UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE MOTA BARRETO

COAP-CTX: EXTENSÃO SENSÍVEL AO CONTEXTO PARA DESCOBERTA DE OBJETOS INTELIGENTES EM INTERNET DAS COISAS

FORTALEZA

2017

FELIPE MOTA BARRETO

COAP-CTX: EXTENSÃO SENSÍVEL AO CONTEXTO PARA DESCOBERTA DE OBJETOS INTELIGENTES EM INTERNET DAS COISAS

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-graduação em Ciência da Computação (MDCC) da Universidade Federal do Ceará (UFC) como requisito parcial para obtenção do grau de Mestre em Ciência da Computação. Área de Concentração: Engenharia de Software.

Orientador: Prof. Dr. Windson Viana de Carvalho.

Coorientadora: Profa. Dra. Rossana Maria de Castro Andrade.

Coorientador: Prof. Dr. Marcio Espíndola Maia.

FORTALEZA
2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B262c Barreto, Felipe Mota.
CoAP-CTX: Extensão sensível ao contexto para descoberta de objetos inteligentes em internet das coisas / Felipe Mota Barreto. – 2017.
94 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2017.
Orientação: Prof. Dr. Windson Viana de Carvalho.
Coorientação: Profa. Dra. Rossana Maria de Castro Andrade.

1. Internet das coisas. 2. Descoberta de objetos inteligentes. 3. Sensibilidade ao contexto. 4. CoAP. I. Título.

CDD 005

FELIPE MOTA BARRETO

COAP-CTX: EXTENSÃO SENSÍVEL AO CONTEXTO PARA DESCOBERTA DE OBJETOS INTELIGENTES EM INTERNET DAS COISAS

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-graduação em Ciência da Computação (MDCC) da Universidade Federal do Ceará (UFC) como requisito parcial para obtenção do grau de Mestre em Ciência da Computação. Área de Concentração: Engenharia de Software.

Aprovada em: 21/08/2017

BANCA EXAMINADORA

Prof. Dr. Windson Viana de Carvalho. (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Rossana Maria de Castro Andrade. (Coorientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcio Espíndola Freire Maia. (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Emanuel Bezerra Rodrigues.
Universidade Federal do Ceará (UFC)

Prof. Dr. Kiev Santos da Gama.
Universidade Federal de Pernambuco (UFPE)

Profa. Dra. Manuele Kirsch Pinheiro.
Université Paris 1 - Panthéon Sorbonne

Dedico este trabalho aos meus pais.

AGRADECIMENTOS

A Deus, por todos os obstáculos superados e conquistas alcançadas.

Aos meus pais, Maria e Luiz, por todo o suporte durante os momentos bons e ruins.

Aos meus orientadores, professor Windson Viana, professora Rossana Andrade e professor Marcio Maia, por todo o esforço dedicado no acompanhamento deste trabalho.

Aos professores Emanuel, Kiev e Manuele, que compõem a banca examinadora e certamente contribuirão para o sucesso da solução desenvolvida.

À Universidade Federal do Ceará, pela oportunidade de fazer o mestrado.

Ao Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat), pelas experiências adquiridas.

A todos os amigos e colegas que acompanharam e compartilharam experiências durante o mestrado. Em especial ao Paulo Artur, Anderson Almada.

A todos os participantes do projeto IoT, incluindo o Alessandro Menezes, Pedro Sávio, Wítalo Benício, Ítalo Feitosa e Cláudio, que contribuíram com suas ideias para a base deste trabalho.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“O primeiro dever da inteligência é desconfiar dela mesma.”

(Albert Einstein)

RESUMO

Em um ambiente de Internet das Coisas (IoT), objetos inteligentes estão interligados de modo a permitir a criação de aplicações que permeiam os ambientes do nosso cotidiano (por exemplo, casas, carros, escolas, edifícios). O número desses objetos inteligentes tende a crescer rapidamente nos próximos anos, criando assim uma sobrecarga nas etapas de configuração, controle e uso desses objetos. Por exemplo, em aplicações como as de controle universal de ambientes, esse problema é acentuado, uma vez que essas aplicações têm por objetivo fornecer ao usuário final o controle de todos os objetos inteligentes disponíveis em um determinado ambiente. Serviços de descoberta sensíveis ao contexto tem o potencial de minimizar este problema aplicando filtros contextuais para determinar quais objetos de interesse serão descobertos e estarão disponíveis para o uso. Entretanto, no cenário de IoT, essas soluções ainda são escassas. Desta forma, este trabalho propõe o CoAP-CTX, uma extensão do serviço de descoberta padrão do protocolo CoAP (Constrained Application Protocol) que tem por objetivo dar suporte a uma descoberta sensível ao contexto de objetos inteligentes. O CoAP-CTX combina o interesse e o contexto do usuário com informações contextuais dos objetos inteligentes de modo a permitir a descoberta apenas dos objetos que são relevantes em um determinado contexto. Além disso, objetos inteligentes que não são do interesse do usuário podem entrar em modo de espera, otimizando o uso da rede e de suas baterias. Uma prova de conceito foi implementada em Android para ilustrar o funcionamento do CoAP-CTX e uma avaliação de desempenho foi realizada em um simulador de redes de sensores sem fio. Os experimentos simulados mostraram que o CoAP-CTX consegue reduzir em até 80% o número total de mensagens transmitidas na rede local para um cenário de prédio inteligente. O custo adicional no tempo de descoberta apresentou um aumento de 1 segundo e meio no pior caso, o que indica um tempo ainda aceitável na grande maioria das situações.

Keywords: Internet das Coisas. Descoberta de Objetos Inteligentes. Sensibilidade ao Contexto. CoAP.

ABSTRACT

In the Internet of Things paradigm, smart objects are interconnected to allow the creation of applications that are part of environments of everyday life (e.g., houses, cars, schools and buildings). The number of these smart objects tends to increase in elevate rates in the next years, creating an overload on the process of configuration, control and use of these objects. Solve this problem is even more important on applications such as the ones classified as universal control, because the main goal of these applications is to let the user control all smart objects available for him. Context-aware discovery services have potential to minimize this problem by applying contextual filters to determine which smart objects will be discovered and become available to user. However, in the IoT scenario, these solutions are still scarce. Then, this work proposes the CoAP-CTX, an extension of the built-in discovery service present in the CoAP (Constrained Application Protocol) that aims to provide support to a context-aware discovery of smart objects. The CoAP-CTX matches user's interest and context with the smart objects state data to discover only of the most relevant smart objects to the user in this context. Additionally, smart objects that are not of the user's interest go into an idle state, optimizing the network and battery usage. A proof of concept has been implemented on Android to illustrate how CoAP-CTX works and a performance evaluation was performed on a simulator of wireless sensor networks. The experiments have shown that CoAP-CTX can decrease the total number of messages up to 80% in scenarios where a large number smart objects are presents in a generic environment, like working buildings. The overhead of the discovery latency shows a maximum discovery time increase of 1.5 seconds, which represents an acceptable discovery time for the most type of applications.

Keywords: Internet of Things. Smart Objects Discovery. Context-awareness. CoAP.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APIs	*Application Programming Interfaces*
CAM	*Context Acquisition Manager*
CoAP	*Constrained Application Protocol*
DGT	*Distributed Geographic Table*
DLS	*Distributed Location Service*
DNS	*Domain Name System*
GREat	Grupo de Redes de Computadores, Engenharia de Software e Sistemas
HTTP	*Hypertext Transfer Protocol*
LoCCAM	*Loosely Coupled Context Acquisition Middleware*
M2M	*Machine-to-Machine*
MQTT	*Message Queue Telemetry Transport*
OIs	Objetos Inteligentes
PoC	*Proof of Concept*
REST	*Representational State Transfer*
SAC	*Sensor and Actuator Component*
SoA	*Service-Oriented Architecture*
SSDP	*Simple Service Discovery Protocol*
SysSU	*System Support for Ubiquity*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UPnP	*Universal Plug and Play*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo está estruturado da seguinte forma: a Seção 1.1 realiza a contextualização que esta dissertação aborda. A Seção 1.2 apresenta a motivação que levou ao desenvolvimento da solução apresentada neste trabalho. A Seção 1.3 expõe os objetivos e as contribuições principais deste trabalho. A Seção 1.4 descreve a metodologia utilizada neste trabalho. Finalizando, na Seção 1.5 é descrita a organização do restante desta dissertação.

1.1 Contextualização

Internet das Coisas, do inglês *Internet of Things* (IoT), é um novo paradigma tecnológico que emerge no cenário já consolidado das redes de comunicações sem fio. A ideia básica desse conceito é a constante e invisível presença, no cotidiano das pessoas, de uma enorme variedade de dispositivos computacionais, que agora passam a não ser apenas computadores tradicionais, mas também dispositivos embarcados em objetos comuns (BORGES *et al.*, 2010). Esses dispositivos possuem capacidades de processamento, armazenamento e comunicação, e são chamados de Objetos Inteligentes (OIs). Exemplos de OIs são os sensores que monitoram um ambiente, as TVs inteligentes, as câmeras de vídeo conectadas à rede, as lâmpadas inteligentes, os *smartphones*, etc. Esses objetos são unicamente endereçáveis, capazes de interagir, trocar dados entre si e ainda cooperar com seus vizinhos para realizarem tarefas em comum (ATZORI *et al.*, 2010).

Ao longo dos últimos anos, o volume mundial de OIs cresceu rapidamente. A previsão é de que no ano de 2020 será alcançada uma marca de 50 bilhões de OIs conectados (DAVE, 2011) e o número de dispositivos conectados à rede pode chegar à 13,6 por pessoa (CISCO, 2016). É interessante ressaltar que grande parte desses OIs já são compatíveis com arquiteturas e protocolos Web (HTTP, REST, entre outros). Essa previsão impõe uma discussão sobre as abordagens tradicionais de descoberta de OIs e se elas ainda serão adequadas após esse grande aumento na quantidade total. Essas abordagens tem por principal objetivo descobrir e tornar acessível todos os OIs alcançáveis, como por exemplo uma aplicação móvel que funciona como um controle universal de ambiente (THEBAULT *et al.*, 2013) e seja capaz de descobrir todos os OIs próximos que possam ser controlados ou configurados pelo usuário. Uma estratégia mais apropriada seria priorizar, ou recomendar, os OIs de maior interesse para o usuário, por

exemplo, a partir do contexto capturado por seu *smartphone* (e.g., localização, período do dia, histórico, etc.). Além das vantagens diretamente relacionadas ao usuário, com uma descoberta mais seletiva é possível economizar os recursos computacionais desses dispositivos, que possuem, por natureza, limitações computacionais e energéticas (DUARTE *et al.*, 2014).

Dispositivos móveis, como o *smartphone*, são diversas vezes utilizados como clientes do serviço de descoberta de modo a permitir a interação do usuário com os OIs presentes ao seu redor. Essa interação está associada a um aumento no consumo de energia, o que pode ser um grande problema, visto que esses dispositivos apresentam limitações com relação à suficiência energética (TALAVERA *et al.*, 2016). Dentre as principais operações que consomem energia nos dispositivos móveis, as que mais têm impactado para o aumento desse consumo são as operações de comunicação, como troca de mensagens (TARKOMA *et al.*, 2014). Portanto, otimizar o uso das interfaces de comunicação sem fio implica em uma redução no consumo de energia (PATHAK *et al.*, 2012). Uma forma de realizar essa otimização é aumentando o tempo que o dispositivo fica em modo de espera (PENTIKOUSIS, 2010). Dispositivos ainda mais limitados computacionalmente, como é o caso da grande parte dos OIs existentes, sofrem do mesmo problema energético, porém, de maneira mais intensificada.

Tanto a descoberta de OIs quanto a forma de interação com esses dispositivos estão sujeitas a diversos protocolos e especificações, visando garantir o máximo de interoperabilidade possível. Porém, a falta de padronização é uma característica marcante em IoT, o que geralmente acontece é que para cada domínio específico, uma aplicação é desenvolvida seguindo especificações particulares àquele domínio. Exemplos disso são os trabalhos de (CASTILLEJO *et al.*, 2013) e (ZHOU; MA, 2012), que abordam as áreas de *e-Health* e redes veiculares, respectivamente. Algumas especificações se sobressaem com relação à sua adoção nas mais diversas soluções de IoT, como o *Message Queue Telemetry Transport* (MQTT)[1] e o CoAP[2]. O CoAP é um protocolo de troca de mensagens desenvolvido especificamente para atender aos requisitos de dispositivos limitados computacionalmente, o que o torna adequado para utilização em cenários de IoT. Para este trabalho, o CoAP foi adotado como base da solução proposta, pois já possui um serviço de descoberta integrado, que pode ser estendido para atender a novos requisitos de descoberta para IoT.

[1] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
[2] https://tools.ietf.org/html/rfc7252

1.2 Motivação

O objetivo principal das soluções de descoberta de OIs existentes é tornar esses objetos disponíveis para as aplicações de uso e controle, ou até mesmo para serem descobertos por outros OIs. Essas soluções usam primariamente dois grupos de características no processo de descoberta: i) características funcionais, em que os requisitos da aplicação e as capacidades dos OIs são analisados; ii) características não funcionais, nos quais os atributos da execução, como latência e carga, são considerados. Entretanto, especialmente em IoT, o número de OIs tende a crescer rapidamente. Desta forma, considerar apenas características funcionais e não funcionais clássicas dos OIs pode não ser suficiente. De fato, o número de OIs retornados pelo serviço de descoberta pode ser bem elevado ou a granularidade dos requisitos de aplicação pode ser difícil de ser atingida. Por exemplo, descobrir a TV correta em uma determinada casa.

Para reduzir o número total de OIs descobertos, o processo de descoberta deve considerar não só as características funcionais e não funcionais de cada OI, mas também o contexto do usuário e do próprio OI (e.g., localização e histórico de descoberta).

Para melhor ilustrar a importância da utilização do contexto no processo de descoberta, considere o seguinte cenário. Ana é uma pessoa que costuma usar a tecnologia para facilitar as tarefas do dia a dia. Pela manhã, antes de sair de casa para o trabalho, ela utiliza seu *smartphone* para ligar a TV e poder assistir as notícias locais, mais especificamente a previsão do tempo. No carro, já em direção ao trabalho, ela programa o destino pelo assistente de direção, também utilizando o *smartphone*, que já possui o histórico do trajeto. Quando chega na empresa, Ana liga as luzes e o ar-condicionado da sala de seminários e já ajusta o projetor para a apresentação que ela fará em alguns minutos tudo a partir de se dispositivo móvel.

Descobrir os OIs mais adequados em cada um dos ambientes requer a utilização das informações de cada um dos OI juntamente com informações contextuais de Ana. Por exemplo, o serviço de descoberta pode usar a localização e a atividade (e.g., dirigindo, em reunião) de Ana para limitar o espaço de busca por OIs. A tarefa de aquisição dessas informações contextuais pode ser facilitada pela utilização de *Application Programming Interfaces* (APIs), serviços e plataformas de *middleware* de aquisição de contexto, que tratam grande parte dos desafios desse domínio (e.g., configuração e acesso aos sensores, agregação de dados, mecanismos de inferência,

etc.). Google Awareness[3] e *Loosely Coupled Context Acquisition Middleware* (LoCCAM) são soluções que podem ser utilizadas com esse propósito.

A redução do espaço de busca a partir do uso de informação contextual não só traz benefícios na interação entre usuário e aplicação, mas também pode otimizar o consumo energético dos dispositivos envolvidos. Os OIs que não são relevantes ao usuário em um determinado contexto podem entrar momentaneamente em estado de espera, reduzindo o número total de mensagens trocadas.

Devido à problemática apresentada, a seguinte questão de pesquisa norteou esta dissertação de mestrado:

- **Questão de Pesquisa:** Como utilizar informações contextuais do usuário e dos objetos inteligentes para realizar uma descoberta seletiva, na qual apenas os objetos inteligentes acessíveis e de interesse do usuário sejam selecionados?

1.3 Objetivos e Contribuições

Com o intuito de responder adequadamente a questão de pesquisa exposta na seção anterior, este trabalho de mestrado tem por objetivo primário a criação de um serviço de descoberta sensível ao contexto que permita selecionar os OIs que são relevantes ao usuário em um determinado momento. Esse serviço de descoberta pode ser utilizado por desenvolvedores de aplicações móveis que controlem os OIs do ambiente para apresentá-los, de maneira mais seletiva e eficiente, ao usuário final. Com base nisso, derivam-se duas contribuições principais:

- Reduzir o número total de OIs que são encontrados pelo serviço de descoberta e apresentados ao usuário final da aplicação. Isso é feito por meio da priorização dos OIs que são relevantes ao usuário;
- Otimizar a gerência dos OIs que participam do processo de descoberta, minimizando a troca de mensagens na rede. OIs que não são do interesse do usuário entram em modo de espera reduzindo o número de mensagens enviadas e, consequentemente, economizam recursos computacionais e energéticos.

A solução proposta é denominada CoAP-CTX (CoAP **C**on**T**e**X**tual). Ela consiste em

[3] https://developers.google.com/awareness/

uma extensão do serviço de descoberta presente no *Constrained Application Protocol* (CoAP). O CoAP-CTX estende o CoAP de modo a considerar as informações contextuais do usuário e dos OIs durante o processo de descoberta. O serviço de descoberta integrado ao CoAP foi modificado de modo a gerenciar o espaço de busca com base em informações contextuais, fazendo com que apenas OIs relevantes aos usuários sejam retornados à aplicação cliente do protocolo (e.g., uma aplicação móvel de controle universal). O CoAP-CTX pode ser utilizado por aplicações que desejam controlar, de forma mais eficiente, os OIs presentes em ambientes inteligentes como casa, carros e prédios.

É importante ressaltar que o CoAP-CTX não predetermina quais mecanismos de aquisição de contexto e de *matching* contextual são os mais adequados para descobrir os OIs de interesse do usuário. O escopo desta pesquisa se limita a, uma vez estabelecidos o contexto e os filtros contextuais utilizados no *matching*, gerenciar a interação entre os elementos CoAP (clientes, servidores e OIs) de forma a determinar quais dos OIs são relevantes e quais devem ser retornados como resultado da descoberta.

Para atingir e comprovar as duas contribuições descritas, o CoAP-CTX utilizou o middleware LoCCAM[4] para realizar a aquisição de contexto e para gerenciar a comunicação entre *smartphone* e OIs. Além disso, foi desenvolvida uma aplicação, chamada U-Control, que permite ao usuário acessar e controlar todos os OIs que foram descobertos pelo CoAP-CTX.

1.4 Metodologia

Este trabalho seguiu a seguinte metodologia científica:

- **Revisão da Literatura**: Inicialmente, um estudo bibliográfico foi realizado sobre Internet das Coisas e o estado da arte em descoberta de OIs. Após identificadas as lacunas das abordagens encontradas, foram realizados estudos sobre sensibilidade ao contexto e tecnologias específicas, como protocolos de comunicação para IoT e simuladores de OIs;
- **Estudo dos Trabalhos Relacionados**: A partir da revisão da literatura, trabalhos relacionados foram escolhidos com base em dois critérios: i) se apresentavam algum serviço de descoberta de objetos inteligentes para IoT; ii) se propunham

[4] O LoCCAM foi desenvolvido no Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat). Mais informações em: http://loccam.great.ufc.br

alguma aplicação que permitisse ao usuário controlar os OIs em um ambiente inteligente. Os trabalhos encontrados foram classificados com relação aos requisitos específicos de descoberta para IoT definidos por (GUINARD *et al.*, 2010) detalhados no Capítulo 2;

- **Definição das funcionalidades do CoAP-CTX**: Com a classificação dos trabalhos relacionados, percebeu-se lacunas nos serviços existentes de descoberta de OIs. Então, foram definidas as funcionalidades que deveriam estar presentes na solução apresentada por este trabalho, como a utilização de técnicas de sensibilidade ao contexto e suporte à alocação sob demanda de OIs;
- **Modelagem**: Nesta etapa, a arquitetura geral do CoAP-CTX foi estabelecida, bem como os modelos de interações entre componentes da solução. Além disso, definiu-se os protocolos de troca de mensagens, diagramas contendo a especificação dos módulos que compõem a solução (e.g., diagramas de classes);
- **Implementação**: À medida em que a modelagem de algum componente do sistema era terminada, a implementação do mesmo era iniciada, seguindo um modelo iterativo e incremental. Cada módulo do CoAP-CTX foi testado individualmente ao final de sua implementação. Testes de integração foram feitos quando todos os componentes foram concluídos, para garantir o correto funcionamento da solução; e
- **Avaliação**: Após a implementação do serviço de descoberta, foi criado uma *Proof of Concept* (PoC) com o objetivo de comprovar a viabilidade do CoAP-CTX e se o número de OIs encontrados pode ser diminuído. Além disso, foi realizada uma simulação para melhor avaliar o CoAP-CTX e seus impactos positivos e negativos com relação ao CoAP. Nela, o número de OIs foi variado em três cenários estudados (casa, carro e prédio). Com essa simulação, foi possível estimar o número de mensagens que o CoAP-CTX consegue reduzir, bem como analisar o aumento no tempo de descoberta ocasionado pelo processamento adicional que a solução possui.

1.5 Organização da Dissertação

O restante desta dissertação está organizado da seguinte forma:

- O Capítulo 2 apresenta a fundamentação teórica, na qual são abordados temas como Internet das Coisas, Sensibilidade ao Contexto e Descoberta de OIs. Também é apresentada uma base teórica sobre as tecnologias e protocolos utilizados nesse trabalho, como o LoCCAM e o CoAP.
- O Capítulo 3 detalha os trabalhos relacionados que foram selecionados baseados nos critérios de descoberta de OIs. Esses trabalhos são serviços de descoberta para IoT, que utilizam alguma forma de informação contextual, ou trabalhos que possuem o CoAP como base para a realização da descoberta.
- Já o Capítulo 4 apresenta a visão geral e a arquitetura do CoAP-CTX. Mostra como a solução foi desenvolvida e integrada ao LoCCAM e ao CoAP. Um processo de descoberta sensível ao contexto é descrito no capítulo, mapeando todas as etapas necessárias para se realizar essa descoberta sensível ao contexto.
- No Capítulo 5, é descrita a prova de conceito desta dissertação, uma aplicação Android que controla os OIs disponíveis no ambiente, utilizando o CoAP-CTX para a descoberta. Além disso, uma avaliação de desempenho é detalhada. Ela utiliza um simulador para redes de sensores sem fio. Os resultados da avaliação são discutidos a fim de comprovar a redução no número de mensagens trocadas e o impacto no tempo de descoberta.
- O Capítulo 6 traz as considerações finais sobre o trabalho. São discutidos os resultados alcançados, as limitações do trabalho, as possibilidades de continuação e aperfeiçoamento da pesquisa. Todas as publicações em conferências e *workshops* que foram realizadas durante o período deste trabalho de mestrado são listadas nesse capítulo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, os conceitos que constituem a base teórica deste trabalho são apresentados. A Seção 2.1 traz um estudo realizado sobre Internet das Coisas, incluindo as arquiteturas de referência existentes e aspectos relacionados à descoberta de OIs. Além disso, a Seção 2.2 apresenta o tema de sensibilidade ao contexto, trazendo definições e o conceito de *matching* contextual. A Seção 2.3 apresenta o *middleware* de aquisição de contexto utilizado neste trabalho, chamado LoCCAM. A Seção 2.4 detalha o CoAP, protocolo de troca de mensagens que possui um serviço de descoberta integrado usado como base para este trabalho. Por fim, a Seção 2.5 traz as considerações finais sobre os temas abordados neste capítulo.

2.1 Internet das Coisas

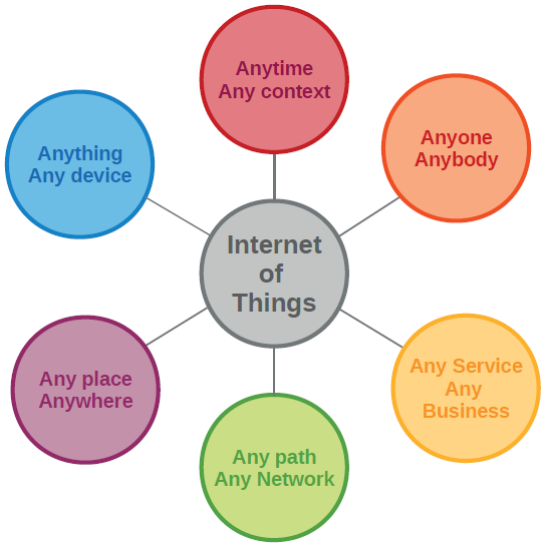
Devido ao avanço na indústria de sensores, tais componentes estão se tornando cada vez mais baratos, menores e poderosos. Como resultado, a utilização desses sensores já atingiu uma escala considerável, além de que esse número deve crescer ainda mais rapidamente nos próximos anos (SUNDMAEKER *et al.*, 2010). Esse grande número de sensores criará uma grande massa de dados, ou *big data* (ZASLAVSKY *et al.*, 2012). Esses dados podem não ter nenhum valor caso não sejam analisados, interpretados e o mais importante, entendidos. É por isso que todos os desafios de sensibilidade ao contexto que eram aplicáveis aos paradigmas de computação ubíqua continuarão existindo, agora em IoT (PERERA *et al.*, 2014).

Gubbi *et* al. definem Internet das Coisas como sendo a interconexão de sensores e atuadores provendo a habilidade de compartilhar informação ao longo de plataformas através de um *framework* unificado, desenvolvendo um cenário operacional comum para permitir aplicações inovadoras. Isso é alcançado por um sensoriamento ubíquo, análise dos dados e representação da informação, tendo a *cloud computing* como o *framework* unificador (GUBBI *et al.*, 2013). Já objetos inteligentes são dispositivos que possuem identidade e personalidade virtual que operam em ambientes inteligentes usando interfaces para conectar e se comunicar dentro de um contexto social, local e pessoal (TAN; WANG, 2010).

Uma outra definição de IoT, bem mais abrangente, é dada por (GUILLEMIM; FRIESS, 2009), na qual os autores afirmam que IoT é o ambiente que permite que pessoas e

OIs estejam conectados a qualquer momento, em qualquer lugar, com qualquer coisa e qualquer um, idealmente usando qualquer rede e qualquer serviço. Essa definição pode ser visualizada na Figura 1, em que IoT é vista como a união entre o conjunto de características e propriedades apresentadas.

Figura 1 – Definição de Internet das Coisas como sendo a união entre diversas propriedades.



Fonte – (GUILLEMIM; FRIESS, 2009)

Embora IoT não possua uma definição única, a grande maioria delas giram em torno de uma mesma ideia, de que objetos inteligentes estão interconectados de forma a permitir a troca de dados entre eles mesmos e o ambiente, enquanto reagem a eventos criados com ou sem a intervenção humana. Outro aspecto importante é a utilização de padrões e protocolos bem especificados para a comunicação com, e entre, esses OIs, buscando assim atingir a máxima interoperabilidade possível entre os mais diversos dispositivos. Dentre essas especificações, destacam-se as que se adéquam às características comuns dos OIs, como baixo poder computacional e limitações energéticas. Limitações essas que fazem com que protocolos amplamente adotados na WEB tradicional, como o *Hypertext Transfer Protocol* (HTTP)[1], percam espaço para protocolos mais adequados aos cenários de IoT, como o CoAP e o MQTT.

2.1.1 Arquitetura de Referência

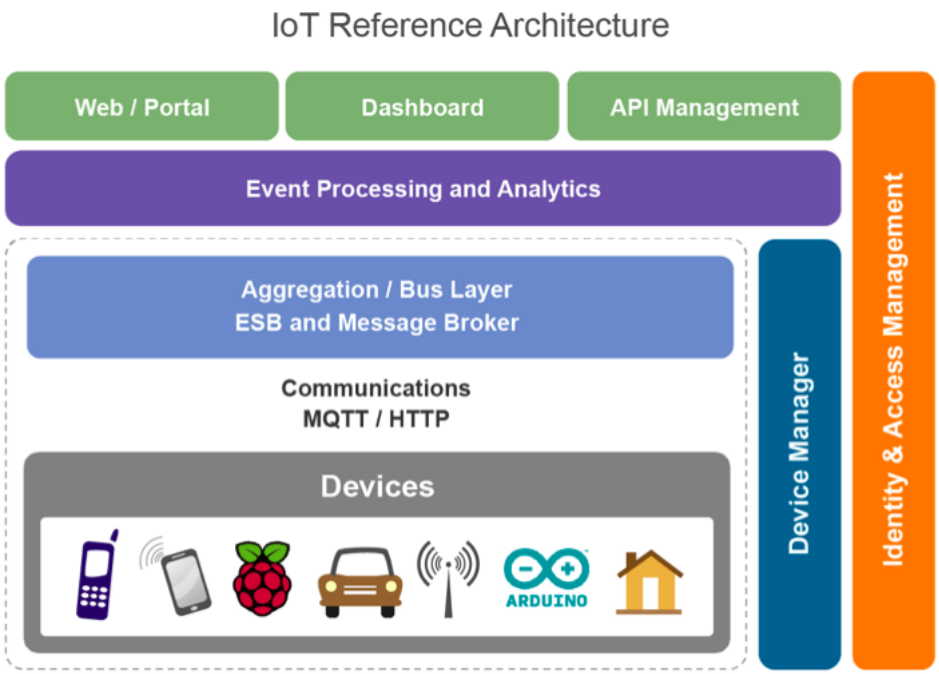
Um cenário de Internet das Coisas é caracterizado por um alto grau de heterogeneidade, no que diz respeito a dispositivos com diferentes configurações, funcionalidades e

[1] https://tools.ietf.org/html/rfc2616

protocolos de comunicação. Para tratar isso, várias plataformas vem sendo propostas visando abstrair as especificidades de cada um desses dispositivos, promovendo assim a interoperabilidade entre eles. Porém, a falta de padronização em IoT faz com que essas plataformas muitas vezes não cumpram vários requisitos importantes (CAVALCANTE *et al.*, 2015). A adoção de uma arquitetura de referência pode ajudar nessa tarefa, pois define um conjunto de pequenos blocos independentes para a construção de um sistema de IoT e, com isso, cria uma base sólida para a sua ampla adoção.

Existem várias arquiteturas de referência para Internet das Coisas, dentre as quais está a proposta pela empresa WSO2 e que foi baseada na experiência da companhia no desenvolvimento de soluções para IoT (FREMANTLE, 2014). O principal objetivo dessa e das outras arquiteturas de referência é fornecer aos desenvolvedores um ponto inicial capaz de cobrir a maioria dos requisitos de sistemas para IoT. Sendo assim, a arquitetura de referência apresentada na Figura 2 não é especifica e nem é limitada a nenhum conjunto específico de tecnologias ou protocolos de comunicação.

Figura 2 – Arquitetura de referência para Internet das Coisas.



Fonte – (CAVALCANTE *et al.*, 2015)

A arquitetura de referência em questão possui cinco camadas, cada uma com funcionalidades bem definidas:

1. ***Device Layer*** é a camada em que se deve garantir que cada dispositivo é unicamente endereçável e possua uma comunicação, direta ou indireta, para a Internet;
2. ***Communications Layer*** é onde se deve garantir a conectividade com cada dispositivo, possivelmente utilizando múltiplos protocolos de comunicação;
3. ***Aggregation/Bus Layer*** é a camada onde os dados oriundos de cada dispositivo são agregados e convertidos em uma única forma de representação e um único protocolo;
4. ***Event Processing and Analytics Layer***, camada esta que reage a eventos vindos da *Aggregation/Bus Layer*, além de também poder armazenar os dados;
5. Finalmente, usuários podem acessar os dispositivos, ou os dados armazenados, através da camada ***External Communications Layer***.

Além dessas cinco camadas, a arquitetura de referência proposta pela WSO2 possui duas camadas adicionais: *(i) Device Management Layer*, que garante a gerência e o acesso remoto aos dispositivos; e *(ii) Identity and Access Management*, responsável por questões de segurança e controle de acesso.

Embora essa arquitetura de referência trate da maioria dos requisitos relacionados à Internet das Coisas, Cavalcante *et al.* elencam alguns pontos essenciais para IoT que não foram totalmente cobertos pela arquitetura proposta, tais como descoberta de dispositivos, sensibilidade ao contexto, gerenciamento de grande volume de dados e adaptação dinâmica (CAVALCANTE *et al.*, 2015). Essas lacunas devem ser preenchidas para uma solução completa em IoT. Como veremos mais adiante, o LoCCAM atua em todas as camadas propostas pela arquitetura de referência, com exceção da *External Communications Layer*. Essa camada, entretanto, será abordada pela aplicação U-Control, proposta como prova de conceito deste trabalho.

2.1.2 Descoberta de Objetos Inteligentes

Em ambientes altamente voláteis, arquiteturas orientadas a serviços, ou do inglês (*Service-Oriented Architecture* (SoA)), representam uma abordagem viável para garantir o desacoplamento na interação entre as diversas entidades que compõem o sistema (MAIA *et al.*, 2013), facilitando assim o desenvolvimento e o gerenciamento desse sistema. Esse tipo de arquitetura aumenta a flexibilidade das aplicações que seguem esse paradigma, permitindo uma fácil integração entre plataformas e aplicações heterogêneas (GAMA *et al.*, 2012). No paradigma SoA, um serviço é uma abstração para representar funcionalidades requeridas e

providas, acessados usando uma interface bem conhecida. Durante o desenvolvimento de um sistema baseado em SoA, o engenheiro de *software* especifica os serviços que serão utilizados, em tempo de desenvolvimento, e o serviço de descoberta é o responsável por encontrar esses serviços necessários, baseado na sua descrição.

O conceito de descoberta de serviços é amplamente conhecido no desenvolvimento de sistemas distribuídos. Crasso *et al.* definem essa descoberta como um processo de encontrar os serviços adequados para resolver uma determinada tarefa em particular (CRASSO *et al.*, 2008). Esses serviços podem ser vistos tanto como abstrações de softwares disponíveis (e.g., um servidor de um jogo) quanto como recursos ou dispositivos computacionais passíveis de serem encontrados (e.g., impressoras em uma rede).Por exemplo, *Simple Service Discovery Protocol* (SSDP)[2] e o *Universal Plug and Play* (UPnP)[3] são protocolos que possuem um serviço de descoberta integrado. Entretanto, esses protocolos foram criados para utilização em aplicações móveis/pervasivas/ubíquas, não considerando requisitos e restrições impostas pelo paradigma de IoT.

Esses requisitos e restrições impostos pela IoT são oriundas principalmente das características de *hardware* dos OIs, pois: i) grande parte dos OIs apresentam algum nível de limitações na utilização de recursos, sejam elas de processamento, armazenamento, transmissão de dados ou consumo energético; ii) quando implantados em um ambiente, os OIs devem ser capazes de interagir com o mínimo de intervenção humada possível; e iii) a quantidade de OIs tende a aumentar rapidamente. Com essas características em mente, (GUINARD *et al.*, 2010) propõe que um serviço de descoberta desenvolvido para IoT deve atender aos seguintes requisitos:

1. **REQ 1 - Mínimo *overhead* no serviço de descoberta:** como a maioria dos OIs do mundo real são dispositivos com baixo poder computacional, existe uma necessidade de utilização de paradigmas de descoberta mais leves. Técnicas de redução do tamanho das mensagens trocadas podem ser utilizadas nesse sentido, bem como a adoção de políticas que priorizem a eficiência na troca de mensagens em troca da garantia de entrega;
2. **REQ 2 - Mínimo esforço no registro de OIs:** um OI deve ser capaz de anunciar seus serviços a um servidor de registros através da rede. Esse processo precisa ser realizado

[2] https://tools.ietf.org/html/draft-cai-ssdp-v1-03
[3] https://openconnectivity.org/resources/specifications/upnp

sem nenhuma intervenção humana. A quantidade de informações necessárias para esse registro também deve ser bem reduzida. Além disso, o servidor de registros deve ser capaz de gerenciar, de maneira eficiente, a entrada e saída de novos OIs do ambiente;

3. **REQ 3 - Suporte à busca contextual e dinâmica de OIs:** os algoritmos de busca devem ir além de uma simples busca por palavras-chave. Eles devem levar em consideração dados dinâmicos do contexto do usuário, como localização. Informações contextuais devem ser utilizadas para otimizar o processo de descoberta, reduzindo o espaço de busca; e
4. **REQ 4 - Suporte à alocação de OIs sob demanda:** os serviços ofertados pelos OIs devem ser ativados sob demanda, evitando assim a má utilização dos mesmos. Idealmente, OIs que não estivessem sendo efetivamente acessados, deveriam entrar em modo de espera, a fim de economizar recursos.

Os dois primeiros requisitos (Mínimo *overhead* no serviço de descoberta e mínimo esforço no registro de OIs) foram a principal motivação para a utilização do serviço de descoberta integrado ao CoAP como base para a descoberta de OIs, visto que esse protocolo já os cumpre sem a necessidade de nenhum esforço de implementação adicional. A solução proposta nesta dissertação busca estender o mecanismo padrão de descoberta do CoAP de modo a atender aos dois requisitos restantes (**REQ 3 e 4**).

Sob o ponto de vista dos clientes que consomem dos dados providos pelo serviço de descoberta, Datta *et al.* definem vários princípios que devem ser levados em consideração durante o desenvolvimento desse serviço de descoberta, dentre eles destacam-se: i) o serviço de descoberta deve ser independente da tecnologia de comunicação; ii) o serviço de descoberta deve expor os OIs encontrados por meio de interfaces *Representational State Transfer* (REST) ou APIs que abstraiam esse acesso; iii) o serviço de descoberta deve dar suporte tanto à buscas simples por OIs, baseadas em ID, quanto à buscas mais complexas, baseadas em categorias e localização; e iv) o serviço de descoberta deve garantir a eficiência energética dos OIs (DATTA; BONNET, 2015).

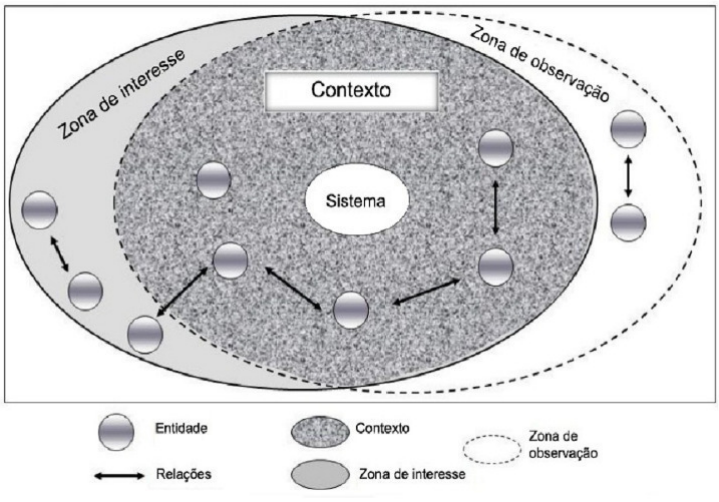
2.2 Sensibilidade ao Contexto

Sensibilidade ao contexto é a habilidade do sistema de se modificar devido a mudanças no contexto do usuário (e.g., localização, humor, atividade) (MAIA *et al.*, 2013), com pouca ou nenhuma intervenção direta do usuário. Sensibilidade ao contexto é uma característica

essencial em IoT para melhorar a interação entre usuários e OIs, além de representar um grande potencial na sua utilização nos serviços de descoberta (TURUNEN *et al.*, 2015). Baldauf *et al.* definem que sistemas sensíveis ao contexto são aqueles que conseguem adaptar sua operação de acordo com um determinado contexto, sem a intervenção explícita do usuário (BALDAUF *et al.*, 2007). Ainda segundo os autores, sistemas sensíveis ao contexto tem por objetivo melhorar a usabilidade e deixar mais efetiva a interação entre usuário e sistema, por meio da utilização do contexto do ambiente.

Dey *et al.* definem contexto como qualquer informação que pode ser usada para caracterizar a situação de um elemento que é relevante para a interação entre usuário e sistema, incluindo como elementos, o próprio usuário e o sistema (DEY *et al.*, 2001). Além de considerar a situação atual do usuário, caracterizada muitas vezes pela localização e demais informações obtidas pelo *smartphone*, o interesse do usuário também deve ser levado em consideração para uma boa caracterização do contexto (PINHEIRO *et al.*, 2006). A definição que será adotada neste trabalho, proposta por Viana *et al.*, pode ser vista como uma extensão da definição de Dey, removendo a limitação do contexto sobre a necessidade de interação entre usuário e sistema. Contexto agora passa a ser um conjunto de informações que podem descrever a situação das entidades (e suas relações) envolvidas em uma ação que seja julgada importante para o sistema (e.g., interações, busca de dispositivos, entre outros) (VIANA *et al.*, 2011). A Figura 3 exemplifica visualmente essa definição.

Figura 3 – Definição de Contexto como interseção entre as Zonas de Interesse e de Observação.



Fonte – (DUARTE *et al.*, 2015a)

O contexto é definido como a interseção entre duas zonas. A primeira delas, chamada de Zona de Interesse (ZI), representa o conjunto de entidades que o sistema julga importante em um determinado instante de tempo, ou seja, quais informações contextuais o sistema tem interesse em obter em um instante *t*. A segunda zona, chamada de Zona de Observação (ZO), é composta pelas entidades que podem ser acessadas e obtidas nesse mesmo instante de tempo pela infraestrutura de aquisição de contexto. Essa definição de contexto possui duas características importantes. O contexto é tanto dinâmico (as informações fornecidas por cada uma das entidades muda com o tempo) quanto evolutivo (as próprias entidades que compõem o contexto podem mudar) (DUARTE *et al.*, 2015a). Isso faz com que a aplicação dessa definição em sistemas móveis, ubíquos e de IoT seja bastante adequada, pois os elementos que compõem esses sistemas são bastante voláteis, o que faz com que as zonas mudem constantemente.

A sensibilidade ao contexto pode ser utilizada para aprimorar o processo de descoberta de OIs, selecionando apenas os OIs que são relevantes ao usuário em um determinado contexto. A etapa na qual as informações contextuais, geralmente representando características não funcionais, são analisadas e processadas de modo a determinar se esses OIs são relevantes ao usuário, é chamada de *matching* contextual. Os algoritmos de *matching* são utilizados na seleção de serviços, recomendações de música e fotos. Existem diversas maneiras de se realizar um *matching* contextual, que vão desde uma simples comparação textual entre o contexto atual do usuário e do objeto ou item a ser filtrado (e.g., OIs, serviço Web) até soluções mais complexas, como por exemplo utilizando algoritmos baseados em grafos (PINHEIRO *et al.*, 2008).

2.3 LoCCAM

A tarefa de adquirir informações contextuais pode ser facilitada pela utilização de APIs e plataformas de *middleware* de aquisição e gerenciamento de contexto, que tratam da maioria dos desafios desse domínio (e.g., configuração e acesso a sensores, agregação de dados, mecanismos de inferência). Google Awareness, Octopus (FIRNER *et al.*, 2011), Fiware[4], CA4IOT (PERERA *et al.*, 2012), SeeMon (KANG *et al.*, 2008) e EEMSS (WANG *et al.*, 2009) são exemplos de soluções que agregam a maioria dessas características. Para este trabalho, foi adotado o LoCCAM, que é um *middleware* de gerenciamento e aquisição de contexto desenvolvido no laboratório Grupo de Redes de Computadores, Engenharia de Software e

[4] https://www.fiware.org/

Sistemas (GREat) para a plataforma Android. O LoCCAM permite a realização da aquisição de contexto de forma auto-adaptativa seguindo à definição de contexto proposta por (VIANA *et al.*, 2011). O desenvolvimento do LoCCAM foi guiado por cinco princípios de *design*, listados a seguir:

- Os requisitos da aquisição de contexto devem ser projetados de acordo com os conceitos do desenvolvimento de software baseado em componentes.
- A informação contextual deve ser disponibilizada por meio de um espaço de tuplas ou via notificações do tipo *Publish/Subscribe*.
- A camada de aquisição de contexto e as aplicações devem ser desacopladas.
- A aquisição de informação contextual deve ocorrer de forma transparente.
- A adaptação da aquisição de contexto deve possibilitar a economia de recursos (e.g., memória, bateria).

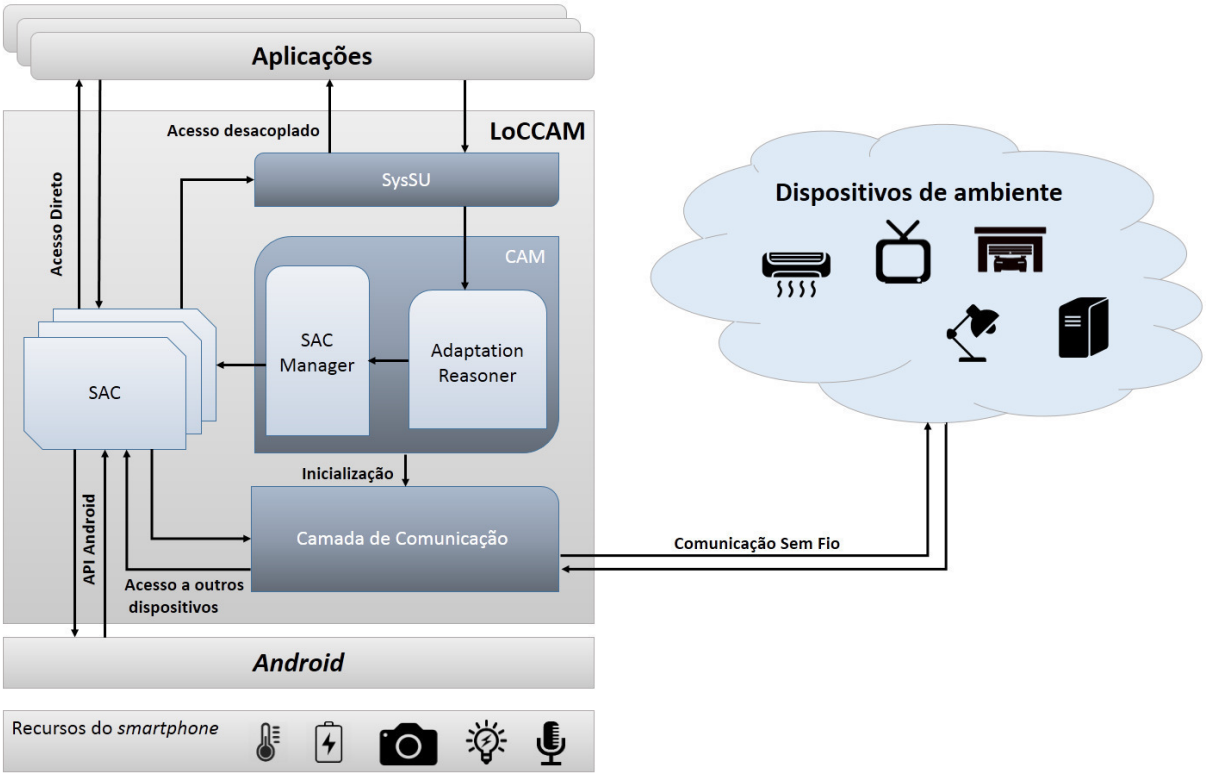
Ao longo dos últimos cinco anos, o LoCCAM passou por modificações com o objetivo de adequar seus componentes e funcionalidades para o emergente paradigma de IoT. A principal mudança foi que em sua versão atual, além de adquirir informações contextuais, o LoCCAM também é capaz de atuar no ambiente, trocando mensagens com dispositivos próximos a fim de realizar alguma modificação em seu estado (e.g., ligar ou desligar o ar condicionado). Além disso, foi adicionada uma camada de comunicação, que abstrai o acesso e controle desses OIs presentes no ambiente, de maneira extensível, ou seja, novos dispositivos com protocolos de comunicação diferentes (e.g., CoAP) podem ser facilmente integrados a essa camada de comunicação.

2.3.1 Arquitetura

O LoCCAM foi desenvolvido para a plataforma Android, inicialmente com o objetivo de obter informações contextuais oriundas dos próprios sensores presentes no *smartphone*, porém essa ideia evoluiu e atualmente é possível adquirir dados contextuais de *webservices* ou de OIs presentes no ambiente, além de também atuar nesses dispositivos. O LoCCAM é composto por 5 componentes principais, que compõem a arquitetura exposta na Figura 4, apresentados a seguir:

- ***System Support for Ubiquity* (SysSU) (LIMA *et al.*, 2011):** Representa o espaço de tuplas que coordena a interação entre aplicações e componentes de mais baixo nível,

Figura 4 – Arquitetura do LoCCAM.



Fonte – Elaborado pelo Autor.

chamados *Sensor and Actuator Component* (SAC). O objetivo do SysSU é desacoplar as entidades responsáveis por sensorear ou atuar no ambiente e as aplicações.

- ***Adaptation Reasoner*:** É responsável pelo processamento das regras de adaptação, baseado nas zonas de interesse e de observação. É esse componente que indica quando um SAC deve ser iniciado ou retirado de execução, baseado no interesse da aplicação e dos outros SACs ativos.
- ***SAC Manager*:** Gerencia o ciclo de vida dos SACs. É possível instalar, desinstalar, ativar e desativar SACs em tempo de execução, baseado no contexto atual, utilizando um *loader* específicos para o SACs, integrado junto ao LoCCAM.
- **SAC:** Os SACs são responsáveis por encapsular o acesso a sensores e atuadores, sejam eles do próprio *smartphone* ou do ambiente. O encapsulamento permite que todos esses dispositivos sejam acessados de maneira única, seguindo uma interface bem definida. O conceito de componentes de software garante a reusabilidade dos SACs, componentes podem ser desenvolvidos uma vez e reusados por todas as aplicações que precisarem de sua respectiva informação contextual.
- ***Camada de Comunicação*:** Para acessar os dispositivos presentes no ambiente, o LoC-

CAM possui uma camada de comunicação que abstrai as trocas de mensagens e especificidades de cada tecnologia de comunicação (e.g., bluetooth, Wi-Fi). Essa camada pode ser acessada pelos SACs para sensorear ou atuar no ambiente.

Com exceção da camada de comunicação, que será abordada a seguir, os detalhes sobre os demais componentes que compõem a arquitetura do LoCCAM podem ser encontrados no Apêndice A.

Fazendo um paralelo da arquitetura do LoCCAM com a arquitetura de referência apresentada na Seção 2.1.1, é possível identificar pontos em comum: i) As camadas de gerenciamento de dispositivos e gerenciamento de acesso são representadas pela camada de comunicação no LoCCAM; ii) A cada de dispositivos é representada pela camada CAM; iii) A camada de agregação e processamento de eventos é representada na arquitetura do LoCCAM pelo componente SysSU. Entretanto, o LoCCAM não possui componentes gráficos, como sugere a arquitetura de referência, nesse caso as aplicações que utilizam o LoCCAM é que são responsáveis por esta camada.

2.3.2 Camada de Comunicação

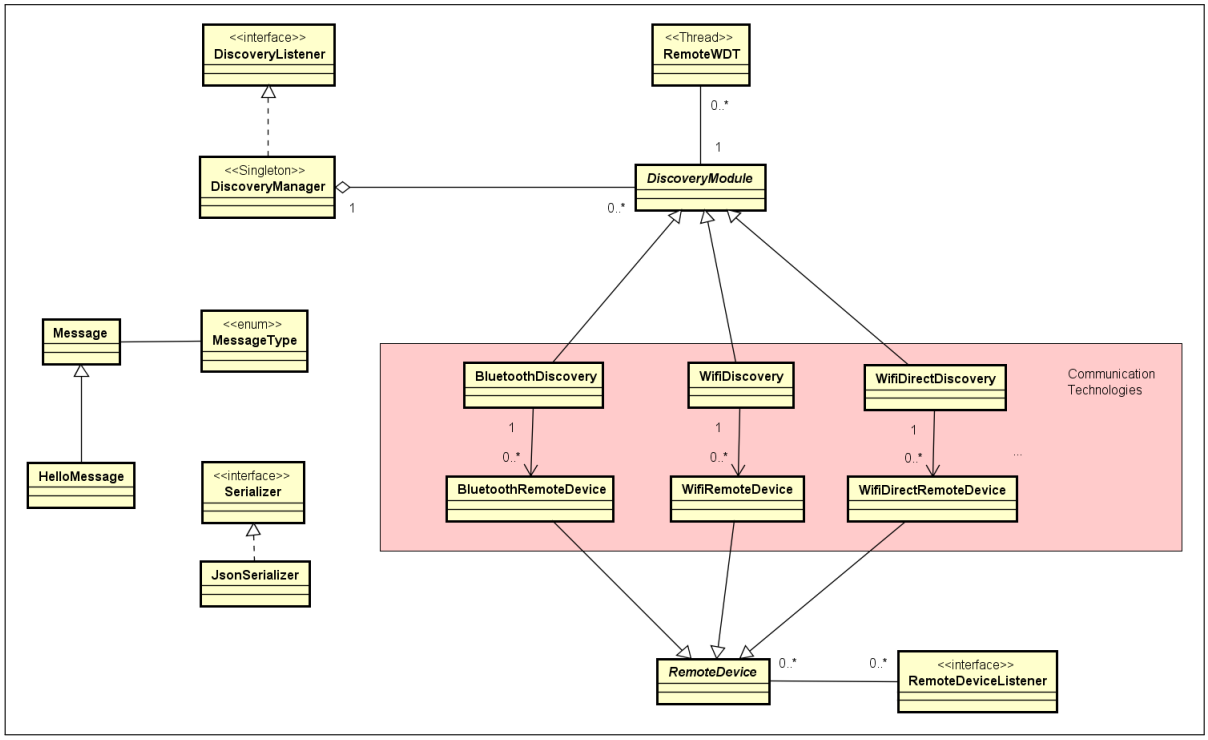
Uma das principais funcionalidades adicionadas ao LoCCAM para uma melhor adequação em um cenário de IoT foi a capacidade de interagir com dispositivos no ambiente, e com suporte a múltiplas tecnologias de comunicação. Essa camada é utilizada na abordagem CoAP-CTX proposta nesta dissertação[5]. O conjunto de protocolos e especificações existentes no quesito de tecnologias de comunicação é bem vasto, devido à heterogeneidade dos diversos OIs presentes em um ambiente inteligente. Esse fato cria uma necessidade de comunicação em diferentes tecnologias e padrões utilizados na indústria, como Wi-Fi, Bluetooth, WiFi-Direct e até mesmo protocolos de mais alto nível, como CoAP (MAIA *et al.*, 2014). Para atender esse requisito, o LoCCAM possui uma camada de comunicação que abstrai o acesso aos OIs presentes no ambiente por meio das mais diversas tecnologias.

Nessa camada de comunicação, um serviço de descoberta é inicializado junto com o processo de inicialização do LoCCAM. Os principais componentes desse serviço de descoberta

[5] Toda a camada de comunicação do LoCCAM foi desenvolvida durante um projeto de P&D em 2015. Esse projeto, coordenado pelo GREat, contou com a participação de vários professores, pesquisadores e desenvolvedores, incluindo o autor dessa dissertação, e teve como principal resultado a adequação do LoCCAM para um paradigma de Internet das Coisas.

podem ser visualizados na Figura 5. Aplicações e SACs acessam esse serviço por meio da classe *DiscoveryManager*, que é responsável por manter o registro de todos os OIs disponíveis no ambiente. A interface *DiscoveryListener* pode ser utilizada como *callback* para receber notificações a cada mudança na lista de OIs disponíveis, assim como detecção de desconexões dos OIs.

Figura 5 – Diagrama de classes da camada de comunicação do LoCCAM.



Fonte – Elaborado pelo Autor.

O *DiscoveryManager* inicializa um conjunto de módulos dependentes de tecnologia de comunicação, os *DiscoveryModule*. São esses módulos que gerenciam os OIs específicos de cada tecnologia (e.g., Wi-Fi e Bluetooth), além disso, cada módulo possui um *RemoteWatchDogTimer* responsável por identificar desconexões, já que OIs uma vez reconhecidos, devem enviar mensagens de *ping* periodicamente. O acesso de fato aos OIs é feito por uma dupla de classes, o módulo de descoberta específico (e.g., *WiFiDiscovery*) é responsável por encontrar na rede os OIs disponíveis, já a abstração de acesso à esses OIs (e.g., *WiFiRemoteDevice*) é responsável por prover métodos para envio, recebimento e subscrição de mensagens.

O serviço de descoberta do LoCCAM utiliza um protocolo próprio, baseado em mensagens auto contidas representadas pela classe *Message*. Cada mensagem possui um tipo, *MessageType*, que pode ser *GET_REQUEST*, *GET_RESPONSE*, *SET_REQUEST*, *SET_RESPONSE*,

BEACON e *UNKNOWN*. Os tipos de GET e SET são mensagens convencionais para envio e recebimento de dados, já o *BEACON* é usado pelos OIs como forma de *ping*, indicando que ainda está ativo e acessível. O tipo *UNKNOWN* é utilizado como indicativo de algum erro durante o envio e recebimento das mensagens. As mensagens são serializadas antes do envio, por meio das classes *Serializer* e *JsonSerializer*, bem como desserializadas conforme a necessidade dos receptores dessas mensagens. Por fim, o protocolo estabece que a primeira mensagem enviada a um novo OI deva seguir um formato específico, representado pela *HelloMessage*, que nada mais é do que uma requisição do tipo *GET_REQUEST* em que é solicitado que o OI informe todos os recursos disponíveis e controlados por ele (e.g., sensores de luminosidade, temperatura, e controle das portas).

Essa camada de comunicação do LoCCAM permite que aplicações e SACs interajam com os OIs de maneira independente de tecnologia, por meio da abstração de *RemoteDevice*. Clientes desse serviço de descoberta apenas solicitam ao *DiscoveryManager* a lista de *RemoteDevices* que possuem uma determinada informações contextual ou que atuam sobre determinado recurso no ambiente. Essa abstração permite uma fácil extensão dessa camada para dar suporte a novas tecnologias e protocolos de comunicação, como o CoAP. Para realizar essa extensão, basta a implementação de um novo módulo de descoberta, capaz de identificar, encontrar e acessar os OIs dessa nova tecnologia.

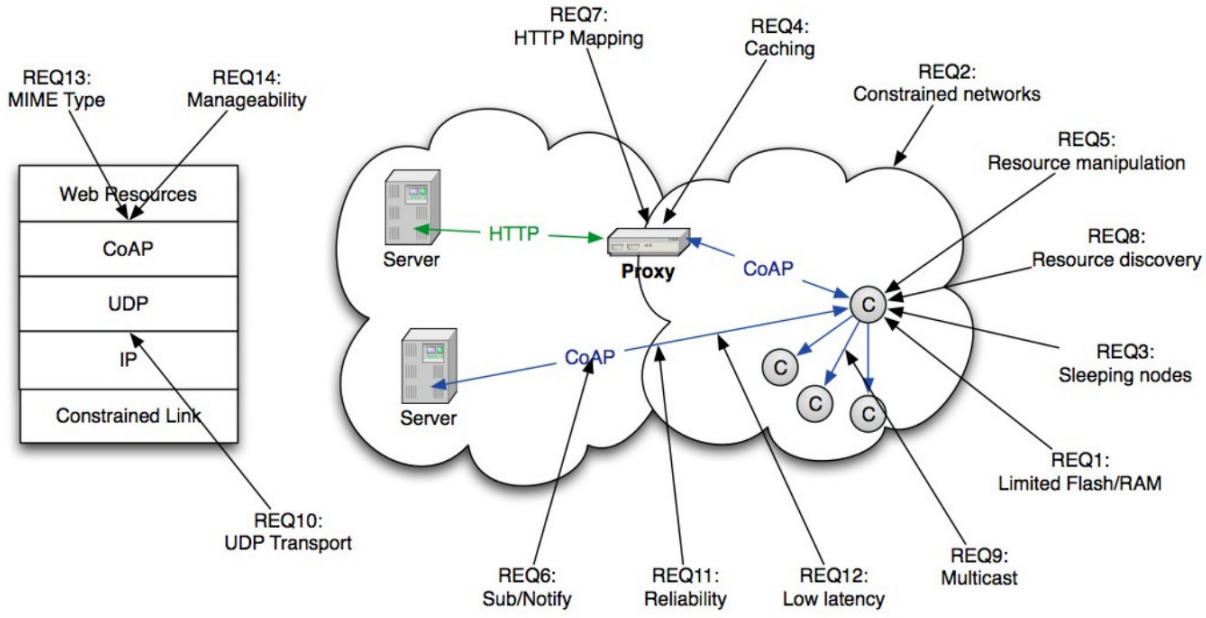
2.4 CoAP

O CoAP é um protocolo de camada de aplicação desenvolvido especificamente para comunicação entre dispositivos limitados computacionalmente, característica inerentes em um ambiente de Internet das Coisas. O CoAP é um protocolo que segue o modelo arquitetural cliente/servidor e que dá suporte à comunicação entre aplicações e OIs seguindo um paradigma requisição/resposta. Possui um serviço de descoberta já implementado, baseado no conceito de diretórios de dispositivos. Sua estrutura é baseada no HTTP, facilitando assim a integração com os recursos disponíveis na Web. Porém, diferente do HTTP, o CoAP cumpre alguns requisitos específicos para dispositivos com limitações computacionais, como o baixo *overhead* na troca de mensagens.

O CoAP possui três elementos básicos: i) o cliente CoAP, que pode ser tanto um dispositivo que consome os dados do servidor (e.g., *smartphone* controlando o ambiente) quanto

algum OI que produza dados (e.g., sensores em geral). ii) O Servidor CoAP gerencia os diversos clientes CoAP, bem como viabiliza a descoberta desse clientes, por meio de diretórios de recursos. iii) O *proxy*, que continua realizando seu objetivo tradicional de otimizar as consultas por recursos em um servidor que se encontra distante, embora no CoAP também é responsável por fazer o mapeamento e a tradução de mensagens HTTP em CoAP e vice-versa.

Figura 6 – Requisitos e funcionalidades do CoAP.



Fonte – Disponível em: http://coap.technology/

Durante a especificação do CoAP, diversos requisitos foram levados em consideração, como mostrado na Figura 6, de modo a melhor se adequar a um cenário no qual os dispositivos possuem limitações computacionais e energéticas. Cada um desses requisitos foram planejados com base em uma das entidades do CoAP. Dentre esses requisitos, alguns serviram como motivação para a escolha do CoAP como base para este trabalho. Destacam-se:

- **REQ 2 - Redes com Dispositivos Limitados:** O CoAP foi pensado de forma a se adequar às características específicas dos dispositivos com baixo poder computacional e baixa suficiência energética. Para isso foi reduzido o tamanho do cabeçalho das mensagens, com relação do HTTP, mantendo apenas informações essenciais, como identificadores de mensagens, campos de *checksum* e códigos de erro. O objetivo é deixar a camada de comunicação o mais leve possível, diminuindo o *overhead* na troca de mensagens.
- **REQ 3 - Nós em Modo de Espera:** O CoAP dá suporte a uma política de modo de

espera para dispositivos que precisam economizar recursos, principalmente energéticos. O número de mensagens trocadas pelos dispositivos que estão em modo de espera é extremamente reduzido, no qual são mantidas apenas as mensagens de controle, com taxa de envio reduzida, para identificar se o nó continua disponível na rede ou não.

- **REQ 6 - Paradigma *publisher/subscriber*:** Além de permitir trocas de mensagens de maneira síncrona, por meio das operações convencionais como GET e POST, o CoAP permite que clientes se registrem para serem notificados a cada mudança no estado de um recurso (e.g., temperatura), permitindo assim uma comunicação assíncrona entre nós da rede.
- **REQ 8 - Descoberta de Recursos:** Para possibilitar a comunicação entre clientes CoAP, o protocolo estabelece um serviço de descoberta baseado em diretórios de recursos, em que cada recurso pode ser acessado por um endereço único (e.g., coap://<coap server IP:Port>**/sensor/temp**).
- **REQ 10 - Transporte via UDP:** Uma das principais adaptações do CoAP com relação ao HTTP foi a mudança do protocolo da camada de transporte de *Transmission Control Protocol* (TCP) para UDP. Com isso é possível eliminar o *overhead* gerado pelos mecanismos de retransmissão e garantia de entrega das mensagens via TCP, otimizando assim a troca de mensagens entre dispositivos limitados.

O CoAP já possui inúmeras implementações disponíveis, para as mais diversas plataformas[6]. Isso é um bom indicativo que existem inúmeras soluções que são compatíveis com essa especificação. Com base nisso, criar novas soluções que também utilizem o CoAP pode garantir a interoperabilidade com essa gama imensa de dispositivos já ativos. Embora essa interoperabilidade não possa ainda ser estendida para IoT em geral, garantir a compatibilidade com o CoAP é uma estratégia que potencializa a interoperabilidade da solução proposta.

2.4.1 Troca de Mensagens

O CoAP dá suporte a quatro tipos de requisições, também existentes no HTTP, são elas: i) GET; ii) POST; iii) PUT; iv) DELETE. A funcionalidade de cada tipo de requisição também é semelhante ao HTTP, no qual o GET é utilizado para recuperar algum recurso no servidor, o POST pode ser utilizado em requisições mais complexas no qual é preciso enviar

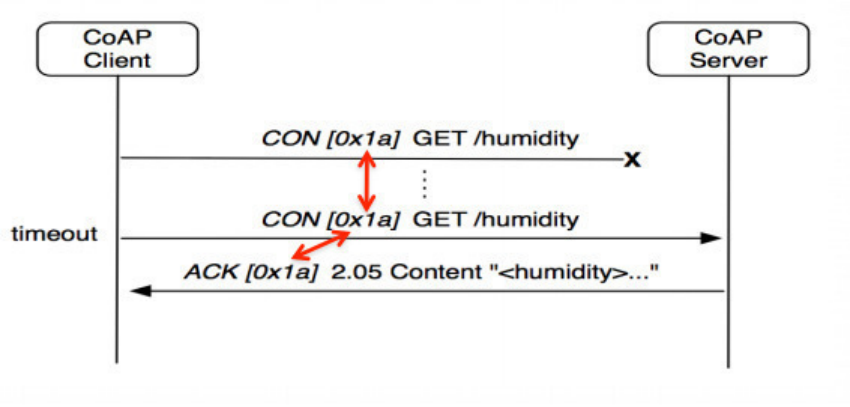
[6] http://coap.technology/impls.html

dados, geralmente sigilosos, no corpo da requisição. O PUT é utilizado quando o cliente deseja adicionar algum recurso ou parâmetro de configuração no servidor, já o DELETE é usado para deleção de recursos. Cada uma dessas requisições possui um cabeçalho de mensagens com tamanho fixo igual a quatro *bytes*, seguidas de um *payload* com tamanho também limitado. Um dos campos presentes no cabeçalho é um identificador único de mensagens, composto por 16 bits, o suficiente para identificar até 250 mensagens por segundos, utilizando os parâmetros padrões do CoAP.

Para garantir esse baixo *overhead*, o CoAP adota o *User Datagram Protocol* (UDP) como protocolo da camada de transporte. Embora o UDP não garanta a entrega de mensagens, o CoAP disponibiliza meios para uma troca de mensagens com essa garantia de entrega, por meio de políticas de confirmação e retransmissão de mensagens aplicadas na camada de aplicação. Existem dois parâmetros adicionais relacionados à confirmação de entrega de mensagens, quando o cliente explicitamente deseja que a mensagem seja confirmada, ou seja, que haja garantia na entrega ou pelo menos uma notificação em caso de erro, deve enviar uma mensagem do tipo CON (*confirmable*). Essas mensagens confirmáveis são geralmente utilizadas para controle (e.g., ligar ou desligar um ar-condicionado) ou para envio de dados altamente sensitivos a erros (e.g., dados de sensores médicos). A outra forma de se enviar mensagens é utilizando o parâmetro NON (*non confirmable*), dessa forma o cliente envia a mensagem e não espera nenhuma forma de confirmação de entrega, geralmente usada em *streamimg* de dados sensoriais comuns.

O identificador de mensagem é usado no processo de confirmação na entrega de mensagens. Cada mensagem do tipo CON, com um respectivo identificador de mensagem, espera receber uma resposta do tipo ACK (*acknowledgement*) com o mesmo identificador. A Figura 7 mostra um exemplo de retransmissão de mensagem utilizando CoAP, no qual um pacote contendo uma requisição do tipo GET é perdida. O cliente CoAP espera pela mensagem de ACK até um limite máximo de tempo, que pode ser configurado de acordo com as necessidades de cada cliente, caso não receba essa confirmação, reenvia a mensagem e repete o processo até receber uma confirmação do servidor, ou atingir o número máximo de retransmissões, que por padrão são quatro.

Figura 7 – Exemplo de retransmissão de mensagem usando o CoAP.



Fonte – Adaptado de (SHELBY *et al.*, 2014)

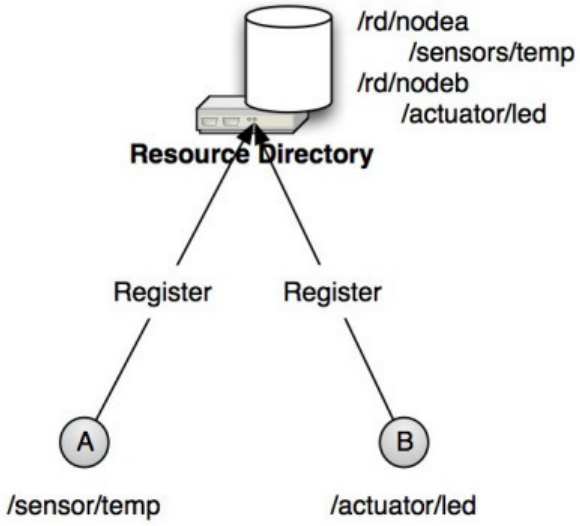
2.4.2 Serviço de Descoberta

Clientes utilizam diretórios de recursos para acessar o serviço de descoberta do CoAP. A busca inicial por servidores pode ser feita por meio de *multicast*, otimizando assim o processo de descoberta do servidor quando comparado a técnicas tradicionais utilizando *broadcast*. Geralmente esses diretórios de recursos estão presentes no mesmo dispositivo físico do servidor CoAP padrão, mas podendo ser uma máquina dedicada. Uma vez conhecendo o endereço do servidor que possui os diretórios de recursos, o cliente CoAP pode acessar a lista de dispositivos mantida por eles. Cada OI é representado por uma URI, seguindo o formato especificado pelo *Constrained RESTful Environments* (CoRE)[7]. Os OIs são responsáveis pelo seu registro nos diretórios de recursos, conforme indicado na Figura 8, e manter seu registro por meio de mensagens de *ping*, que indicam que um recurso continua disponível na rede. Clientes, por sua vez, podem consultar esses diretórios em busca da lista de recursos disponíveis em cada um dos dispositivos presentes naquele ambiente.

Em todo serviço de descoberta CoAP existe um recurso padrão, chamado *well-known/core*. Esse recurso tem por objetivo representar a lista de todos os demais recursos registrados no diretório de recursos presente no servidor. Clientes CoAP podem realizar uma operação do tipo GET para obter o valor do recurso *well-known/core*, desse modo, o cliente tem acesso a lista de endereços de cada um dos recursos disponíveis no ambiente, conforme indicado na Figura 9. Parâmetros adicionais, seguindo a especificação CoRE, podem ser passados junto à requisição para realizar uma filtragem mais específica, como por exemplo receber a lista de OIs

[7] https://tools.ietf.org/html/rfc6690

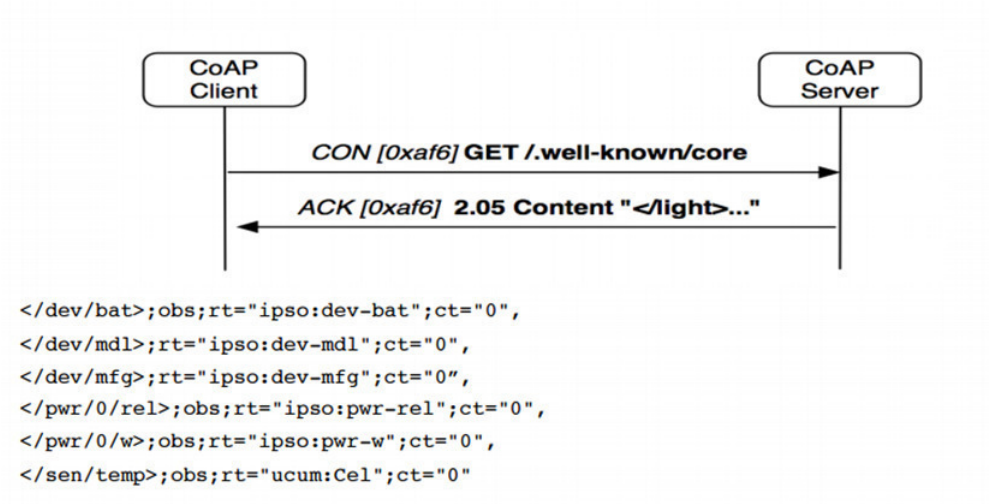
Figura 8 – Registro de OIs no diretório de recursos CoAP.



Fonte – Disponível em: http://coap.technology/

que possuam apenas sensores de temperatura.

Figura 9 – Cliente CoAP acessando a lista de recursos disponíveis no ambiente.



Fonte – Disponível em: http://coap.technology/

O CoRE define um conjunto de atributos que representam os recursos presentes em um diretório. Os principais atributos são: *Resource Type*, que é o responsável por identificar a função de um determinado recurso (e.g., temperatura, luminosidade, impressora, etc.); *Interface Description*, que indica os métodos que podem ser utilizados para a comunicação com esse recurso (*GET, POST*, etc); *Context Type*, que representa o formato dos dados fornecidos pelo recurso. Todos esses parâmetros estão presentes na URI que representa o recurso. Por exemplo,

um sensor (*interface description = sensor*) de temperatura (*resource type = temp*), que envia seus dados sensoreados seguindo um formato próprio dos dados, que deve ser conhecido pela aplicação a priori (e.g., sensor-json-temp-a) (*content type = sensor-json-temp-a*), poderia ser representado por: *</sensor/temp>; rt="temp"; if="sensor"; ct="sensor-json-temp-a"*

Cada um dos métodos que o CoAP dá suporte (GET, POST, PUT, DELETE) possui um papel específico quando usado para descoberta de recursos. OIs podem enviar requisições GET para o diretório de recursos para obter informações de outros recursos ou dele próprio (*lookup*). Requisições do tipo POST são utilizadas pelos OIs para se registrarem no serviço de descoberta, em que no corpo da mensagem é informado a URI e todos os demais atributos CoRE relativos aos recursos presentes no OI. Por sua vez, o PUT é utilizado para informar atualizações sobre os valores dos recursos presentes no OIs, por exemplo, sensores de luminosidade enviam sua *stream* de dados sensoreados com sucessivas requisições PUT. O DELETE é usado para remover a referência a um recurso que não mais existe no diretório de recursos, pode ser utilizado pelo próprio OI em caso de detecção de erro em algum sensor, ou por uma entidade externa que monitore o estado dos OIs presentes no ambiente.

2.5 Conclusão

Neste capítulo, foram apresentadas as bases teóricas que estruturaram o planejamento e o desenvolvimento do CoAP-CTX. As principais características e desafios de Internet das Coisas foram descritas, com foco nos requisitos específicos para uma descoberta eficiente de OIs em IoT. Esses requisitos serviram como critérios de análise e comparação dos trabalhos relacionados (descritos no próximo capítulo), bem como para direcionar o desenvolvimento da solução proposta. A pesquisa sobre arquiteturas de referência de IoT mostrou que ainda existe uma lacuna sobre serviços de descoberta de OIs.

Este capítulo também fundamenta duas escolhas importantes que foram feitas para esta pesquisa: a utilização do CoAP como protocolo de comunicação e do LoCCAM como *middleware* de suporte para aquisição de contexto. Na verdade, são muitas as complexidades de se implementar um serviço de descoberta eficiente para IoT, assim usar um serviço existente como base e então estendê-lo para atender aos novos requisitos se apresenta como uma excelente opção. Além disso, a interoperabilidade entre o maior número possível de OIs é uma característica desejável, ou seja, a adoção de um protocolo de comunicação bastante utilizado em IoT, como

o CoAP, é um diferencial importante para o CoAP-CTX. Atualmente, os dois protocolos de comunicação mais utilizados em IoT são o CoAP e o MQTT, mas apenas o primeiro possui o serviço de descoberta integrado, característica essa que determinou a escolha do CoAP para este trabalho.

Já a escolha do LoCCAM como *middleware* de aquisição de contexto e gerenciamento de OIs pode ser justificada por dois fatos: i) características sobre o contexto e; ii) experiência com a plataforma. O LoCCAM segue a mesma definição de contexto adotada por este trabalho e utilizada para criação do processo de descoberta sensível ao contexto da solução. Devido a isso, a integração entre CoAP-CTX e LoCCAM é natural, já que o contexto é modelado de forma simples, permitindo um fácil mapeamento entre informações contextuais e os parâmetros que descrevem os OIs compatíveis com o CoAP. O outro fator que influenciou na escolha do LoCCAM foi a baixa curva de aprendizagem com a plataforma, visto que o autor deste trabalho já trabalhou com o LoCCAM durante projetos de P&D em 2014 e 2015.

3 TRABALHOS RELACIONADOS

Neste capítulo, a Seção 3.1 apresenta e discute os trabalhos relacionados ao tema de descoberta de objetos inteligentes em um cenário de IoT. Foram selecionados trabalhos que propõem serviços ou infraestruturas de descoberta de OIs, com suporte a algum tipo de *matching* contextual ou filtragem contextual. Além disso, priorizou-se as abordagens que utilizam o serviço de descoberta integrado ao CoAP. Os trabalhos relacionados foram analisados e classificados na Seção 3.2, com base nos requisitos de descoberta de OIs em IoT (vide Seção 2.1.2). Por fim, a Seção 3.3 conclui o capítulo, identificando as tendências e as possíveis lacunas sobre descoberta sensível ao contexto de OIs.

3.1 Infraestruturas de Descoberta de Objetos Inteligentes

A descoberta de OIs é uma etapa essencial para qualquer sistema ou aplicação que objetiva interagir com os OIs presentes em um ambiente inteligente. Diversos trabalhos propuseram soluções para realizar essa descoberta, que vão desde a utilização de uma rede totalmente descentralizada, criando uma descoberta baseada em uma arquitetura *peer-to-peer*, até servidores centralizados especializados em obter informações de todos os OIs presentes em um ambiente. Seja qual for a arquitetura do serviço de descoberta, o mesmo deve atender aos requisitos inerentes de um cenário de IoT para permitir que sua utilização seja viável e eficiente em um ambiente real.

O levantamento dos trabalhos apresentados nesta seção foi um processo contínuo realizado durante os últimos dois anos. Três bases de dados foram utilizadas durante a busca por trabalhos: a *ACM Digital Library*, a *IEEE Xplore* e a *Science Direct*. Em cada uma delas, um conjunto de combinações entre as seguintes palavras-chave foram pesquisadas: *Internet of Things OR Web of Things*, *Things Discovery OR Smart Object Discovery OR Service Discovery*, CoAP, *Context-Aware OR Context-Awareness*. Após as filtragem com base nas leituras, primeiramente do resumo e depois dos artigos completos, cinco artigos foram selecionados. O principal critério de seleção foi a adequação do trabalho em pelo menos um dos requisitos de descoberta de OIs apresentados na Seção 2.1.2. Além disso, mais um artigo foi selecionado por indicação direta dos orientadores, completando um total de seis trabalhos relacionados que serão apresentados a seguir.

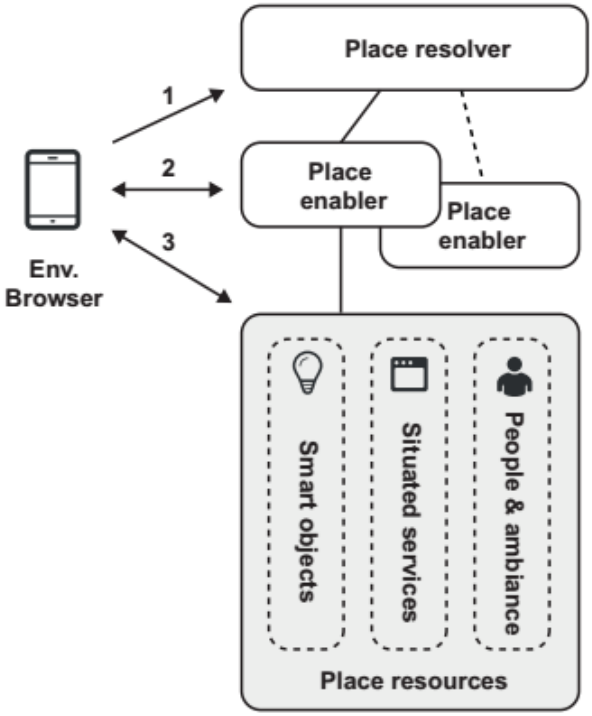
3.1.1 Env. B

Thebault *et* al. apresentam um protótipo de aplicação que funciona como um *browser* do ambiente (Env. B). Nessa aplicação, usuários podem interagir com OIs que estejam próximos (e.g., na mesma casa ou no mesmo restaurante). Para a aplicação encontrar e tornar os OIs disponíveis para o usuário, esses OIs tem que fornecer: i) uma descrição da interface gráfica (HTML); ii) uma API de acesso, juntamente com uma descrição sintática das operações que podem ser realizadas (e.g., ligar e desligar); e, iii) uma descrição semântica das operações (WSDL) (THEBAULT *et al.*, 2013). A descoberta de serviços é centralizada, com um único elemento sendo responsável por manter o registro de todos os OIs disponíveis para um determinado local.

A Figura 10 mostra como o *browser* do ambiente pode acessar os recursos presentes ao seu redor. É possível acessar diretamente os recursos, por meio dos componentes *Place Resources*, que possuem interfaces REST bem definidas e encapsulam os sensores e atuadores físicos presentes no ambiente. Outra forma de acesso é usando os componentes chamados Place Enablers, que representam a entidade virtual do ambiente em questão (e.g., casa, restaurante, carro). Esses componentes agregam a informação de todos os *Place Resources* presentes em um dado ambiente. Por fim, aplicações podem utilizar o *Place Resolver*, responsável por identificar que ambientes são relevantes ao usuário em um determinado contexto, por exemplo ambientes próximos ao usuário com base na localização obtida pelo GPS.

Os autores avaliaram a experiência do usuário na utilização do *browser* de ambiente. Para isso foi utilizado uma abordagem baseada em tarefas pré-determinadas. Um cenário com OIs reais foi montado dentro de um laboratório, simulando o comportamento desses dispositivos distribuídos na cidade. Um questionário foi respondido por 20 participantes, contendo perguntas sobre a sua experiência na utilização da aplicação. Os resultados mostraram que 60% dos usuários acharam a aplicação útil para ajudar das tarefas do dia a dia, enquanto cerca de 80% consideraram satisfatória a usabilidade da aplicação. Embora a avaliação do trabalho tenha sido extensa, não foi feita nenhuma análise sobre o desempenho do sistema, métricas como tempo de descoberta e consumo de energia foram deixados fora do escopo.

Figura 10 – Formas de Interação entre o Navegador de Ambiente e os Objetos Inteligentes.



Fonte – (THEBAULT *et al.*, 2013)

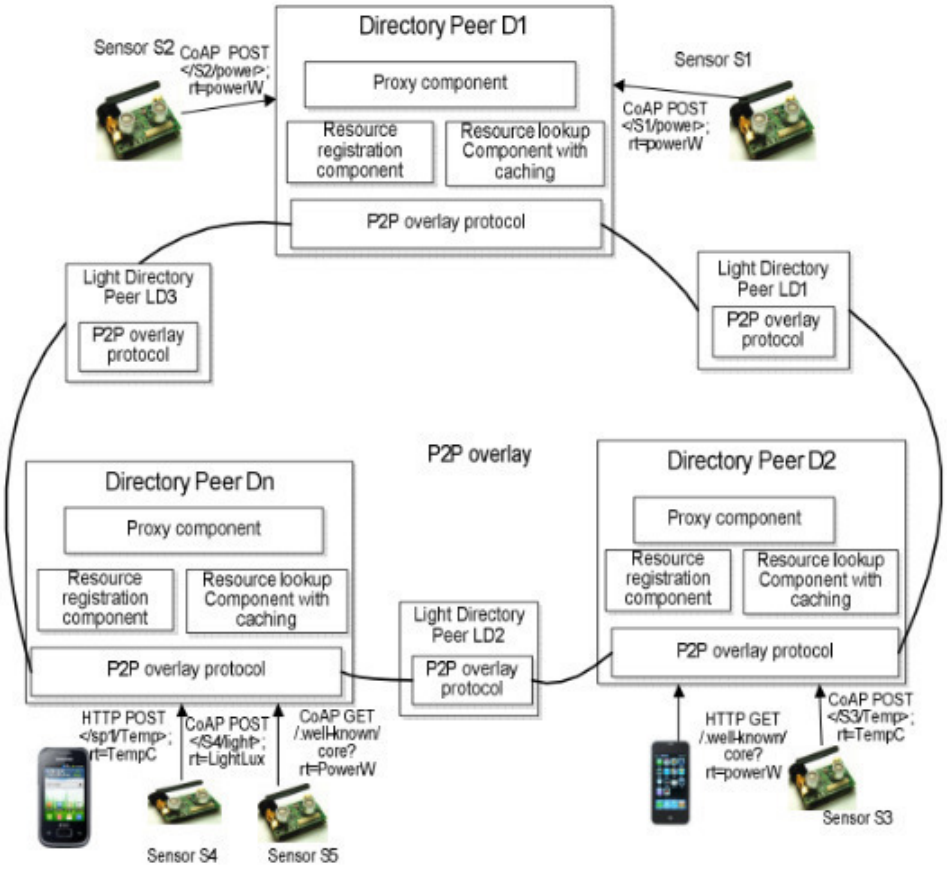
3.1.2 DRD4M

Liu *et al.* propõem uma arquitetura de descoberta distribuída de dispositivos focada em Internet das Coisas. A proposta do trabalho é criar uma arquitetura para um diretório de recursos distribuído para *Machine-to-Machine* (M2M), ou do inglês *Distributed Resource Directory for M2M (DRD4M)*. Cada dispositivo é considerado um nó de uma rede P2P e é capaz de realizar o processo de registro, bem como ajudar no serviço de descoberta geral (LIU *et al.*, 2013). A identificação de cada dispositivo é feita utilizando URIs que seguem o modelo CoAP, contendo o nome e o caminho até o dispositivo final. Para a geração desses nomes, é aplicada uma técnica sobre o endereço MAC de cada OI de modo a gerar identificadores únicos para esses dispositivos.

A arquitetura do DRD4M é composta basicamente por dois elementos principais, os chamados *Directory Peers* gerenciam o registro de novos OIs e o *lookup* dos mesmos, além de interpretar e executar requisições CoAP e HTTP. Já os *Light Directory Peer* são diretórios mais simples, em que apenas o *lookup* e o registro de OIs são suportados, não permitindo assim requisições diretas. Os *Light Directory Peer* são ideais para os dispositivos com baixo poder

computacional, geralmente dispositivos móveis, enquanto que os diretórios mais robustos podem servir como *gateway* para os demais dispositivos que formam a rede de *overlay P2P*. A Figura 11 apresenta a rede de *overlay* formada pelos dois tipos de diretórios e os diversos sensores e atuadores presentes no ambiente.

Figura 11 – Arquitetura do DRD4M.



Fonte – (LIU *et al.*, 2013)

Os autores criaram um protótipo simples para avaliar a solução, no qual o endereço dos diretórios são conhecidos a priori e os demais componentes enviam requisições do tipo POST para realizar o *lookup* dos OIs presentes. Foi analisado o tempo de resposta na realização do *lookup* usando o DRD4M com relação ao uso do *smartphone* como elemento centralizador. Os resultados mostraram que a arquitetura proposta reduz significativamente o tempo de descoberta quando o *lookup* é utilizado juntamente com uma estratégia de *cache* no registro dos OIs.

Com o uso dos *Light Directory Peers* e sua abordagem totalmente distribuída, os autores minimizaram os possíveis problemas causados pela mobilidade e volatilidade intrínsecos aos cenários de IoT. Por outro lado, cada dispositivo deve ser capaz de realizar o registro, além

de também dar suporte ao cache de recursos preferivelmente. Isso faz com que a sobrecarrega computacional no processo de registro de cada OI seja inevitável, o que pode ser um grande problema para dispositivos extremamente limitados em termos de capacidade processamento e memória.

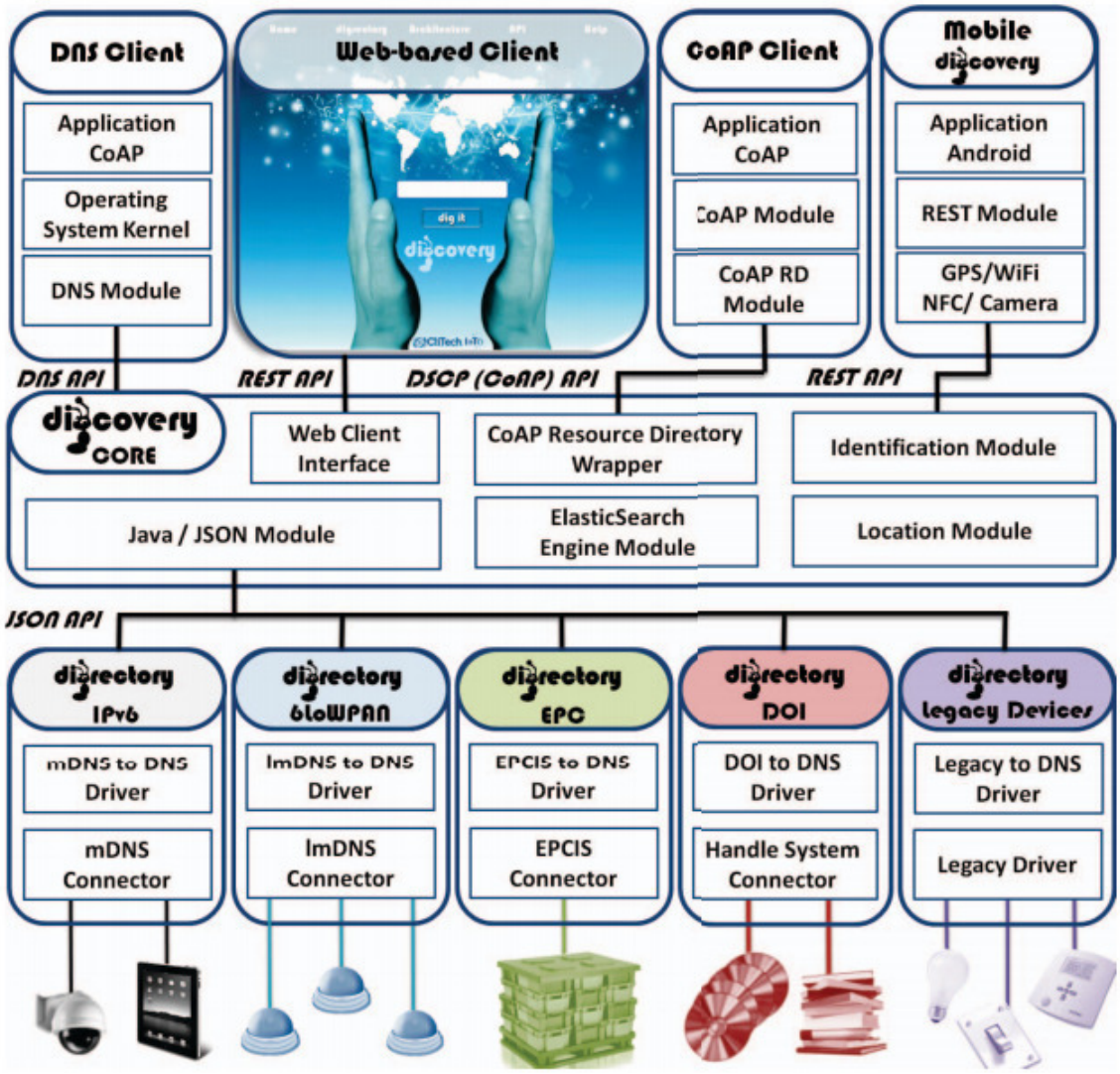
3.1.3 Digcovery

Digcovery (JARA *et al.*, 2013) é uma proposta de descoberta global de objetos inteligentes que se utiliza de uma infraestrutura centralizada na qual dispositivos podem se registrar. Para o acesso a essa infraestrutura, foi desenvolvido um serviço para dispositivos móveis, que permite a descoberta e o acesso aos OIs. Características contextuais de geo-localização são levadas em consideração durante a fase de descoberta. Nesse trabalho, tem-se o *smartphone* como ponto de acesso do usuário aos OIs. Diversas tecnologias de comunicação são suportadas, tais como RFID, NFC, Bluetooth, QR Codes, entre outras.

A arquitetura do Digcovery pode ser dividida em três camadas: i) Uma camada de aplicação, na qual clientes podem realizar a descoberta de OIs; ii) Uma camada de agregação, chamada de *Digcovery Core*, que reúne os dados de vários OIs, utilizando diversas tecnologias de comunicação, e que são posteriormente usados pelo mecanismo de busca para atender requisições de descoberta; iii) Uma camada de sensoriamento e atuação, cujos componentes chamados de *Drivers* e *Connectors* são usados em conjunto para encapsular e abstrair o acesso aos dispositivos reais. A Figura 12 mostra a arquitetura em questão, apresentando as diversas formas de realizar a descoberta, como um portal WEB, clientes CoAP, clientes DNS e a aplicação *Android* proposta.

Os autores realizaram uma avaliação funcional da solução, criando vários módulos para suportar as diversas tecnologias apresentadas. Os resultados mostraram que os OIs registrados no *Digcovery Core* podem, de fato, serem encontrados e controlados por todos os clientes, independente da tecnologia de comunicação. Entretanto, não foi realizada nenhuma avaliação de performance, ou seja, embora arquiteturalmente a solução seja extensível, nenhuma consideração sobre escalabilidade pode ser tomada. Uma das formas de interação é por meio de um cliente CoAP, que por definição possui um processo de registro de OIs leve, porém, a aplicação *Android* não utiliza esse cliente CoAP, e sim um protocolo próprio com base em REST. Isso implica no fato de que não é possível garantir que o serviço de descoberta proposto também continue leve, requisito importante para a descoberta de OIs em IoT.

Figura 12 – Arquitetura do Digcovery.



Fonte – (JARA *et al.*, 2013)

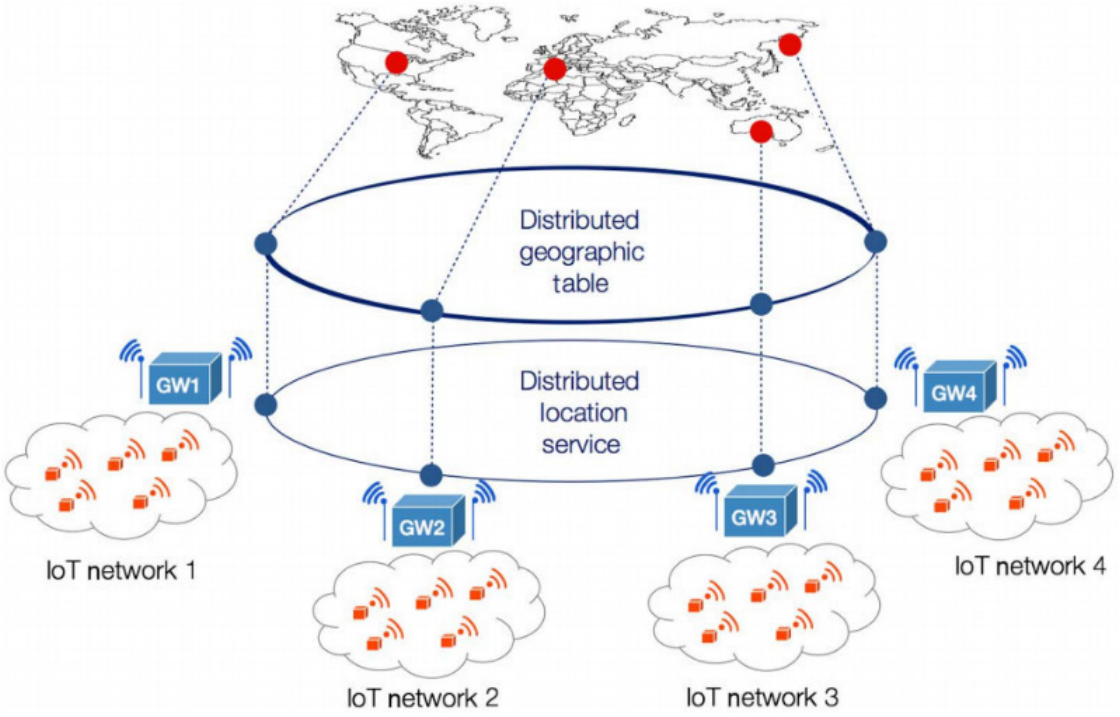
3.1.4 Cirani et. al.

Cirani *et al.* apresentam uma arquitetura auto-configurável e escalável de descoberta de serviços. No trabalho, é proposta uma topologia P2P com a utilização de diversos *gateways* que gerenciam o processo de descoberta. Eles são responsáveis por manter a lista de objetos inteligentes presentes na rede, permitindo o *lookup* dos mesmos por clientes da infraestrutura. Esses *gateways* são baseados nos servidores CoAP cujos clientes podem realizar uma operação GET para a URI */.well-know/core* para receber a lista de dispositivos disponíveis (CIRANI *et al.*, 2014).

A arquitetura do trabalho consiste na interconexão de redes P2P *overlay* por meio

dos *gateways*. Cada *gateway* monitora os OIs da sua sub-rede, e sempre que houver alguma alteração (e.g., novo OI se registrou no servidor CoAP), são enviadas mensagens indicando essa mudança para dois outros componentes: i) o *Distributed Location Service* (DLS) funciona como um serviço de nomes, semelhante ao *Domain Name System* (DNS), mas com algumas vantagens, como menor tempo de propagação e suporte a todos os elementos da URI; e ii) o *Distributed Geographic Table* (DGT) que permite que cada componente da rede recupere, de maneira eficiente, todas as informações dos OIs que estão localizados próximos a uma região geográfica. A Figura 13 ilustra como as diversas redes P2P são combinadas e formam os DLS, que por sua vez formam os DGT de modo a permitir um acesso global ao serviço de descoberta.

Figura 13 – Arquitetura do serviço global de descoberta proposto por Cirani *et al.*.



Fonte – (CIRANI *et al.*, 2014)

Os autores utilizaram duas métricas para avaliar o trabalho: i) O tempo de descoberta no cliente, que é o intervalo desde o envio da requisição até o recebimento da lista de OIs que satisfazem o critério da busca; ii) O tempo de descoberta no servidor, que é o tempo decorrido desde o recebimento da requisição do cliente, até a montagem e envio da resposta com a lista de OIs. Para uma topologia linear, com suporte a multi-salto, o tempo de resposta do cliente aumentou linearmente com o número total de OIs, pois a cada novo OI, um salto adicional na rede deveria ser feito. Ainda na topologia linear, o tempo de descoberta do servidor permaneceu

constante a medida que o número de OIs aumentava. Já para uma topologia em *grid*, ambos os tempos de descoberta permaneceram quase constantes, sendo a distância entre o cliente e o servidor o fator mais impactante para um atraso na descoberta.

A solução proposta por Cirani *et al.* pode ser considerada uma abordagem híbrida entre os trabalhos DRD4M e Digcovery, pois apresenta uma arquitetura descentralizada, mas com toda a descoberta sendo gerenciada exclusivamente pelos *gateways*. Isto minimiza os requisitos computacionais dos demais elementos da rede. Esse trabalho pode ser considerado como um processo de descoberta leve o suficiente para ser compatível com os requisitos de IoT, uma vez que o CoAP é a base para a descoberta e que clientes CoAP podem acessar diretamente o serviço de descoberta. Isso é comprovado com a vasta avaliação do trabalho. Ela mostrou que a solução é de fato escalável para cenários com até 1000 OIs simultâneos. Embora seja possível realizar uma busca por OIs baseada na localização dos mesmos, o cliente precisa especificar todos os parâmetros de geo-localização diretamente, então não se pode considerar o serviço de descoberta proposto pelos autores como sendo sensível ao contexto.

3.1.5 DiscoWOT

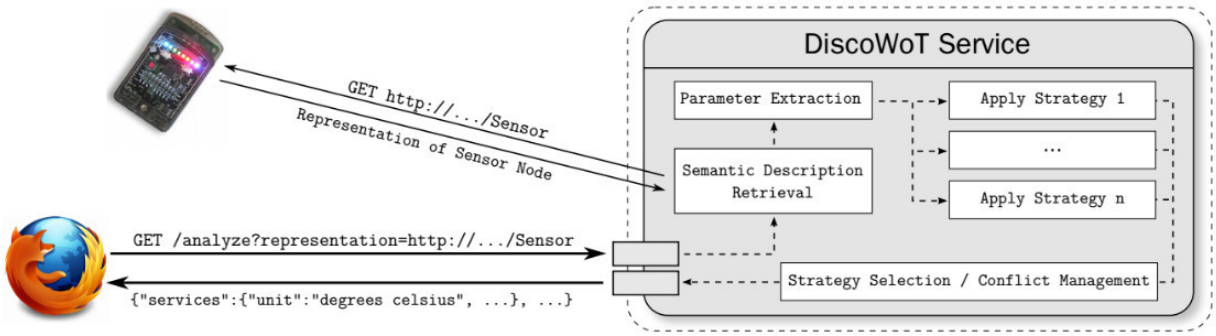
DiscoWoT é um serviço de descoberta de OIs proposto por (MAYER; GUINARD, 2011), em que é possível definir estratégias de descoberta em tempo de execução por meio de serviços RESTful, que são aqueles que seguem o modelo arquitetural REST. Esse serviço de descoberta se utiliza de *Microformats*[1] e *Microdata*[2] em conjunto com outras tecnologias WEB para representar semanticamente os OIs. Esses dados semânticos são representados em JSON, estratégia que potencializa a interoperabilidade.

O DiscoWoT é baseado em estratégias de descoberta que identificam semanticamente quais OIs satisfazem os parâmetros de uma determinada consulta. Essas consultas são feitas por meio de requisições HTTP do tipo GET, conforme ilustrado na Figura 14. O módulo *Semantic Description Retrieval* é responsável por interpretar as requisições com informações semânticas atreladas. Com base na requisição, são extraídos os parâmetros que são utilizados pelas estratégias (e.g., tipo do OI, localização, entre outros). As estratégias cadastradas em tempo de execução usam esses parâmetros para recuperar a lista de OIs que satisfazem aos

[1] http://microservices.io/patterns/microservices.html
[2] https://www.w3.org/TR/microdata/

critérios selecionados. Por fim, é realizada uma verificação de possíveis conflitos entre as listas, retornadas por cada estratégia, de OIs encontrados. Se houver conflito, um parâmetro de nível de confiança de cada estratégia é utilizado para decidir qual o valor prioritário.

Figura 14 – Interação entre clientes e o serviço de descoberta DiscoWoT.



Fonte – (MAYER; GUINARD, 2011)

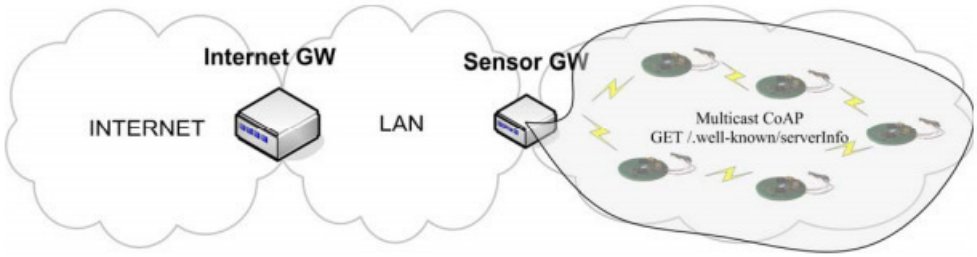
Os autores realizaram apenas uma avaliação funcional da proposta, desenvolvendo um sistema WEB para expor o serviço de descoberta do DiscoWoT aos clientes. Os autores mostraram que é possível utilizar a solução em três cenários: para a criação de infraestruturas genéricas de descoberta de OIs, para serviços de descoberta com tolerância a falhas e para uma descoberta *ad-hoc* de OIs. Além disso, o fato de que as estratégias podem ser adicionadas em tempo de execução faz com que a solução seja extensível, permitindo até a criação de estratégias com suporte a gerenciamento dos OIs com alocação dinâmica. Como a interação entre clientes e serviço de descoberta utiliza HTTP padrão e nenhuma avaliação de performance foi realizada, não é possível afirmar que o DiscoWoT atende aos requisitos de um serviço de descoberta leve o suficiente para se adequar ao paradigma de IoT.

3.1.6 Ishaq et. al.

Ishaq *et al.* propõem um mecanismo de descoberta baseado em CoAP e DNS. Traduções entre CoAP e HTTP são disponibilizadas, permitindo a descoberta de objetos inteligentes compatíveis com o padrão *IPv6*. Clientes se conectam ao serviço de descoberta por meio do endereço do servidor CoAP mais o identificador do OI, e a interação com os dispositivos se dá através de interfaces RESTful (ISHAQ *et al.*, 2012). A solução foi desenvolvida com base em duas premissas: i) A rede deve possuir uma topologia hierárquica, na qual os OIs se conectam a

um *Sensor Gateway* para ter acesso à rede local, que por sua vez estaria conectada a um *Internet Gateway*, responsável por disponibilizar o acesso dos OIs para toda a internet; e ii) Os OIs são dispositivos com o mínimo de processamento possível, todos os algoritmos de descoberta devem ser gerenciados pelos *gateways* e não pelos OIs. A Figura 15 mostra a topologia hierárquica proposta, na qual a descoberta de OIs é feita utilizando requisições *multicast* entre o servidor e clientes CoAP.

Figura 15 – Topologia proposta por Ishaq *et al.*, baseada nos *gateways* da internet e dos sensores.



Fonte – (ISHAQ *et al.*, 2012)

Para garantir a interoperabilidade entre CoAP e HTTP, os autores criaram um mecanismo de interpretação de mensagens seguindo o formato CoRE. A partir dessas mensagens, ocorre a geração automática de páginas HTML, que podem ser acessadas por qualquer navegador. Como a principal contribuição do trabalho era permitir que a descoberta de OIs fosse globalmente possível utilizando CoAP ou HTTP, a avaliação limitou-se a comprovar que essa descoberta era, de fato, possível. Para isso criaram um sistema Web, implementaram os *gateways* de internet e de sensores, e usaram o *framework* IDRA para implementar os OIs (POORTER *et al.*, 2011).

O diferencial do trabalho é o esforço empregado na garantia da interoperabilidade, incluindo o CoAP e o HTTP como protocolos possíveis, permitindo especialmente a descoberta de OIs por clientes Web, como navegadores. Semelhante ao trabalho de (CIRANI *et al.*, 2014), a proposta é utilizar o CoAP padrão como base para descoberta, sem nenhum alteração que possa comprometer a interoperabilidade do protocolo. Logo, pode-se afirmar que o trabalho herdou todos os benefícios do CoAP para a descoberta, como baixo *overhead* no registro e descoberta de OIs. Entretanto, nenhuma informação contextual é utilizada para otimizar a descoberta, cabendo aos clientes especificarem diretamente todos os parâmetros da busca por OIs.

3.2 Análise Crítica e Comparativa

Para comparar os trabalhos apresentados neste capítulo, foram utilizados os requisitos levantados por (GUINARD *et al.*, 2010) (vide Seção 2.1.2). Esses critérios foram escolhidos como base de comparação com intuito de identificar lacunas nos trabalhos que compõem o estado da arte. Mais especificamente, a solução ideal para um serviço de descoberta deverá atender a todos os requisitos de descoberta de OIs para IoT levantados por Guinard.

A seguir, são detalhados individualmente cada requisito, e os critérios usados para classificar um trabalho que atende ou não a esse requisito:

1. **REQ 1 - Mínimo *Overhead* no serviço de descoberta:** foram analisados aqui as etapas necessárias para que clientes obtenham a lista de OIs encontrados. Os componentes analisados foram os clientes e servidores, que possuem a lista de OIs disponíveis. Considera-se que um trabalho atende a esse requisito se a arquitetura proposta foi desenvolvida para minimizar os recursos gastos na descoberta e o tempo para realizar essa descoberta. Com isso, considerou-se que atendiam a esse requisito trabalhos que possuíam um cliente e um servidor CoAP padrão como base de descoberta, ou que faziam uma avaliação detalhada sobre a performance da descoberta. Entre os trabalhos apresentados neste capítulo, apenas o Digcovery e o DiscoWoT não foram selecionados, pois o cliente da descoberta era baseado em requisições HTTP convencionais, e nenhuma avaliação detalhada sobre performance foi realizada.
2. **REQ 2 - Mínimo esforço no registro de OIs:** para esse requisito foram levadas em consideração as etapas que um OI precisa realizar para se registrar em um servidor de recursos. Os dispositivos analisados foram os OIs e o servidor. Considerando os OIs limitados computacionalmente, o mínimo possível de processamento no processo de registro deve ser feito, o que já exclui a pesquisa DRD4M, pois nela cada OI deve ser capaz também de gerenciar esse processo de registro. Outros trabalhos como o Env. B e DiscoWoT não apresentaram nenhuma avaliação que comprove um bom desempenho no processo de registro dos OIs. Os demais trabalhos são baseados no CoAP, que por sua vez foi desenvolvido seguindo princípios que objetivam esse baixo *overhead* no registro, e já possui avaliações que comprovam sua eficiência (THANGAVEL *et al.*, 2014).
3. **REQ 3 - Suporte à busca contextual e dinâmica de OIs:** esse requisito é atendido por trabalhos que são capazes de utilizar alguma informação contextual para reduzir a lista de

OIs encontrados. A informação contextual mais utilizada dentre os trabalhos estudados foi a localização, nos trabalhos Env. B, Digcovery e DiscoWoT. Outra classificação com relação ao contexto foi feita nessa seção, indicando se o trabalho obtinha e utilizava as informações contextuais do usuário, dos OIs, ou de ambos.

4. **REQ 4 - Suporte à alocação de OIs sob demanda:** dentre todos os trabalhos analisados nesse capítulo, nenhum deu suporte total a esse requisito, em que os OIs deveriam ser alocados, ou entrar em modo ativo, com base nos requisitos da aplicação. Entretanto, vale ressaltar que a solução DiscoWoT se propôs a criar estrategias de descoberta em tempo real. Essa abordagem permite a utilização de uma descoberta com suporte à alocação sob demanda, embora os autores não citaram essa possibilidade em seu estudo.

A Tabela 1 sumariza os trabalhos relacionados classificados com relação aos requisitos da descoberta de OIs em IoT:

Tabela 1 – Comparativo entre os Trabalhos Relacionados.

Trabalhos Relacionados	REQ 1	REQ 2	REQ 3	REQ 4
Env. B	-	✓	Usuário + OI	-
DRD4M	-	✓	-	-
Digcovery	✓	-	Usuário	-
Cirani et al.	✓	✓	-	-
DiscoWoT	-	-	OI	✓
Ishaq et al.	✓	✓	-	-

Fonte – Elaborado pelo autor

3.3 Conclusão

Este capítulo apresentou uma comparação entre os trabalhos relacionados, realizando uma classificação com base nos requisitos levantados por (GUINARD *et al.*, 2010) para uma descoberta otimizada de OIs em IoT. É possível perceber que técnicas de sensibilidade ao contexto ainda não são amplamente utilizadas nas soluções de descoberta de OIs.

Dos três trabalhos que utilizam informações contextuais para realizar a descoberta, apenas o *Environment Browser* reuniu dados contextuais de usuário e OI simultaneamente. Outro

ponto importante é a ampla adoção do CoAP como base para o serviço de descoberta, pois quatro dentre os seis trabalhos seguem esse protocolo para descobrir e interagir com os OIs. Embora o CoAP já tenha suporte para OIs em modo de espera, apenas o DiscoWoT possui uma arquitetura que suporta uma estratégia de alocação dinâmica de OIs.

Uma análise comparativa mais profunda dos trabalhos citados neste capítulo foi dificultada pela falta de acesso ao código da solução. Apenas um dos trabalhos, DiscoWoT, disponibilizou a solução para download e utilização pela comunidade.

Nenhum trabalho relacionado atende a todos os requisitos listados. Visto isso, a solução proposta neste trabalho de Mestrado é criar um mecanismo de descoberta de OIs, baseado em CoAP, que atenda a todos esses quatro requisitos.

4 COAP CONTEXTUAL (COAP-CTX)

Neste capítulo, o CoAP-CTX (CoAP-Contextual) é apresentado. A solução consiste em uma extensão do serviço de descoberta padrão do CoAP que agrega informações contextuais ao processo de descoberta de OIs. A Seção 4.1 apresenta a visão geral do CoAP-CTX, incluindo os princípios que guiaram seu desenvolvimento. Na Seção 4.4, é descrito o processo proposto de descoberta sensível ao contexto cuja entrada é o contexto do usuário e a saída é a lista de OIs relevantes. Partindo desse processo, na Seção 4.2 a arquitetura do CoAP-CTX é apresentada, com os principais elementos que a compõem e suas interações. Detalhando a arquitetura, a Seção 4.3 discute a etapa de *matching* contextual, trazendo um exemplo de um filtro de OIs baseado em localização. A Seção 4.5 detalha a integração do CoAP-CTX com o LoCCAM, de modo a permitir a aquisição do contexto e o controle dos OIs. A Seção 4.6 mostra como os OIs entram em modo de espera caso não sejam do interesse do usuário em um determinado contexto. Por fim, a Seção 4.7 conclui o capítulo com as considerações finais sobre o que foi apresentado.

4.1 Visão Geral

O CoAP-CTX (CoAP-Contextual) é uma extensão do serviço de descoberta padrão do CoAP que agora considera em seu processo de descoberta de OIs informações contextuais tanto do usuário quanto dos OIs presentes no ambiente. O CoAP-CTX, embora seja um serviço de descoberta, também atua na camada de comunicação, de acordo com a arquitetura de referência apresentada na Seção 2.1.

O foco do CoAP-CTX é fornecer para criadores e desenvolvedores de aplicações do tipo controle universal de ambientes, ou *browsers* de ambiente, um serviço de descoberta que seja capaz de selecionar os OIs mais adequados ao interesse do usuário em um determinado contexto. Isso reduz a sobrecarga no número final de OIs apresentados na interface visual, o que ajuda e facilita a utilização da aplicação por parte dos usuários finais. Além disso, no ponto de vista de economia de recursos, o CoAP-CTX consegue reduzir o número total de mensagens trocadas na rede, melhorando assim a eficiência energética dos OIs. Entretanto, o CoAP-CTX não limita-se a esse tipo de aplicação, podendo ser utilizado por qualquer aplicação que precise realizar a descoberta de OIs compatíveis com o CoAP.

Baseado na fundamentação teórica em IoT, nas características do CoAP, nos requisitos de descoberta apresentados por (GUINARD *et al.*, 2010) e nos princípios apresentados por (DATTA; BONNET, 2015), foram definidos 7 princípios de *design* que são seguidos no desenvolvimento do CoAP-CTX:

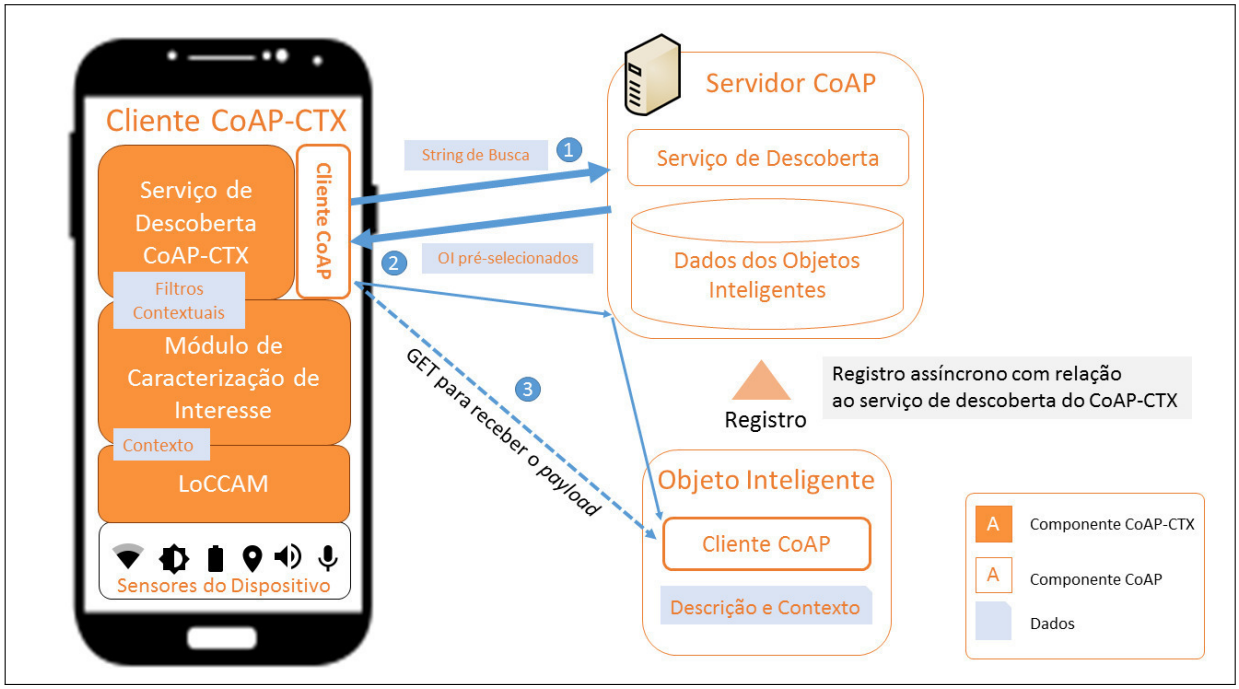
1. **Compatibilidade máxima com o CoAP.** O CoAP-CTX deve manter a máxima compatibilidade com a especificação CoAP padrão. Servidores e OIs compatíveis com o CoAP devem também ser compatíveis com o CoAP-CTX;
2. **Dispositivo móvel dos usuários como coordenador da descoberta.** O *smartphone* do usuário, ou dispositivo móvel similar, deve ser utilizado como dispositivo coordenador do processo de descoberta. Esse dispositivo deve adquirir as informações contextuais do usuário e utilizá-las para realizar a descoberta sensível ao contexto;
3. **Filtragem de OIs baseada em informações contextuais dos usuários e OIs.** O CoAP-CTX deve combinar as informações contextuais do usuário e dos OI para realizar a descoberta seletiva desses OIs;
4. **Expansão da descrição dos OIs.** A descrição padrão dos OIs que seguem a especificação CoRE deve ser mantida. Todas as informações contextuais adicionais devem estar presentes no *payload* do OI, cabendo ao CoAP-CTX extrair e analisar essas informações;
5. **Representação de contexto baseado em chaves hierarquizadas.** A representação do contexto deve seguir uma estrutura hierárquica na forma de árvore, onde cada nó representa uma chave contextual, ou do inglês *context-key*;
6. **Algoritmo de filtragem de OIs baseado em *matching* contextual.** O CoAP-CTX deve selecionar os OIs por meio de um processo de *matching* contextual. Um conjunto de filtros contextuais devem ser aplicados a todos os OIs, e apenas os que passarem em todos os filtros deverão ser retornados pelo serviço de descoberta; e
7. **Gerenciamento de OIs.** O CoAP-CTX deve otimizar o gerenciamento dos OIs de modo a diminuir a carga computacional e energética. Essa otimização deve acontecer em virtude da redução do número de mensagens trocadas na rede pelos OIs.

4.2 Arquitetura

A solução proposta foi desenvolvida de forma a atuar em ambientes inteligentes que possuam três tipos de elementos básicos: um *smartphone* Android, um ou mais servidores CoAP

e um conjunto de objetos inteligentes. O *smartphone* funciona como o elemento coordenador de todo o processo de descoberta, gerenciando assim todos os OIs que são de interesse do usuário. A Figura 16 apresenta a arquitetura do CoAP-CTX.

Figura 16 – Arquitetura do CoAP-CTX.



Fonte – Elaborado pelo Autor.

O cliente CoAP-CTX, executando no *smartphone*, é composto por quatro componentes: i) o *middleware* LoCCAM, responsável por adquirir e fornecer as informações contextuais sobre o usuário (e.g., o usuário está em casa ou esta dirigindo); ii) o módulo de caracterização do interesse, responsável por listar os OIs de interesse para um determinado contexto; iii) o serviço de descoberta do CoAP-CTX, onde são criadas as requisições CoAP que correspondem ao interesse do usuário, bem como é realizado o *matching* contextual; iv) o cliente CoAP tradicional, capaz de interagir com o servidor CoAP seguindo a especificação original. Dessa forma, toda a interação com os demais elementos CoAP (servidores e OIs) é feita estritamente pelo cliente CoAP, o que garante a interoperabilidade entre CoAP-CTX e CoAP. Entretanto, para uma melhor eficiência do serviço de descoberta, a descrição e a formatação dos dados dos OIs devem ser conhecidas pelo CoAP-CTX, mais especificamente para a aplicação dos filtros contextuais. Caso essas informações não sejam conhecidas, o *matching* contextual não terá nenhum efeito e o serviço de descoberta se tornaria equivalente ao do CoAP padrão.

4.2.1 Representação e Aquisição do Contexto

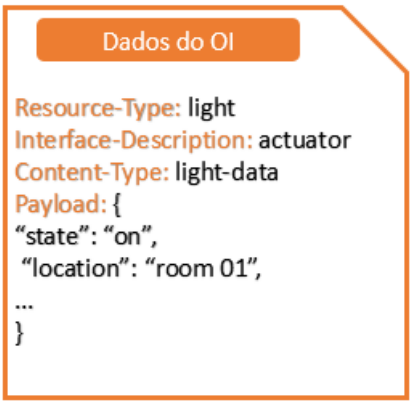
Na implementação atual do CoAP-CTX, assume-se que os dados contextuais seguem o mesmo formalismo utilizado pelo LoCCAM. Informações contextuais são compostas por uma série de chaves textuais separadas por pontos. A primeira chave é sempre "context"ou "control", especificando se a informação contextual é referente à um elemento que realiza o sensoriamento do ambiente ("context") ou que atua no ambiente ("control"). A segunda chave representa o dispositivo, ambiente ou entidade que está sendo monitorado ou controlado (e.g., "ambient", "device", "smartphone", "user", entre outros.) A terceira chave representa o tipo específico do dispositivo que está sendo descrito, para um sensor de temperatura por exemplo, o valor da terceira chave seria "temperature". As demais chaves são informações específicas sobre determinada informação contextual, e depende de cada tipo de contexto, como a localização do OI ou a escala usada para medir temperatura (e.g., Celsius, Kelvin, Fahrenheit). A informação contextual de um atuador que controla a TV do quarto de uma casa inteligente, por exemplo, pode ser representada por "control.ambient.tv.quarto". Essa forma de representação é chamada de *context-keys*.

A primeira tarefa do sistema é obter o contexto por meio do LoCCAM. Este, por sua vez, detecta a situação em que o usuário se encontra (e.g., usuário chegou em casa) e fornece as informações contextuais suficientes para caracterizar seu interesse naquele momento. Com base nessas informações contextuais, o Módulo de Caracterização de Interesse constrói uma lista de *context-keys* que representam de forma simples o interesse do usuário dado um determinado contexto. Por exemplo, se o usuário tem interesse em acessar a TV e o mesmo se encontra no quarto, esse interesse é representado por *control.ambient.tv* e *context.user.location.quarto*.

Com essa representação do interesse, o Serviço de Descoberta do CoAP-CTX realiza um mapeamento entre interesse e *String* de consulta CoAP. Cada campo das *context-keys* é mapeado para campos que definem e descrevem os OIs que seguem a especificação CoRE, como apresentado na Figura 17. Os campos "control"e "ambient"dizem que o objeto deve ser capaz de atuar no ambiente, o que pode ser representado por um filtro no atributo *Interface-Description* do OI, limitando seu valor para apenas *actuator*. Já o campo *tv* da *String* de interesse, representa o tipo, ou classe, do objeto inteligente. Esse campo pode ser mapeado para o atributo *Resource-Type*, aceitando apenas OIs que possuam esse atributo igual a *tv*. Com isso, é possível gerar uma requisição, a partir do Cliente CoAP, para o Servidor CoAP em busca de todas as TVs que

podem ser controladas pelo usuário.

Figura 17 – Exemplo de descrição de um Objeto Inteligente seguindo a especificação CoRE.



Fonte – Elaborado pelo Autor.

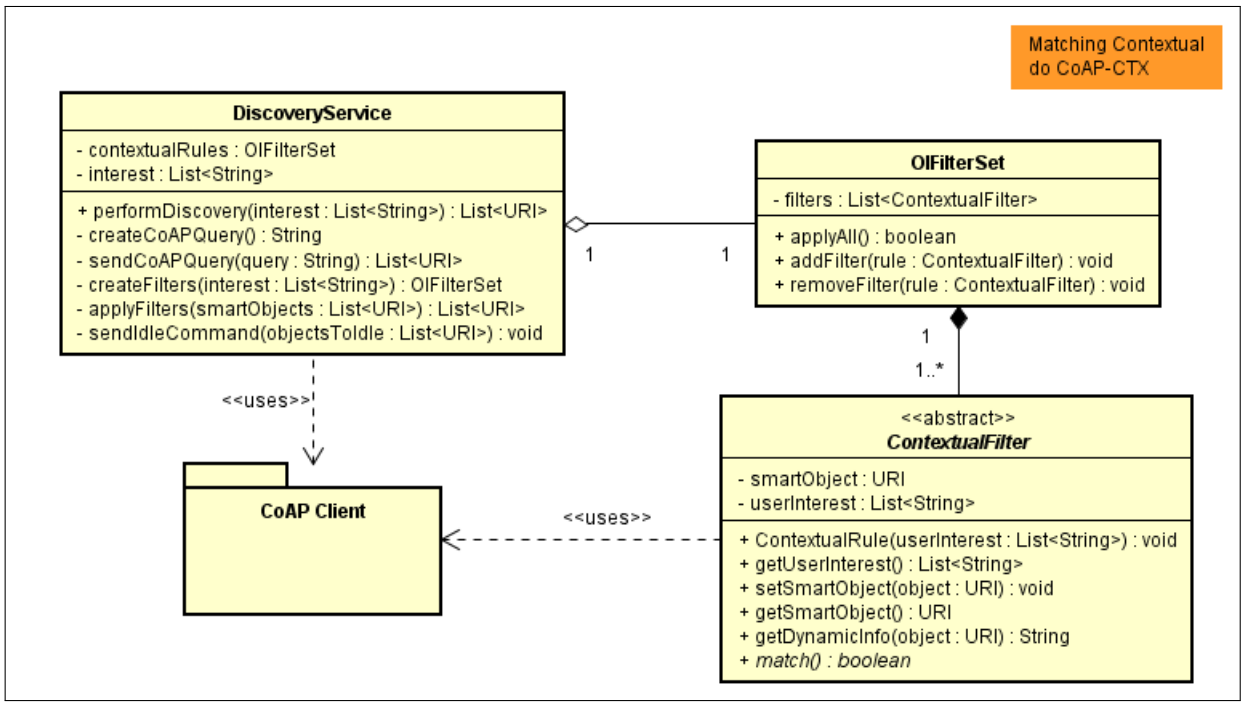
A atualização da descrição de um OI se dá por meio de mensagens CoAP do tipo PUT, informando um novo estado para os parâmetros dinâmicos, presentes no *payload* do OI. Já os campos estáticos, como os definidos previamente no CoRE, não podem ser atualizados a qualquer momento, sendo necessária uma mudança no *firmware* do OI para realizar essa atualização.

4.3 *Matching* Contextual

Uma das etapas mais importantes do serviço de descoberta do CoAP-CTX é a aplicação dos filtros contextuais durante a etapa de *matching*. Nessa etapa, são analisadas todas as informações contextuais dinâmicas dos OIs pré-selecionados pelo servidor CoAP. O contexto dinâmico é descrito no *payload* dos OIs por meio de um conjunto de campos do tipo chave-valor (*key-value*).

Os filtros contextuais são aplicados pelo serviço de descoberta do CoAP-CTX logo após o recebimento da lista de OIs pré-selecionados. A Figura 18 apresenta o diagrama de classes do serviço de descoberta e as entidades relacionadas ao *matching* contextual, que são a *OIFilterSet* e *ContextualFilter*. Durante a criação dos filtros contextuais, o serviço de descoberta inicializa todas os filtros a serem aplicadas a todos os OIs pré-selecionados. Esses filtros são gerenciados pela classe *OIFilterSet*, que por sua vez é utilizada para selecionar os OIs que são realmente de interesse do usuário, ou seja, cada OI é submetido a todos os filtros contextuais antes que seja selecionado e disponibilizado ao usuário.

Figura 18 – Diagrama de classes do *matching* contextual do CoAP-CTX.



Fonte – Elaborado pelo Autor.

Cada filtro contextual possui um único OI de referência, no qual a informação contextual dinâmica é analisada para verificar se há ou não um *match* entre OI e o contexto do usuário. Durante o processo de *matching*, o objeto *ContextualFilter* realiza uma operação de GET ao servidor CoAP, por meio do método *getDynamicInfo()*, para obter o *payload* do OI. A informação contextual é extraída do *payload* e comparada com o valor de referência, que é obtido do interesse do usuário. Caso os valores obtidos do OI e do interesse do usuário sejam compatíveis, esse OI é selecionado e disponibilizado ao usuário. A Figura 19 traz um exemplo de um filtro contextual de localização, no qual é verificado se o OI se encontra no mesmo local que o usuário. Note que a comparação implementada é uma simples conferência da igualdade dos valores de descrição (e.g., "livingRoom"). Medidas de similaridade mais complexas ou outros algoritmos de *matching* mais rebuscados poderiam ser incluídos por meio da extensão desses filtros.

4.4 Processo de Descoberta Sensível ao Contexto de OIs

A arquitetura de descoberta de OIs proposta segue um processo de descoberta sensível ao contexto que pode ser dividido em 8 etapas. (1) Aquisição do contexto do usuário, obtendo assim as informações contextuais que disparam o início do processo; (2) Inferência

Figura 19 – Implementação JAVA do filtro contextual baseado em localização.

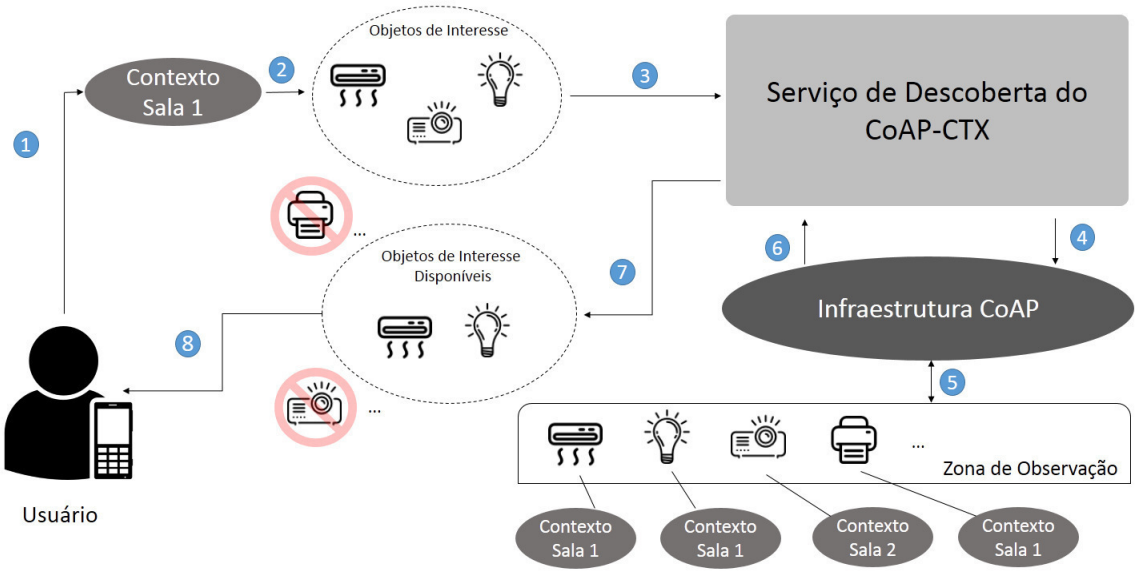
```java
public class LocationFilter extends ContextualFilter{

 public LocationFilter(List<String> userInterest) {
  super(userInterest);
 }

 @Override
 public boolean match() {
  String location = extractUserLocation(getUserInterest()); // User location
  String response = getDynamicInfo(getSmartObject()); // Smart Object payload
  response = extractLocationFromResponse(response); // Smart Object location
  // Match check
  if("Unknown".equals(location) ||
     "Unknown".equals(response) ||
      location.equals(response)){
   return true;
  }
  return false;
 }

 private String extractLocationFromResponse(String response) {
  int index = response.indexOf("location"); // Finding the location field
  if(index != -1){
   String location = response.substring(index);
   location = location.split(" ")[0]; // extracting location field
   location = location.split(":")[1]; // extracting location value
   return location;
  }
  return "Unknown";
 }

 private String extractUserLocation(List<String> userInterest) {
  // Verifying if there is information about user location
  for(String interest : userInterest){
   if(interest.contains("user.location")){
    // context.user.location.livingRoom => livingRoom
    int index = interest.split(".").length;
    return interest.split(".")[index - 1]; // extracting the last key
   }
  }
  return "Unknown";
 }
}
```

dos objetos inteligentes de interesse; (3) Representação e identificação dos OIs de interesse; (4) Criação de *strings* de consulta CoAP que realizam uma pré-filtragem; (5) Busca por OIs que satisfaçam os filtros da consulta CoAP; (6) Aquisição da lista de objetos inteligentes disponíveis no ambiente e que são de interesse do usuário; (7) Criação das abstrações de acesso aos OIs. (8) Listagem dos OIs selecionados, que representam a saída do processo. Todas essas etapas podem ser visualizadas na Figura 20.

O processo de descoberta é analisado com base no cenário motivador apresentado na Seção 1.2. Na primeira etapa (1), as informações que descrevem o contexto atual de Ana (e.g., localização, situação, atividade, etc.) são obtidas e utilizadas para a identificação de quais OIs são de interesse de Ana em um dado contexto (2). A Figura 20 mostra o interesse de Ana no

cenário da empresa, em que os OIs relevantes são os que possuem as seguintes funcionalidades: refrigeração (ar-condicionado), iluminação (lâmpadas) e apresentação (projetor). Cada OI é representado por meio de uma única *String* que agrega informações contextuais e que pode ser utilizada para identificação desse dispositivo (3). Como os OIs utilizam o CoAP como protocolo de comunicação e interação, pode-se utilizar mecanismos já disponíveis nesse protocolo, como as *strings* de consulta CoAP, geradas na etapa (4). Nesse caso, um exemplo de consulta ao servidor CoAP seria: *GET /.well-known/core?rt="air-conditioner light projector"*.

Figura 20 – Visão geral do processo de descoberta de objetos inteligentes.



Fonte – Elaborado pelo Autor.

A referência dos OIs que se registraram no servidor CoAP (5) e que atenderam às restrições expressas na consulta realizada é retornada ao mecanismo de descoberta (6). Nessa etapa a impressora é descartada, pois não atende ao filtro especificado pelo *Resource Type* na consulta CoAP. Antes de gerar a lista final de objetos relevantes e disponíveis (7), é preciso realizar o *matching* contextual, onde são verificadas as informações dinâmicas do contexto do usuário e dos OIs, como localização (e.g., Ana e OIs estão na mesma sala da empresa). Nessa etapa o projetor é descartado, por que não está na mesma localização de Ana. Por fim, a lista final de OIs é apresentada a Ana (8), que por sua vez poderá interagir com esses OIs e realizar operações do cotidiano mais facilmente. Os OIs que estão disponíveis na zona de observação mas foram descartados da lista final da descoberta são postos em modo de espera até que uma próxima descoberta se inicie.

O algoritmo apresentado na Figura 21 resume o processo de descoberta sensível ao contexto apresentado nesta seção. O contexto do usuário é utilizado como entrada para o processo, isso quer dizer que o CoAP-CTX não impõe ou limita nenhuma forma de aquisição de contexto. A partir do contexto, é determinada a lista de OIs de interesse, e com essa lista é realizada uma consulta CoAP para retornar todos os OIs cuja descrição são compatíveis com os OIs de interesse. Para cada OI, é realizada uma leitura do *payload* a fim de obter as informações contextuais dinâmicas, como localização, e a partir daí realizar o *matching* contextual, de forma a selecionar apenas os OIs que são realmente relevantes ao usuário final. Por fim os OIs que não foram selecionados são postos em modo de espera para economizar recursos.

Figura 21 – Algoritmo utilizado para a descoberta sensível ao contexto.

Algoritmo 1: Descoberta Sensível ao Contexto

```
Data: Contexto
Result: Lista de OIs descobertos
inicialização;
inferir lista de OIs de interesse;
realizar consulta CoAP;
for OIs retornados pela consulta CoAP do
    leitura do payload;
    verifica o matching contextual;
    if passar no matching then
        adiciona o OI na lista de descobertos;
    else
        coloca o OI em modo de espera;
    end
    passar para próximo OI;
end
retorna a lista de OIs descobertos;
```

Sobre a compatibilidade do processo de descoberta com relação ao CoAP padrão, nenhuma mudança na especificação é necessária para cumprir todas as etapas. É possível apenas estender os mecanismos de busca e descoberta já integrados no protocolo base, agora agregando informações contextuais baseadas tanto na descrição do OI (estáticas) quanto nos seus dados (*payload* dinâmico).

4.5 Integração com o LoCCAM

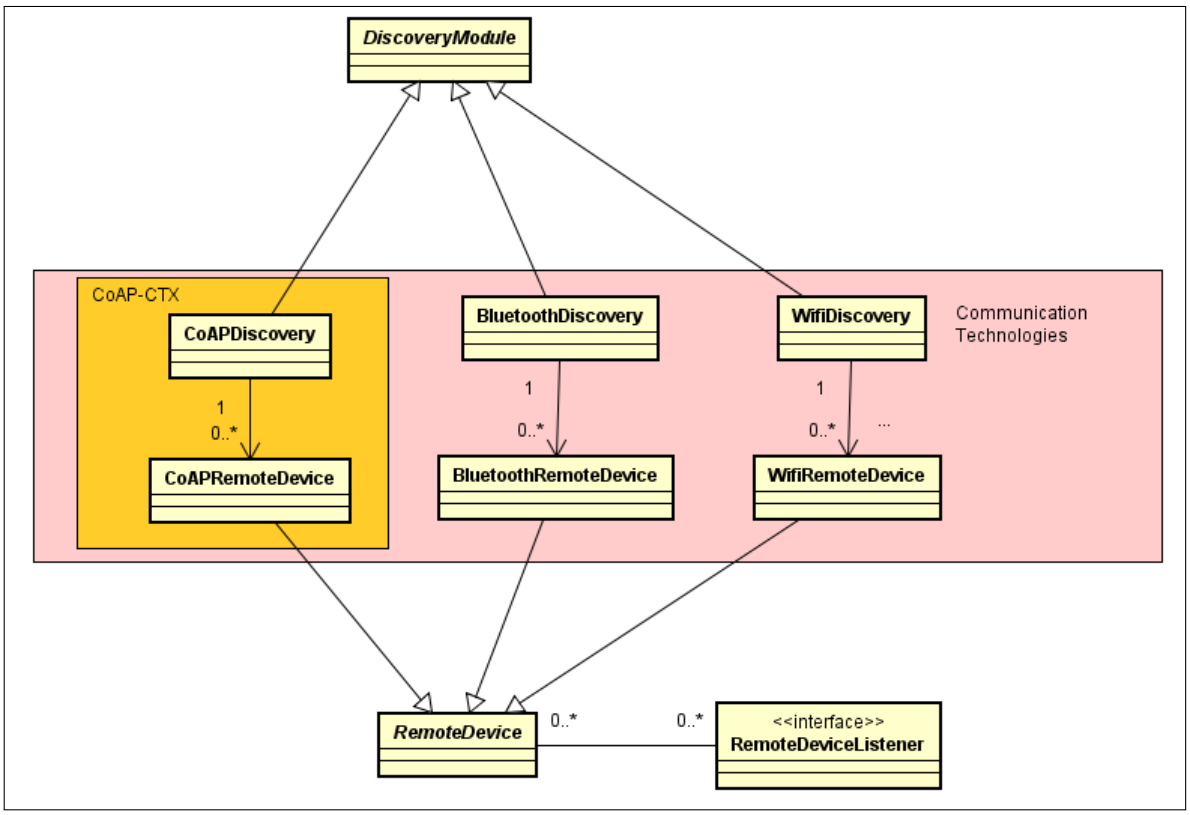
A proposta do CoAP-CTX é descobrir OIs de interesse do usuário em um determinado contexto, permitindo assim que aplicações possam acessar e controlar remotamente esses OIs(BARRETO *et al.*, 2017a). Logo, a integração com o LoCCAM tem por objetivo tanto obter o contexto do usuário quanto fazer com que os OIs se tornem acessíveis a esses usuários de aplicações Android. Esta seção apresenta como a camada de comunicação do LoCCAM foi estendida de modo a permitir a descoberta CoAP e o uso dos SACs para obter o contexto do usuário.

O LoCCAM já possui uma camada de comunicação responsável por acessar OIs das mais diversas tecnologias e prover interfaces que abstraem o acesso e controle desses OIs. Atualmente, o LoCCAM dá suporte a descoberta de OIs via *Bluetooth*, *Wi-Fi* e *Wi-Fi Direct* seguindo um protocolo próprio de requisição-resposta. Para estender essa camada, adicionando suporte ao CoAP, foi necessário criar duas novas classes, *CoAPRemoteDevice* e *CoAPDiscovery*, que funcionam como uma espécie de tradutores entre o protocolo conhecido pelo LoCCAM e a especificação CoAP. A Figura 22 mostra essa extensão da camada de comunicação do LoCCAM a partir da integração do CoAP-CTX.

A classe *CoAPDiscovery* encapsula toda a arquitetura do CoAP-CTX apresentada anteriormente, além de fornecer as informações necessárias ao LoCCAm para que a lista de OIs descobertos pelo CoAP fique disponível para aplicações. Sempre que houver alguma alteração no contexto do usuário, é realizada uma nova descoberta no CoAP-CTX, e a lista de OIs selecionados é atualizada no gerenciador de descoberta do LoCCAM, que por sua vez torna essa lista acessível para todos os SACs e aplicações que tenham interesse nela. Todos os OIs ativos, ou seja, que foram selecionados na última descoberta, precisam periodicamente enviar mensagens de *ping* para indicar que ainda estão disponíveis. Se houver defeito ou falha em algum OI em tempo de execução, o LoCCAM automaticamente detecta e notifica aplicações e SACs sobre mudanças na lista de OIs.

A lista de OIs descobertos pelo CoAP-CTX é agora uma lista de *CoAPRemoteDevice*. Ela abstrai o acesso a esses OIs de maneira unificada, ou seja, aplicações enviam e recebem mensagens aos OIs da mesma forma, independente da tecnologia de comunicação (e.g.,*Bluetooth*, *Wi-Fi* ou até CoAP (independente de tecnologia)). A classe *CoAPRemoteDevice* também permite

Figura 22 – CoAP-CTX como uma extensão do serviço de descoberta do LoCCAM.



Fonte – Elaborado pelo Autor.

dois tipos de interações com os OIs, leituras e escritas síncronas, ou na forma de subscrição. Logo, sempre que houver nova informação sobre esse OI as aplicações são notificadas automaticamente por meio de *callbacks*.

4.6 Gerenciamento dos Objetos Inteligentes

Uma das contribuições desse trabalho é fazer com que os recursos computacionais e energéticos dos OIs sejam economizados por uma política em que OIs que não são de interesse do usuário entrem em modo de espera. Nesse caso, o CoAP-CTX gerencia o estado dos OIs por meio de mensagens de controle que fazem os OIs, compatíveis com o protocolo, entrarem ou saírem do modo de espera. Um OI em modo de espera não envia mensagens do tipo *ping*, além de diminuir a frequência de leitura da interface sem fio, em busca de novas mensagens.

O CoAP-CTX mantém tanto o cliente CoAP padrão (que executa no dispositivo móvel) como um ou mais servidores CoAP. Um servidor CoAP mantém o registro de todos os OIs disponíveis em um ambiente, isso inclui até os que não fazem parte do interesse do

usuário naquele momento. Ao receber a requisição feita em busca dos OIs, o servidor aplica os filtros sobre os campos *Resource-Type* e *Interface-Description* e retorna a lista de OIs que atendem à requisição. Pode ocorrer de vários OIs serem retornados, por exemplo as TVs do quarto e da sala. Não é possível realizar uma consulta para identificar, ainda no Servidor CoAP, qual é a televisão do quarto especificamente. Isso porque a informação contextual do OI, que é dinâmica, não fica disponível no serviço de descoberta do Servidor CoAP. Para ter acesso a essas informações, clientes CoAP devem realizar uma operação de GET nesses OIs e receber seu contexto no *payload* da mensagem.

O Serviço de Descoberta do CoAP-CTX, ao receber a lista com os OIs pré-selecionados pelo servidor, gera uma requisição GET para cada um deles. Com base nas informações contidas no *payload* da mensagem, é possível o *matching* contextual que envolve características dinâmicas, como a localização do usuário. A referência dos OIs selecionados é por fim repassada para o gerenciador da interface gráfica da aplicação, que é responsável por exibir na tela do *smartphone* os OIs selecionados, além de permitir que o usuário os acesse. Todos os outros OIs que estão registrados no servidor CoAP são postos em modo de espera, até que o contexto mude e uma nova descoberta aconteça.

Sempre que ocorrer uma mudança no contexto do usuário que implica na necessidade de uma nova descoberta, o CoAP-CTX envia uma mensagem do tipo *SET_REQUEST* a todos os OIs que estavam anteriormente em modo de espera. A mensagem indica que agora eles devem se manter ativos durante a descoberta. Isso é necessário para que todos os OIs estejam disponíveis durante a etapa de pré-seleção, e posteriormente na etapa de *matching* contextual. Uma vez que a nova lista de OIs é selecionada, os que não estiverem nessa lista recebem novamente uma mensagem indicando a nova entrada em modo de espera, até que a descoberta seja disparada novamente.

Como os OIs em modo de espera não enviam mensagens do tipo *ping* para o servidor CoAP, isso poderia indicar que o OI não está mais disponível na rede, o que na verdade não é verdade, apenas está em modo de espera. Para resolver esse problema, o CoAP-CTX periodicamente envia mensagens internas de *ping* no lugar dos OIs em modo de espera, fazendo com que aplicações ou SACs ainda consigam acessar esses OIs, mesmo que não utilizando o CoAP-CTX como módulo de descoberta. Como essas mensagens simuladas não trafegam pela rede, são apenas comandos internos na arquitetura do LoCCAM, o custo adicional é irrelevante

quando comparado ao custo que haveria caso os OI não estivessem em modo de espera.

4.7 Conclusão

Neste capítulo, foram apresentados os principais componentes que compõem o CoAP-CTX e como foi feita a integração entre CoAP-CTX e LoCCAM. Além disso, um processo de descoberta sensível ao contexto foi apresentado. Ele define o conjunto de etapas necessárias para a partir do contexto do usuário obter a lista de OIs relevantes. O capítulo também trouxe uma visão geral sobre como os OIs são gerenciados a partir do interesse do usuário, entrando ou não em modo de espera para economizar recursos computacionais e energéticos.

O processo de descoberta sensível ao contexto proposto é seguido pela arquitetura do CoAP-CTX, com as etapas distribuídas entre o *smartphone*, o servidor CoAP e os diversos OIs. O contexto é obtido por meio de SACs gerenciados pelo LoCCAM, que por sua vez publicam as informações contextuais no espaço de tuplas para serem lidas pelo CoAP-CTX a fim de dar início ao processo de descoberta. A obtenção do contexto por meio dos SACs implica também na facilidade de extensão da solução, como a iteração entre CoAP-CTX e SACs é desacoplada, a inclusão de novos SACs ou modificação dos existentes pode ocorrer de maneira menos impactante para o sistema.

Na implementação do CoAP-CTX, todos os serviços fornecidos pelo LoCCAM para acesso e controle dos OIs foram reutilizados, fazendo com que aplicações consigam se comunicar com OIs compatíveis com o CoAP da mesma forma que com outros OIs de outras tecnologias. O protocolo de requisição-resposta especificado pelo LoCCAM também foi reutilizado para a gerência dos OIs em modo de espera, fazendo com que os OIs compatíveis com o CoAP-CTX também possam ser utilizados explicitamente por outros módulos de descoberta, como *Bluetooth* ou *Wi-Fi*.

O CoAP-CTX possui duas contribuições principais: i) combinar as informações contextuais do usuário e dos OIs para realizar uma descoberta mais seletiva; e ii) permitir que OIs que não sejam de interesse da aplicação (usuário) em um determinado momento entrem em modo de espera, de modo a economizar recursos. Essas duas contribuições juntamente com o CoAP como base para descoberta fazem com que o CoAP-CTX atenda aos quatro requisitos de descoberta de OIs em IoT descritos na Seção 2.1.2.

5 AVALIAÇÃO: PROVA DE CONCEITO E SIMULAÇÃO

Este capítulo apresenta a aplicação Android chamada U-Control, utilizada como PoC para o CoAP-CTX. Essa aplicação obtém a lista de OIs disponíveis no ambiente e permite que usuários acessem e controlem esses OIs. A lista de OIs muda de acordo com mudanças no contexto do usuário, baseado nas informações de localização e histórico. Um simulador de redes sem fio foi utilizado para realizar para avaliar o desempenho do CoAP-CTX. Definiu-se uma estratégia de variações do número de OIs presentes no ambiente, de modo a analisar o comportamento do CoAP-CTX para diversos cenários e diferentes tipos de ambientes inteligentes.

A Seção 5.1 apresenta a prova de conceito desenvolvida para demonstrar o funcionamento do CoAP-CTX, incluindo as definições no contexto do usuário, detalhes da aplicação U-Control e informações sobre os OIs implementados. Os resultados obtidos na simulação são apresentados na Seção 5.2, no qual são discutidos os detalhes da criação dos ambientes inteligentes, como foram extraídas as métricas de tempo de descoberta e número total de mensagens. Por fim, a Seção 5.3 traz as conclusões baseadas na prova de conceito e na avaliação.

5.1 Prova de Conceito

O objetivo da PoC é ilustrar uma descoberta seletiva de OIs, com base no contexto do usuário. Para essa PoC, foram implementadas uma aplicação *Android*, dois SACs para obtenção de informações contextuais, um servidor CoAP baseado na implementação jCoAP[1] e quatro OIs utilizando a plataforma Arduino para tornar acessíveis duas TVs, um ar-condicionado, e um sensor de luminosidade do jardim (e.g., que atesta a incidência de luminosidade sobre as plantas que o usuário cultiva).

5.1.1 SACs Implementados

Para obter o contexto do usuário foram implementados dois SACs, um baseado na localização do usuário e outro no histórico de acesso do usuário aos OIs. O CoAP-CTX publica o interesse nessas duas informações contextuais no LoCCAM, que por sua vez inicializa os SACs que periodicamente publicam valores de localização e preferência no espaço de tuplas. O CoAP-CTX utiliza essas informações para enriquecer o processo de descoberta de maneira a

[1] http://www.ws4d.org/ws4d-jcoap/

selecionar os OIs mais utilizados pelo usuário no ambiente onde o mesmo se encontra.

O SAC de localização utiliza uma lógica simples para identificação da localização do usuário, a distância euclidiana entre pontos pré-definidos e a localização atual obtida pelo GPS. Um ponto central foi definido, que representa o centro do ambiente monitorado, no caso uma casa, caso a distância obtida pelo GPS a esse ponto seja menor que 10 metros, é calculada a distância para cada um dos cômodos da casa (também pré-definidos). O ambiente cuja distância for a menor comparada aos demais é considerado o ambiente onde o usuário se encontra. Embora esse algorítimo não possua uma precisão alta, e ainda dependa de um configuração inicial, o mapeamento do ambiente, ele foi suficiente para indicar variações na localização do usuário, o que ocasiona a uma diferente lista de OIs encontrados, mostrando assim a descoberta sensível ao contexto do CoAP-CTX.

O histórico, por sua vez, é obtido por meio de um SAC que lê as ultimas informações do usuário, armazenadas em um arquivo de texto, e extrai os últimos parâmetros dos OIs acessados. Cada interação do usuário com um OI gera um entrada no arquivo de histórico, contendo o *timestamp* e os atributos CoRE que descrevem os OIs acessado. Sempre que houver uma alteração na posição do usuário, obtida pelo SAC de localização, o SAC de histórico analisa o arquivo e extrai as informações dos últimos 7 dias, com base nisso, verifica os campos CoRE que mais se repetem. Um campo que se repita mais de 10 vezes nos últimos 7 dias é considerado de preferência do usuário e é publicado no espaço de tuplas para ser utilizado pelo CoAP-CTX para criar as *context-keys* que definem o interesse do usuário.

5.1.2 Informações de Contexto

Os SACs de localização e histórico foram utilizados para obter as informações contextuais do usuário. No cenário proposto para a PoC, o usuário acaba de chegar em casa (distância menor que 10m do ponto de referência da casa) e abre a aplicação U-Control para acessar e controlar os OIs presentes no ambiente. Foi criado um histórico inicial, contendo informações de acesso do usuário aos OIs no decorrer de uma semana para exemplificar o uso do CoAP-CTX. Partindo do principio de que os usuários tendem a acessar mais os atuadores do ambiente quando chegam em casa, como portas, luzes, entre outros, o histórico inicial representa essa priorização dos atuadores com relação aos sensores. Isso implica que idealmente o CoAP-CTX deve, quando o usuário chegar em casa, encontrar apenas os OIs que sejam do tipo

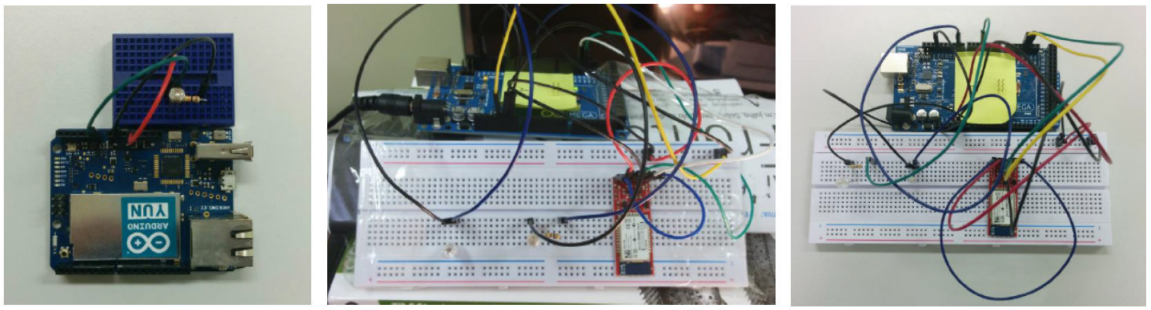
atuador.

Com base no histórico, o CoAP-CTX realiza a seleção de OIs utilizando um filtro aplicado no campo *Interface-Description*, selecionando apenas os atuadores (if="actuator"), no caso, as duas TVs e o ar-condicionado, omitindo assim o sensor de luminosidade, que baseado no contexto do usuário era irrelevante no momento da chegada à casa. O *matching* contextual seleciona os OIs que estejam na mesma localização do usuário. Caso a localização *indoor* do usuário ou dos OIs não estejam definidas, todos os OIs pré-selecionados serão o resultado final da descoberta e serão apresentados ao usuário final da aplicação U-Control.

5.1.3 Objetos Inteligentes

Para a implementação dos OIs, foram utilizados três Arduinos Mega e um Arduino Yún, com interfaces de comunicação *Bluetooth* e *WiFi*, respectivamente. A Figura 23 apresenta os hardwares utilizados para a PoC. Em (a) está o Arduino Yún com o sensor de luminosidade. Em (b) os componentes para controlar o Ar Condicionado. Em (c) são exibidos os componentes utilizados para controlar as TVs (replicação do hardware). O controle das TVs e do Ar Condicionado foi feito utilizando infravermelho (IR), com os códigos IR de cada comando obtidos por engenharia reversa.

Figura 23 – Arduinos utilizados para criação dos Objetos Inteligentes.



(a) Sensor de luminosidade (b) Controlador do Ar condicionado (c) Controlador da TV

Fonte – Elaborado pelo autor

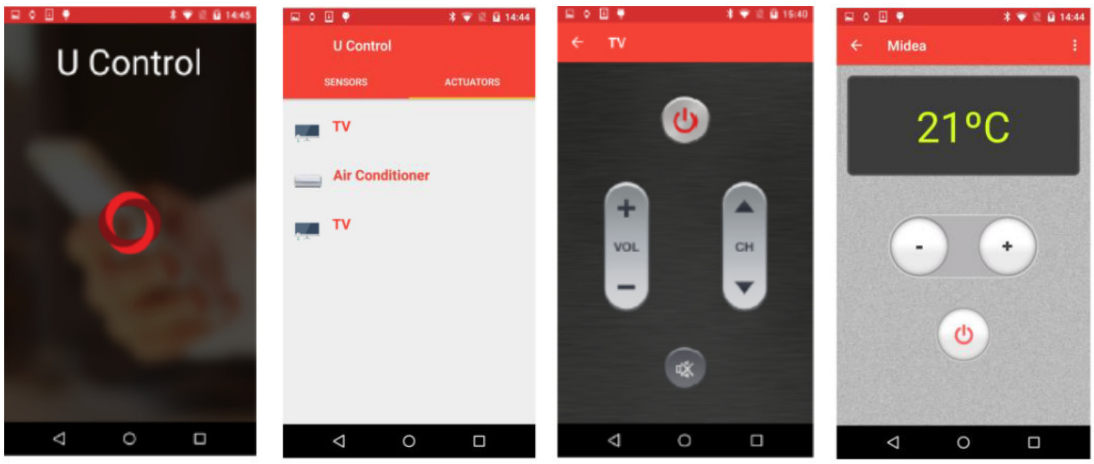
5.1.4 U-Control

Usuários podem acessar os OIs encontrados utilizando a aplicação Android U-Control, que internamente lista os OIs disponibilizados pelo LoCCAM. Esses OIs são todos os encontrados seguindo a especificação padrão do protocolo de requisição-resposta do LoCCAM,

incluindo os selecionados pelo CoAP-CTX. Essa aplicação foi inicialmente desenvolvida durante o mesmo projeto de P&D que aperfeiçoou o LoCCAM para o cenário de IoT, apenas sendo preciso algumas adaptações na interface gráfica para demonstrar o funcionamento do CoAP-CTX.

A Figura 24 apresenta *prints* da tela da aplicação executando. Durante o carregamento da aplicação (a), são realizadas as consultas ao servidor CoAP e realizado o *matching* contextual. Em (b), são listados os OIs de interesse do usuário. Já as telas (c) e (d) mostram a interface de controle da TV e do Ar Condicionado (AC), respectivamente.

Figura 24 – Aplicação Android que acessa e controla os OIs descobertos.



(a) Busca por OI (b) Lista de OI (c) Controle da TV (d) – Controle do AC

Fonte – Elaborado pelo autor

5.1.5 Discussão

Essa PoC demonstrou como usuários podem se beneficiar do serviço de descoberta proposto para acessar OIs disponíveis no ambiente. Embora poucos OIs tenham sido utilizados nessa PoC, por limitações de recursos (incluindo tempo para o desenvolvimento), o objetivo principal foi alcançado, o de demonstrar que o número de OIs pode ser reduzido com base no contexto do usuário. Essa mesma aplicação pode ser utilizada em cenários com muito mais OIs, ou com diferentes informações contextuais servindo como filtros para a descoberta, já que toda a iteração entre aplicação e serviço de descoberta é desacoplada devido ao espaço de tuplas.

5.2 Simulação

O CoAP-CTX adiciona uma camada de complexidade ao serviço de descoberta padrão do CoAP que pode causar um aumento do tempo de descoberta final. Com intuito de mensurar esse impacto e o volume total de mensagens trocadas na rede, o CoAP-CTX foi simulado com o objetivo de realizar uma avaliação de desempenho de forma a verificar o comportamento da solução proposta com relação ao número de mensagens trocadas na rede e o tempo gasto para realizar a descoberta de OIs. Levantou-se, portanto, duas hipóteses sobre o CoAP-CTX que são respondidas com base nas avaliações conduzidas nas simulações:

- **Hipótese 1:** O CoAP-CTX reduz o número total de mensagens trocadas na rede local, quando o resultado da filtragem contextual de OIs diminui o número final de objetos encontrados;
- **Hipótese 2:** O *matching* contextual realizado pelo CoAP-CTX aumenta o tempo de descoberta de OIs, mas não inviabiliza sua utilização.

5.2.1 Materiais e Métodos

Para essa avaliação, foi utilizado o simulador Cooja[2], que é voltado especificamente para redes de sensores sem fio. O Cooja foi executado por meio de uma máquina virtual Linux, disponibilizada pelos próprios desenvolvedores do simulador. Todos os componentes foram implementados e simulados para a plataforma de *hardware Tmote Sky*[3]. Essa plataforma possui um baixo tempo de transição para sair do modo de espera, além disso, utiliza o módulo CC2420[4] para comunicação sem fio, que é compatível com o padrão IEEE 802.15.4. O Cliente CoAP realiza a descoberta a cada 10 segundos, enquanto os OIs que não estão em modo de espera enviam seu estado a cada 200*ms* para o servidor CoAP.

5.2.2 Procedimento

Os resultados dos experimentos dependem diretamente do contexto do usuário e dos OIs, podendo sofrer grandes variações para contextos diferentes (BARRETO *et al.*, 2017b). Como forma de avaliar o CoAP-CTX em um maior número de variações de contexto, foram

[2] http://anrg.usc.edu/contiki/index.php/Cooja_Simulator
[3] http://wirelesssensornetworks.weebly.com/1/post/2013/08/tmote-sky.html
[4] http://www.ti.com/product/CC2420

propostos três cenários tipo: casa, carro e prédio inteligente. Para cada cenário, um número de OIs pré-selecionados e um número final de OIs descobertos foi estabelecido de forma empírica. Esse número varia conforme três casos específicos de descoberta: o melhor e o pior caso, e um intermediário. O melhor ocorre quando nenhum OI é pré-selecionado na requisição CoAP, permitindo assim que todos os OIs existentes entrem em modo de espera. Para não considerar esse caso trivial, vamos considerar como melhor caso quando apenas um OI é pré-selecionado e não há necessidade de consulta por informações dinâmicas no *payload* da mensagem desse OI específico. Já o pior caso ocorre quando todos os OIs são pré-selecionados e há necessidade da consulta individual, resultando em uma seleção final de todos os OIs. O caso intermediário é analisado separadamente para cada cenário simulado. Foi então analisada o desempenho da descoberta usando diretamente o CoAP (sem contexto), e depois usando o CoAP-CTX (com contexto).

A escolha dos parâmetros de cada cenário de forma empírica foi um recurso utilizado para adicionar mais significado aos resultados obtidos pela simulação. A comparação entre CoAP-CTX e CoAP padrão com base no pior caso pode não representar o valor obtido em um cenário real, dependendo das características inerentes de cada ambiente. Porém, o pior e o melhor caso estabelecem os limites inferiores e superiores do desempenho do CoAP-CTX nesses cenários reais.

Cada ambiente é composto por um cliente CoAP, que representa o *smartphone*, um ou mais servidores CoAP, e um conjunto de OIs que se registram nesses servidores. A Figura 25 apresenta a topologia de cada rede gerada. Os servidores utilizados na simulação permitem multi-salto, logo basta que qualquer cliente esteja no alcance de no mínimo um servidor para que suas mensagens possam chegar a qualquer outro nó da rede. Para cada caso de cada ambiente foram realizadas 10 simulações de 15 minutos. Os resultados são apresentados na forma de média dos valores obtidos em cada conjunto de simulações.

5.2.3 Resultados

Nesta subseção, são apresentados os resultados obtidos para os três cenários propostos: casa, carro e prédio inteligente. Para cada cenário, é feita uma breve explicação sobre a escolha dos números de OIs pré-selecionados e o número final de OIs descobertos. Os resultados são apresentados na formas de gráficos do tipo *box-plot*, com o objetivo de exibir tanto os

Figura 25 – Redes simuladas para os três cenários: (a) Casa; (b) Carro; (c) Prédio Inteligente.

Fonte – Elaborado pelo autor

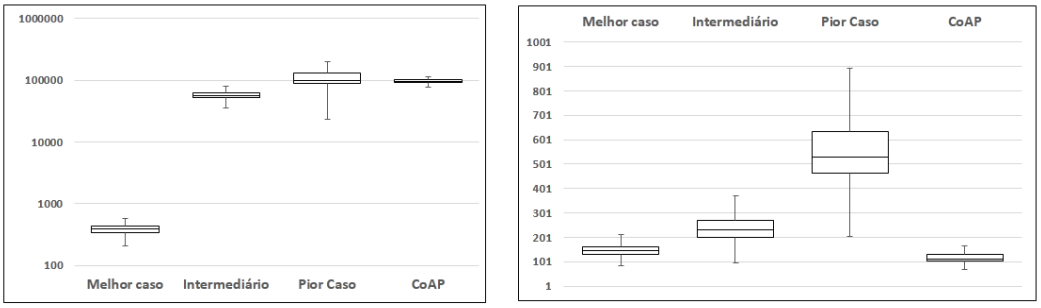
valores médios quanto tornar possível uma visualização dos parâmetros de variância. A escala dos gráficos é logarítmica, pois geralmente o melhor caso apresenta números extremamente reduzidos com relação aos demais, isso faz com que a diferença entre o caso intermediário e o pior caso muitas vezes pareçam próximos, mesmo sendo consideravelmente diferentes.

5.2.3.1 Casa

Casas inteligentes geralmente possuem um número moderado de OIs, para a simulação foram utilizados 20 OIs, 1 Servidor CoAP e 1 Cliente CoAP. Para o caso intermediário, supôs-se que o interesse do usuário é mais genérico, fazendo assim com que a seleção final retorne ainda um grande número de OIs. Neste caso, 15 OIs são pré-selecionados pelas *Strings* de consulta CoAP, desses, 5 são descartados após a aplicação do *matching* contextual com informações dinâmicas, totalizando 10 OIs ativos e 10 em modo de espera. A Figura 26 mostra os resultados encontrados para esse ambiente.

É possível observar na Figura 26 que uma redução de 50% do número de OIs descobertos gerou uma redução de aproximadamente 41% no número total de mensagens. Já o tempo de descoberta aumentou em 100*ms*, também no caso intermediário, com relação ao CoAP padrão. Com exceção do pior caso, que apresentou uma média do número de mensagens um pouco acima do CoAP padrão, todos os demais se comportaram como esperado, reduzindo o tráfego na rede e consequentemente otimizando o consumo energético dos OI. Ainda com relação ao tempo de descoberta, o melhor caso do CoAP-CTX se aproximou ao CoAP padrão,

Figura 26 – Tempo de descoberta e número total de mensagens para casa inteligente.



(a) Número total de mensagens na rede (b) Tempo de descoberta de OI

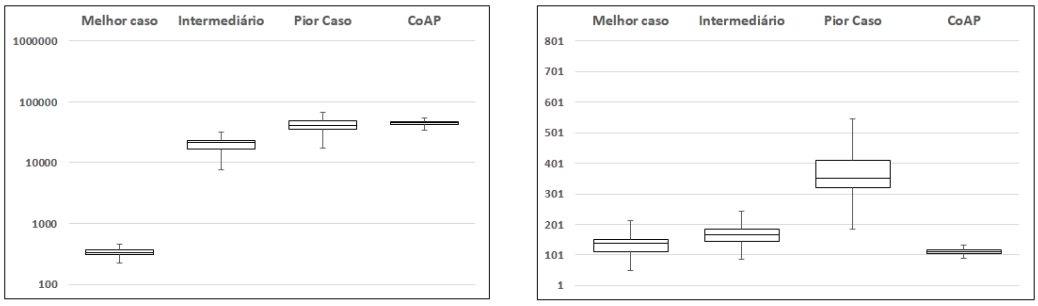
Fonte – Elaborado pelo autor

enquanto o pior caso obteve um tempo de resposta superior em cerca de 500 milissegundos.

5.2.3.2 Carro

Carros possuem um número de OIs acessíveis mais reduzido que o cenário da casa inteligente, por isso foram utilizados 10 OIs, 1 Servidor CoAP e 1 Cliente CoAP. O cenário de utilização dos OIs do carro também são mais específicos. Enquanto dirige, o usuário tende a ter mais interesse em OIs relacionados a navegação, já os OIs de multimídia ganham prioridade quando o carro está parado. Com base nisso, definiu-se o caso intermediário como sendo 4 OIs pré-selecionados pelo servidor CoAP, dentre esses 3 se tornam disponíveis ao usuário, totalizando 7 OIs em modo de espera. Os resultados são apresentados na Figura 27.

Figura 27 – Tempo de descoberta e número total de mensagens para carro inteligente.



(a) Número total de mensagens na rede (b) Tempo de descoberta de OI

Fonte – Elaborado pelo autor

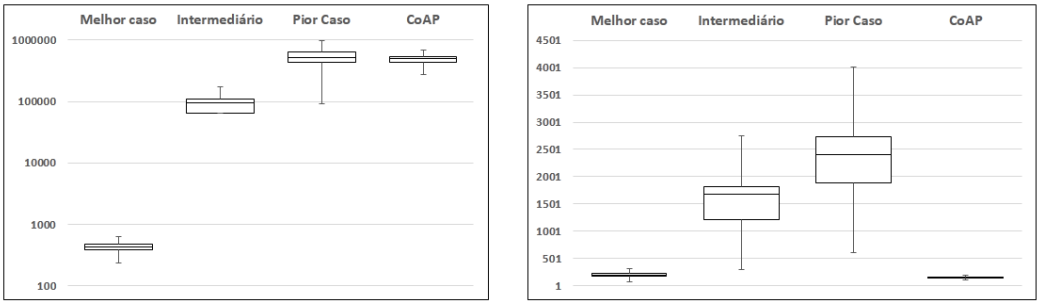
Os resultados mostram que, para o caso intermediário, o CoAP-CTX obteve uma redução de cerca de 57% no número de mensagens, com um aumento no tempo de descoberta médio de apenas 63*ms*. O aumento no tempo de descoberta, para todos os casos, foi menor do

que o cenário da casa por causa contexto empírico considerado para o carro, onde os OIs são mais específicos, permitindo assim uma melhor pré-seleção com base na informações estáticas de descrição. O impacto de uma boa pré-seleção é refletido no número de mensagens trocadas utilizando o CoAP-CTX, onde até o pior caso apresentou média de mensagens inferior ao CoAP tradicional.

5.2.3.3 Prédio

Um prédio inteligente é uma estrutura física, geralmente maior que a casa, que possui uma elevada densidade de OIs disponíveis, sendo esses de propósito mais geral, para atender vários usuários. Para esse cenário, utilizou-se um total de 100 OIs, 3 Servidores CoAP e um Cliente CoAP. Nesse cenário, informações de contexto dinâmicas, como localização *indoor*, são fundamentais para limitar os resultados da descoberta. Para o caso intermediário, considerou-se que 70 OIs seriam pré-selecionados, mas que apenas 15 seriam de interesse do usuário após a obtenção das informações contextuais dinâmicas. A Figura 28 mostra os resultados obtidos.

Figura 28 – Tempo de descoberta e número total de mensagens para prédio inteligente.



(a) Número total de mensagens na rede (b) Tempo de descoberta de OI

Fonte – Elaborado pelo autor

Houve uma drástica redução no número de mensagens trocadas na rede quando utilizada a solução proposta, cerca de 80%, para o caso intermediário. Entretanto, para esse caso, o tempo de descoberta médio ficou acima em 1,5 segundos quando comparado ao CoAP tradicional. Esse cenário de prédio inteligente é o oposto do cenário do carro, já que os OIs do prédio em geral possuem características menos específicas, geralmente de uso coletivo (e.g., lâmpadas, TVs, portas). Nesse caso, é necessário um maior processamento das regras dinâmicas, ocasionando um elevado *overhead* no tempo de descoberta, para reduzir consideravelmente o número final de mensagens trocadas na rede.

5.2.4 Discussão

Os experimentos mostraram que, em geral, quando comparado ao serviço de descoberta padrão do CoAP, o CoAP-CTX apresenta uma redução no número total de mensagens trocadas na rede, ao custo de um aumento no tempo necessário para a realização da descoberta. A redução do número de mensagens ocorre em todos os cenários exceto nos que se aproximam do pior caso, no qual as informações contextuais fazem com que nenhum OI seja descartado pelo *matching* contextual. Isso faz com que o número de mensagens se aproxime dos valores encontrados para o CoAP padrão, podendo até aumentar um pouco. Aumento esse que, mesmo no pior caso, é bem pequeno, pois é devido ao envio de requisições para obtenção de informações contextuais dinâmicas, que representa um percentual bem menor quando comparado a mensagens de atualização dos OIs, já que esse tipo de requisição só é enviada a cada nova descoberta.

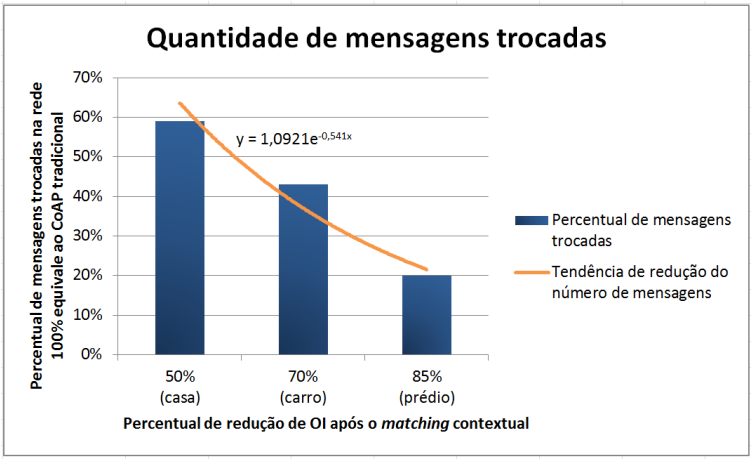
Notou-se que o número total de mensagens depende fortemente do número final de OIs em modo de espera, enquanto que o tempo de descoberta é afetado pela quantidade de OIs com informações dinâmicas a serem obtidas para a realização do *matching* contextual. Portanto, conclui-se que em cenários em que os OIs tem uma função mais especializada, ou seja, em que seja possível pré-selecionar um conjunto próximo do final de OIs de interesse, o aumento no tempo de descoberta é baixo. Já em ambientes mais genéricos, como prédios inteligentes, em que a maior parte da seleção ocorre com base nas informações contextuais dinâmicas, o tempo de descoberta aumenta consideravelmente. Uma solução para esse aumento seria a migração do processo de *matching* contextual para o próprio servidor CoAP, pois assim todo o processo de descoberta seria realizado com uma única requisição ao servidor CoAP. Entretanto, essa migração faria com que a solução perdesse parte da interoperabilidade que o CoAP proporciona, pois ambientes que já possuem servidores CoAP tradicionais não seriam compatíveis com a aplicação dos filtros contextual aplicados durante o *matching* de OIs.

Com base nos resultados expostos neste capítulo, pode-se verificar a validade das duas hipóteses levantadas acerca do CoAP-CTX:

- **Hipótese 1:** O CoAP-CTX reduz o número total de mensagens trocadas na rede local, quando o resultado da filtragem contextual de OIs diminui o número final de objetos encontrados;

A quantidade de mensagens trocadas na rede se mostrou diretamente proporcional à quantidade de OIs que permanecem ativos após uma descoberta. A Figura 29 mostra que, a medida que o número de OIs é reduzido (OIs postos em modo de espera), a quantidade média de mensagens trocadas durante 15 minutos de simulação diminui, quando comparado com o número de mensagens trocadas nesse mesmo período utilizando o serviço de descoberta do CoAP padrão. Todos os eixos são representados em percentagem, no qual o eixo das abcissas representa a proporção de OIs que não foram postos em modo de espera para cada cenário (casa, carro e prédio), já o eixo da ordenadas representa o percentual de mensagens reduzidas com a utilização do CoAP-CTX, comparado com o CoAP padrão (100% das mensagens). A Figura 29 também exibe uma curva que representa a tendência na redução do número de mensagens a medida que mais OIs entram em modo de espera. Logo, a hipótese de redução do número total de mensagens quando o *matching* contextual reduz o número de OIs é verdadeira.

Figura 29 – Redução do número de mensagens trocadas.



Fonte – Elaborado pelo autor

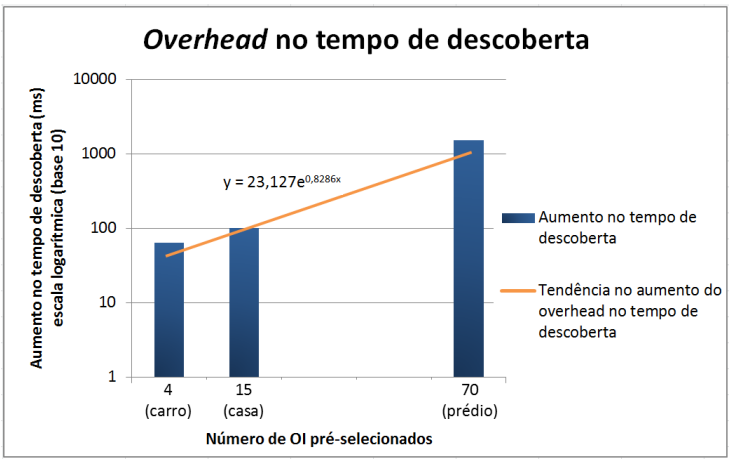
- **Hipótese 2:** O *matching* contextual realizado pelo CoAP-CTX aumenta o tempo de descoberta de OIs, mas não inviabiliza sua utilização.

Idealmente, uma vez que um conjunto de OIs é descoberto, não é preciso realizar uma nova descoberta até que o contexto do usuário e dos OIs sofram alguma alteração relevante. Nesse caso, o aumento no tempo de descoberta traz um impacto de desempenho sempre que o contexto mudar. Entretanto, em um cenário real, no qual assume-se que o usuário se movimenta dentro de um ambiente inteligente e depois para de modo a realizar alguma atividade, a lista de OIs mais relevante ao usuário tende a ser, na maioria dos casos, a obtida após o movimento

(momento em que o contexto já não varia tanto). Logo, o tempo de descoberta de OIs não deve ser muito alto, a ponto de afetar a utilização da aplicação pelo usuário em um momento em que o contexto não mude rapidamente. Contudo, mais experimentações precisam ser realizadas para comprovar essa argumentação.

A Figura 30 apresenta o *overhead* no tempo médio de descoberta com o CoAP-CTX comparado ao tempo do serviço de descoberta padrão do CoAP. Os cenários da casa, carro e prédio foram utilizados para verificar o comportamento do tempo de descoberta com relação a variação no número de OIs pré-selecionados, fator esse que é o maior responsável pelo aumento do overhead na descoberta realizada pelo CoAP-CTX. A quantidade de OIs pré-selecionados no caso do prédio foi muito maior que nos outros dois cenários, então usou-se uma escala logarítmica para o eixo das ordenadas, de modo a melhorar a visualização dos resultados. O gráfico também mostra uma curva de tendência do aumento do tempo de descoberta, isso indica que, embora a variação seja pequena em escala, a arquitetura atual do CoAP-CTX é penalizada quando o número de OIs pré-selecionados atinge valores extremamente elevados. Uma técnica para aumentar a escalabilidade da solução, baseada na migração do *matching* contextual para o servidor CoAP, é apresentada como trabalho futuro na Seção 6.4.

Figura 30 – *Overhead* no tempo de descoberta de OIs.



Fonte – Elaborado pelo autor

Considerando cenários em que o número de OIs presentes no ambiente não ultrapassa o limite de 100, o tempo de descoberta aumenta em no máximo 1,5 segundos. Isso quer dizer que, no pior caso, usuários precisam esperar 1,5 segundos para acessar os OIs relevantes a ele naquele contexto. Esse tempo é, empiricamente, comparável ao tempo médio de carregamento

de diversas aplicações bastante utilizadas no dia a dia dos usuários. Logo, a hipótese de que o *overhead* do CoAP-CTX no tempo de descoberta não inviabiliza seu uso é verdade, para ambientes que não possuam uma quantidade extremamente elevada de OIs.

5.3 Conclusão

Este capítulo apresentou uma aplicação Android utilizada como prova de conceito na utilização do servido de descoberta do CoAP-CTX. A PoC mostrou a utilização do CoAP-CTX em um cenário real, com OIs implementados utilizando plataformas de prototipagem rápida, no caso os Arduínos. O Contexto do usuário foi definido com base no histórico e na localização, e obtido por meio dos SACs carregados e executados pelo LoCCAM. Com essa PoC foi possível visualizar uma redução no número de OIs encontrados, com base no contexto do usuário e das informações que descrevem cada OI.

Uma avaliação de desempenho do CoAP-CTX foi realizada utilizando o simulador Cooja. Nessa avaliação, foram extraídas as métricas de quantidade total de mensagens trocadas na rede e aumento no tempo necessário para realizar a descoberta de OIs, com ambas as métricas calculadas com base no serviço de descoberta tradicional do CoAP. A avaliação simulou três ambientes inteligentes: casa, carro e prédio. Cada ambiente com um número diferente de OIs presentes no ambiente, pré-selecionados e retornados pelo serviço de descoberta. Com essa avaliação foi possível comprovar as duas hipóteses levantadas na Seção 1.2: i) o CoAP-CTX reduz o número de mensagens trocadas na rede uma vez que o *matching* contextual consegue reduzir o número final de OIs; ii) o *overhead* no tempo de descoberta do CoAP-CTX não inviabiliza sua utilização em cenários onde o número de OIs não seja muito elevado.

6 CONSIDERAÇÕES FINAIS

Este capítulo resume o que foi apresentado e discutido nesta dissertação de mestrado. A Seção 6.1 apresenta os resultados alcançados com o CoAP-CTX, além de fazer um comparativo da solução proposta com os trabalhos relacionados apresentados no Capítulo 3. A Seção 6.2 apresenta as principais limitações do CoAP-CTX. Na Seção 6.3 são listadas as principais publicações realizadas durante o período do mestrado. Por fim, a Seção 6.4 apresenta as possibilidades de trabalhos futuros relacionados ao CoAP-CTX.

6.1 Resultados Alcançados

Este trabalho apresentou o CoAP-CTX, uma proposta de extensão do CoAP que possibilita uma descoberta sensível ao contexto de objetos inteligentes. São utilizadas informações contextuais do usuário e dos OIs para realizar uma seleção mais direcionada, bem como reduzir o número de mensagens trocadas pelos dispositivos, otimizando assim o consumo de energia. O CoAP-CTX foi integrado ao *middleware* LoCCAM de modo a realizar a aquisição de contexto e permitir que os usuários finais acessem e controlem os OIs presentes no ambiente.

A aplicação U-Control foi utilizada como prova de conceito para mostrar que a solução proposta é factível e pode ser utilizada em um cenário real. A PoC reproduziu um cenário no qual o usuário deseja interagir com os OIs presentes no ambiente, e para isso a aplicação utiliza o serviço de descoberta do CoAP-CTX para obter a lista de OIs relevantes ao usuário. O contexto utilizado para a PoC foi baseado em histórico e localização, mas pode ser estendido para dar suporte a novas informações contextuais.

Para mostrar que o CoAP-CTX consegue reduzir o número de mensagens trocadas na rede, e consequentemente aumentar a suficiência energética dos OIs, foi realizada uma avaliação por meio do simulador de redes sem fio Cooja. Três ambientes inteligentes foram simulados e analisados durante a avaliação: casa, carro e prédio. Além de comprovar a diminuição de mensagens, essa avaliação demonstrou que o CoAP-CTX apresenta um tempo de descoberta aceitável para um ambiente no qual o número de OIs não seja muito maior que 100, uma vez que as aplicações que utilizem o serviço de descoberta não tenham requisitos de tempo real.

Com base no que foi apresentado neste trabalho, a seguinte questão de pesquisa pode

ser respondida:

- **Questão de Pesquisa:** Como utilizar informações contextuais do usuário e dos objetos inteligentes para realizar uma descoberta seletiva, na qual apenas os objetos inteligentes acessíveis e de interesse do usuário sejam selecionados?

Para realizar essa descoberta seletiva, são necessárias duas etapas específicas: i) obter a descrição e o contexto dos OIs inteligentes presentes no ambiente; ii) realizar um *matching* contextual de modo a identificar quais desses OIs são relevantes ao usuário. Em termos práticos, uma forma de implementar essas duas etapas é utilizando o CoAP para obter as informações dos OIs e em seguida utilizar uma estratégia de *matching* baseada em filtros aplicados às informações contextuais para interpretar esses dados e selecionar os OIs que são relevantes em um contexto específico. Foi essa a abordagem implementada no CoAP-CTX.

Quando comparado com os demais trabalhos relacionados a descoberta de OIs em IoT, o CoAP-CTX se destaca por atender aos quatro requisitos definidos por (GUINARD *et al.*, 2010). A Tabela 2 mostra essa comparação entre CoAP-CTX e demais trabalhos. O CoAP-CTX é o único trabalho que utiliza o CoAP para garantir o baixo *overhead* no serviço de descoberta e no registro dos OIs, e ao mesmo tempo agrega informações contextuais para melhorar o processo de descoberta. Além disso, apenas o CoAP-CTX e o DiscoWoT dão suporte à alocação sob demanda de OIs, sendo que o DiscoWoT não apresenta nenhum exemplo real onde esse requisito é atendido.

Tabela 2 – Comparativo entre o CoAP-CTX e os demais trabalhos relacionados.

Trabalhos Relacionados	REQ 1	REQ 2	REQ 3	REQ 4
Env. B	-	✓	Usuário + OI	-
DRD4M	-	✓	-	-
Digcovery	✓	-	Usuário	-
Cirani et al.	✓	✓	-	-
DiscoWoT	-	-	OI	✓
Ishaq et al.	✓	✓	-	-
CoAP-CTX	✓	✓	Usuário + OI	✓

Fonte – Elaborado pelo autor

6.2 Limitações

O CoAP-CTX possui algumas limitações oriundas de decisões de *design* ou características que foram consideradas fora do escopo desse trabalho de Mestrado:

- Embora o CoAP-CTX não imponha ou limite nenhuma estratégia de aquisição de contexto, a prova de conceito foi implementada baseada em apenas duas informações contextuais, localização e histórico. A etapa de *matching* também se limitou a utilizar apenas uma filtragem baseada em localização;
- Uma das contribuições do CoAP-CTX é diminuir o número de mensagens trocadas na rede de modo a melhorar a eficiência energética dos OIs. Porém, devido a limitações do simulador, não foi possível analisar o gasto de energia diretamente, então a avaliação se limitou ao número de mensagens, que na prática representa o fator que mais influencia no consumo de energia dos OIs; e
- Todos os cenários implementados e avaliados possuíam apenas um único cliente CoAP, que representava o *smartphone* do usuário. A utilização de mais de um cliente CoAP introduziria vários desafios que acabaram ficando fora do escopo dessa pesquisa, como o controle de acesso aos OIs por múltiplos usuários simultâneos.

6.3 Produção Bibliográfica

Durante o período do mestrado, vários trabalhos foram publicados em eventos nacionais e internacionais, incluindo artigos completos em conferências e publicações em *workshops* de teses e dissertações. A seguir é apresentada a lista de trabalhos produzidos, sendo os três primeiros relacionados diretamente ao tema desta dissertação:

- ***CoAP-CTX: A Context-Aware CoAP Extension for Smart Objects Discovery in Internet of Things* (BARRETO *et al.*, 2017a):** publicado como artigo completo na XLI Conferência Anual de *Software* e Aplicações de Computadores (COMPSAC - 2017), com Qualis A2. O trabalho apresenta como o CoAP-CTX foi integrado ao LoCCAM para fornecer o serviço de descoberta sensível ao contexto. Autores: Felipe Mota Barreto, Paulo Artur de Sousa Duarte, Marcio Espíndola Freire Maia, Rossana Maria de Castro Andrade e Windson Viana de Carvalho;
- **CoAP-CTX: Extensão Sensível ao Contexto para Descoberta de Objetos Inteligentes**

em Internet das Coisas (BARRETO *et al.*, 2017b): publicado como artigo completo no XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC - 2017), com Qualis B1. O trabalho apresenta o serviço de descoberta sensível ao contexto do CoAP-CTX, com foco na avaliação. Autores: Felipe Mota Barreto, Windson Viana de Carvalho, Marcio Espíndola Freire Maia e Rossana Maria de Castro Andrade;

- **Um mecanismo de descoberta sensível ao contexto de objetos inteligentes em Internet das Coisas (BARRETO *et al.*, 2016):** publicado no XVI Workshop de Teses de Dissertações (WTD - 2016). O trabalho apresentou o mecanismo de descoberta sensível ao contexto que posteriormente se tornaria a base do CoAP-CTX. Autores: Felipe Mota Barreto, Windson Viana de Carvalho e Rossana Maria de Castro Andrade;
- ***Towards context-aware behaviour generation* (DUARTE *et al.*, 2015b):** publicado como resumo estendido no XXX Annual ACM Symposium on Applied Computing (SAC - 2015), com Qualis A1. O trabalho apresenta a ContextRuleML, uma DSL desenvolvida para aplicações CAM. Autores: Paulo Artur de Sousa Duarte, Felipe Mota Barreto, Francisco Anderson de Almada Gomes, Windson Viana de Carvalho e Fernando Antônio Mota Trinta; e
- **CRITiCAL: *A Configuration Tool for Context Aware and mobiLe Applications* (DUARTE *et al.*, 2014):** publicado como artigo completo na XXXIX Conferência Anual de *Software* e Aplicações de Computadores (COMPSAC), com Qualis A2, o trabalho apresenta uma abordagem para modelagem e geração de aplicações móveis e sensíveis ao contexto utilizando Engenharia baseada em Modelos e o LoCCAM. Autores: Paulo Artur de Sousa Duarte, Felipe Mota Barreto, Francisco Anderson De Almada Gomes, Windson Viana de Carvalho e Fernando Antônio Mota Trinta.

6.4 Trabalhos Futuros

O CoAP-CTX é apenas o primeiro passo rumo a uma solução mais completa em descoberta e interação com OIs em IoT. Diversos aspectos do CoAP-CTX podem ser melhorados, não limitados aos que são listados a seguir:

- Notou-se a necessidade de tornar a solução proposta mais genérica, permitindo que desenvolvedores estendam os mecanismos de aquisição de contexto e *matching* contextual. Logo, como trabalho futuro existe a possibilidade de transformar o CoAP-CTX em um

framework para descoberta de objetos inteligentes;

- Espera-se realizar mais avaliações, sobretudo para analisar se a lista final de OIs descobertos condiz com o real interesse do usuário. Além disso, realizar uma avaliação com métricas reais de energia, explicitando assim os benefícios da utilização do CoAP-CTX;
- A solução atual não dá suporte a múltiplos usuários, sendo necessária a implementação de uma política de controle de acesso para permitir vários usuários interagindo com os OIs ao mesmo tempo; e
- Para reduzir o tempo de descoberta em ambientes com muitos OIs, o processo de *matching* contextual poderia ser realizado por servidores CoAP com suporte a essa funcionalidade, ou, caso contrário, executadas no próprio *smartphone*, como é feito hoje.

REFERÊNCIAS

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787 – 2805, 2010. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>.

BALDAUF, M.; DUSTDAR, S.; ROSENBERG, F. A survey on context aware systems. **Int. J. Ad Hoc Ubiquitous Comput.**, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, v. 2, n. 4, p. 263–277, jun. 2007. ISSN 1743-8225. Disponível em: <http://dx.doi.org/10.1504/IJAHUC.2007.014070>.

BARRETO, M. F.; ANDRADE, M. C. R.; VIANA, C. W. Um mecanismo de descoberta sensível ao contexto de objetos inteligentes em internet das coisas. In: **XVI Workshop de Teses de Dissertações, WTD 2016, Piauí, Brasil**. [S.l.: s.n.], 2016. (WebMedia '16).

BARRETO, M. F.; DUARTE, P. A. d. S.; MAIA, E. F. M.; ANDRADE, M. C. R.; VIANA, C. W. Coap-ctx: A context-aware coap extension for smart objects discovery in internet of things. In: **41st IEEE Annual Computer Software and Applications Conference, COMPSAC 2017, Turin, Italy, July 4-8, 2017**. [S.l.: s.n.], 2017.

BARRETO, M. F.; MAIA, E. F. M.; ANDRADE, M. C. R.; VIANA, C. W. Coap-ctx: Extensão sensível ao contexto para descoberta de objetos inteligentes em internet das coisas. In: **Anais do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2017**. [S.l.: s.n.], 2017. p. 387–400.

BORGES, J. B. N.; RIBEIRO, P. F. N.; ANDRADE, R. M. C. Wireless sensor networks advances for ubiquitous computing. In: **Designing Solutions-Based Ubiquitous and Pervasive Computing: New Issues and Trends**. [S.l.: s.n.], 2010. p. 175–189.

CASTILLEJO, P.; MARTINEZ, J. F.; RODRIGUEZ-MOLINA, J.; CUERVA, A. Integration of wearable devices in a wireless sensor network for an e-health application. **IEEE Wireless Communications**, v. 20, n. 4, p. 38–49, August 2013. ISSN 1536-1284.

CAVALCANTE, E.; ALVES, M. P.; BATISTA, T.; DELICATO, F. C.; PIRES, P. F. An analysis of reference architectures for the internet of things. In: **Proceedings of the 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures**. New York, NY, USA: ACM, 2015. (CobRA '15), p. 13–16. ISBN 978-1-4503-3445-7. Disponível em: <http://doi.acm.org/10.1145/2755567.2755569>.

CIRANI, S.; DAVOLI, L.; FERRARI, G.; LéONE, R.; MEDAGLIANI, P.; PICONE, M.; VELTRI, L. A scalable and self-configuring architecture for service discovery in the internet of things. **IEEE Internet of Things Journal**, v. 1, n. 5, p. 508–521, Oct 2014. ISSN 2327-4662.

CISCO. Cisco global cloud index: Forecast and methodology, 2015-2020. **CISCO white paper**, 2016.

CRASSO, M.; ZUNINO, A.; CAMPO, M. Easy web service discovery: A query-by-example approach. **Science of Computer Programming**, v. 71, n. 2, p. 144 – 164, 2008. ISSN 0167-6423. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167642308000282>.

DATTA, S. K.; BONNET, C. Search engine based resource discovery framework for internet of things. In: **2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)**. [S.l.: s.n.], 2015. p. 83–85.

DAVE, E. The internet of things: how the next evolution of the internet is changing everything. **CISCO white paper**, 2011.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. **Hum.-Comput. Interact.**, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, v. 16, n. 2, p. 97–166, dez. 2001. ISSN 0737-0024. Disponível em: <http://dx.doi.org/10.1207/S15327051HCI16234_02>.

DUARTE, P. A.; GOMES, F. A.; BARRETO, F. M.; VIANA, W.; TRINTA, F. M. Loccamconfigurator: Uma ferramenta para modelagem visual de configurações de um middleware de suporte à aplicações conscientes de contexto. In: **Proceedings of the 20st Brazilian Symposium on Multimedia and the Web**. [S.l.: s.n.], 2014. (WebMedia '14).

DUARTE, P. A.; SILVA, L. F. M.; GOMES, F. A.; VIANA, W.; TRINTA, F. M. Dynamic deployment for context-aware multimedia environments. In: **Proceedings of the 21st Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: ACM, 2015. (WebMedia '15), p. 197–204. ISBN 978-1-4503-3959-9. Disponível em: <http://doi.acm.org/10.1145/2820426.2820443>.

DUARTE, P. A. de S.; BARRETO, F. M.; GOMES, F. A. de A.; CARVALHO, W. V. de; TRINTA, F. A. M. Towards context-aware behaviour generation. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2015. (SAC '15), p. 596–598. ISBN 978-1-4503-3196-8. Disponível em: <http://doi.acm.org/10.1145/2695664.2696057>.

FIRNER, B.; MOORE, R. S.; HOWARD, R.; MARTIN, R. P.; ZHANG, Y. Poster: Smart buildings, sensor networks, and the internet of things. In: **Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems**. New York, NY, USA: ACM, 2011. (SenSys '11), p. 337–338. ISBN 978-1-4503-0718-5. Disponível em: <http://doi.acm.org/10.1145/2070942.2070978>.

FREMANTLE, P. A reference architecture for the internet of things - version 0.8.2. **WSO2 white paper**, 2014.

GAMA, K.; TOUSEAU, L.; DONSEZ, D. Combining heterogeneous service technologies for building an internet of things middleware. **Computer Communications**, v. 35, n. 4, p. 405 – 417, 2012. ISSN 0140-3664. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0140366411003586>.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of things (iot): A vision, architectural elements, and future directions. **Future Generation Computer Systems**, v. 29, n. 7, p. 1645 – 1660, 2013. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X13000241>.

GUILLEMIM, P.; FRIESS, P. Internet of things strategic research roadmap. **The Cluster of European Research Projects**, 2009.

GUINARD, D.; TRIFA, V.; KARNOUSKOS, S.; SPIESS, P.; SAVIO, D. Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. **IEEE Transactions on Services Computing**, v. 3, n. 3, p. 223–235, July 2010. ISSN 1939-1374.

ISHAQ, I.; HOEBEKE, J.; ROSSEY, J.; POORTER, E. D.; MOERMAN, I.; DEMEESTER, P. Facilitating sensor deployment, discovery and resource access using embedded web services. In: **Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on**. [S.l.: s.n.], 2012. p. 717–724.

JARA, A. J.; LOPEZ, P.; FERNANDEZ, D.; CASTILLO, J. F.; ZAMORA, M. A.; SKARMETA, A. F. Mobile digcovery: A global service discovery for the internet of things. In: **Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on**. [S.l.: s.n.], 2013. p. 1325–1330.

KANG, S.; LEE, J.; JANG, H.; LEE, H.; LEE, Y.; PARK, S.; PARK, T.; SONG, J. Seemon: Scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In: **Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services**. New York, NY, USA: ACM, 2008. (MobiSys '08), p. 267–280. ISBN 978-1-60558-139-2. Disponível em: <http://doi.acm.org/10.1145/1378600.1378630>.

LIMA, F. F. P.; ROCHA, L. S.; MAIA, P. H. M.; ANDRADE, R. M. C. A decoupled and interoperable architecture for coordination in ubiquitous systems. In: **Software Components, Architectures and Reuse (SBCARS), 2011 Fifth Brazilian Symposium on**. [S.l.: s.n.], 2011. p. 31–40.

LIU, M.; LEPPäNEN, T.; HARJULA, E.; OU, Z.; YLIANTTILA, M.; OJALA, T. Distributed resource discovery in the machine-to-machine applications. In: **2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems**. [S.l.: s.n.], 2013. p. 411–412. ISSN 2155-6806.

MAIA, M. E. F.; ANDRADE, R. M. C.; FILHO, C. A. B. d. Q.; BRAGA, R. B.; AGUIAR, S.; MATEUS, B. G.; NOGUEIRA, R.; TOORN, F. Usable – a communication framework for ubiquitous systems. In: **2014 IEEE 28th International Conference on Advanced Information Networking and Applications**. [S.l.: s.n.], 2014. p. 81–88. ISSN 1550-445X.

MAIA, M. E. F.; FONTELES, A.; NETO, B.; GADELHA, R.; VIANA, W.; ANDRADE, R. M. C. Loccam - loosely coupled context acquisition middleware. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2013. (SAC '13), p. 534–541. ISBN 978-1-4503-1656-9. Disponível em: <http://doi.acm.org/10.1145/2480362.2480465>.

MAYER, S.; GUINARD, D. An extensible discovery service for smart things. In: **Proceedings of the Second International Workshop on Web of Things**. New York, NY, USA: ACM, 2011. (WoT '11), p. 7:1–7:6. ISBN 978-1-4503-0624-9. Disponível em: <http://doi.acm.org/10.1145/1993966.1993976>.

PATHAK, A.; HU, Y. C.; ZHANG, M. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof. In: **Proceedings of the 7th ACM European Conference on Computer Systems**. New York, NY, USA: ACM, 2012. (EuroSys '12), p. 29–42. ISBN 978-1-4503-1223-3. Disponível em: <http://doi.acm.org/10.1145/2168836.2168841>.

PENTIKOUSIS, K. In search of energy-efficient mobile networking. **IEEE Communications Magazine**, v. 48, n. 1, p. 95–103, January 2010. ISSN 0163-6804.

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Ca4iot: Context awareness for internet of things. In: **Proceedings of the 2012 IEEE International Conference on Green Computing and Communications**. Washington, DC, USA: IEEE Computer Society, 2012. (GREENCOM '12), p. 775–782. ISBN 978-0-7695-4865-4. Disponível em: <http://dx.doi.org/10.1109/GreenCom.2012.128>.

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context aware computing for the internet of things: A survey. **IEEE Communications Surveys Tutorials**, v. 16, n. 1, p. 414–454, First 2014. ISSN 1553-877X.

PINHEIRO, M. K.; OLIVER, M. V.; GENSEL, J.; MARTIN, H. A Personalized and Context Aware Adaptation Process for Web Based Groupware Systems. In: **4th Int. Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS'06)**. Luxembourg: [s.n.], 2006. p. 884–898. Disponível em: <http://hal.archives-ouvertes.fr/hal-00120290>.

PINHEIRO, M. K.; VANROMPAY, Y.; BERBERS, Y. Context-Aware Service Selection Using Graph Matching. In: **2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop**. Ireland: [s.n.], 2008. p. 1–14.

POORTER, E. D.; TROUBLEYN, E.; MOERMAN, I.; DEMEESTER, P. Idra: A flexible system architecture for next generation wireless sensor networks. **Wireless Networks**, v. 17, n. 6, p. 1423–1440, Aug 2011. ISSN 1572-8196. Disponível em: <http://dx.doi.org/10.1007/s11276-011-0356-5>.

SHELBY, Z.; HARTKE, K.; BORMANN, C. **The Constrained Application Protocol (CoAP)**. [S.l.], 2014. 1-112 p. Disponível em: <https://tools.ietf.org/html/rfc7252>.

SUNDMAEKER, H.; GUILLEMIN, P.; FRIESS, P.; WOELFFLé, S. (Ed.). **Vision and Challenges for Realising the Internet of Things**. Luxembourg: Publications Office of the European Union, 2010. ISBN 978-92-79-15088-3.

TALAVERA, L.; ENDLER, M.; COLCHER, S. An energy-aware iot gateway, with continuous processing of sensor data. 2016.

TAN, L.; WANG, N. Future internet: The internet of things. In: **2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)**. [S.l.: s.n.], 2010. v. 5, p. V5–376–V5–380. ISSN 2154-7491.

TARKOMA, S.; SIEKKNEN, M.; LAGERSPETZ, E.; XIAO, Y. Smartphone energy consumption: modeling and optimization. 2014.

THANGAVEL, D.; MA, X.; VALERA, A.; TAN, H. X.; TAN, C. K. Y. Performance evaluation of mqtt and coap via a common middleware. In: **2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)**. [S.l.: s.n.], 2014. p. 1–6.

THEBAULT, P.; DECOTTER, D.; BOUSSARD, M.; LU, M. Embodying services into physical places: Toward the design of a mobile environment browser. **ACM Trans. Interact. Intell. Syst.**, ACM, New York, NY, USA, v. 3, n. 2, p. 8:1–8:34, ago. 2013. ISSN 2160-6455. Disponível em: <http://doi.acm.org/10.1145/2499474.2499477>.

TURUNEN, M.; SONNTAG, D.; ENGELBRECHT, K.; OLSSON, T.; SCHNELLE-WALKA, D.; LUCERO, A. Interaction and humans in internet of things. In: **Human-Computer Interaction - INTERACT 2015 - 15th IFIP TC 13 International Conference, Bamberg, Germany, September 14-18, 2015, Proceedings, Part IV**. [s.n.], 2015. p. 633–636. Disponível em: <http://dx.doi.org/10.1007/978-3-319-22723-8_80>.

VIANA, W.; MIRON, A. D.; MOISUC, B.; GENSEL, J.; VILLANOVA-OLIVER, M.; MARTIN, H. Towards the semantic and context-aware management of mobile multimedia. **Multimedia Tools Appl.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 53, n. 2, p. 391–429, jun. 2011. ISSN 1380-7501. Disponível em: <http://dx.doi.org/10.1007/s11042-010-0502-6>.

WANG, Y.; LIN, J.; ANNAVARAM, M.; JACOBSON, Q. A.; HONG, J.; KRISHNAMACHARI, B.; SADEH, N. A framework of energy efficient mobile sensing for automatic user state recognition. In: **Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services**. New York, NY, USA: ACM, 2009. (MobiSys '09), p. 179–192. ISBN 978-1-60558-566-6. Disponível em: <http://doi.acm.org/10.1145/1555816.1555835>.

ZASLAVSKY, A.; PERERA, C.; GEORGAKOPOULOS, D. Sensing as a service and big data. **International Conference on Advances in Cloud Computing**, Bangalore, India, p. 21–29, July 2012.

ZHOU, M.; MA, Y. A web service discovery computational method for iot system. In: **2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems**. [S.l.: s.n.], 2012. v. 03, p. 1009–1012. ISSN 2376-5933.

APÊNDICE A – COMPONENTES DO LOCCAM

Neste apêndice encontram-se detalhes sobre os demais componentes que não foram cobertos na Seção 2.3. O objetivo dessa separação foi deixar o texto mais coeso, evitando detalhes de implementação durante a fundamentação teórica do trabalho. A leitura das informações a seguir é recomendada para desenvolvedores ou leitores interessados em entender melhor como aplicações podem utilizar o LoCCAM para interagir com os OIs do ambiente.

A.1 SysSU

O SysSU é um espaço de tuplas que coordena a interação entre aplicações e SACs. Todos os tipos de dados, sejam eles contextuais ou representando um comando a ser executado por um OI, são representados por tuplas, que por sua vez são um conjunto de campos no formato chave/valor. Existem tuplas especiais utilizadas pelo LoCCAM para representar o interesse da aplicação em determinada funcionalidade (e.g, obter temperatura ou acender as luzes da casa), essas tuplas possuem os seguintes campos: i) ***AppId*** representa um identificador único da aplicação, utilizado para gerenciar o grafo de aplicações utilizado para identificar quando uma adaptação se faz necessária; ii) ***InterestElement*** é uma *String* que caracteriza que tipo de informação a aplicação tem interesse naquele momento (e.g, o interesse em temperatura pode ser representado por *context.ambient.temperature*). Antes de ter acesso a alguma informação contextual, aplicações precisam registrar seu interesse nessa informação, permitindo assim que o SAC responsável por fornecer essa informação seja carregado dinamicamente.

O SysSU fornece métodos de acesso ao espaço de tuplas, alguns de forma síncrona e outros de maneira assíncrona. São eles, com suas respectivas variações: i) ***put*** é utilizado para armazenar uma tupla no espaço de tuplas, é uma operação bloqueante, embora o tempo para se armazenar uma tupla seja bem baixo; ii) ***read*** retorna uma lista de tuplas que correspondem aos parâmetros especificados (e.g, todas as tuplas de temperatura), é possível realizar uma leitura bloqueante de modo a retornar a próxima tupla que for armazenada no SysSU, ou uma busca imediata pela última já armazenada; iii) ***take*** é o equivalente ao ***read*** mas também apaga as tuplas selecionadas do espaço de tuplas; iv) ***subscribe*** pode ser utilizado pelos clientes do espaço de tuplas para serem notificados a cada nova tupla solicitada; v) ***unsubscribe*** informa ao SysSU que a aplicação, ou um próprio SAC, não está mais interessado em determinada informação, e

deseja não ser mais informado a cada nova tupla.

A utilização do SysSU traz inúmeros benefícios, pois desacopla a interação, abstraindo aos clientes questões inerentes de um cenário de IoT no qual os dispositivos são voláteis, ou seja, constantes desconexões devido a mobilidade ou deficiência energética. Entretanto, esse desacoplamento causa um custo adicional no tempo de acesso às informações, o que pode ser um problema para aplicações de tempo real (e.g, aplicações de realidade virtual). Nesse caso, o LoCCAM permite que aplicações acessem diretamente os SACs, sem a necessidade da utilização do espaço de tuplas. Assim, aplicações acessam mais rapidamente informações sensoriais em troca de um menor controle com relação a consistência e sincronia dos dados.

A.2 CAM

O *Context Acquisition Manager* (CAM) é responsável por gerenciar as modificações geradas a partir de mudanças nas zonas de interesse e de observação. É composto por dois componentes principais, o *Adaptation Reasoner* e o *SAC Manager*. O objetivo do CAM é garantir que apenas os SACs que representam informações contextuais relevantes a alguma aplicação ativa, estejam de fato carregados na memória do *smartphone* e executando. O mesmo conceito vale para a atuação, se duas aplicações estão controlando um ar condicionado no ambiente, o SAC que permite esse controle continua executando até que as duas aplicações se encerrem ou digam explicitamente que não possuem mais interesse nesse OI.

O *Adaptation Reasoner* cria um grafo usado para monitorar as zonas de interesse e de observação. A zona de interesse é formada pelos elementos informados pelos clientes do espaço de tuplas, por meio da tupla especificada pelos campos ***AppId*** e ***InterestElement***. O *Adaptation Reasoner* realiza uma operação de *subscribe* no SysSU a fim de receber todas as tuplas que possuem esses dois campos. Já a zona de observação nada mais que é que o conjunto de OIs que os SACs podem monitorar ou atuar, a cada novo SAC instalado ou desinstalado, a zona de observação muda. Como o contexto é representado pela interseção entre a zona de interesse e a zona de observação, a cada variação em qualquer uma dessas zonas, o *Adaptation Reasoner* refaz o grafo de interesse e verifica se ocorreu alguma mudança no contexto, em caso positivo, notifica ao ***SAC Manager*** para realizar as alterações necessárias na lista de SACs ativos.

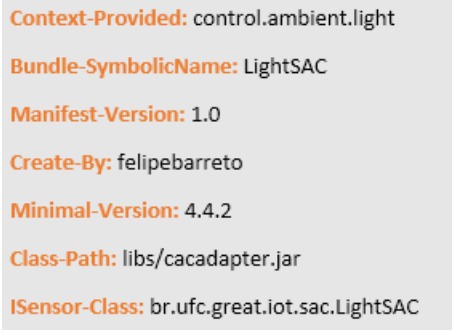
O ***SAC Manager*** é o gerenciador do ciclo de vida dos SACs. A partir de necessidade

de adaptação, o ***SAC Manager*** pode iniciar ou parar um determinado SAC, de modo a garantir que apenas os SACs que são necessários para a execução das aplicações, estejam ativos e carregados na memória. Quando um novo SAC é iniciado, o ***SAC Manager*** cria uma *thread* cuja função é isolar a execução de cada SAC, ou seja, um SAC nunca irá interferir na execução de outro, de maneira direta. Dessa forma, métodos bloqueantes em um SAC não bloqueiam nenhum outro, assim como exceções que ocorram em um componente não são propagadas para os demais. Embora isolados um dos outros, os SACs ainda podem se comunicar, utilizando o espaço de tuplas, seguindo o mesmo paradigma de interação usado pelas diversas aplicações que utilizam o LoCCAM.

A.3 Componentes de Sensoriamento e Atuação

Os Componentes de Sensoriamento e Atuação, ou SAC, encapsulam o acesso a sensores e dispositivos, sendo eles do próprio *smartphone* ou do ambiente, de forma a garantir uma interface única de acesso a esses dispositivos. Cada SAC possui um arquivo de manifesto, utilizado para armazenar os meta-dados referentes ao SAC e que é utilizado pelo ***SAC Manager*** no momento de instalação do componente. Os campos presentes no arquivo de manifesto podem ser visualizados na Figura 31. Dentre esses campos, destacam-se o o ***Context-Provided*** que diz que tipo de informação esse SAC representa (e.g, acesso à informações de luminosidade), ***Minimal-Version*** é importante na hora de realizar o carregamento dinâmico por reflexão, em que apenas os SACs compatíveis com a versão do *Android* são carregados. O campo ***Class-Path*** indica o conjunto de bibliotecas que devem ser compiladas junto ao SAC, e por fim o campo ***ISensor-Class*** é o caminho completo para a classe principal do SAC, sujeita ao mecanismo de reflexão.

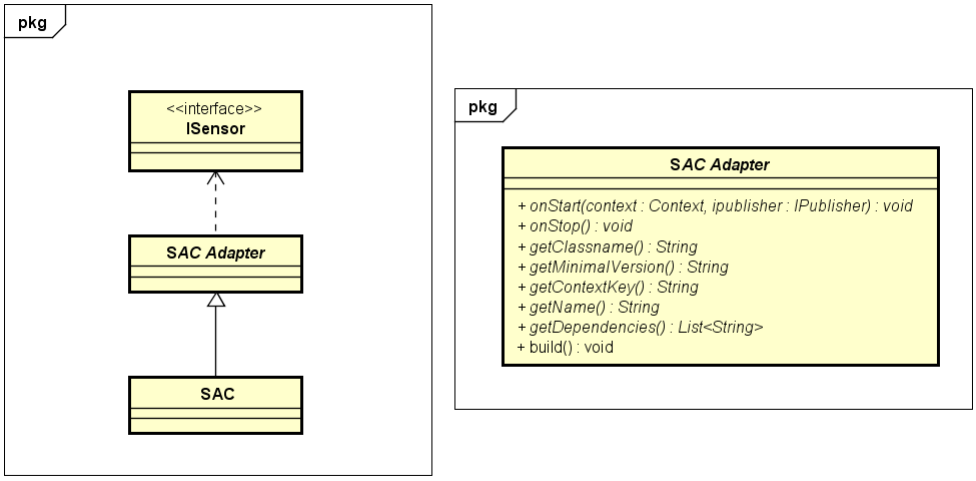
Figura 31 – Arquivo de manifesto dos SACs.



Fonte – Elaborado pelo Autor.

Um SAC, em termos de código, nada mais é do que uma implementação da interface *ISensor*, que possui os métodos necessários para gerenciar seu ciclo de vida, como *start* e *stop*. Porém, para facilitar o desenvolvimento de novos SACs, o LoCCAM disponibiliza uma biblioteca, chamada *SACAdapter* que abstrai toda a criação do arquivo de manifesto e também as etapas de criação do arquivo .jar, que é o formato usado para distribuição de um SAC. A Figura 32 mostra que a maneira ideal de se desenvolver um SAC é estendendo a classe *SACAdapter*, que por sua vez implementa os métodos da interface *ISensor*. Quando o SAC está desenvolvido, basta uma chamada ao método *build* que o SAC já é instalado no LoCCAM. Os demais métodos que o SAC deve implementar por herdar de *SACAdapter* tem objetivo justamente gerar o arquivo de manifesto, já o objeto *iPublisher* recebido no método *onStart* é uma interface para acesso ao SysSU, por onde o SAC pode ler ou escrever tuplas.

Figura 32 – Diagrama de classes de um SAC tradicional.



Fonte – Elaborado pelo Autor.