. Desse modo, ambos os tipos de código-fonte podem evoluir separadamente.

A investigação de soluções de middleware para reduzir a complexidade no desenvolvimento de aplicações de IoT e resolver outros problemas é uma linha de pesquisa ativa [8]. Quantidades significativas de soluções de middleware foram propostas na literatura. Cada tipo de solução geralmente concentra-se em diferentes aspectos, tais como gerenciamento de dispositivos, interoperabilidade, portabilidade de plataforma, ciência de contexto, qualidade de contexto, segurança, privacidade e

muitos outros. Ainda que algumas iniciativas tentem abordar vários desses aspectos, uma solução de middleware ideal, capaz de lidar com todas as questões, ainda não surgiu [86]. Li e Feng [64] consideram que as pesquisas sobre o provisionamento de qualidade em middleware de contexto, embora sejam relevantes para a construção de sistemas de alta qualidade, ainda são insuficientes para resolver todos os problemas.

De fato, embora haja muitas propostas de middleware voltadas para o desenvolvimento de aplicações cientes de contexto, poucas delas oferecem algum tipo de suporte à QoC. Desse modo, ainda que tenha soluções de middleware à sua disposição, ainda está a cargo do desenvolvedor lidar com diversos desafios relacionados ao cumprimento de requisitos de qualidade de contexto. A maior parte das soluções propostas na literatura, entre as que lidam com aspectos de QoC, está mais focada no provisionamento de parâmetros de QoI. Uma minoria delas está voltada ao provisionamento de QoS, de modo que pouco é provido em relação à confiabilidade de entrega dos dados de contexto em ambientes móveis. É perceptível também a carência de suporte de middleware ao monitoramento de QoC. Além disso, poucas abordagens de middleware possuem suporte nativo para gateways de IoT/IoMT, que permitam a interação oportunista com vários sensores heterogêneos.

Diante das lacunas deixadas pelas soluções propostas, esta pesquisa se justifica pela necessidade de continuar investigando uma solução de middleware eficiente para alguns problemas relacionados à qualidade de contexto, caracterizados na seção anterior. Isso permite gerar novas contribuições e avanços no estado da arte.

1.4 Hipóteses de Investigação

Esta tese parte de duas principais hipóteses de investigação. A primeira delas é a seguinte:

A complexidade no desenvolvimento de aplicações de IoT/IoMT, em grande parte, ainda é elevada devido à ausência de abordagens de middleware que atendam a todos requisitos de gerenciamento e qualidade de contexto dessas aplicações. Portanto, o desenvolvimento de novas abordagens de middleware mais eficientes, que atendam a todos os requisitos, ou pelo menos a maior parte deles, facilitará consideravelmente a tarefa do programador, que ao invés de gastar tempo e esforço com a implementação de mecanismos genéricos, que devem ser providos pelo middleware de contexto, estaria mais livre para se concentrar na implementação de requisitos específicos, inerentes à lógica de negócio das aplicações.

A segunda hipótese de investigação se resume à seguinte afirmação:

A qualidade de contexto, quando devidamente provida pelo middleware, não impacta negativamente no desempenho das aplicações cientes de contexto. Entretanto, quando provida em nível satisfatório, a QoC pode melhorar a performance dessas aplicações, impactando positivamente na qualidade de experiência dos usuários.

1.5 Objetivos

1.5.1 Objetivo Geral

O objetivo geral desta pesquisa de doutorado é facilitar o desenvolvimento de aplicações de IoT/IoMT, principalmente aquelas que exigem múltiplos parâmetros de QoC. Para isso, esta tese propõe a criação de uma nova abordagem de middleware, capaz de atender prontamente a vários requisitos de gerenciamento e qualidade de contexto das aplicações, diminuindo assim, consideravelmente, a complexidade do desenvolvimento e exigindo menos esforço por parte do programador da aplicação. Dentre os requisitos que deverão ser atendidos pela abordagem proposta estão: gerenciamento de sensores heterogêneos, comunicação distribuída, confiabilidade de entrega em redes instáveis, comunicação segura, descoberta de serviços de contexto baseada em QoC, provisionamento, avaliação e monitoramento de QoI e QoS. Além disso, objetiva-se que essa nova solução seja mais completa e eficiente que outras abordagens de middleware encontradas na literatura, em relação a esses requisitos.

1.5.2 Objetivos Específicos

Para atingir o objetivo geral, definiu-se os seguintes objetivos específicos:

- Identificar problemas no desenvolvimento de aplicações de IoT/IoMT, de modo a refletir sobre os desafios existentes e propor contribuições ao estado da arte.
- Analisar as principais abordagens de middleware com suporte à QoC existentes, a fim identificar suas limitações e definir requisitos para criação de uma proposta.
- Propor uma abordagem de middleware de contexto com suporte à QoC, levando em consideração requisitos que a tornem mais eficiente que outras da literatura.
- Implementar a abordagem de middleware proposta, visando uma plataforma de sistema operacional bem difundida, a fim de facilitar a sua utilização e avaliação.
- Avaliar a abordagem de middleware proposta quantitativa e qualitativamente, por meio da realização de experimentos e estudos de caso, respectivamente.

1.6 Metodologia

Para classificar os métodos de pesquisa utilizados nesta tese, adotou-se a classificação proposta por Wazlawick [108], voltada para a Ciência da Computação.

Quanto à natureza, trata-se de uma pesquisa voltada ao desenvolvimento de um **trabalho original**, ou seja, um trabalho que tem como objetivo apresentar conhecimento novo a partir de observações e teorias construídas para explicá-lo. Assume-se que o conhecimento gerado é relevante porque o trabalho tem implicação na forma como se entente os processos e sistemas de computação cientes de contexto.

É importante destacar que, mesmo que a abordagem de middleware proposta nesta tese possa ser concretizada por meio de um artefato de software, que no caso corresponde ao middleware de contexto propriamente dito, esta pesquisa não trata apenas da criação de uma ferramenta, o que seria uma contribuição meramente tecnológica, mas principalmente da proposição de um modelo conceitual e arquitetural que, uma vez comprovada a sua eficiência para desenvolvimento de aplicações de IoT/IoMT com requisitos de QoC, pode ser estendido e reimplementado por terceiros.

Quanto aos objetivos, trata-se de uma **pesquisa explicativa**, pois além de analisar problemas relacionados ao desenvolvimento de aplicações de IoT/IoMT com requisitos de QoC, ela procura causas e explicações para um problema em aberto.

Quanto aos procedimentos técnicos, este trabalho utilizou três métodos conhecidos: **pesquisa bibliográfica**, **pesquisa experimental** e **pesquisa de levantamento**. A seguir é explicado o motivo da utilização de cada método.

Em relação à pesquisa bibliográfica, esta foi realizada a fim de buscar fundamentação teórica, identificar problemas em aberto, analisar os avanços e limitações de outras propostas da literatura e refletir sobre possíveis contribuições ao estado da arte. Esse levantamento não foi feito apenas no início da pesquisa, mas se estendeu ao longo de toda sua execução. Foram levantados trabalhos publicados em periódicos especializados e anais de eventos, bem como capítulos de livros nas área de Computação Móvel, Computação Ciente de Contexto, Qualidade de Contexto e Internet das Coisas. Esses materiais foram obtidos a partir de consultas às principais bases científicas disponíveis na Web, tais como: *Science Direct, IEEEExplore, ACM Digital Library, Scopus, Web of Science e Google Scholar*.

Em relação à pesquisa experimental, foi realizado um conjunto de experimentos reproduzíveis (avaliação quantitativa), a fim de medir determinadas métricas de desempenho do middleware. A análise dos resultados obtidos permitiu fazer algumas generalizações e tirar conclusões sobre a eficiência da solução.

Em relação à pesquisa de levantamento, foram realizados dois estudos de caso envolvendo o desenvolvimento de aplicações cientes de contexto com múltiplos requisitos de QoC, a fim de avaliar o middleware qualitativamente. No primeiro, conclusões foram obtidas com base em observações pontuais sobre o uso do middleware. No segundo, foram obtidas conclusões sobre a eficiência do middleware proposto feitas com base na análise de dados coletados junto aos desenvolvedores participantes por meio da criação e aplicação de questionários de avaliação.

Apesar da realização de ambos os tipos de avaliação, os resultados obtidos com elas não podem ser comparados com os de outras abordagens. O motivo é que os processo de avaliação das abordagens encontradas na literatura ou são inexistentes ou não são facilmente reproduzíveis. No melhor do nosso conhecimento, não existe ainda

uma metodologia de avaliação formal para middleware de contexto, quantitativa ou qualitativa, que seja amplamente aceita.

Segundo Nazario [81], a argumentação e o convencimento verificam a consistência de uma técnica, modelo ou teoria de pesquisas em áreas emergentes e novas, onde os pesquisadores têm dificuldade em obter dados em quantidade suficiente para dar suporte empírico a suas conclusões. Sendo assim, além das análises dos resultados obtidos nas avaliações, a argumentação acerca da eficiência do middleware proposto é fortalecida por uma análise comparativa que levou em consideração as principais características de todas as abordagens de middleware com suporte à QoC propostas na literatura, o que permitiu avaliar qual delas é capaz de atender de forma satisfatória a maior quantidade de requisitos relevantes para o desenvolvimento de aplicações de IoT/IoMT com múltiplos requisitos de QoC.

Quanto à metodologia de desenvolvimento do middleware proposto, foram adotados os princípios dos chamados métodos ágeis de desenvolvimento de software [23] que preveem a construção de sistemas de forma incremental, iterativa, adaptativa e em curtos períodos de tempo. O processo de desenvolvimento é composto de etapas de levantamento de requisitos, projeto, implementação e testes.

1.7 Contribuições

Considera-se que as principais contribuições desta pesquisa são:

1. Uma definição de QoC mais abrangente que as apresentas na literatura, que leva em consideração não apenas a qualidade da informação, mas também a qualidade de serviço e a qualidade de dispositivo.

2. Uma ampla comparação entre diferentes abordagens de middleware de contexto com suporte a QoC, pois, de modo geral, as revisões de literatura existentes, embora analisem diversos aspectos, não discutem detalhadamente a forma como cada middleware lida com problemas relacionados ao provisionamento de QoC.

3. A proposição de uma abordagem de middleware de inovadora, que incorpora conceitos e mecanismos de provisionamento, avaliação e monitoramento de

qualidade de contexto, capaz de facilitar significativamente o desenvolvimento de aplicações de IoT que possuem requisitos de QoC.

4. Uma metodologia de avaliação de middleware de contexto com suporte à QoC, que leva em consideração tanto aspectos quantitativos (que incluem a realização de experimentos) e qualitativos (que incluem provam de conceito e avaliação junto ao desenvolvedores de aplicações de IoT).

1.8 Organização da Tese

Os Capítulos 2, 3 e 4 apresentam um resumo da fundamentação teórica obtida nesta pesquisa, referente aos temas ciência de contexto e qualidade de contexto, e bases tecnológicas utilizadas para o desenvolvimento da solução, respectivamente. O Capítulo 5 apresenta a abordagem de middleware proposta. Essa apresentação contempla os requisitos da solução, a arquitetura, as funções dos principais componentes e aspectos de implementação. Os Capítulos 6 e 7 descrevem as avaliações qualitativa e quantitativa, respectivamente, da abordagem de middleware proposta e também as análises e discussões dos resultados obtidos. O Capítulo 8 apresenta um levantamento e descrição das principais abordagens de middleware com suporte à QoC existentes na literatura, bem como uma análise comparativa entre essas abordagens e a que está sendo proposta neste trabalho. O Capítulo 9 apresenta as conclusões desta pesquisa e os trabalhos futuros.

2 Ciência de Contexto

Este capítulo contém uma fundamentação teórica sobre Ciência de Contexto. Ele apresenta algumas definições, características, classificações, modelos de representação, processos de gerenciamento e modelos de comunicação entre produtores e consumidores de contexto. Algumas considerações são apresentadas no final do capítulo.

2.1 Definições de Contexto

A computação ubíqua visa tornar mais agradável a experiência do usuário na utilização de recursos computacionais. Para Weiser [109], um dos primeiros a definir o termo "Computação Ubíqua", as tecnologias mais profundas e duradouras são as que parecem ser "invisíveis", ou seja, que se integram harmoniosamente ao ambiente a ponto de tornarem-se indistinguíveis de outros objetos.

Em cenários da computação ubíqua, o foco dos usuários seria a tarefa e não a ferramenta utilizada. Além disso, eles sequer perceberiam ou necessitariam de conhecimentos técnicos sobre os recursos computacionais usados. Então, para que esse nível de invisibilidade seja atingido, é fundamental que as aplicações sejam capazes de obter informações do ambiente no qual o usuário se encontra a fim de prover a ele serviços úteis. Portanto, as aplicações ubíquas requerem **ciência de contexto**, também chamada de **sensibilidade ao contexto**.

Para Byun e Cheverst [18], um sistema é sensível ao contexto se este for capaz de extrair, interpretar e utilizar informações de contexto para adaptar suas funcionalidades.

Diversas aplicações ubíquas tiram proveito das informações de contexto para dirigir suas ações, tais como redes sociais, sistemas de monitoramento de paciente em domicílio, aplicativos de orientação de rotas e/ou de sugestão de serviços baseados em localização e preferências dos usuários, dentre outras. Com o advento da

computação móvel, novas áreas estão cada vez mais se beneficiando da capacidade de ciência de contexto para ofertar novos serviços aos usuários.

A Ciência de Contexto refere-se à capacidade de um sistema perceber características dos usuários e do ambiente que sejam de seu interesse [95]. Portanto, as aplicações sensíveis ao contexto monitoram o estado do ambiente de execução no qual estão inseridas e tomam decisões de acordo com mudanças de contexto, reagindo a ações executadas pelas entidades presentes no ambiente (e.g., pessoas, objetos e outros sistemas), com pouca ou nenhuma necessidade de intervenção humana.

A definição formal de contexto ainda é uma tarefa em andamento. Vários autores o fazem de diferentes formas. Segundo Schilit et al. [96], o contexto indica onde o usuário está, com quem está e quais recursos estão disponíveis na vizinhança. Dey [33] considera contexto como qualquer informação que possa ser usada para caracterizar a situação de uma entidade (e.g., pessoas, objetos, o hardware, o sistema) que seja relevante para a interação entre o usuário e a aplicação. Bolchini et al. [12] afirmam que o contexto é formado por um conjunto de variáveis que podem ser de interesse de um agente e que influencia as ações dele.

Segundo Hong et al. [52], as características encontradas em sistemas e aplicações sensíveis ao contexto são:

- **Autonomia:** o sistema pode executar ações sem requerer interações com o usuário. Para isso é necessário que a aplicação ciente de contexto conheça o contexto do usuário e suas preferências, que podem ser customizadas. Ações são executadas quando uma determinada situação acontece.
- **Antecipação:** capacidade de prever situações futuras e antecipar a execução de ações que beneficiarão o usuário com base no histórico de dados de contexto e análise de padrões.
- **Integração:** capacidade que os dispositivos do ambiente têm de reconhecer uns aos outros quando estão próximos e, a partir disso, estabelecerem conexões visando prestar serviços de maneira integrada. Nesse caso, a distância é um fator fundamental para que os dispositivos possam interagir.
- **Mobilidade:** capacidade do sistema de prover serviços de forma transparente e conveniente aos usuários em movimento.

2.2 Classificação de Contexto

Para Zimmermann et al. [113], os elementos que servem para a descrição da informação de contexto se enquadram em cinco categorias: individualidade, atividade, localização, tempo e relações. Schilit et. al [95] propuseram uma classificação de contexto com três categorias: usuário, computacional, físico. Analisando essas categorias, Chen e Kotz [22] adicionaram uma quarta categoria: o contexto temporal. Essas categorias são discutidas a seguir.

O **Contexto Computacional** engloba aspectos técnicos relacionados às capacidades dos recursos computacionais, tais como memória, processamento, níveis de energia e redes disponíveis. Muitos sistemas utilizam estas informações para se adaptar ao ambiente. Por exemplo, aplicações de transmissão de vídeo na web como Netflix [76] se adaptam à largura de banda disponível alterando a qualidade da transmissão de acordo com a capacidade da conexão a cada instante. Em outro exemplo, quando o nível de bateria está baixo, os dispositivos pessoais móveis reduzem a luminosidade do visor ou desativam as interfaces de rede ou sensores que não estão em uso no momento. Em sistemas de *Ambiente Assisted Living* (AAL) que usam dispositivos móveis, por exemplo, otimizar o consumo de bateria é um requisito importante para que o monitoramento do paciente dure o máximo possível.

O **Contexto Físico** aborda os aspectos que representam o mundo real e que são acessíveis por sensores ou recursos dos equipamentos ao redor. Dados como localização, velocidade, nível de ruído, temperatura e outros são exemplos de informações de contexto físico. Devido à sua natureza, as informações de contexto físico são muito suscetíveis a erros devidos a falhas e imprecisões nos processos de coleta dos dados. Como exemplo, podemos citar aplicativos de auxílio à navegação que utilizam informações de localização para orientar motoristas a chegarem ao seu destino. Em sistemas de AAL, dados como temperatura, umidade do ar e níveis de gás carbônico podem ajudar a inferir se as condições ambientais são favoráveis a permanência do paciente naquele ambiente. Sinais vitais (ECG, batimento cardíaco, frequência respiratória) ajudam aplicações de apoio ao diagnóstico do paciente. Já a localização do paciente pode ser utilizada para guiar as equipes de pronto-atendimento em caso de emergência.

O **Contexto Temporal** captura a dimensão temporal dos eventos que ocorrem no ambiente. Tais informações podem ser, por exemplo, o dia da semana ou a hora em que determinada atividade acontece. Podemos dividir as informações de contexto temporais em duas categorias: esporádicas e periódicas. Eventos esporádicos são aqueles que ocorrem ocasionalmente ou ao menos uma vez. Por exemplo, quando a largura de banda de uma conexão diminui, um evento esporádico pode ser detectado e disparar uma adaptação que reduz a qualidade de um vídeo que está sendo transmitido. Eventos periódicos são os que ocorrem de forma previsível e repetidamente. Como exemplo, podemos citar um sistema que reduz o volume de toque do celular todos os dias, a partir das onze horas da noite. Em sistemas de AAL, podemos enxergar situações de emergência como eventos esporádicos. Por outro lado, o sistema pode disparar alarmes periódicos que solicitam ao paciente que tome sua medicação de acordo com os horários prescritos pelo médico.

O **Contexto do Usuário** compreende a dimensão social do mesmo, tais como perfil, pessoas ao redor, situação social e outras. Diversos sistemas de redes sociais utilizam informações do perfil do usuário para definir quais notícias estes devem receber. Sistemas podem utilizar as informações de contexto do usuário para, por exemplo, inferir que determinados usuários estão interessados no mesmo tipo de conteúdo por estarem na mesma área geográfica ou possuírem laços de amizade. Em sistemas de AAL, algumas informações do perfil do usuário, como por exemplo a lista de contatos, podem ser utilizadas para auxiliar o paciente em algumas situações. Por exemplo, se o sistema consegue detectar quedas, ele pode acessar a lista de contatos com o objetivo de enviar uma mensagem pedindo ajuda para aqueles que estiverem mais próximos do paciente.

Outra classificação mais recente foi proposta por Emmanouilidis et al. [37], que define cinco categorias de informações de contexto: usuário, sistema, ambiente, social e serviço. As categorias usuário e sistema equivalem às categorias usuário e computacional, respectivamente, da classificação anterior. A categoria ambiente equivale à junção entre o contexto físico e o contexto de tempo anteriormente descritos. Já as categorias que se distinguem se referem aos contexto social e contexto de serviço, discutidos a seguir.

O **Contexto Social** caracteriza as possíveis formas de relacionamento e de interações entre pessoas. O termo está relacionado ao ambiente social do usuário (e.g.

uma festa ou uma reunião) e a relação que pode ser estabelecida com outros usuários (e.g., colega de trabalho, amigos, parentes, vizinhos). Informações de contexto social podem ser extraídas por meio de redes sociais (e.g. lista de amigos em redes sociais) ou sensores (e.g. pessoas localizadas sempre próximas à casa do usuário podem ser seus vizinhos) [3]. Em cenários de AAL também temos relações sociais, tais como entre médico e paciente, entre pacientes que possuem a mesma enfermidade ou passam por tratamentos semelhantes, bem como aqueles que são atendidos pelo mesmo médico.

Por fim, o **Contexto de Serviço** representa o estado de serviços disponíveis para o usuário. O estado de um serviço pode ser, por exemplo, a qualidade da iluminação pública das vias (neste caso a iluminação pública é o serviço) ou o valor do pedágio (neste caso, a própria via é o serviço) ou a quantidade de horas de espera para ser atendido na clínica (nesse exemplo, a clínica é o serviço). O contexto pode levar em consideração ainda as restrições impostas para o uso dos serviços (e.g. uma consulta que só pode ser feita com hora marcada) ou a dependência entre serviços (e.g o serviços de monitoramento remoto de pacientes depende de provedores de Internet para transmitir os dados aos profissionais de saúde).

Schuster et al. [97] desenvolveram o conceito de Contexto Social Pervasivo. O contexto social pervasivo de um indivíduo é o conjunto de informações que surgem a partir de interações diretas e indiretas entre pessoas que carregam dispositivos móveis equipados com sensores e que estejam conectadas ao mesmo serviço de rede social.

Os autores ainda classificaram e diferenciaram várias formas em que o contexto social pervasivo pode ser utilizado com base nas *W5H Questions*, como visto abaixo:

- **Quem?** (***W**ho?*): expressa quem são os participantes envolvidos na produção e consumo das informações de contexto;
- **O que?** (***W**hat?*): diz respeito a qual tipo de contexto é utilizado ou se é importante para a aplicação;
- **Onde?** (***W**here*): relacionado à localização física na qual as relações sociais acontecem;
- **Quando?** (***W**hen?*): caracterização das interações entre usuários e as informações de contexto que eles produziram em uma perspectiva temporal;

- **Porquê?** (*Why?*): expressa o porquê de uma informação de contexto estar sendo usada. Essa motivação está relacionada ao objetivo de cada aplicação;
- **Como?** (*How?*): expressa como a informação de contexto pode influenciar ou comprometer o funcionamento das aplicações.

2.3 Modelagem de Contexto

Um **modelo de contexto** determina o conjunto de todos os tipos e entidades de contexto e as relações entre eles. A definição de um modelo de contexto é uma parte essencial da implementação de um sistema sensível ao contexto. Um modelo de contexto define os conceitos relevantes para o domínio de aplicação, bem como define os relacionamentos entre esses conceitos [31]. Por exemplo, conceitos como Paciente, Domicílio, Doença, Atividade e Histórico Médico são relevantes em determinadas aplicações de AAL.

Diversas abordagens para modelagem de contexto em sistemas computacionais ubíquos são propostas na literatura. Cada uma tem seus próprios custos de representação e processamento e limitações de expressividade.

Para Helf [50], uma abordagem para a representação de contexto abrangente deve ter as seguintes características:

- **Estruturação:** um modelo de representação bem estruturado facilita a filtragem e/ou extração das informações do contexto que são relevantes para a aplicação. Além disso, reduz a possibilidade de ambiguidade de atributos[1] por meio do uso de identificadores únicos.
- **Intercambialidade:** capacidade de promover o intercâmbio de informações de contexto entre diferentes componentes do sistema sensível ao contexto.
- **Composição/Decomposição:** capacidade de compor ou decompor as informações de contexto em diferentes atributos. Essa característica ajuda no processo de atualização da informação, pois ela possibilita alterar apenas partes da informação, ao invés de alterar a informação como um todo.

[1] O termo atributo, também conhecido como variável de classe, se refere a elementos que definem a estrutura de uma classe

- **Extensibilidade:** capacidade de adicionar ou remover atributos a fim de atender necessidades futuras de alteração no modelo existente.

- **Padronização**: capacidade de representar as informações utilizando um formato padrão que possa ser compreendido por diferentes entidades dos sistemas sensíveis ao contexto.

De acordo com Strang e Linnhoff-Popien [101], as abordagens para modelagem de contexto mais utilizadas atualmente são: par chave-valor, esquema de marcação, orientado a objetos, lógicos e baseados em ontologias.

A abordagem **Par Chave-valor** é considerada a mais simples. Ela utiliza tuplas (chave, valor), ou seja, pares compostos por uma chave, que identifica o atributo de contexto, e por um valor associado a essa chave, sendo que o valor não pode ser outra tupla[2]. Por exemplo, o contexto de localização de um paciente poderia ser representando pelo seguinte conjunto de pares: (latitude = -21,978887), (longitude = -43,234207), (altitude = 15,434323). Essa abordagem é bastante limitada, pois não permite a organização da informação em níveis hierárquicos e não fornece mecanismos para validação dos dados e verificação de ambiguidades. Essas limitações impedem a utilização de algoritmos eficientes de recuperação das informações de contexto [79]. Geralmente a recuperação da informação de contexto nesta abordagem é feita utilizando o casamento de nomes (*string matching*).

O **Esquema de Marcação** é uma abordagem que permite a representação de contexto em uma estrutura hierárquica, na qual a especificação das informações é textual e delimitada por rótulos (*tags*) de marcação. Nessa abordagem, a linguagem XML (*eXtended Markup Language*) é usada como padrão para a modelagem das informações de contexto e permite definir um vocabulário comum e extensível. Isso facilita o compartilhamento das informações de contexto por diferentes aplicações [107]. Uma desvantagem é que essa abordagem não consegue solucionar ambiguidades. Apesar de sua maior representatividade em relação ao esquema anterior, faltam à esta abordagem capacidades importantes para descrever eventos complexos como, por exemplo, herança.

[2]Não se pode confundir o modelo par chave-valor com estruturas de dados mais complexas como *Hash Map*, disponibilizada pela linguagem Java, por exemplo. Estruturas como a *Hash Map* permitem simular uma tupla na qual o valor (um objeto) pode ser outra tupla.

A abordagem **Orientada a Objetos** tira proveito das vantagens do paradigma da Orientação a Objetos (Modelo OO), tais como herança, encapsulamento e reuso. As informações de contexto são representadas através do uso de classes e atributos. Esta abordagem, ao contrário das anteriores, possui a capacidade de validação dos dados como uma de suas características principais. Por exemplo, uma classe do tipo Pessoa, que possui atributos como nome e CPF, pode implementar, a partir de mecanismos de encapsulamento, métodos que lançam uma exceção caso a aplicação tente inserir um número de CPF com menos de 11 caracteres. Entretanto, esse modelo não explora a descrição semântica das informações contextuais e, por isso, há uma incapacidade de implementação de algoritmos que realizem a inferência e interpretação de novos contextos a partir apenas da descrição destas informações [87]. Nesta tese, a abordagem orientada a objetos empregada na modelagem das mensagens, incluindo àquelas referentes aos dados de contexto publicadas pelo middleware.

Em abordagens baseadas na **Lógica** utiliza-se a lógica formal para definir as condições em que uma expressão de conclusão ou fato pode ser derivada a partir de um conjunto de outras expressões ou fatos em um processo conhecido como raciocínio ou inferência. Para descrever essas condições em um conjunto de regras, um sistema formal é aplicado, onde o contexto é definido como fatos, expressões e regras. Os fatos podem ser adicionados manualmente na base ou inferidos a partir de regras. Uma desvantagem dessa abordagem é a dificuldade para validar dados.

Já a abordagem baseada em **Ontologia** tira proveito da capacidade de expressar relações mais complexas. Nessa abordagem é possível validar os dados por meio da imposição de restrições da ontologia [10]. Ontologias conseguem agregar conceitos e fatos. Com o uso de seus padrões é mais fácil promover o reuso e compartilhamento das informações de contexto, bem como eliminar ambiguidades [79]. Sistemas baseados em ontologias geralmente requerem maior capacidade de armazenamento e processamento. Como dito, embora o modelo de contexto do middleware proposto siga a abordagem orientada a objetos, a abordagem baseada em ontologias poderá ser utilizada em trabalhos futuros, a fim de aumentar a expressividade do referido modelo.

2.4 Processo de Gerenciamento de Contexto

Não há na literatura uma uniformidade em relação a quais fases devem compor o processo de gerenciamento de contexto. Existem algumas propostas, que variam com relação à quantidade de fases consideradas essenciais.

Por exemplo, Bellavista et al. [10] definem o gerenciamento de contexto como um processo composto por quatro fases: produção, processamento, distribuição e armazenamento.

Na fase de **produção**, as informações de contexto são coletadas do ambiente por meio de diversos processos e fontes. Sensores podem fornecer dados de contextos físicos como temperatura ou nível de ruído. Informações de contexto temporais podem ser obtidas a partir de relógios ou temporizadores (tanto externos como internos em relação ao sistema). Informações de contexto do usuário podem ser coletadas em perfis ou por meio de associações com outros sistemas. Os sistemas sensíveis ao contexto devem viabilizar formas de adquirir o contexto do usuário de forma mais automática possível.

Na fase de **processamento**, os dados de contexto podem ser submetidos a diferentes tipos de operações como filtragens, agregações e inferências. Cada uma dessas operações tem sua própria utilidade para o sistema ciente de contexto. Filtragem é uma operação que visa reduzir o volume de dados a serem enviados ou recebidos pela aplicação. Desse modo, a filtragem permite que apenas dados de interesse da aplicação sejam tratados, ao invés de gastar recursos computacionais com processamentos desnecessários. Agregação é o processo por meio do qual um sistema produz novas informações de contexto a partir de informações já coletadas de outras fontes. Já a inferência pode ser utilizada para detectar eventos ou situações (contexto derivado ou contexto de alto nível) a partir da análise de dados de sensores (contexto de baixo nível) e outras informações do usuário.

Na fase de **distribuição**, os produtores enviam as informações de contexto para os consumidores interessados. Em alguns casos, a distribuição de dados de contexto exige interação entre produtores e consumidores que executam em um rede local (distribuição local). Em outro, a distribuição requer a interação entre produtores e consumidores que executam em redes diferentes (distribuição remota).

Na fase de **armazenamento**, a informação de contexto é guardada em um histórico temporário ou permanente. O tamanho do histórico depende da capacidade dos recursos de armazenamento. É necessário que haja mecanismos que permitam que os consumidores de contexto consultem o histórico. Considerando que recursos computacionais como memória e disco são limitados, por vezes torna-se necessário que o serviço de gerenciamento de contexto mantenha no histórico somente os dados mais recentes, adotando mecanismos que removem do histórico dados cujos prazos de validade expiraram.

Perera et al. [86] promovem uma análise dos processos de gerenciamento de contexto mais recorrentes na literatura, e com base nessa análise propõem um processo, semelhante ao anterior, que contém as seguintes fases: aquisição, modelagem, raciocínio e disseminação. Nessa proposta, em que o processamento ocorre em duas fases (modelagem e raciocínio), o armazenamento não é considerado uma fase essencial. Já as fases de aquisição e disseminação correspondem respectivamente às fases de produção e distribuição da proposta anterior.

2.5 Modelos de Interação

Existem vários modelos de interação entre produtores e consumidores de contexto que tornam possível a distribuição de dados. Geralmente os modelos se diferenciam em relação ao acoplamento e à sincronia. Os principais modelos empregados são: passagem de mensagem, filas, publicação/subscrição e espaço de tuplas.

O modelos de interação baseado em passagem de mensagens tem a desvantagem de forte acoplamento entre os participantes. Neste modelo, é necessário que o emissor tenha conhecimento da identidade exata e do endereço do receptor. Além disso, existe a necessidade de sincronização, isto é, o emissor precisa esperar até que o receptor esteja pronto para trocar informações. Em sistemas sensíveis ao contexto, esta característica é bastante restritiva pois em várias aplicações não é possível ou desejável que produtores e consumidores estejam ativos ao mesmo tempo (por exemplo, fóruns e redes sociais) [43].

O modelo baseado em filas permite a interação um-para-um.Nesse modelo, o produtor conhece o destinatário e envia mensagens para uma fila específica do consumidor. Portanto, este modelo deve ser usado somente quando cada mensagem deve ser processada por apenas um consumidor, independente de qual seja. A comunicação é assíncrona, ou seja, o modelo não exige que o produtor esteja em execução no momento em que o consumidor lê a mensagem, assim como não é necessário que o consumidor esteja em execução no momento que o produtor envia a mensagem. Quando a mensagem é lida com sucesso, o consumidor envia um aviso de confirmação para o produtor [80].

O modelo publicação/assinante permite a interação um-para-muitos assíncrona. Nesse modelo, cada participante do sistema pode desempenhar o papel de publicador ou assinante (também conhecido como subscritor) de informações. Os publicadores produzem informações na forma de eventos que são consumidos pelos assinantes. As assinaturas são usadas para filtrar parte dos eventos produzidos pelo publicadores. A filtragem dos dados pode ser implementada no assunto (tópico) ou no conteúdo dos eventos. A principal característica do modelo publicação/assinatura é que publicadores e assinantes não acessam diretamente um ao outro [30]. O modelo publicação/assinatura é considerado mais adequado para distribuição de dados de contexto em ambientes móveis e cenários de larga escala [45], podendo ser implementado de forma descentralizada, por meio de espaços globais de dados (barramentos de evento) [85], ou de forma centralizada, utilizando servidores de distribuição (*brokers*) [54].

O modelo baseado em espaço de tuplas também é fracamente acoplado, assíncrono e permite a comunicação muitos-para-muitos. Sintaticamente, uma tupla é uma lista de valores separados por vírgulas. Um espaço de tuplas é uma região de memória compartilhada globalmente por todas as entidades *hosts* participantes. Nesse modelo, os produtores inserem as tuplas no espaço compartilhado enquanto que os consumidores leem as tuplas neste espaço [45]. Em termos de implementação, um espaço de tuplas é uma memória distribuída compartilhada que pode ser usada na comunicação entre entre as aplicações, atuando como um repositório de estruturas de dados. As aplicações consumidoras só terão conhecimento da existência de novas mensagens caso façam uma nova busca no espaço de tuplas. A busca de conteúdo geralmente é feita por meio da especificação de propriedades desejadas na tupla a ser

buscada. Portanto, o espaço de tuplas deve ser usado em cenários onde a troca de mensagens não exige uma notificação de que novas mensagens estão disponíveis.

Nesta tese, a abordagem de middleware proposta adota o modelo de interação publicação/assinante. Um dos motivos que justifica essa escolha é que o referido middleware poderá ser empregado no desenvolvimento de aplicações de IoT voltadas para ambientes bastante dinâmicos, onde a a interação entre produtores e consumidores de contexto tende a ser de muitos-para-muitos. Outro motivo, é que abordagem de middleware proposta visa apoiar o desenvolvimento de aplicações para cenários com larga escala de produtores e consumidores de contexto. Além disso, a abordagem é voltada para cenários onde os consumidores de contexto precisam ser notificados da publicação de novas mensagens, outra característica inerente ao modelo publicação/assinante.

2.6 Conclusão do Capítulo

Esse capítulo apresentou uma fundamentação teórica sobre ciência de contexto. Os pontos abordados foram: definição, classificação, modelagem, processo de gerenciamento e modelos de interação entre produtores e consumidores de contexto

Em relação às definições e classificações, cada autor tem sua própria concepção sobre o que é contexto e o categoriza de diferentes maneiras. Esta pesquisa considera que contexto pode ser definido como "qualquer informação que descreve usuários, sistemas, recursos, tempo, espaço e condições do ambiente". Uma vez que este trabalho é focado no desenvolvimento de middleware para a Internet das Coisas, o tipo de contexto mais relevante para a solução proposta é o contexto físico, visto que ele pode ser coletado por sensores. A mobilidade é uma das principais características das aplicações cientes de contexto. Assim como conveniente, a mobilidade também traz alguns problemas ao desenvolvimento de aplicações para a IoT, tais como a baixa confiabilidade de entrega dos dados em ambientes móveis devido à intermitência de conectividade nesses ambientes. Em IoMT, a mobilidade irrestrita de gateways e objetos inteligentes pode fazer a qualidade de contexto variar bastante visto que os sensores descobertos pelo gateway IoT ao longo do trajeto podem fornecer parâmetros e níveis de QoC diferentes.

Mostrou-se também que existem diversas abordagens para modelagem de contexto: par chave-valor, esquema de marcação, orientado a objetos, lógicos e baseados em ontologias. Cada abordagem tem limitações em relação a expressividade, validação, verificação de ambiguidades e custo de processamento. Portanto, ainda não há um modelo de representação único e completo que possa ser aplicado a todos os tipos de sistemas cientes de contexto.

No que diz respeito ao processo de gerenciamento de contexto, foi mostrado que os autores propõem processos com diferentes fases. Um dos processos de gerenciamento mais conhecido é aquele composto por fases de aquisição, processamento e distribuição dos dados de contexto. Outro é aquele no qual as fases são: aquisição, modelagem, inferência e disseminação. No geral, as propostas de processo de gerenciamento de contexto não evidenciam tanto a etapa de descoberta de serviços. Entretanto, no âmbito desta tese, a descoberta de serviços é um requisito fundamental para a distribuição de dados de contexto.

Por fim, foi visto que existem diferentes modelos de interação entre produtores e consumidores, tais como: passagem de mensagem, filas, publicação/assinatura e espaço de tuplas. Para a IoT/IoMT, onde o potencial de produtores e consumidores que podem estar conectados à rede é muito grande e variável, torna-se necessário um modelo de interação que permita baixo acoplamento entre os participantes e comunicação assíncrona. Em razão disso, modelos assíncronos como filas, espaços de tuplas e principalmente publicação/assinatura se mostram mais adequados para a distribuição de dados de contexto que modelos síncronos como passagem de mensagem.

3 Qualidade de Contexto

Este capítulo contém uma fundamentação teórica sobre Qualidade de Contexto. Ele apresenta algumas definições, parâmetros e motivações para o uso da QoC. Algumas considerações são apresentadas no final do capítulo.

3.1 Definições de Qualidade de Contexto

Há diversas definições de QoC, mas não existe um consenso na literatura acerca de qual é a mais correta. Algumas das definições propostas por alguns autores são apresentadas a seguir.

Buchholz et al. [17] definem QoC como sendo qualquer informação que descreve a qualidade da informação que é usada como informação de contexto. Nessa definição, QoC refere-se à qualidade inerente à informação (QoI) e não aos serviços ou componentes de hardware que a fornecem. Na visão desses autores, QoC é uma qualidade intrínseca à informação e que difere de qualidade de serviço (QoS) e de dispositivo (QoD). Nesse mesmo trabalho, a qualidade de serviço é definida como sendo qualquer informação que descreve o quão bem um serviço executa. Já a qualidade de dispositivo é definida como sendo qualquer informação que descreve as características técnicas, configurações e as capacidades do hardware.

Ainda que distintas, os autores afirmam que QoC, QoS e QoD possuem uma relação de interdependência, sendo que qualquer uma delas pode influenciar no comportamento ou ser influenciada pelas demais. A Figura 3.1, ilustra a visão Buchholz et al. em relação à interdependência de seus conceitos de qualidade de contexto, qualidade de serviço e qualidade de dispositivo. Nessa visão, cada tipo de qualidade é uma camada. A QoC corresponde à camada superior. A QoS corresponde à camada intermediária. A QoD corresponde à camada inferior. Para eles, existem duas formas pelas quais uma camada de qualidade pode influenciar as demais: a abordagem ascendente e a abordagem descendente. A seguir essas abordagens são explicadas e exemplificadas.

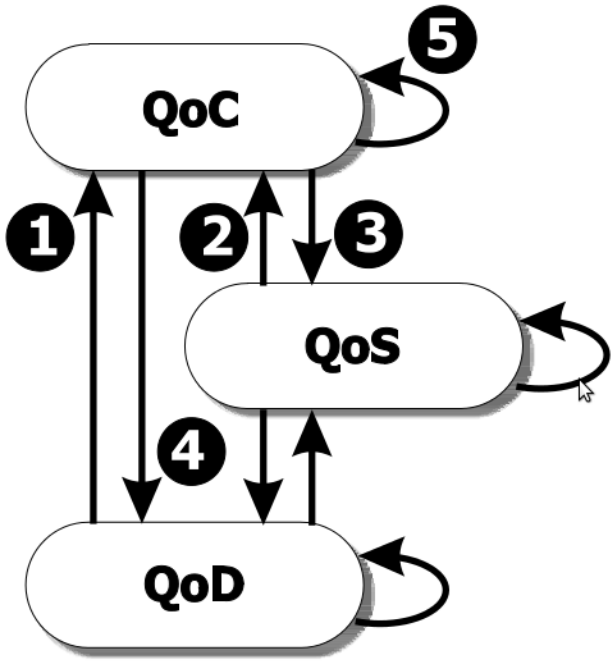

Figura 3.1: Interdependência entre QoC, QoS e QoD [17]

Na **abordagem ascendente** (*bottom-up approach*), mostrada na Figura 3.1 por meio das setas que apontam para cima, uma camada de qualidade tem impacto técnico direto sobre outra. Sendo assim, uma camada influência todas as camadas acima dela, ou seja, QoD influencia QoS e QoC, enquanto QoS influencia apenas a QoC.

A seguir são dados dois exemplos da abordagem ascendente:

- **QoD -> QoC**: a acurácia do serviço de localização varia de acordo com a técnica ou dispositivos empregados para determinar a posição (por exemplo, o GPS tem precisão de até 4 metros, dependendo do número de satélites, enquanto que a rede GSM tem precisão de até 500 metros, dependendo da concentração de estações radio base).

- **QoS -> QoC**: A informação de contexto envelhece durante a transmissão dos dados. Nesse caso, a idade e a validade são afetadas pelo atraso na entrega das informações ao consumidor. Quanto maior o atraso, mais velha será a informação.

Na **abordagem descendente** (*top down approach*), mostrada na Figura 3.1 por meio das setas que apontam para baixo, as noções de qualidade influenciam umas às outras por meio de requisitos que elas impõem. Nessa abordagem uma

camada influencia todas as camadas abaixo dela, ou seja, requisitos de QoC podem afetar requisitos de QoS e QoD, enquanto que requisitos de QoS podem afetar apenas requisitos de QoD.

A seguir são dados dois exemplos da abordagem descendente:

- **QoC -> QoD**: se o consumidor quer acurácia (QoC) de 4 metros, será forçado a escolher um serviço que disponibilize dados coletados a partir de um dispositivo com receptor GPS (QoD). Entretanto, às vezes o dispositivo pode estar sem sinal GPS. Nesses casos, a utilização da rede GSM pode ser necessária.
- **QoC -> QoS**: se o consumidor exige informações com a maior atualidade (QoC) possível, ele pode estabelecer um contrato de QoS com um provedor de serviços cujo atraso e confiabilidade de entrega sejam compatíveis com suas exigências de qualidade de contexto.

Krause e Hochstatter [57] adicionaram o conceito de valor (*worth*) para introduzir o ponto de vista (subjetividade) da aplicação consumidora. Para eles, existe uma distinção entre qualidade de contexto e valor. Enquanto a QoC é algo inerente à informação e pode ser estabelecida de maneira objetiva e independente de sua utilização, a estimativa do quão valiosa é a informação (o valor) é algo que, embora seja influenciada pela qualidade, vai além dela pois depende das características e necessidades do consumidor em relação à informação, sendo algo mais subjetivo. O consumidor deve estimar o valor que a uma informação tem para ele, não apenas com base na qualidade, mas também a partir de uma análise de seu custo/benefício. Considerando essa distinção, esses autores definiram QoC como sendo qualquer informação que descreve a qualidade da informação de contexto e que pode ser utilizada para determinar o valor que ela possui para uma aplicação específica.

Manzoor et al. [71–73] inspiraram-se na noção de valor proposta por Krause e Hochstatter [57]. Porém, ao invés de diferenciar QoC e valor, eles propuseram dividir a noção de QoC em duas visões: a objetiva e a subjetiva. A visão objetiva é independente da situação em que as informações de contexto são utilizadas e das exigências dos consumidores. Essa visão objetiva de QoC é determinada pelas características dos sensores utilizados para coletar os dados de contexto, bem como as condições em que a medição do valor de contexto foi feita (contexto de medição).

A visão subjetiva de QoC expressa o quanto uma informação de contexto está em conformidade com as exigências de uma aplicação consumidora em particular. Os autores argumentam que o contexto adequado para a utilização da informação por uma aplicação pode não ser o recomendado para a utilização dessa mesma informação por outra aplicação. Com base nessas visões, os autores definiram QoC como sendo uma indicação do grau de conformidade das informações de contexto coletadas por sensores em relação à situação que prevalece no ambiente e às exigências de um determinado consumidor. Isto significa que a QoC pode variar de acordo com os diferentes consumidores de contexto.

Bellavista et al. [10] reviram a definição de QoC proposta Buchholz et al. [17]. Eles chegaram à conclusão de que essa definição é muito restritiva e afirmam que nem sempre é possível separar claramente as dimensões de QoC. Eles propuseram então que a noção de QoC não abranja somente a qualidade da informação, mas também a qualidade do serviço de distribuição dos dados de contexto (por exemplo, tempo de entrega dos dados e confiabilidade de entrega), como forma de garantir a satisfação do consumidor.

3.2 Parâmetros e Métricas de Qualidade de Contexto

Esta seção apresenta parâmetros e métricas de QoC propostas por diferentes autores. Assim como ocorre com as definições, não existe uma uniformização em relação à nomenclatura, semântica e taxonomia dos parâmetros e métricas de qualidade. Algumas vezes, autores diferentes usam nomenclaturas diferentes para parâmetros que têm a mesma semântica. Em outras, eles usam a mesma nomenclatura para parâmetros que possuem significados diferentes.

Em relação à classificação, os parâmetros são divididos em dois grupos. O primeiro contém os parâmetros de qualidades da informação e/ou dos dispositivos. O segundo contém os parâmetros de qualidade do serviço de distribuição. Esse agrupamento se deve ao fato de que algumas qualidades, dependendo do autor, são associadas a uma ou a outra dessas duas dimensões de qualidade. Um caso típico é a acurácia, que no trabalho de Buchholz et al. [17] é uma qualidade associada à informação. Já nos trabalhos de Manzoor et al. [73] a acurácia é tida como

uma característica do sensor. Mas, independentemente do autor, essas qualidades são inseridas na informação em forma de metadados. O segundo grupo contém parâmetros de qualidade tidos como exclusivos do serviço de distribuição e que podem ser mais facilmente distinguidos dos parâmetros das demais dimensões de qualidade. Essa divisão tem o objetivo de mostrar a diferença entre os parâmetros que expressam a **qualidade dos dados entregues** à aplicação consumidora (metadados de QoC) e os parâmetros que expressam a **qualidade da entrega dos dados** (configurações que determinam ao serviço de distribuição onde, como e quando entregar as informações).

3.2.1 Parâmetros de Qualidade da Informação/Dispositivos

Acurácia (*Accuracy*)

A acurácia representa o quão próximo o valor mensurado está do valor real. Manzoor et al. [73] e Preuveneers et al. [88] concordam com essa definição. Buchholz et al. [17] usam o termo precisão (*precision*) como sinônimo de acurácia. O erro absoluto de um sensor físico pode ser calculado pela diferença entre o valor real e o valor mensurado pelo sensor. A equação referente ao erro absoluto é a seguinte:

$$E = R - M \tag{3.1}$$

na qual, E é o erro na medição, R é o valor real e M é o valor mensurado pelo sensor. Geralmente, o valor real é acordado antes da medição ou em alguns casos é uma verdade absoluta. Já a acurácia de um sensor físico pode ser calculada através da seguinte equação:

$$Acurracy = 1 - \frac{|E|}{R} \tag{3.2}$$

sendo E o valor da inacurácia e R o valor real. Entende-se que a acurácia é calculada subtraindo o erro relativo de 1. Geralmente, as aplicações de IoT trabalham com a acurácia estimada pelo sensor. Em alguns casos, defeitos e/ou falhas dos sensores podem fazer com que o valor de acurácia estimado seja significantemente distante do valor real.

Distância do Sensor (*Sensor Range/Span*)

Manzoor et al. [73] definem o parâmetro distância do sensor como a distância máxima com a qual o sensor consegue capturar a informação a respeito de uma entidade. Por exemplo, câmeras profissionais conseguem obter imagens a vários metros de distância do local alvo. Já um telescópio consegue imagens dos astros mesmo estando a milhões ou bilhões de quilômetros deles.

Localização da Fonte (*Source Location*)

Manzoor et al. [73] definem o parâmetro localização da fonte como o local onde a informação de contexto foi medida. Entretanto, sabe-se que a maioria dos sensores não possui um sistema de localização embutido. Sendo assim, muitas vezes, a única alternativa é considerar a localização conhecida do dispositivo que estiver mais próximo ao sensor. Por exemplo, um celular com GPS próximo a um sensor de temperatura.

Frequência(*Frequency*)

Para Gray e Salber [49] o parâmetro frequência é definido como a taxa de amostragem do sensor (em hertz).

Intervalo de Medição (*Time period*)

Manzoor et al. [73] define o parâmetro intervalo de medição como o tempo entre medições sucessivas do sensor. Em algumas situações, o usuário do sensor pode ajustar esse parâmetro. Em outras, o sensor já possui um intervalo pré-definido que não pode ser alterado pelo usuário.

Resolução (*Resolution*)

Para Buchholz et al. [17], o parâmetro resolução, chamado por Manzoor et. al [73] de **granularidade** (*granularity*), indica o grau de detalhamento da informação de contexto. A resolução depende da natureza da informação. Por exemplo, informações de localização podem ser fornecidas no nível de rua (granularidade fina) ou no nível de

cidade (granularidade grossa). Para informações do tipo numéricas, a granularidade dos dados pode ser definida com relação à quantidade de casas decimais. Por exemplo, o valor 17,9 tem resolução menor que o valor 17,99.

Instante da Medição (*Mensurement Time*)

Manzoor et al. [73] definem o instante da medição como sendo o momento em que a informação de contexto foi medida.

Idade (*Age*)

Manzoor et al. [73] definem o parâmetro idade como o tempo transcorrido entre o instante da medição da informação e o instante atual. A equação para calcular a idade é a seguinte:

$$Age = t_{curr} - t_{mens} \tag{3.3}$$

onde t_{curr} é o tempo atual e t_{mens} é o instante da medição.

Sendo assim, a idade é um parâmetro muito dinâmico, que deve ser recalculado sempre que a aplicação necessita dele para alguma tomada de decisão, ou seja, não é útil calcular e armazenar a idade para uso posterior. Buchholz et al. [17] utilizam a nomenclatura *up-to-dateness*, enquanto que Kim e Lee [56] e Sheikh et al. [100] usam o termo *freshness* para esse mesmo conceito.

Tempo de Validade (*Validity Time*)

Manzoor et al. [73] definem tempo de validade como o período pelo qual a informação de contexto permanecerá útil para o consumidor. Nessa definição, o próprio consumidor especifica o parâmetro. Entretanto, esta tese considera que pode haver situações em que compete ao produtor determinar o tempo de validade das informações. Fazendo uma analogia com o mundo real, geralmente são os fabricantes que especificam até quando o produto pode ser usado pelo consumidor com segurança.

Embora o tempo de validade seja usualmente classificado como parâmetro de QoI, este pode ser utilizado pelo serviço de distribuição para descartar dados cujo tempo de validade já expirou.

Atributos Disponíveis (*Avaliable Attributes*)

Manzoor et al. [73] definem o parâmetro atributos disponíveis como a quantidade de atributos que são disponibilizados para uma determinada informação. Por exemplo, informações de localização coletadas por um GPS podem ser disponibilizadas com três atributos: latitude, longitude e altitude. Nesta tese, considera-se que a quantidade de atributos por si só não é muito útil ao consumidor. Ao invés disso, deveria-se fornecer uma lista descrevendo os atributos disponíveis.

Atributos Requeridos (*Required Attributes*)

Manzoor et al. [72, 73] definem o parâmetro atributos requeridos como o número de atributos exigidos pelo consumidor.

Assim como no caso anterior, considera-se que é mais útil que consumidor informe uma lista contendo os atributos que necessita.

Valor Crítico (*Critical Value*)

Manzoor et al. [73] definem o parâmetro valor crítico como o nível de importância da informação de contexto de um tipo específico. Esse nível de importância é definido pelo consumidor. Nesta tese, considera-se que pode ser útil também a possibilidade do produtor especificar o valor crítico de uma informação. Nesse caso, o valor poderia ser usado em situações onde existe uma relação de subordinação do consumidor para com o produtor, permitindo que o produtor use o valor crítico para indicar ao consumidor a prioridade com a qual deseja que os eventos sejam processados. Fazendo uma analogia com o mundo real, é como se um chefe de setor indicasse a prioridade de uma tarefa especificando um valor crítico para ela.

Cobertura (*Coverage*)

Para Dey et al. [34], o parâmetro cobertura define o conjunto de todos os valores possíveis para um atributo de contexto. Por exemplo, um atributo que define os possíveis estados de ativação de um sensor: ligado ou desligado.

Nível de Acesso Requerido (*Required Access Level*)

Manzoor et al. [73] definem o parâmetro nível de acesso requerido como o grau de permissão exigido pelo consumidor em relação a uma determinada informação de contexto.

Nível de Acesso Concedido (*Allowed Access Level*)

Manzoor et al. [73] definem o parâmetro nível de acesso permitido como o grau de permissão concedido pelo produtor em relação a uma determinada informação de contexto.

3.2.2 Métricas de Qualidade da Informação/Dispositivos

Esta seção apresenta algumas métricas propostas por Manzoor et al. [73] para avaliar a qualidade das informações/dispositivos.

Confiabilidade (*Reliability*)

A métrica confiabilidade avalia o grau de confiança na exatidão da informação. É calculada com base na acurácia (ver parâmetro acurácia na Seção 3.2.1) com a qual o sensor coletou a informação de contexto e a distância do sensor em relação à entidade da qual a informação é coletada. A equação referente a essa métrica é mostrada a seguir.

$$Reliability = \begin{cases} (1 - \frac{d(s,\varepsilon)}{d_{max}}) \times \delta & : \quad if \ d(s,\varepsilon) < d_{max} \\ 0 & : \quad otherwise \end{cases} \tag{3.4}$$

onde $d(s, \varepsilon)$ é a distância entre o sensor e a entidade e d_{max} é a distância máxima na qual confia-se na observação deste sensor.

Como mostrado, a confiabilidade na exatidão é diretamente proporcional à acurácia do sensor e inversamente proporcional à distância entre o sensor e a entidade sobre a qual as informações de contexto são coletadas.

Pontualidade (*Timeliness*)

A métrica pontualidade avalia o grau de atualidade da informação. Em uma visão objetiva, se a idade (ver parâmetro idade na Seção 3.2.1) for menor que o intervalo entre as medições do sensor (ver parâmetro intervalo de medição na Seção 3.2.1), a pontualidade é calculada com base na razão entre a idade e o intervalo de medição. Do contrário, a pontualidade terá valor igual a zero. A pontualidade é um métrica normalizada em uma faixa de 0 a 1, conforme mostra a equação a seguir:

$$Timeliness = \begin{cases} 1 - \frac{Age}{TimePeriod} & : \quad if\ Age < TimePeriod \\ 0 & : \quad otherwise \end{cases} \tag{3.5}$$

Na visão subjetiva, o parâmetro tempo de validade (ver Seção 3.2.1) substitui o parâmetro intervalo de medição, conforme mostra a equação a seguir:

$$Timeliness = \begin{cases} 1 - \frac{Age}{ValidityTime} & : \quad if\ Age < ValidityTime(0) \\ 0 & : \quad otherwise \end{cases} \tag{3.6}$$

O valor de pontualidade decresce à medida em que a idade aumenta. Quanto mais próximo de zero for o valor da pontualidade, menos atual é a informação de contexto e menor será a utilidade que ela terá para o consumidor.

Completude (*Completeness*)

A métrica completude avalia o quão completa é a informação recebida pelo consumidor. Indica a quantidade de informação que é fornecida por um objeto de contexto, podendo ser calculada com base na razão entre a soma dos pesos dos atributos disponíveis (ver parâmetro atributos disponíveis na Seção 3.2.1) e a soma dos

pesos do número total de atributos que são requeridos pelo consumidor. A equação referente a essa métrica é a seguinte:

$$Completeness = \frac{\sum_{j=0}^{m} w_j}{\sum_{i=0}^{n} w_i} \tag{3.7}$$

onde m é o número de atributos disponíveis e n é o número total ou necessário de atributos para a informação desejada. Os pesos w são empregados devido ao fato de que cada atributo pode ter uma importância diferente para cada consumidor. Quanto menor o valor de completude, mais incompleta é a informação recebida.

Significância (*Significance*)

A métrica significância avalia o grau de importância da informação de contexto recebida. É calculada com base na razão entre o valor crítico da informação recebida (ver parâmetro valor crítico na Seção 3.2.1) e o valor crítico máximo que ela poderia ter. A equação referente a essa métrica é a seguinte:

$$Significance = \frac{CV}{CV_{max}} \tag{3.8}$$

onde o CV é o valor crítico da informação de contexto recebida. O CV_{max} é o valor crítico máxima que ela poderia ter. Quanto maior o valor de significância, mais importante é a informação de contexto recebida para o consumidor.

Usabilidade (*Usability)*

A métrica usabilidade avalia o grau de detalhamento da informação. A equação referente a essa métrica é a seguinte:

$$Usability = \begin{cases} 1 & : \ if\ ReceivedGranularity \geq RequiredGranularity \\ 0 & : \ otherwise \end{cases} \tag{3.9}$$

A equação mostra que se a granularidade recebida for maior ou igual à granularidade requerida, então o valor de usabilidade será igual a 1. Contrário a usabilidade recebe valor zero.

Esta tese considera que o termo usabilidade não é empregado corretamente para essa métrica pois esse termo está mais relacionado à facilidade de uso de sistemas computacionais.

Direito de Acesso (*Access Right*)

A métrica direito de acesso avalia o nível de acesso à informação, podendo ser calculada com base na razão entre o nível de acesso permitido pelo dono da informação e nível de acesso que é requerido pelo consumidor. A equação referente a essa métrica é a seguinte:

$$AccessRight(\mathcal{O}) = \begin{cases} 1 & : \quad if\ AccessLevel(\mathcal{O}) \geq AccessLevel(\mathcal{CR}) \\ 0 & : \quad otherwise \end{cases} \tag{3.10}$$

Tal como mostrado pela equação acima, o direito de acesso será igual a 1 se o nível de acesso da informação de contexto recebida é maior do que o nível de acesso solicitado pelo consumidor para essa informação. Caso contrário, será igual a zero.

Consistência de Representação (*Representation Consistency*)

A métrica consistência representa o grau do esforço necessário para transformar a informação de contexto recebida, do modelo de dados em que está, para o modelo que o consumidor necessita. A equação referente a essa métrica é a seguinte:

$$RepresentationConsistency = \frac{k}{TransformationEffort} \tag{3.11}$$

onde k é uma constante utilizada para normalizar o valor de consistência de representação no intervalo [0..1]. O valor dessa constante depende do esforço mínimo e máximo que um consumidor de contexto tem para executar a transformação.

Tal como mostrado pela equação acima, a consistência de representação é inversamente proporcional ao esforço requerido para transformar a informação para o modelo de dados que o consumidor precisa.

3.2.3 Parâmetros de Qualidade do Serviço de Distribuição

Taxa de Atualização (*Refresh rate*)

Para Huebscher e McCann [53], o parâmetro taxa de atualização descreve a frequência com a qual as aplicações consumidoras querem receber as informações de contexto, independentemente da frequência com a qual são produzidas. Pardo-Castellote [85] usa a nomenclatura intervalo mínimo de separação (*minimum separation time*) para expressar o menor intervalo entre amostras recebidas exigido pelo consumidor.

Confiabilidade (*Reliability*)

Segundo Pardo-Castellote [85], o parâmetro confiabilidade determina se o serviço de distribuição deve adotar a política do melhor esforço (*best-effort*) ou utilizar mecanismos de confirmação de entrega e retransmissão, caso a informação de contexto não chegue ao destinatário. Entretanto, alguns protocolos para a IoT, como o MQTT [61], implementam três níveis de confiabilidade:

- No máximo uma vez (*At most once*) ou nível 0: emprega a política do melhor esforço. A mensagem é entregue uma vez ou não é entregue, sendo que não há confirmação de entrega.
- Pelo menos uma vez (*At least once*) ou nível 1: exige confirmação de entrega, sendo que retransmissões são realizadas em caso de não-confirmação. Dados podem ser entregues mais de uma vez ao destinatário.
- Exatamente uma vez (*Exactly once*) ou nível 2: exige confirmação da entrega e retransmissões são realizadas em caso de não-confirmação. Porém, usa um protocolo de confirmação com duplo aperto de mão (*double handshake*) para assegurar que o destinatário receberá cada mensagem apenas uma vez.

Atraso (*Delay*)

Para Bu et al. [16], o atraso é o intervalo de tempo entre a saída da informação de contexto do produtor e sua chegada no consumidor.

Histórico (*History*)

Segundo Pardo-Castellote [85], o parâmetro histórico indica a quantidade de informações de contexto que o consumidor deseja armazenar em cache.

Vida útil (*Lifespan*)

Segundo Pardo-Castellote [85], o parâmetro vida útil especifica o tempo pelo qual o serviço manterá as informações no histórico. Quando o prazo expira, as informações são deletadas.

Vivacidade (*Liveliness)*

Segundo Pardo-Castellote [85], o parâmetro vivacidade indica se o serviço de distribuição deve tornar os subscritores cientes de falhas nos publicadores.

Ordem de Destino (*Destination Order*)

Segundo Pardo-Castellote [85], o parâmetro ordem de destino especifica se as informações inseridas no histórico devem ser organizadas com base no tempo em a informação foi enviada pelo publicador ou com base no tempo de chegada do dado ao subscritor. Ao considerar o tempo do publicador, é necessário que haja um mecanismo de sincronização entre os relógios do publicador e do subscritor.

Tempo de Recuperação dos Dados (*Data Retrieval Time*)

Para Corradi et al. [27], este parâmetro corresponde ao tempo máximo pelo qual o consumidor está disposto a esperar pela informação solicitada. Pardo-Castellote [85] chama esse parâmetro de prazo (*deadline*).

Prioridade de Transporte (*Priority)*

Para Corradi et al. [27] e Pardo-Castellote [85], a prioridade de transporte visa permitir o tráfego diferenciado entre múltiplos fluxos de dados.

3.3 Motivações para o uso da QoC

Os trabalhos de Buchholz et al. [17] e Sheikh et al. [99] mostram algumas razões para se considerar explicitamente o uso de QoC. São elas:

1. **Estabelecer Contratos de QoC**: a QoC pode ser usada para firmar, quando necessário, um contrato de qualidade entre produtores e consumidores. Por exemplo, se a aplicação deseja receber dados de localização a cada 30 segundos, mas o produtor só pode enviá-los a cada 60 segundos, então um contrato não é fechado. Logo, o o serviço não será contratado.

2. **Selecionar Provedores de Contexto**: ainda que a seleção do provedor de contexto não dependa de um contrato, o consumidor pode escolher aquele cuja qualidade de contexto atende melhor aos seus requisitos.

3. **Aumentar a Eficiência**: para aumentar a eficiência na utilização da largura de banda, o middleware de contexto poderá armazenar informações de contexto na *cache* do consumidor. Por exemplo, dependendo do intervalo entre as medições das informações, as aplicações consumidoras poderão obtê-las a partir da *cache* local ao invés de solicitar o envio de um novo dado ao provedor.

4. **Melhorar a Qualidade de Experiência do Usuário**: considerando que as informações de contexto são utilizadas para adaptar o conteúdo e os serviços providos pelas aplicações sensíveis ao contexto, é certo que a qualidade de contexto tem impacto direto sobre o funcionamento dessas aplicações e consequentemente sobre a experiência de seus usuários. Se a experiência dos usuários melhora na medida em que a qualidade das informações de contexto aumenta, é preciso buscar mecanismos que mantenham a QoC em níveis suficientes para que os usuários das aplicações sensíveis ao contexto fiquem satisfeitos com seus serviços.

5. **Prover Privacidade**: existe uma relação entre a privacidade e a QoC. O *middleware* precisa fornecer aos usuários meios para limitar a QoC de suas informações pessoais. Por exemplo, a localização do usuário pode ter a acurácia reduzida propositalmente em situações em que ele não deseja ser encontrado.

3.4 Fatores que degradam a Qualidade de Contexto

Diversos fatores podem degradar a qualidade de contexto. Por exemplo, os sensores físicos, devido às suas próprias naturezas, podem estar sujeitos a erros de leitura provocados por má calibração ou falha de fabricação. Os sensores também podem estar mal posicionados e gerar informações incorretas. Para informações de contexto baseadas em perfis do usuário, a omissão do usuário em fornecer entradas pode fazer com que algumas informações sejam desconhecidas ou incompletas.

Alguns fatores que degradam a qualidade do serviço de distribuição:

1. **Falhas na Comunicação**: perda de pacotes, atrasos (*delay*) e variações de atrasos (*jitter*), conectividade fraca ou intermitente, desconexões, *handovers* de nodos móveis;

2. **Falhas em componentes de hardware**: rompimento de cabos, defeitos em antenas ou satélites, quedas de energia que ocasionam o desligamento de servidores e estações de trabalho, problemas em estações rádio base utilizadas em redes móveis;

3. **Falhas em componentes de software**: travamento de aplicações, erros de lógica de programação que levam a exceções não previstas, falhas do sistema operacional ou na própria aplicação;

4. **Limitações de escalabilidade**: situações em que a quantidade de usuários, o grande volume de dados transmitidos e alocação ineficiente dos recursos de processamento, armazenamento e comunicação provocam congestionamentos e aumento do tempo de resposta ao consumidor.

Problemas inerentes às fases de aquisição podem influenciar, mesmo que indiretamente, no tempo de entrega das informações ao consumidor. Por exemplo, falhas nos sensores ou longos intervalos entre medições podem fazer com que a informação de contexto demore a ser enviada. Consequentemente haverá atrasos e as taxas de atualização contratadas deixarão de ser cumpridas.

3.5 Conclusão do Capítulo

Este capítulo mostrou que não há consenso na literatura sobre o conceito de qualidade de contexto. A definição tradicional considera que a QoC está restrita à qualidade da informação. Outra definição, mais recente, considera que a noção de QoC deve incluir também a qualidade do serviço de distribuição dos dados de contexto.

Esta teste concorda com essa definição mais recente. Entretanto, sugere-se que a noção de QoC deve incluir também a qualidade dos dispositivos, pois a QoD pode influenciar tanto a QoI como a QoS. Diante disso, esta tese propõe um conceito de QoC ainda mais abrangente que os existentes na literatura: *"QoC é qualquer informação que descreva a qualidade ofertada e/ou exigida, tanto em relação aos dados entregues (QoI), como no que diz respeito ao serviço de entrega (QoS) e dispositivos que proveem os dados (QoD)"*.

Existe uma ampla diversidade de parâmetros e métricas de QoC propostos na literatura. Acredita-se que este capítulo apresentou a grande maioria deles. Mostrou-se que também não há consenso em relação às nomenclaturas, semânticas e taxonomias dos parâmetros de QoC. Os parâmetros de QoC apresentados foram divididos propositalmente em dois grupos: qualidade das informações/dispositivos e qualidade do serviço de distribuição.

Parte das abordagens de middleware com suporte a QoC estão mais voltadas ao provisionamento de qualidade da informação, enquanto outras estão mais voltadas ao provisionamento de qualidade do serviço de distribuição. Essas abordagens são apresentadas no capítulo 8. Neste trabalho, está sendo proposta uma abordagem de middleware mais completa, que oferece suporte a uma grande quantidade de parâmetros de QoC, tanto de QoI como QoS.

Apresentou-se também algumas motivações para o uso de QoC em sistemas sensíveis ao contexto. Alguns trabalhos utilizam QoC para se adaptar e selecionar o melhor provedor de contexto. Outros consideram a noção de QoC para tomar decisões que melhorem a eficiência do middleware. Há ainda aqueles que utilizam QoC com o intuito de prover privacidade aos usuários.

Por fim, mostrou-se alguns fatores que degradam a qualidade de contexto. No geral, as principais falhas estão relacionadas aos sensores, falhas na comunicação e falhas em componentes de hardware e software.

4 Bases Tecnológicas Fundamentais

Este capítulo apresenta um resumo da fundamentação teórica referente a algumas bases tecnológicas utilizadas no desenvolvimento da solução proposta, que será apresentada no Capítulo 5.

4.1 Processamento de Eventos Complexos

Diversos ramos de negócios exigem que seus sistemas computacionais reajam rapidamente às mudanças do ambiente. Essa capacidade pode ser um fator decisivo para o sucesso ou fracasso das empresas. Com o número cada vez maior de redes de sensores e dispositivos inteligentes coletando ininterruptamente grandes quantidades de dados de contexto, que refletem as mudanças do ambiente, o problema de analisar um fluxo cada vez maior de eventos, em tempo quase real, se torna crítico.

O processamento de eventos complexos (*Complex Event Processing* - CEP) [68] é uma tecnologia que permite correlacionar eventos de entrada contínuos e padrões de interesse, onde os resultados do processamento podem ser outros eventos complexos, ou seja, eventos que são derivados dos eventos de entrada. CEP é um método de rastreamento e análise de baixa latência (em termos de processamento) de fluxos de dados. Esse método combina dados de múltiplas fontes para inferir eventos ou padrões que sugerem circunstâncias mais complicadas [46]. CEP é usado para processar e analisar fluxos de dados em tempo próximo ao real, bem como produzir resultados sem atrasos, mesmo em casos onde o fluxo de eventos é grande. Em uma inversão em relação aos sistemas gerenciadores de bancos de dados tradicionais, onde uma consulta é executada em dados armazenados, o CEP executa dados em uma consulta armazenada. Dados irrelevantes para uma consulta CEP podem ser descartados imediatamente. CEP permite que se detecte situações em uma série de eventos (históricos ou em tempo real), de modo que ações específicas sejam tomadas quando ocorrerem condições previamente definidas.

O processamento de eventos complexos pode ser aplicado em uma ampla variedade situações. Por exemplo, CEP é bastante utilizado em aplicações financeiras, a fim de detectar tendências no mercado de ações e fraudes de cartão de crédito. Uma outra aplicação cotidiana é o monitoramento e rastreamento baseado em identificação por radiofrequência (*Radio-Frequency IDentification* - RFID), por exemplo, para detecção de furtos em supermercados. O CEP também pode ser usado para detectar invasões em redes de computadores, por meio da especificação de padrões de comportamentos suspeitos dos usuários.

4.1.1 Redes de Processamento de Eventos Complexos

Em CEP, eventos são criados por produtores (*producers*), que são entidades (por exemplo, sensores e aplicações clientes) que geram ocorrências de interesse de um domínio da aplicação. O fluxo de eventos (*event data stream*) é o resultado de uma sequência de eventos criada e enviada por produtores. O *workflow* CEP processa, analisa e manipula continuamente essa sequência de eventos de entrada. Em seguida, têm-se eventos derivados na saída, que são entregues para entidades consumidoras (*consumers*) (por exemplo, aplicações de monitoramento). Essas saídas geralmente representam notificações sobre situações detectadas [92]. Mais precisamente, é possível identificar hierarquias de eventos, onde cada consulta contínua é executada por um agente intermediário de processamento, chamado *Event Processing Agent* - EPA. A partir da comunicação entre EPAs, produtores e consumidores, feita através da interconexão entre seus canais de entrada e saída, forma-se uma rede de processamento de eventos (*Event Processing Network* - EPN) [39], como ilustrado na Figura 4.1.

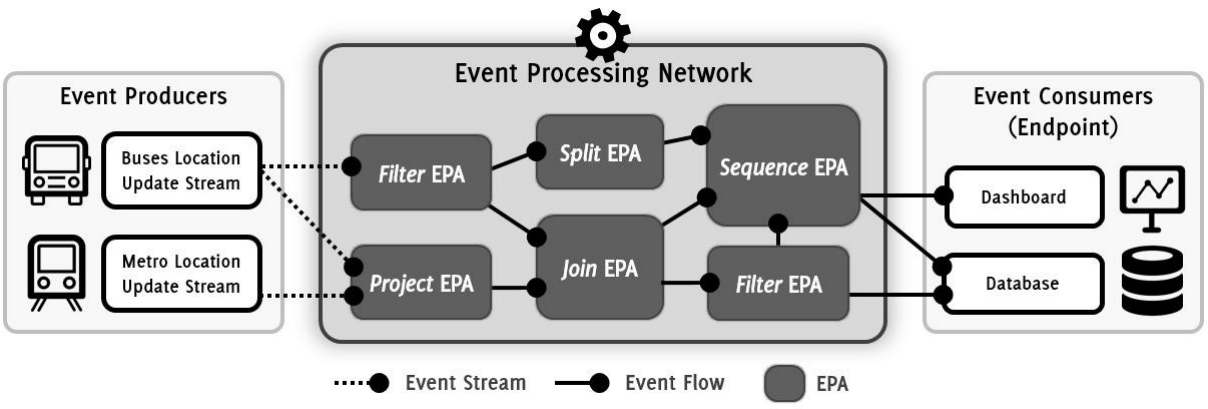

Figura 4.1: Rede de Processamento de Eventos [39]

4.1.2 Janelas de Tempo

Um conceito importante para processar o fluxo de eventos de entrada em CEP é o de janelas de tempo. Precisamente, uma janela de tempo é um contexto temporal [68] [39] que subdivide um fluxo de eventos em intervalos de tempo usando o atributo *timestamp*. Uma janela de tempo divide o fluxo de eventos usando a *tag timestamp* de cada evento, de tal forma que inclui somente eventos que estão dentro do intervalo dado, por exemplo, (*now* - Δ, *now*), onde *now* é o *timestamp* atual. Por exemplo, a declaração de janela de tempo *LocationUpdate* [SLIDE 30 segundos] cria automaticamente uma partição de contexto temporal que retém os últimos eventos *LocationUpdate* (Δ = 30 segundos). Quando um EPA processa um evento com *timestamp* *t* em tal janela, as primitivas de tempo real devem considerar somente os eventos recebidos no intervalo (*t* - 30, *t*). Este conceito é útil para garantir que apenas os eventos mais recentes do fluxo de dados sejam considerados.

Os dois principais tipos de janelas de tempo em CEP são:

- **Landmark Windows** (ou *Time Window*): fornecem a capacidade de processar o fluxo de eventos em lote (*batch*). Ele armazena todos os eventos produzidos durante um intervalo de tempo Δ e aplica as consultas contínuas a todo esse conjunto de eventos. Precisamente, pode haver um atraso entre a hora de chegada do evento e seu tempo de processamento. Por exemplo, na Figura 4.2, apesar de os eventos E1 e E2 terem chegado no tempo t = 1, eles serão processados somente quando o período do lote (*batch*) chegar ao fim (t = 2). Ao armazenar em *buffer*[1] os eventos, é possível usar primitivas, funções de agregação, como *min* e *max*, para sumarizar todo o conteúdo do lote em um evento complexo [5].

- **Slide Windows** (ou *Time Batch*): se movem (deslizam) junto com os eventos recebidos. Assim, em vez de ter períodos de *batch* predefinidos, os limites da janela de tempo deslizam para o *timestamp* atual do evento. Mais especificamente, uma janela deslizante é uma *Landmark Window* móvel. Para ilustrar esse cenário, considere o fluxo de eventos com uma janela de tempo deslizante de 1 segundo na Figura 4.2(b). Por exemplo, quando t = 3, a janela de tempo inclui os eventos do segundo passado (t = 2) até o tempo atual. Esse movimento deslizante,

[1]Uma região de memória temporária utilizada para escrita e leitura de dados.

ajustando os intervalos de janela de tempo para o atual evento sendo analisado, atenua o problema de ter eventos correlacionados adjacentes em diferentes períodos de *batch* (lote), ou seja, na Figura 4.2(a), os eventos E3 e E4 não podem ser correlacionados em um mesmo lote, devido à separação em lotes distintos nos tempos t=2 e t=4. Ao deslizar a janela de tempo é possível incluir os eventos adjacentes anteriores que seriam colocados em lotes diferentes. Ao usar a janela de tempo deslizante com primitivas em tempo real, é possível vislumbrar os eventos passados e fornecer processamento de eventos contínuo e próximo ao tempo real [5].

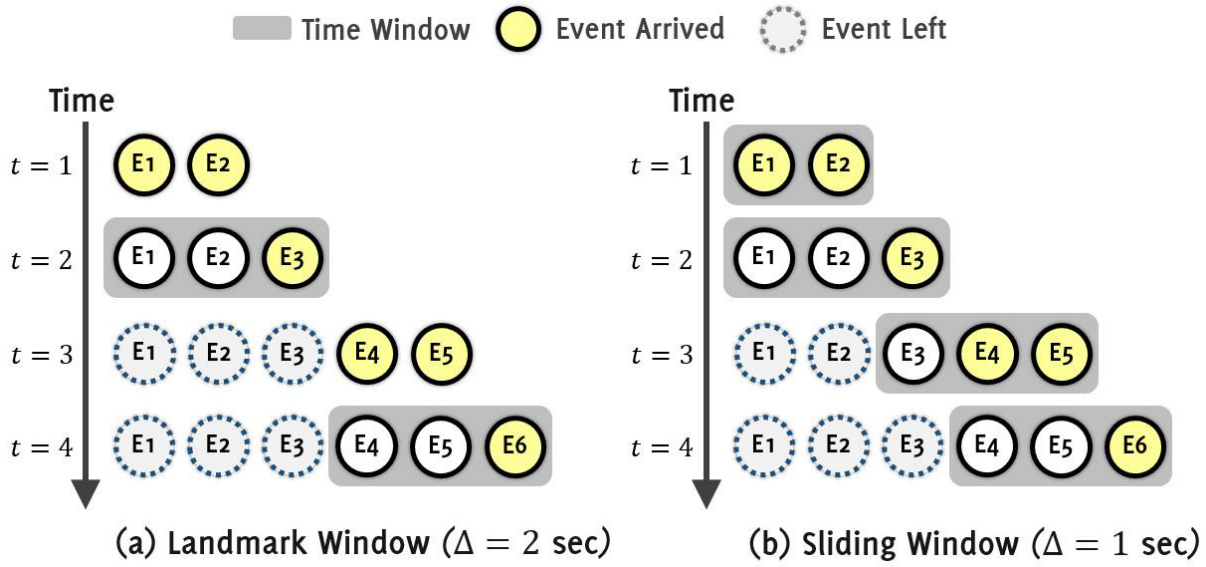

Figura 4.2: *Landmark* e *Sliding Time Windows* [92].

4.1.3 Linguagens e Engines CEP

Uma das vantagens do CEP é a possibilidade de utilizar linguagens declarativas para definir consultas de processamento sobre os fluxos eventos, conhecidas como *Event Processing Languages* - EPLs) [68]. EPLs permitem expressar condições ricas e correlações entre eventos, bem como conceitos de janelas de tempo, minimizando dessa forma, o esforço de desenvolvimento necessário para configurar sistemas que possam reagir a situações complexas [38]. Devido à expressividade das EPLs, cenários complexos de detecção de eventos, que anteriormente eram difíceis de se implementar usando outras tecnologias, podem agora ser especificados com poucas linhas de código, favorecendo a reutilização das soluções.

O suporte à execução de consultas sobre fluxos de eventos, escritas em EPL, é provido por motores de processamento de eventos complexos, simplesmente conhecidos como *Engines CEP*. Existem várias *Engines CEP* disponíveis. Alguns exemplos conhecidos de Engines CEP são: Esper [38], Sase [111], Apache Flink [20], Red Hat Drools [89] e TelegraphCQ [21]. Algumas Engines CEP são distribuídas como software livre, enquanto outras são proprietárias. Porém, a principal diferença entre as *Engines* CEP são as construções de linguagens usadas para definir tipos de eventos e primitivas de tempo real suportadas.

Em geral, as construções das EPLs implementadas pelas *engines* CEP podem ser divididas em dois modelos: orientadas por fluxos (*stream-oriented*) e orientadas por regras (*rule-oriented*) [39] [46] [58]. As engines CEP que implementam o modelo orientado por fluxos seguem o design padrão formal da linguagem de consulta contínua (*Continuous Query Language* - CQL) [5]. A CQL é uma extensão SQL que provê suporte para operadores de *streams*. Várias *engines* CEP usam uma linguagem CQL-like devido ao seu formalismo e sua similaridade com a linguagem SQL, que é bastante popular. Em contraste com as linguagens orientadas por fluxos, as engines CEP que implementam o modelo orientado por regras baseiam-se em inferência e cláusulas de Evento-Condição-Ação (*Event-Condition-Action* - ECA) [58]. Inferências e regras ECA separam o tratamento dos eventos, a verificação das condições e a ação em diferentes cláusulas. Primeiramente, uma consulta contínua é acionada quando ocorre um determinado evento. Em seguida, uma restrição ou condição é verificada. Finalmente, se a condição for satisfeita, as regras executam a cláusula da ação.

Uma das *engines* CEP mais conhecidas é a *Esper CEP*[2] [38], uma *engine Open Source*, destinada a arquiteturas EDA (*Event Driven Architecture*) de tempo real, escrita inteiramente em Java, que permite que ela seja embutida em qualquer processo Java. A EPL Esper é uma rica linguagem declarativa para especificação de consultas contínuas, que inclui todos os operadores SQL (*SQL-like extended*), construções da linguagem para iteração e definição de janelas, bem como para a geração de eventos de saída [46].

Por ter uma sintaxe bastante semelhante à SQL, o vocabulário da EPL Esper inclui cláusulas como SELECT, FROM, WHERE, GROUP BY, HAVING e ORDER BY. Os fluxos de eventos substituem as tabelas como fontes de dados. Já os eventos

[2] `http://www.espertech.com/esper/`

substituem as linhas das tabelas como unidade básica de dados. Uma vez que os eventos são compostos de dados, os conceitos SQL de correlação através de associações, filtragem e agregação através de agrupamento podem ser implementados.

Entretanto, existem diferenças importantes entre EPL Esper e SQL. Uma delas consiste no fato da EPL permitir que o fluxo de saída de uma consulta contínua seja inserido em outro fluxo de eventos, de modo que esses eventos de saída sejam consumidos por outras consultas contínuas, algo que a SQL padrão não permite. Outra diferença em relação à SQL, é a capacidade da EPL de usar janelas de tempo (*time windows*), isto é, analisar e correlacionar eventos de entrada com um subconjunto temporal do fluxo de eventos principal.

Para exemplificar o uso de CEP em sistemas de contexto, apresenta-se um cenário de *Ambient Assisted Living*, no qual supõe-se que determinada aplicação de IoT monitora a temperatura corporal de um paciente, por meio da utilização de um sensor vestível que coleta dados com um intervalo de 1 segundo entre as medições. Nesse cenário de AAL, a aplicação de monitoramento define 3 tipos de eventos a serem detectados:

- **Temperatura Média**: informa à aplicação de monitoramento apenas a média das leituras de temperatura do paciente, considerando uma janela de 10 segundos.
- **Atenção**: avisa a aplicação quando três leituras seguidas forem maiores que 38°C.
- **Crítico**: avisa quando a temperatura subir abruptamente e a leitura inicial for maior que 40 °C. Se a temperatura aumentar durante cinco leituras consecutivas e a última leitura for 10% maior que a primeira, considera-se que o estado é crítico.

Utilizando-se EPL pode-se criar três consultas para modelar cada um dos padrões de eventos mencionados. A estas consultas pode-se anexar *listeners*, de modo que, quando o padrão for detectado, uma rotina seja ativada. A seguir apresenta-se exemplos de consultas escritas em EPL Esper para os três tipos de eventos citados.

Listing 4.1: EPL para eventos do tipo "Temperatura Média".

```
select avg(value) as avg_val
from TemperatureEvent.win:time_batch(10 sec)
```

Listing 4.2: EPL para detecção de eventos do tipo "Atenção".

```
select * from TemperatureEvent
 match_recognize (
   measures A as temp1, B as temp2, C as temp3
   pattern (A B C)
   define
     A as A.temperature > 38,
     B as B.temperature > 38,
     C as C.temperature > 38)
```

Listing 4.3: EPL para detecção de eventos do tipo "Crítico".

```
select * from TemperatureEvent
 match_recognize (
   measures A as temp1, B as temp2, C as temp3, D as temp4,
   E as temp5
   pattern (A B C D E)
   define
     A as A.temperature > 40,
     B as (A.temperature < B.temperature),
     C as (B.temperature < C.temperature),
     D as (C.temperature < D.temperature),
     E as (D.temperature < E.temperature)
      and E.temperature > (A.temperature * 1/100)
```

No âmbito desta tese, a *engine* Asper [35], uma versão reduzida da Esper, voltada para dispositivos móveis, como mostrado no Capítulo 5, foi empregada no desenvolvimento de mecanismos de descoberta de serviços, avaliação, filtragem e monitoramento baseados em QoC, implementados como parte da solução proposta. Apresenta-se detalhes sobre como esses mecanismos funcionam no referido capítulo.

4.2 Projeto ContextNet

O projeto intitulado ContextNet [3], desenvolvido pelo *Laboratory for Advanced Collaboration* (LAC) da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), com colaboração do Laboratório de Sistemas Distribuídos Inteligentes (LSDi) da Universidade Federal do Maranhão, visa o desenvolvimento de um middleware para o processamento de fluxos de dados móveis em larga escala e baixa latência, com suporte para a cooperação e coordenação de atividades de nódos móveis, ciência de contexto, balanceamento de conexões e integração com a núvem. Além do middleware, o projeto ContextNet também abrange o desenvolvimento de aplicações de colaborativas móveis, em diversos domínios, como saúde e transporte.

Duas das principais iniciativas que integram o projeto ContextNet são o *Mobile Hub* (M-Hub) [47, 48, 91, 105] e o *Scalable Data Distribution Layer* [32, 47, 48, 103, 104, 106]. M-Hub é um serviço de middleware responsável pela aquisição de dados brutos a partir de sensores via WPAN. Por padrão, o M-Hub utiliza a SDDL como camada de distribuição de dados para aplicações conectadas à nuvem.

No âmbito desta tese, como mostrado no Capítulo 5, foram reutilizados alguns subcomponentes da arquitetura do M-Hub, no desenvolvimento da abordagem de middleware proposta. Entretanto, alguns dos componentes que foram reutilizados tiveram suas implementações estendidas e/ou modificadas, a fim de adequá-las aos objetivos da solução proposta. Essa solução não usa o SDDL como camada de distribuição, mas sim outra camada, projetada e avaliada neste trabalho, chamada *Context Data Distribution Layer* - CDDL.

A CDDL surgiu da necessidade de prover mecanismos de QoC que não são oferecidos pela SDDL para as aplicações, sendo que a SDDL foca em outros aspectos da distribuição. A concepção e desenvolvimento da CDDL equivalem à contribuição original da tese. A seguir, apresenta-se resumidamente a arquitetura e os componentes do M-Hub e da SDDL, referentes à versão original do projeto ContextNet. Considerando que este é um capítulo que aborda os fundamentos da tese, as explicações sobre quais subcomponentes específicos do M-Hub foram reutilizados, bem como os detalhes sobre as extensões/modificações realizadas nesses componentes, são fornecidos no Capítulo 5, junto com a descrição da solução proposta.

[3] `http://www.lac.inf.puc-rio.br/dokuwiki/doku.php`

4.2.1 Mobile Hub (M-Hub)

O M-Hub é um serviço de middleware, que executa em dispositivos pessoais móveis, responsável pela descoberta, conexão e aquisição de dados de baixo nível junto a objetos inteligentes (e.g., sensores, atuadores, relógios, robôs, veículos) próximos e acessíveis a partir de tecnologias *Wireless Personal Area Network* (WPAN) como Bluetooth Classic, BLE, ZigBee, NFC, ANT+. Portanto, o M-Hub é um *gateway* de IoMT, servindo de ponte entre os objetos inteligentes e a Internet [48, 91, 105].

Na arquitetura proposta, o M-Hub fornece as seguintes funcionalidades:

- **Descoberta de sensores próximos:** periodicamente, o M-Hub irá procurar objetos inteligentes próximos que estão anunciando seus IDs e capacidades. Os dados sobre os objetos inteligentes alcançáveis são mantidos em uma base local;
- **Conexão com sensores:** dependendo do protocolo WPAN usado, o M-Hub pode precisar primeiramente estabelecer um link de comunicação com o sensor, pelo qual ele irá emitir solicitações síncronas ou assíncronas de recebimento de dados. A conexão com os sensores obedece os protocolos de segurança implementados por cada WPAN/dispositivo, de forma que o M-Hub consegue se conetar aos objetos inteligentes utilizando técnicas como pareamento e geração de chaves (comuns, por exemplo, em dispositivos Bluetooth Classic e Bluetooth Low Energy) a fim de prover comunicação encriptada, confiável e preservar a integridade dos dados e a privacidade dos usuários;
- **Protocolo de transcodificação:** Os pacotes de dados recebidos dos sensores podem ter diferentes formatos e codificações. Assim, o M-Hub deve transcodificá-los e serializá-los, antes de transmiti-los. A transcodificação de dados é altamente dependente do tipo, marca e fabricante do sensor;
- **Caching de dados:** a fim de otimizar a transmissão para a nuvem através da Internet móvel, o M-Hub pode agrupar várias amostras de dados obtidas a partir de vários sensores próximos antes de enviá-las ao gateway em rajada única. Para isso, o M-Hub armazena em cache as amostras de dados recebidas dos sensores;
- **Configuração e Controle dos sensores:** dependendo do tipo de sensor, o M-Hub pode, eventualmente ou periodicamente, enviar comandos, definições de

parâmetros ou requisições de consultas de dados através da WPAN para os sensores conectados;

- **Pré-processamento de dados do sensor:** antes de enviar os dados, pode ser necessário aplicar uma função de pré-processamento (e.g. transcodificação, formatação, agregação, filtragem, ou comparação com leituras anteriores etc.). Esse pré-processamento é feito no M-Hub.

- **Carregamento dinâmico de módulos do sensor:** uma vez que não é possível ter módulos internos para todos os sensores que podem estar disponíveis na medida que o gateway se move, o M-Hub oferece um mecanismo de implantação de módulo em tempo de execução e gerenciamento do ciclo de vida desses módulos;

- **Processamento nas pontas**: o M-Hub oferece mecanismos que permitem aos desenvolvedores de aplicações distribuírem as funcionalidades do seu código entre o dispositivo móvel e nodos da nuvem, movendo parte do processamento das informações de contexto para "as pontas" do sistema. A isso dá-se o nome de *In-Network Processing*, uma técnica que ajuda a diminuir a quantidade de informações a ser transmitida para a nuvem, podendo melhorar a escalabilidade do sistema. O processamento ao qual esta funcionalidade se refere pode ser implementado em código Java convencional ou utilizando EPL Esper.

- **Processamento Ciente de Energia**: por meio de um componente de gerenciamento de energia, o M-Hub monitora o nível da bateria do dispositivo móvel, disparando ações adaptativas que ajustam os comportamento dos seus serviços, de acordo com a disponibilidade de energia do dispositivo móvel. Por exemplo, a frequência de publicação dos dados pode ser reduzida quando o nível da bateria está baixo (por exemplo, menor que 20%);

Em relação às WPANs, o M-Hub oferece suporte a conexão com objetos inteligentes via *Classic Bluetooth* (BT 2.0 e 3.0) e *Bluetooth Low Energy* (BLE 4.0). Por ter uma arquitetura aberta e independente de tecnologia, o M-Hub pode ser estendido para dar suporte a novas WPANs. Apesar de seu maior consumo de energia, o *Classic Bluetooth* ainda é utilizado por uma ampla variedade de dispositivos empregados na área da saúde, tais como o Zephyr BioHarness 3 [4] e Zephyr HxM BT [5]. A BLE

[4]http://zephyranywhere.com/ produtos/BioHarness-3/

[5]http://zephyranywhere.com/products/hxm-bluetooth-heart-rate-monitor/

está emergindo como uma tecnologia muito promissora. Isso porque ela é mais eficiente em relação ao consumo de energia e também permite uma descoberta mais rápida de dispositivos periféricos. Além disso, a BLE suporta cerca de 2.500 conexões simultâneas. Mas a razão mais importante para a escolha da BLE é o fato de que ela está disponível em um número crescente de modelos de *smartphones* (e.g., Android, iOS e BlackBerry) e está sendo incorporada em uma crescente variedade de dispositivos.

A arquitetura do M-Hub, conforme ilustrado na Figura 4.3, é composta por diversos serviços e gerenciadores locais, todos executando em *background* e concorrentemente com as aplicações móveis do usuário.

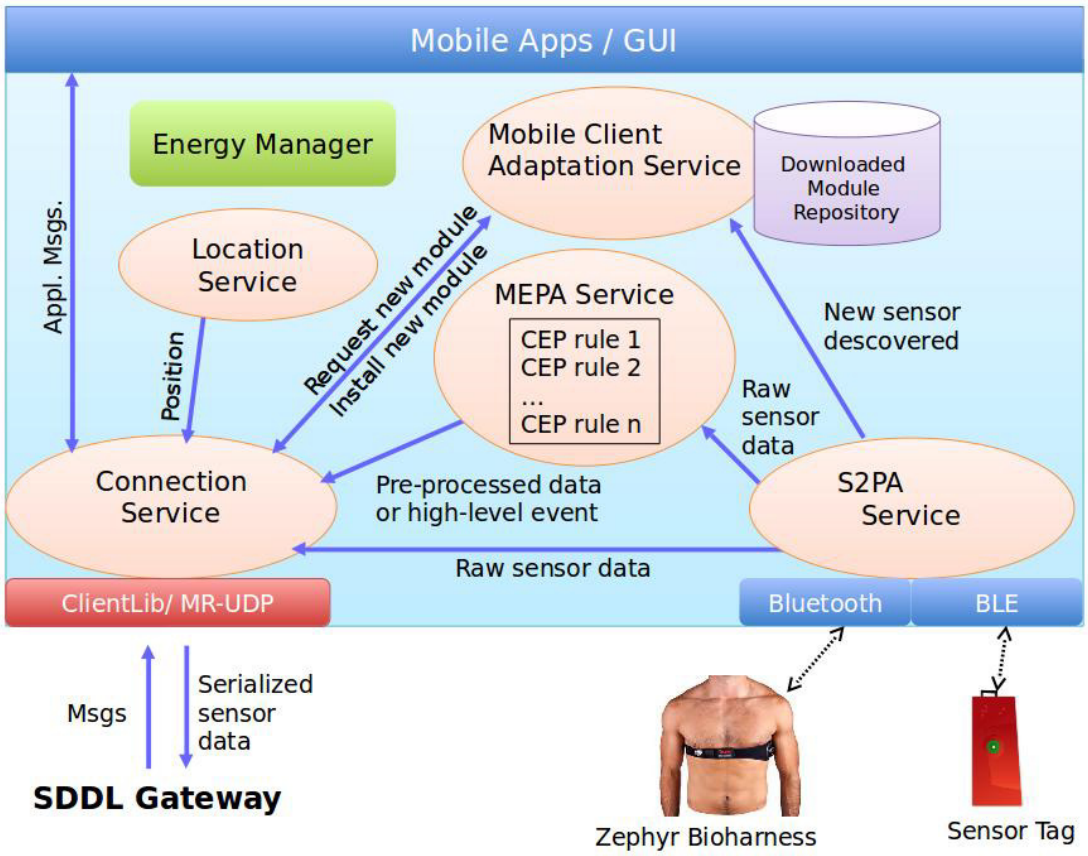

Figura 4.3: Arquitetura do M-Hub [48]

Os principais componentes da arquitetura do M-Hub são:

- **Location Service**: responsável por coletar a posição atual do M-Hub. Essa localização é incorporada em todas as amostras de dados publicados pelo M-Hub. Isso permite conhecer a localização de usuários e sensores próximos a ele;

- **S2PA** (*Short-range Sensing, Presence & Actuation Service*): é responsável pela descoberta, monitoramento e registro de sensores e atuadores nas proximidades, fazendo periodicamente varreduras para cada WPAN suportada;

- **Connection Service**: é responsável por estabelecer e manter uma conexão confiável com a rede, a fim de transmitir os dados coletados pelo S2PA para servidores na nuvem, utilizando para isso os serviços de distribuição da SDDL;

- **Energy Manager**: verifica o nível da bateria do dispositivo periodicamente e, de acordo com a classificação (alta, média e baixa), define as frequências da busca por novos sensores, da consulta ao serviço de localização e do envio dos dados ao gateway SDDL, de modo a minimizar o consumo de energia do dispositivo;

- **Mobile Client Adaptation Service**: responsável por instanciar e gerenciar o ciclo de vida dos códigos Java carregados dinamicamente. Ele é usado para a implantação de novos módulos de sensores e/ou funcionalidades da aplicação, que são baixados a partir de um serviço em execução na nuvem SDDL. Os módulos baixados são armazenados no dispositivo móvel, a fim de evitar o reenvio daqueles já presentes localmente, em caso de uma reinicialização do M-Hub;

- **MEPA (Mobile Event Processing Agent) Service**: responsável por processar continuamente os fluxos de dados gerados pelos sensores, a fim de procurar certos padrões de correlações e detectar eventos complexos utilizando a linguagem EPL Esper, sendo que a *engine* CEP emprega é a Asper [35].

4.2.2 Scalable Data Distribution Layer (SDDL)

O SDDL é um middleware de comunicação que conecta nodos estacionários em uma rede cabeada (a nuvem SDDL) a nodos móveis por meio de uma conexão baseada em redes IP sem fio. Na arquitetura do SDDL, mostrada na Figura 4.4, parte dos nodos estacionários localizados na nuvem SDDL são destinados ao processamento de dados de contexto. Alguns nodos são gateways, que servem de ponte para a comunicação entre a nuvem e os nodos móveis. Há também nodos de monitoramento e controle do sistema. Esses nodos exibem a posição atual dos nodos móveis (ou qualquer outra informação) e também são capazes de gerenciar grupos de nodos

e enviar mensagens para os nodos móveis individualmente ou em grupo. Nessa arquitetura, o M-Hub (ver Seção 4.2.1) é um tipo especial de nodo móvel, que descobre e se conecta oportunisticamente a sensores e atuadores próximos a ele, coletando dados de contexto desses objetos inteligentes via WPANs e enviando esses dados para a nuvem SDDL, utilizando redes locais ou a Internet, e/ou recebendo dados/comandos vindos da nuvem SDDL e retransmitindo-os ao objetos próximos [32, 47, 103, 104, 106].

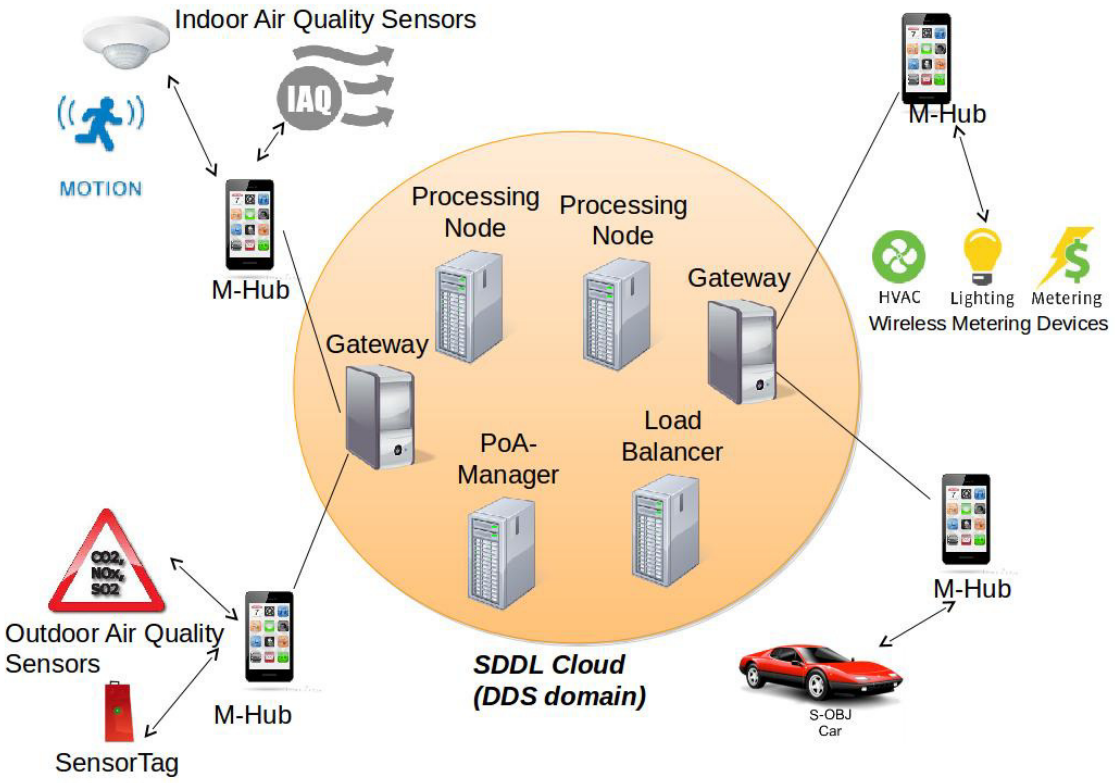

Figura 4.4: Arquitetura do SDDL [48].

O SDDL emprega dois protocolos de comunicação:

- **UDP confiável móvel (MR-UDP)**: protocolo usado na comunicação entre os nodos móveis e a nuvem. O MR-UDP implementa funcionalidades típicas do TCP, como confirmação de entrega e retransmissão de pacotes, mas em cima de uma pilha de protocolo UDP. MR-UDP é otimizado para conexões intermitentes, oferecendo suporte à travessia de NAT/Firewall e troca de endereço IP [106].
- **Data Distribution Service (DDS) / *Real-Time Publish-Subscribe* (RTPS)**: protocolo utilizado para comunicação local entre os nodos estacionários da nuvem SDDL. Trata-se um protocolo de comunicação *publish/subscribe* totalmente

descentralizado e escalável, com suporte a diversas políticas de QoS, tais como controle de confiabilidade, prazo, latência, histórico, ordenamento, etc. [85].

É importante frisar que, embora o MR-UDP tenha suporte à confiabilidade, essa QoS não é oferecida como política que possa ser configurada pela aplicação consumidora. Algo semelhante acontece no caso do DDS, pois embora esse protocolo tenha suporte a diversas políticas de qualidade de serviço, essas políticas não são tratadas como algo que o SDDL torne visível e/ou configurável por parte das aplicações.

Os nodos estacionários da nuvem SDDL podem ser de quatro tipos [106]:

- **Gateway**: define um único ponto de ligação (PoA - *Point of Attachment*) para a conexão dos nodos móveis com a nuvem SDDL. Ele é responsável por gerenciar uma conexão separada com cada nodo, bem como encaminhar qualquer mensagem dos nodos móveis para a nuvem SDDL e na direção oposta. Sendo o manipulador de conexões com os nodos móveis, o Gateway também é responsável por notificar outros nodos da nuvem SDDL quando um novo nodo móvel se torna disponível ou quando eles são desconectados. Esta informação é necessária para algumas funcionalidades da nuvem SDDL, tais como o armazenamento temporário de mensagens endereçadas a nodos móveis temporariamente *offline*, possibilitando a entrega posterior dessas mensagens;

- **PoA-Manager**: responsável por duas tarefas: distribuir periodicamente uma lista de Pontos de Ligação (PoA-List) para os nodos móveis e, eventualmente, solicitar que alguns nodos móveis mudem para um novo gateway/PoA (*handover*). A PoA-List é sempre um subconjunto de todos os gateways disponíveis em SDDL, e a ordem na lista é relevante, ou seja, o primeiro elemento aponta para o gateway/PoA preferencial e assim por diante. Por ter uma PoA-List atualizada, um nodo móvel pode sempre mudar seu gateway se detectar uma conexão fraca ou uma desconexão com o gateway atual. Além disso, através da distribuição de diferentes POA-List para diferentes grupos de nodos móveis, o PoA-Manager é capaz de equilibrar a carga entre os gateways, bem como anunciar para os nodos móveis quando gateways são adicionados ou removidos;

- **Processing Node**: servidor que estende o poder de processamento dos nodos móveis, sendo capaz de realizar tarefas de computação intensiva. Quando um nodo de processamento recebe um dado de contexto, ele verifica se é responsável pelo tratamento desse dado. Se for, o dado será processado de acordo com a lógica de processamento implementada. Do contrário, o dado será descartado, uma vez que será processado pelo(s) nodo(s) de processamento responsável(is) por ele;

- **Load Balancer**: servidor responsável pelo monitoramento da carga de trabalho dos nodos de processamento (*Processing Node*), a fim de redistribuir essa carga entre os demais nodos de mesma finalidade, caso haja grande desbalanceamento.

4.3 Message Queue Transport Telemetry (MQTT)

O MQTT (*Message Queue Telemetry Transport*)[6] é um protocolo *publish/subscribe* orientado a mensagens que se baseia na utilização de tópicos (assuntos) gerenciados por componentes que atuam como intermediários da comunicação entre publicadores e subscritores, denominados *brokers* [61] [54]. O MQTT foi adotado como protocolo padrão subjacente da camada de distribuição de dados provida pela solução proposta, a CDDL. Por esse motivo, esse protocolo é explicado mais detalhadamente que os demais protocolos apresentados anteriormente.

O MQTT é o protocolo de IoT padrão adotado pela OASIS[7] (*Organization for the Advancement of Structured Information Standards*) [83]. Esse protocolo foi criado em meados de 1999 por Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech). O MQTT foi projetado para aplicações de telemetria que utilizam dispositivos com recursos limitados (e.g. memória, processamento, bateria, conectividade) em ambientes de rede com baixa largura de banda, alta latência e pouco confiáveis. MQTT é considerado um protocolo leve, apresentando baixo *overhead* de comunicação. A tendência é que quanto mais leve for o protocolo de troca de mensagens, menos recursos de rede serão utilizados e menor será a quantidade de energia consumida na comunicação.

[6] `http://mqtt.org/`

[7] `https://www.oasis-open.org`

4.3.1 Arquitetura

A arquitetura do MQTT é ilustrada na Figura 4.5. Nota-se que essa arquitetura é implementada sobre o protocolo TCP, o que provê confiabilidade na entrega das mensagens em nível de transporte. Cada mensagem trocada entre os clientes e o *broker* é encapsulada em uma conexão TCP. O MQTT não especifica nenhuma técnica de roteamento ou rede. Ele simplesmente assume que a rede subjacente fornece um serviço de transporte orientado a sessão, com auto-segmentação e entrega em ordem, empregando esse serviço para a troca de mensagens.

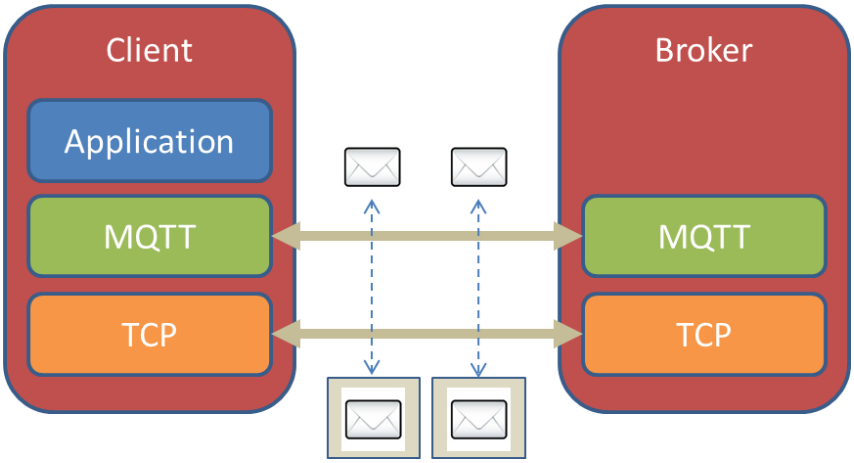

Figura 4.5: Arquitetura do MQTT baseada no protocolo TCP/IP [36]

O modelo comunicação do MQTT requer que cada cliente se conecte ao *broker* fornecendo seu próprio identificador único (ID do cliente). Opcionalmente pode-se utilizar mecanismos de autenticação baseados em usuário e senha, além de criptografia utilizando SSL (*Secure Sockets Layer*) / TLS (*Transport Layer Security*), que dependem do suporte oferecido pela implementação da biblioteca cliente e do *broker*. Quando um cliente não precisa receber quaisquer mensagens por um longo tempo, ele deve enviar periodicamente uma mensagem *keep-alive* para indicar ao *broker* que deve manter a conexão ativa, caso contrário o *broker* irá fechar a conexão após um tempo limite, que é especificado pelo cliente ao se conectar ao *broker*.

Há várias implementações de clientes e server *brokers* MQTT disponíveis para diversas linguagens de programação como Java, C, C#, Javascript e Python. Na implementação da solução proposta, por exemplo, utilizou-se o cliente MQTT Eclipse Paho Android Service 1.0.2[8] (versão para Android) e o *broker* Mosquitto 1.4.5 [9] (versão

[8] `www.eclipse.org/paho/`
[9] `http://mosquitto.org/`

para Linux), sendo ambos livres e de código aberto. O *broker* Mosquitto tem suporte a autenticação autorização e criptografia. Existem também versões reduzidas de *brokers*, voltadas para dispositivos móveis, tais como o Micro Broker Moquette[10]. Esse Micro Broker também foi utilizado na implementação da solução proposta (ver Seção 5).

4.3.2 Confiabilidade de Entrega

A confiabilidade de entrega é a única política de QoS provida pelo MQTT, sendo que, em nível de aplicação, esse protocolo oferece três níveis de confiabilidade: QoS 0 (*at most once*), QoS 1 (*at least once*), QoS 2 (*exactly once*).

Uma fragilidade do mecanismo de confiabilidade do MQTT é que as tentativas de retransmissão do publicador só duram enquanto a conexão deste com o *broker* perdurar. Caso a conexão falhe, o cliente deixa de tentar reenviar as mensagens. Além disso, o *buffer* de mensagens em trânsito (*in-flight*) no cliente MQTT é limitado, o que significa que as mensagens mais antigas poderão ser removidas do *buffer* antes da confirmação de entrega, a fim de dar lugar às mensagens mais recentes [69].

4.3.3 Configuração da QoS

Publicadores e subscritores definem suas configurações de confiabilidade de maneira independente. O publicador escolhe o nível máximo de confiabilidade que está disposto a prover, enquanto que o subscritor escolhe o nível mínimo de confiabilidade com o qual deseja receber as mensagens. Por exemplo, se uma mensagem é publicada com QoS 2, mas o subscritor assina o tópico usando QoS 0, então a mensagem será entregue a esse subscritor com QoS 0. Se um segundo subscritor, do mesmo tópico, escolheu a QoS 2, então ele receberá a mesma mensagem que o subscritor anterior, mas com QoS 2. Em outro caso, se o subscritor assina o tópico utilizando QoS 2, mas a mensagem é publicada em QoS 0, então essa mensagem será enviada ao subscritor (podendo não ser entregue) com QoS 0. Portanto, o nível de confiabilidade escolhido pelo subscritor só será atendido se ele for menor ou igual ao nível definido pelo publicador.

[10] `https://github.com/technocreatives/moquette`

A escolha do nível de confiabilidade mais adequado para cada aplicação dependerá dos seus requisitos e das características do ambiente de rede. Por exemplo, em cenários em que o ambiente de rede é estável e relativamente confiável, mas uma pequena taxa de perda de dados é aceitável, sendo que as informações são publicadas em intervalos curtos, então a QoS 0 pode ser empregada, visto que o requisito de obter um baixo tempo na transmissão da mensagem é mais importante que o de garantir a entrega a qualquer custo. Em cenários mais instáveis e exigentes, por exemplo, no qual a rede não é confiável mas a perda de dados não é tolerada sob quaisquer circunstâncias, pode-se usar QoS 1 ou QoS 2. A escolha nesse caso dependerá ainda se o recebimento de mensagens duplicadas interfere no bom funcionamento da aplicação.

4.3.4 Sessão Persistente

Quando um cliente se conecta a um *broker* MQTT, ele precisa criar assinaturas para todos os tópicos de interesse, a fim de receber mensagens do *broker*. Ao se reconectar, as assinaturas referentes a esses tópicos são perdidas e o cliente precisa se inscrever novamente. Este é o comportamento padrão do *broker* MQTT. Para evitar que clientes com recursos limitados tenham maiores custos com novas assinaturas a cada reconexão, o protocolo MQTT tem suporte a sessões persistentes que guardam informações de estado do cliente. A sessão é identificada pelo ID fornecido pelo cliente no ato da conexão.

Em sua sessão persistente, o *broker* deve armazenar os seguintes itens:

- ID do cliente, mesmo que não haja assinaturas.
- Assinaturas do cliente;
- Mensagens em trânsito com QoS 1 ou 2 que não foram confirmadas pelo cliente;
- Novas mensagens com QoS 1 ou 2 que o cliente perdeu quando estava *off-line*;
- Mensagens recebidas com QoS 2 que ainda não foram confirmadas para o cliente;

Isso significa que, mesmo se o cliente for desconectado, todos os itens acima serão armazenados pelo *broker* e estarão disponíveis logo após ele se reconectar.

Cada cliente do MQTT também deve manter uma sessão persistente do seu lado. Portanto, quando um cliente solicita ao *broker* que guarde os dados da sessão, ele também tem a responsabilidade de guardar algumas informações localmente:

- Mensagens em trânsito com QoS 1 ou 2 não confirmadas pelo broker;
- Mensagens recebidas com QoS 2 que não foram confirmadas ao broker.

A implementação da sessão persistente varia de acordo com o cliente e o *broker* MQTT utilizados. O volume de dados que podem ser armazenados na sessão dependerá de um conjunto de configurações de conexão tanto do cliente como do broker

4.3.5 Testamento

Outra característica presente no protocolo MQTT é o suporte ao envio de mensagens de testamento (*Last Will and Testament* - LWT). A LWT é uma mensagem que o cliente opcionalmente pode registrar junto ao *broker* no estabelecimento da conexão. Ao definir a mensagem LWT, o cliente deve indicar também o tópico no qual ela poderá ser publicada futuramente, denominado Tópico LWT. Caso o cliente se desconecte de forma anormal, ou seja, sem que ele tenha espontaneamente solicitado a desconexão ao *broker*, a mensagem de testamento será enviada pelo *broker* para todos os demais clientes ativos que assinaram o tópico LWT correspondente. Clientes podem usar esse recurso para detectar falhas em outros clientes da rede.

4.3.6 Retenção

O mecanismo de retenção do protocolo MQTT consiste simplesmente em manter armazenada no broker a última mensagem publicada de cada tópico. Para que isso aconteça, é necessário que o publicador marque a mensagem a ser enviada como “retained”. Mensagens com essa marcação poderão ser entregues a subscritores atrasados, ou seja, aqueles que assinaram o tópico depois da publicação da mensagem retida pelo *broker*. Esse mecanismo é útil, por exemplo em situações onde o subscritor atrasado não pode esperar até a próxima publicação para receber uma atualização. É

importante ressaltar que a retenção da mensagem por parte do *broker* não depende e nem tão pouco interfere na confiabilidade com a qual ela será publicada ou recebida.

4.3.7 Filtragem por Tópicos

Tópicos são cadeias de caracteres UTF-8 usadas pelo *broker* para filtrar mensagens para cada cliente conectado. Um tópico consiste em um ou mais níveis hierárquicos, sendo que cada nível de tópico é separado por uma barra invertida (“/”), como se fosse uma URI. Não há necessidade do cliente criar o tópico desejado antes de publicar ou se inscrever, porque o *broker* aceita cada tópico válido sem qualquer inicialização prévia. A seguir apresenta-se uma exemplo de tópicos simples:

Listing 4.4: Filtrando dados do sensor de temperatura do piso terreo da casa de Maria .

```
casademaria/pisoterreo/temperatura
```

Nesse exemplo, o tópico usado permite filtrar com exatidão os dados do sensor de temperatura da casa de Maria. O primeiro nível do tópico (“casademaria”) representa o domínio. O segundo nível (“pisoterreo”) equivale ao cômodo da casa a ser monitorado. O terceiro nível (“temperatura”) indica o tipo de sensor específico.

Para que seja válido, cada tópico deve ter pelo menos um caractere, podendo conter espaços (mas isso não é uma boa prática de modelagem). Os tópicos diferenciam letras maiúsculas de minúsculas (*case-sensitive*). Logo, os tópicos **“minhacasa/pisoterreo/temperatura”** e **“MinhaCasa/PisoTerreo/Temperatura”** são diferentes. Além disso, a barra invertida sozinha (“/”) também é um tópico válido.

Quando um cliente se inscreve em um tópico, ele pode usar o tópico exato em que a mensagem foi publicada (como mostrado nos exemplos anteriores) ou pode se subscrever em mais tópicos de uma só vez usando caracteres curingas (*wildcards*). Curingas só podem ser usados para assinar tópicos, não sendo permitidos ao publicar as mensagens. Existem dois caracteres curingas: nível único (“+”) e multinível (“#”).

Como o nome sugere, um curinga de nível único é um substituto para um nível de tópico. No exemplo a seguir mostra-se como usar esse tipo de curinga para filtrar dados de sensores de temperatura de todos os cômodos da casa de Maria.

Listing 4.5: Filtrando dados dos sensores de temperatura em quaisquer cômodos da casa de Maria .

```
casademaria/+/temperatura
```

Enquanto o curinga de nível único abrange apenas um nível de tópico, o curinga multinível cobre diversos níveis de tópicos. Uma restrição, para que a assinatura seja válida, é que que o curinga multinível seja sempre o último caractere no tópico e seja precedido por uma barra invertida, como mostra o exemplo a seguir.

Listing 4.6: Filtrando dados de todos os sensores da casa de Maria.

```
casademaria/#
```

Nesse exemplo, o curinga multinível é usado pra filtrar dados de todos sensores existentes na casa de Maria, independentemente do cômodo em que se encontra. A assinatura do exemplo seria equivalente a "**casademaria/+/+**".

Ressalta-se que, quando um cliente se subscreve em um tópico com um curinga multinível, ele passará a receber todas as mensagens cujos tópicos começam com o padrão antes do caractere curinga. Se o cliente especificar apenas o curinga multinível como um tópico ("#"), isso significa que ele receberá as mensagem publicadas em todos os tópicos. Nessas situações, o cliente deve estar ciente de que talvez ele receba um volume de mensagens muito elevado, devendo possuir recursos computacionais suficientes que o permitam processar a grande quantidade de dados.

4.4 Conclusão do Capítulo

Este capítulo apresentou três bases tecnológicas fundamentais o desenvolvimento da solução: CEP, ContextNet e MQTT.

Em relação ao CEP, apresentou-se como essa tecnologia pode ser empregada para detectar padrões de correlação e derivar situações à partir de análise de fluxos de eventos contínuos. Mostrou-se algumas vantagens do uso de CEP, tais como sua capacidade de processar um grande volume de eventos em tempo próximo ao tempo-real. Apresentou-se ainda os elementos que compõem uma rede de processamento de eventos e alguns aspectos das linguagens e *engines* utilizadas para o processamento de

eventos. Na solução proposta neste trabalho, CEP é base para o desenvolvimento dos mecanismos de avaliação, filtragem, monitoramento e descoberta de serviços baseada em QoC.

Em relação ao M-Hub e ao SDDL, apresentou-se como esses sistemas de middleware, desenvolvidos no âmbito do projeto ContextNet, pode ser empregado no desenvolvimento de aplicações móveis colaborativas. Explicou-se que M-Hub, um gateway de IoT para objetos inteligentes com recursos limitados, foi reutilizado e estendido no âmbito da pesquisa para o desenvolvimento da solução proposta. No entanto, optou-se por desenvolver uma nova camada de distribuição com suporte a QoC, a CDDL, em virtude da SDDL não estar focada provisionamento de qualidade.

Por fim, foi apresentado o protocolo de comunicação utilizado pela CDDL: o MQTT, que se trata de um protocolo muito leve e projetado para o uso em redes de comunicação com baixa largura de banda e alta latência. Mostrou-se algumas características desse protocol, tais como o suporte a confiabilidade, retenção de mensagens e sessão persistente.

5 Solução

A M-Hub/CDDL[1] é uma abordagem de middleware de contexto voltada para o desenvolvimento de aplicações de IoT que possuem requisitos de gerenciamento (aquisição, processamento e distribuição) e qualidade de contexto (provisionamento, avaliação, descoberta e monitoramento de QoI e QoS). As próximas Seções apresentam uma visão geral, os requisitos, a arquitetura, os componentes e serviços, bem como as abstrações de programação e aspectos de implementação da solução M-Hub/CDDL. As conclusões estão no final do Capítulo.

5.1 Visão Geral

Para facilitar o desenvolvimento de aplicações de IoT que exigem QoC, a M-Hub/CDDL combina conceitos e serviços de um gateway móvel de IoT (M-Hub) com uma camada de distribuição de dados (CDDL). O M-Hub é responsável pela aquisição de dados com base na descoberta e interação com objetos inteligentes. A SDDL provê abstrações e serviços que permitem que a aplicação especifique, controle e monitore requisitos de qualidade da informação e do serviço de distribuição de dados. A solução sugere que os desenvolvedores usem as camadas M-Hub e a CDDL de forma integrada. Porém, essas camadas foram projetadas de forma que podem ser executadas independentemente uma da outra, ou seja, o M-Hub pode divulgar dados usando outras camadas de distribuição (além da CDDL) e a CDDL pode publicar dados coletados usando outras camadas de aquisição. Como foi dito na Seção 4.2, o M-Hub foi originalmente desenvolvido por pesquisadores do LAC/Puc-Rio. O M-Hub foi reutilizado para a implementação da solução proposta, sendo estendido no âmbito deste trabalho para dar suporte a novas WPANs e sensores, coletar metadados de QoC e adicionar novas políticas de interação com sensores. A principal extensão provida ao M-Hub foi a inclusão da CDDL como camada de distribuição alternativa à SDDL. Embora projetada pelo aluno de doutorado, a implementação e os testes da CDDL receberam colaborações de alunos de mestrado do LSDi/UFMA e do LAC/Puc-Rio.

[1] http://www.lsdi.ufma.br/projetos/cddl/doku.php

Entretanto, a contribuição desta tese não está na implementação do middleware em si, mas nos conceitos apresentados pela abordagem de middleware proposta visando o gerenciamento da QoC.

O M-Hub é executado em dispositivos móveis pessoais (por exemplo, *smartphones* e *tablets*) com sistema operacional Android. Atualmente, existem duas versões da CDDL, uma para dispositivos Android e outra para computadores pessoais, estações de trabalho e servidores com uma Maquina Virtual Java (*Java Virtual Machine* - JVM) instalada. A interação entre produtores e consumidores de dados de contexto locais ou remotos segue o modelo de comunicação *publish/subscribe* intermediada por *brokers* e baseada em tópicos. A CDDL implementa a comunicação distribuída usando o protocolo *Message Queuing Telemetry Transport* (MQTT) [54] (Ver Seção 4.2.1).

A Figura 5.1 ilustra o uso da infraestrutura M-Hub/CDDL em um cenário de *Ambient Assisted Living* (AAL). Nesse cenário, os pacientes monitorados usam uma rede de sensores corporais, que fornece dados relacionados ao seu estado de saúde, e um *smartphone*, que executa o middleware M-Hub/CDDL para coletar dados de sensores e distribuí-los através de brokers localizados em uma nuvem. Os dados coletados podem ser entregues aos médicos, cuidadores e familiares do paciente. Este cenário é um exemplo de Internet das Coisas Móveis (IoMT), uma vez que assume requisitos de mobilidade irrestrita de sensores portáteis/vestíveis, gateways locais e aplicações consumidoras executando em dispositivos móveis.

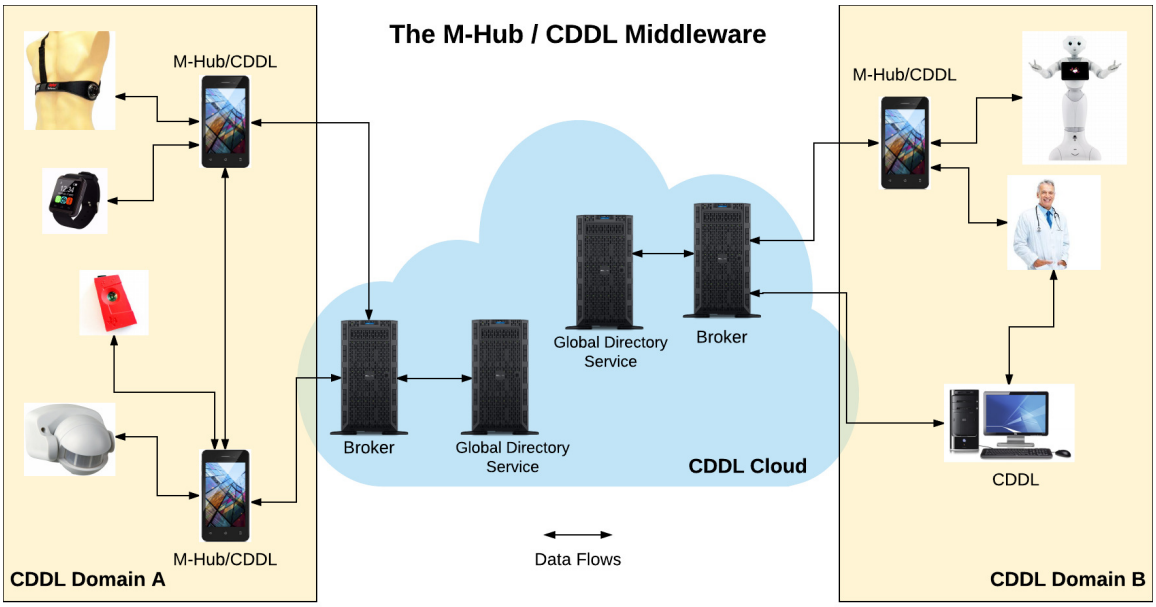

Figura 5.1: Visão Geral do M-Hub/CDDL em um cenário de AAL/IoMT

5.2 Requisitos

Mecanismos genéricos de gerenciamento e qualidade de contexto, em geral, são comuns a uma grande variedade de aplicações cientes de contexto, mas são complexos de serem implementados. A concepção da M-Hub/CDDL leva em consideração a necessidade de incorporar tais mecanismos à solução, de modo que o programador possa se concentrar na implementação de requisitos inerentes à lógica de negócio das aplicações de IoT. O resultado disso é a redução da complexidade.

Os requisitos atendidos pela solução desenvolvida são:

- **Heterogeneidade de Tecnologias**: a solução oferece mecanismos para a interação com sensores, a fim de tornar possível a aquisição direta de dados a partir de objetos inteligentes disponíveis no ambiente. Para tornar possível a interação com sensores que possuem recursos limitados de maneira *ad-hoc*, a solução provê suporte nativo à comunicação usando as tecnologias *Classic Bluetooth* e *BLE*, podendo ser facilmente estendida e integrada com outras tecnologias.

- **Distribuição de Dados Local e Remota**: a solução provê mecanismos de distribuição de dados de contexto, a fim de possibilitar o compartilhamento de informações entre produtores e consumidores de contexto. O middleware provê suporte tanto a distribuição local (produtores e consumidores executando no mesmo dispositivo) como a distribuição remota (produtores e consumidores executam em dispositivos distintos).

- **Descoberta Distribuída de Serviços**: a solução provê mecanismos que permitem descrever as características dos serviços de contexto disponíveis, a fim de que possam ser registrados em diretórios locais, que armazenam dados de serviços disponíveis no mesmo dispositivo, ou globais, que armazenam na nuvem dados de serviços pertencentes a um ou mais produtores de um domínio: o domínio CDDL. As aplicações descobrem os serviços disponíveis por meio do recebimento de anúncios espontâneos dos serviços registrados ou por meio da realização de consultas aos diretórios. A consulta é escrita em uma linguagem que permite expressar requisitos de contexto e de QoC da aplicação.

- **Modelo de Programação Uniforme e Transparente**: esse requisito expressa que, independentemente da localização dos produtores, consumidores, diretórios e

brokers, sejam eles locais ou remotos, as interfaces de programação utilizadas para registro, publicação, consulta e subscrição de dados de contexto são uniformes, ou seja, a maneira de programar uma aplicação que consome dados de contexto de origem local é a mesma utilizada para desenvolver uma aplicação que consome dados de contexto de origem remota.

- **Provisionamento e Monitoramento de QoI**: a solução provê mecanismos que permitem inserir na informação de contexto o valor de determinados atributos que descrevem a qualidade dos dados (metadados de QoI) e/ou meios para que o middleware possa calculá-los dinamicamente. Os mecanismos de registro e descoberta de serviços levam em consideração a QoI. Por um lado, os diretórios de serviços estão cientes da qualidade das informações disponíveis. Por outro lado, as consultas de contexto permitem também a especificação de requisitos de QoI dos consumidores. Além disso, são providos mecanismos de monitoramento, que permitem que as aplicações consumidoras sejam notificadas quando variações significativas na QoI ocorrem.

- **Provisionamento e Monitoramento de QoS**: a solução provê mecanismos que permitem que produtores e consumidores configurem políticas que definem a qualidade do serviço de distribuição dos dados de contexto, possibilitando que eles controlem como e quando as informações de contexto serão publicadas, recebidas e armazenadas. Os mecanismos de registro e descoberta de serviços levam em consideração também a necessidade de armazenar e consultar informações sobre a qualidade da distribuição. Para algumas políticas de QoS, são providos mecanismos de monitoramento, permitindo que publicadores e subscritores sejam notificados quando a qualidade exigida deixa de ser atendida.

- **Suporte a Confiabilidade em Redes Instáveis**: a fim de dar suporte à distribuição de dados de contexto para aplicações e gateways que executam em ambientes móveis com conectividade intermitente, a solução provê mecanismos de gerenciamento de conexão, sessão, confiabilidade de entrega, e *bufferização*, que garantem o armazenamento de todas as mensagens não publicadas ou não confirmadas, de modo a (ré)enviá-las tão logo a conexão seja reestabelecida.

5.3 Arquitetura

Os requisitos da seção anterior são atendidos por diferentes componentes da arquitetura do M-Hub/CDDL, ilustrada na Figura 5.2. Essa arquitetura contempla alguns componentes da camada M-Hub (`S2PA`, `BT Technology`, `BLE Technology`, `Internal Technology`, `Technology Device`) e da CDDL (`QoC Evaluator`, `Local Directory Service`, `Publisher Impl`, `Subscriber Impl`, `Monitor Impl`, `Filter Impl`, `Connection Impl` e `Micro Broker`), em seus devidos pacotes. Todos os componentes da CDDL foram desenvolvidos no âmbito desta tese. Alguns deles implementam mecanismos que visam resolver problemas apresentados no Capítulo 1. Os componentes pintados de cinza são aqueles que executam algum tipo de processamento de eventos complexos, utilizando para isso a linguagem EPL da Esper e a *engine* Asper [35] (versão *mobile* da Esper). Embora M-Hub e CDDL possam ser usadas de maneira isolada, a arquitetura mostra como ambas podem ser interligadas. Na arquitetura constam ainda as interfaces usadas pelas aplicações cientes de contexto (`MHubCDDL`, `Publisher`, `Subscriber`, `Connection`, `Monitor` e `Filter`). A seguir, apresenta-se os componentes de cada camada.

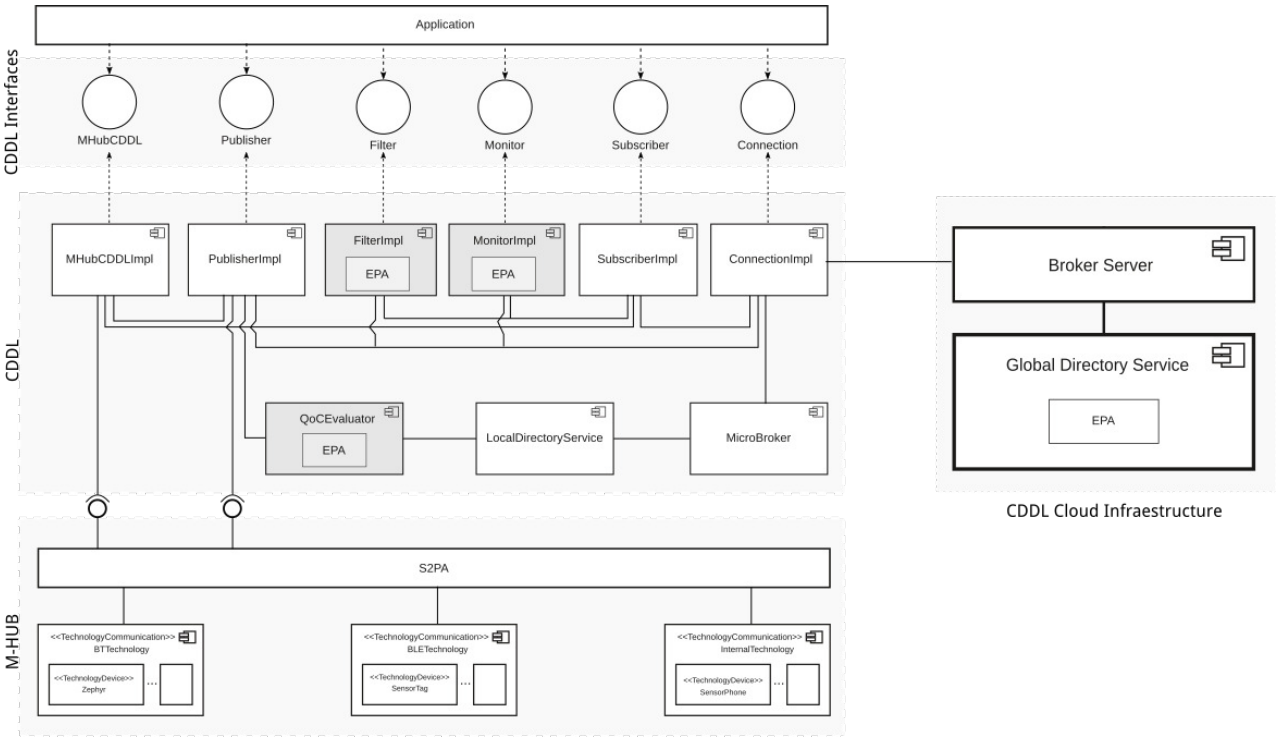

Figura 5.2: Arquitetura do M-Hub/CDDL

5.3.1 Componentes da Camada Mobile Hub

Os componentes e serviços da camada M-Hub são:

- **S2PA (Short-Range Sensor, Presence and Actuation)**: serviço responsável pelo gerenciamento das operações de descoberta, conexão e interação com objetos inteligentes. O S2PA é capaz de comunicar com dispositivos que utilizam diferentes tecnologias para o compartilhamento de dados por meio da interface *Technology Communication*. Graças a essa interface genérica, o S2PA não precisa conhecer detalhes do hardware e dos protocolos de comunicação. Os dados brutos recebidos pelo S2PA são pré-processados e enriquecidos com metadados de QoI. Em seguida esses dados são encaminhados à CDDL para distribuição. Por padrão, o S2PA coleta dados de todos os sensores com os quais consegue interagir, a fim de dar suporte à aquisição oportunista de dados em cenários de IoMT. No âmbito desta tese, chegou-se à conclusão de que há situações onde a ativação seletiva ou sob demanda de sensores é necessária, principalmente em situações onde se quer poupar recursos computacionais como memória, processamento e bateria em dispositivos móveis. Por isso, nesta tese, o S2PA foi estendido, incluindo uma política de habilitação de sensores por demanda. O produtor escolhe a política que deseja utilizar.

- **BT Technology e BLE Technology**: componentes que implementam as funcionalidades necessárias para a interação com dispositivos que se comunicam por meio das tecnologias WPAN Bluetooth Classic (e.g., Zephyr Bioharness[2]) e Bluetooth Low Energy (e.g., Sensor Tag [3]), respectivamente, tendo como base as operações definidas pela interface *Technology Communication*. O suporte a Bluetooth Classic e sensores Zephyr HXM e Bioharness foi implementado no âmbito dos trabalhos da tese. O suporte a BLE já estava disponível.

- **Internal Technology**: componente que implementa as funcionalidades de interação com sensores embutidos no mesmo dispositivo Android em que executa o M-Hub. Esse suporte foi implementado no âmbito do trabalho da tese.

[2] `https://www.zephyranywhere.com/`

[3] `http://processors.wiki.ti.com/index.php/SensorTag_User_Guide`

5.3.2 Componentes e Serviços da Context Data Distribution Layer

Os componentes e serviços desenvolvidos para a camada CDDL são:

- **Publisher Impl**: componente responsável pela publicação de dados, que podem ser oriundos do S2PA ou fornecidos pela camada de aplicação. Do ponto de vista das aplicações consumidoras, cada *Publisher Impl* instanciado é um provedor de serviços de contexto. Além de dados de contexto enriquecidos com metadados de QoI, esse componente também pode publicar consultas e respostas de contexto. A entrega dos dados ao *broker* é realizada de acordo com um conjunto de parâmetros e políticas de QoS implementadas pelo componente;

- **Subscriber Impl**: componente responsável pela subscrição em tópicos. Além do recebimento de dados de contexto enriquecidos com meta-dados de QoI, o *Subscriber* também pode ser usado para receber consultas e respostas de contexto. A entrega dos dados à camada de aplicação é feita de acordo com alguns parâmetros e políticas de QoS implementadas pelo componente;

- **Local Directory Service**: componente responsável pelo gerenciamento dos registros dos provedores de serviços de contexto que executam no próprio dispositivo. Esses registros incluem dados sobre os tipos de contexto disponíveis, a qualidade média das informações publicadas e o nível da qualidade do serviço de distribuição oferecido pelo provedor;

- **Connection Impl**: componente responsável pelo gerenciamento de conexões e sessões estabelecidas pelos clientes com os *brokers*, sejam locais ou remotos;

- **Micro Broker**: A função de um broker é intermediar a comunicação entre publicadores e subscritores, com base na definição de sequências e filtros de tópicos. O Micro Broker é versão reduzida do *Broker Server* Moquette[4], voltada para dispositivos móveis da plataforma *Android*. Portanto, além de intermediar a comunicação entre publicadores e subscritores locais (que executam no mesmo dispositivo móvel), o Micro Broker também permite que os dados de contexto sejam enviados para subscritores remotos (que executam em dispositivo móvel distinto daquele em que executa o publicador), conectados a esse broker, sem

[4] `https://github.com/technocreatives/moquette`

a necessidade de um Server Broker. Naturalmente, é necessário que ambos os dispositivos móveis (o do publicador e o do subscritor) estejam conectados à mesma WLAN.

- **QoC Evaluator**: componente que recebe dados de contexto enriquecidos com meta-dados de QoI vindos do S2PA. Uma das responsabilidades do `QoC Evaluator` é computar dinamicamente o valor de alguns parâmetros de QoI, que não são fornecidos na etapa de aquisição. Sua outra função é calcular periodicamente a qualidade média dos dados de contexto fornecidos por cada sensor. Essa média é uma informação relevante para algumas aplicações consumidoras, que podem adotá-la como um critério de seleção dos serviços de contexto disponíveis. Essa informação é armazenada nos diretórios de serviços. Após a avaliação das QoIs, estas são adicionadas na forma de metadados às informações de contextos que são encaminhadas ao componente *Publisher*;

- **Filter Impl**: componente que filtra dados de contexto e meta-dados de QoI usando consultas CEP. Esse componente pode ser usado tanto pelo *Publisher*, para definir quais dados podem ser publicados, como pelo *Subscriber*, para definir quais dos dados recebidos podem ser repassados para a aplicação consumidora. A filtragem por conteúdo não substitui a filtragem por tópicos executada pelos *brokers*, sendo apenas uma segunda camada de filtragem;

- **Monitor Impl**: componente responsável pelo monitoramento de eventos de interesse da aplicação utilizando consultas CEP. Para cada consulta de monitoramento é associado um *listener* pelo qual a aplicação é notificada em caso de ocorrência do evento. Este componente pode ser usado para o monitoramento de QoI e QoS. Algumas políticas de QoS, especificamente, são automonitoradas, ou seja, a própria política possui um mecanismo de monitoramento padrão, que notifica a aplicação caso a qualidade do serviço seja violada;

- **Server Broker**: componente cuja função é intermediar a comunicação entre publicadores e subscritores conectados à nuvem. Esse componente possui maior capacidade de distribuição de dados que o *Micro Broker*, pois executa num hardware com recursos menos limitados;

- **Global Directory Service**: componente responsável pelo registro dos provedores de serviços conectados à nuvem que formam um domínio CDDL. A base de dados do *Global Directory Service* é atualizada a partir das listas de serviços disponíveis que são publicadas periodicamente pelo diretório local que é executado em cada nodo do domínio CDDL.

5.3.3 Interfaces da Context Data Distribution Layer

As principais interfaces que a CDDL expõe para a camada de aplicação são:

- **M-Hub/CDDL**: é uma interface usada pela aplicação para iniciar os principais serviços do middleware. Por meio dela, por exemplo, a aplicação produtora (aplicação que gerencia o gateway) pode ativar os sensores dos quais deseja publicar dados de contexto, bem como habilitar suas respectivas tecnologias de comunicação. A aplicação também pode usar essa interface para especificar o intervalo de medição dos sensores, mas essa configuração é valida para sensores cuja frequência pode ser configurada via software, como é o caso de alguns sensores do Android. Essa interface também é utilizada para inicializar o Micro Broker Local. Além disso, essa interface pode ser utilizada para obter instâncias dos componentes de conexão, publicação e subscrição de dados de contexto;

- **Conection**: interface utilizada pela aplicação para estabelecer conexões com brokers locais e remoto, bem como gerenciar aspectos de qualidade de serviço das conexões e mecanismos de autenticação, autorização e criptografia.

- **Publisher**: interface utilizada pela aplicação para publicar dados de contexto e consultas de serviços de contexto. Por meio dessa interface, a aplicação também fornece os parâmetros de configuração das políticas de qualidade de serviço inerentes ao envio dos dados de contexto;

- **Subscriber**: interface utilizada pela aplicação para subscrever em tópicos de dados, anúncios de serviços e respostas de contexto. Por meio dessa interface a aplicação também fornece os parâmetros de configuração das políticas de qualidade de serviço inerentes ao recebimento dos dados de contexto;

- **Filter**: interface utilizada pela aplicação para filtrar dados de contexto e QoC por meio da especificação de consultas EPL, que podem ser aplicadas ao fluxos de envio (no caso do publicador) ou de recebimento (no caso do subscritor).

- **Monitor**: interface utilizada pela aplicação para monitorar dados de contexto e QoC, tanto no fluxo de envio como no de recebimento dos dados de contexto. De modo semelhante à filtragem, a aplicação usa essa interface para fornecer uma consulta EPL.

Embora a MHub/CDDL seja uma abordagem de middleware que atende a diversos requisitos de aplicações de IoT, esta tese está mais focada nos desafios relacionados ao problema de pesquisa, que foram mostrados na Seção 1.2. A próxima Seção detalha a maneira como a solução desenvolvida aborda cada um daqueles desafios, separando os mecanismos inerentes a cada um deles em subseções específicas.

5.4 Gerenciamento de Sensores

O termo gerenciamento de sensores pode ter um sentido muito amplo na literatura. Nesta tese utilizamos esse termo simplesmente para designar a capacidade do middleware de contexto de habilitar e/ou desabilitar os recebimento de dados à partir de determinados sensores. Para lidar com o desafio da grande heterogeneidade de dispositivos e tecnologias de comunicação, com os quais os gateways e aplicações de IoT precisam interagir para coletar dados de contexto, a M-Hub/CDDL reutiliza e estende componentes e mecanismos providos pela versão original do M-Hub. A seguir, explica-se mais como foi estendido o suporte à heterogeneidade do M-Hub original.

Como explicado na Seção 4.2.2, o M-Hub original já provê um conjunto de interfaces genéricas, isto é, que definem um conjunto padrão de operações comuns a uma grande variedade de sensores e tecnologias de comunicação. Os programadores estendem essas interfaces e implementam os métodos de interação específicos de cada sensor, pois isso requer o uso de bibliotecas e *drivers* que variam de um sensor para outro. Ao implementar as interfaces pré-definidas, os componentes especializados podem ser gerenciados pelo S2PA. Logo, o M-Hub já define uma

metodologia de abstração de heterogeneidade própria, que esconde das camadas superiores os detalhes de baixo nível inerentes à comunicação do gateway com os objetos inteligentes, tais como funções de codificação e decodificação de pacotes de dados. O funcionamento do S2PA original consiste numa abordagem oportunista que funciona da seguinte maneira: uma vez iniciado, esse serviço ativa todas interfaces de comunicação suportadas pelo dispositivo móvel e promove uma varredura no ambiente, procurando todos os sensores e atuadores próximos disponíveis, inclusive aqueles embutidos no próprio dispositivo móvel. A vantagem dessa abordagem é que o gateway estará pronto para interagir com qualquer sensor que encontrar ao longo do caminho, desde que possua a tecnologia de comunicação compatível e que o *driver* do sensor esteja instalado, ou que este possa ser carregado dinamicamente.

Uma desvantagem da abordagem oportunista é que ela tende a consumir muitos recursos, pois a quantidade de dados a ser recebida e processada pelo gateway num curto espaço de tempo pode ser muito elevada, dependendo da quantidade de sensores disponíveis e da frequência das medições. Outro ponto de limitação é que a versão original do S2PA não define mecanismos que permitam que a aplicação gerenciadora do gateway habilite os sensores de acordo com os seus interesses de publicação de dados. Sendo assim, todos os dados recebidos pelo gateway são enviados pela rede, uma abordagem que não é muito adequada para gateways conectados a redes com baixa largura de banda e alta latência, além de não ser muito conveniente quando se tem preocupações com a privacidade dos dados de contexto.

A fim de implementar um gerenciamento de sensores mais inteligente e que poupe recursos do gateway, a MHub/CDDL estende o S2PA de modo a implementar uma política de habilitação seletiva de sensores e tecnologias de comunicação, que funciona da seguinte maneira: a partir da CDDL, a aplicação envia comandos para o S2PA que visam habilitar e desabilitar as interfaces que usam as tecnologias de comunicação e/ou sensores com base em identificadores, de maneira bastante flexível. Ao habilitar uma tecnologia, pode-se habilitar seletivamente qualquer sensor compatível com essa tecnologia. Ao desabilitar a tecnologia, desabilita-se todos os sensores compatíveis com essa tecnologia. Para ser mais flexível, foi incluído na política de gerenciamento a possibilidade de habilitar e desabilitar todos os sensores de uma só vez. Um aspecto inerente a essa política de gerenciamento é que, uma vez enviado o comando de habilitação de um sensor, o S2PA armazena esse

comando em sua cache. Sendo assim, mesmo que o sensor a ser habilitado não esteja disponível no momento de envio do comando de ativação, o S2PA poderá continuar procurando esse sensor, conectando-se a ele assim que o mesmo for encontrado. O mecanismos de gerenciamento também permite que seja especificada a frequência de medição do sensor. Entretanto, essa especificação de *QoD* só se aplica a sensores cuja implementação do *driver* permite tal configuração. Sensores embutidos em *smartphones* Android, por exemplo, aceitam esse tipo de configuração para vários sensores. Para habilitar e desabilitar sensores e tecnologias de comunicação, a aplicação deve utilizar a interface *MHubCDDL* (implementada pelo componente *MHubCDDLImpl*), ilustrada na Figura 5.3. Dentre outros métodos, essa interface define métodos para iniciar e parar os sensores e tecnologias de comunicação.

A Listagem 5.1 mostra um exemplo de utilização da interface `MHubCDDL`, em um cenário no qual a aplicação habilita a tecnologia *Bluetooth Classic* e posteriormente ativa o sensor de batimento cardíaco do dispositivo *Zephyr HXM*, compatível com essa tecnologia. Nesse exemplo, usa-se o método `getInstance()` para obter uma instância do componente `MHubCDDLImpl`. Os serviços do middleware são iniciados usando o método `startServices(Context context)`. Esse método recebe como parâmetro um objeto do tipo `Context`. A inicialização da tecnologia é feita usando-se o método `startCommunicationTechnology(int id)`, que recebe como parâmetro um identificador numérico da tecnologia. A inicialização do sensor é feita utilizando-se o método `startSensor(String name)`, que recebe como parâmetro o nome do sensor/serviço. Nos próximos exemplos, explicações sobre métodos já mostrados previamente poderão não se repetir.

Listing 5.1: Exemplo de Inicialização de Sensor (1).

```
//...Oculta trechos de codigo anteriores
//Obtem instancia da classe  MHubCDDL (Padrao Singleton)
MHubCDDL mHubCDDL = MHubCDDL.getInstance();
//Inicia os servicos do middleware, dentre eles o S2PA
mHubCDDL.startServices(context);
//Inicia a tecnologia Bluetooth Classic
mHubCDDL.startCommunicationTechnology(BTTechnology.ID);
//Inicia o sensor de batimento cardiaco do Zephyr HxM
mHubCDDL.startSensor("HXM030655_HeartRateSensor");
//...
```

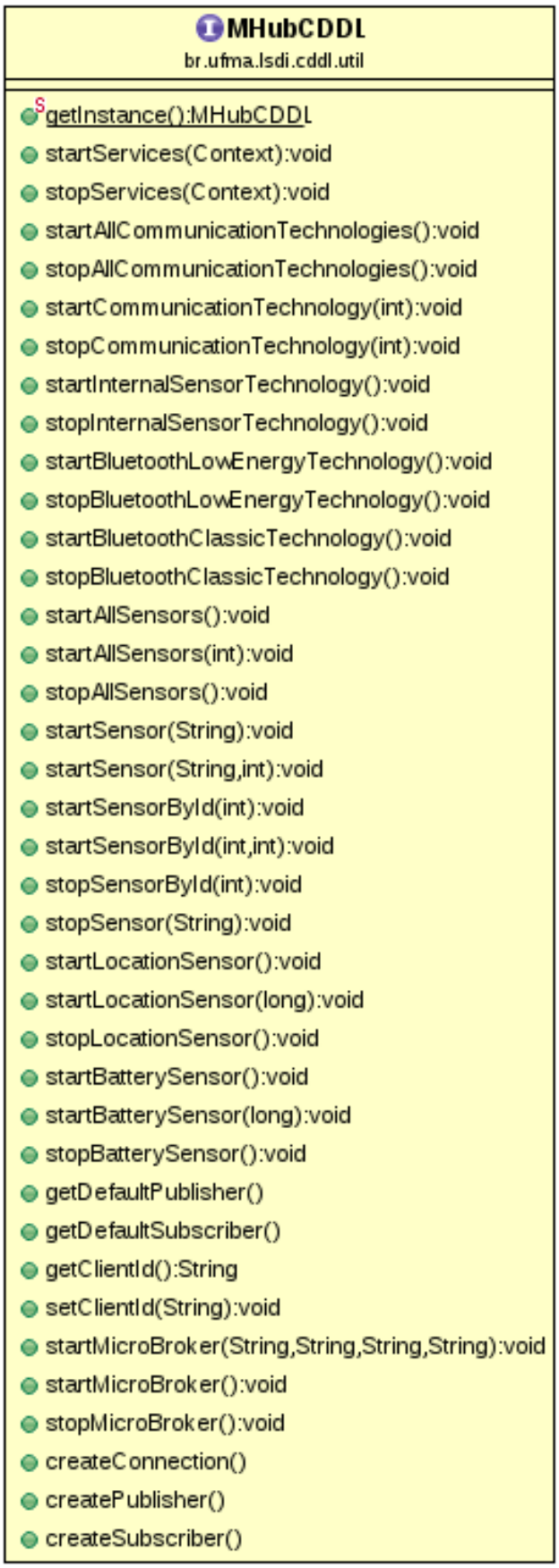

Figura 5.3: Interface M-hub/CDDL

A Listagem 5.2 mostra um exemplo no qual a aplicação habilita a tecnologia de sensores internos (`Internal Technology`) e posteriormente ativa o sensor de aceleração. Nesse exemplo, usa-se o método `startSensor(String name, int delay)`, que recebe como parâmetro o nome e a frequência de atualização do sensor a ser iniciado. Se a frequência fornecida é invalida, então o sensor funcionará com sua frequência normal.

Listing 5.2: Exemplo de Inicialização de Sensor (2).

```
//...Oculta trechos de codigo anteriores

//Obtem instancia da classe  MHubCDDL (Padrao Singleton)
MHubCDDL mHubCDDL = MHubCDDL.getInstance();

//Inicia os servicos do middleware, dentre eles o S2PA
mHubCDDL.startServices(context);

//Inicia a tecnologia de sensores internos
mHubCDDL.startCommunicationTechnology(InternalTechnology.ID);

//Inicia o sensor de aceleracao
mHubCDDL.startSensor("Accelerometer",SensorManager.SENSOR_DELAY_FASTEST);

//...
```

Listagem 5.3 mostra um exemplo no qual a aplicação desabilita o acelerômetro utilizando o método `stopSensor(String name)`, que recebe como parâmetro o nome do sensor a ser parado.

Listing 5.3: Exemplo de Inicialização de Sensor (3).

```
//...Oculta trechos de codigo anteriores

//Obtem instancia da classe  MHubCDDL (Padrao Singleton)
MHubCDDL mHubCDDL = MHubCDDL.getInstance();

//Para o acelerometro
mHubCDDL.stopSensor("Accelerometer");

//...
```

A Listagem 5.4 mostra um exemplo no qual a aplicação desabilita todos os sensores de uma única vez, independentemente da tecnologia de comunicação que foi anteriormente habilitada. Nesse exemplo, usa-se o método `stopAllSensors()`, a fim de parar todos os sensores e `stopAllCommunicationTechnologies()` de modo a parar todas as tecnologias de comunicação.

Listing 5.4: Exemplo de Inicialização de Sensor (4).

```
//...Oculta trechos de codigo anteriores
//Obtem instancia da classe  MHubCDDL (Padrao Singleton)
MHubCDDL mHubCDDL = MHubCDDL.getInstance();

//Para todos os sensores
mHubCDDL.stopAllSensors();

//Para todas os tecnologias de comunicacao
mHubCDDL.stopAllCommunicationTechnologies;
//...
```

A Listagem 5.5 mostra um exemplo no qual a aplicação inicia todas as tecnologias de comunicação e sensores, utilizando para isso os métodos `startAllCommunicationTechnologies()` e `startAllSensors()`, respectivamente. O efeito dessa configuração é igual ao do modo oportunista de interação padrão do M-Hub original.

Listing 5.5: Exemplo de Inicialização de Sensor (5)

```
//...Oculta trechos de codigo anteriores
//Obtem instancia da classe  MHubCDDL (Padrao Singleton)
MHubCDDL mHubCDDL = MHubCDDL.getInstance();

//Inicia todas as tecnologias
mHubCDDL.startAllCommunicationTechnologies();

//Inicia todos os sensores
mHubCDDL.startAllSensors();
//...
```

Uma limitação do mecanismo de gerenciamento de sensores da M-Hub/CDDL é que, com exceção dos sensores Android, ele não implementa ou adota

uma padronização ou ontologia específica de domínio para as nomenclaturas dos sensores, exceto para os sensores internos, cujos nomes são padronizados pela API Android, mas que nem sempre são adotados fielmente pelo fabricante do sensor. Logo, existe a possibilidade da aplicação habilitar e desabilitar simultaneamente um ou mais sensores que possuam o mesmo nome.

Além de permitir gerenciar os sensores e tecnologias de comunicação utilizados pela aplicação, o M-Hub/CDDL permite gerenciar também o Micro Broker de cada nodo móvel, ou seja, é possível as aplicações podem habilitar e desabilitar esse tipo broker dinamicamente. A Listagem 5.6 mostra um exemplo no qual a aplicação inicia o Micro Broker e se conecta a ele.

Listing 5.6: Exemplo de Inicialização de Micro Broker.

```
//...Oculta trechos de codigo anteriores
//Obtem instancia da classe  MHubCDDL (Padrao Singleton)
MHubCDDL mHubCDDL = MHubCDDL.getInstance();

//Inicializa o Micro Broker, fazendo que este aceite conexoes
mHubCDDL.startMicroBroker();

//informa o ID do cliente que sera usado por todas as conexoes
mHubCDDL.setClientId("bertodetacio@gmail.com");

//Obtem a instancia da fabrica de conexoes
ConnectionFactory connectionFactory = ConnectionFactory.getInstance();

//obtem uma instancia da conexao
Connection connection = connectionFactory.createConnection();

//Informa a URL do Micro Broker local
connection.setHost("localhost");

//Conecta
connection.connect();
```

5.5 Provisionamento de Qualidade de Contexto

Para lidar com o desafio da grande diversidade de requisitos de QoC exigidos pelas aplicações de IoT emergentes, a M-Hub/CDDL provê uma grande variedade de parâmetros de qualidade tanto da informação (QoI) como do serviço de distribuição (QoS). Apresenta-se a seguir os mecanismos inerentes ao provisionamento de cada tipo de qualidade de contexto.

5.5.1 Provisionamento de Qualidade da Informação

Para prover qualidade de informação para as aplicações de IoT, o middleware define um conjunto inicial (ou padrão) de atributos de QoI que fazem parte do seu modelo de contexto, mais precisamente do modelo de mensagem que a CDDL pode publicar. A M-Hub/CDDL adota uma abordagem híbrida de provisionamento de qualidade, que combina o provisionamento manual com o provisionamento automático de atributos de qualidade. O provisionamento manual consiste em deixar que o produtor informe o valor de todos os atributos de QoI, exceto aqueles para os quais não faz sentido atribuir um valor estático, pois devem ser calculados dinamicamente. O provisionamento automático é aquele em que o próprio middleware obtém de alguma forma ou computa o valor de determinados parâmetros, inserindo-os na mensagem caso o produtor não o faça. Além disso, o provisionamento automático visa calcular métricas de avaliação da QoC, independentemente de o valor dos atributos terem sido fornecidos pela aplicação ou computados/atribuídos pelo middleware. A seguir apresenta-se a lista inicial de atributos de QoI que fazem parte do modelo de mensagens publicadas pela CDDL. Para cada parâmetro de QoI, informa-se se o provisionamento é manual ou automático.

- **Acurácia**: é uma estimativa de quão perto a informação de contexto obtida está do valor real. Esse atributo pode ser informado pelo produtor, quando disponível. Entretanto, para sensores internos do Android, o S2PA tem suporte à obtenção automática do valor da acurácia dos dados, dispensando do produtor a tarefa de informar esse atributo de QoI. A acurácia como atributo não é computada pelo `QoC Evaluator`, devido à natureza heterogênea dos

sensores. Contudo, a média da acurácia é computada periodicamente pelo `QoC Evaluator`.

- **Confiança**: é um estimativa do grau de certeza que o produtor tem que a informação de contexto informada está correta. Sensores físicos como o *Zephyr Bioharness*, por exemplo fornecem essa estimativa. O produtor deve informar manualmente o valor da confiança no ato da publicação. A média da confiança é calculada dinamicamente pelo `QoC Evaluator`, podendo também ser divulgada para as aplicações consumidoras ou consultada por elas.

- **Localização da Fonte**: indica a posição aproximada da fonte de contexto no instante da medição. Se o produtor não informar a localização da fonte, então o `QoC Evaluator` irá atribuir esse valor automaticamente, com base na última leitura fornecida pelo sensor de localização. Essa abordagem é considerada pertinente, pois os sensores externos que se comunicam via WPAN estarão num raio próximo ao gateway. No caso dos sensores internos, a localização será obviamente ainda mais precisa, já que estão embutidos no próprio gateway. Pela natureza dessa QoI, não é calculada uma média para ela. Entretanto, os consumidores de contexto poderão efetuar consultas de serviços levando em consideração a última localização conhecida do provedor.

- **Tempo de Medição** indica o tempo aproximado em que a informação foi mensurada pelo sensor. Para sensores internos do Android o S2PA já obtém automaticamente esse atributo de QoI, dispensando o produtor de informá-lo. Entretanto, sensores externos geralmente não possuem um relógio embutido, de modo que o produtor não consegue obter e informar o instante exato da medição. Nesse caso, se a informação chegar ao `QoC Evaluator` sem o tempo de medição, esse componente atribuirá ao tempo de medição o mesmo valor do tempo de chegada da informação no publicador. Essa abordagem é considerada pertinente já que o tempo de transmissão do dado via WPAN é relativamente curto se comparado ao da transmissão do dado pela Internet.

- **Tempo de Chegada** tempo de chegada da informação de contexto no consumidor. O valor desse atributo é computado automaticamente pelo middleware.

- **Tempo de Validade**: indica o período máximo pelo qual a informação de contexto pode ser considerada válida, contado a partir do tempo de medição. O tempo de validade deve ser especificado pelo produtor da informação, podendo ser considerado ou não pelas aplicações consumidoras. Ao receber a informação de contexto, o consumidor pode aceitar o tempo de validade vigente ou até mesmo estender o prazo de acordo com seus próprios interesses.

- **Idade**: parâmetro calculado dinamicamente pelo middleware com base na diferença entre o tempo de medição e o instante atual. Quando o gateway local e as aplicações consumidoras executam em dispositivos diferentes, então é necessário que haja um mecanismo de sincronização (e.g., ClockSync) [5] entre ambos, de modo que a idade da informação de contexto seja calculada com a maior precisão possível.

- **Intervalo de Medição**: intervalo de separação entre duas leituras consecutivas. Esse atributo é calculado dinamicamente pelo `QoC Evaluator`.

- **Lista de Atributos**: lista de atributos que compõem a informação de contexto. Por exemplo, informações de localização fornecidas por GPS são compostas por três atributos: latitude, longitude e altitude. Se a informação de contexto não é composta, então sua lista de atributos de contexto é vazia. A lista de atributos deve ser informada pelo produtor, exceto para alguns sensores internos de movimentação, orientação e localização, para os quais o middleware obtém automaticamente a lista de atributos da informação de contexto.

- **Atributos Disponíveis**: corresponde à diferença entre a quantidade de atributos declarados e a quantidade de atributos cujo valor foi realmente informado pelo produtor. Por exemplo, sensores de localização de diversos *smartphones* Android, apesar do modelo de dados de localização considerar latitude, longitude e altitude como atributos do objeto `Location`, devido ao tipo de hardware utilizado e outros fatores, só conseguem fornecer o valor dos atributos latitude e longitude. Logo, embora a lista de atributos previstos para essa informação seja três, a quantidade de atributos disponíveis dessa lista é de apenas dois. A quantidade de atributos disponíveis é calculada automaticamente pelo `QoC Evaluator`.

[5] `https://play.google.com/store/apps/details?id=ru.org.amip.ClockSync&hl`

- **Completude**: o grau de disponibilidade das informações de contexto, calculada automaticamente pelo `QoC Evaluator` com base na razão entre a quantidade de atributos declarados e a quantidade de atributos disponíveis. Por exemplo, se dados de aceleração têm como atributos declarados os eixos X,Y e Z, mas apenas os eixos X e Y são fornecidos pelo produtor, então a completude calculada pelo `QoC Evaluator` será de 66,6%.

- **Resolução**: é o grau de detalhamento ou granularidade da informação de contexto. Para dados de sensores, o `QoC Evaluator` consegue calcular automaticamente a resolução numérica das leituras, isto é, precisão em relação à quantidade de casas decimais. Por exemplo, um sensor que fornece leituras com uma casa decimal (38,5º) tem menor resolução numérica que um sensor que fornece leituras com duas casas decimais (38,99º), o que não significa que a segunda tenha maior acurácia que o primeiro. É importe explicar que não há como conhecer previamente o tipo de objeto que o produtor irá publicar, pois a CDDL não se resume a publicar dados de sensores físicos, mas também de sensores lógicos e dados da aplicação que podem ser de qualquer tipo de classe/objeto. Por esse motivo, a fim de conseguir calcular a resolução numérica, quando possível, a M-Hub/CDDL utiliza um mecanismo baseado em reflexão computacional, que analisa a hierarquia de classes do objeto a ser publicado. Se o resultado da análise indicar que se trata de um objeto de tipo numérico, então o `QoC Evaluator` será capaz de determinar a quantidade de casas decimais.

- **Atraso**: tempo decorrido entre o envio da informação pelo publicador e sua chegada ao subscritor. É computado automaticamente pelo middleware, que também calcula o atraso médio levando em consideração as leituras do histórico. Embora relacionado à QoS, esse atributo é inserido na informação de contexto para fins de monitoramento.

- **Tempo Total de Entrega**: tempo decorrido desde o instante da medição do dado e sua efetiva entrega para a aplicação. É computado automaticamente pelo middleware. Pelo mesmo motivo do parâmetro anterior, o tempo total de entrega é inserido como atributo da informação de contexto.

A Listagem 5.7 mostra um exemplo de como produtores usam a M-Hub/CDDL para criar e publicar mensagens contendo atributos de contexto

e de QoC. Nesse exemplo, a aplicação produtora cria uma mensagem do tipo `SensorDataMessage` (que estende a classe `Message`). Para informar o nome do serviço, que no exemplo é um acelerômetro, utiliza-se o método `setServiceName(String service)`. O valor do serviço é informado utilizando-se o método `setServiceValue(Object value)`. É importante frisar que esse método pode receber qualquer objeto serializável Java. No exemplo, ele recebe um vetor que contém os três eixos de aceleração (X,Y,Z). Os parâmetros de qualidade de contexto informados no exemplo são acurácia e lista de atributos, utilizado-se para isso os métodos `setAccuracy(Double value)` e `setAttributesList(String[] list)`. A publicação da mensagem é solicitada utilizado-se o método `publish(Message message)` da classe `Publisher`, que recebe como parâmetro um objeto do tipo `Message`. A avaliação de QoC é um processo transparente para a aplicação. Note que o exemplo oculta códigos pertinentes à inicialização dos serviços do middleware, aquisição dos dados de contexto a partir dos sensores, conexão com o broker, criação de publicadores e subscritores. Além disso, como mencionado, a M-Hub/CDDL já coleta e publica automaticamente dados dos sensores internos quando habilitados.

Listing 5.7: Exemplo de como o produtor informa atributos de QoC.

```
//...oculta codigos anteriores
SensorDataMessage sensorDataMessage = new SensorDataMessage();
//Especifica o nome do servico
sensorDataMessage.setServiceName(new String("Accelerometer"));
//Informa o valor ou vetor de valores do servico
sensorDataMessage.setServiceValue(new Double[]{0.0,0.0.0.0});
//Informa a lista de atributos
sensorDataMessage.setAttributesList(new String[]{"X,"Y,"Z"});
//Informa a acuracia da leitura
sensorDataMessage.setAccuracy(new Double(0.0));
//Solicita a publicacao da mensagem ao Publisher
DefaultPublisher.getInstance().publish(sensorDataMessage);
```

A Listagem 5.8 mostra um exemplo de como o consumidor recebe a mensagem publicada no exemplo anterior e acessa seus atributos de contexto e de QoC. Nesse exemplo, a aplicação assina o serviço de dados de aceleração da gravidade usando o método `subscribeServiceByName(String service)` do

componente `Subscriber`, ao qual associa-se um `SubscriberListener`. Uma vez recebida a mensagem, a aplicação acessa os valores do serviço por meio do método `Object getServiceValue()`. Os atributos de QoC acurácia, lista de atributos, tempo de medição, latitude e longitude da fonte são obtidos por meio dos métodos `Double getAccuracy()`, `String [] getAttributesList()`, `Long getMeasurementTime()`, `Double getSourceLocationAltitude()` e `Double getSourceLocationLongitude()`. Note que, considerando o exemplo anterior, os valores do tempo de medição, latitude e longitude da fonte recebidos são aqueles calculados pelo `QoCvaluator` do publicador, já que não foram informados manualmente.

Listing 5.8: Exemplo de como o consumidor acessa de QoC da mensagem.

```
//...oculta codigos anteriores
//Cria o subscritor e posteriormente adiciona um listener
Subscriber subscriber = DefaultSubscriber.getInstance();
subscriber.setSubscriberListener(new ISubscriberListener(){
    public void onMessageArrived(Message message){
      //Faz o cast de Message para SensorDataMessage
      SensorDataMessage sensorDataMessage = (SensorDataMessage) message;
      //Obtem os valores da medicao
      Double [] values = (Double[]) sensorDataMessage.getServiceValue();
      // Obtem a lista de atributos
      String [] atributesList = sensorDataMessage.getAttributesList();
      //Obtem a acuracia
      Double accuracy = sensorDataMessage.getAccuracy();
      //ontem o instante da medicao
      Long measurementTime = sensorDataMessage.getMeasurementTime();
      //Obtem a localizacao da fonte
      Double latitude = sensorDataMessage.getSourceLocationAltitude();
      Double longitude = sensorDataMessage.getSourceLocationLongitude();
    };
});
//subscreve o servico
subscriber.subscribeServiceByName("Accelerometer");
```

5.5.2 Provisionamento de Qualidade de Serviço

Para prover qualidade de serviço para aplicações de IoT, a M-Hub/CDDL define um conjunto extensível de parâmetros e políticas de QoS. Uma política de QoS é um mecanismo transparente que controla o envio e o recebimento de dados de modo a atender requisitos diversos da aplicação, tais como confiabilidade, pontualidade, persistência e disponibilidade na distribuição de dados de contexto. O comportamento da política dependerá dos parâmetros de QoS que a configuram. Sendo assim, o provisionamento de QoS vai além de simplesmente tornar a aplicação ciente da qualidade do serviço, mas principalmente permitir que essa qualidade seja controlada para satisfazer os requisitos de produtores e consumidores.

Os parâmetros e políticas de qualidade de serviço são configurados de forma simétrica para ambos os lados da distribuição, ou seja, tanto no publicador como no subscritor. Por um lado, os parâmetros de QoS informados pelo produtor controlarão a forma como os dados são enviados para o broker. Por outro, os parâmetros de QoS informados pelo subscritor vão controlar a maneira como os dados serão recebidos do broker e entregues para a camada de aplicação. Os parâmetros/políticas providos pela M-Hub/CDDL são:

- **Prazo de Entrega**: define o tempo máximo pelo qual a aplicação está disposta a esperar para enviar ou receber pelo menos uma mensagem. Essa política monitora as filas de envio e recebimento na camada de middleware, notificando a aplicação caso o prazo especificado não seja cumprido.
- **Taxa de Atualização (Filtro baseado em Tempo)**: define o intervalo mínimo de separação entre chegadas sucessivas de mensagens, possibilitando assim que a aplicação controle a taxa de recebimento e processamento dos dados, independentemente da frequência com a qual eles são produzidos.
- **Controle de Latência**: permite que a aplicação defina um atraso adicional programado na distribuição de dados de contexto. Durante o período especificado, as mensagens produzidas são agrupadas a fim de que possam ser enviadas ou recebidas em uma única rajada.
- **Histórico**: define a maneira como as mensagens publicadas ou recebidas pela aplicação são armazenadas. Há dois modos: *keep all* e *keep last*. No primeiro, todas

as mensagens são armazenadas na cache. No segundo, apenas as *n* últimas são armazenadas. O histórico pode ser acessado por meio de operações *read* e *take*. A primeira obtém apenas uma cópia da mensagem armazenada, que pode ser lida novamente no futuro, pois a original permanecerá no histórico. A segunda remove a mensagem original do histórico, que não poderá ser lida novamente.

- **Ordem de Destino**: define se as mensagens armazenadas no histórico da aplicação serão organizadas com base no tempo da publicação ou do recebimento da mensagem. Se o subscritor quiser considerar o tempo da publicação, então deverá estar ciente de que existe a necessidade de utilizar um mecanismo de sincronização entre os relógios do produtor e do consumidor.

- **Vida Útil**: remove do histórico da aplicação as mensagens cujo tempo de validade expirou. O tempo de validade pode ser especificado pelo publicador. Entretanto, o subscritor pode ignorar esse tempo de validade e até mesmo estabelecer um prazo de validade adicional.

- **Retenção (Durabilidade)**: define que o broker deve reter sempre a última mensagem publicada em cada tópico. Se novos subscritores surgirem, então a mensagem retida será enviada imediatamente a eles, evitando que os mesmos tenham que esperar até a próxima publicação para receber uma mensagem. Essa política é baseada no mecanismo de retenção padrão do MQTT (ver Seção 4.3.6)

- **Vivacidade**: política utilizada por clientes que desejam ser notificados quando outros clientes perderem a conexão com o broker devido a falhas. Esse mecanismo é baseado em mensagens *Last Will and Testament* - LWT do protocolo MQTT (ver Seção 4.3.5)

- **Confiabilidade**: define o nível de confiabilidade empregado pela CDDL na entrega das mensagens. Essa política foi implementada com base nos três níveis de confiabilidade providos pelo protocolo MQTT: *at most once*, *at least once* e *exactly once* (Ver Seção 4.3.2). Entretanto, essa confiabilidade é incrementada pela utilização de um *buffer* intermediário, cujo funcionamento será explicado na Seção 5.6

- **Sessão**: define se a sessão mantida entre o cliente e o broker é persistente ou não. Esse mecanismo é baseado no gerenciamento de sessões implementado pelo broker MQTT (Ver Seção 4.3.4).

Do ponto de vista da aplicação, cada política de qualidade de serviço é uma classe que contém métodos específicos para a configuração dos parâmetros de QoS. Com exceção da política Sessão, todas políticas herdam da classe `AbstractQoS`, que pode ser utilizada para implementar novas políticas. Uma vez definidas, usa-se as interfaces `Publisher` e `Subscriber` para alterar as políticas do publicador e do subscritor, respectivamente.

A Figura 5.4 mostra um diagrama de classe das políticas de QoS. As Figuras 5.5 e 5.6 mostram as interfaces do publicador e do subscritor, respectivamente. Nessas figuras é possível observar os métodos de configuração das políticas de QoS.

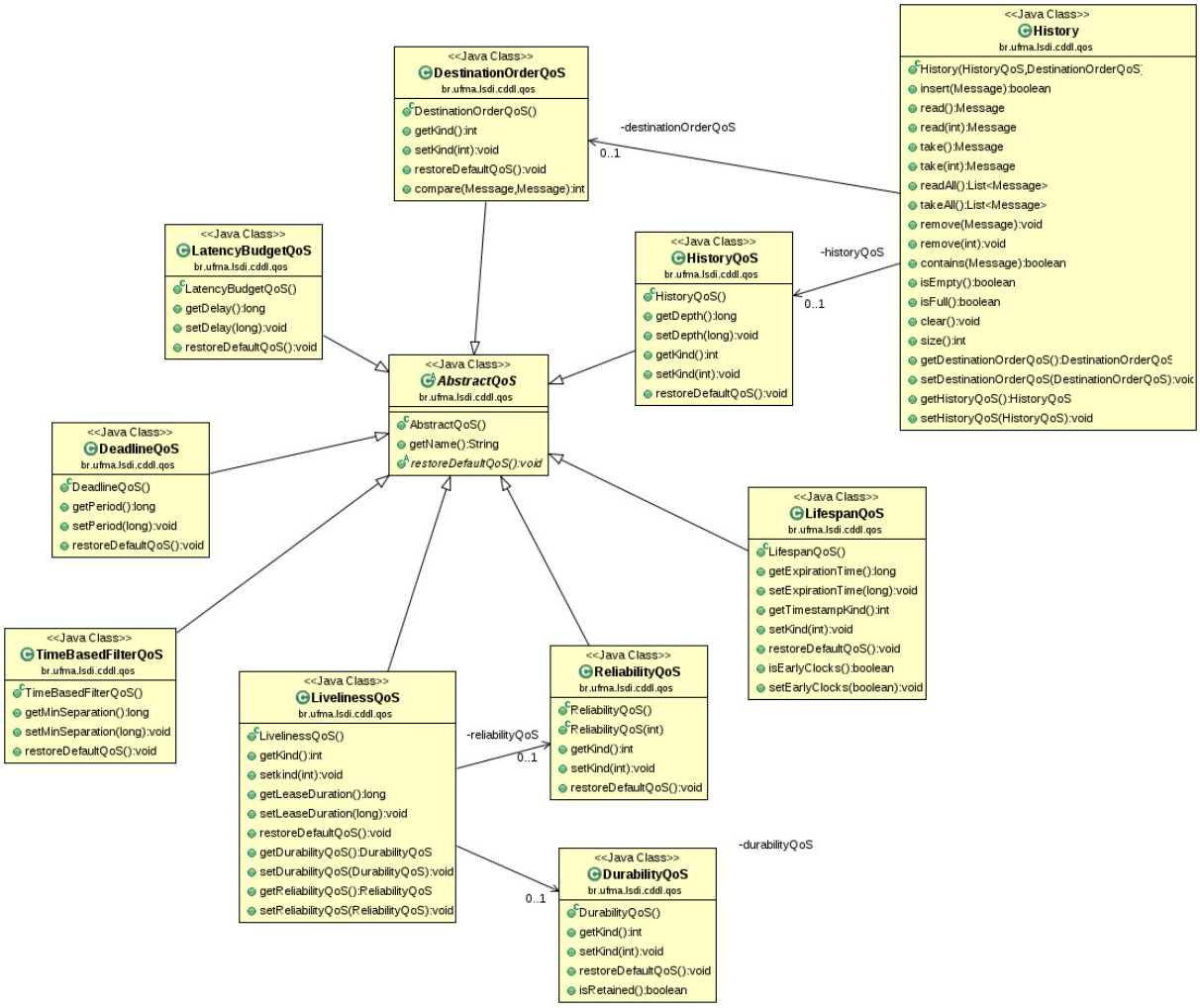

Figura 5.4: Diagrama de Classes das Políticas de QoS

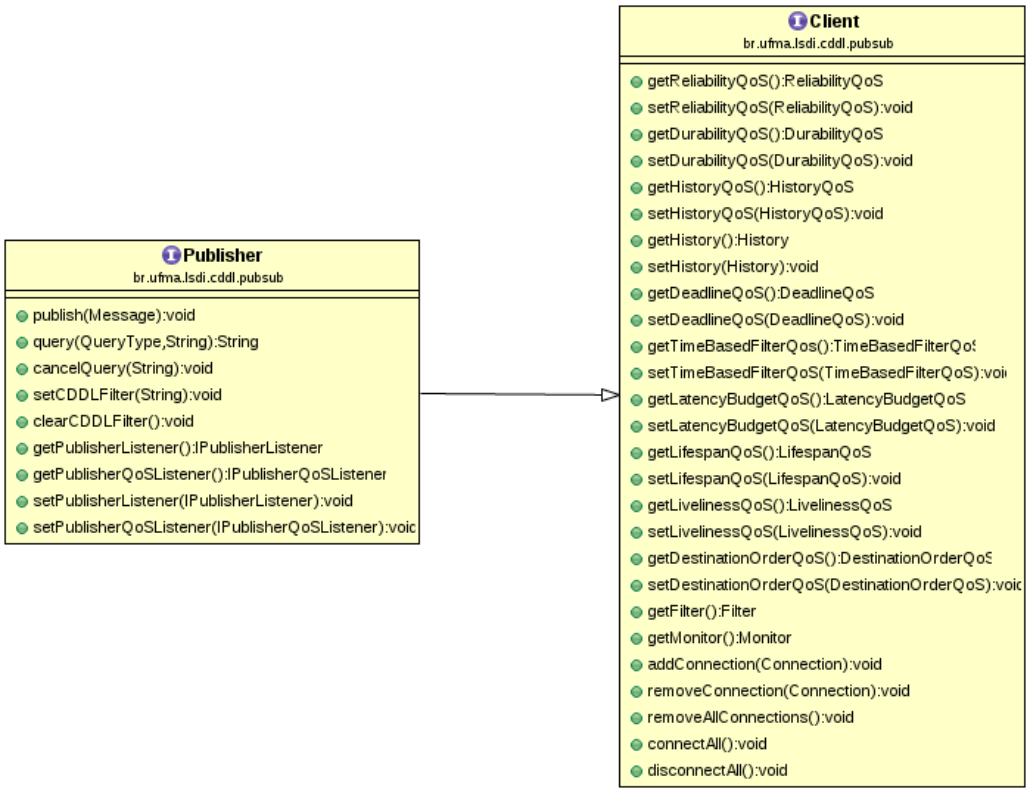

Figura 5.5: Interface Publisher

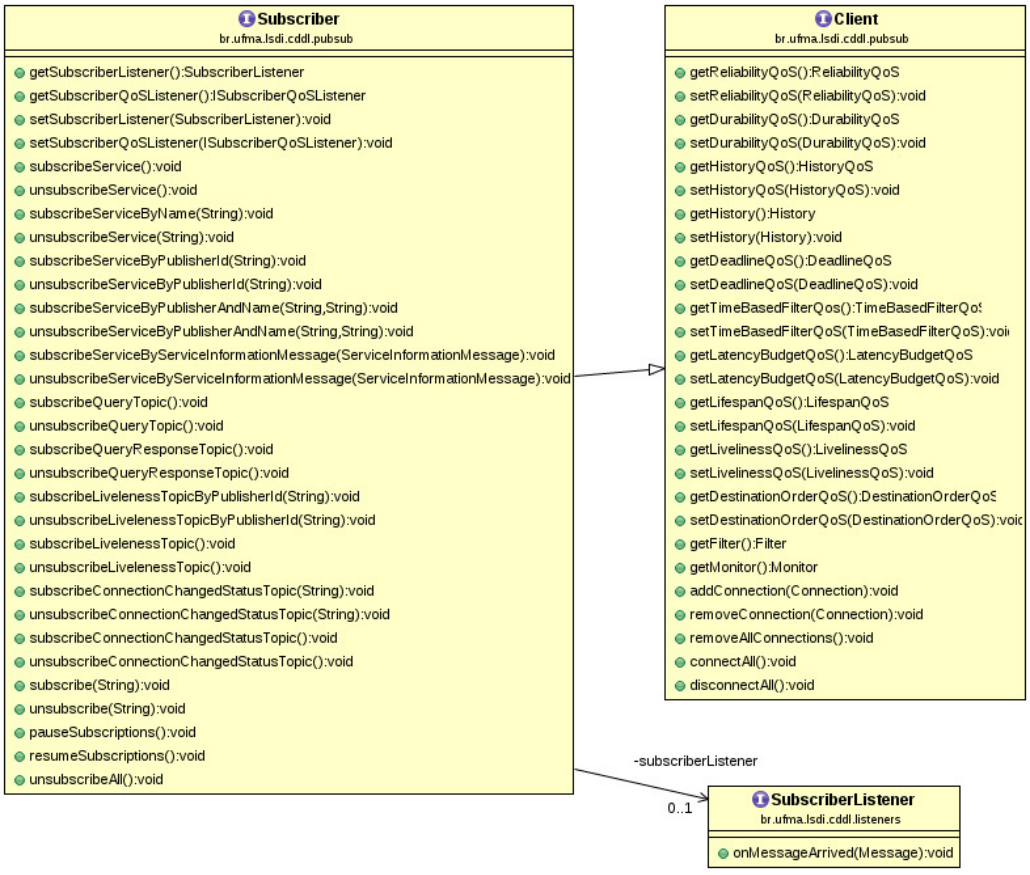

Figura 5.6: Interface Subscriber

As Listagens 5.9 e 5.10 mostram exemplos de como configurar a política Histórico no publicador e no subscritor, respectivamente. Nesse exemplo, o histórico do subscritor é configurado de modo a armazenar todos os dados, enquanto que o histórico do publicador é configurado de modo a armazenar apenas as últimas cem mensagens. A política é configurada utilizando-se um objeto da classe `HistoryQoS`. O tipo de histórico (*Keep All* ou *Keep Last*) é informado utilizando-se o método `setKind(int kind)` dessa classe. A definição do histórico como *Keep All* no subscritor indica que ele guardará todas as mensagens (até o limite dos recursos). A definição de profundidade do histórico do subscritor requer o uso do método `setDepth(int depth)`. A política é alterada no publicador e no subscritor invocando-se o método `setHistoryQoS(HistoryQoS qos)`.

Listing 5.9: Exemplo de configuração da política Histórico no publicador.

```
//...Oculta codigos anteriores
//Cria uma instancia da politica de QoS para o publicador
HistoryQoS pubHistoryQoS = new HistoryQoS();
//Configura o historico como keep last
pubHistoryQoS.setKind(HistoryQoS.LAST);
//configura a profundidade do historico
pubHistoryQoS.setDepth(100);
//Obtem instancia do publicador
Publisher publisher = DefaultPablisher.getInstance();
//muda a politica do publicador
publisher.setHistoryQoS(historyQoS);
//...
```

Listing 5.10: Exemplo de configuração da política Histórico no subscritor.

```
//instancia a politica de QoS do subscritor
HistoryQoS subHistoryQoS = new HistoryQoS();
//configura o historico para armazenar tudo
subHistoryQoS.setKind(HistoryQoS.KEEP_ALL)
//Obtem instancia do subscritor
Subscriber subscriber = DefaultSuscriber.getInstance();
//muda a politica no subscritor
subscriber.setHistoryQoS(historyQoS);
```

As Listagem 5.11 e 5.12 mostram exemplos de como configurar a política Taxa de Atualização (Filtro baseado em Tempo) no publicador e no subscritor, respectivamente. Essa política é representada pela classe `TimeBasedFilterQoS`. O intervalo mínimo de separação entre as amostras é informado pela aplicação utilizando-se o método `setMinSeparation(long time)` dessa classe. Uma vez instanciada e configurada, a política é alterada tanto no publicador como no subscritor invocando-se o método `setTimeBasedFilterQoS(TimeBasedFilterQoS qos)`.

Listing 5.11: Exemplo de configuração da política Taxa de Atualiação no publicador.

```
//...Oculta codigos anteriores
//Obte uma instancia da politica de QoS
TimeBasedFilterQoS timeBasedFilterQoS = new TimeBasedFilterQoS();
//configura o intervalo minimo de separacao (milisegundos)
timeBasedFilterQoS.setMinSeparation(1000);
Publisher publisher = DefaultPublisher.getInstance();
//muda a politica no subscritor
publisher.setTimeBasedFilterQoS(timeBasedFilterQoS);
```

Listing 5.12: Exemplo de configuração da política Taxa de Atualiação no no subscritor.

```
//...Oculta codigos anteriores
//Obte uma instancia da politica de QoS
TimeBasedFilterQoS timeBasedFilterQoS = new TimeBasedFilterQoS();
//configura o intervalo minimo de separacao (milisegundos)
timeBasedFilterQoS.setMinSeparation(5000);
Subscriber subscriber = DefaultSuscriber.getInstance();
//muda a politica no subscritor
subscriber.setTimeBasedFilterQoS(timeBasedFilterQoS);
```

Considerando a dinamicidade das aplicações cientes de contexto, cujos requisitos de QoC podem mudar de um momento para o outro, a M-Hub/CDDL permite alterar o valor de todos parâmetros de QoS em tempo de execução, exceto a Sessão. Por exemplo, em relação à política Confiabilidade, se o publicador estava enviando mensagens com confiabilidade em nível 0 (*at most once*) e muda dinamicamente essa confiabilidade para o nivel 1 (*at least once*), isso significa que os envios das novas mensagens já serão feitos utilizando-se a configuração mais recente.

5.6 Confiabilidade de Entrega em Redes Instáveis

Como explicado na Seção 4.3.2, o mecanismo de confiabilidade padrão do MQTT não é tão eficiente em cenários com desconexões de longa duração, pois o tamanho do *buffer* MQTT, bem como o número de tentativas de retransmissão de mensagens em caso de falhas na entrega, são limitados. Diante disso, a fim de evitar perdas no envio de mensagens em ambientes móveis, a M-Hub/CDDL adota uma abordagem que combina a confiabilidade padrão, nativamente provida pelo protocolo MQTT, com um mecanismo de *bufferização* intermediária, implementado pela CDDL.

Esse mecanismo funciona da seguinte forma: durante a transmissão, todas as mensagens do tipo *at least once* e *exactly once* são armazenadas em um *buffer* da CDDL, permanecendo lá enquanto a entrega das mesmas não for confirmada. Em caso de falha na conexão, a CDDL pausa o envio de mensagens, entrando em *roam mode*. Ao reestabelecer a conexão, CDDL enviará primeiro aquelas mensagens cujas entregas não foram confirmadas nas tentativas anteriores. Em seguida, a CDDL enviará as demais mensagens da fila. O mecanismo de *bufferização* da CDDL é gerenciado pelo componente *Connection* e sua utilização é opcional. Portanto, o middleware utilizará a confiabilidade padrão MQTT caso o referido mecanismo não seja habilitado. Ressalta-se que a ideia de uso de um *buffer* intermediário em comunicações usando MQTT propriamente dita não é nova, pois já foi originalmente apresentada na literatura por Luzuriag et al. [69]. Entretanto, no melhor do nosso conhecimento, esta é a primeira vez que esse tipo de abordagem é explorada de fato por um middleware de contexto.

A Listagem 5.13 mostra um exemplo de como a aplicação produtora usa a CDDL para configurar a confiabilidade e o mecanismo de *buffer* intermediário. Nesse exemplo, a aplicação produtora ativa o *buffer* intermediário usando o método `setEnableIntermediateBuffer(boolean flag)` da classe `Connection`. Esse método recebe um parâmetro lógico que se verdadeiro habilitará o mecanismo. Se o parâmetro informado tem valor falso, então o mecanismo é desabilitando. A configuração da confiabilidade, assim como a maioria das políticas de QoS, requer que um objeto da política correspondente seja instanciado. Como mencionado, o *buffer* intermediário só se aplica quando o nível de confiabilidade do cliente é 1 (*at least once*) ou 2 (*exactly once*). No exemplo, a aplicação usa o método `setKind(int kind)` da classe `ReliabilityQoS` para configurar a confiabilidade no nível 1.

Listing 5.13: Exemplo de configuração de Confiabilidade usando buffer intermediario.

```
//...Oculta codigos anteriores

//informa o ID do cliente que sera usado por todas as conexoes
MHubCDDL.getInstance().setClientId("bertodetacio@gmail.com");

//Obtem a instancia da fabrica de conexoes
ConnectionFactory connectionFactory = ConnectionFactory.getInstance();
//obtem uma instancia da conexao
Connection connection = connectionFactory.createConnection();
//Informa a URL do Broker remoto
connection.setHost("lsdi.ufma.br");
//ativa o buffer intermediario
connection.setEnableIntermediateBuffer(true);

//instancia a politica de Confiabilidade
ReliabilityQoS reliabilityQoS = new ReliabilityQoS();
//configura a confiabilidade no nivel 1
reliabilityQoS.setKind(ReliabilityQoS.AT_MOST_ONCE);
//Obtem uma instancia do publicador default

Publisher publisher = DefaultPublisher.getInstance();
//altera a politca de confiabilidade do publicador
publisher.setReliabilityQoS(reliabilityQoS);
//adiciona uma conexao para o publicador
publisher.addConnection(connection);

//Conecta
connection.connect();
```

5.7 Registro e Descoberta de Serviços baseada em Qualidade de Contexto

Para permitir que aplicações de IoT locais e remotas tenham conhecimento dos provedores de serviços de contexto disponíveis, a M-Hub/CDDL define e implementa mecanismos de registro e descoberta de serviços. A etapa de registro

é aquela na qual os diretório local e globais têm suas listas de serviços de contexto atualizadas. A etapa de descoberta é aquela na qual as aplicações têm acesso às informações disponibilizadas pelos diretórios. Essas etapas são explicadas a seguir.

5.7.1 Registro de Serviços baseado em Qualidade de Contexto

Conforme descrito na Seção5.3.2, existem dois tipos de diretório: local e global. Cada diretório local é um Agente de Processamento de Eventos (*Event Processing Agent* - EPA - ver Seção 4.1) que, dentre outras funções, executa consultas EPL continuamente sobre os fluxos de eventos vindos dos publicadores (também locais), a fim de detectar se um novo serviço está disponível ou se informações sobre um serviço já existente mudaram. O diretório local armazena dados como o nome do serviço e atributos de contexto e de QoC. Se houver uma conexão com a nuvem, o diretório local enviará atualizações dos serviços para um ou mais diretórios globais, que também são EPAs. O processo de registro dos serviços nos diretórios é ilustrado no digrama de sequência da Figura 5.7

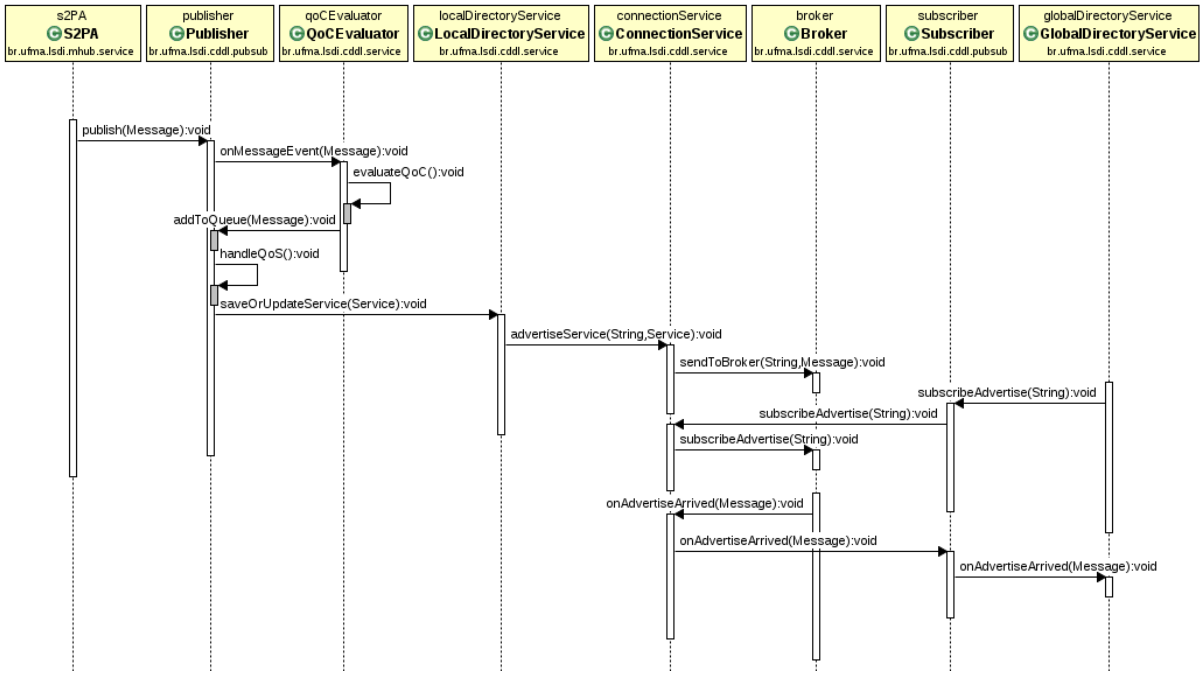

Figura 5.7: Diagrama de Sequência do Processo de Registro de Serviços

O processo de registro inicia-se quando o S2PA (ou a própria aplicação produtora) solicita a publicação da mensagem (contendo o dado de contexto) ao componente `Publisher`. Em seguida, o `Publisher` envia a mensagem para o `QoC`

`Evaluator`. Após a avaliação de QoC, a mensagem retorna ao `Publisher` com os atributos de QoC já computados. Após ser adicionada na fila do `Publisher` de fato, acontece o tratamento da qualidade de serviço, em conformidade com as políticas e parâmetros por este definidos. Em seguida, o `Publisher` envia uma mensagem contendo a descrição do serviço ao `Local Directory Service`, que registra ou atualiza o serviço, se necessário. Caso haja alteração nos registros, o `Local Directory Service` solicitará diretamente ao `Connection Service` que publique um anúncio de descoberta/atualização de serviço num determinado tópico. O processo pressupõe ainda que o `Global Directory Service` assine o mesmo tópico no qual o diretório local publica. Se isso tiver acontecido, o broker encaminhará o anúncio ao componente `Connection` usado pelo diretório global, que finalmente repassará a mensagem ao `Local Directory Service`. Uma vez recebido o anúncio, o diretório global atualizará seus registros.

As aplicações consumidoras só poderão ter acesso às informações sobre os serviços disponíveis registrados nos diretórios de duas maneiras: via anúncios (descoberta passiva) ou via consultas (descoberta ativa). Essas abordagens são explicadas nas subseções a seguir.

5.7.2 Descoberta baseada em Anúncios

A descoberta baseada em anúncios é bastante simples. Periodicamente, os diretórios locais e globais publicam anúncios contendo informações sobre os serviços disponíveis em um tópico previamente conhecido por todas as aplicações consumidoras. Os clientes que tiverem assinado esses tópicos receberão os anúncios, que contêm as características do serviço, tais como atributos de contexto de QoC, bem como o tópico no qual esse serviço está sendo publicado. Com essas informações, a aplicação consumidora poderá assinar o serviço anunciado.

Se a aplicação estiver conectada apenas ao Micro Broker Local, então ela receberá atualizações do seu diretório local. Se a aplicação estiver conectado ao Server Broker, então ela receberá atualizações do diretório global. Existe ainda a possibilidade de um cliente receber atualizações publicadas pelos diretórios locais de outros clientes, sem intermédio do diretório global. Essa é uma configuração opcional que requer uma pré-definição dos tópicos compartilhados entre os clientes. Assim como acontece com

a publicação/subscrição dos serviços propriamente ditos, uma aplicação consumidora pode se subscrever para receber anúncios de um ou mais produtores. A aplicação pode assinar ou cancelar o recebimento de anúncios a qualquer tempo.

O processo de descoberta baseada em anúncios é demonstrado no diagrama de sequência ilustrado na Figura 5.8

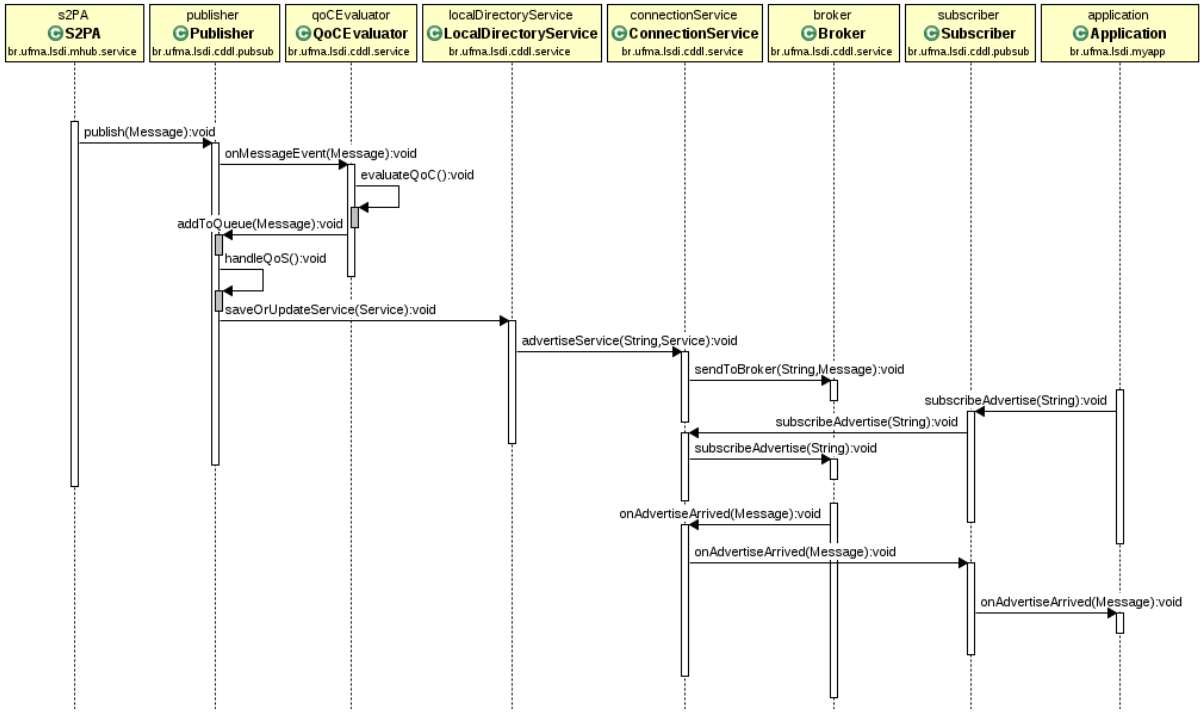

Figura 5.8: Diagrama de Sequência do Processo de Descoberta baseada em Anúncios

Pode-se observar nesse diagrama que a única diferença em relação ao processo demonstrado na Figura 5.7 é que o consumidor dos anúncios de serviço agora é a própria aplicação, ao invés do diretório global. Na prática, a maneira como a aplicação se subscreve para receber anúncios dos diretórios locais e globais é praticamente a mesma utilizada pelo `Global Directory Service` para receber anúncios publicados pelo `Local Directory Service`.

5.7.3 Descoberta baseada em Consultas

Nesta abordagem, a aplicação consumidora só poderá descobrir os serviços disponíveis se definir e publicar uma consulta de serviços. Uma vez publicada, a consulta será recebida e resolvida por um diretório (local ou global) que publicará, em um tópico conhecido apenas pelo solicitante, a resposta contendo uma lista de serviços

compatíveis com os critérios especificados na consulta. Os critérios das consultas podem levar em consideração todos os atributos da informação de contexto, inclusive os atributos de QoC. A linguagem utilizada para a especificação da consulta é a EPL Esper, o que garante uma grande expressividade de requisitos de contexto e de QoC.

A definição de qual diretório resolverá a consulta, se local ou global, dependerá do tipo de conexão do publicador. Se o publicador estiver conectado apenas ao Micro Broker Local, isso significa que apenas o diretório local responderá a consulta. Se o publicador estiver conectado ao Server Broker, a priori, apenas o diretório global pode responder a consulta. Entretanto, existe ainda a possibilidade do cliente publicar consultas visíveis por diretórios locais de outros clientes, dispensando o uso de um diretório global. Entretanto, isso requer uma configuração opcional que define um tópico de publicação de consultas comum para todos os clientes do domínio CDDL.

As Figuras 5.9 e 5.10 mostram os diagramas de sequências referentes ao processo de consulta. Pode-ser observar que a única diferença está no tipo de diretório que resolve e responde a consulta, sendo que na primeira é o `Local Directory Service`, pois é uma consulta local, enquanto que na segunda é o `Global Directory Service`, pois se trata de uma consulta global.

O processo inicia-se com o diretório de serviços enviando uma solicitação de subscrição no tópico em que as consultas serão publicadas. Em seguida, a aplicação consumidora se registra no tópico onde as respostas das consultas serão publicadas, de modo que esteja pronta para receber a reposta antes mesmo de efetuar a consulta. Concluída a subscrição, a aplicação define e publica uma consulta. Na sequência, o diretório recebe, resolve e publica a resposta da consulta. Por fim, a aplicação consumidora recebe a resposta da consulta. Observa-se que toda a comunicação entre a aplicação e o broker passa antes obrigatoriamente pelos componentes de subscrição e conexão. O mesmo vale para toda a comunicação entre os diretórios e o broker.

A fim de dar suporte a diferentes cenários de IoT, a M-Hub/CDDL define e implementa dois tipos de consulta: instantânea e contínua. A consulta instantânea retorna, uma única vez, uma lista contendo os publicadores e serviços disponíveis que satisfazem os critérios definidos pela aplicação, no momento em que a consulta é realizada. Após a resposta ser entregue, a execução da consulta é encerrada. A consulta contínua, por sua vez, além de retornar a lista de publicadores e serviços disponíveis

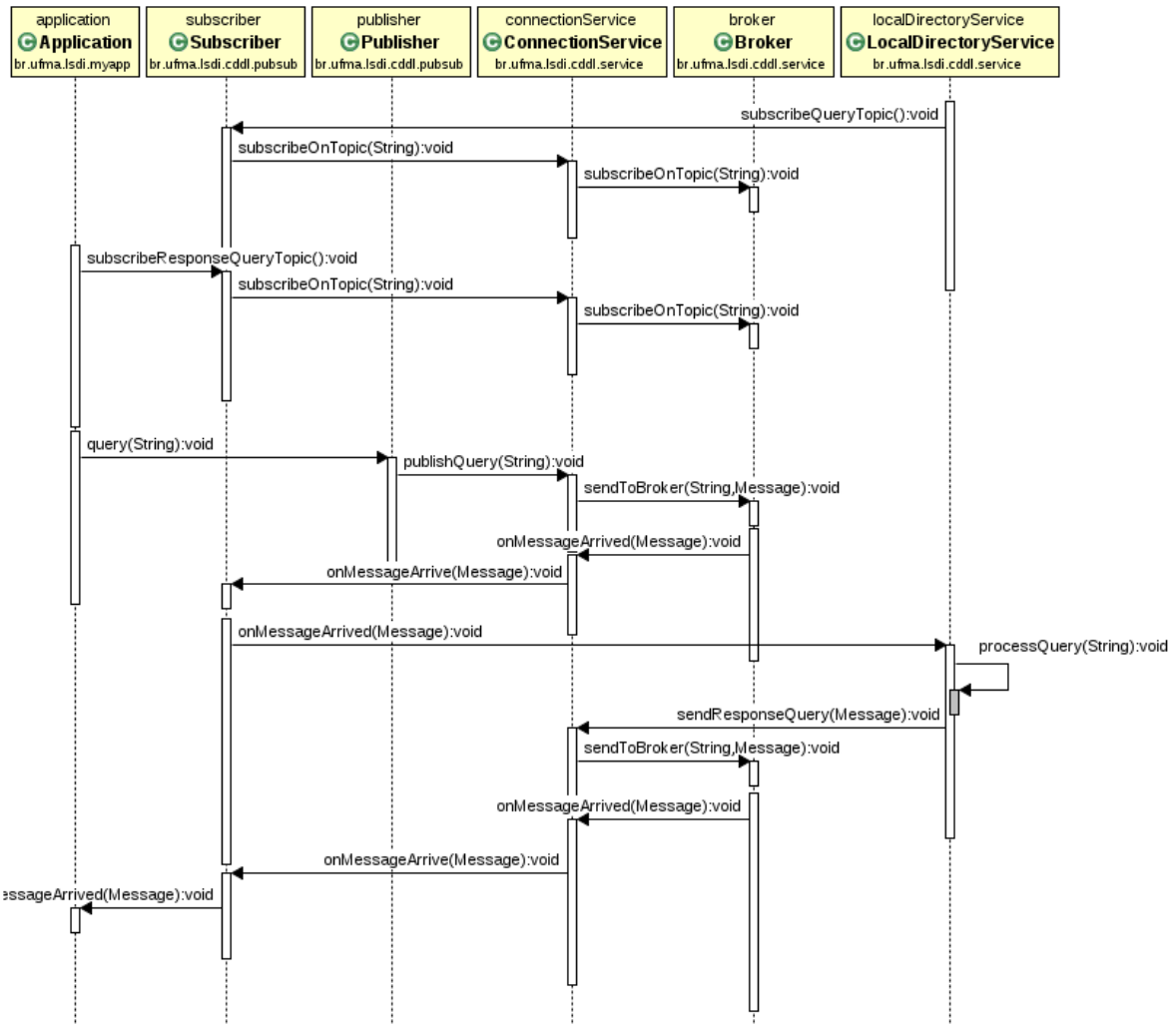

Figura 5.9: Diagrama de Sequência do Processo de Descoberta de Serviços (Local)

no momento da primeira execução da consulta, permanece em execução por tempo indeterminado, fazendo com que a aplicação requisitante seja notificada em caso de surgimento de novos provedores de serviços que atendam os requisitos da consulta. A aplicação consumidora pode cancelar a execução da consulta contínua à qualquer momento.

Naturalmente, em cenários de grande dinamismo e mobilidade (com conexões e desconexões de dispositivos acontecendo a todo momento), como IoMT, a consulta contínua tende a ser mais adequada, pois permite que a aplicação consumidora seja periodicamente notificada do surgimento de novos objetos inteligentes. Levando-se em consideração a QoC, a descoberta contínua agrega a vantagem de permitir que a aplicação consumidora tenha conhecimento do surgimento de serviços com QoC superior aos já assinados/utilizados. Desse modo, a aplicação poderá substituir o provedor de serviço em uso por aquele que apresenta

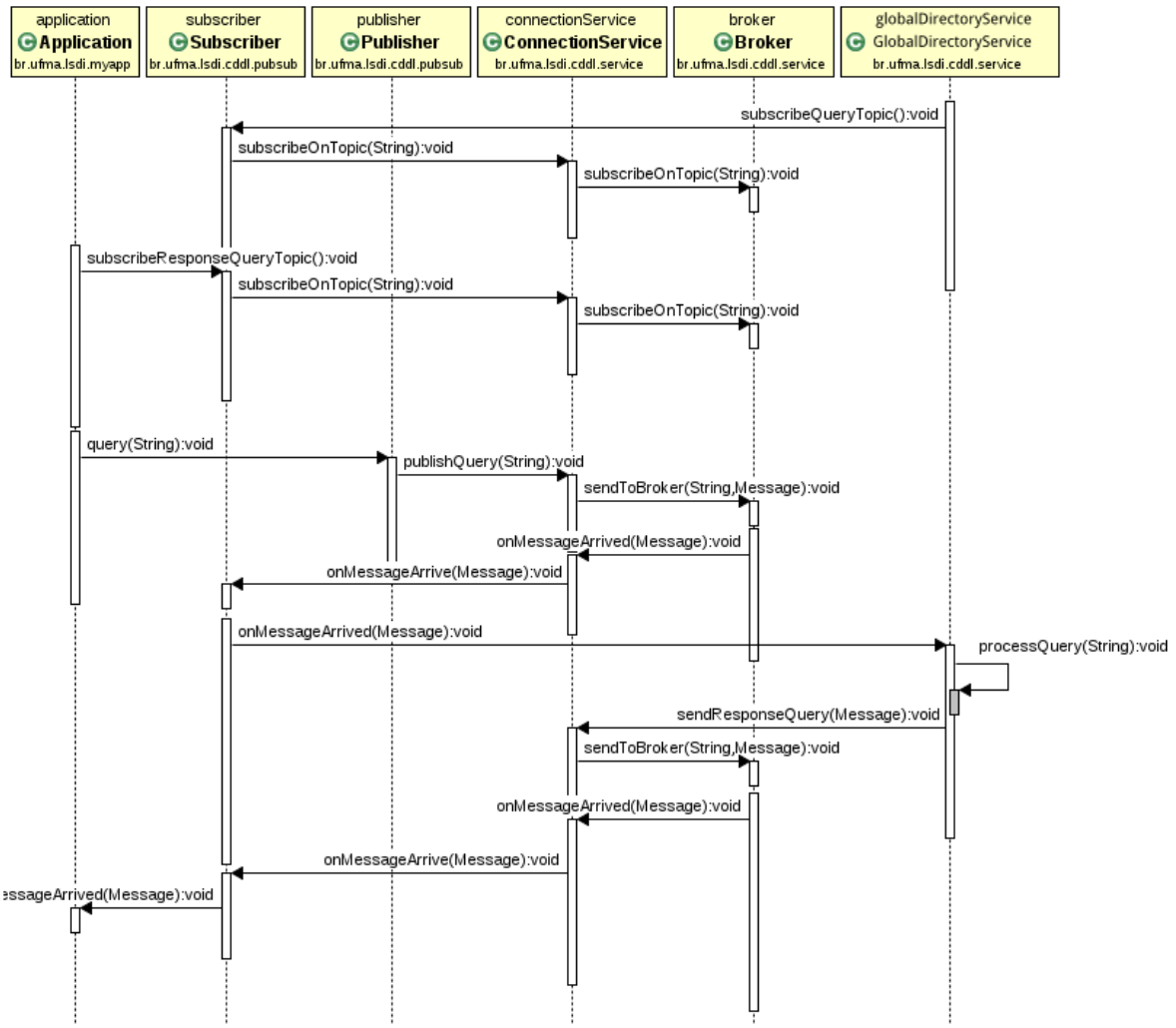

Figura 5.10: Diagrama de Sequência do Processo de Descoberta de Serviços (Global)

maior QoC, o que poderá impactar positivamente no funcionamento da aplicação e na qualidade de experiência dos usuários.

A Listagem 5.14 mostra um exemplo de como a aplicação publica consultas usando o CDDL. Neste exemplo, a aplicação invoca o método `query (QueryType queryType, String epl)` do componente *Publisher*. Este método recebe como um seus parâmetros a consulta EPL a ser publicada. A consulta especificada procura todos os serviços de localização cuja precisão seja inferior a 5 m. O exemplo faz referência a mensagens do tipo `ServiceInformationMessage` na cláusula EPL uma vez que esta é a estrutura de dados que contém a descrição dos serviços. Neste exemplo, a consulta é contínua, portanto, o consumidor receberia todos os serviços disponíveis compatíveis com a consulta conforme forem descobertos.

Listing 5.14: Exemplo de consulta contínua

```
//obtem referencia para o publicador default e subscritor default
publisher = DefaultPublisher.getInstance();
subscriber = DefaultSubscriber.getDefaultSubscriber();
// especifica os criterios da consulta
String query = ``serviceName = 'LOCATION' and accuracy < 5'';
// codigo de retorno pode ser usado para cancelar a consulta
int returnCode = publisher.query(QueryType.CONTINUOUS, query);
// cria um listener para receber as notificacoes
subscriber.setSubscriberListener(new ISubscriberListener() {
  @Override
  public void onMessageArrived(Message message) {
    // trata as respostas da consulta
    if (message instanceof QueryResponseMessage) {
      QueryResponseMessage response = (QueryResponseMessage) message;
        // is the query response about location of johndoe@example.com?
        if (message.returnCode == returnCode) {
          // obtem a lista de servicos contidos na resposta da consulta
          List<ServiceInformationMessage> services =
            response.getServiceInformationMessageList();
          // subscreve-se em todos os servicos contidos na resposta
          for(ServiceInformationMessage info : services) {
            subscriber.subscribeServiceTopic(info.getTopic())
          }
        }
      return;
    }
    // trata mensagens contendo os dados de sensores
    if (message instanceof SensorDataMessage) {
      SensorDataMessage sensorDataMessage = (SensorDataMessage) message;
      // simula a exibicao da mensagem na tela
      showMessageOnUI(message);
      return;
    }
  }
  // caso a aplicacao precise cancelar a consulta
  // publisher.cancelQuery(returnCode)
});
```

Uma limitação do mecanismo de descoberta de serviços é a sua incapacidade de descobrir serviços levando em consideração aspectos relacionados à semântica, existindo a possibilidade de ambiguidades na descoberta de serviços.

5.8 Detecção de Variações de Contexto baseada em Qualidade de Contexto

Para dar suporte à detecção de variações de contexto e de QoC, a M-Hub/CDDL provê um mecanismo de monitoramento que se baseia no processamento de eventos complexos. Conforme descrito na Seção 5.3.2, o componente responsável pelo monitoramento é o `Monitor`. Esse componente permite a especificação de regras EPL que ativam notificações de eventos a partir de critérios estabelecidos pela aplicação. Para isso, o `Monitor` contém seu próprio EPA. As notificações de detecção dos eventos de monitoramento são recebidas pela aplicação por meio de um *listener* específico: o `MonitorListener`

A Figura 5.11 mostra um diagrama de classes que expressa a relação entre os clientes (publicadores e subscritores) e os componentes utilizados no monitoramento. Nela é possível observar que, além de outros métodos, existe um método `Monitor getMonitor()`. Esse método é utilizado pelo cliente para obter a instância do monitor. Cada publicador e subscritor possui seus respectivos monitores. A interface `Monitor` define apenas três métodos:

- `String addRule(String epl, MonitorListener listener)`: serve para adicionar uma string contendo uma regra de monitoramento. O segundo parâmetro deste método é um *listener*, que implementa a interface `MonitorListener`, utilizado pelo monitor para enviar as notificações quando as regras de monitoramento são satisfeitas. O *listener* deve implementar o método `void onEvent(Message message)`, que será chamado pelo monitor para repassar a mensagem que satisfez a regra de monitoramento.
- `void removeRule(String ruleId)`: remove uma regra de monitoramento. Desse modo, a aplicação deixa de receber notificações para a regra em questão.

- `int getNumRules()`: informa quantas regras de monitoramento estão ativas em determinado momento.

A Figura 5.12 apresenta um diagrama de sequência onde pode-se observar uma aplicação cliente incluindo o monitoramento no processo de subscrição de uma informação de contexto. Primeiramente, a aplicação obtém uma referência para o monitor utilizando o método `Monitor getMonitor()`. Em seguida, especifica a regra de monitoramento através do método `String addRule(String epl)` passando uma regra na linguagem EPL e o *listener*, que receberá as notificações quando a regra for satisfeita. Neste momento, o componente `Monitor` cria uma regra no seu EPA para processar as novas mensagens. Após especificar a regra de monitoramento, a aplicação pode subscrever um serviço (no diagrama, utilizando o método `void subscribeServiceByName(String serviceName)`. Como existe uma regra de monitoramento ativa, o componente `Subscriber` solicita ao *Monitor* que este também se subscreva no tópico de interesse da aplicação. À medida que as mensagens chegam ao *Broker*, ele as envia para o *Monitor*, que utiliza o EPA para processá-las. Durante o processamento, as mensagens que atenderem à regra são enviadas para o `MonitorListener` definido pela aplicação. O monitoramento permanece ativo enquanto a regra não for removida do componente `Monitor`.

5.9 Conclusão do Capítulo

Este capítulo apresentou uma solução de middleware voltada para o desenvolvimento de aplicações de IoT com requisitos de gerenciamento e qualidade de contexto: a M-Hub/CDDL. Essa abordagem combina duas camadas: a M-Hub e a CDDL. A camada M-Hub foi reutilizada do middleware ContextNet. Ela permite que a solução colete dados de contexto a partir da interação com objetos inteligentes. No âmbito desta tese, algumas extensões foram providas ao M-Hub de modo a atender os objetivos da solução. A CDDL é uma camada de distribuição de dados de contexto que atende a diversos requisitos como distribuição local e remota transparente, descoberta de serviços, filtragem e monitoramento de dados com suporte à qualidade da informação e do serviço de distribuição. A CDDL considera a existência de brokers que intermedeiam a comunicação entre produtores e consumidores de contexto.

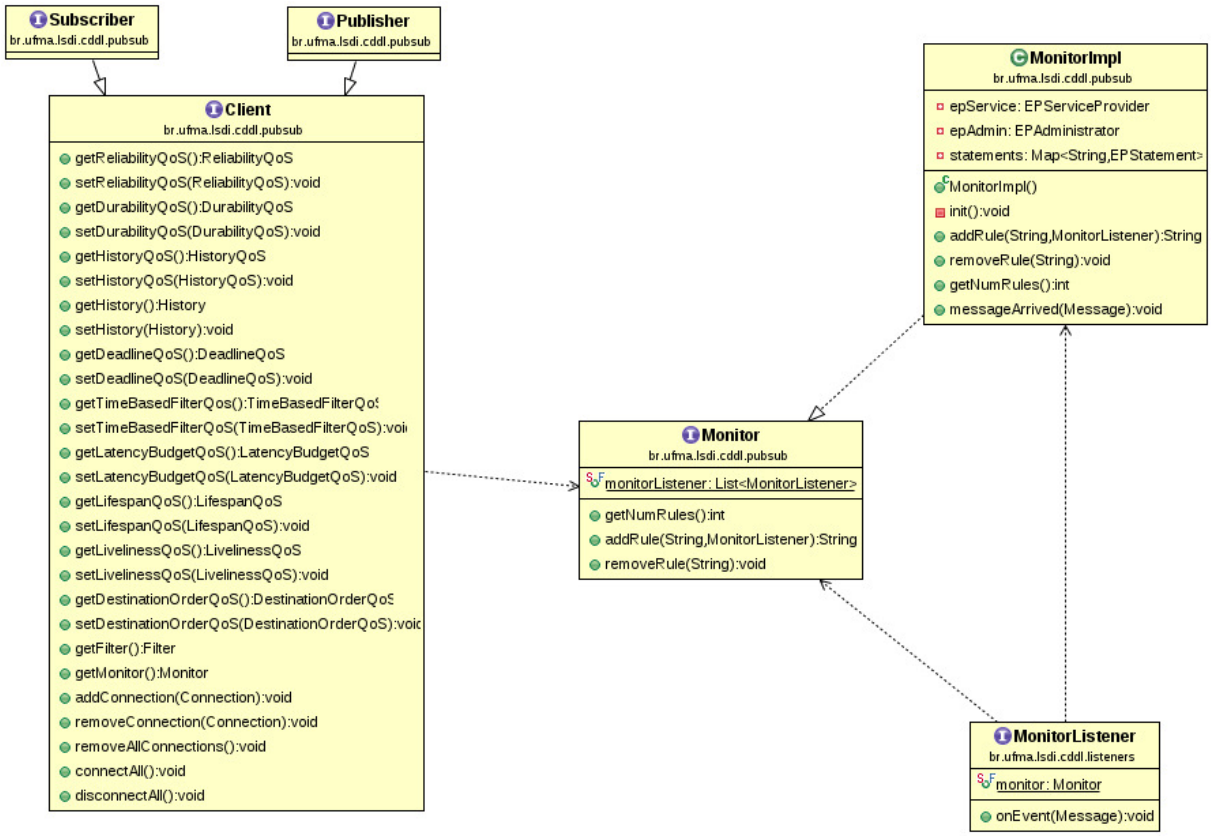

Figura 5.11: Diagrama de Classes relacionadas ao Monitoramento

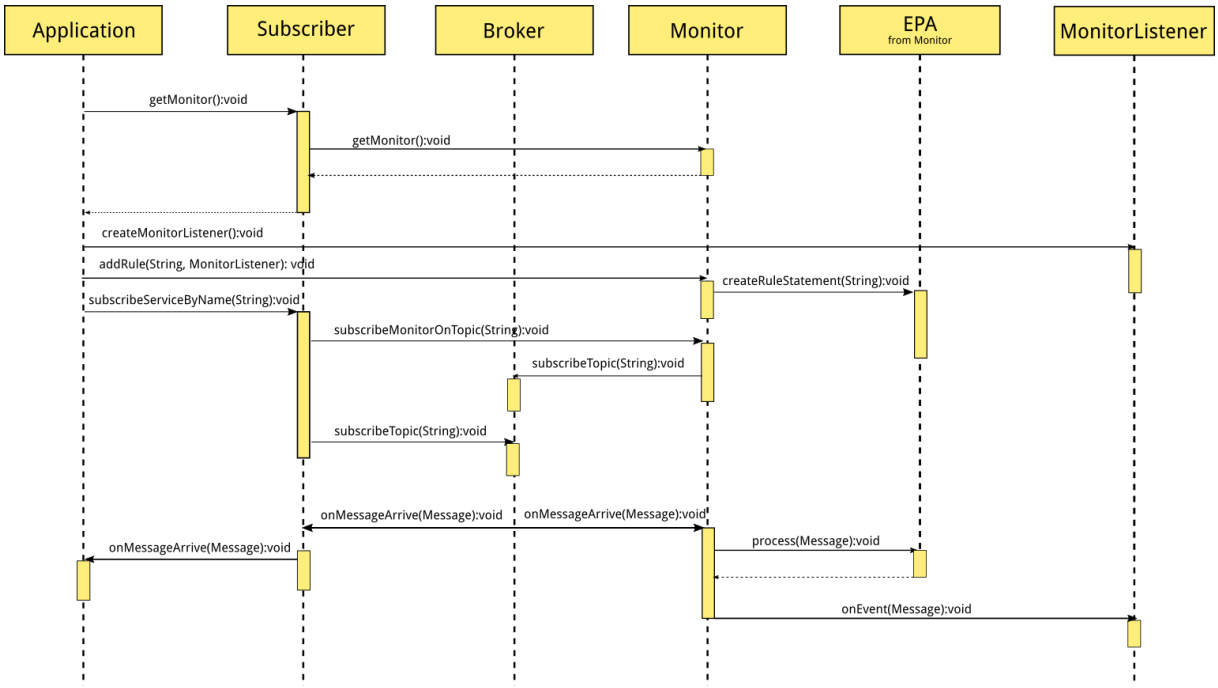

Figura 5.12: Diagrama de sequência do processo de monitoramento

Embora a M-Hub seja uma importante camada da solução, a principal contribuição do aluno de doutorado está na concepção, desenvolvimento e avaliação da CDDL, na qual concentra-se a maioria dos mecanismos que foram explicados, além de ser a camada

com a qual as aplicações de IoT interagem diretamente. A CDDL possui diversos mecanismos, mas este capítulo focou em mostrar apenas os mecanismos que abordam os desafios apresentados no Capítulo 1: gerenciamento de sensores, provisionamento de qualidade de contexto, confiabilidade de entrega em redes instáveis, descoberta de serviços e monitoramento baseado em QoC.

O gerenciamento de sensores da M-Hub/CDDL visa abstrair a heterogeneidade do hardware dos dispositivos e das tecnologias de comunicação usadas na aquisição dos dados de contexto. Para alcançar esse objetivo a solução desenvolvida reutiliza a M-Hub e ainda a estende para permitir um controle maior acerca de quais sensores e tecnologias de comunicação podem ser iniciados. Embora o modo de interação oportunista padrão da versão original seja adequado para cenários de IoMT, existem situações em que a aplicação precisa ter controle sobre a inicialização dos sensores de modo a poupar recursos do dispositivo móvel.

Os mecanismos de provisionamento de QoI e QoS da solução desenvolvida visam atender os diversos requisitos de qualidade de contexto de aplicações de IoT emergentes. Por esse motivo a abordagem já define e implementa diversos parâmetros e políticas de qualidade a fim de transferir menos responsabilidades para o programador da aplicação que, ao dispor de tais mecanismos, poderá se concentrar na implementação de requisitos específicos da aplicação. Em relação à QoI, foi apresentado que a aplicação pode fornecer o valor de diversos parâmetros, sendo que alguns deles são computados/avaliados dinamicamente pela solução, com o intuito de prover qualidade para a aplicação ainda que o produtor não informe todos os atributos de qualidade. O mecanismo de avaliação de QoC é transparente para a aplicação, tendo sido desenvolvido com base no processamento de eventos complexos. Em relação à QoS, significativamente mais complexa de ser provida que a QoI, a M-Hub/CDDL define um conjunto padrão de políticas que permitem que a aplicação tenha o controle sobre diversos requisitos de qualidade de serviço, bem como possibilita alterações em seus parâmetros em tempo de execução, a fim de permitir que aplicações adaptem sua qualidade de serviço a diferentes situações. Alguns mecanismos como confiabilidade e sessão são providos nativamente pelo protocolo MQTT, enquanto outros dependem apenas dos componentes da CDDL.

O mecanismo de confiabilidade em redes instáveis do M-Hub/CDDL visa diminuir a perda de mensagens em cenários nos quais a aplicação utiliza redes

com conectividade intermitente. O mecanismo adotado combina a solução de confiabilidade nativa do MQTT com uma *buffer* intermediário, que permite que os dados sejam retransmitidos e entregues, mesmo após desconexões de longa duração, superando limitações do uso simples do MQTT. Como explicado, a ideia do *buffer* intermediário não é nova, mas a sua integração no âmbito de um middleware de contexto, no melhor do nosso conhecimento, não é algo que tenha sido explorado.

Os mecanismos de registro e descoberta de serviços visam lidar com situações onde eventualmente as aplicações não conhecem os serviços disponíveis ou não sabem de algumas características que podem ser cruciais no momento de escolher um provedor, tais como qualidade de contexto, por exemplo. A abordagem adotada pela solução é muito flexível pois permite que as aplicações conheçam os serviços tanto com base em anúncios como a partir da especificação de consultas. O próprio mecanismo de consulta também é flexível, de modo que as consultas podem ser instantâneas ou contínuas. Enquanto a primeira se restringe a descobrir serviços disponíveis no momento da consulta, a segunda permite que a aplicação seja notificada tão logo novos serviços de seu interesse sejam descobertos. Esse mecanismo foi pensado para cenários de IoMT, nos quais o gateway pode descobrir e interagir com diversos objetos inteligentes de forma oportunista à medida em que se movimenta. O mecanismo de descoberta também é baseado em CEP.

O mecanismo de monitoramento visa permitir a detecção de variações de contexto e de QoC em tempo de execução. Esse mecanismo é importante porque permite que as aplicações de IoT se adaptem a essas variações de forma a minimizar os prejuízos sobre o funcionamento de seus serviços. M-Hub/CDDL oferece um mecanismo de monitoramento flexível, que permite monitorar fluxos de dados no envio e no recebimento, bem como adicionar e remover regras de monitoramento em tempo de execução. A implementação do mecanismo de monitoramento também é baseada no processamento de eventos complexos.

Por fim, considera-se que a M-Hub/CDDL soluciona os problemas apresentados na Seção 1.2. A abordagem tem a vantagem de prover abstrações de programação que permitem que os desenvolvedores utilizem esses mecanismos de forma simples. Logo, a contribuição não está apenas nos mecanismos em si, mas na forma como a abordagem de middleware permite que tais mecanismos sejam configurados e reconfigurados. Uma das limitações da abordagem é a falta de suporte

a semântica, algo que limita aspectos do gerenciamento dos sensores e da descoberta de serviços, uma vez que esses mecanismos não tratam as ambiguidades. O suporte à semântica no âmbito do middleware é um trabalho futuro.

6 Avaliação Qualitativa

A avaliação qualitativa da abordagem de middleware proposta consistiu na realização de dois tipos de estudos de caso:

- **Prova de conceito**: nesta avaliação, o objetivo era apenas provar que os conceitos e mecanismos da abordagem de middleware proposta podem ser aplicados ao desenvolvimento de aplicações de IoT com requisitos de QoC do mundo real. Em outras palavras, a prova de conceito permitiu observar o comportamento da solução proposta por meio de sua aplicação em um domínio específico, bem como verificar se ela é realmente adequada ao problema para o qual foi projetada.
- **Avaliação com os usuários**: nesta avaliação, além de provar que os conceitos e mecanismos da abordagem de middleware se aplicam ao desenvolvimento de aplicações de IoT em um domínio diferente do utilizado na prova de conceito, o objetivo era também avaliar a abordagem de middleware proposta junto aos usuários a que se destina, que neste caso são os desenvolvedores de aplicações de IoT, considerando diversos aspectos qualitativos relevantes para o projeto de soluções de middleware, tais como: usabilidade, facilidade de configuração, adequação das abstrações e interfaces de programação, eficiência dos serviços implementados, amplitude do suporte a QoC e documentação. Além de permitir avaliar a eficiência da solução considerando a perspectiva dos usuários, por meio da aplicação de entrevistas e questionários, o estudo de caso serviu para detectar e corrigir *bugs* na implementação, bem como para colher sugestões de melhorias.

Cabe ressaltar que, de modo geral, os trabalhos relacionados a middleware de contexto com suporte a QoC encontrados na literatura (ver Capítulo 8) mostram apenas avaliações qualitativas que se restringem a provas de conceito, sem que seja apresentado qualquer tipo de avaliação junto aos usuários. Sendo assim, a própria avaliação qualitativa do M-Hub/CDDL pode ser considerada uma contribuição desta tese, visto que é mais completa que as avaliações aplicadas a outras abordagens de middleware conhecidas. A seguir cada tipo de avaliação realizada, com os seus respectivos resultados e discussões, é apresentado em uma subseção a parte.

6.1 1º Estudo de Caso: Prova de Conceito

A aplicação da prova de conceito consistiu no desenvolvimento de uma aplicação de IoMT no domínio da saúde, denominada *Mobile Human Activity Recognition System* (MHARS), versão 2.0. Trata-se de uma nova versão do sistema MHARS, desenvolvido por Ribeiro Filho et al. [41, 42]. Para fins de diferenciação das versões 1.0 e 2.0 do MHARS, é importante explicar que a versão 1.0 foi desenvolvida utilizando a versão original do M-Hub, como camada de aquisição, e o *Scalable Data Distribution Layer* (SDDL) (ver Seção 4.2). A versão 2.0 do MHARS foi desenvolvida utilizando a abordagem de middleware proposta neste trabalho, que utiliza uma versão modificada do M-Hub e substitui a SDDL, que não oferece suporte a QoC, pela *Context Data Distribution Layer* CDDL (ver Seção 5), que oferece esse tipo de suporte e corresponde à contribuição original da solução proposta.

Escolheu-se uma aplicação de *Ambiente Assisted Living* (AAL) pelo desejo de provar que a solução proposta poderia ser aplicada no desenvolvimento de sistemas pertencentes a um domínio relevante para a melhoria de vida das pessoas, tal como é o domínio da saúde. Para um melhor entendimento da importância de aplicações como a utilizada na prova de conceito, a Seção 6.1.1 apresenta uma breve contextualização acerca do que são sistemas de reconhecimento de atividades humanas. A Seção 6.1.2 apresenta os requisitos do MHARS 2.0. A Seção 6.1.3 mostra a implementação do sistema usando o M-Hub/CDDL. A Seção 6.1.4 apresenta os resultados e discussões da prova de conceito.

6.1.1 Sistemas de Reconhecimento de Atividades Humanas

O tratamento medicamentoso frequentemente não é suficiente para garantir a melhora do quadro clínico e a qualidade de vida dos pacientes. Em algumas situações, o acompanhamento das atividades físicas que eles realizam permite a elaboração de uma estratégia terapêutica individualizada, complementando o tratamento medicamentoso e ajudando a otimizar seus resultados. A prática de exercícios físicos com periodicidade e intensidade corretas pode ajudar a melhorar os resultados no tratamento de doenças cardíacas (e.g., hipertensão, insuficiência cardíaca e fibrilação atrial) e respiratórias (e.g., bronquite crônica, enfisema pulmonar e asma).

Sistemas de reconhecimento de atividades humanas têm a finalidade de inferir padrões de movimentação humana (e.g., andar, correr, sentar, ficar em pé), bem como calcular o estresse corporal (intensidade) com a qual a atividade está sendo executada pelo paciente. A inferência pode ser feita a partir da agregação de dados fornecidos por diferentes tipos de sensores de baixo nível (e.g., frequência cardíaca e acelerômetro). A taxa de acerto dos mecanismos de inferência usados pelo sistema varia de acordo com diversos fatores, dentre eles a acurácia dos dados/sensores e a técnica de classificação da atividade adotada. Ao detectar a atividade que o paciente está executando e o nível de estresse corporal correspondente, o sistema pode detectar situações em que a atividade executada favorece ou prejudica a saúde do paciente.

Sistemas de reconhecimento de atividades humanas permitem que profissionais de saúde acompanhem se o paciente está praticando ou não as atividades físicas que eles recomendam. Por exemplo, é possível saber se o paciente caminha ou corre com frequência ou se adota uma postura mais sedentária. A medição da intensidade permite verificar se o nível de esforço empregado na realização da atividade é compatível com os limites físicos do paciente. Esses limites variam de acordo com a condição crônica, idade, peso e outros fatores. O monitoramento dessa intensidade permite que o sistema alerte os profissionais de saúde em caso de fuga dos níveis de intensidade recomendados. Outra possibilidade é que o sistema de monitoramento solicite ao paciente que aumente ou diminua o esforço físico durante a execução da atividade, a fim de se adequar aos seus limites.

6.1.2 Requisitos do Sistema

O MHARS é um sistema de reconhecimento de atividades projetado para o uso em dispositivos móveis da plataforma Android e utiliza uma infraestrutura de computação em nuvem para armazenar e recuperar as informações inferidas do paciente. Esse sistema é capaz de reconhecer as seguintes atividades: caminhar, correr, saltar, de pé, deitado, subir e descer escadas. Para o reconhecimento da atividade, o sistema usa dados de aceleração e algorítimos de aprendizado de máquina [110]. O cálculo da intensidade é feito com base na frequência cardíaca e é mapeado para uma escala composta de uma série de intervalos chamados de zonas de intensidade [84].

O MHARS permite correlacionar a atividade e a intensidade inferidas com outros dados de contexto (tais como altitude, temperatura corporal e do ambiente, etc) para detectar situações específicas de alguma maneira relacionadas ao atual estado de saúde do paciente. Ele também fornece um mecanismo de decisão para definir um plano de ação (conjunto de ações) que deve ser executado sempre que situações perigosas para a saúde são detectadas. As ações a serem executadas são representadas através do uso de regras ECA (*Event-Condition-Action*). Por exemplo, para um paciente com doença cardiovascular crônica, é possível caracterizar uma situação como perigosa quando ele corre durante um longo período com uma intensidade alta não compatível com as recomendações médicas. Nessa situação, avisar o paciente sobre o risco de continuar a atividade e/ou informar o seu médico via SMS são exemplos de ações que podem ser executadas pela aplicação.

O MHARS foi desenvolvido pelo Laboratório de Sistemas Distribuídos Inteligentes (LSDi) da Universidade Federal do Maranhão (UFMA) em parceria com o Núcleo de Pesquisa em Nefrologia do Hospital Universitário (HUUFMA). Os requisitos gerais do sistema foram definidos por meio de entrevistas com profissionais de saúde. Os principais requisitos são:

- **Heterogeneidade de sensores**: o sistema deve ser capaz de interagir com variados tipos de sensores (e.g, portáteis, vestíveis ou embutidos em ambientes inteligentes), a fim de obter diferentes tipos de dados de contexto sobre o paciente (e.g., fisiológicos, movimentação e localização) e sobre o ambiente no qual ele se encontra (e.g., luz, temperatura, qualidade do ar);

- **Reconhecimento de atividades**: o sistema deve ser capaz de determinar a atividade executada pelo paciente em tempo próximo ao real, com base no agrupamento, classificação e processamento dos dados coletados;

- **Medição da intensidade**: o sistema deve ser ser capaz de medir a intensidade com a qual o paciente realiza cada atividade, bem como determinar se essa intensidade é baixa, alta ou moderada;

- **Inferência de situação**: o sistema deve ser capaz de inferir o estado de saúde em que o paciente se encontra (e.g., paciente com fibrilação atrial correndo com intensidade moderada em altitude de 1800 metros às 15h00min) por meio da

especificação de regras cuja execução infere diferentes situações com base em uma análise que combina dados dos sensores, atividade e intensidade, levando ainda em consideração o tratamento específico do paciente;

- **Tomada de decisão e execução de ações**: o sistema deve ser capaz de executar ações pré-definidas pelos profissionais de saúde para cada tipo de situação (e.g., enviar uma notificação de emergência para os profissionais de saúde), de modo a tentar melhorar ou manter o estado de saúde do paciente;

- **Suporte a mobilidade**: o sistema deve prover mobilidade ao usuários, permitindo que os pacientes e profissionais de saúde usem o sistema de reconhecimento de atividades de maneira flexível e em situações diversas;

- **Distribuição local e remota**: o sistema dever capaz de distribuir os dados e contexto, tanto para a aplicação que executa no dispositivo móvel do paciente como para as aplicações que executam nos dispositivos dos profissionais de saúde que acompanham o paciente remotamente.

Os requisitos do MHARS em relação à qualidade da informação são:

- **Acurácia**: o sistema deve ser capaz de obter a acurácia dos dados de contexto. A acurácia é importante pelo fato da taxa de acerto no reconhecimento de atividades, medição de intensidade e inferência de situação depender do quanto os dados de contexto coletados refletem a realidade em que o paciente se encontra. Com base na acurácia, o sistema pode estimar a probabilidade da atividade reconhecida, intensidade calculada e situação inferida estarem corretas;

- **Atributos disponíveis e completude**: o sistema deve ser capaz de descobrir quais atributos estão disponíveis para uma determinada informação, a fim de tornar possível calcular o grau de completude. Considerar a completude é relevante porque a eficiência dos mecanismos de reconhecimento da atividade e inferência da situação diminui se as informações exigidas estão incompletas;

- **Localização da fonte**: o sistema deve ser capaz de conhecer a localização da fonte de contexto a fim de determinar os locais onde as atividades reconhecidas e situações inferidas aconteceram. Em caso de emergência, a localização pode ser útil para a equipe de atendimento, parentes e cuidadores que desejam encontrar o

paciente. Entretanto, o paciente deve ter a prerrogativa de ocultar sua localização em certos lugares, atividades ou situações em que desejar ter privacidade.

Os requisitos do MHARS em relação à qualidade de serviço são:

- **Confiabilidade de entrega**: o sistema deve prover confiabilidade na entrega das informações de contexto, mesmo em situações onde a conectividade do gateway IoT e/ou das aplicações com a Internet é fraca ou intermitente. Eventos que caracterizam situações de emergência devem ser entregues apenas uma vez, de modo a evitar tomada de decisão e execução de ações redundantes;
- **Taxa de Atualização**: o sistema deve permitir que a taxa de envio e recebimento dos dados seja controlada, de modo que o monitoramento das atividades do paciente aconteça em intervalos regulares pré-programados.
- **Histórico**: o sistema deve ser capaz de armazenar dados/eventos acerca das atividades detectadas e situações inferidas ao longo do tempo de monitoramento;
- **Tempo de Validade**: o sistema deve prover meios para que os profissionais de saúde possam determinar o tempo de validade dos dados. Sendo assim, dados cujas validades expiraram devem ser removidos do histórico da aplicação.

6.1.3 Implementação do Sistema utilizando o Middleware Proposto

A Figura 6.1 apresenta a arquitetura da versão 2.0 do MHARS, que foi implementada com o M-Hub/CDDL. Essa arquitetura contempla tanto os componentes da camada de middleware (`S2PA`, `QoC Evaluator`, `Publisher`, `Subscribe e Connection Service`) como os componentes específicos da lógica de negócio da aplicação (`HARS`, `IMS`, `SAS e DMS`), que são descritos a seguir.

O MHARS utiliza o `S2PA` nas etapas de descoberta, conexão e aquisição de dados brutos a partir de sensores externos ao *smartphone*. O `S2PA` obtém dados da frequência cardíaca e aceleração nos três eixos (X,Y,Z) a partir de sensores embutidos no dispositivo vestível *Zephyr BioHarness 3*[1]. Esse dispositivo usa a tecnologia *Bluetooth Classic* (existem versões que utilizam BLE) para transmitir seus

[1] `http://zephyranywhere.com/products/BioHarness-3/`

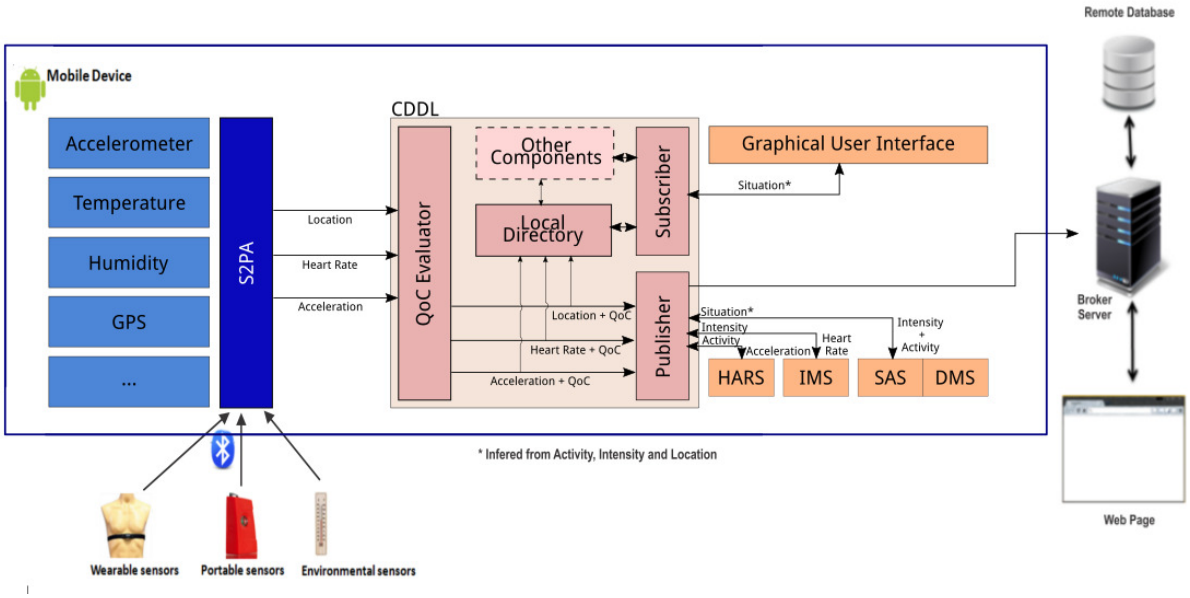

Figura 6.1: Arquitetura do MHARS implementada com o M-Hub/CDDL

dados a outros dispositivos compatíveis com essa WPAN. Para tornar o mecanismo de reconhecimento de atividades mais eficiente, `S2PA` também coleta dados de aceleração fornecidos por sensor embutido no próprio dispositivo móvel. Outro tipo de informação obtida pelo S2PA, a partir da interação com sensores internos do *smartphone*, é a localização (latitude, longitude e altitude) do paciente. Além das informações de contexto propriamente ditas, o `S2PA` obtém também meta-dados sobre a acurácia e os atributos disponíveis. Após serem coletados pelo `S2PA`, os dados de contexto são processados pelo `QoC Evaluator`, que avalia a qualidade da informação (que neste caso exige apenas o grau de completude) e calcula a QoC média. Em seguida, os dados de contexto são colocados na fila do `Publisher` [2], onde são tratados de acordo com os parâmetros de QoS do produtor. Na sequência, os dados de contexto são enviados ao *broker* através de uma conexão estabelecida pelo componente `Connection Service`. Uma vez que as etapas de reconhecimento das atividade, medição de intensidade e inferência de situação são realizadas por uma aplicação que executa no próprio dispositivo móvel do paciente, doravante chamada de aplicação móvel, o gateway publica os dados de contexto coletados apenas localmente, utilizando o `Micro Broker`. Por padrão, as mensagens contendo os dados coletados são publicadas com um intervalo mínimo de separação de 2,58

[2]Note que na versão atual do M-Hub/CDDL, o itinerário dos dados de contexto foi alterado, de modo que agora os dados passam pelo componente `Publisher` antes de chegar ao `QoCEvaluator`. Essa alteração visa expor para a camada de aquisição apenas a interface de publicação/subscrição.

segundos. De acordo com os especialistas em reconhecimento de atividades [4], essa janela de tempo é suficiente para a detecção de atividades executadas repetidamente.

A aplicação móvel recebe dados publicados utilizando um `Subscriber`, também conectado ao *Micro Broker* local. Essa aplicação processa as informações de contexto utilizando quatro componentes principais:

- **Human Activity Recognition Service** (`HARS`): componente que faz o reconhecimento da atividade que está sendo feita pelo usuário com base nos dados de aceleração. Esse componente executa uma etapa de pré-processamento (conversão, filtragem e refinamento) que coloca esses dados em um formato de entrada compatível com o requerido pelo classificador de atividades. O classificador utiliza algoritmo de aprendizagem de máquina IBk [110] para reconhecer os reais padrões de atividade com base nos valores previamente processados. Há uma fase de treinamento prévia para que o HARS possa funcionar corretamente. Dependendo da acurácia dos dados de contexto e do algoritmo de inferência utilizado, a taxa de acerto no reconhecimento das atividades pode chegar a 83%.

- **Intensity Measurement Service** (`IMS`): componente que realiza o cálculo da intensidade da atividade inferida pelo HARS. Esse cálculo leva em consideração a frequência cardíaca atual e a frequência cardíaca máxima que pode ser alcançada pelo paciente. A partir dessas informações, o IMS consegue determinar se a intensidade da atividade é baixa, moderada ou intensa.

- **Situation-Aware Service** (`SAS`): componente responsável por inferir as diferentes situações pré-estabelecidas nas quais o paciente pode estar durante o curso de uma atividade. A inferência de situação pode levar em consideração várias informações, tais como a atividade detectada, a intensidade medida, a localização obtida, a condição crônica do paciente, além de outros dados de contexto.

- **Decision Make Service** (`DMS`): componente responsável pela definição e execução das ações em resposta a uma situação inferida. A tomada de decisão é feita com base na execução de regras do tipo *Evento-Condição-Ação* (ECA).

A aplicação móvel envia os resultados do processamento dos dados de contexto para um *Server Broker* na nuvem. O reconhecimento de atividades gera um

fluxo de eventos contínuo. Os eventos gerados são publicados com confiabilidade em nível 0 (*at-most-once*), que não exige confirmação. Considera-se que essa configuração é adequada pois o intervalo entre um evento e outro é muito curto e a perda de alguns eventos não representa prejuízo significativo ao monitoramento do paciente. Eventos que informam atividades são publicados com uma taxa de atualização de 2.58 segundos, a mesma frequência com a qual a aplicação detecta as atividades. Os eventos que informam a situação do paciente são publicados com confiabilidade em nível 2 (*exaclty once*), pois não devem ser processados em duplicidade e a perda de um único evento de emergência pode prejudicar o paciente. Uma vez que são eventos periódicos, não cabe a definição de uma taxa de atualização. Sendo assim, esse tipo de evento é publicado tão logo aconteça. Por questões de privacidade dos dados de localização, o mecanismo de filtragem restringe a publicação de informações sobre atividades realizadas em determinadas coordenadas geográficas.

Uma aplicação web, que tem componentes executando na nuvem e no *browser* do cliente, exibe os dados de atividades e situações dos pacientes publicadas pelas aplicações móveis. Para receber essas informações, a aplicação web utiliza o componente `Subcriber` que também está conectado ao `Server Broker`. A taxa de atualização pode ser definida pela aplicação web, pois depende do intervalo mínimo com o qual essa aplicação exige ser notificada sobre as atividades e situações do paciente. As taxas de atualização padrão da aplicação web são as mesmas da aplicação móvel, mas podem ser configuradas. A aplicação web pode especificar também o tamanho do histórico e o tempo de validade dos dados. A configuração padrão do middleware armazena todas as atividades e situações (*keep-all*) recebidas pelo `Subscriber` por tempo indeterminado. Uma cópia dos dados recebidos vai para um banco de dados, onde ficam armazenados para fins de consulta.

6.1.4 Resultados e Discussões

O MHARS é uma aplicação de IoT com certo grau de complexidade pois além de requisitos como reconhecimento de atividades, medição de intensidade, inferência de situação e outros que são inerentes à sua lógica de negócio, essa aplicação possui diversos requisitos de gerenciamento de contexto (aquisição, processamento, distribuição e armazenamento), qualidade de contexto, tanto de QoI (acurácia,

atributos disponíveis, completude) como QoS (confiabilidade, taxa de atualização, histórico, tempo de validade), além da mobilidade. Na versão 1.0 do MHARS, a implementação da maior parte desses requisitos era responsabilidade do programador, visto que tanto M-Hub como SDDL, componentes da camada de middleware utilizadas para o desenvolvimento dessa versão, não tinham suporte a QoC na época.

A prova de conceito mostrou que os requisitos de gerenciamento e qualidade de contexto do MHARS foram atendidos de forma muito satisfatória pelo M-Hub/CDDL. Ao contrário do que ocorreu na versão 1.0 do MHARS, desta vez não foi mais necessário que o programador implementasse tais requisitos na camada de aplicação e/ou middleware , pois o M-Hub/CDDL proveu todo suporte a QoC que a aplicação necessitou. Logo, a partir deste caso particular, pode-se generalizar, em certo grau, que a abordagem proposta nesta tese é aplicável ao desenvolvimento de uma série de aplicações de IoT do domínio da saúde. Além disso, considerando que aplicações de outros domínios, como transporte, comércio e indústria, por exemplo, podem apresentar requisitos de gerenciamento de contexto e de QoC semelhantes, pode-se deduzir também que o M-Hub/CDDL pode ser utilizado no desenvolvimento de aplicações de IoT em diversos domínios, ainda que isso exija algumas extensões e/ou ajustes na solução proposta. Embora não tenha sido possível aplicar no MHARS todos os conceitos e mecanismos presentes na abordagem de middleware proposta, esta prova de conceito mostrou que eles são válidos, ou seja, que podem ser aplicados na resolução do problema ao qual a solução se destina, cumprindo assim o objetivo do M-Hub/CDDL, que é facilitar o desenvolvimento de aplicações de IoT cientes de contexto, principalmente aquelas que possuem múltiplos requisitos de QoC.

6.2 2º Estudo de Caso: Avaliação com Usuários

A avaliação qualitativa foi realizada com quatro participantes da disciplina Computação Móvel, ministrada pelo Prof. Dr. Francisco José da Silva e Silva, para alunos dos Programas de Pós-Graduação em Engenharia de Eletricidade (PPGEE) e Ciência da Computação (PPGCC) da Universidade Federal do Maranhão (UFMA), no primeiro semestre letivo de 2017. Uma vez que parte da avaliação da disciplina exige o desenvolvimento de uma aplicação de IoT móvel com requisitos de QoC, foi sugerido aos alunos que implementassem essa aplicação utilizando o M-Hub/CDDL.

Para que tivessem plenas condições de participar da avaliação, os alunos tiveram, ao longo da disciplina, aulas de Computação Ciente de Contexto e Qualidade de Contexto, bem como receberam um treinamento de curta duração sobre como programar aplicações móveis utilizando o M-Hub/CDDL. Os alunos também tiveram acesso à documentação *on-line* do middleware [3] e a um fórum de discussões, no qual podiam interagir entre si e com os desenvolvedores do M-Hub/CDDL.

6.2.1 Aplicação Proposta: Sistema Móvel de Rastreamento de Frotas

Foi sugerido aos participantes que desenvolvessem, num período de quatro semanas, um sistema móvel de monitoramento remoto de frotas de veículos, a fim de possibilitar a avaliação do middleware proposto com aplicações de IoT no domínio de transportes. O sistema proposto deveria ser composto por dois subsistemas básicos:

1. *Mobile Vehicle Monitoring System* (MVMS): aplicação que executa em dispositivos móveis embarcados nos veículos monitorados. Essa aplicação coleta dados de localização e velocidade. Os dados coletados são enviados para uma central de monitoramento remoto. Os motoristas podem utilizar essa aplicação para solicitar ajuda a profissionais de apoio que trabalham na central de monitoramento, em algumas situações, tais como, por exemplo, pane mecânica.

2. *Vehicle Tracking Center* (VTC): aplicação móvel utilizada pelos profissionais que atuam na central de monitoramento veicular. Essa aplicação recebe os dados dos veículos monitorados e pode ser usada para a comunicação com os motoristas.

Os requisitos de contexto do sistema são:

1. **Rastreamento em tempo próximo ao real**: o VTC deve receber periodicamente do MVMS dados de localização (latitude, longitude, altitude) do veículo. Ambas as aplicações devem exibir um mapa de rotas, permitindo que seus respectivos usuários acompanhem o caminho percorrido pelo veículo em tempo próximo ao real.

2. **Monitoramento de paradas**: o MVMS deve monitorar a velocidade média e a localização do veículo, a fim de determinar se este permanece muito tempo

[3] http://www.lsdi.ufma.br/projetos/cddl/doku.php

parado no mesmo local. Em caso de detecção de parada prolongada, o MVMS deve notificar a ocorrência desse evento por meio do envio de mensagem ao VTC.

3. **Pedido de socorro**: o MVMS deve prover aos motoristas um mecanismo de envio de pedidos de socorro ao VTC. Por sua vez, o VTC deve exibir aos profissionais da central de monitoramento uma lista de todos os pedidos de socorro recebidos.

Os requisitos de qualidade de contexto do sistema são:

- **Filtragem de dados de localização baseada na acurácia**: a fim de traçar no mapa o caminho percorrido pelo veículo da forma mais correta possível, o MVMS deve filtrar as publicações dos dados de localização com base na acurácia, ou seja, dados de localização com baixa acurácia não devem ser enviados ao VTC.

- **Taxa de atualização adaptável**: Para evitar o uso excessivo da interface de rede no dispositivo móvel (e consequentemente evitar que a bateria do dispositivo seja rapidamente consumida), o intervalo mínimo de separação entre as publicações de dados de contexto deve ser controlado e também adaptável, de acordo com o nível de bateria disponível (3, 8 e 15 segundos de intervalo entre publicações de dados para bateria apresentando níveis alto, médio e baixo, respectivamente). Naturalmente, para atender esse requisito de QoC, é necessário monitorar a ocorrência de eventos de mudança no nível da bateria do dispositivo móvel.

- **Confiabilidade de entrega**: dados de localização do veículo, por serem coletados e publicados de forma contínua, podem ser entregues ao VTC utilizando-se a política do melhor esforço. Eventos de detecção de parada prolongada exigem confiabilidade de entrega, mas não há problema se forem recebidos em duplicidade pelo VTC. Já as mensagens de pedido de socorro, além de exigir confiabilidade, não podem ser ser entregues ao VTC em duplicidade, a fim de evitar que a mesma solicitação seja registrada e atendida mais de uma vez.

- **Histórico de dados e ordem de destino**: o VTC deve organizar e exibir ao usuário um histórico dos pedidos de socorro recebidos. Considerando que os relógios dos dispositivos móveis usados por motoristas e profissionais da central de monitoramento estão sincronizados, o sistema deve ordenar as solicitações com base na ordem de publicação dos eventos, de modo a permitir que os pedidos de socorro mais antigos sejam registrados e atendidos primeiro.

6.2.2 Questionários de Avaliação

Cada desenvolvedor assinou um termo de consentimento de participação na pesquisa e preencheu dois tipos de questionário: pré-avaliação e avaliação. Como o nome sugere, o questionário de pré-avaliação foi aplicado antes da fase de treinamento no uso do M-Hub/CDDL, sendo utilizado para coletar informações sobre o perfil dos participantes, tais como: idade, formação acadêmica e experiência em desenvolvimento de software, tanto no que diz respeito a aplicações voltadas para computadores pessoais, como no que se refere a aplicações voltadas para dispositivos móveis. Também foi questionado se os participantes tinham conhecimentos sobre Computação Ciente de Contexto e Qualidade de Contexto, bem como se eles tinham algum tipo de experiência no desenvolvimento de aplicações de IoT com requisitos de QoC. O questionário de avaliação, aplicado somente após o desenvolvimento da aplicação proposta, foi utilizado para avaliar o grau de satisfação dos usuários em relação a aspectos considerados fundamentais no projeto de middleware voltado para aplicações de IoT com requisitos de qualidade de contexto, tais como: abstrações de programação, serviços, suporte à QoC e documentação. A metodologia empregada no treinamento dos participantes no uso do M-Hub/CDDL também foi avaliada por eles.

A Tabela 6.1 resume o perfil dos quatro participantes da avaliação.

Participante	Idade (anos)	Formação Acadêmica	Atuação Profissional	Experiência em Desenvolvimento de Software Desktop e/ou Web	Experiência em Desenvolvimento de Software para Dispositivos Móveis
1	22	Bacharel em Sistemas de Informação	-Estudante de Pós-Graduação (atual) -Iniciação Científica (anterior)	Sim, no âmbito acadêmico	Sim, no âmbito acadêmico
2	33	Bacharel em Ciência da Computação e Especialista em Engenharia de Sistemas	-Estudante de Pós-Graduação (atual) -Analista de Sistemas (atual)	Sim, no âmbito profissional	Não
3	21	Bacharel em Sistemas de Informação	-Estudante de Pós-Graduação (atual) -Iniciação Científica (anterior)	Sim, no âmbito acadêmico	Sim, no âmbito acadêmico
4	25	Tecnólogo em Análise e Desenvolvimento de Sistemas e Especialista em Engenharia de Software	-Estudante de Pós-Graduação (atual) -Professor de Informática (atual) -Analista de Sistemas (anterior)	Sim, no âmbito profissional	Não

Tabela 6.1: Perfil dos Participantes da Avaliação Qualitativa usando o M-Hub/CDDL

A idade dos participantes variou entre 21 e 33 anos. Todos são formados na área de Informática, sendo que um possui especialização em Engenharia de Software e outro possui especialização em Engenharia de Sistemas. Dois participantes afirmaram que atuaram apenas no âmbito acadêmico, como bolsistas de projetos de iniciação científica, enquanto que os outros dois já possuem alguma experiência profissional na área de informática. Desses dois, um é atualmente analista de sistemas e o outro é professor de informática, sendo que esse último também já atuou anteriormente como analista de sistemas. Em relação à experiência no desenvolvimento de software, todos afirmaram ter algum tipo de experiência, seja profissional ou acadêmica, no que se refere a aplicações *desktop* e/ou *web*, sendo que dois participantes afirmaram que também possuem experiência acadêmica no desenvolvimento de software para dispositivos móveis. De modo geral, os participantes afirmaram que não possuem experiência no desenvolvimento de aplicações cientes de contexto com requisitos de QoC, embora a maioria tenha afirmando que já tinha conhecimentos básicos sobre os temas ciência e qualidade de contexto antes do treinamento no uso do M-Hub/CDDL. Diante desse perfil, e principalmente após o treinamento, considera-se que os participantes estavam qualificados a utilizar e avaliar o middleware proposto. O treinamento do middleware durou duas semanas, sendo que foram realizados quatro encontros presenciais. Depois disso, os participantes tiveram quatro semanas para desenvolver a aplicação proposta utilizando o M-Hub/CDDL e avaliar o middleware.

O questionário de avaliação contém dezessete perguntas, sendo que as respostas possíveis seguem uma escala Likert: (1) Nada Satisfeito; (2) Pouco Satisfeito; (3) Consideravelmente Satisfeito; (4) Muito Satisfeito. Portanto, o nível zero indica o menor grau de satisfação do usuário em relação ao item avaliado, enquanto que o nível três indica o maior grau de satisfação em relação ao item avaliado. Para cada aspecto avaliado, os participantes eram motivados a fazerem comentários que julgassem pertinentes. O questionário de avaliação foi elaborado com a colaboração do Dr. Davi Viana dos Santos, docente da UFMA e pesquisador do LSDi, que atua nas áreas de Engenharia de Software (incluindo Software Experimental) e Qualidade de Software. As perguntas contidas no questionário de avaliação qualitativa, bem como uma síntese das respostas dos usuários participantes do estudo de caso, constam no Apêndice A.

6.2.3 Resultados e Discussões

A Tabela 6.2 mostra uma síntese dos resultados obtidos com a aplicação do questionário de avaliação do M-Hub/CDDL, levando em consideração critérios gerais e específicos de avaliação, bem como o grau de satisfação dos usuários do middleware.

Em relação às **abstrações de programação do middleware**, quase todos os critérios específicos avaliados (e.g., facilidade de uso, adequação das interfaces, abstração da heterogeneidade, expressividade das linguagens de monitoramento e filtragem) atingiram o maior grau, "Muito Satisfeito", de acordo com 100% dos usuários. A exceção à regra acontece na avaliação da expressividade da linguagem de consulta, na qual 50% dos usuários se declararam"Consideravelmente Satisfeitos"', enquanto que os outros 50% se declararam "Muito Satisfeitos" nesse critério. No geral, atribui-se o excelente resultado ao fato de que o M-Hub/CDDL oferece uma API de fácil entendimento, com métodos cujas assinaturas são bastante claras, que abstrai a complexidade do processo de gerenciamento de contexto (aquisição, processamento e distribuição de dados), permitindo que o programador concentre mais esforços na implementação de requisitos inerentes à lógica de negócio da aplicação.

Em relação aos **serviços do middleware**, no que se refere especificamente à estabilidade na execução dos serviços, o grau de satisfação declarado por 75% dos usuários foi "Consideravelmente Satisfeito", sendo que 25% deles se declarou "Muito Satisfeito" nesse critério. Em se tratando especificamente da ausência de erros, o grau de satisfação dos usuários variou de forma muito equilibrada entre os graus "Consideravelmente Satisfeito" e "Muito Satisfeito", cada um com 50% dos usuários. No geral, atribui-se o bom resultado ao fato de que os serviços do M-Hub/CDDL executam de maneira bastante estável e sem erros, sendo que poucos *bugs* foram reportados pelos desenvolvedores, mas corrigidos ainda no período de avaliação.

Em relação ao **suporte à QoC**, 100% dos participantes se declararam "Muito Satisfeitos" no que diz respeito a quase todos os critérios específicos avaliados: funcionamento correto dos parâmetros de QoC, quantitativo suficiente de parâmetros de QoC disponibilizados pelo middleware e utilidade desses parâmetros para outras aplicações de IoT. Especificamente no que se refere ao atendimento dos requisitos de QoC da aplicação desenvolvida no estudo de caso, 50% dos usuários se declarou "Muito Satisfeito" e 50% se declarou "Consideravelmente Satisfeito". Atribui-se o

excelente resultado ao fato de que o M-Hub/CDDL oferece ao programador um conjunto bastante amplo parâmetros de qualidade de contexto, que não se restringe apenas à qualidade da informação, como geralmente acontece em outras abordagens de middleware da literatura, mas que abrangem também a qualidade do serviço de distribuição, podendo atender assim uma grande variedade de aplicações de IoT/IoMT. Além disso, todos os parâmetros de QoC requisitados pela aplicação do estudo de caso foram atendidos pelo middleware e funcionaram corretamente, sendo que praticamente todos se comportaram como esperado durante o período de testes.

Em relação à **documentação do software/código**, tanto no que diz respeito à qualidade da documentação *on-line*, como no que se refere à compreensão das mensagens utilizadas no tratamento de exceções, 50% dos usuários se declararam "Consideravelmente Satisfeitos" e os outros 50% deles se declararam "Pouco Satisfeitos". Atribui-se o resultado regular, no geral, ao fato da documentação do M-Hub/CDDL ainda não estar completa, requerendo mais exemplos de uso da API, bem como algumas atualizações para que se torne totalmente consistente com a versão mais recente do middleware. Além disso, ficou evidenciada a necessidade de enriquecer a documentação com maiores detalhes sobre o tratamento de exceções. Apesar disso, como enaltecido pelos próprios usuários, a facilidade de compreensão da API do M-Hub/CDDL compensou, de certo modo, as inconsistências da documentação, de forma que não houve prejuízo significativo no aprendizado de uso do middleware.

Em relação à **metodologia de treinamento/avaliação**, 100% dos usuários se declarou "Muito Satisfeito", tanto com a maneira como o treinamento de uso do middleware foi realizado, como com as ferramentas empregadas, a exemplo do fórum de discussão *on-line*. Atribui-se o excelente resultado graças à forte interação entre os usuários e os desenvolvedores do middleware, que estiveram em contato de forma presencial e também à distância. Além do treinamento prático no uso do middleware propriamente dito, os participantes tiveram aulas teóricas sobre ciência de contexto e qualidade de contexto, temas que alguns já conheciam, mas apenas superficialmente.

Portanto, além de provar que os conceitos do M-Hub/CDDL se aplicam também ao domínio de transportes, os resultados mostram que M-Hub/CDDL atende de forma considerável ou até muito satisfatória, diversos requisitos de gerenciamento e qualidade de contexto, bem como provaram grande potencial dessa abordagem de middleware como facilitadora do desenvolvimento de aplicações de IoT.

Critério de Avaliação Geral	Critério de Avaliação Específico	Grau de Satisfação
Abstrações de Programação	Facilidade de uso (programação,instalação e configuração)	Muito Satisfatório (100%)
	Adequação das interfaces de programação disponibilizadas	Muito Satisfatório (100%)
	Abstração da heterogeneidade e redução da complexidade	Muito Satisfatório (100%)
	Facilidade/Flexibilidade na distribuição de dados de contexto	Muito Satisfatório (100%)
	Expressividade da linguagem de monitoramento de contexto	Muito Satisfatório (100%)
	Expressividade da linguagem de filtragem de dados de contexto	Muito Satisfatório (100%)
	Expressividade da linguagem de consulta de serviços de contexto	Consideravelmente (50%) ou Muito Satisfatório (50%)
Serviços do Middleware	Comportamento estável na execução dos serviços do middleware	Consideravelmente (75%) ou Muito Satisfatório (25%)
	Ausência de Erros (*Bugs*) na execução dos serviços do middleware	Consideravelmente (50%) ou Muito Satisfatório (50%)
Suporte à Qualidade de Contexto	Atendimento de todos os requisitos de QoC da aplicação alvo	Consideravelmente (50%) ou Muito Satisfatório (50%)
	Funcionamento correto de todos os parâmetros de QoC utilizados	Muito Satisfatório (100%)
	Utilidade dos parâmetros de QoC para outras aplicações de IoT	Muito Satisfatório (100%)
	Quantidade suficiente de parâmetros de QoC disponibilizados	Muito Satisfatório (100%)
Documentação de Software/Código	Facilidade de compreensão das mensagens de tratamento de exceção	Consideravelmente (50%) ou Pouco Satisfatório (50%)
	Qualidade da documentação *on-line* do middleware disponibilizada	Consideravelmente (50%) ou Pouco Satisfatório (50%)
Metodologia de Treinamento/Avaliação	Qualidade da metodologia de treinamento/avaliação do middleware	Muito Satisfatório (100%)
	Contribuições do Fórum de Discussões no aprendizado do middleware	Muito Satisfatório (100%)

Tabela 6.2: Resultado da Avaliação Qualitativa do M-Hub/CDDL junto aos Usuários

6.3 Conclusão do Capítulo

Este capítulo apresentou a metodologia e os resultados de dois estudos de caso, ambos referentes à avaliação qualitativa da abordagem de middleware proposta, considerando diferentes domínios de aplicação relevantes em IoT: saúde e transporte.

O primeiro estudo de caso consistiu no desenvolvimento de um sistema móvel de reconhecimento de atividades humanas. Além dos requisitos específicos da lógica de negócio da aplicação alvo (reconhecimento de atividade, medição da intensidade, inferência de situação, tomada de decisão e execução de ações), o sistema desenvolvido possui requisitos inerentes ao gerenciamento de contexto (aquisição, processamento, armazenamento e distribuição), além de vários requisitos de QoC (acurácia, localização da fonte, atributos disponíveis, completude, confiabilidade, taxa de atualização, histórico e tempo de validade). O objetivo dessa avaliação era provar que os conceitos da abordagem de middleware proposta poderiam ser, de fato, aplicados ao desenvolvimento de aplicações de IoT do mundo real. Os resultados obtidos foram muito satisfatórios, uma vez que foi constatado que o M-Hub/CDDL atendeu todos requisitos de gerenciamento de contexto e de QoC da aplicação alvo.

O segundo estudo de caso consistiu no desenvolvimento de um sistema móvel de monitoramento remoto de frotas de veículos. Assim como a anterior, a aplicação desse estudo de caso também tem requisitos específicos da sua lógica de negócio (rastreamento de localização, monitoramento de paradas e envio de pedidos de socorro), bem como requisitos de gerenciamento e qualidade de contexto (acurácia, confiabilidade de entrega, taxa de atualização, histórico e ordem de destino). O objetivo desse estudo de caso era permitir que os usuários do M-Hub/CDDL, que no caso são os próprios desenvolvedores da aplicação, avaliassem a solução proposta, levando em consideração aspectos relevantes no projeto de middleware de contexto com suporte à QoC (abstrações de programação, serviços do middleware, parâmetros de QoC oferecidos e documentação). Os resultados obtidos nessa avaliação também foram muito satisfatórios, em relação à grande maioria dos aspectos considerados.

Considerando os resultados obtidos, concluí-se que a abordagem proposta tem grande potencial de aplicação em dominós com requisitos de QoC semelhantes aos utilizadas na avaliação. A solução cumpriu o seu objetivo facilitar o desenvolvimento de aplicações com requisito de QoC, validando a hipótese de investigação.

7 Avaliação Quantitativa

O objetivo geral da avaliação quantitativa é medir o desempenho da abordagem de middleware proposta, a partir da realização de experimentos controlados e aferição de métricas. Entretanto, avaliar quantitativamente todos os aspectos de uma abordagem de middleware, como a M-Hub/CDDL, que agrega tantos conceitos, componentes e mecanismos, não é tarefa trivial. Diante disso, os esforços de avaliação quantitativa do M-Hub/CDDL voltaram-se para a realização dos seguintes experimentos: (i) avaliação do tempo total de entrega dos dados ao consumidor; (ii) avaliação da confiabilidade de entrega em redes instáveis; (iii) avaliação do tempo de monitoramento de variações de QoC; (iv) avaliação do consumo de memória e (v) avaliação do consumo de bateria. A seguir cada conjunto de experimentos realizado, com os seus respectivos resultados e discussões, é apresentado em uma subseção à parte.

7.1 Avaliação do Tempo Total de Entrega dos Dados

Antes de chegar ao consumidor, o dado de contexto passa por diferentes etapas/componentes do middleware: (1) o dado bruto e metainformações sobre a QoI disponíveis são coletados pelo `S2PA Service`, onde é transformado para o modelo de contexto do CDDL; (2) `S2PA` envia o dado para o `Publisher`; (3) o dado é interceptado ainda na fila de publicação pelo `QoC Evaluator`, que realiza a avaliação da QoI; (4) o dado retorna ao `Publisher` para aplicação de políticas de QoS configuradas com parâmetros do produtor; (5) o dado segue para o `Connection Service`; (6) o dado é enviado a um `Micro Broker` ou `Server Broker` ou até ambos, que o encaminhará aos subscritores que assinaram o tópico de publicação correspondente (que pode ter sido descoberto pela aplicação consumidora por meio de uma consulta ou anúncio); (7) o dado chega ao `Connection Service` do subscritor (que pode ser o mesmo do publicador em caso de aplicação local); (8) o dado chega ao `Subscriber`, onde aplica-se as políticas de QoS configuradas com parâmetros fornecidos pela aplicação

consumidora; (9) o dado é repassado para a aplicação consumidora por meio de um `Subscriber Listener`; (10) o dado é processado pela aplicação consumidora.

O objetivo específico deste conjunto de experimentos é avaliar a eficiência do serviço de distribuição do M-Hub/CDDL, em relação ao tempo total de entrega dos dados ao consumidor, levando-se em consideração diferentes níveis de confiabilidade de entrega e tipos de redes, pois são fatores que influenciam na qualidade do serviço.

Foram estabelecidas três métricas de desempenho:

- **Tempo de Comunicação entre Publicador e Subscritor**: equivale à soma do atraso imposto pela comunicação em rede, ou barramento local, com o tempo de permanência/processamento da mensagem no *Broker* MQTT local ou remoto. O tempo de comunicação entre publicador e subscritor é diretamente afetado pelo nível de confiabilidade de entrega utilizado pelos publicadores e subscritores, bem como por fatores externos, sobre os quais o middleware não tem controle, tais como a largura de banda, latência e taxa de perda de pacotes da rede.

- **Tempo de Processamento da Informação no Middleware**: equivale à soma dos tempos que a informação de contexto leva para transitar ou ser processada pelos componentes do middleware no produtor (intervalo entre a chegada do dado no publicador e envio da mensagem ao *Broker* MQTT) e no consumidor (intervalo entre a chegada da mensagem no subscritor e entrega do dado para a aplicação). O tempo de processamento da informação no middleware inclui operações *marshalling* e *unmarshalling*, avaliação da QoI e aplicação de políticas de QoS de acordo com parâmetros estabelecidos pelos publicadores e subscritores.

- **Tempo Total de Entrega da Informação de Contexto ao Consumidor**: equivale à soma dos tempos de comunicação entre publicador e subscritor e o tempo de processamento da informação no middleware, ou seja, é o tempo decorrido entre a chegada do dado no gateway e sua efetiva entrega à aplicação consumidora.

Todas essas métricas já são nativamente computadas pelo M-Hub/CDDL, em milissegundos, sendo disponibilizadas para a aplicação como metadados.

O tempo de comunicação é calculado com base na diferença entre o *timestamp* de envio da mensagem pelo componente `Publisher` e o *timestamp* de

chegada da mensagem no componente `Subscriber`, conforme equação a seguir:

$$TempoDeComunicacao = TimestampDoPublicador - TimestampDoSubscritor \quad (7.1)$$

O tempo total de entrega dos dados é calculado com base na entre o *timestamp* de chegada da informação no componente `S2PA` e o *timestamp* de efetiva entrega da mensagem para a aplicação consumidora, conforme equação a seguir:

$$TempoTotalDeEntrega = TimestampDoS2PA - TimestampDaAplicacao \quad (7.2)$$

Ressalta-se que o tempo de chegada da informação no `Subscriber` não é igual ao tempo de entrega da informação para a aplicação. A diferença entre esses tempos, que podem até ser muito próximos, depende dos parâmetros de QoS do subscritor. Por exemplo, se o subscritor utiliza o parâmetro controle de latência, então o middleware irá impor um atraso adicional na entrega da mensagem para a aplicação.

O tempo de processamento do dado no middleware é calculado subtraindo-se o tempo de comunicação do tempo total de entrega, conforme equação a seguir:

$$TempoDeProcessamento = TempoTotalDeEntrega - TempoDeComunicacao \quad (7.3)$$

7.1.1 Descrição do Experimento

O experimento consistiu na execução de uma aplicação de teste que envia e recebe dados. Em cada repetição do experimento foram publicadas 1800 mensagens, sendo que a frequência de publicação era de 1 mensagem por segundo. A duração de cada repetição do experimento é de 30 minutos. O tamanho do *payload* de cada mensagem, neste conjunto de experimentos, assim como em todos os experimentos que são apresentados nas próximas seções, é de 1024 bytes.

Além da aplicação de teste, foram utilizados os seguintes recursos:

- **01 Sensor Vestível**: utilizado para a geração dos dados de contexto. Foi utilizado o dispositivo Zephyr Bioharness 3, o mesmo utilizado na prova de conceito (ver Seção 6.1.2). O S2PA foi configurado para coletar apenas dados de batimento cardíaco. A frequência de medição do sensor é de aproximadamente 1 Hz.

- **01 Smartphone**: usado para executar a aplicação teste. Foi utilizado um LG-K430F, que possui 2 GB de memória RAM, processador quad-core 1.2 GHz, sistema operacional Android 6.0, Bluetooth e Wifi 802.11.

- **01 Notebook**: dispositivo utilizado para a execução da IDE de monitoramento da aplicação teste (Android Studio) e para execução de um server broker em experimentos com uso rede local. Foi utilizado um DELL Inspiron 15-5557, que possui 16 GB de memória RAM, processador Intel Core i7 2.5 GHz e sistema operacional Ubuntu 16.04 LTS.

- **01 Servidor**: servidor do LSDi que executa um *Server Broker* MQTT, empregado nos experimentos com uso da Internet. Possui 32 GB de RAM e processador Intel Xeon 1.7GHz. O *Broker* pode ser acessado pela url `www.lsdi.ufma.br:1883`.

Em relação à confiabilidade, os experimentos variam em três tipos:

- **Experimentos com nível zero de confiabilidade**: neste tipo de experimento foi empregado o parâmetro *at most once*, fazendo com que o middleware não exigisse confirmação de entrega das mensagens;

- **Experimentos com nível 1 de confiabilidade**: neste tipo de experimento foi empregado o parâmetro *at least once*, fazendo com que o middleware exigisse confirmação de entrega utilizando um único *handshake*.

- **Experimentos com nível 2 de confiabilidade**: neste tipo de experimento foi empregado o parâmetro *exacly once*, fazendo com que o middleware exigisse confirmação de entrega utilizando um duplo *handshake*.

Em relação às configurações rede, os experimentos variam em três tipos:

- **Experimentos sem uso da rede**: neste tipo de experimento não foi usada nenhuma conexão de rede. Publicador e subscritor se conectavam a um Micro Broker que executava no mesmo dispositivo da aplicação teste (o smartphone).

- **Experimentos com uso de rede local**: neste tipo de experimento foi utilizada uma rede WLAN. Publicador e subscritor se conectam a um *Server Broker* externo. Neste caso, o *Server Broker* executava no notebook, que também atuou como

ponto de acesso à rede. A WLAN era formada apenas pelo smartphone e o notebook, sendo que não havia qualquer conexão com a Internet.

- **Experimentos com uso de Internet Fixa**: neste tipo de experimento foi utilizada uma rede Wifi doméstica. O acesso a Internet era feito por meio de um roteador/modem ADSL sem fio. Publicador e subscritor se conectavam a um *Server Broker* que executava no servidor do LSDi.

- **Experimentos com uso de Internet Móvel**: neste tipo de experimento foi utilizada conexão 4G. Publicador e subscritor se conectavam um *Server Broker* que executava no servidor do LSDi.

É importante detalhar que, no período de realização dos experimentos utilizando a Internet, a aplicação teste executava em uma rede distinta daquela em que executava o broker. O servidor do LSDi estava conectado à rede institucional da Universidade Federal do Maranhão, localizada na cidade de São Luís, enquanto que a aplicação teste se conectava à Internet utilizando os serviços de uma operadora de telefonia/banda denominada Oi [1], na cidade de São José de Ribamar. Ambas as cidades fazem parte de uma região metropolitana denominada Ilha de São Luís, no Estado do Maranhão. A distância entre essas cidades é de aproximadamente 20 km.

Para cada tipo de rede foram testados todos os níveis de confiabilidade possíveis. Portanto, foram realizados experimentos com 12 configurações diferentes. Considerando que cada tipo de experimento/configuração foi repetido 5 vezes, alcançou-se um total de 60 execuções.

Portanto, justifica-se o conjunto de experimentos apresentados devido ao fato de que ele permite atingir o objetivo da avaliação, uma vez que foram caracterizados cenários nos quais se pode avaliar abordagem de middleware em diferentes tipos de rede e níveis de confiabilidade. Por um lado, é possível observar o desempenho da solução em redes fixas e móveis, utilizando um Server Broker. Por outro, há cenários em que a abordagem de middleware foi avaliada utilizando somente o Micro Broker, permitindo assim observar o desempenho da solução sem qualquer influência da rede. Sendo assim, os cenários simulados são bastante heterogêneos, o

[1] `http://www.oi.com.br/`

que permite avaliar o desempenho da solução sob diferentes condições do ambiente de execução das aplicações de IoT.

7.1.2 Resultados e Discussões

As tabelas 7.1, 7.2 e 7.1 apresentam os resultados obtidos nos experimentos em relação ao tempo de comunicação entre publicador e subscritor, tempo de processamento da informação nos componentes do middleware e tempo total de entrega do dados para a aplicação consumidora, levando-se em consideração a média obtida com as repetições dos experimentos de cada configuração e níveis de confiabilidade testados. Calculou-se também os valores máximos, mínimos e o desvios padrões médios, referentes à cada métrica. Definiu-se também que o desempenho representativo de cada tipo de rede seria dado uma média dos resultados obtidos.

#	Configuração de Rede	Confiabilidade	Média (ms)	Máximo (ms)	Minimo (ms)	Desvio Padrão (ms)
1	Sem Rede (Micro Broker)	0	62.25	66.61	59.60	3.20
2	Sem Rede (Micro Broker)	1	186.17	194.99	177.92	6.45
3	Sem Rede (Micro Broker)	2	236.68	241.77	232.87	3.35
4	**Média dos Experimentos sem uso da Rede**		**161.70**	**167.79**	**156.80**	**4.33**
5	Rede Local (Server Broker)	0	74.37	80.79	71.22	3.94
6	Rede Local (Server Broker)	1	92.51	99.92	88.59	4.60
7	Rede Local (Server Broker)	2	126.23	131.18	121.82	4.48
8	**Média dos Experimento usando Rede Local Isolada**		**97.70**	**103.97**	**93.88**	**4.34**
9	Internet Fixa (Server Broker)	0	233.35	241.40	228.61	5.56
10	Internet Fixa (Server Broker)	1	417.04	490.48	375.22	46.57
11	Internet Fixa (Server Broker)	2	681.42	711.17	666.79	17.64
12	**Média dos Experimentos usando Internet Fixa (Wifi+ADSL)**		**443.94**	**481.01**	**423.54**	**23.26**
13	Internet Móvel (Server Broker)	0	262.55	273.69	254.81	7.98
14	Internet Móvel (Server Broker)	1	472.12	483.05	456.64	9.98
15	Internet Móvel (Server Broker)	2	807.14	845.54	782.28	25.36
16	**Média dos Experimentos usando Internet Móvel (4G)**		**513.94**	**534.09**	**497.91**	**14.44**

Tabela 7.1: Avaliação do Serviço de Distribuição em relação ao Tempo de Comunicação

Resultados e Discussões dos Experimentos sem Uso da Rede

Em relação aos experimentos sem uso da rede, pode-se observar que o tempo médio de comunicação com nível de confiabilidade 0 (62,25 ms), praticamente duplica ao elevar o nível de confiabilidade para 1 (chegando a 186,17 ms) e triplica,

#	Configuração de Rede	Confiabilidade	Média (ms)	Máximo (ms)	Mínimo (ms)	Desvio Padrão (ms)
1	Sem Rede (Micro Broker)	0	37.21	42.05	34.01	4.21
2	Sem Rede (Micro Broker)	1	31.34	36.91	26.15	3.88
3	Sem Rede (Micro Broker)	2	34.95	37.66	32.84	2.26
4	**Média dos Experimentos sem uso da Rede**		**34.50**	**38.87**	**31.00**	**3.45**
5	Rede Local (Server Broker)	0	34.66	37.20	32.28	2.13
6	Rede Local (Server Broker)	1	37.14	39.44	34.36	1.83
7	Rede Local (Server Broker)	2	36.46	42.37	34.24	3.39
8	**Média dos Experimentos usando Rede Local isolada**		**36.09**	**39.67**	**33.63**	**2.45**
9	Internet Fixa (Server Broker)	0	33.79	37.34	31.75	2.31
10	Internet Fixa (Server Broker)	1	34.60	36.20	33.68	0.97
11	Internet Fixa (Server Broker)	2	35.97	40.06	33.28	3.11
12	**Média dos Experimentos usando Internet Fixa (Wifi+ADSL)**		**34.79**	**37.87**	**32.90**	**2.13**
13	Internet Móvel (Server Broker)	0	36.42	42.83	34.27	3.64
14	Internet Móvel (Server Broker)	1	34.60	36.20	33.68	0.97
15	Internet Móvel (Server Broker)	2	34.37	36.92	32.14	2.35
16	**Média dos Experimentos usando Internet Móvel (Wifi+ADSL)**		**35.13**	**38.65**	**33.36**	**2.32**

Tabela 7.2: Avaliação do Serviço de Distribuição em relação ao Tempo de Processamento

#	Configuração de Rede	Confiabilidade	Média (ms)	Máximo (ms)	Mínimo (ms)	Desvio Padrão (ms)
1	Sem Rede (Micro Broker)	0	99.46	108.19	93.60	7.35
2	Sem Rede (Micro Broker)	1	217.51	225.84	204.07	8.89
3	Sem uso da Rede (Micro Broker)	2	271.64	278.91	266.51	5.31
4	**Média dos experimento sem uso da rede**		**196.20**	**204.31**	**188.06**	**7.18**
5	Rede Local (Server Broker)	0	112.46	120.45	105.30	6.52
6	Rede Local (Server Broker)	1	128.60	134.85	123.47	4.24
7	Rede Local (Server Broker)	2	162.69	167.02	156.37	4.49
8	**Média dos Experimentos usando Rede Local isolada**		**134.58**	**140.77**	**128.38**	**5.08**
9	Internet Fixa (Server Broker)	0	267.14	278.74	260.41	7.73
10	Internet Fixa (Server Broker)	1	440.07	472.06	412.94	24.68
11	Internet Fixa (Server Broker)	2	715.80	743.30	703.24	16.59
12	**Média dos Experimentos usando Internet Fixa (Wifi+ADSL**		**474.33**	**498.03**	**458.86**	**16.33**
13	Internet Móvel (Server Broker)	0	298.97	309.12	289.59	9.24
14	Internet Móvel (Server Broker)	1	506.72	519.25	490.32	10.77
15	Internet Móvel (Server Broker)	2	844.15	882.05	822.06	23.86
16	**Média dos Experimentos usando Internet Móvel (4G)**		**549.95**	**570.14**	**533.99**	**14.62**

Tabela 7.3: Avaliação do Serviço de Distribuição em relação ao Tempo Total de Entrega

ao elevar o nível de confiabilidade para 2 (chegando a 236,68 ms). Já o tempo médio total de entrega dos dados usando confiabilidade 0 (99,46 ms) praticamente duplica ao elevar o nível de confiabilidade para 1 (chegando a 217,51 ms) e triplica ao elevar o nível de confiabilidade para 2 (chegando a 271,64 ms). Como esperado, o nível

0 de confiabilidade apresentou melhor desempenho que os níveis 1 e 2 porque não exige confirmação de entrega das mensagens. Já o nível 1 teve melhor desempenho que o nível 2 porque o mecanismo de confirmação de entrega utilizado pelo primeiro (handshake simples) requer menos trocas de mensagens que o empregado pelo segundo (handshake duplo). Sendo assim, o desempenho do Micro Broker se degrada à medida em que o nível de confiabilidade utilizado aumenta, pois quanto maior o nível de confiabilidade, mais recursos de memória e processamento são requeridos.

Resultados e Discussões dos Experimentos usando Rede Local Isolada

Em relação aos experimentos com uso de rede local isolada, que obteve o melhor desempenho médio, pode-se observar que, assim como nos experimentos sem o uso da rede, e também pelos mesmos motivos, o tempo de comunicação e o tempo de entrega total dos dados usando o nível de confiabilidade 0 (74,37 ms e 112,46 ms, respectivamente) também foram menores que aqueles utilizando os níveis 1 (no qual o tempo de comunicação foi de 92,51 ms e o tempo total de entrega dos dados foi de 128,60 ms) e 2 (no qual o tempo de comunicação foi de 126,23 ms e o tempo total de entrega dos dados foi de 162,69 ms). Entretanto, a diferença entre os tempos de comunicação, e consequentemente entre os tempos totais de entrega, utilizando os níveis 0 e 2, é bem menor que aquelas obtidas nos experimentos sem o uso da rede.

Desta vez, no que diz repeito ao tempo de comunicação, a diferença entre os níveis 0 e 1, bem como a diferença entre os níveis 1 e 2, é menor que 25%. Em relação ao tempo total de entrega, a diferença entre os níveis 0 e 1 é menor que 25%, enquanto que a diferença entre os níveis 1 e 2 é menor que 30%. O motivo pelo qual isto ocorre está fortemente relacionado ao tipo de broker que cada um utiliza. Enquanto que nos experimentos sem uso da rede utilizou-se um micro broker, que executava no próprio dispositivo móvel, nos experimentos usando a rede local empregou-se um server broker, que executava no notebook. A diferença em relação ao poder de processamento e memória disponíveis em ambas as plataformas de execução faz com que o desempenho do micro broker se deteriore mais rapidamente que o do server broker, quando se eleva a confiabilidade, fazendo com que o segundo envie mensagens aos destinatários mais rapidamente que o primeiro.

Essa diferença de desempenho, em favor do server broker, é quase sempre suficiente para compensar o atraso imposto pela comunicação da rede local isolada. A exceção à regra acontece quando ambos os tipos de experimentos utilizam o nível 0 de confiabilidade. Neste caso, graças a taxa de publicação utilizada, o custo de processamento das mensagens (que não exigem confirmação de entrega) é tão baixo, que o atraso imposto pela rede local isolada se tornou mais significativo que o custo do processamento realizado pelo broker, fazendo com que o tempo de comunicação e o tempo total de entrega dos dados obtidos nos experimentos sem uso da rede, para esse nível de confiabilidade, sejam os menores dentre todos os experimentos realizados.

Resultados e Discussões dos Experimentos usando Internet Fixa e Móvel

Em relação aos experimentos usando Internet Wifi e 4G, em ambos os casos, os tempos de comunicação obtidos utilizando o nível 0 de confiabilidade (233,35 ms usando Wifi e 262,55 ms usando 4G) praticamente duplicam ao elevar a confiabilidade para o nível 1 (chegando a 417,04 usando Wifi e 472,12 usando 4G) e triplicam ao elevar a confiabilidade para o nível 2 (681,42 usando Wifi e 807,14 usando 4G) Já os tempos de entrega total dos dados, utilizando o nível de confiabilidade 0 (267,14 ms usando Wifi e 298,97), aumentam cerca de 60% ao elevar o nível de confiabilidade para 1 (chegando a 440,07 ms usando Wifi e 506,72 usando 4G). Já os tempos obtidos utilizando o nível de confiabilidade 2 aumentam cerca de 60% em relação ao nível 1 (chegando a 715,80 ms usando Wifi e 844,15 usando 4G). Como esperado, o atraso imposto pela Internet fez com que esses tivessem desempenhos inferiores às demais configurações de rede.

Conclusão do Experimento

Analisando-se todos os tipos de experimentos, pode-se observar que o desempenho do middleware nos cenários testados em relação ao tempo médio de processamento dos dados antes do envio e após o recebimento dos dados é sempre baixo, próximo a 35 ms, com pequenas variações. Sendo assim, de maneira geral, pode-se concluir que o tempo de processamento imposto pelo middleware possui um impacto consideravelmente menor sobre o tempo total de entrega dos dados que aquele imposto pela comunicação (atraso de transmissão + tempo de processamento no broker). Os tempos de comunicação observados para todas as configurações de

rede, mesmo nos experimentos em que se utilizou a Internet, cujos resultados tendem a variar de acordo com o estado da rede, permitem concluir que, em condições de rede semelhantes, o MHub/CDDL oferece comunicação com baixo *overhead*, possibilitando uma transmissão de dados de sensores em tempo próximo ao real, graças à adoção de um protocolo otimizado para a IoT.

7.2 Avaliação da Confiabilidade na Entrega dos Dados

Aplicações de IoT, de maneira geral, estão sujeitas à ocorrência de falhas na entrega das informações de contexto. Em ambientes móveis, nos quais a conectividade tende a ser mais fraca e/ou intermitente, a possibilidade de falhas na entrega dos dados de contexto aumenta consideravelmente, o que prejudica ainda mais as aplicações IoT, principalmente aquelas que possuem a mobilidade irrestrita como requisito (IoMT).

O objetivo específico deste conjunto de experimentos é avaliar a eficiência do serviço de distribuição do M-Hub/CDDL. Entretanto, o aspecto de avaliação a ser considerado é a confiabilidade de entrega do middleware em cenários de conectividade intermitente.

A métrica adotada é o percentual de perda de mensagens.

7.2.1 Descrição do Experimento

Foi utilizada a mesma aplicação teste do cenário anterior, bem como os mesmos recursos de hardware. Novamente, cada experimento teve duração aproximada de 30 minutos e a taxa de publicação das mensagens era de 1 mensagem por segundo. As desconexões de rede eram geradas pela própria aplicação, por meio de um desligamento programado da interface de rede sem fio do dispositivo móvel, fazendo com que a conexão TCP/IP, entre a aplicação teste e o *Server Broker*, fosse fechada de maneira inesperada. Em relação à intermitência de conectividade, foram simulados dois tipos de cenários:

- **Baixa Intermitência**: neste cenário, a conectividade com *Server Broker* MQTT era interrompida programaticamente em um intervalo que variava randomicamente entre 120 e 180 segundos, de acordo com uma distribuição uniforme .

- **Alta Intermitência**: neste cenário, a conectividade com *Server Broker* MQTT era interrompida programaticamente em um intervalo que variava randomicamente entre 12 e 18 segundos, de acordo com uma distribuição uniforme no intervalo. Portanto, a taxa de intermitência era 10 vezes maior que a do cenário anterior.

Em ambos os cenários, a duração da desconexão variava randomicamente entre 3 e 6 segundos, de acordo com uma distribuição uniforme.

Os cenários dos experimentos também variaram levando em consideração os três níveis de confiabilidade oferecidos pelo M-Hub/CDDL. Cabe ressaltar que, nos experimentos utilizando os níveis de confiabilidade 1 (*at-least-once*) e 2 (*exactly-one*), o middleware habilita o mecanismo de *buffer* intermediário, implementado pelo componente `Connection` (ver Seção 5). Considerando que cada experimento foi repetido 5 vezes, obteve-se um total de 60 execuções.

Portanto, justifica-se o conjunto de experimentos apresentados devido ao fato de que ele permite atingir o objetivo da avaliação, uma vez que a eficiência na entrega dos dados de contexto da abordagem de middleware pode ser avaliada em cenários com diferentes tipos de rede e intervalos de desconexão.

7.2.2 Resultados e Discussões

Os resultados são apresentados na tabela 7.4.

É possível observar que o middleware conseguiu entregar 100% das mensagens em todas as situações em que o nível de confiabilidade empregado era maior que zero, permitindo assim concluir que o MHub/CDDL possui um alto grau de eficácia na entrega quando se utiliza tais configurações de confiabilidade. Esse desempenho foi obtido devido à combinação do mecanismo padrão de confirmação de entrega provido pelo MQTT, aliado ao mecanismo de *buffer* intermediário implementado pelo MHub/CDDL.

O mecanismo de *buffer* intermediário aumenta a confiabilidade padrão do MQTT. Conforme explicado na Seção 4.2.1, isso acontece devido ao fato de que, diferentemente do *buffer* padrão, provido pelas bibliotecas clientes MQTT, o *buffer* intermediário do M-Hub/CDDL armazena todas as mensagens publicadas até que seja recebida uma confirmação de entrega, mesmo que a conexão TCP/IP expire.

#	Configuração de Rede	Confiabilidade	Intervalo entre Desconexões	Média Taxa de Perda	Máximo Taxa de Perda	Mínimo Taxa de Perda	Desvio Padrão Taxa de Perda
1	Rede Local Isolada	0	120 a 180 s	3.8%	4.2%	3.6%	0.3%
2	Rede Local Isolada	0	12 a 18 s	23.8%	28.6%	21.1%	2.9%
3	Rede Local Isolada	1	120 a 180 s	0.0%	0.0%	0.0%	0.0%
4	Rede Local Isolada	1	12 a 18 s	0.0%	0.0%	0.0%	0.0%
5	Rede Local Isolada	2	120 a 180 s	0.0%	0.0%	0.0%	0.0%
6	Rede Local Isolada	2	12 a 18 s	0.0%	0.0%	0.0%	0.0%
7	Internet Fixa (Wifi + ADSL)	0	120 a 180 s	3.8%	3.9%	3.5%	0.2%
8	Internet Fixa(Wifi + ADSL)	0	12 a 18 s	35.5%	36.3%	33.9%	1.0%
9	Internet Fixa (Wifi + ADSL)	1	120 a 180 s	0.0%	0.0%	0.0%	0.0%
10	Internet Fixa (Wifi + ADSL)	1	12 a 18 s	0.0%	0.0%	0.0%	0.0%
11	Internet Fixa(Wifi + ADSL)	2	120 a 180 s	0.0%	0.0%	0.0%	0.0
12	Internet Fixa (Wifi + ADSL)	2	12 a 18 s	0.0%	0.0%	0.0%	0.0%

Tabela 7.4: Avaliação do Serviço de Distribuição em relação à Confiabilidade de Entrega

Nesse caso, todas mensagens publicadas usando os níveis 1 e 2 de confiabilidade serão reenviadas tão logo a conexão seja reestabelecida e a sessão do cliente com o *Server Broker* MQTT seja recuperada.

É possível observar também que a taxa média de perda de mensagens, no cenário de alta intermitência utilizando o nível 0 de confiabilidade na Internet (35,51%), é significativamente maior do que aquela apresentada pela rede local isolada no experimento que utilizou a mesma taxa de intermitência e o mesmo nível de confiabilidade (23,79 %). A diferença é cerca de 12%.

Atribui-se essa diferença significativa aos seguintes fatos: (i) a comunicação em rede local é tipicamente mais confiável e menos sujeita a falhas do que em redes de longa distância; (ii) a alta intermitência forçava a aplicação a se reconectar mais vezes ao *Server Broker* MQTT.

Considerando que o tempo observado de conexão com rede local (que variava entre 9 e 10 milisegundos) era cerca de 30 a 50 vezes menor que o tempo observado de conexão com a Internet (que variava entre 300 e 500 milisegundos), conclui-se que a aplicação teste passava mais tempo conectada ao *Server Broker*, nos experimentos usando rede local, do que naqueles usando a Internet, possibilitando assim que o primeiro entregasse um número maior de mensagens que o segundo.

Entretanto, nos casos em que aplicação não precisa se reconectar tantas vezes ao *Server Broker* MQTT, o tempo de conexão não tem impacto significativo no desempenho. Por esse motivo, a taxa de perda para ambos os tipos de configuração de rede, local e Internet, são bem próximas nos cenários de baixa intermitência.

Por fim, o baixo desvio padrão apresentado permite concluir que os resultados tendem se manter estáveis com o uso de aplicações da IoT/IoMT que executem em ambientes de rede com condições semelhantes às simuladas neste experimento.

7.3 Avaliação do Tempo de Monitoramento

O tempo de monitoramento, ou seja, o tempo gasto para processar a informação e detectar se houve algum tipo de variação de contexto ou de QoC, precisa ser ágil. Do contrário, caso ocorram oscilações significativas, estas serão percebidas muito tardiamente pela aplicação consumidora. Consequentemente, isso pode impedir que ela reaja à tais mudanças em tempo hábil. Diante disso, o objetivo específico deste conjunto de experimentos é avaliar a eficiência do mecanismo de monitoramento do MHub/CDDL.

7.3.1 Descrição do Experimento

Conduziu-se um único tipo de experimento, no qual a taxa de chegada das informações de contexto (geradas sinteticamente por um sensor virtual) era de 1 mensagem por milissegundo (1/ms), uma frequência que pode ser considerada relativamente alta e que exige que os eventos sejam processados muito rapidamente. A aplicação de teste utilizada foi a mesma dos experimentos anteriores. Em cada experimento foram publicadas 1800 mensagens. Desse total, 180 amostras possuíam valores de acurácia maiores que zero, enquanto que as demais possuíam acurácias iguais a zero. A regra de monitoramento utilizada neste experimento consistiu em detectar a chegada de amostras cujas acurácias fossem maiores que zero, sendo que, a cada 10 milissegundos, uma amostra com acurácia diferenciada era publicada simulando uma variação na QoC. A regra de monitoramento utilizada no experimento, escrita em *Event Processing Language* (EPL), é a seguinte:

Listing 7.1: Regra EPL usada no experimento de avaliação do tempo de monitoramento.

```
Select * from SensorDataMessage where accuracy > 0.0
```

O experimento foi repetido 5 vezes, utilizando-se sempre o *Micro Broker* e o nível 0 de confiabilidade. O *smartphone* no qual a aplicação teste foi executada era o mesmo dos experimentos anteriores. O tempo de monitoramento considerado equivale ao intervalo entre a chegada da mensagem no subscritor e o envio da notificação da variação de QoC para a aplicação consumidora. Sendo assim, a configuração de rede e nível de confiabilidade não interferem nos resultados obtidos.

Portanto, justifica-se o conjunto de experimentos apresentados devido ao fato de que ele permite atingir o objetivo da avaliação, uma vez que o tempo de de monitoramento da abordagem de middleware pode ser avaliada em um cenário cuja taxa de chegada das informações de contexto pode ser considerada alta. Desse modo, é possível observar se a solução proposta oferece suporte adequado para aplicações que precisam se adaptar rapidamente à mudanças no ambiente de execução.

7.3.2 Resultados e Discussões

	Média (ms)	**Máximo (ms)**	**Mínimo (ms)**	**Desvio Padrão**
Tempo de Monitoramento	12,49	14,53	10,15	1,56

Tabela 7.5: Avaliação do Tempo de Monitoramento

Os tempos de monitoramento médio, máximo e mínimo obtidos nos experimentos foram 12,49, 14,53 e 10,15 milissegundos, conforme mostra a Tabela 7.5. O desvio padrão foi de 1,56 milissegundos. Esses resultados permitem mostrar que o tempo de monitoramento do M-Hub/CDDL é bastante baixo, mesmo em condições onde a taxa de chegada dos eventos é muito alta. Pode-se concluir que o serviço de monitoramento do M-Hub/CDDL é eficiente, pois permite que as aplicações possam reagir rapidamente às variações de contexto e de QoC. O baixo desvio padrão do experimento indica que a tendência é que esse desempenho se mantenha estável em condições semelhantes.

7.4 Avaliação do Consumo de Memória

Avaliação e monitoramento de QoC são operações que exigem processamento contínuo sobre o fluxo de dados. Portanto, essas operações estão entre aquelas que tendem a consumir mais memória. Em dispositivos móveis, é necessário que esse consumo não seja elevado, visto que a memória nos dispositivos móveis, assim como outros recursos computacionais, é geralmente menos abundante do que em computadores pessoais e servidores.

O objetivo deste experimento é medir a quantidade de memória alocada para os componentes `QoC Evaluator` e `Monitor`, responsáveis pela avaliação e monitoramento de QoC no M-Hub/CDDL, respectivamente. Conforme explicado na Seção 5, a implementação desses componentes é baseada no Processamento de Eventos Complexos (*Complex Event Processing* - CEP). Logo, a medição do consumo de memória por parte desses componentes permite avaliar se é viável que aplicações de IoT, principalmente aquelas que executam em dispositivos móveis com recursos limitados, façam uso dos mecanismos de monitoramento e avaliação de QoC providos pelo M-Hub/CDDL.

7.4.1 Descrição do Experimento

Foi utilizada a mesma aplicação teste empregada no experimento anterior, variando-se apenas a taxa de publicação, que neste experimento foi de 1 mensagem por segundo. Essa frequência pode ser considerada moderada e compatível com aquela praticada por muitos sensores, tais como o *Zephyr Bioharness* 3 e o *Zephyr HxM*, utilizados nos testes do M-Hub/CDDL. A aplicação foi executada durante 30 minutos. A partir do uso da ferramenta Android Monitor (embutida na IDE Android Studio[2]), foram coletados dados relativos ao consumo de memória em 5 instantes de execução da aplicação: 5, 10, 15, 20 e 25 minutos. O experimento foi repetido 5 vezes.

Portanto, justifica-se o experimento apresentado devido ao fato de que ele permite atingir o objetivo da avaliação, uma vez que o consumo de memória por parte componentes responsáveis pelo processamento contínuo eventos pode ser avaliado

[2] `http://tools.android.com/`

levando em consideração uma frequência de medição comum à diversos tipos de sensores que podem ser utilizados por aplicações de IoT.

7.4.2 Resultados e Discussões

A Tabela 7.6 exibe os resultados obtidos.

	5 min	**10 min**	**15 min**	**20 min**	**25 min**	**Média**	**Desvio Padrão**
QoC Evaluator	263	262	268	269	270	266	2.77
Monitor	213	213	216	219	280	214	3.14

Tabela 7.6: Avaliação de Consumo de Memória do M-Hub/CDDL em Kbytes

As médias de consumo de memória dos componentes `QoC Evaluator` e `Monitor` são 266 e 214 Kbytes, respectivamente. A partir desses resultados, é possível observar que os componentes de avaliação e monitoramento de QoC requerem uma pequena quantidade de memória para processar fluxos de dados com uma frequência moderada. A quantidade de memória alocada para os componentes da aplicação permanece estável durante todo o período de execução. Diante disso, pode-se concluir que é viável utilizar os mecanismos de avaliação e monitoramento de QoC providos pelo M-Hub/CDDL em dispositivos móveis com recursos limitados, em condições semelhantes às dos experimentos realizados.

7.5 Avaliação do Consumo de Bateria

Para aplicações de IoT móveis que precisam executar continuamente ao longo dia, tais como aplicações de monitoramento de pacientes, é importante que consumo de bateria não seja elevado. Do contrário, a aplicação, além de comprometer a autonomia energética do dispositivo, só poderá ser utilizada por pouco tempo.

O objetivo deste experimento é avaliar o consumo energético do M-Hub/CDDL em dispositivos móveis, a fim de analisar a viabilidade de uso da solução proposta em dispositivos com limitação de bateria.

7.5.1 Descrição do Experimento

Foi utilizada a mesma aplicação teste do experimento anterior e igual taxa de publicação de dados. A aplicação foi executada durante 12 horas, sendo que o nível da bateria foi medido antes e depois do tempo de execução da mesma. É importante destacar que durante a execução da aplicação a tela do dispositivo permaneceu desligada. Além disso, com a exceção dos processos do sistema operacional, não haviam outros processos executando no dispositivo.

Portanto, a justificativa para a adoção desse cenário de experimento, assim como na avaliação do consumo de memória, consiste no fato de que ele permite permite que componentes responsáveis pelo processamento de eventos complexos tenham o seu consumo de bateria mensurado, levando em consideração uma aplicação que recebe dados de contexto com uma frequência de medição comum a diversos tipos de sensores, durante um período de execução da aplicação que pode ser considerado relativamente longo.

7.5.2 Resultados e Discussões

	Mínimo	**Máximo**	**Média**	**Desvio Padrão**
Consumo de Bateria	15%	16%	15.6%	0.55

Tabela 7.7: Avaliação de Consumo de Bateria do M-Hub/CDDL

O consumo de bateria medido foi de 15.6 % após as 12 horas. Considera-se que o consumo medido é baixo para uma aplicação de IoT que publica e monitora dados com uma frequência de 1 mensagem por segundo. Esse resultado mostra que o M-Hub/CDDL pode ser utilizado por aplicações de IoT/IoTM de longa duração, pois não compromete a autonomia enérgica de dispositivos móveis de uso convencional.

7.6 Conclusão do Capítulo

Este capítulo apresentou uma avaliação quantitativa que abordou alguns dos principais aspectos da abordagem de middleware proposta: (i) avaliação do tempo

total de entrega dos dados ao consumidor; (ii) avaliação da confiabilidade de entrega em redes instáveis; (iii) avaliação do tempo de monitoramento; (iv) avaliação do consumo de memória e (v) avaliação do consumo de bateria.

Em relação ao tempo total de entrega dos dados ao consumidor, a abordagem de middleware proposta foi avaliada em cenários com diferentes configurações de rede e níveis de confiabilidade. Os resultados obtidos mostraram que o tempo de processamento da informação no middleware é muito baixo e estável. Sendo assim, o tempo de comunicação, que pode variar muito de acordo com as condições da rede, tem um impacto significativamente maior sobre o tempo total de entrega dos dados que o tempo de processamento no middleware. Contudo, ainda que mais susceptível às interferências do meio de comunicação, o tempo de comunicação obtido nos experimentos realizados, tanto no âmbito local como usando a Internet, também foram relativamente baixos. Atribui-se o bom desempenho na comunicação não apenas às condições das redes utilizada nos experimentos, mas sobretudo devido ao fato do serviço de distribuição do M-Hub/CDDL levar em consideração uso de um protocolo projetado para redes com baixa largura de banda e alta latência.

Em relação à confiabilidade de entrega, a abordagem de middleware proposta foi avaliada em cenários de alta e baixa intermitência de conexões, variando também as configurações de rede e os níveis de confiabilidade. Os resultados mostraram que, graças à combinação entre confiabilidade nativa do protocolo MQTT e o mecanismo de *buffer* intermitente implementado como parte do serviço de conexão do M-Hub/CDDL, a abordagem de middleware proposta é capaz de garantir a entrega de mensagens mesmo em situações onde a conexão com a rede expira.

Em relação ao monitoramento, que se aplica tanto às mudanças de contexto como às variações de QoC, a abordagem de middleware foi avaliada em um cenário no qual a taxa de chegada dos eventos é extremamente elevada. Os resultados obtidos mostraram que o mecanismo de monitoramento do M-Hub/CDDL é capaz de detectar variações de contexto e QoC em poucos milissegundos, possibilitando que as aplicações de IoT possam reagir em tempo hábil a essas oscilações.

Em relação ao consumo de recursos por parte do M-Hub/CDDL, avaliou-se tanto o consumo de memória como de bateria. A avaliação do consumo de memória, que levou em consideração uma aplicação que publicava dados de contexto com

uma frequência moderada, mostrou que os componentes responsáveis por operações de processamento contínuo do fluxos de eventos, que tendem a ser aqueles que consomem mais recursos, apresentaram consumo de memória baixo e estável ao longo do tempo de execução da aplicação teste. A avaliação de consumo de bateria mostrou que o consumo energético por parte do M-Hub/CDDL também é baixo, mesmo após muitas horas de execução interrupta dos mecanismos de avaliação e monitoramento. Esses resultados mostram a eficiência do middleware no consumo dos recursos computacionais, permitindo concluir que o M-Hub/CDDL é uma solução viável para aplicações de IoT voltadas para dispositivos móveis com recursos limitados.

De modo gral, os resultados e as discussões apresentadas neste capítulo de avaliação quantitativa mostram que a abordagem de middleware proposta apresenta desempenho muito satisfatório em relação às métricas estabelecidas, dentro das condições avaliadas. Considerando-se que o desempenho é um fator muito importante na escolha do middleware a ser utilizado para desenvolver e executar aplicações de IoT, a avaliação quantitativa realizada demonstra o potencial de utilização do M-Hub/CDDL em cenários onde as aplicações exigem baixo tempo de entrega dos dados, alta confiabilidade em redes instáveis e baixo consumo de recursos. Diante do exposto, esta tese cumpre não apenas o objetivo de desenvolver uma abordagem de middleware de contexto eficiente, mas também o objetivo de avaliá-la quantitativamente.

Cabe ressaltar que, de modo geral, os trabalhos relacionados a middleware de contexto com suporte a QoC encontrados na literatura (ver Capítulo 8) apresentam pouca ou nenhuma avaliação quantitativa, restringindo-se na maioria dos casos a mostrar avaliações qualitativas baseadas em provas de conceito. Sendo assim, a avaliação quantitativa do M-Hub/CDDL é considerada como uma contribuição da tese. Naturalmente, desejava-se avaliar também o desempenho da abordagem proposta em relação à diversos outros aspectos. Entretanto, devido à complexidade de realizar uma quantidade maior de experimentos dentro do tempo de desenvolvimento da tese, isso não foi possível. Um novo conjunto de avaliações quantitativas, incluindo, por exemplo, experimentos de escalabilidade, será o objetivo de trabalhos futuros.

8 Trabalhos Relacionados

Li et al [63], Yürür et al. [112], Perera et al. [86] e Bandyopadhyay et al. [8], por exemplo, produziram importantes *surveys* nos quais um grande quantitativo de propostas de middleware de contexto para IoT foi analisado. Embora relevantes, esses trabalhos não têm o objetivo de analisar a maneira como cada abordagem de middleware apoia o desenvolvimento de aplicações cientes de contexto que possuem requisitos de QoC. O fato é que apenas uma minoria das abordagens de middleware conhecidas na literatura, tais como AWARENESS [102], COSMOS [1, 24], COPAL [62], INCOME [6,66,74,75] e SALES [26–28,40], apresenta algum tipo de suporte a qualidade de contexto. Este capítulo apresenta inicialmente uma síntese dos principais trabalhos relacionados a middleware de contexto com suporte a QoC. Em seguida apresenta-se uma análise comparativa entre essas abordagens e a que está sendo proposta neste trabalho. Ao final deste capítulo apresenta-se as conclusões.

8.1 AWARENESS

AWARENESS [102] é uma infraestrutura para o desenvolvimento de aplicações móveis cientes de contexto adaptativas. O middleware oferece suporte a mecanismos de descoberta de serviços, distribuição, inferência, adaptação, armazenamento e privacidade. A arquitetura geral do AWARENESS é ilustrada por meio da Figura 8.1. O suporte ao desenvolvimento de aplicações do middleware provê um conjunto de APIs e componentes em duas camadas: a camada de aplicação e a camada de infraestrutura. Na camada de aplicação estão as aplicações móveis cientes de contexto e pró-ativas, divididas em componentes clientes e servidores, que podem interagir entre si e com a infraestrutura. Esta camada também compreende sensores específicos dessas aplicações, que fornecem dados localmente para elas. A camada de infraestrutura dá suporte à execução das aplicações. Ela concentra funções de gerenciamento de contexto distribuído, inferência, registro, descoberta de serviços e persistência. Essa camada compreende também sensores específicos da infraestrutura. Esses sensores fornecem dados que serão distribuídos para as aplicações clientes. A

heterogeneidade do hardware e de protocolos de comunicação é encapsulada por *wrappers*. Entretanto, o middleware não padroniza nenhuma tecnologia ou protocolo de comunicação com os sensores e/ou com a rede. Sendo assim, o programador precisa implementar todos os detalhes da comunicação.

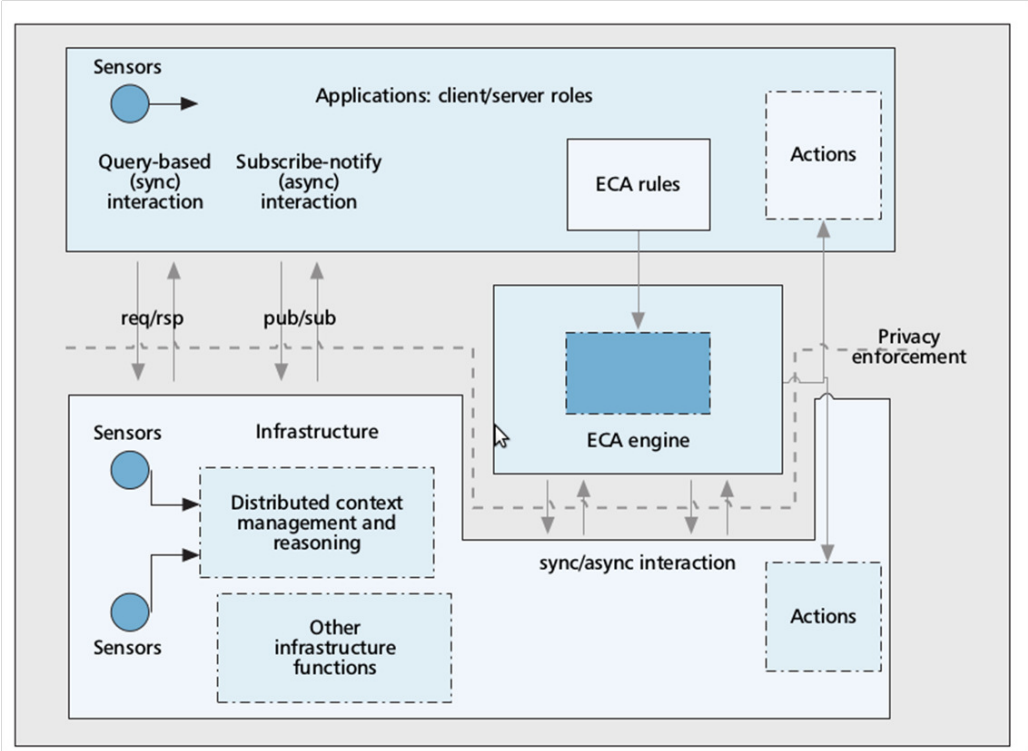

Figura 8.1: Arquitetura Geral do AWARENESS [102]

A arquitetura do *Context Managment Service* (CMS) do AWARENESS é ilustrada na Figura 8.2. A seguir descreve-se seus principais componentes e serviços.

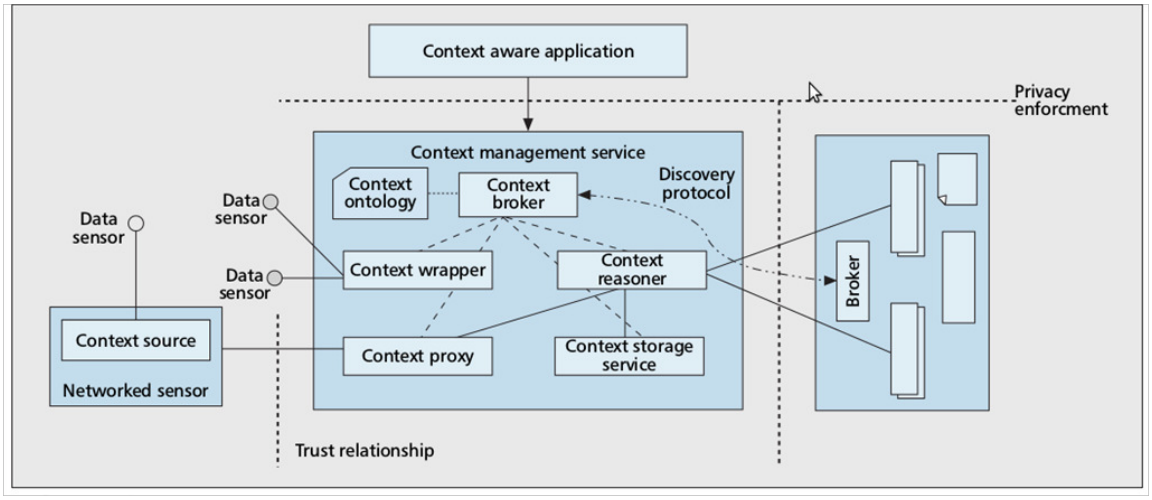

Figura 8.2: Componentes do CMS do AWARENESS [102]

O CMS do AWARENESS provê duas interfaces principais: a *Context Broker* e a *Context Source*. A interface *Context Broker* fornece os métodos para registro e a descoberta dos tipos e fontes de contexto. O tipo de informação que pode ser registrado consta em uma ontologia particular. O Serviço de Descoberta de Serviços (*Service Discovery Services*) do AWARENESS é baseado em consultas, sendo que existem dois tipo de consultas que as aplicações podem realizar: a instantânea (ou ativa) e subscrita (ou passiva). Na consulta instantânea, a aplicação requisita e recebe imediatamente uma resposta contendo uma lista com serviços disponíveis. Na consulta subscrita, a aplicação publica a requisição e se subscreve para receber a resposta posteriormente, durante um período de tempo especificado. A resposta será enviada assim que for encontrada a primeira ou melhor correspondência positiva entre as informações exigidas e as disponíveis (*matching*). A interface *Context Source* fornece métodos para recuperação de contexto, que pode ser assíncrona (baseada na subscrição de eventos) ou síncrona (baseada em consulta ativa). No modo assíncrono, o cliente se subscreve e aguarda o recebimento de notificações de mudança no valor de uma determinada informação de contexto. No modo síncrono, o cliente realiza uma consulta de recuperação de contexto, que retorna o valor atual da informação requerida naquele instante. Essas consultas podem ser especificadas em SQL *Structured Query Language* (SQL) e *RDF Data Query Language* (RDQL) [19].

Em relação à QoS[1], AWARENESS provê mecanismos de persistência e histórico, que são providos pelo *Context Storage Service*. Esse serviço provê uma interface de acesso direto ao banco de dados e uma interface de raciocínio baseada em Jena [2]. Em relação à QoI[3], o middleware está focado no provisionamento dos seguintes parâmetros: *Accuracy*, *Probability of Correctness*, *Trustworthiness* e *Up-to-dateness*. Cabe às fontes de contexto utilizadas pelo AWARENESS a tarefa de especificar o valor desses parâmetros, ou seja, não existe um componente de avaliação de QoC capaz de computá-los automaticamente. Posteriormente, o trabalho de Sheikh et al. [99] propõe diferentes alternativas de como cada um desses parâmetros podem ser computados.

AWARENESS permite detectar eventos e inferir situações de interesse da aplicação com base na especificação de regras do tipo *Evento-Condição-Ação* (ECA),

[1]Os autores não usam os termos QoS. Entretanto, para fins de comparação, considerar-se-à que o mecanismo de histórico implementado pelo AWARENESS é de fato um parâmetro de QoS (*History*).

[2]`https://jena.apache.org/tutorials/rdf_api.html`

[3]Os autores consideram que o conceito de QoC é restrito à qualidade da informação de contexto

que são executadas por uma *Engine ECA*. Quando os eventos/situações de interesse do consumidor ocorrem, a infraestrutura dispara uma ação que será executada pela aplicação em resposta às mudanças de contexto. Um serviço denominado *ECA Controlling Service* (ECA-CS) é responsável por invocar o *Service Discovery Services* a fim de descobrir as fontes de contexto e serviços correspondentes às ações especificadas.

AWARENESS provê ainda um mecanismo de privacidade das informações de contexto. Esse mecanismo combina duas abordagens: a abordagem de pré consentimento e a abordagem de interação direta com o usuário. A abordagem de pré-consentimento é aquela baseada em políticas de privacidade, ou seja, o usuário especifica detalhadamente na política o tipo de informação de contexto que deseja compartilhar, com quem, com que nível de QoI e em quais condições. Na abordagem de interação direta com o usuário, o proprietário da informação recebe uma solicitação de acesso. A informação só será liberada caso ele autorize. O mecanismo de privacidade do AWARENESS só usa a abordagem direta nos casos em que nenhuma política de privacidade foi especificada, ou nos casos em que a própria política de privacidade exige o consentimento do proprietário da informação.

8.2 COSMOS

COntext entitieS coMpositiOn and Sharing[4] (COSMOS) [24] é definido como um framework para gerenciamento de contexto em ambientes ubíquos, implementado em Java, que permite o desenvolvimento de aplicações adaptativas. O middleware utiliza um modelo de programação baseado em componentes denominado Fractal [14]. Esse modelo de programação se destina ao projeto, implementação e gerenciamente sistemas de software complexos (isto é, monitorar, controlar e configurar dinamicamente), inclusive em sistemas operacionais e middleware específicos. Os componentes podem ser compostos (componentes que contêm subcomponentes) ou compartilhados (subcomponentes de múltiplos componentes compostos). Os componentes são descritos utilizando uma Linguagem Específica de Domínio (*Domain Specific Language* - DSL) [44], denominada Fractal ADL [59]. A comunicação entre os componentes do middleware é orientada a mensagens (MOM) e

[4] `http://picolibre.int-evry.fr/projects/cosmos`.

foi implementada usando a DREAM *framework*[5] [60], uma biblioteca que dá suporte à comunicação usando TCP ou UDP.

A estrutura básica utilizada para a composição de componentes do COSMOS é denominada *Context Node*, cuja arquitetura está ilustrada na Figura 8.3. Os nodos de contexto são organizados dentro de uma hierarquia na qual podem ser ativos ou passivos. Um nodo ativo é aquele que está associado a uma *thread*, podendo iniciar o processo de recuperação de informações por conta própria. Um nodo passivo é aquele que só obtém informações de contexto quando invocado por outro nodo. A disseminação de informações de contexto dentro da hierarquia pode acontecer em dois sentidos: de baixo para cima (notificação) ou de cima pra baixo (observação). *Pull* e *Push* são as interfaces usadas para observação e notificação, respectivamente .

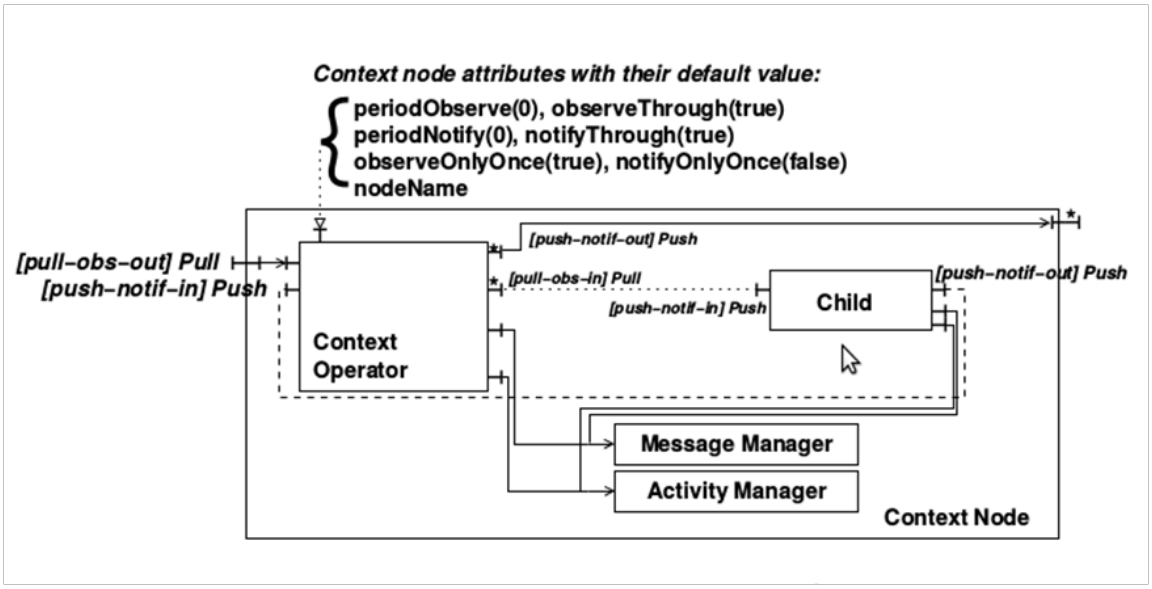

Figura 8.3: Arquitetura do *Context Node* do COSMOS [1]

A arquitetura do COSMOS é conceitualmente dividida em três camadas: inferior, intermediária e superior. A camada inferior do COSMOS define a noção de *Context Collector*, que são nodos de contexto responsáveis por coletar dados de contexto de baixo nível. Portanto, o coletor de contexto age como um *wrapper*, escondendo das demais camadas a heterogeneidade de tecnologias e protocolos de comunicação com as fontes de contexto. COSMOS não oferece um serviço nativo a qualquer tecnologia de comunicação que possibilite a aquisição de dados a partir de sensores, transferindo para o programador essa responsabilidade. Entretanto, o middleware fornece suporte à aquisição de dados sobre recursos físicos gerenciados pelo sistema operacional (e.g.,

[5] `http://dream.ow2.org/`

bateria, processador, memória, interface de rede) ou lógicos (e.g., *sockets*, *threads*). Esse suporte foi implementado com uso do framework SAJE [29].

A camada intermediária define a noção de processador de contexto. O *Context Processor* é um nodo de contexto responsável por filtrar e agregar dados de baixo nível provenientes dos coletores de contexto, a fim de obter informações de contexto de alto nível. Na camada superior estão os nodos de contexto responsáveis por inferir situações a partir dos dados coletados e processados nas camadas inferiores. As situações inferidas poder ser usadas para desencadear algum tipo de adaptação da aplicação. Os autores consideram que as técnicas de processamento e inferência a serem utilizadas dependem do domínio de cada aplicação. Por isso, o middleware não oferece suporte a uma técnica específica de processamento, como CEP ou ECA *rules*, por exemplo. Portanto, compete ao programador da aplicação a tarefa de definir isso.

COSMOS não foi inicialmente projetado para dar suporte à qualidade de contexto, sendo que a DSL utilizada pelo middleware não permite a especificação de QoC. Posteriormente, um trabalho em progresso de Abid et al. [1] propõe a extensão da arquitetura do COSMOS, adicionando-se a ela um novo tipo de nodo de contexto, denominado *QoC Context Node*, componente responsável por processar os dados de contexto coletados, a fim de extrair, avaliar e filtrar a qualidade das informações. O *QoC Context Node*, cuja arquitetura é ilustrada na Figura 8.4, é um nodo de contexto composto, que possui diversos coletores de contexto e outros subcomponentes que processam a QoC, dentre eles o *QoC Operator*, responsável por gerenciar o cômputo dos parâmetros de qualidade da informação.

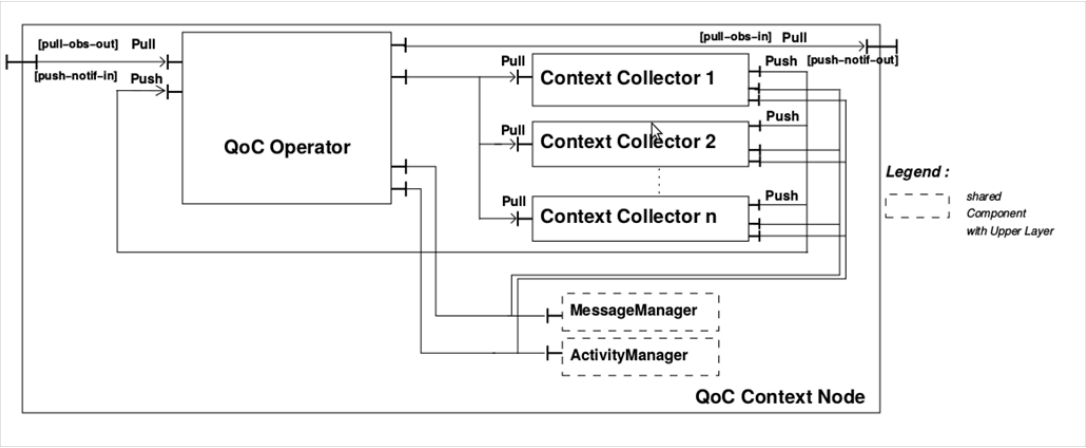

Figura 8.4: Arquitetura do *QoC Context Node* do COSMOS [1]

O *QoC Operator*, cuja arquitetura é ilustrada na Figura 8.5, também é um nodo de composto. Dentro desse componente, cada parâmetro de QoC é computado separadamente pelo seu respectivo *QoC Parameter Operator*. Após serem calculados, os parâmetros de QoC são enviados para o subcomponente *QoC Aware Operator*, que é responsável por transmitir as informações de QoC para as camadas superiores. Esse componente divulga informações sobre a QoC de dois modos: *QoC embutida na informação* e *QoC separada da informação*. No modo *QoC embutida na informação*, todas as amostras da informação de contexto são enriquecidas com QoC. Esse modo é útil para aplicações interessadas tanto na informação de contexto em si quanto na QoC associada a ela. Este modo permite a filtragem das informações com base na QoC, apesar impor maior custo computacional. No modo *QoC separada da informação*, a QoC é enviada em mensagens separadas, periodicamente (*pull*) ou mediante uma requisição da aplicação (*push*). A informação sobre a qualidade de contexto enviada pode corresponder, por exemplo, à última QoC computada ou a uma média das últimas leituras. O envio da QoC separadamente, além de consumir menos recursos, garante a compatibilidade da aplicação com versões do COSMOS sem suporte à QoC.

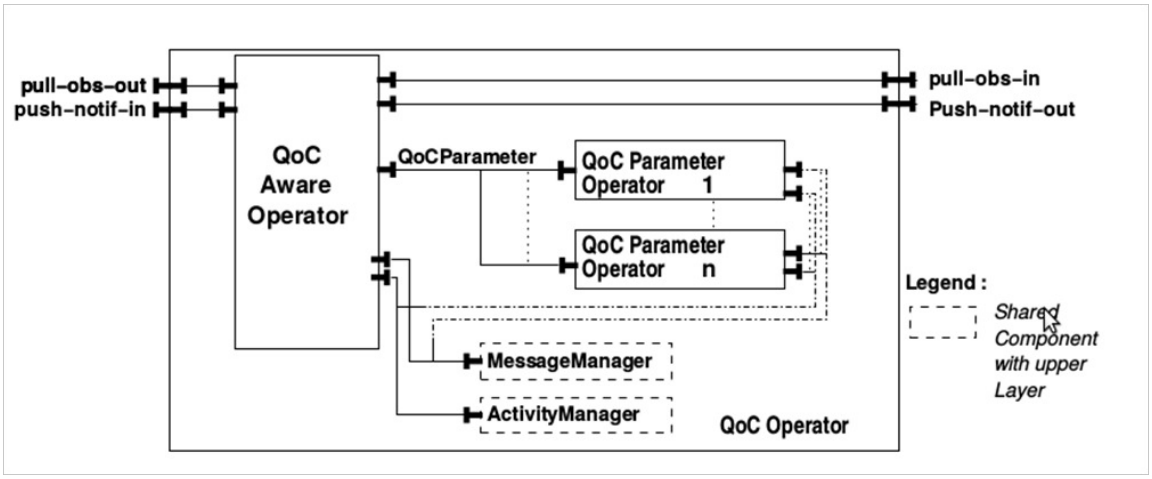

Figura 8.5: Arquitetura do *QoC Operator* do COSMOS [1]

Portanto, a ideia do gerenciamento de QoC no COSMOS é que cada desenvolvedor defina seus próprios parâmetros de QoI e respectivas funções de avaliação, a partir do uso de operadores lógicos disponibilizados pelo middleware. Os autores promoveram um estudo de caso envolvendo o desenvolvimento de uma aplicação que utiliza o middleware para definir e computar três parâmetros de QoI: *UpToDateness*, *Trustworthiness*, *Dateness* e *Precision*. Os trabalhos do COSMOS não mencionam a existência qualquer suporte a parâmetros de QoS.

8.3 COPAL

COntext Provisioning for ALl [6] (COPAL) [62] é um middleware de contexto implementado em Java voltado ao desenvolvimento de aplicações adaptativas. Para isso, COPAL fornece suporte à especificação e processamento (e.g., filtragem, agregação e sumarização, diferenciação) de eventos complexos, que refletem as mudanças de contexto no ambiente de execução da aplicação. Ações específicas definidas pelas aplicações podem ser executas pelo middleware diante da ocorrência de determinados eventos. COPAL está inserido no contexto de desenvolvimento do projeto *Smart Home for All* (SM4ALL). O suporte ao desenvolvimento de aplicações do COPAL provê uma linguagem para a descrição de componentes chamada de COPAL-DSL. Posteriormente em [98] foi apresentada a linguagem COPAL ML (COPAL *Macro Language*), que estende a linguagem Java com algumas palavras-chave. Segundo os autores, a COPAL-ML esconde a complexidade e reduz o código necessário para o desenvolvimento de aplicações. A intenção é usar a COPAL-ML para escrever a lógica dos componentes, enquanto que a COPAL-DSL descreveria a sua implementação. Opcionalmente, pode-se programar os componentes usando a COPAL API. A arquitetura do COPAL, ilustrada na Figura 8.6, é dividida em três camadas: serviços de dispositivos, núcleo COPAL e serviços cientes de contexto.

A camada **Serviços de Dispositivos** descreve os dispositivos e sensores com os quais o COPAL interage. Para diminuir a discrepância entre o modelo de dados fornecido por um sensor e o modelo de dados utilizado pelo COPAL, é necessário um componente que faça uma ponte entre estes dois ambientes. Este componente é chamado de *Publisher*, cuja tarefa é acessar o sensor, recuperar informações e traduzir a informação recebida para um evento conhecido pelo COPAL. Portanto, cada *Publisher* é um *wrapper* que implementam as tecnologias de comunicação e protocolos de acesso específicos de cada sensor. As descrições de serviço são feitas conforme o padrão UPnP [7](*Universal Plug and Play*). Os serviços de cada sensor são expostos para as camadas superiores na forma de *Web Services*. O serviço *Device Manager* armazena informações de hardware, mantém um catálogo de dispositivos e monitora o status de cada um.

[6] `http://www.infosys.tuwien.ac.at/m2projects/sm4all/copal/downloads.html`
[7] `http://www.upnp.org/`

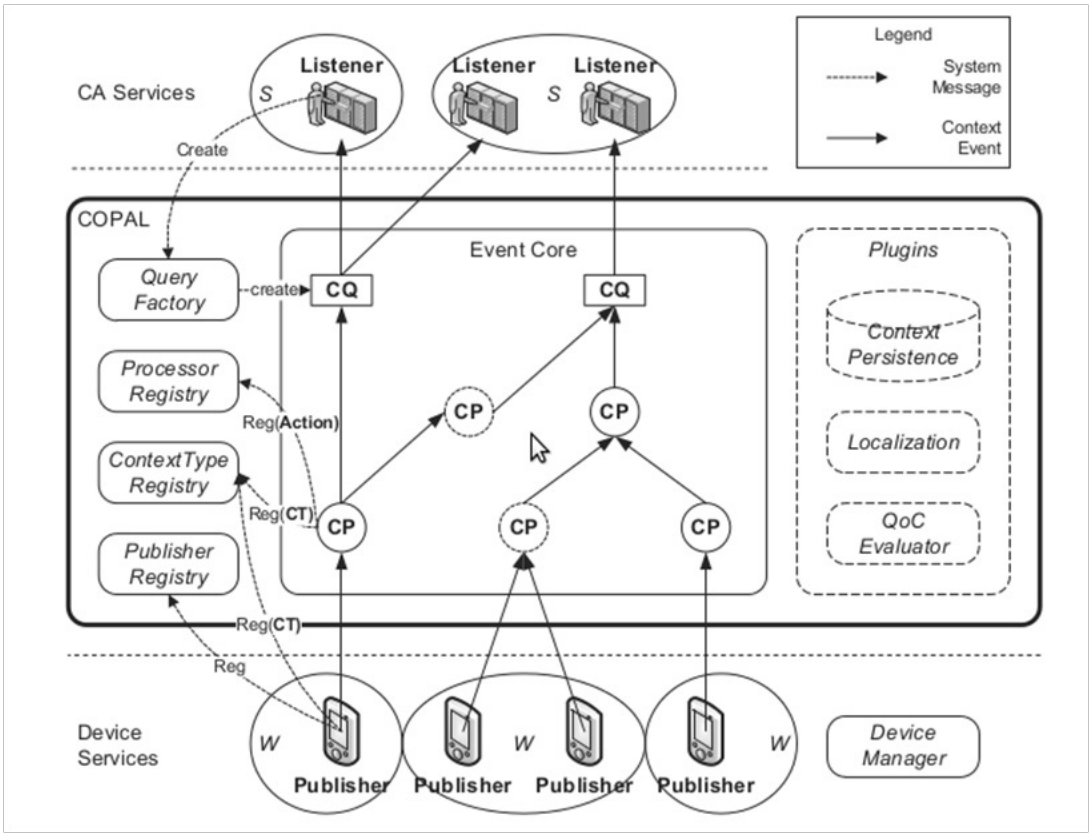

Figura 8.6: Arquitetura do COPAL [62]

A camada **Núcleo COPAL** registra componentes, realiza processamento de eventos e executa ações. Os serviços *Publisher Registry* e *Context Type Registry* são responsáveis pelo registro dos publicadores e tipos de informação de contexto (*Context Type*), respectivamente. O *Context Processor* tem a função de processar os tipos de contexto. Esse processamento pode resultar na execução de uma ação a qual ele está vinculado. Cada *Context Processor* deve ser registrado junto ao *Context Process Registry*.

A camada **Serviços Cientes de Contexto** contém *listeners* que as aplicações usam para que sejam notificadas quando suas consultas de recuperação de contexto são respondidas. Essas consultas especificam o tipo de informação de contexto e um conjunto de critérios que permitem filtrar eventos com base nos atributos das informações. Cada *Listener* está associado a uma consulta, que pode ser reutilizada por vários *listeners*. O componente de geração de consulta é o *Query Factor*. O compartilhamento de dados de contexto e a interação entre os componentes *Publisher*, *Listener* e *Context Processor* são providos pelo *Event Core Service*, o serviço responsável pelo processamento de eventos complexos (CEP), que utiliza para isso a *Engine Esper* [8].

[8] `http://esper.codehaus.org`

Eventos no COPAL são documentos XML, que estão em conformidade com os tipos de contexto definidos no esquema. As consultas de recuperação de contexto são transformadas em *Event Processing Language* (EPL), de modo que possam ser executas pela *Engine Esper* [15]. Os eventos possuem os quatro atributos essenciais *Source ID*, que identifica o evento de forma única; *Timestamp*, que indica o instante de ocorrência do evento; *Priority*, que determina a prioridade de processamento do evento; *Time To Live (TTL)*, que indica por quanto tempo o evento permanece válido, sendo que eventos cujo TTL expirou são automaticamente descartados. Eventos COPAL podem ter ainda mais seis atributos complementares, cujo suporte é fornecido por componentes opcionais denominados *Plugins*. O atributo *Source Location*, que indica a localização da fonte, é fornecido pelo *plugin Localization*. Os parâmetros de QoC [9] *Freshness*, *Trust-worthiness* e *Precision* são calculados pelo *plugin QoC Evaluator*. O *plugin Context Persistence* provê mecanismos de consulta sobre dados armazenados em histórico. COPAL provê ainda um mecanismo de autorização que utiliza o atributo *Autorization* para descrever quais serviços podem ter acesso a um determinado evento. A distribuição de eventos no COPAL é apenas local. Entretanto, segundo os autores, o COPAL será estendido para uma arquitetura distribuída, em trabalhos futuros.

8.4 INCOME

INCOME [6, 66, 74, 75] é um middleware de contexto distribuído cujo roteamento é baseado em conteúdo e na QoC[10]. A distribuição de dado de contexto é feita com base na publicação e subscrição de eventos, sendo que as aplicações consumidoras filtram os eventos de seu interesse com base em funções escritas em *JavaScript*[11] e *XML Path Language (XPath)*[12].

[9] Na visão dos autores do COPAL, a noção de QoC é aquela se restringe à qualidade da informação. Para eles, apenas os atributos *Freshness*, *Trust-worthiness* e *Precision* são parâmetros de QoC. Entretanto, para fins de comparação, nesta tese, considerar-se-à que os atributos *Timestamp*, *Source Location* são parâmetros de QoI, enquanto que *Time To Live (TTL)*, *Priority* e *Autorization* e *History* serão classificados aqui como parâmetros de QoS, ainda que os autores não empreguem o termo qualidade de serviço.

[10] Na visão dos autores do INCOME, a QoC também se refere apenas à qualidade da informação.

[11] `https://www.javascript.com/`

[12] `https://www.w3.org/TR/xpath/`

A arquitetura do middleware é ilustrada na Figura 8.7, sendo que o INCOME provê três componentes principais: *Context Collector*, *Context Capsule* e *Broker*.

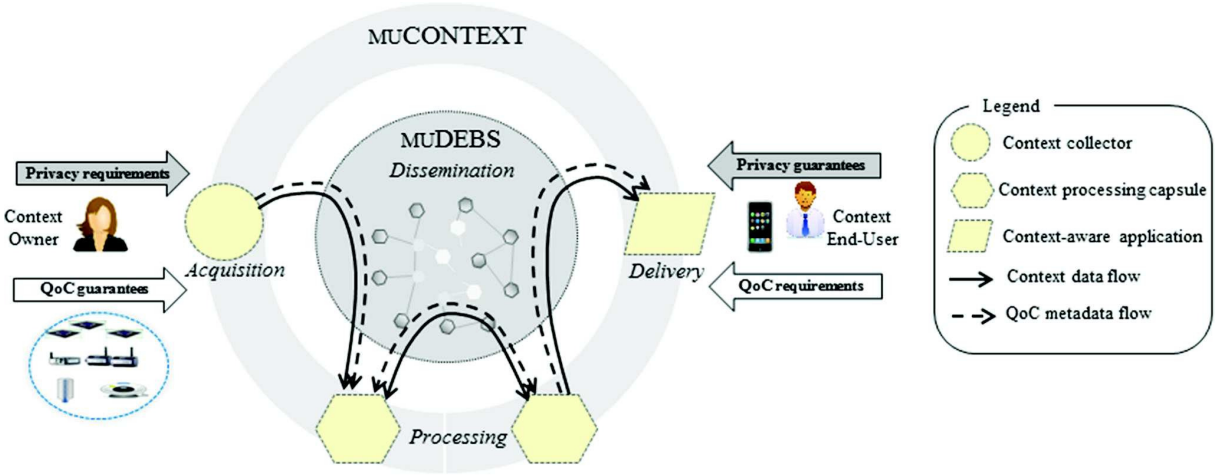

Figura 8.7: Arquitetura do INCOME [66]

O componente *Context Collector* é responsável pela aquisição de dados de baixo nível, ou seja, que ainda não foram processados ou transformados. O INCOME fornece uma API que permite declarar o tipo de informação de contexto e os tipos de metadados de QoC que os coletores de contexto publicam.

O componente *Context Capsule* é responsável por processar dados de baixo nível recebidos dos coletores de contexto e transformá-los em informações de contexto de alto nível, que podem ser disponibilizadas para outras cápsulas de contexto ou aplicações consumidoras. O *Context Capsule* pode ser tanto um produtor quanto um consumidor e utiliza uma API para especificar suas garantias ou requisitos de QoC respectivamente. Por exemplo, em estudos de caso, o componente *Context Capsule* é utilizado para implementar funções que avaliam a QoC como *alta*, *média* ou *baixa*.

Os consumidores de contexto também usam uma API para especificar seus requisitos de contexto e de QoC. As informações de contexto são roteadas até os consumidores apropriados levando em consideração as garantias do produtor e os requisitos do consumidor. A comunicação entre eles é intermediada por *brokers*, organizados em uma rede de sobreposição. O mecanismo de distribuição do INCOME, baseado em eventos, foi implementado com o framework *MuDEBS* [65]. INCOME considera que os *brokers* são entidades confiáveis e que os eventos enviados são autênticos. Por isso, o middleware não se preocupa com mecanismos de criptografia e autenticação, centrando esforços em mecanismos de autorização, a fim de controlar

o acesso às informações de contexto. A solução adotada, denominada *Attribute-Based Access Control* (ABAC) [66], é um mecanismo de autorização baseada em políticas, que protege a privacidade dos dados. Uma política descreve quais operações podem ser realizadas sobre um objeto ou recurso. Pessoas ou serviços solicitam autorização para executar algumas operações e fornecem atributos associados ao objeto e operações solicitadas, inclusive atributos de QoC. Se os atributos fornecidos satisfazem a política especificada, então o acesso é concedido, caso contrário o acesso é negado.

A ideia do gerenciamento de QoC no INCOME é que os desenvolvedores possam definir seus próprio parâmetros de QoI e funções de avaliação, de acordo com os requisitos das aplicações, seguindo uma abordagem dirigida por modelos[13], utilizando o framework *Quality of Context Information Model* (QoCIM) [75]. Os autores mostram estudos de caso nos quais o middleware foi utilizado para implementar e avaliar cinco parâmetros de QoI: *Freshness*, *Precision*, *Completeness*, *Accuracy* e *Spatial Resolution*. Nos trabalhos não é mencionado qualquer suporte a parâmetros de QoS.

8.5 SALES

Scalable context-Aware context Aware middleware for mobiLe EnviromentS (SALES) [26–28, 40] é uma infraestrutura de software voltada para a distribuição de dados de contexto (*Context Data Distribution Infrastructure* - CDDI) baseada em QoC[14].

No ambiente distribuído, os nodos/hosts que executam uma instância do middleware estão organizados logicamente de maneira hierárquica, podendo se comunicar uns com uns outros a fim de enviar ou receber consultas e dados de contexto. A comunicação entre os nodos da hierarquia pode utilizar tanto redes infraestruturadas fixas como redes móveis *ad-hoc*. A arquitetura lógica do SALES, ilustrada na Figura 8.9, possui quatro tipos de nodos: CN (*Central Node*), BN (*Base node*), CUN (*Coordinator User Node*) e SUN (*Simple User Node*). O papel de cada um desses nodos é explicado a seguir.

[13]O processo de engenharia dirigida por modelos permite gerar uma implementação do sistema completa ou parcial a partir do modelo do sistema [55].

[14]Os autores do SALES, diferente dos autores dos trabalhos relacionados anteriores, consideraram que a noção de QoC abrange tanto a qualidade da informação como a qualidade do serviço de distribuição.

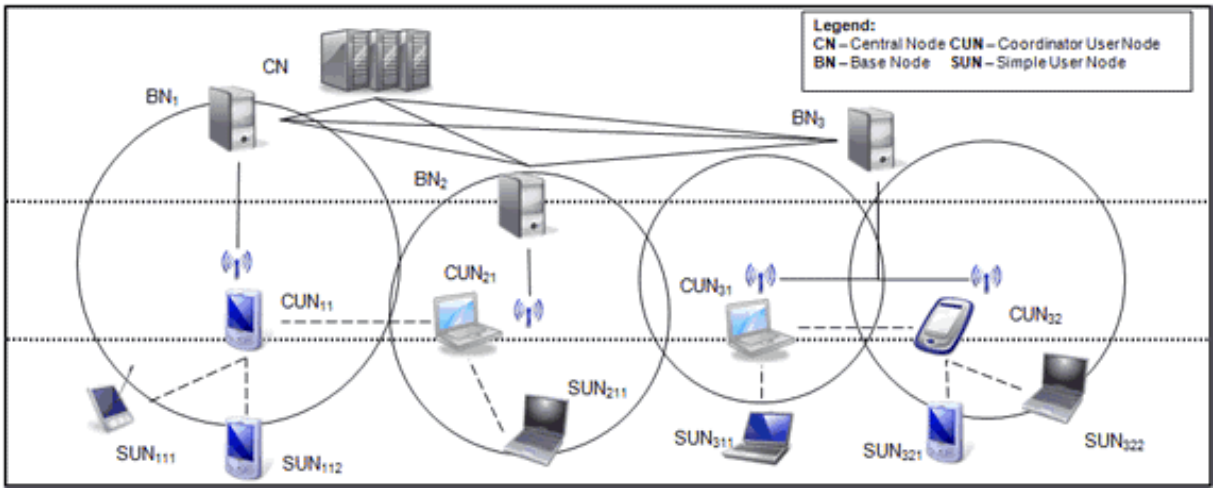

Figura 8.8: Arquitetura Lógica do SALES [26]

O *Central Node* é o nodo raiz que provê serviços de persistência e consulta de dados de contexto. O CN só pode ser acessado diretamente por nodos fixos da mesma LAN. O sistema pode agrupar múltiplos CNs em um cluster a fim de atender requisitos de escalabilidade e disponibilidade. O *Base node* é um nodo fixo que atua como ponto de entrada da infraestrutura fixa. Cada BN gerencia e integra nodos móveis espalhados em diversas redes infraestruturadas móveis (e.g. Wifi e 3G/4G). O *Coordinator User Node* é um nodo móvel especial que agrupa os demais nodos móveis em clusters e se responsabiliza por receber e encaminhar os dados de contexto e consultas oriundas desses nodos para o BN ao qual está conectado via rede infraestruturada e/ou para outros CUNs com os quais estabelece uma comunicação *ad-hoc mult-hop* via Wifi ou Bluetooth. O *Simple User Node* corresponde a cada nodo móvel do usuário que não é o líder do cluster. Cada SUN pode atuar como provedor ou consumidor de contexto. Tal como a comunicação entre CUNs, cada SUN se conecta de modo *ad-hoc* ao seu CUN.

A troca de dados de contexto pode acontecer tanto entre nodos pertencentes a níveis adjacentes como entre nodos pertencentes ao mesmo nível. Inicialmente, o fluxo de dados de contexto é roteado apenas verticalmente da fonte até o CN (chamado de caminho de disseminação de contexto) para que as informações sejam persistidas e ganhem visibilidade. Para orientar a disseminação por diferentes caminhos, SALES divulga **consultas de recuperação de contexto** (que expressam os requisitos de contexto dos consumidores) tanto horizontal como verticalmente. Ao receber uma consulta, o nodo verifica os dados de contexto armazenados localmente e, caso haja

uma correspondência positiva, cria uma **resposta de contexto** que será encaminhada de volta para o nodo solicitante utilizando uma solução de roteamento *hop-by-hop*.

O suporte ao desenvolvimento de aplicações do SALES fornece um conjunto de APIs, módulos e componentes pertencentes a uma arquitetura dividida em duas camadas: Facilidades e Mecanismos. Essa arquitetura é ilustrada na Figura 8.9.

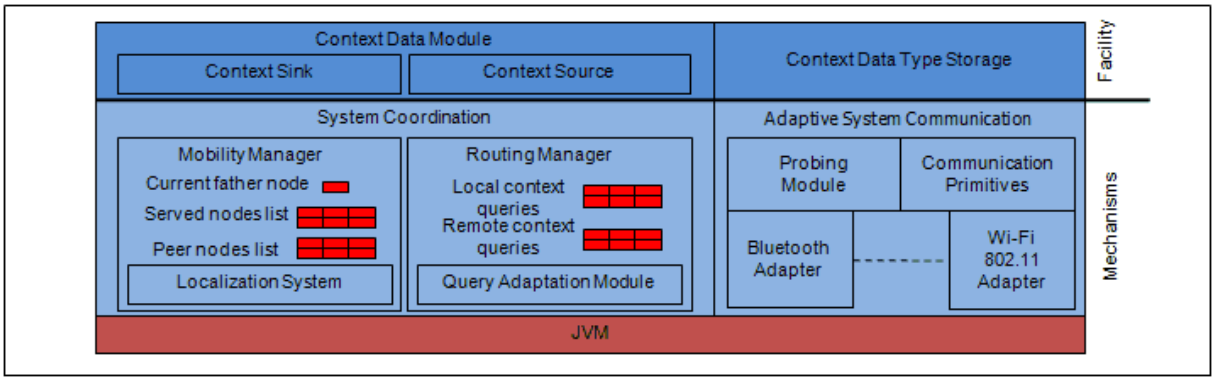

Figura 8.9: Arquitetura de Software do SALES [26]

A Camada de Facilidades implementa funcionalidades de alto nível para envio e recuperação de dados de contexto. O componente *Context Data Type Storage* mantém o registro de produtores, consumidores e tipos de dados de contexto por meio de um esquema XML. O *Context Data Module* é um componente genérico que foi especializado em dois sub-componentes: o *Context Source*, que permite criar e publicar dados de contexto, e o *Context Sink*, que permite recuperar dados de contexto por meio de consultas. Essas consultas fornecem informações de roteamento que são usadas pela camada de mecanismos para a disseminação dos dados de contexto.

A Camada de Mecanismos implementa funcionalidades de baixo nível relacionadas ao acesso ao sistema e roteamento de dados de contexto. O módulo *Adaptative System Communication* oferece uma API de comunicação genérica para a troca de mensagens que esconde detalhes específicos da comunicação sem-fio. Funções de comunicação de baixo nível são tratadas pelos componentes *Bluetooth Adapter* e *Wi-Fi 802.11 Adapter*. O módulo *System Coordination* é responsável pelo gerenciamento de mobilidade e seleção da infraestrutura de roteamento. O componente *Mobility Manager* recupera a localização dos nodos e lida com protocolos de gerenciamento de mobilidade cujo objetivo é manter a organização da estrutura hierárquica em três níveis, garantindo que a cada nodo móvel possa ser concedida uma conexão com um nodo superior. O *Routing Manager* realiza o roteamento de dados de contexto em cada

um dos diferentes níveis da hierarquia. Ele é responsável por receber consultas de contexto de outros nodos, encaminhá-las quando necessário e retornar as respostas.

SALES foca na qualidade do serviço de distribuição em detrimento da qualidade da informação. Os requisitos de QoC são expressos no SALES por meio de um *Context Data Distribution Level Agreement* (CDDLA). SALES considera três parâmetros de QoC. *Freshness*: valor lógico que estabelece limites em relação à idade da informação a ser recuperada; *Data Retrieval Time*: corresponde ao tempo máximo pelo qual o nodo está disposto a esperar por uma resposta de contexto, contado a partir da geração da consulta; *Priority*: especifica a classe do usuário e tem por objetivo permitir a diferenciação de tráfego quando vários dados têm de ser encaminhados. Existem três classes de usuário definidas estaticamente, sendo que os valores dos parâmetros são pré-fixados para cada classe. Classe *Ouro*: tem prioridade 0 e exige receber a versão mais recente do dado em um tempo de recuperação de até 2 segundos; Classe *Prata*: tem prioridade 1 e aceita receber dados válidos (mesmo que não seja o mais recente) em um tempo de recuperação de até 4 segundos; Classe *Bronze*: tem prioridade 2 e aceita receber dados já expirados em um tempo de recuperação de até 6 segundos.

Para que a recuperação de dados ocorra de acordo com o CDDLA, SALES mapeia automaticamente os parâmetros de QoC para parâmetros de consulta de recuperação de contexto. Os parâmetros de consulta especificam a forma como a infraestrutura roteará e tratará as consultas. O *Freshness* é mapeado para *Horizontal Time To Live (HTTL)*, que limita o número de saltos da consulta. Por exemplo, se o consumidor requer a versão mais atual do dado de contexto, então o HTTL receberá valor 0, indicando que a consulta deve chegar até o CN. O *Data Retrieval Time* é mapeado para *Query Total Lifetime (QTL)*, que faz com que a infraestrutura descarte a consulta depois que o tempo de validade da mesma expira. Portanto, se o tempo de recuperação exigido é de 2 segundos, então a consulta não pode permanecer válida por tempo maior que esse, já que qualquer resposta encaminhada após esse prazo irá representar tráfego inútil. Por fim, o *Priority* é mapeado para *Query Priority* (QP), que determina a prioridade de encaminhamento do dado conforme a classe. Para evitar uma sobrecarga do sistema de distribuição de dados de contexto e cumprir de QoC para cada classe de usuário, SALES monitora ativamente os recursos computacionais disponíveis e a demanda de serviços imposta pelos usuários, empregando algorítimos de roteamento adaptativos, que se ajustam de acordo com largura de banda disponível.

8.6 Análise Comparativa

As Tabelas 8.1 e 8.2 comparam as diferentes abordagens apresentadas nos trabalhos relacionados com o M-Hub/CDDL, levando em consideração a maneira como cada middleware aborda diferentes desafios do desenvolvimento de aplicações de IoT com requisitos de QoC. Os critérios adotados, que em parte são baseados em trabalhos que comparam middleware e/ou propõe critérios [70] [63] [112] [86] [8], são:

- **Gerenciamento de sensores heterogêneos**: verifica se o middleware provê serviços de gerenciamento (descoberta, registro e interação) de diferentes tipos de sensores e se há suporte para alguma tecnologia ou protocolo de comunicação.
- **Comunicação distribuída**: verifica se o middleware oferece mecanismos de suporte à comunicação distribuída, permitindo o desenvolvimento de aplicações tanto locais como remotas, ou se este define apenas uma API genérica, exigindo que o programador implemente detalhes da comunicação entre os nodos da rede.
- **Segurança**: verifica se o middleware possui mecanismos de suporte à segurança na distribuição dos dados, tais como autenticação, autorização e criptografia.
- **Tolerância a falhas de conectividade**: verifica se o middleware oferece suporte a mecanismos que garantam a confiabilidade de entrega dos dados em ambientes móveis com conectividade intermitente, tais como reconexão automática, sessão persistente, *buffer*, confirmação de entrega e retransmissão em caso de falha.
- **Descoberta de serviços de contexto baseada em QoC**: verifica se o middleware oferece mecanismos de registro, divulgação e consultas de serviços. Verifica-se ainda se o mecanismo de descoberta leva em consideração critérios de QoC.
- **Provisionamento de QoC**: verifica os tipos e quantidade de parâmetros de QoC providos pelo middleware, bem como se é possível definir novos parâmetros.
- **Avaliação de QoC**: verifica se o middleware oferece mecanismos que permitam calcular o valor dos parâmetros e/ou métricas de QoC em tempo de execução.
- **Monitoramento de Contexto baseado em QoC**: verifica se o middleware oferece mecanismos de suporte ao monitoramento de eventos, de modo que permita detectar tanto variações de contexto como oscilações na qualidade de contexto.

Middleware	Gerenciamento de Sensores Heterogêneos	Comunicação Distribuída	Segurança	Tolerância à Falhas de Conexão
M-Hub/CDDL	-Gerenciamento de sensores via S2PA; -Suporte nativo a Bluetooth Classic e BLE; -Extensível a implementação de tecnologias -Suporte a interação com sensores Android.	-Comunicação orientada a tópicos MQTT; -Baixo *overhead* na comunicação; -Suporte a aplicações locais e remotas; -Cliente Paho e Micro Broker Moquette; -Pode usar qualquer Server Broker MQTT.	-Autenticação de Usuários -Autorização de Acesso à Tópicos; -Criptografia Padrão SSL e TLS;	-Detecta falhas dos nodos; -Reconexão automática; -Sessão persistente; -3 Níveis de confiabilidade; -*Bufferização* intermediária.
AWARENESS	-Sem serviço de gerenciamento de sensores; -*Wrappers* escondem a heterogeneidade;	-Não padroniza nenhuma tecnologia ou protocolo de comunicação de rede; -Define apenas a API de comunicação.	-Não diz como faz autenticação -Autorização baseada em QoC;	-Não aborda.
COSMOS	-Sem serviço de gerenciamento de sensores; -Suporte a aquisição de dados sobre recursos do sistema operacional (e.g, memória e bateria); -*Context Collectors* escondem a heterogeneidade;	-Comunicação orientada à mensagens; -Suporte a aplicações locais e remotas; -Implementado com DREAM Framework.	-Não aborda.	-Não aborda.
COPAL	-Gerenciamento de sensores via *Device Manager*; -Sensores mapeados para Serviços Web; -Descrição de serviços baseada em UPnP; -*Wrappers* escondem a heterogeneidade.	-Comunicação orientada à mensagens; -Suporte apenas para aplicações locais. -Engine CEP é utilizada como Broker.	-Não diz como faz autenticação -Autorização baseada em atributos.	-Não aborda.
INCOME	-Sem serviço de gerenciamento de sensores -*Context Collectors* escondem a heterogeneidade;	-Comunicação orientada à mensagens; -Suporte a aplicações locais e remotas. -Implementado com MuDEBS framework.	-Autenticação a cargo da aplicação; -Autorização baseada em atributos.	-Não aborda.
SALES	-Sem serviço de gerenciamento de sensores -*Context Sources* escondem a heterogeneidade; -Suporte a Bluetooth Classic e Wifi é voltado para comunicação entre nodos da hierarquia.	-Comunicação orientada à mensagens; -Suporte a aplicações locais e remotas. -Implementado com Sockets Java UDP.	-Não aborda.	-Não aborda.

Tabela 8.1: Comparativo entre os trabalhos relacionados e o M-Hub/CDDL (Parte 1).

Middleware	Descoberta de Serviços baseada em QoC	Provisionamento de QoC	Avaliação de QoC	Monitoramento de Informações de Contexto com base em QoC
M-Hub/CDDL	-Ativa (via Consultas) ou Passiva (via Anúncios); -Tipos de Consulta: Instantânea ou Contínua; -Resposta das consultas é uma lista de serviços; -QoC é critério opcional da consulta de serviços.	-Define um conjunto padrão de 20 parâmetros de QoC ao todo (10 de QoI e 10 de QoS); -QoI e QoS são extensíveis.	-Define funções de avaliação de QoC para diversos parâmetros, tanto para QoI como para QoS; -Métrica padrão é a QoC média. -Extensível para novas funções.	-Monitoramento de Contexto e QoC com base na especificação de regras CEP escritas em EPL e definidas pela própria aplicação.
AWARENESS	-Descoberta de Serviços Ativa (via consultas); -Tipos de consultas: Instantânea e Subscrita; -Resposta da consulta é uma lista de serviços.	-Define um conjunto padrão de 06 parâmetros de QoC ao todo (5 de QoI e 1 de QoS); -Não prevê extensão da QoC.	-Funções são definidas pelo programador da aplicação;	-Monitoramento de Contexto e QoC com base na especificação de regras Evento-Condição-Ação definidas pela própria aplicação;
COSMOS	-Ativa (via Consultas) ou Passiva (via Anúncios); -QoC é critério opcional na consulta de contexto; -QoC pode ser divulgada de dois modos: junto ou separadamente das informações de contexto.	-Cabe ao programador estender o middleware e definir os parâmetros; -Aborda apenas a QoI.	-Funções são definidas pelo programador da aplicação;	-Sem suporte a monitoramento; -Técnica de Monitoramento de Contexto/QoC deve ser definida pelo programador da aplicação.
COPAL	-Descoberta de Serviços Ativa (via consultas); -Resposta da consulta é o dado de contexto; -QoC é critério opcional da consulta de contexto.	-Define um conjunto padrão de 09 parâmetros de QoC ao todo (5 de QoI e 4 de QoS); -Não prevê extensão da QoC.	-Define funções de avaliação de para alguns parâmetros de QoI; -Não prevê extensões de funções.	-Monitoramento de Contexto e QoC com base em *queries* COPAL definidas pela própria aplicação, convertidas em regras EPL/CEP.
INCOME	-Descoberta de Serviços Ativa (via consultas); -Resposta das consultas é o dado de contexto; -QoC é critério opcional da consulta de contexto.	-Cabe ao programador estender o middleware e definir os parâmetros; -Aborda apenas a QoI.	-Funções são definidas pelo programador da aplicação;	-Monitoramento de Contexto e de QoC baseado em expressões que são escritas em JavaScript/XPath e definidas pela própria aplicação.
SALES	-Descoberta de Serviços Ativa (via consultas); -Resposta das consultas é o dado de contexto; -QoC é um critério obrigatório da consulta.	-Define um conjunto padrão com 03 parâmetros de QoC, mas aborda somente a QoS; -Não prevê extensão da QoC.	-Valor dos parâmetros de QoC é definido estaticamente com base na classe de cada usuário.	-Aplicações não podem definir suas próprias regras de monitoramento; -Há uma única política que monitora recursos do sistema para dar suporte ao roteamento adaptativo de dados, a fim de garantir contratos de QoC.

Tabela 8.2: Comparativo entre os trabalhos relacionados e o M-Hub/CDDL (Parte 2).

A seguir, cada critério de comparação é discutido em uma subseção. Depois, concluí-se qual abordagem de middleware teve o melhor desempenho de modo geral.

8.6.1 Gerenciamento de Sensores Heterogêneos

A comparação mostra que todas as abordagens de middleware fornecem componentes que escondem das camadas superiores toda a heterogeneidade do hardware e tecnologias de comunicação utilizadas pelos sensores. Afinal, essa é a função natural de um middleware. No entanto, apenas COPAL e M-Hub/CDDL apresentam serviços de gerenciamento voltados especificamente para a descoberta, registro e interação com sensores. No caso do COPAL, o componente *Device Manager* é que tem essa responsabilidade, enquanto que no M-Hub/CDDL essa funcionalidade compete ao *S2PA Service*. COPAL adota UPnP como mecanismo padrão de descrição dos serviços. Já o M-Hub/CDDL implementa sua própria forma de representação dos dispositivos, cujas informações são armazenadas em um diretório de serviços local.

Tanto M-Hub/CDDL como COPAL são abordagens de middleware abertas à implementação de suporte à diversas tecnologias e protocolos de comunicação com sensores. Entretanto, considera-se que o M-Hub/CDDL leva vantagem sobre as demais abordagens, principalmente, pelo fato de que já fornece suporte nativo a Bluetooth Classic e BLE, que estão entre as principais tecnologias de comunicação de curta distância adotadas por sensores. Sendo assim, o esforço do programador que utiliza o M-Hub/CDDL é consideravelmente menor caso a aplicação de IoT exija interação com sensores compatíveis com essas tecnologias.

Cabe ressaltar que o middleware SALES, embora ofereça suporte a *Bluetooth Classic* e *Wifi*, utiliza essas tecnologias apenas para a implementação de um serviço de comunicação *ad-hoc* entre os nodos da hierarquia que executam uma instância do middleware. Sendo assim, o gerenciamento de dispositivos do SALES só se aplica a esses nodos e não aos sensores. COSMOS, embora não ofereça suporte nativo a nenhuma tecnologia de comunicação com sensores, implementa um mecanismo baseado no framework SAJE que permite monitorar e coletar dados sobre recursos do sistema operacional.

8.6.2 Comunicação Distribuída

A comparação mostra que COSMOS, INCOME, SALES e M-Hub/CDDL apresentam suporte à comunicação distribuída. COSMOS implementa esse suporte utilizando o DREAM framework, orientado à mensagens, podendo utilizar tanto TCP como UDP na camada de transporte. INCOME usa o framework MuDEBS, também orientado à mensagens. SALES implementa a comunicação utilizando *sockets* UDP Java. M-Hub/CDDL adota um modelo de interação baseado em tópicos, implementando o suporte à comunicação com base na *Paho Client* e no *Micro Broker Moquette*, podendo utilizar qualquer *Server Broker* compatível com o protocolo MQTT.

Embora essas quatro abordagens de middleware ofereçam suporte a aplicações de IoT distribuídas, considera-se que o M-Hub/CDDL leva vantagem sobre as demais. Essa vantagem advém, principalmente, do fato do M-Hub/CDDL ser o único middleware que adota um protocolo que lida com questões críticas da comunicação em IoT, tais como baixa largura de banda e alta latência das redes. INCOME, COSMOS e SALES se baseiam no uso de tecnologias de comunicação que, por si só, não são capazes de lidar adequadamente com essas questões. MQTT é um protocolo de comunicação leve, que consome pouca energia, o que é ideal para uso em dispositivos com recursos limitados. Além disso, sua especificação aberta permite a implementação de *frameworks* e aplicações baseadas em MQTT em diversas plataformas. A vantagem é que isso favorece a interoperabilidade entre M-Hub/CDDL e diversos sistemas baseados no mesmo protocolo. M-Hub/CDDL apresenta ainda importantes características relacionadas à comunicação distribuída que, no melhor do nosso conhecimento, não são encontradas nos trabalhos relacionados, tais como: transparência de distribuição, gerenciamento de múltiplas conexões e interfaces de programação que tornam uniforme o desenvolvimento de aplicações locais e remotas.

As abordagens AWARENESS e COPAL não possuem suporte nativo à comunicação distribuída. No caso do AWARENESS, embora ele defina um conjunto de interfaces para a comunicação distribuída (*Context Broker e Context Source*), esse middleware não padroniza a tecnologia de comunicação em rede, deixando essa tarefa para o programador da aplicação. COPAL usa uma *Engine CEP* como *broker* local. Uma vez que esse *broker* não é acessível remotamente, não há como aplicações distribuídas se conectarem a ele, sem que haja que esforço adicional por parte do programador.

8.6.3 Segurança

A comparação mostra que quatro das seis abordagens de middleware lidam com aspectos de segurança. As soluções COPAL, AWARENESS e INCOME possuem mecanismos de segurança voltados ao controle de acesso aos dados de contexto baseados na especificação de atributos. Em relação à autenticação, os trabalhos do COPAL e AWARENESS não explicam se tal mecanismo é provido pelo middleware ou se este fica a cargo da aplicação. Os trabalhos do AWARENESS dizem explicitamente que a aplicação deve se encarregar da autenticação, pois o mecanismo de autorização provido pelo middleware pressupõe que todos os usuários conectados ao broker são confiáveis. Nenhuma dessas abordagens menciona suporte à criptografia.

M-Hub/CDDL fornece ao programador mais mecanismos de segurança que as demais abordagens de middleware, fazendo uso do suporte à autenticação, autorização e criptografia (*Secure Socket Layer* (SSL) e *Transport Layer Security* (TLS)) providos pelo *Server Broker* MQTT. Entretanto, diferente das demais abordagens, o mecanismo de controle se acesso ao dados de contexto do M-Hub/CDDL é baseado em tópicos e não nos atributos da mensagem. Sendo assim, define-se apenas quais usuários estão autorizados a assinar o tópico, mas esse mecanismo não leva em consideração requisitos de acesso relacionados à qualidade das informações publicadas no tópico. Logo, considera-se que neste critério, M-Hub/CDDL possui vantagens e desvantagens sobre as demais abordagens.

8.6.4 Tolerância a Falhas de Conectividade

A comparação mostra que apenas M-Hub/CDDL apresenta explicitamente mecanismos que permitem o desenvolvimento de aplicações tolerantes a falhas de conectividade. Em cenários de IoMT, abordar esse desafio é ainda mais relevante pois os gateways de IoT e as próprias aplicações cientes de contexto fazem uso de redes móveis cuja conectividade por vezes é fraca ou intermitente. Um dos motivos que tornam o M-Hub/CDDL mais qualificado que as demais abordagens de middleware, no que se refere a este critério, é o fato do M-Hub/CDDL implementar um serviço de distribuição que se baseia num protocolo de comunicação que, além de leve, tem suporte a três níveis de confiabilidade na camada de aplicação. M-Hub/CDDL permite

que a aplicação configure facilmente o nível de confiabilidade mais adequado às suas necessidades e ao ambiente de rede no qual executam, sem expor para a aplicação os detalhes inerentes às configurações exigidas pelo protocolo MQTT. M-Hub/CDDL aumenta a confiabilidade padrão do MQTT por meio de um mecanismo de *bufferização* intermediária. Esse mecanismo é responsável por armazenar mensagens publicadas enquanto a aplicação estava *off-line* e retransmiti-las assim que uma conexão com o *broker* é reestabelecida. Além disso, M-Hub/CDDL oferece mecanismos de sessão persistente, reconexão automática e monitoramento de conexão. Esses mecanismos permitem um gerenciamento mais efetivo da comunicação em redes instáveis.

Em relação às demais abordagens de middleware, COSMOS, que utiliza o framework DREAM, por exemplo, tem à sua disposição apenas a confiabilidade de nível de transporte, provida pelo protocolo TCP. Sendo assim, sem a implementação de mecanismos adicionais de confiabilidade, não há como a camada de aplicação controlar com precisão quais de suas mensagens foram efetivamente entregues, pois a confirmação padrão do protocolo TCP/IP e os mecanismos de retransmissão se aplicam aos pacotes e não às mensagens da aplicação. SALES, por outro lado, que utiliza puramente sockets UDP, sem recorrer a outras bibliotecas de comunicação, não tem a sua disposição nem mesmo a confiabilidade básica da camada de transporte.

8.6.5 Descoberta de Serviços de Contexto Baseada em QoC

Existem basicamente dois tipos de descoberta de serviços que podem ser adotados por um middleware de contexto: descoberta ativa ou descoberta passiva. Na descoberta ativa, a aplicação ciente de contexto toma a iniciativa de fazer uma consulta e espera que a infraestrutura de gerenciamento dos serviços responda a essa consulta. Na descoberta passiva, a aplicação ciente de contexto apenas "escuta" anúncios de divulgação dos serviços disponíveis, publicados pelo serviço de descoberta, sem que ela precise realizar qualquer tipo de consulta. As abordagens de middleware que têm suporte a ambos os tipos de descoberta de serviços, tendem a ser mais flexíveis, neste aspecto, pois têm maiores chances de se adequarem aos diferentes cenários da IoT.

Todas as abordagens de middleware apresentadas possuem mecanismos de consulta (descoberta ativa). Entretanto, a maneira como cada uma implementa esse mecanismo é diferente. Por um lado, COSMOS, COPAL, INCOME e SALES

implementam **consultas de recuperação de contexto**, para as quais a resposta é uma instância/amostra da informação de contexto propriamente dita, que atende aos critérios de contexto e de QoC especificados pela aplicação na consulta. Por outro lado, AWARENESS e M-Hub/CDDL implementam **consultas de listas serviços**, para as quais a resposta é uma lista contendo os provedores de serviços e os respectivos tipos de informação de contexto que cada um disponibiliza, mas não a informação de contexto em si. Considera-se que as consultas de listas de serviços são mais flexíveis que as consultas de recuperação de contexto, pois as primeiras promovem o desacoplamento entre as etapas de descoberta de serviços e recuperação de contexto.

Nas abordagens M-Hub/CDDL, COSMOS, COPAL e INCOME, a QoC é um critério opcional das consultas, visto que nem todos os atributos da informação de contexto ou de QoC são de interesse da aplicação consumidora e, portanto, não precisam ser especificados. No SALES, considera-se que QoC é um critério de consulta obrigatório, visto que os parâmetros de QoC são utilizados pelo middleware para o estabelecimento de um contrato de QoC entre produtores e consumidores. Logo, especificar os parâmetros de QoC na consulta é necessário para que o SALES determine a maneira como essas consultas serão roteadas pelos nodos da infraestrutura.

M-Hub/CDDL e AWARENESS destacam-se das demais abordagens de middleware pelo fato de serem as únicas que apresentam mais de um mecanismo de consulta. A semântica das consultas instantânea e subscrita do AWARENESS são muito semelhantes à das consultas instantânea e contínua, respectivamente, do M-Hub/CDDL. Entretanto, enquanto AWARENESS executa suas consultas sobre uma ontologia específica de domínio, que descreve os tipos de contexto disponíveis, armazenada em disco, M-Hub/CDDL executa suas consultas sobre fluxos de eventos processados em memória. Sendo assim, teoricamente, a execução das consultas do M-Hub/CDDL tende a ser mais rápida que do AWARENESS. M-Hub/CDDL e COSMOS são as únicas abordagens de middleware que, além do mecanismo de consulta, oferecem também mecanismos de divulgação espontânea dos serviços de contexto (descoberta passiva). Diante do exposto, considera-se que o M-Hub/CDDL leva vantagem sobre as demais abordagens, neste critério, pois atende a um número maior de requisitos relacionados à descoberta de serviços: (i) descoberta ativa e passiva; (ii) consulta de recuperação de listas de serviços; (iii) consultas instantânea e continua; (iv) consultas sobre fluxos de eventos processados em memória.

8.6.6 Provisionamento de QoC

A comparação mostra que M-Hub/CDDL é a abordagem de middleware que define/implementa a maior quantidade padrão de parâmetros de qualidade de contexto (20 parâmetros de QoC ao todo), tanto no que diz respeito à qualidade da informação (10 parâmetros de QoI), como em relação à qualidade do serviço (10 parâmetros de QoS). Além disso, é possível estender o middleware, a fim de adicionar novos parâmetros de QoI e QoS. Sendo assim, considera-se que, dentre as abordagens de middleware comparadas, o M-Hub/CDDL é aquela que se mostra mais apropriada ao desenvolvimento de uma grande variedade de aplicações de IoT/IoMT em diferentes domínios, principalmente aquelas que exigem múltiplos parâmetros de QoC. Por exemplo, é bastante evidente que diversas aplicações cientes de contexto emergentes, tais como aquelas desenvolvidas nos estudos de caso utilizando o M-Hub/CDDL (ver Seção 6), não teriam todos os seus requisitos de QoC prontamente atendidos pelas abordagens dos trabalhos relacionados, exigindo muito esforço de implementação com extensões do suporte à QoC do middleware, principalmente no que se refere à QoS, que foi o tipo de QoC mais exigida naqueles estudos de caso.

Em relação às demais abordagens de middleware, COSMOS e INCOME voltam suas atenções exclusivamente para a qualidade da informação. Em ambas as abordagens, cabe ao programador estender o middleware, definir e implementar os parâmetros de QoI. Mesmo que fosse levado em consideração que os parâmetros de QoC implementados nos estudos de casos foram incorporados aos respectivos arcabouços, o que não é garantido, ainda assim o número de parâmetros que seria disponibilizado por padrão em cada middleware (COSMOS com 3 e INCOME com 5 parâmetros de QoI) é pouco, se comparado ao M-Hub/CDDL. O suporte à QoC do middleware AWARENESS é quase totalmente voltado para a qualidade da informação (5 parâmetros de QoI), oferecendo um suporte mínimo à qualidade de serviço (apenas 1 parâmetro de QoS). COPAL oferece suporte tanto a QoI como QoS, de forma equilibrada (5 parâmetros de QoI e 4 de QoS), mas a quantidade padrão de parâmetros de QoC provida nativamente pelo middleware, ainda que superior às abordagens dos demais trabalhos relacionados, é consideravelmente menor que a disponibilizada pelo M-Hub/CDDL. SALES foca completamente na qualidade da distribuição, oferecendo a mesma quantidade de parâmetros de QoS que o COPAL (3 parâmetros de QoS).

Os trabalhos referentes a COPAL, AWARENESS e SALES não indicam a possibilidade de estender o suporte à QoC do middleware a fim de permitir a adição de novos parâmetros. Para essas abordagens, o número de parâmetros de QoC definidos por padrão também é significativamente menor que a disponibilizada pelo M-Hub/CDDL.

As experiências adquiridas com o desenvolvimento do M-Hub/CDDL mostraram que, mesmo diante da possibilidade de estender o middleware, a fim de e adicionar parâmetros de QoI e QoS ao arcabouço, a implementação de mecanismos de provisionamento de QoC não é algo trivial. Entre as lições aprendidas, chegou-se a conclusão que implementar parâmetros de QoS pode ser ainda mais difícil que implementar parâmetros de QoI, pois a qualidade de serviço requer que o programador que estenderá o middleware lide com vários aspectos de controle da comunicação, tolerância a falhas, políticas de gerenciamento de filas e conexões, dentre outras complexidades que tendem a aumentar bastante o tempo de desenvolvimento. Portanto, a complexidade desse tipo de implementação reforça a importância do middleware de contexto prover suporte nativo a uma quantidade significativa de parâmetros de QoI e QoS, pois isso poupará esforços do programador da aplicação.

8.6.7 Avaliação de QoC

A expressão "Avaliação de QoC", levando em consideração a literatura, geralmente, é usada para se referir à capacidade de calcular o valor dos parâmetros de qualidade da informação, principalmente aqueles que não são prontamente fornecidos pelos sensores e/ou pela própria aplicação, ou a capacidade de computar métricas de QoI. Em relação à qualidade de serviço, o termo "Avaliação de QoC" não é geralmente é aplicado, embora parâmetros e métricas de QoS também possam ser computados.

As abordagens COSMOS e INCOME permitem que o programador estenda o middleware e implemente suas próprias funções de avaliação de parâmetros e/ou métricas de QoI, visto que, como explicado no critério anterior, compete ao programador a tarefa de definir os parâmetros de QoC. Logo, não é possível fornecer funções de avaliação antes de definir quais parâmetros farão parte do modelo de contexto usado pelo middleware. AWARENESS, embora defina um conjunto padrão de parâmetros de QoI, não define as funções de avaliação desses parâmetros, deixando essa tarefa a cargo do programador. COPAL, que também define um conjunto padrão

de parâmetros de QoI e QoS, define funções de avaliação para alguns parâmetros de QoI, mas apenas em casos nos quais isso é necessário, pois há parâmetros cujo valor precisa ser fornecido pela aplicação. Assim como no caso dos parâmetros, não é dito se o middleware pode ser estendido para adicionar funções de avaliação. SALES é um caso particular, pois o valor do parâmetros de QoS, correspondentes à classe do usuário, são definidos estaticamente pelo middleware, dispensando o cômputo desses parâmetros. Tanto no caso do COPAL como do AWARENESS, há algumas variáveis de controle diretamente relacionadas ao provisionamento de QoS que são computadas dinamicamente pelo middleware, mas que não são consideradas parâmetros de QoS.

Diante do exposto, neste critério, considera-se que o M-Hub/CDDL leva certa vantagem sobre as demais abordagens de middleware, pois, além de avaliar a QoC média associado aos provedores de serviços, bem como computar dinamicamente o valor de vários parâmetros de QoI e QoS, o middleware pode ser estendido a fim de implementar novas funções de avaliação. Entretanto, para adicionar algumas funções de avaliação realizadas pelo *QoC Evaluator* (pois nem todo tipo de avaliação depende desse componente), ainda se faz necessário promover modificações no middleware, a fim de permitir que a aplicação defina funções de avaliação em tempo de execução, que serão executadas dinamicamente pelo referido componente de avaliação de QoC.

8.6.8 Monitoramento de Contexto baseado em QoC

A comparação mostra que, em alguma medida, a maioria das abordagens de middleware apresentadas possui algum tipo de suporte ao monitoramento de contexto e de QoC. Entretanto cada tipo de monitoramento possui suas especificidades.

AWARENESS permite que aplicações definam requisitos de monitoramento de contexto e QoC, com base na especificação de regras do tipo *Evento-Condição-Ação* e raciocínio sobre ontologias. O foco da abordagem está na detecção de situações (contexto de alto nível) a partir do uso de mecanismos de inferência sobre dados de baixo nível. Os eventos/situações detectadas podem desencadear a execução de ações de adaptação. A vantagem dessa abordagem é o grau de expressividade semântica da linguagem utilizada no monitoramento. A desvantagem está relacionada ao custo do processamento baseado em ontologias em dispositivos móveis, que geralmente é elevado, se comparado, por exemplo, com o processamento sobre fluxos de eventos.

COPAL e M-Hub/CDDL oferecem suporte ao monitoramento de contexto e QoC, com base no processamento de eventos complexos (CEP). Essas abordagens, se comparadas ao AWARENESS, tem a desvantagem de não ter suporte processamento semântico do eventos, mas tem as vantagens da grande velocidade do processamento e menor exigência de recursos computacionais ao executar o monitoramento. Mesmo que ambos adotem CEP, COPAL e M-Hub/CDDL, diferem significativamente na maneira como os requisitos de monitoramento são especificados pela aplicação.

COPAL opta pela especificação indireta, ou seja, a aplicação define seus requisitos de monitoramento escrevendo *queries* COPAL, que são convertidas pelo middleware em regras EPL, de modo transparente. Entretanto, os autores do trabalho não explicam se as *queries* COPAL são tão expressivas quanto as regras EPL, ou seja, não é sabido se todos os padrões de processamento de eventos complexos (e.g., filtragem, agregação, diferenciação, sumarização, etc.), especificados com EPL, podem ser definidos utilizando *queries* COPAL. Considera-se que o mapeamento completo, entre regras EPL e *queries* COPAL, é indispensável, pois permite que as aplicações, que utilizam o COPAL, possam definir regras de monitoramento realmente complexas.

A fim de explorar toda a expressividade e padrões de processamento de eventos complexos providos pela EPL, M-Hub/CDDL adota a especificação direta, ou seja, a aplicação define próprios seus requisitos de monitoramento escrevendo regras EPL explicitamente, sem conversões. Outra motivação para o M-Hub/CDDL adotar a definição de regras EPL diretamente, advém do fato dessa linguagem já ser um padrão difundido, que tende a ser mais facilmente assimilado pelo desenvolvedores, principalmente, devida à semelhança com a SQL, que é bastante popular. Considera-se que, definir uma sintaxe própria para as regras de monitoramento, só faria aumentaria a curva de aprendizado do middleware nos estudos de caso com usuários (ver Seção 6.2). O suporte ao provisionamento de QoC do M-Hub/CDDL ainda coloca o seu mecanismo de monitoramento em vantagem numérica sobre todas as abordagens. Por ser aquela que define mais parâmetros de QoC, consequentemente, M-Hub/CDDL é também a que permite monitorar a maior quantidade padrão de parâmetros de QoC.

INCOME permite que as aplicações consumidoras definam seus requisitos de monitoramento de contexto e QoC com base em funções de filtragem JavaScript, que avaliam expressões XPath. Os padrões de processamento de eventos providos por esse mecanismo de monitoramento são semelhantes àqueles providos pela EPL/CEP.

COSMOS não tem nenhum tipo suporte de nativo para o monitoramento, visto que esse middleware não define nenhuma técnica de processamento de dados de contexto. Portanto, fica a cargo do programador da aplicação a tarefa de implementar todo suporte ao monitoramento de contexto e QoC que a aplicação vier a precisar.

SALES é um caso particular. Os autores desse trabalho não mencionam a possibilidade das aplicações definirem seus próprios requisitos de monitoramento. Contudo, há um mecanismo de monitoramento implementado nesse middleware, mas que só se aplica ao monitoramento dos recursos e da qualidade do serviço de distribuição, sendo que a aplicação não tem qualquer conhecimento ou controle sobre o mecanismo. O objetivo desse monitoramento é dar suporte à políticas de roteamento adaptativo dos dados de contexto, a fim de garantir a QoC contratada, de acordo com a classe dos usuários, bem como gerenciar a escalabilidade do serviço de distribuição.

É importante ressaltar ainda que existem trabalhos focados em sistemas de monitoramento de QoC, tais como o QoMonitor [9], que foi integrado ao middleware OpenCOPI [67]. A comparação não considerou o OpenCOPI devido à natureza distinta desse middleware em relação aos outros. Enquanto M-HUb/CDDL, AWARENESS, COSMOS, COPAL, INCOME e SALES visam apoiar o desenvolvimento de aplicações cientes de contexto, o OpenCOPI é uma plataforma que visa a integração entre aplicações, serviços e sistemas de middleware cientes de contexto heterogêneos, criando um sistema ciente de contexto unificado e transparente. Com base nessa integração, o QoMonitor é capaz de monitorar metadados de QoI e QoS fornecidos pelos diferentes serviços e sistemas de middleware, bem como notificar as aplicações.

8.6.9 Resultado da Análise Comparativa

De acordo com os aspectos de comparação analisados, de maneira geral, verifica-se que o M-Hub/CDDL mostrou-se uma abordagem de middleware bem mais completa que as apresentadas nos trabalhos relacionados. A abordagem aqui proposta, além de atender todos os critérios estabelecidos, também apresentou as soluções mais eficientes para todos os desafios/problemas explorados. Portanto, considera-se que o M-Hub/CDDL, dentre todas as abordagem de middleware analisadas, é aquela apresenta maior potencial para atender requisitos de gerenciamento de contexto e QoC, sendo aquela que tende a facilitar mais o desenvolvimento de aplicações de IoT.

8.7 Conclusão do Capítulo

Este capítulo apresentou as principais abordagens de middleware de contexto com suporte à QoC encontradas na literatura: AWARENESS, COSMOS, COPAL, INCOME e SALES. Essa é uma das contribuições desta pesquisa, uma vez que, no melhor do nosso conhecimento, não há revisões de literatura que analisem com profundidade os diversos aspectos do provisionamento de QoC em middleware de contexto. Foi promovida então uma ampla análise comparativa entre as abordagens dos trabalhos relacionados e aquela que está sendo proposta nesta tese, que levou em consideração diversos critérios relevantes, ligados ao gerenciamento de contexto (aquisição, processamento e distribuição), provisionamento e monitoramento de QoC.

Em relação à heterogeneidade de sensores, além de uma API e um serviço de gerenciamento que abstraem a complexidade na interação com objetos inteligentes, M-Hub/CDDL tem suporte a um número maior de tecnologias de comunicação. Em relação à comunicação distribuída e descoberta de serviços, M-Hub/CDDL permite o desenvolvimento de aplicações locais e remotas de maneira transparente, sendo que essas aplicações podem descobrir os provedores de serviços e seus respectivos índices de qualidade de diferentes modos suportados pelo middleware. O serviço de distribuição do M-Hub/CDDL é o que se mostra mais adequado para o uso em redes com baixa largura de banda e alta latência, principalmente por ser o único entre os analisados que adota um protocolo de IoT padrão. Em relação à segurança, M-Hub/CDDL se destaca das demais abordagens, uma vez que oferece mecanismos de autenticação, autorização e criptografia, enquanto que as outras oferecem apenas autorização. Em relação à falhas na comunicação, M-Hub/CDDL é o único, dentre os apresentados, que provê suporte à confiabilidade de entrega dos dados em nível de aplicação. M-Hub/CDDL é ainda a abordagem que possui o provisionamento de QoC mais amplo, abrangendo suporte nativo ao maior número de parâmetros de QoI e QoS, podendo ser estendido. Consequentemente, M-Hub/CDDL é também a abordagem que avalia e monitora a maior quantidade padrão de parâmetros de QoC, sendo que esses mecanismos conta com as vantagens do processamento de eventos complexos.

Diante do exposto, pode-se concluir que o M-Hub/CDDL é a abordagem de middleware que, dentre as que foram comparadas, atende melhor os requisitos de gerenciamento e qualidade de contexto, impostos pelas diferentes aplicações de IoT.

9 Considerações Finais

A Internet das Coisas (IoT) tem motivado cada vez mais o desenvolvimento de novas aplicações e serviços emergentes que possuem requisitos de qualidade de contexto (QoC). É muito importante atender todos os requisitos de QoC das aplicações, tanto de QoI como de QoS, pois a qualidade de contexto, de modo geral, tem impacto direto e significativo sobre as experiências dos usuários, que podem ser positivas ou negativas, dependendo do tipo e nível de QoC disponível para a aplicação. O problema em questão reside na grande complexidade do desenvolvimento dessas aplicações, principalmente as que possuem múltiplos requisitos de QoI e QoS a serem atendidos. Se por um lado, a tarefa de implementar requisitos inerentes à lógica de negócio das aplicações de IoT, por si só, já é difícil, por outro, a tarefa de implementar requisitos de gerenciamento de contexto (aquisição, processamento e distribuição) e qualidade de contexto (provisionamento, avaliação e monitoramento) aumenta significativamente o tempo e esforço empregados pelo programador no desenvolvimento dessas aplicações.

O estado da arte em middleware de contexto é amplo, possuindo inúmeras abordagens focadas em resolver diferentes problemas ligados ao desenvolvimento de aplicações de IoT. Entretanto, especificamente no que se refere a middleware de contexto com suporte à QoC, há poucas propostas na literatura, sendo que as abordagens de middleware mais representativas desse tipo específico de middleware são: AWARENESS, COSMOS, COPAL, INCOME e SALES. Analisando-se essas abordagens, pode-se concluir que, embora relevantes no sentido de tentar abordar aspectos de gerenciamento e qualidade de contexto, elas ainda possuem algumas limitações.

Com o objetivo de facilitar o desenvolvimento de aplicações de IoT que têm requisitos de QoC, esta tese propôs uma nova abordagem de middleware de contexto, denominada M-Hub/CDDL. A fim de resolver o problema, essa abordagem propõe um amplo conjunto de abstrações de programação e serviços que simplificam o gerenciamento de contexto e atendem a diversos requisitos relacionados à qualidade da informação e do serviço de distribuição. Esta tese focou na apresentação dos mecanismos desenvolvidos mais diretamente relacionados ao problema de pesquisa:

gerenciamento de sensores, provisionamento de QoC, descoberta de serviços e monitoramento de QoC.

Comparando-se o M-Hub/CDDL com as abordagens relacionadas, pode-se concluir que houve avanços significativos em relação ao estado arte, no sentido de que algumas das limitações das abordagens existentes na literatura foram superadas pela solução proposta. Um dos pontos de avanço foi no gerenciamento de dispositivos, cujo mecanismo proposto pela M-Hub/CDDL permite que a aplicação tenha maior controle sobre os sensores que serão utilizados, sendo que as demais abordagens, no geral, se limitam a apenas esconder a heterogeneidade de sensores da camada de aplicação, mas não a gerenciá-la. Outro ponto de avanço significativo foi o provisionamento de QoC. Enquanto que as abordagens, de modo geral, deixam de atender requisitos de QoS por estarem muito focadas na QoI, a M-Hub/CDDL é uma abordagem mais ampla, que contempla de forma muito consistente tanto a qualidade da informação como a qualidade do serviço de distribuição. Embora a abordagem possa ser estendida para a definição de novos parâmetros, a M-Hub/CDDL já define um conjunto padrão de parâmetros e políticas que permitem atender uma grande variedade de aplicações de IoT. A maior parte das abordagens transfere toda a responsabilidade da definição e implementação dos parâmetros de QoC para o desenvolvedor da aplicação.

Embora a vantagem numérica seja importante para atender mais requisitos de QoC com menor esforço do desenvolvedor, o avanço mais significativo está na definição de mecanismos que permitem que a aplicação controle a qualidade de serviço, ao invés de apenas estar ciente dela. Além disso, a M-Hub/CDDL é uma abordagem projetada de tal forma que os parâmetros de QoS podem ser modificados em tempo de execução, um aspecto dinâmico da qualidade de contexto que não é explorado nas abordagens relacionadas. Em relação à descoberta de serviços, os avanço da M-Hub/CDDL sobre outras abordagens reside na flexibilidade dos mecanismos, permitindo descoberta ativa e passiva, bem como possibilitando a especificação de consultas instantâneas e contínuas, sendo que este último tipo de consulta é bastante relevante para cenários de IoMT. As demais abordagens, embora contemplem mecanismos de descoberta de serviços, não foram projetadas para permitir uma descoberta tão flexível quanto a que a M-Hub/CDDL possui. Uma vez que nenhuma das abordagens analisadas lida com tolerância a falhas na entrega de dados, o mecanismo de confiabilidade configurável em nível de aplicação também é

considerado um avanço, pois diversas aplicações executam em ambientes móveis com conectividade intermitente. Logo, conclui-se que a M-Hub/CDDL é uma proposta que oferece soluções que abrangem todos os desafios apresentados , enquanto que as demais abordagens lidam com apenas alguns deles, sendo que essas soluções são geralmente menos eficientes que as propostas neste trabalho em cada aspecto. Apesar disso, a M-Hub/CDDL tem limitações quanto à expressividade semântica, não estando isenta de problema relativos a ambiguidades.

Avaliou-se a abordagem proposta quantitativa e qualitativamente, obtendo-se resultados muito satisfatórios. Em termos quantitativos, os resultados da avaliação de desempenho provaram a eficiência do M-Hub/CDDL em relação ao tempo de entrega dos dados, confiabilidade em redes instáveis, tempo de monitoramento e consumo de recursos (memória e bateria). Em termos qualitativos, os resultados da prova de conceito mostraram que os mecanismos do M-Hub/CDDL se aplicam a sistemas cientes de contexto do mundo real, enquanto que os resultados do estudo de caso com usuários mostraram um alto grau de satisfação dos mesmos em relação às abstrações de programação, serviços e suporte à QoC providos pelo middleware. Portanto, no geral, esses resultados comprovam a eficiência do M-Hub/CDDL como abordagem facilitadora do desenvolvimento de aplicações de IoT com requisitos QoC, validando a hipótese de investigação e comprovando que os objetivos foram atingidos.

9.1 Contribuições

Considera-se que as principais contribuições desta pesquisa são:

- Uma definição de QoC mais abrangente que as apresentas na literatura, que leva em consideração não apenas a qualidade da informação, mas também a qualidade de serviço e a qualidade de dispositivo.
- Uma ampla comparação entre diferentes abordagens de middleware de contexto com suporte a QoC, pois, de modo geral, as revisões de literatura existentes, embora analisem diversos aspectos, não discutem detalhadamente a forma como cada middleware lida com problemas relacionados ao provisionamento de QoC.

- A proposição de uma abordagem de middleware de inovadora, que incorpora conceitos e mecanismos de provisionamento, avaliação e monitoramento de qualidade de contexto, capaz de facilitar significativamente o desenvolvimento de aplicações de IoT que possuem requisitos de QoC.

- Uma metodologia de avaliação de middleware de contexto com suporte à QoC, que leva em consideração tanto aspectos quantitativos (que incluem a realização de experimentos) e qualitativos (que incluem provam de conceito e avaliação junto ao desenvolvedores de aplicações de IoT).

9.2 Trabalhos Futuros

Os próximos passos desta pesquisa vão em direção aos seguintes objetivos:

- Investigar o uso de ontologias específicas no domínio de IoT, de modo que seja possível incorporar ao M-Hub/CDDL mecanismos de divulgação, descoberta e assinatura de serviços de contexto que tratem problemas relacionados a possíveis ambiguidades na semântica dos dados de contexto e dos parâmetros de QoC.

- Investigar abordagens de monitoramento que combinem mecanismos baseados no processamento de eventos complexos (CEP) com mecanismos de inferência baseados no raciocínio sobre ontologias, a fim de incorporar ao M-Hub/CDDL uma abordagem de monitoramento híbrida que aproveite o melhor de cada estratégia.

9.3 Publicações

Ao longo do desenvolvimento desta pesquisa, o aluno de doutorado obteve algumas publicações em periódicos especializados e anais de eventos científicos, que são apresentados a seguir. As publicações diretamente relacionadas à tese de doutorado são aquelas em que o aluno é o autor principal. As indiretamente relacionadas são aquelas em que o aluno é co-autor.

9.3.1 Artigos de Periódicos

• **Gomes, B. T. P**; Muniz, L. C; da Silva e Silva, F. J; dos Santos, D. V.; Lopes, R. F; Coutinho, L. R.; Carvalho, R. O; Endler, Markus. A Middleware with Comprehensive Quality of Context Support for the Internet of Things Applications. Sensors (Basel), ISSN: 1424-8220, v. 17, nº 12, p. 1-45, 2017.

Qualis CAPES(2013-2016): A1 em Ciência da Computação e em Engenharia IV.

Situação: Publicado.

• Ribeiro Filho, José Daniel Pereira ; da Silva e Silva, Francisco José ; Coutinho, Luciano Reis ; **Gomes, Berto de Tácio Pereira**. MHARS: A mobile system for human activity recognition and inference of health situations in ambient assisted living. Journal of Applied Computing Research, ISSN: 2236-8434, v. 5, p. 44-58, 2016.

Qualis CAPES(2013-2016): B5 em Engenharias IV e C em Ciência da Computação.

Situação: Publicado.

• **Gomes, Berto**; Muniz, Luiz; da Silva e Silva, Francisco Jose; Rios, Luis Eduardo Talavera; Endler, Markus. A Comprehensive and Scalable Middleware for Ambient Assisted Living Based on Cloud Computing and IoT. Concurrency and Computation: Practice and Experience, ISSN: 1532-0634, 2017.

Qualis CAPES(2013-2016): A2 em Ciência da Computação.

Situação: Publicado.

• Ribeiro Filho, José Daniel Pereira ; da Silva e Silva, Francisco José ; Coutinho, LUCIANO REIS ; **Gomes, Berto de Tácio Pereira**. MHARS: A mobile system for human activity recognition and inference of health situations in ambient assisted living. Journal of Applied Computing Research, ISSN: 2236-8434, v. 5, p. 44-58, 2016.

Qualis CAPES(2013-2016): B5 em Engenharias IV e C em Ciência da Computação.

Situação: Publicado.

• Pinheiro, Dejailson Nascimento; Teles, Ariel Soares; **de Tácio Pereira Gomes, Berto**; da Silva e Silva, Francisco José. A Middleware for Developing Context-Aware Mobile Applications in Low-Cost Acquisition Devices. Lecture Notes in Electrical Engineering. : Springer Berlin Heidelberg, ISSN 1876-1100, v. 330, p. 419-424, 2015.

Qualis CAPES: C em Engenharias IV.

Situação: Publicado.

9.3.2 Trabalhos Completos em Anais de Eventos

• **Gomes, Berto**; Muniz, Luiz; da Silva e Silva, Francisco Jose; Rios, Luis Eduardo Talavera; Endler, Markus. A comprehensive cloud-based IoT software infrastructure for Ambient Assisted Living. In: Cloud Technologies and Applications (CloudTech), 2015 International Conference on , vol., no., pp.1-8, 2-4 June 2015.

Qualis Perfil CC SBC 2016: B4 .

Situação: Publicado.

• Ribeiro Filho, J. D. P. ; Silva, F. J. S. E. ; **Gomes, B. T. P.**; Coutinho, L. R. . A Movement Activity Recognition Pervasive System for Patient Monitoring in Ambient Assisted Living. In: ACM Symposium on Applied Computing, 2016, Pisa.

Qualis Perfil CC SBC 2016: A1.

Situação: Publicado.

• Ribeiro Filho, J. D. P. ; da Silva e Silva, Francisco José ; Luciano Reis Coutinho; **Berto de Tácio Pereira Gomes**. MHARS: Sistema Móvel de Reconhecimento de Atividades em Ambient Assisted Living. In: Anais do 7º Simpósio Brasileiro de Computação Ubíqua e Pervasiva - SBCUP 2015, 2015, Recife, Brasil.

Qualis Perfil CC SBC 2016: B4 .

Situação: Publicado.

• Oliveira Vasconcelos, R.; Talavera, L.; Vasconcelos, I.; Roriz, M.; Endler, M.; **de Tácio Pereira Gomes, B.**; da Silva e Silva, F.J. An Adaptive Middleware for Opportunistic Mobile Sensing,"in Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on , vol., no., pp.1-10, 10-12 June 2015.

Qualis Perfil CC SBC 2016: A2.

Situação: Publicado

Referências Bibliográficas

[1] Z. Abid, S. Chabridon, and D. Conan. A framework for quality of context management. In K. Rothermel, D. Fritsch, W. Blochinger, and F. Dürr, editors, *Quality of Context*, volume 5786 of *Lecture Notes in Computer Science*, pages 120–131. Springer Berlin Heidelberg, 2009.

[2] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 304–307, London, UK, UK, 1999. Springer-Verlag.

[3] B. Adams, D. Q. Phung, and S. Venkatesh. Sensing and using social context. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 5(2), 2008.

[4] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*, Bruges, Belgium 2013.

[5] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. *The VLDB Journal—The International Journal on Very Large Data Bases*, 15(2):121–142, 2006.

[6] J.-P. Arcangeli, A. Bouzeghoub, V. Camps, M.-F. Canut, S. Chabridon, D. Conan, T. Desprats, R. Laborde, E. Lavinal, S. Leriche, H. Maurel, A. Peninou, C. Taconet, and P. Zazraté. INCOME : multi-scale context management for the Internet of Things. In *AmI '12 : International Joint Conference on Ambient Intelligence*, volume 7683, pages 338–347, Pisa, Italy, Nov 2012. Springer.

[7] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, June 2007.

[8] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta. *A Survey of Middleware for Internet of Things*, pages 288–296. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[9] C. Batista, E. Cavalcate, G. Alves, and P. F. Pires. A metadata monitoring system for ubiquitous computing. In *Proc. of 6th Int. Conf. on Mobile Ubiquitous Computing, Systems, Services andTechnologies*, pages 60–66, 2012.

[10] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini. A survey of context data distribution for mobile ubiquitous systems. *ACM Comput. Surv.*, 44(4):24:1–24:45, sep 2012.

[11] G. Blair, D. Schmidt, and C. Taconet. Middleware for internet distribution in the context of cloud computing and the Internet of Things. *Annals of Telecommunications*, 71, Apr 2016.

[12] C. Bolchini, C. A. Curino, G. Orsi, E. Quintarelli, R. Rossato, F. A. Schreiber, and L. Tanca. And what can context do for data? *Commun. ACM*, 52(11):136–140, nov 2009.

[13] E. Borgia. The INTERNET of THINGs vision: Key features, applications and open issues. *Computer Communications*, 54(0):1 – 31, 2014.

[14] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani. The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems. *Softw. Pract. Exper.*, 36(11-12):1257–1284, sep 2006.

[15] R. Bruns, J. Dunkel, S. Lier, and H. Masbruch. Ds-epl: Domain-specific event processing language. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, pages 83–94, New York, NY, USA, 2014. ACM.

[16] Y. Bu, T. Gu, X. Tao, J. Li, S. Chen, and J. Lu. Managing quality of context in pervasive computing. In *Quality Software, 2006. QSIC 2006. Sixth International Conference on*, pages 193–200, Oct 2006.

[17] T. Buchholz, A. Küper, and M. Schiffers. Quality of context: What it is and why we need it. In *In Proceedings of the 10th Workshop of the OpenView University Association: (HPOVUA)*, 2003.

[18] H. E. Byen and K. Cheverst. Utilizing context history to provide dynamic adaptations. *Applied Artificial Intelligence*, 18(6):533–548, 2004.

[19] K. S. Candan, H. Liu, and R. Suvarna. Resource description framework: Metadata and its applications. *SIGKDD Explor. Newsl.*, 3(1):6–19, July 2001.

[20] P. Carbone, S. Ewen, S. Haridi, A. Katsifodimos, V. Markl, and K. Tzoumas. Apache Flink: Unified Stream and Batch Processing in a Single Engine. *Data Engineering*, pages 28–38, 2015.

[21] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous Dataflow Processing. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, page 668, New York, NY, USA, 2003. ACM.

[22] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[23] A. Cockburn. *Agile Software Development*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[24] D. Conan, R. Rouvoy, and L. Seinturier. *Scalable Processing of Context Information with COSMOS*, pages 210–224. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[25] D. J. Cook and S. K. Das. Review: Pervasive computing at scale: Transforming the state of the art. *Pervasive Mob. Comput.*, 8(1):22–35, feb 2012.

[26] A. Corradi, M. Fanelli, and L. Foschini. Implementing a scalable context-aware middleware. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 868–874, July 2009.

[27] A. Corradi, M. Fanelli, and L. Foschini. Adaptive context data distribution with guaranteed quality for mobile environments. In *IEEE International Symposium on Wireless Pervasive Computing (ISWPC'10)*, pages 373–380, 2010.

[28] M. F. Corradi, Antonio and L. Foschini. towards adaptive and scalable context aware middleware. *Technological Innovations in Adaptive and Dependable Systems: Advancing Models and Concept*, pages 21–37, 2012.

[29] L. Courtrai, F. Guidec, N. Le Sommer, and Y. Mahéo. Resource management for parallel adaptive components. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, IPDPS '03, pages 134.2–, Washington, DC, USA, 2003. IEEE Computer Society.

[30] G. Cugola and H.-A. Jacobsen. Using publish/subscribe middleware for mobile systems. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(2):25–33, oct 2002.

[31] R. da Rocha and M. Endler. Foundations of context management in distributed and dynamic environments. In *Context Management for Distributed and Dynamic Context-Aware Computing*, SpringerBriefs in Computer Science, pages 9–27. Springer London, 2012.

[32] L. David, R. Vasconcelos, L. Alves, R. André, and M. Endler. A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes. *Journal of Internet Services and Applications (JISA)*, 4(1):16, 2013.

[33] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1).

[34] A. K. Dey. *Providing Architectural Support for Building Context-aware Applications*. PhD thesis, Atlanta, GA, USA, 2000. AAI9994400.

[35] M. Eggum. Smartphone Assisted Complex Event Processing. Masther thesis, University of Oslo, 2014.

[36] Embedded101. Developing M2M internet-of-things devices on windows embedded platforms. `http://goo.gl/DKlcFc`. Accessed: 2015-01-30.

[37] C. Emmanouilidis, R.-A. Koutsiamanis, and A. Tasidou. Mobile guides: Taxonomy of architectures, context awareness, technologies and applications. *Journal of Network and Computer Applications*, 36(1):103–125, 2013.

[38] EsperTech. Esper - Complex Event Processing. `http://www.espertech.com/esper/`. Accessed: 2017-05-24.

[39] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.

[40] M. Fanelli. *Middleware for quality-based context distribution in mobile systems*. PhD thesis, alma, Maggio 2012.

[41] J. D. P. R. Filho, F. J. da Silva e Silva, L. R. Coutinho, and B. d. T. P. Gomes. MHARS: A mobile system for human activity recognition and inference of health situations in Ambient Assisted Living LIVING. *Mobile Information Systems*, 5(1):48–55, 2016.

[42] J. D. P. R. Filho, F. J. da Silva e Silva, L. R. Coutinho, B. de Tácio Pereira Gomes, and M. Endler. A movement activity recognition pervasive system for patient monitoring in Ambient Assisted Living LIVING. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 155–161, New York, NY, USA, 2016. ACM.

[43] M. P. Forum. MPI: A message-passing interface standard. Technical report, Knoxville, TN, USA, 1994.

[44] M. Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 1st edition, 2010.

[45] A. Gaddah and T. Kunz. A survey of middleware paradigms for mobile computing. Technical report, Carleton University, 2003.

[46] A. M. Gianpaolo Cugola. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(2), 2012.

[47] B. Gomes, L. Muniz, F. J. da Silva e Silva, L. E. T. Rios, and M. Endler. A comprehensive cloud-based iot software infrastructure for Ambient Assisted Living LIVING. In *Cloud Technologies and Applications (CloudTech), 2015 International Conference on*, pages 1–8, June 2015.

[48] B. d. T. P. Gomes, L. C. M. Muniz, F. J. da Silva e Silva, L. E. T. Ríos, and M. Endler. A comprehensive and scalable middleware for Ambient Assisted Living LIVING based on cloud computing and Internet of Things. *Concurrency and Computation: Practice and Experience*, 29:1–19, 2017.

[49] P. D. Gray and D. Salber. Modelling and using sensed context information in the design of interactive applications. In *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction*, EHCI '01, pages 317–336, London, UK, UK, 2001. Springer-Verlag.

[50] A. Held, S. Buchholz, and A. Schill. Modeling of context information for pervasive computing applications. In *Procceding of the World Multiconference on Systemics, Cybernetics and Informatics*. Springer, 2002.

[51] K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 33–37, March 2004.

[52] J. Hong, E. Suh, and S. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509 – 8522, 2009.

[53] M. Huebscher and J. McCann. A learning model for trustworthiness of context-awareness services. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 120–124, March 2005.

[54] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. MQTT-S: A publish/subscribe protocol for wireless sensor networks. In *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pages 791–798, Jan 2008.

[55] S. Kent. Model driven engineering. In *Proceedings of the Third International Conference on Integrated Formal Methods*, IFM '02, pages 286–298, London, UK, UK, 2002. Springer-Verlag.

[56] Y. Kim and K. Lee. A Quality Measurement Method of Context Information in Ubiquitous Environments. In *Hybrid Information Technology, 2006. ICHIT '06. International Conference on*, volume 2, pages 576–581, Nov 2006.

[57] M. Krause and I. Hochstatter. Challenges in modelling and using quality of context (QoC). In T. Magedanz, A. Karmouch, S. Pierre, and I. Venieris, editors, *Mobility Aware Technologies and Applications*, volume 3744 of *Lecture Notes in Computer Science*, pages 324–333. Springer Berlin Heidelberg, 2005.

[58] S. Kudyba. *Big Data, Mining, and Analytics*. 2014.

[59] M. Leclercq, A. E. Ozcan, V. Quema, and J. B. Stefani. Supporting heterogeneous architecture descriptions in an extensible toolset. In *29th International Conference on Software Engineering (ICSE'07)*, pages 209–219, May 2007.

[60] M. Leclercq, V. Quema, and J.-B. Stefani. Dream: A component framework for constructing resource-aware, configurable middleware. *IEEE Distributed Systems Online*, 6(9):1, sep 2005.

[61] S. Lee, H. Kim, D.-K. Hong, and H. Ju. Correlation analysis of mqtt loss and delay according to qos level. In *Information Networking (ICOIN), 2013 International Conference on*, pages 714–717, Jan 2013.

[62] F. Li, S. Sehic, and S. Dustdar. Copal: An adaptive approach to context provisioning. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE 6th International Conference on*, pages 286–293, Oct 2010.

[63] X. Li, M. Eckert, J.-F. Martinez, and G. Rubio. Context aware middleware architectures: Survey and challenges. *Sensors*, 15(8):20570–20607, 2015.

[64] Y. Li and L. Feng. A quality-aware context middleware specification for context-aware computing. In *Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International*, volume 2, pages 206–211, July 2009.

[65] L. Lim and D. Conan. Distributed event-based system with multiscoping for multiscalability. In *Proceedings of the 9th Workshop on Middleware for Next Generation Internet Computing*, MW4NG '14, pages 3:1–3:6, New York, NY, USA, 2014. ACM.

[66] L. Lim, P. Marie, D. Conan, S. Chabridon, T. Desprats, and A. Manzoor. Enhancing context data distribution for the Internet of Things using QoC-awareness and attribute-based access control. *Annales des Télécommunications*, 71:121–132, 2016.

[67] F. Lopes, F. C. Delicato, T. Batista, E. Cavalcante, T. Pereira, P. F. Pires, P. Ferreira, and R. Mendes. Opencopi: middleware integration for ubiquitous computing. *International Journal of Parallel, Emergent and Distributed Systems*, 29(2):178–212, 2014.

[68] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[69] J. E. Luzuriaga, M. Perez, P. Boronat, J. C. Cano, C. Calafate, and P. Manzoni. Improving MQTT data delivery in mobile scenarios: Results from a realistic testbed. *Mobile Information Systems*, 2016:178–212, 2016.

[70] J. Madhusudanan and V. P. Venkatesan. Metrics for evaluating pervasive middleware. *International Journal of Intelligent Systems and Applications*, 6(1).

[71] A. Manzoor. *Quality of context in pervasive systems: models, techniques, and applications*. na, 2010.

[72] A. Manzoor, H.-L. Truong, and S. Dustdar. On the evaluation of quality of context. In *Smart Sensing and Context*, pages 140–153. Springer, 2008.

[73] A. Manzoor, H.-L. Truong, and S. Dustdar. Quality of context: models and applications for context-aware systems in pervasive environments. *The Knowledge Engineering Review*, 29:154–170, 3 2014.

[74] P. Marie, T. Desprats, S. Chabridon, M. Sibilla, and C. Taconet. From ambient sensing to iot-based context computing: An open framework for end to end QoC management. *Sensors*, 15(6):14180–14206, 2015.

[75] P. Marie, L. Lim, A. Manzoor, S. Chabridon, D. Conan, and T. Desprats. QoC-aware context data distribution in the Internet of Things. In *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT*, M4IOT '14, pages 13–18, New York, NY, USA, 2014. ACM.

[76] J. Martin, Y. Fu, N. Wourms, and T. Shaw. Characterizing netflix bandwidth consumption. In *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, pages 230–235, Jan 2013.

[77] J. McNaull, J. C. Augusto, M. Mulvenna, and P. McCullagh. Data and information quality issues in Ambient Assisted Living systems. *J. Data and Information Quality*, 4(1):4:1–4:15, oct 2012.

[78] M. Memon, S. R. Wagner, C. F. Pedersen, F. H. A. Beevi, and F. O. Hansen. Ambient Assisted Living LIVING healthcare frameworks, platforms, standards, and quality attributes. *Sensors*, 14(3).

[79] M. Miraoui, C. Tadj, J. Fattahi, and C. B. Amar. Dynamic context-aware and limited resources-aware service adaptation for pervasive computing. *Adv. Soft. Eng.*, 2011:7:7–7:7, jan 2011.

[80] M. Musolesi. Designing a context-aware middleware for asynchronous communication in mobile ad hoc environments. In *Proceedings of the 1st International Doctoral Symposium on Middleware*, DSM '04, pages 304–308, New York, NY, USA, 2004. ACM.

[81] D. C. Nazario. *CUIDA - Um modelo de conhecimento de Qualidade de Contexto aplicado a ambientes ubíquos internos em domicílios assistidos*. PhD theses, Universidade Federal de Santa Catarina, March 2015.

[82] D. C. Nazario, I. V. B. Tromel, M. A. R. Dantas, and J. L. Todesco. Toward assessing quality of context parameters in a ubiquitous assisted environment. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–6, June 2014.

[83] OASIS. Foundational IoT messaging protocol, MQTT, becomes international OASIS standard. `https://goo.gl/TKF4RA`. Accessed: 2015-01-30.

[84] A. C. of Sports Medicine, M. Whaley, P. Brubaker, R. Otto, and L. Armstrong. *ACSM's Guidelines for Exercise Testing and Prescription*. Lippincott Williams & Wilkins, 2006.

[85] G. Pardo-Castellote. OMG data-distribution service: Architectural overview. In *Proceedings of the 2003 IEEE Conference on Military Communications - Volume I*, MILCOM'03, pages 242–247, Washington, DC, USA, 2003. IEEE Computer Society.

[86] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the Internet of Things: A survey. *Communications Surveys Tutorials, IEEE*, 16:414–454, May 2014.

[87] R. M. Pessoa. Infraware: Um middleware de suporte a aplicações sensíveis ao contexto. Master's thesis, Universidade Federal do Espírito Santo (UFES), Vitória, ES, Brazil, 2006.

[88] D. Preuveneers and Y. Berbers. Architectural backpropagation support for managing ambiguous context in smart environments. In C. Stephanidis, editor, *Universal Access in Human-Computer Interaction. Ambient Interaction*, volume 4555 of *Lecture Notes in Computer Science*, pages 178–187. Springer Berlin Heidelberg, 2007.

[89] M. Proctor. Drools: A Rule Engine for Complex Event Processing. In *Proceedings of the 4th International Conference on Applications of Graph Transformations with Industrial Relevance*, AGTIVE'11, pages 2–2, Berlin, Heidelberg, 2012. Springer-Verlag.

[90] V. Raychoudhury, J. Cao, M. Kumar, and D. Zhang. Middleware for pervasive computing: A survey. *Pervasive and Mobile Computing*, 9(2):177 – 200, 2013. Special Section: Mobile Interactions with the Real World.

[91] L. Rios, M. Endler, I. Vasconcelos, M. C. R. Vasconcelos, and F. S. e Silva. The mobile hub concept: Enabling applications for the internet of mobile things. In *12th IEEE Workshop on Managing Ubiquitous Communications and Services (MUCS 2015), IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, March 2015.

[92] M. Roriz Junior. *DG2CEP : An On-line Algorithm for Real-time Detection of Spatial Clusters from Large Data Streams through Complex Event Processing*. PhD thesis, Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro.

[93] C. G. Sahu and D. S. Adane. A survey on context-aware middleware. *International Journal of Advanced Research in Computer and Communication Engineering (IJARCE)*, 4:650–656, May 2015.

[94] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8:10–17, Aug 2001.

[95] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[96] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, Sept 1994.

[97] D. Schuster, A. Rosi, M. Mamei, T. Springer, M. Endler, and F. Zambonelli. Pervasive social context - taxonomy and survey. *ACM Transactions on Intelligent Systems and Technology*, 9:1–22, 2012.

[98] S. Sehic, F. Li, and S. Dustdar. COPAL-ML: A macro language for rapid development of context-aware applications in wireless sensor networks. In *Proceedings of the 2Nd Workshop on Software Engineering for Sensor Network Applications*, SESENA '11, pages 1–6, New York, NY, USA, 2011. ACM.

[99] K. Sheikh, M. Wegdam, and M. van Sinderen. Middleware support for quality of context in pervasive context-aware systems. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 461–466, March 2007.

[100] K. Sheikh, M. Wegdam, and M. van Sinderen. Quality-of-context and its use for protecting privacy in context aware systems. *Journal of software*, 3(3):83–93, March 2008.

[101] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[102] M. J. van Sinderen, A. T. van Halteren, M. Wegdam, H. B. Meeuwissen, and E. H. Eertink. Supporting context-aware mobile applications: an infrastructure approach. *IEEE Communications Magazine*, 44(9):96–104, Sept 2006.

[103] R. Vasconcelos, L. Silva, and M. Endler. Towards efficient group management and communication for large-scale mobile applications. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 551–556. IEEE Computer Society, 2014.

[104] R. O. Vasconcelos, M. Endler, B. d. T. P. Gomes, and F. J. d. S. e Silva. Design and Evaluation of an Autonomous Load Balancing System for Mobile Data Stream Processing Based on a Data Centric Publish Subscribe Approach. *International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, 5(2), 2014.

[105] R. O. Vasconcelos, L. Talavera, I. Vasconcelos, M. Roriz, M. Endler, B. d. T. P. Gomes, and F. J. d. S. e. Silva. An adaptive middleware for opportunistic mobile sensing. In *2015 International Conference on Distributed Computing in Sensor Systems*, pages 1–10, June 2015.

[106] R. O. Vasconcelos, I. Vasconcelos, and M. Endler. Dynamic and coordinated software reconfiguration in distributed data stream systems. *J. Internet Services and Applications*, 7(1):8:1–8:21, 2016.

[107] V. Vieira, P. Tedesco, and A. C. Salgado. Designing context-sensitive systems: An integrated approach. *Expert Systems with Applications*, 38(2):1119–1138, feb 2011. Intelligent Collaboration and Design.

[108] R. S. Wazlawick. Uma Reflexão sobre a Pesquisa em Ciência da Computação à Luz da Classificação das Ciências e do Método Científico. *Revista de Sistemas de Informação da FSMA*, 6:3–10, 2010.

[109] M. Weiser. The computer for the twenty-first century. *SIGMOBILE Mobile Computing and Communications Review.*, 9:94–100, Sep 1991.

[110] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[111] Yanlei Diao Neil Immerman and D. Gyllstrom. SASE+: An Agile Language for Kleene Closure over Event Streams. Technical report, Department of Computer Science, University of Massachusetts Amherst, 2007.

[112] O. Yurur, C. H. Liu, Z. Sheng, V. C. M. Leung, W. Moreno, and K. K. Leung. Context-awareness for mobile sensing: A survey and future directions. *IEEE Communications Surveys Tutorials*, 18(1):68–93, December 2014.

[113] A. Zimmermann, A. Lorenz, and R. Oppermann. An operational definition of context. In *Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context*, CONTEXT'07, pages 558–571, Berlin, Heidelberg, 2007. Springer-Verlag.

A Questionário de Avaliação Qualitativa da Abordagem de Middleware M-Hub/CDDL

1ª) *Avalie se o middleware o M-Hub/CDDL é fácil de usar, ou seja, se a instalação, configuração e programação do middleware são facilmente realizadas. Faça comentários que achar pertinente.*
(0) não, o middleware **não é fácil de usar**.
(1) sim, o middleware é **um pouco fácil de usar**.
(2) sim, o middleware é **consideravelmente fácil de usar**.
(3) sim, o middleware é **muito fácil de usar**.

Respostas: Todos os participantes deram nota 3, ou seja, por unanimidade, eles consideraram que o middleware, no que se refere à instalação, configuração e programação é muito fácil de usar. Nos comentários, um dos participantes destacou que o middleware facilita muito o desenvolvimento de aplicações complexas porque requer poucas linhas de código. Outro participante considerou que o fato dele já ter conhecimento de programação Java/Android também ajudou bastante na tarefa.

2ª) *Avalie se interfaces de programação do M-Hub/CDDL são adequadas, ou seja, se as assinaturas dos métodos são claras e as abstrações de programação providas pelo middleware são adequadas para o domínio do desenvolvimento de aplicações de IoMT. Se não são adequadas e/ou suficientes, o que você sugere de mudança?*

(0) não, as interfaces e abstrações de programação **não são adequadas**.
(1) sim, as interfaces e abstrações de programação **são um pouco adequadas**.
(2) sim, as interfaces e abstrações de programação **são consideravelmente adequadas**.
(3) sim, as interfaces e abstrações de programação **são muito adequadas**.

Respostas: Todos os participantes deram nota 3, ou seja, por unanimidade, eles consideraram que as interfaces de programação do middleware são muito adequadas. Nos comentários, alguns participantes afirmaram que as assinaturas dos métodos são autoexplicativas e que isso ajudou bastante na compreensão da finalidade dos mesmo.

3ª) *Avalie se as abstrações e interfaces de programação disponibilizadas pelo middleware evitam que o desenvolvedor tenha que conhecer os detalhes referentes às funcionalidades de descoberta,*

conexão e obtenção de dados a partir de sensores heterogêneos, ou seja, se o middleware abstrai a complexidade de interação com objetos inteligentes. Faça comentários que achar pertinente.

(0) não, o middleware **não abstrai** a heterogeneidade e complexidade
(1) sim, o middleware **abstrai um pouco** a heterogeneidade e complexidade.
(2) sim, o middleware **abstrai consideravelmente** a heterogeneidade e complexidade.
(3) sim, o middleware **abstrai muito** a heterogeneidade e complexidade.

Respostas: Todos os participantes deram nota 3, ou seja, por unanimidade, eles consideraram que as interfaces disponibilizadas pelo M-Hub/CDDL abstraem muito a heterogeneidade e a complexidade. Um participante comentou a necessidade de incluir o nome dos sensores internos na documentação do middleware.

4ª) *Avalie se a utilização do M-Hub/CDDL facilita a distribuição de dados contexto, tanto local como remota, ou seja, se o middleware permite configurar, de maneira simples, diversos fluxos de dados de contexto entre produtores e consumidores. Faça comentários que achar pertinente.*

(0) não, o middleware **não facilita** a distribuição de dados de contexto.
(1) sim, o middleware **facilita um pouco** a distribuição de dados de contexto.
(2) sim, o middleware **facilita consideravelmente** a distribuição de dados de contexto.
(3) sim, o middleware **facilita muito** a distribuição de dados de contexto.

Respostas: Todos os participantes deram nota 3, ou seja, por unanimidade, eles consideraram que M-Hub/CDDL facilita muito a distribuição dos dados de contexto. Um dos participantes comentou que a implementação é facilitada pelo fato das configurações de distribuição de produtores e consumidores serem simétricas.

5ª) *Avalie se a expressividade da linguagem de consulta disponibilizada pelo M-Hub/CDDL atende os requisitos da aplicação. Em que aspectos você considera que o mecanismo de descoberta de serviços oferecido pelo middleware pode ser melhorado?*

(0) não, a linguagem de consulta **não atende** a requisitos de descoberta.
(1) sim, a linguagem de consulta **atende** um pouco a requisitos de descoberta.
(2) sim, a linguagem de consulta **atende consideravelmente** a requisitos de descoberta.
(3) sim, a linguagem de consulta **atende muito bem** a requisitos de descoberta.

Respostas: A avaliação da linguagem de consulta era facultativa, devido ao fato de que o sistema proposto no estudo de caso poderia ser implementado sem utilizar o mecanismo de consulta, desde que tópicos nos quais os provedores de serviços

de contexto publicavam os dados já fossem conhecidos pela aplicação consumidora. Mesmo assim, os participantes foram encorajados a testar o mecanismo de consulta e avaliá-lo. Metade dos participantes deu nota 3 e metade deles deu nota 2, ou seja, no geral, eles consideraram que linguagem de consulta do M-Hub/CDDL atende, consideravelmente ou muito bem, a requisitos de descoberta de serviço de contexto.

6ª) *Avalie se a expressividade da linguagem de monitoramento atende os requisitos da aplicação. Em que aspectos você considera que o mecanismo de detecção de variações de contexto e de QoC oferecido pelo middleware pode ser melhorado?*

(0) não, a linguagem **não atende** os requisitos de monitoramento.
(1) sim, a linguagem **atende um pouco** os requisitos de monitoramento.
(2) sim, a linguagem **atende consideravelmente bem** os requisitos de monitoramento.
(3) sim, a linguagem **atende muito bem** os requisitos de monitoramento.

Respostas: Todos os participantes deram nota 3, ou seja, por unanimidade eles consideraram que a linguagem oferecida pelo middleware para detectar variações de contexto e de QoC atendeu muito bem os requisitos de monitoramento da aplicação.

7ª) *Avalie se a expressividade da linguagem utilizada para a filtragem de publicação e subscrição de dados disponibilizada pelo M-Hub/CDDL atende os requisitos da aplicação. Em que aspectos você considera que esse mecanismo pode ser melhorado?*

(0) não, a linguagem **não atende** os requisitos de filtragem.
(1) sim, a linguagem **atende um pouco** os requisitos de filtragem.
(2) sim, a linguagem **atende consideravelmente bem** os requisitos de filtragem.
(3) sim, a linguagem **atende muito bem** os requisitos de filtragem.

Respostas: Todos os participantes deram nota 3, ou seja, por unanimidade, eles consideraram que linguagem de filtragem de publicações e subscrições de dados, disponibilizada pelo M-Hub/CDDL, atendeu muito bem os requisitos da aplicação.

8ª) *Avalie se o M-Hub/CDDL executou de maneira estável, ou seja, diga se durante a execução do middleware foi notado algum tipo de comportamento anormal, que prejudicasse o funcionamento da aplicação. Faça os comentários que achar pertinentes.*

(0) percebeu-se **muita instabilidade** na execução do middleware.
(1) percebeu-se **razoável instabilidade** na execução do middleware.

(2) percebeu-se **pouca instabilidade** na execução do middleware.

(3) percebeu-se **nenhuma instabilidade** na execução do middleware.

Respostas: Um dos participantes deu nota 3, enquanto que três participantes deram nota 2, ou seja, a maioria deles percebeu pouca instabilidade na execução do middleware. Uma das justificativas dos participantes consiste no fato de que, ao tentar controlar o ciclo de vida da aplicação (por exemplo, minimizar a aplicação ou alterar as janelas), eles perceberam um pouco de instabilidade ao tentar encerrar a execução do serviço de aquisição de dados do middleware e, logo em seguida, tentar reiniciá-lo, como forma de tentar economizar recursos enquanto a aplicação estivesse inativa. Essa instabilidade fazia com que aplicação encerrasse abruptamente, em alguns casos. A partir desse *feedback* dos usuários, investigou-se o problema, que foi solucionado posteriormente à avaliação. Outro problema relatado pelos usuários está relacionado ao fato de que, em versão anterior à apresentada na Seção 4, o middleware iniciava a aquisição de dados a partir de todos os sensores disponíveis automaticamente, a fim de permitir a descoberta e interação oportunista com objetos inteligentes. Embora útil em cenários de IoMT, foi detectado que, em dispositivos móveis com muitos sensores internos, ou ainda sensores que, por padrão, produzem dados com frequências muito elevadas (por exemplo, acelerômetro e giroscópio), esse modo de inicialização do gateway fazia com que a aplicação travasse, ou se comportasse da maneira instável, em situações em que não havia recursos computacionais suficientes para que o S2PA pudesse receber e processar o grande fluxo de dados produzidos pelos sensores. A partir desse *feedback* dos usuários, o M-Hub/CDDL foi alterado, de tal forma que a inicialização padrão dos sensores passou a ser por demanda, ou seja, o middleware inicialmente não habilita nenhum sensor a enviar dados para o S2PA. Nesse caso, o recebimento de dados deve ser habilitado dinâmica e individualmente por sensor, ou por grupo de sensores, mediante necessidade do gateway. Mesmo assim, o modo de descoberta e aquisição oportunista, que habilita a interação do gateway com todos os sensores disponíveis, ainda está presente no middleware e pode ser utilizado. Cabe explicar que, uma vez que o problema foi relatado pelos participantes antes da aplicação do questionário de avaliação, a alteração do middleware visando solucioná-lo foi realizada logo na segunda semana de treinamento, o que gerou a liberação de uma atualização do M-Hub/CDDL. Graças a isso, logo nas semanas seguintes, os desenvolvedores puderam fazer uso da nova versão. Mesmo assim, a fim de preservar

o rigor da avaliação, os participantes foram encorajados a relatar no questionário todos os problemas encontrados no middleware, desde a primeira semana de treinamento.

9ª) *Avalie o grau de ocorrência de erros (bugs) durante a execução do middleware.*

(0) houve **grande** ocorrência de erros (*bugs*) durante a execução do middleware.
(1) houve **considerável** ocorrência de erros (*bugs*) durante a execução do middleware.
(2) houve **pouca** ocorrência de erros (*bugs*) durante a execução do middleware.
(3) **não houve** quaisquer ocorrência de erros (bugs) durante a execução do middleware.

Respostas: Metade dos participantes deu nota 3 e metade deles deu nota 2. Os participantes que deram nota 3 justificaram que, para efeitos de avaliação, levaram em consideração a última versão do middleware disponibilizada antes do final do período de testes, pois, nessas versão, os poucos *bugs* reportados já haviam sido corrigidos.

10ª) *Avalie se o middleware atendeu adequadamente os requisitos de QoC da aplicação.*

(0) não, o middleware **não atendeu** os requisitos de QoC da aplicação.
(1) sim, o middleware **atendeu um pouco** os requisitos de QoC da aplicação.
(2) sim, o middleware **atendeu consideravelmente** os requisitos de QoC da aplicação.
(3) sim, o middleware **atendeu muito bem** os requisitos de QoC da aplicação.

Respostas: Metade dos participantes deu nota 2, enquanto que a outra metade deu nota 3, ou seja, no geral, eles consideraram que M-Hub/CDDL atendeu, consideravelmente ou muito bem, os requisitos de QoC da aplicação. Dentre os participantes que deram nota 2, um deles argumentou que o sensor de velocidade não forneceu a acurácia. Talvez isso tenha prejudicado um pouco a avaliação neste quesito. Justifica-se que o M-Hub/CDDL só consegue obter a acurácia em situações em que esse parâmetro de QoC é fornecido pelo sensor. No estudo de caso, a velocidade era fornecida por um sensor virtual que calculava essa informação com base em dados de aceleração. Embora o sensor de aceleração fornecesse a acurácia, não se pode atribuir tal acurácia aos dados de velocidade, pois são informações de contexto diferentes. Logo, trata-se de um problema que o middleware não consegue contornar facilmente.

11ª) *Avalie se os parâmetros de QoC utilizados pela aplicação se comportaram conforme o esperado. Quais alterações na implementação/funcionamento desses parâmetros você sugere?*

(0) **nenhum** dos parâmetros de QoC utilizados se comportaram conforme o esperado.
(1) **poucos** dos parâmetros de QoC utilizados se comportaram conforme o esperado.

(2) **a maioria** dos parâmetros de QoC utilizados se comportaram conforme o esperado.
(3) **todos** os parâmetros de QoC utilizados se comportaram conforme o esperado.

Respostas: Três participantes deram nota 3, enquanto que um participante deu nota 2, ou seja, no geral, a maioria dos avaliadores considerou que todos os parâmetros de QoC utilizados se comportaram corretamente, atendendo o que era esperado. Um dos participantes comentou que esperava que o histórico tivesse filtros que possibilitassem uma consulta aos dados armazenados de forma mais específica, citando como exemplo uma pesquisa de dados de contexto com base nos tópicos assinados pelos subscritores.

12ª) *Você considera que os parâmetros de QoC disponibilizadas pelo middleware são úteis para o desenvolvimento de outros tipos de aplicações de IoT/IoMT. Se sim, para quais outras aplicações de IoT/IoMT você acha que o M-Hub/CDDL é capaz de satisfazer requisitos de QoC?*

(0) não, os parâmetros fornecidos **não são úteis para outras aplicações**.
(1) sim, os parâmetros fornecidos **são um pouco úteis para outras aplicações**.
(2) sim, os parâmetros fornecidos **são consideravelmente úteis para outras aplicações**.
(3) sim, os parâmetros fornecidos **são muito úteis para outras aplicações**.

Respostas: Três participantes deram nota 3, enquanto que um participante deu nota 2, ou seja, no geral, a maioria dos avaliadores considerou que os parâmetros de QoC fornecidos pelo M-Hub/CDDL são muito úteis para outras aplicações de IoT. Nos comentários, por exemplo, foi mencionado que o middleware poderia atender requisitos de aplicações de IoT nas áreas da saúde e gestão de tráfego.

13ª) *Avalie se os parâmetros de QoC disponibilizados pelo M-Hub/CDDL são suficientes para o desenvolvimento de uma grande variedade aplicações de IoT? Quais parâmetros de qualidade da informação ou distribuição você sugere que sejam adicionados ou removidos do middleware?*

(0) A quantidade de parâmetros de QoC disponibilizada pelo middleware **é irrisória** para o desenvolvimento de uma grande variedade de aplicações de IoT.
(1) A quantidade de parâmetros de QoC disponibilizado pelo middleware **é pouca** para o desenvolvimento de uma grande variedade de aplicações de IoT.
(2) A quantidade de parâmetros de QoC disponibilizada pelo middleware **é suficiente** para o desenvolvimento de uma grande variedade de aplicações de IoT.
(3) A quantidade de parâmetros de QoC disponibilizada pelo middleware **é mais que suficiente** para o desenvolvimento de uma grande variedade de aplicações de IoT.

Respostas: Três participantes deram nota 3, enquanto que um participante deu nota 2, ou seja, no geral, a maioria dos avaliadores considerou que os parâmetros de QoC fornecidos pelo M-Hub/CDDL são relevantes para um grande número de aplicações de IoT. A única sugestão de adição de parâmetro de QoC ao middleware foi referente ao parâmetro **probabilidade de exatidão**, relacionado à qualidade da informação.

14ª) *Avalie se a documentação on-line do M-Hub/CDDL ajudou na utilização do middleware proposto. Em que aspectos você considera que essa documentação pode ser melhorada?*

(0) não, a documentação *on-line* disponibilizada para o aprendizado do M-Hub/CDDL **não ajudou em nada** na utilização do middleware, **exigindo total reformulação**.
(1) sim, a documentação *on-line* disponibilizada para o aprendizado do M-Hub/CDDL **ajudou um pouco** na utilização do middleware, mas **ainda requer muitas melhorias**.
(2) sim, a documentação *on-line* disponibilizada para o aprendizado do M-Hub/CDDL **ajudou consideravelmente** na utilização do middleware, **requerendo poucos ajustes**.
(3) sim, a documentação *on-line* disponibilizada para o aprendizado do M-Hub/CDDL **ajudou muito** na utilização do middleware, **não necessitando de quaisquer ajustes**.

Respostas: Dois participantes deram nota 1 e dois participantes deram nota 2, ou seja, pelo menos a metade dos avaliadores considerou que, embora a documentação fornecida tenha contribuído para o aprendizado do middleware, esta ainda requer muitas melhorias. Nos comentários, um dos participantes argumentou que, após a liberação da última atualização do middleware, alguns códigos de exemplo da documentação fornecida estavam um pouco diferentes em relação à API da versão mais atual, o que deixou alguns usuários um pouco confusos. Apesar disso, um dos desenvolvedores comentou que, embora a documentação precise melhorar, ele não teve prejuízo significativo no aprendizado no uso do middleware, graças à clareza das assinaturas dos métodos e materiais complementares que foram disponibilizados.

15ª) *Avalie se as mensagens de erro utilizadas pelo M-Hub/CDDL no tratamento de exceções são facilmente compreendidas. O que você considera que pode ser melhorado no middleware em relação a este aspecto? Faça comentários que achar pertinente.*

(0) **nenhuma** das mensagens de erro foi facilmente compreendida.
(1) **poucas** mensagens de erro foram facilmente compreendidas.
(2) **a maioria** das mensagens de erro foi facilmente compreendida.
(3) **todas** as mensagens de erro foram facilmente compreendidas.

Respostas: Dois participantes deram nota 1 e dois participantes deram nota 2, ou seja, pelo menos a metade dos participantes considerou que a maioria das mensagens de erro ainda precisa ter a semântica aprimorada, a fim de permitir que o programador compreenda mais facilmente as exceções tratadas, enquanto que a outra metade dos participantes considerou que, pelo menos, a maioria das mensagens de erro empregadas no tratamento de exceções foi facilmente compreendida. A principal sugestão dos participantes foi adicionar à documentação de software/código do M-Hub/CDDL explicações mais detalhadas sobre as exceções tratadas pelo middleware.

16ª) *Avalie se o treinamento (aulas presenciais, sessões tira-dúvidas e videoaulas) do M-Hub/CDDL foi útil para o aprendizado do middleware. Faça sugestões de melhorias.*

(0) não, o treinamento **não foi nem um pouco útil** para o aprendizado do middleware.
(1) sim, o treinamento **foi um pouco útil** para o aprendizado do middleware.
(2) sim, o treinamento **foi consideravelmente útil** para o aprendizado do middleware.
(3) sim, o treinamento **foi muito útil** para o aprendizado middleware.

Respostas: Todos os participantes da avaliação deram nota 3, ou seja, por unanimidade, eles consideraram que o treinamento realizado foi muito útil para o aprendizado do M-Hub/CDDL. Nos comentários, um deles considerou o conteúdo do treinamento como suficiente e bem dimensionado para o aprendizado de uso do middleware. Outro participante sugeriu que treinamentos futuros sejam mais práticos.

17ª) *Avalie se o fórum de discussão on-line contribuiu para o aprendizado do M-Hub/CDDL. Que outras ferramentas você acha que poderiam ter sido usadas no treinamento do middleware?*

(0) não, o fórum **não contribuiu nem um pouco** para o aprendizado do middleware.
(1) sim, o fórum **contribuiu um pouco** para o aprendizado do middleware.
(2) sim, o fórum **contribuiu consideravelmente** para o aprendizado do middleware.
(3) sim, o fórum **contribuiu muito** para o aprendizado do middleware.

Respostas: Todos os participantes deram nota 3, ou seja, por unanimidade, eles consideraram que o fórum de discussão *on-line* contribuiu muito para o aprendizado do M-Hub/CDDL. Um participante comentou que o fórum seria mais intuitivo se as mensagens fossem ordenadas de forma que as mais recentes aparecessem no início, enquanto que as mensagens mais antigas deveriam ser exibidas no final da página.