UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Anderson Soares Costa

Descoberta Dinâmica de Serviços em IoT Baseado em Fluxos de Informações Semânticas de Smart Objects

São Luís
2019

Anderson Soares Costa

Descoberta Dinâmica de Serviços em IoT Baseado em Fluxos de Informações Semânticas de Smart Objects

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de MESTRE em Engenharia de Eletricidade.

Orientador: Francisco José da Silva e Silva

Doutor - UFMA

São Luís

2019

Costa, Anderson Soares

Descoberta Dinâmica de Serviços em IoT Baseado em Fluxos de Informações Semânticas de Smart Objects / Anderson Soares Costa. – São Luís, 2019.

134 f.

Orientador: Francisco José da Silva e Silva.

Impresso por computador (fotocópia).

Dissertação (Mestrado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade. São Luís, 2019.

1. Internet das Coisas. 2. Representação Semântica. 3. Descoberta de Serviço. I. da Silva e Silva, Francisco José, orient. II. Título.

CDU 004

Anderson Soares Costa

Descoberta Dinâmica de Serviços em IoT Baseado em Fluxos de Informações Semânticas de Smart Objects

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Anderson Soares Costa e aprovada pela comissão examinadora.

Aprovada em

BANCA EXAMINADORA

Francisco José da Silva e Silva (orientador)

Doutor - UFMA

Denivaldo Cícero Pavão Lopes

Doutor - UFMA

Marcos Roriz Junior

Doutor - UFG

Aos meus pais, meus irmãos, minha namorada, meus amigos e meus professores.

Resumo

A Internet das Coisas (IoT) é uma combinação entre a computação ubíqua e a Internet na qual os dispositivos da IoT (*smart objects*) podem coletar e trocar dados, cooperando com pessoas e o ambiente no qual se encontram. Já a Internet das Coisas Móveis (IoMT), que é uma extensão da IoT, propõe cenários nos quais os *smart objects* e *gateways* são móveis. Nesse contexto, este trabalho está voltado à descoberta de *smart objects* em ambientes de IoT/IoMT considerando os seguintes problemas: a mobilidade tanto de *smart objects* quanto os *gateways*; grande heterogeneidade de *smart objects* e tecnologias de comunicação para acesso aos mesmos; a necessidade de interoperabilidade nesses ambientes; necessidade de combinar dados dos *smart objects* com bases de conhecimento, de forma a permitir consultas complexas para a descoberta de serviços disponibilizados pelos *smart objects*. Diante disso, o objetivo deste trabalho é combinar Processamento de Fluxos Semânticos com técnicas de representação de conhecimento para enriquecer a descoberta instantânea e contínua de *smart objects* e seus serviços em ambientes da IoT/IoMT. Para esse fim, foi desenvolvida uma ontologia capaz de descrever cenários da IoT/IoMT, um *middleware* semântico, uma API para construção de sistemas de informações e aplicações e uma infraestrutura em nuvem para o processamento de consultas e de fluxos semânticos de *smart objects*. A avaliação qualitativa deste trabalho é feita através de um caso de uso no domínio de estacionamentos inteligentes, que serviu para mostrar que os mecanismo proposto se aplica a diversos domínios de aplicação. Já a avaliação quantitativa provou a eficiência da solução em relação ao tempo do processamento de cada componente, além da identificação dos fatores de influência no tempo do processamento semântico.

Palavras-chave: Descoberta de serviço; Representação Semântica; Internet das Coisas; Ontologia; Bases de Conhecimento;

Abstract

The Internet of Things (IoT) is a combination of ubiquitous computing and the Internet, in which IoT (smart objects) devices can collect and exchange data, cooperating with people and the environment in which they find themselves. The Internet of Things Mobile (IoMT), which is an extension of IoT, proposes scenarios in which smart objects and gateways are mobile. In this context, this work is focused on the discovery of smart objects in IoT/IoMT environments considering the following problems: mobility of both smart objects and gateways; great heterogeneity of smart objects and communication technologies to access them; the need for interoperability in these environments; the need to combine data from smart objects with knowledge bases. Therefore, the objective of this work is to combine Semantic Flow Processing with knowledge representation techniques to enrich the instantaneous and continuous discovery of smart objects and their services in IoT/IoMT environments. To this end, an ontology was developed to describe IoT/IoMT scenarios, a semantic middleware, an API for building information systems and applications, and a cloud infrastructure for querying and semantic streaming of smart objects. The qualitative evaluation of this work is done through a use case in the field of intelligent parking, which served to show that the proposed mechanisms apply to several application domains. The quantitative evaluation proved the efficiency of the solution in relation to the processing time of each component, besides the identification of the influence factors in the time of the semantic processing.

Keywords: Service Discovery; Semantic Representation; Internet of Things; Ontology; Knowledge Bases.

Agradecimentos

Este trabalho foi fruto de muito esforço depositado por meu orientador e por mim durante a minha vida acadêmica no mestrado. Contudo, nada disso seria possível sem a contribuição direta e indireta de muitas pessoas.

Ao bom Deus em primeiro lugar, pela minha vida e por todas as maravilhas.

Ao meu orientador, o Prof. Francisco José da Silva e Silva, pelo exemplo, apoio, compreensão, orientação, e por acompanhar sempre de perto o andamento das atividades. Nunca entendi como alguém tão ocupado consegue ser tão presente.

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

Aos meus familiares pelo apoio que me deram nesta trajetória, sei o quanto todos torceram e apoiaram desde sempre. Aos meus tios e avós pelo apoio financeiro. Aos meus primos Márcio, João Henrique e Marla Valéria pela companhia, apoio e compreensão. Em especial, aos meus pais que nunca mediram esforços para propor uma educação a mim. Aos meus irmãos, por todos os momentos compartilhados em toda uma vida e mesmo agora à distância. Eles me fazem querer ser alguém melhor todos os dias.

À minha querida e amável namorada pela paciência, compreensão, e, principalmente, pelo apoio incondicional aplicado em todos os momentos desse mestrado. Graças à sua companhia esta tarefa tornou-se menos árdua.

Ao membros do LSDi, com os quais compartilhei alegrias, tristezas, e conhecimento. Em especial, ao meus amigos Danne, Tércio e Rodolfo por estarem presente nessa caminhada. Sem a ajuda deles, essa jornada teria sido muito mais difícil.

A UFMA e ao PPGEE pela estrutura dada a execução deste trabalho. Agradeço também ao chefe do PPGEE, ao coordenador, aos professores, e ao técnico administrativo Alcídes, pela dedicação desses profissionais.

Agradeço também a todos os meus amigos de Pedro II-PI e São Luís-MA, que direta ou indiretamente proporcionaram momentos de descontração e amizade nesses últimos dois anos.

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original."

Albert Einstein

Lista de Figuras

Lista de Tabelas

Lista de Siglas

C-SPARQL	*Continuous SPARQL Protocol and RDF Query Language*.
IoMT	*Internet of Mobile Things*.
IoT	*Internet of Things*.
JSON	*JavaScript Object Notation*.
LAC	*Laboratory for Advanced Collaboration*.
LSDI	Laboratório de Sistemas Distribuídos Inteligentes.
M-Hub	*Mobile Hub*.
MEPA	*Mobile Event Processing Agent*.
MQTT	*Message Queuing Telemetry Transport*.
MSM	Modelo de Serviço Mínimo.
OWL	*Web Ontology Language*.
PUC-Rio	Pontifícia Universidade Católica do Rio de Janeiro.
QoS	*Quality of Service*.
RDF	*Resource Description Framework*.
RDF-S	*RDF Schema*.
RQ	*Response Query*.
RSL	Revisão Sistemática da Literatura.

SDDL *Scalable Data Distribution Layer*.

SM-Hub *Semantic Mobile Hub*.

SOSA *Sensor, Observation, Sample, and Actuator*.

SP *Semantic Processor*.

SPARQL *SPARQL Protocol and RDF Query Language*.

SSL *Secure Sockets Layer*.

SSN *Semantic Sensor Networks*.

SSQ *Sub Stream and Query*.

TLS *Transport Layer Security*.

UFMA Universidade Federal do Maranhão.

URI *Uniform Resource Identifier*.

W3C *Wide Web Consortium*.

WPAN *Wireless Personal Area Network*.

XML *Extensible Markup Language*.

Sumário

1 Introdução

A Internet das Coisas (*Internet of Things* (IoT)) propõe a extensão da atual estrutura da Internet para uma rede de objetos (também conhecido como *smart objects*) interligados para prestar serviços de transferência de informação, análises, comunicações e aplicações [29, 39], independentemente do tempo e do lugar [47]. Os *smart objects* possuem endereços exclusivos que permitem a troca de dados por meio de protocolos de comunicação, possibilitando o acesso aos recursos ofertados pelos mesmos por meio da Internet [29] ou por meio de tecnologias de comunicação de curto alcance. A IoT integra aspectos e tecnologias de diferentes áreas, tais como: ciência de contexto, computação ubíqua, protocolos e tecnologias de comunicação, *smart objects* e redes. Estima-se que, por volta de 2020, mais de 20 bilhões de *smart objects* estejam conectados à rede.

Uma extensão da IoT denominada de Internet das Coisas Móveis (*Internet of Mobile Things* (IoMT)) trata de cenários nos quais os *smart objects* podem ser movidos ou mover-se de forma autônoma (*e.g.*, veículos com sensores ou robôs com mobilidade), mas permanecendo acessíveis remotamente a partir de um *gateway* também móvel (e.g., *tablets*, *smartphones*). Com isso, é possível o desenvolvimento de aplicações móveis que descobrem e interagem com objetos inteligentes em todos os espaços, de maneira oportunista e ubíqua [20].

A expectativa com relação ao paradigma da IoT/IoMT é muito grande, pois espera-se que tenha muitas aplicações em vários cenários, como: aplicações de controle de ambiente; aplicações na área da saúde; aplicações de produção e automação industrial, logística, entre outros [6]. Essas aplicações, possuem requisitos de baixa latência para a aquisição e o processamento de dados de *smart objects*, e algumas delas requerem a combinação desses dados com uma base de conhecimento a fim de identificar situações de interesse para a aplicação e extrair informações necessárias para a operação correta, eficiente e segura dos sistemas e seus ambientes físicos.

Nos domínios da IoT/IoMT os dados podem ser transmitidos de maneira dinâmica e no formato de fluxos de dados. Fluxos de dados constituem uma sequência

ordenada de eventos [37] que aconteceram no mundo real ou em um sistema de software. Como normalmente é impossível armazenar um fluxo de dados na sua totalidade [8], é preciso consumi-lo rapidamente através de consultas contínuas [7], que são executadas continuamente sobre os fluxos de dados.

O serviço de descoberta é um aspecto importante nos cenários de IoT/IoMT, pois permite aos usuários e aos *smart objects* encontrar e utilizar os recursos disponíveis no ambiente baseados por meio de critérios de buscas [27]. Nesse contexto, este trabalho está voltado à descoberta de *smart objects* em ambientes da IoT/IoMT considerando os seguintes problemas:

- A mobilidade tanto de *smart objects* quanto de *gateways*, tornando-se disponíveis e indisponíveis constantemente no ambiente, além da desconexão intermitente de redes sem-fio [27];
- A grande heterogeneidade de *smart objects* e tecnologias de comunicação para acesso aos mesmos, devido a grande diversidade de *smart objects* e protocolos de tecnologias de curto alcance, como *Bluetooth* e *ZigBEE*. Em cenários da IoMT, essa tarefa é ainda mais complexa, pois não se conhece todos os tipos de *smart objects* que os *gateways* encontrarão ao longo do trajeto;
- A necessidade de interoperabilidade entre o mecanismo de descoberta de *smart objects*, diversas plataformas de *middleware*, *frameworks* e as aplicações da IoT/IoMT, em virtude da enorme quantidade de dados produzidos de diversas fontes e formatos diferentes [39];
- Diversas aplicações requerem combinação de informações dos *smart objects* com bases de conhecimento de forma a estabelecer critérios de combinação na consulta, que chamamos de consultas complexas, como por exemplo, descobrir *smart objects* a partir de sua localização simbólica ou obter os dados do sensor de monitoramento cardíaco de um paciente através do seu nome.

Com o intuito de tratar desses problemas, inicialmente, desenvolveu-se um modelo conceitual, a ontologia MIoT, capaz de expressar conceitos e as relações de ambientes da IoT/IoMT. Após a definição do modelo conceitual, foi proposto um mecanismo integrado a um *middleware* móvel para descoberta, conexão,

comunicação e anotação semântica de *smart objects*, uma API para construções de sistemas de informação e aplicações de IoT/IoMT e uma infraestrutura em nuvem para o processamento de fluxos semânticos de *smart objects* e de consultas instantâneas e contínuas.

1.1 Objetivos

A pesquisa à qual se refere esta dissertação de mestrado tem como objetivo geral combinar o Processamento de Fluxos Semânticos [8] com técnicas de representação de conhecimento, que permitam a especificação de regras e situações complexas, para enriquecer a descoberta instantânea e contínua de *smart objects* e seus serviços em ambientes da IoT/IoMT.

Os objetivos específicos desta pesquisa são:

- Levantar o estado da arte referente a descoberta semântica de serviços em IoT;
- Definir um modelo conceitual para representar os smart objects;
- Analisar, projetar e implementar a arquitetura de software capaz de sanar os desafios expostos neste trabalho;
- Avaliar a proposta qualitativamente e quantitativamente, por meio da implementação de um cenário de uso e da realização de experimentos, respectivamente.

1.2 Metodologia

Para classificar os métodos de pesquisa utilizados nesta dissertação, adotou-se a classificação proposta por Wazlawick [53], voltada para a Ciência da Computação.

Quanto à natureza do trabalho, trata-se de um trabalho original, ou seja, um trabalho que tem como objetivo apresentar conhecimento novo a partir de observações e teorias construídas para explicá-lo. Assume-se que o conhecimento gerado é relevante porque o trabalho tem implicação na forma como se entente os processos e mecanismos de descoberta semântica em IoT/IoMT.

É importante ressaltar que, mesmo que o mecanismo de descoberta proposto possa ser concretizado por meio de um artefato de *software*, esta pesquisa não trata apenas da criação de uma ferramenta, o que seria uma contribuição meramente tecnológica, mas principalmente da proposição de um modelo conceitual e arquitetural que, uma vez comprovada a sua eficiência para o desenvolvimento de aplicações de IoT/IoMT que requerem a combinação de fluxos de dados de *smart objects* com bases de conhecimento, pode ser estendido e reimplementado por terceiros.

Quanto aos objetivos, trata-se de uma pesquisa explicativa, pois além de analisar problemas relacionados a descoberta semântica de *smart objects* em ambientes da IoT/IoMT, ela busca suas causas e explicações.

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

Em relação à pesquisa bibliográfica, esta foi realizada a fim de buscar fundamentação teórica, e para identificar problemas em aberto, analisar os avanços e limitações de outras propostas da literatura e refletir sobre possíveis contribuições ao estado da arte foi adotado a investigação científica denomina Revisão Sistemática da Literatura. Os trabalhos foram obtidos a partir de consultas às principais bases científicas disponíveis na Web, tais como: IEEE, ACM, Scopus e Science Direct.

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

Em relação à pesquisa de levantamento, foram realizados um estudos de caso envolvendo o desenvolvimento de aplicação que requer a combinação do fluxo de dados semânticos com base de conhecimento, a fim de avaliar o mecanismo de descoberta qualitativamente. Foram obtidas conclusões com base em observações pontuais sobre o uso da solução.

Quanto à metodologia de desenvolvimento do mecanismo de descoberta proposto, foram adotados os princípios dos chamados métodos ágeis de desenvolvimento de software [14] que preveem a construção de sistemas de forma incremental, iterativa, adaptativa e em curtos períodos de tempo. O processo de

desenvolvimento é composto de etapas de levantamento de requisitos, projeto, implementação e testes.

1.3 Estrutura da Dissertação

A organização desta dissertação é a seguinte:

- O Capítulo 2 apresenta a fundamentação teórica, abordando conceitos da Web Semântica, como ontologia e suas linguagens de representação, tecnologias de consultas semânticas (SPARQL e C-SPARQL). É apresentado também o projeto ContextNet com ênfase no *middleware* M-Hub e por fim,o protocolo de comunicação MQTT é apresentado.
- O Capítulo 3 apresenta a solução proposta. Essa apresentação contempla um modelo conceitual para ambientes IoT/IoMT, o *middleware* SM-Hub, a arquitetura e o modelo de programação da API desenvolvida.
- O Capítulo 4 mostra a avaliação qualitativa da solução, que consistiu no desenvolvimento de um caso de uso.
- O Capítulo 5 expõe a avaliação quantitativa, mostrando as metodologias utilizadas, bem como as análises e discussões dos resultados obtidos.
- O Capítulo 6 apresenta uma Revisão Sistemática da Literatura feita para caracterizar o estado da arte da pesquisa, relacionar os principais trabalhos relacionados a descoberta semântica de serviços em ambientes da IoT.
- O Capítulo 7 apresenta as conclusões obtidas a partir desta pesquisa e apresenta trabalhos futuros que podem ser desenvolvidos a partir deste esforço inicial.

2 Fundamentação Teórica

Este capítulo contém a fundamentação teórica, abordando conceitos da Web Semântica, como ontologia e suas linguagens de representação, tecnologias de consultas semânticas (SPARQL e C-SPARQL). É apresentado também o projeto ContextNet com ênfase no *middleware* M-Hub e o protocolo de comunicação MQTT. Algumas considerações são apresentadas no final do capítulo.

2.1 Web Semântica e Ontologia

Tim Berners-Lee et al. [10] descrevem a Web Semântica como uma extensão da rede de Internet atual na qual a informação recebe um significado bem definido, permitindo que computadores e pessoas trabalhem em cooperação. Para alcançar um alto nível de interoperabilidade semântica, deve-se estruturar os dados formalmente, de modo a não existir ambiguidades, permitindo sua interpretação indubitável por máquinas. A Web Semântica tem como objetivo integrar e compartilhar recursos, conseguindo alcançar um alto nível de interoperabilidade semântica. Assim, possibilita objetos e aplicações a interpretar dados de maneira inequívoca.

Dentro desse contexto, a ontologia, que é uma descrição de conceitos e relacionamentos existentes entre eles [28], pode realizar um papel fundamental para a inserção formal de semântica aos dados. Ela é responsável por definir um vocabulário para determinado domínio, permitindo a realização da descrição de um recurso de maneira única.

As ontologias são aplicadas à modelagem, tanto em sistemas baseados em bancos de dados quanto em sistemas de representação do conhecimento [21]. Em Representação do Conhecimento, o termo ontologia tem sido usado desde o século passado para se referir a uma estrutura de entidades representadas por termos de um vocabulário lógico. Nos anos de 1990, ontologia passou a ser usado no contexto da Web Semântica

Na ontologia há uma hierarquia entre os conceitos (classe/subclasse) e para a sua construção é necessário o uso dos seguintes objetos:

- Entidades: descrevem um objeto do mundo real e providenciam uma representação lógica. Por exemplo, podemos representar um sensor de detecção pela entidade <SensingDevice>;
- Atributos: descrevem as propriedades das entidades;
- Relações: representam as ligações entre objetos (entidades e atributos);
- Restrições: condições impostas sobre as entidades, atributos ou relações.

Segundo Mizoguchi [40], quanto mais ontológica for a ontologia, melhor. Isso quer dizer que quanto mais claro e explícitos os conceitos representados, melhor é a ontologia. Para isso é preciso desenvolver uma ontologia com definições informais e especificar conceitos genéricos, a fim de obter um vocabulário compartilhável; modelar apenas conceitos essenciais na resolução das classes em um domínio, com o objetivo de maximizar o reuso.

Existem algumas recomendações para construção de ontologias onde uma ontologia: (i) deve levar em consideração a reutilização de ontologias; (ii) deve ser desenvolvida colaborativamente e de forma incremental; (iii) deve ser flexível o suficiente para abranger diferentes cenários de utilização; deve promover uma conceitualização do domínio e ser fácil de usar, isto é, ser amigável.

Além de servirem como um vocabulário comum, as ontologias permitem um ganho de expressividade e flexibilidade nos mecanismos de busca, pois não se limitam a termos e palavras-chave, mas a qualquer fato que se refere a ela, como estrutura e conceitos, por exemplo.

2.1.1 Linguagem de representação de ontologia

A representação de uma ontologia pode ser de duas formas: (i) representação gráfica para que humanos possam compreende-las, que pode estar no formato de grafos, estrutura de árvore, dentre outras; (ii) representação formal, para que os computadores e agentes inteligentes possam consumi-las. Para esse

fim, a ontologia pode ser descrita em algumas linguagens, sendo as mais populares atualmente RDF, RDF-S e OWL.

Para estruturar a informação semântica de maneira formal pode-se utilizar o *Resource Description Framework* (RDF), que é uma especificação proposta pelo *Wide Web Consortium* (W3C). Sua sintaxe é baseada em *Extensible Markup Language* (XML) e seu vocabulário é baseado em *Uniform Resource Identifier* (URI), que é uma identificador único e universal da informação.

O RDF permite a descrição de um recurso que visa vincular as relações entre entidades [45]. Recursos podem ser qualquer coisa, tanto abstratas quanto concretas, por exemplo, uma determinada pessoa, um *smart object*. A relação existente entre recursos é representado por meio de modelos estruturados em formato de triplas <Sujeito, Predicado, Objeto>. O sujeito e objeto descrevem dois recursos que têm vinculo; o predicado corresponde esse vínculo do sujeito com o objeto. A Figura 2.5 ilustra através de um grafo, três triplas de um mesmo recurso.

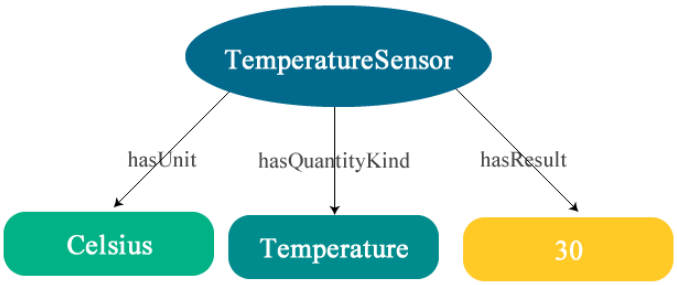

Figura 2.1: Grafo RDF

Esse exemplo mostra um sensor de temperatura estruturado em RDF, no qual o *TemperatureSensor* corresponde ao sujeito, *Celsius*, *Temperature* e 30 os objetos e *hasUnit*, *hasQuantityKind* e *hasResult* representam os predicados, formando assim três triplas. Como se vê, o RDF é uma forma simples de representar triplas.

Já o *RDF Schema* (RDF-S), que é uma extensão RDF que proporciona um maior nível de abstração, possibilita a descrição de classes, superclasses, subclasses e propriedades para os recursos RDF [18]. Estendendo o exemplo da Figura 2.5, o *TemperatureSensor* pode ser uma subclasse de *SensingDevice*, que por sua vez pode ser uma subclasse de *Device*. A classe *Device* pode possuir a propriedade *Name*.

Apesar do RDF-S ser utilizada na descrição de ontologias, ele apresenta algumas limitações, como não suporte a conjunção, disjunção e conectivos lógicos de

negação. Segundo Patel-schneider [31], ela limita o raciocínio computacional, além de limitar expressividade da ontologia. Nessa circunstância, foi desenvolvida uma linguagem mais expressiva denominada *Web Ontology Language* (OWL).

OWL é baseado nas especificações do RDF e RDF-S e atualmente é a linguagem recomendada pelo W3C para o desenvolvimento de ontologias, isto é, o padrão. Ela foi projetada para representar um conhecimento rico e complexo sobre coisas, grupos de coisas e relações entre coisas [5], favorecendo uma maior interoperabilidade do que RDF e RDF-S. Segundo Harmelen e McGuinness [38], a OWL possui três sub-linguagens incrementais:

- OWL Lite: baseado em restrições simples e classificação hierárquica.
- OWL DL: baseado em lógica descritiva, fornecendo um maior nível de detalhamento e restrições.
- OWL Full: projetada para violar restrições da lógica descritiva, com intuito de maximizar a expressividade e liberdade sintática de RDF.

Para acessar os dados RDF, pode-se utilizar o *SPARQL Protocol and RDF Query Language* (SPARQL) que é uma linguagem de consulta e um protocolo para acesso a RDF elaborado pelo W3C RDF *Data Access Working Group* [30]. Ela é uma das principais tecnologias da web semântica e em 2008 tornou-se uma recomendação oficial da W3C. Há uma série de combinações de filtros e comandos que pode ser usado com SPARQL para combinar e extrair informações.

2.1.2 Ontologias para IoT

Os *smart objects* oferecem uma série de serviços, como por exemplo, dados de um sensor ou comando para controla-lo (*e.g.*, cores em uma *smart bulb*). Os serviços e valores dos *smart objects* podem ser descrito semanticamente a partir de uma ontologia, ao invés dos valores atribuídos por seus fabricantes, facilitando assim sua descoberta. Além disso, os termos expressos na ontologia podem ser utilizados como um vocabulário comum para troca de mensagens entre ferramentas tecnológicas (*e.g.*, *middleware*, *frameworks* e aplicações), para realização do raciocínio sobre os dados produzidos pelos *smart objects* e para a geração de conhecimento [3].

Alguns projetos conseguiram conceber ontologias com bom nível de maturidade, tornando-se uma base para novas ontologias. Logo, são constantemente estendidas por pesquisas que desenvolvem ontologias para um determinado domínio da IoT. Dentre as várias ontologias para este domínio, destaca-se a *Semantic Sensor Networks* (SSN), a M3-Lite e a IoT-Lite, já que elas foram utilizadas para gerar uma ontologia de referência para o projeto.

SSN

A ontologia SSN (*Semantic Sensor Networks*) [16] descreve os sensores e suas observações, os procedimentos envolvidos, as características estudadas de interesse, as amostras utilizadas e as propriedades observadas, bem como atuadores. Ela segue uma arquitetura de modularização horizontal (os módulos que estão em camadas horizontais podem depender uns dos outros, ou seja, eles podem confiar na importação direcional de outro módulo horizontal) e vertical (os módulos verticais são construídos um sobre o outro, ou seja, eles são importados de níveis direcionalmente mais baixos), incluindo uma ontologia básica, mas autônoma, chamada *Sensor, Observation, Sample, and Actuator* (SOSA) para suas classes e propriedades elementares, como mostra a Figura 2.2.

Com o seu diferente alcance e diferentes graus de axiomatização, a SSN é capaz de suportar uma ampla gama de aplicações e casos de uso, incluindo imagens de satélites, monitoramento científico em grande escala, infra-estruturas industriais e domésticas, detecção social, ciência cidadã.

A ontologia pode descrever sensores, a precisão e as capacidades de tais sensores, observações e métodos usados para detecção. Também são incluídos conceitos para faixas de operação, pois elas geralmente fazem parte de uma determinada especificação para um sensor, juntamente com seu desempenho dentro dessas faixas. Finalmente, uma estrutura para implantações de campo é incluída para descrever a duração da implantação e a finalidade de detecção do instrumento de macro implantado.

A ontologia completa consiste em 41 conceitos e 39 propriedades de objetos, isso tudo dividido por entre 10 módulos [16]. A Figura 2.2 mostra uma visão desta ontologia.

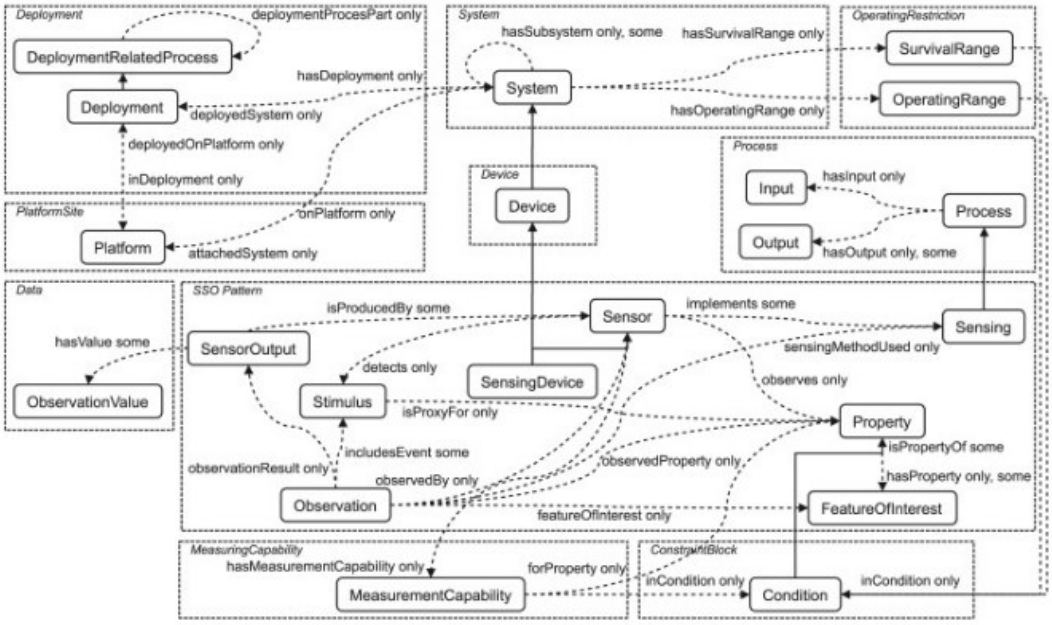

Figura 2.2: Visão geral da ontolgia SSN [16]

M3-Lite

A M3-Lite [2] é uma ontologia para anotar semanticamente e interpretar facilmente os dados da IoT. Ela permite projetar aplicativos interoperáveis específicos de domínio ou de domínio cruzado. Ela aborda principalmente a interoperabilidade semântica em dados de sensores e raciocínio em dados para a construção de aplicativos.

A taxonomia da M3-Lite[1] é uma versão otimizada da ontologia M3 [2], modificada para ambientes da IoT. Essa mudança foi crucial, pois os dados da IoT são provenientes de testes heterogêneos usando termos diferentes para descrever, por exemplo, um mesmo fenômeno físico. O principal benefício da taxonomia M3-lite é alinhar e interligar várias ontologias relacionadas à IoT para facilitar a interoperabilidade.

A M3-Lite faz referência a mais de 30 sensores, medidas, unidades e cerca de 10 domínios diferentes. Seus grande diferencial é a especificação de campos de interesse, a partir do conceito *FeatureOfInterest*, entre eles saúde e agropecuária, que permite uma maior organização das aplicações que utilizarem a ontologia, como apresentado na Figura 2.3.

[1]http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot/doc

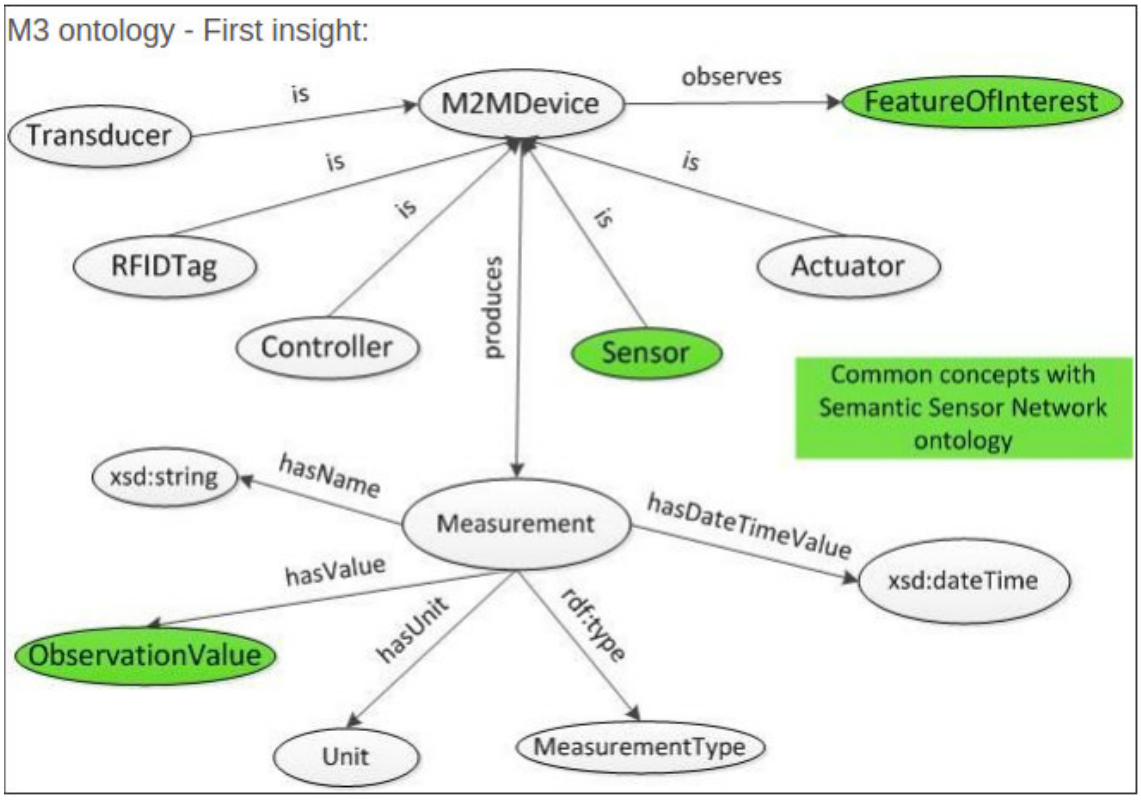

Figura 2.3: Visão geral da ontologia M3-Lite [2]

Outro conceito de bastante importância na M3-Lite é o *MeasurementType* que serve para definir um tipo de medição, como por exemplo um sensor de temperatura, mede dados do tipo temperatura; um sensor de velocidade mede dados do tipo velocidade.

IoT-Lite

A IoT-Lite [9] 2.4 é uma ontologia para representar os recursos, entidades e serviços da Internet da IoT, reutilizando alguns conceitos da SSN. Ela reduz a complexidade de outros modelos de IoT, descrevendo apenas os principais conceitos do domínio IoT. O fato de ser leve permite a representação e o uso de plataformas IoT sem consumir tempo de processamento excessivo ao consultar a ontologia. No entanto, também é uma meta ontologia que pode ser estendida para representar conceitos IoT de forma mais detalhada em diferentes domínios.

IoT-Lite descreve conceitos de IoT em três classes. Objetos, sistema ou recursos e serviços. A IoT-Lite é focada em detecção, embora tenha um conceito de alto nível de atuação que permita qualquer extensão futura nessa área. Os serviços

são descritos com uma cobertura, que pode ser um retângulo ou circulo, por exemplo. Essa cobertura representa o espaço 2D coberto pelo dispositivo IoT. A Figura 2.4 mostra uma visão geral dessa ontologia.

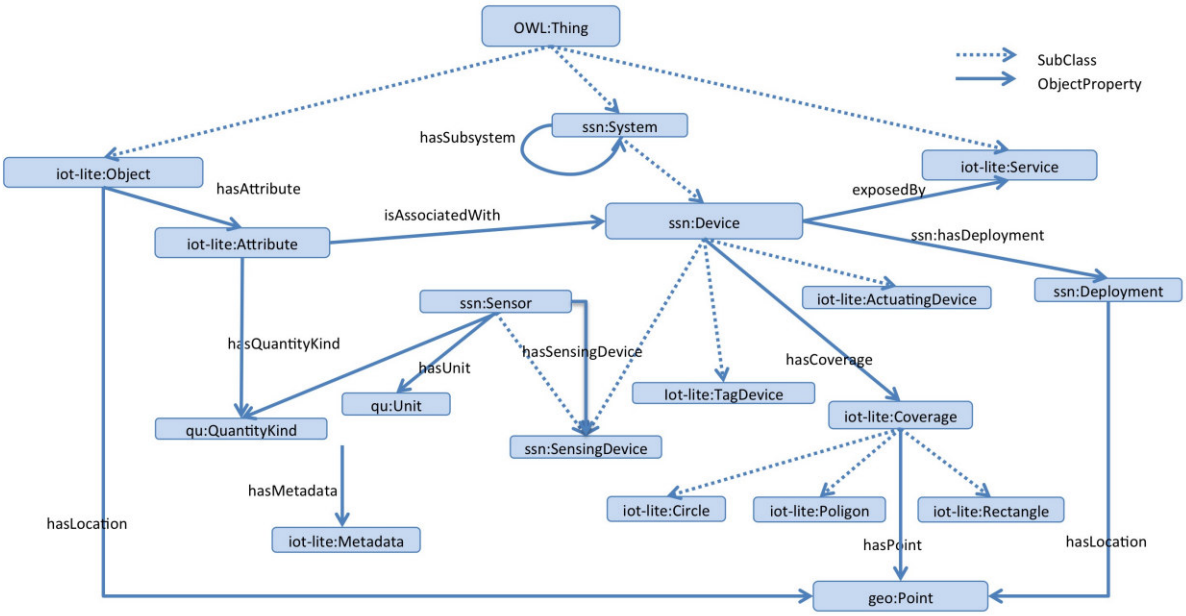

Figura 2.4: Visão geral da ontologia IOT-Lite [9]

2.2 C-SPARQL

O formato RDF foi amplamente aceito pela comunidade científica para representar semanticamente informações armazenadas, que por sua vez, podem estar armazenadas, por exemplo, em um banco de dados relacional com mapeamento para RDF ou em um banco de dados de triplas (*triple store*) [45]. No entanto, para representar um fluxo de dados semânticos, ele deve ser estendido de modo que passe a considerar a temporalidade dos dados, *e.g.*, tempo de ocorrência ou validade dos dados.

O SPARQL é uma linguagem de consulta que permite que usuários acessem dados RDF estáticos. O *Continuous SPARQL Protocol and RDF Query Language* (C-SPARQL) é uma extensão do SPARQL, cuja característica distintiva é o suporte de consultas contínuas, ou seja, consultas registradas em fluxos de dados RDF e em seguida, executadas continuamente [8]. O fluxo RDF considerado em C-SPARQL é uma sequência ordenada de pares RDF, onde cada par é constituído por uma tripla

RDF e seu *timestamp*, sendo representado da seguinte forma: <Sujeito, Predicado, Objeto, *Timestamp*>, formando assim uma quádrupla.

Esses fluxos são processados em tempo próximo ao real e podem ser usados para a geração, por exemplo, de uma mensagem de alerta quando a temperatura subir consideravelmente. Eventualmente eles podem ser combinados com dados armazenados em uma base de dados RDF [8] por meio de uma consulta complexa. Essa combinação pode aumentar significativamente o potencial de análise dos mesmos, pois permite a especificação de regras de processamento mais complexas e de mais alto nível, pois não levam em consideração apenas os dados carregados nos fluxos, mas a combinação dos mesmos com o conhecimento armazenado de forma contínua. Como exemplo desta capacidade, suponha que uma aplicação deseja descobrir todos os serviços de *smart objects* disponíveis em um ambiente físico. Caso o fluxo de eventos traga somente o nome do sensor, seu valor e sua localização geográfica, como por exemplo o sensor de temperatura com 30ºC e latitude e longitude de -2.5571836, -44.3093764, é necessário a combinação dessa informação com a base de conhecimento que possui o mapeamento semântico do espaço físico de cada , para descobrir que uma dada sala, por exemplo, está com temperatura acima do normal.

Através de uma estrutura baseada em regras [50], é possível extrair informações sobre padrões de relacionamentos entre eventos, aplicar filtros, realizar agregações sobre eventos e trazendo a capacidade de cálculo de funções sobre conjuntos de eventos tais como: médias, máximos, mínimos, dentre outros, permitindo assim realizar consultas complexas no fluxo e no conjunto de RDF armazenados. O C-SPARQL oferece, também, diversas funções pré-construídas como: operadores matemáticos (“+”, “-”, “*”, “/”), operadores lógicos (“||”, “&&”, “!”), operadores comparativos (“=,” “>”, “<”, ...), etc. Outra possibilidade é a especificação de janelas de tempo para o processamento de sequências de eventos que ocorram dentro das mesmas.

Precisamente, uma janela de tempo é um contexto temporal [71] [40] que subdivide um fluxo de eventos em intervalos de tempo usando o atributo *timestamp*. Uma janela de tempo divide o fluxo de eventos, de tal forma que inclui somente eventos que estão dentro do intervalo dado. Por exemplo, ao declarar um janela de tempo de 30 segundos, é criado automaticamente uma partição de contexto temporal que retém os últimos eventos que aconteceram em 30 segundos. Esse conceito é

útil para garantir que apenas os eventos mais recentes do fluxo de dados sejam considerados.

Os dois tipos de janelas de tempo em CSPARQL são:

- *Sliding*: janela que avança progressivamente através de um determinado *STEP*, que é um intervalo de tempo mais curto do que o intervalo de tempo da janela. Ao deslizar a janela de tempo é possível incluir os eventos adjacentes anteriores que seriam colocados em lotes diferentes.
- *Tumbling*: janela que avança exatamente em um intervalo de tempo a cada interação. Ela armazena todos os eventos produzidos durante um intervalo de tempo e aplica as consultas contínuas a todo esse conjunto de eventos. Nesse tipo. cada evento é incluído exatamente em uma janela, enquanto que, com janelas *sliding*, alguns eventos podem ser incluídos em várias janelas.

2.3 ContextNet

O projeto intitulado ContextNet, desenvolvido pelo *Laboratory for Advanced Collaboration* (LAC) da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), com colaboração do Laboratório de Sistemas Distribuídos Inteligentes (LSDI) da Universidade Federal do Maranhão (UFMA), visa o desenvolvimento de *middleware* e serviços cientes de contexto para aplicações de colaboração móvel em larga escala, como monitoramento de nodos móveis on-line ou coordenação de atividades de nodos móveis.

Os principais componentes desenvolvidos no âmbito do projeto ContextNet são o *Mobile Hub* (M-Hub) e o *Scalable Data Distribution Layer* (SDDL) [20, 24, 26]. M-Hub é um serviço de *middleware* responsável pela aquisição de dados brutos a partir de sensores via *Wireless Personal Area Network* (WPAN). Já o O SDDL é um *middleware* de comunicação que conecta nodos estacionários em uma rede cabeada (a nuvem SDDL) a nodos móveis por meio de uma conexão baseada em redes IP sem fio.

No âmbito desta dissertação, como mostrado no Capítulo 4, foram estendidos e/ou modificados alguns componentes da arquitetura do M-Hub, a fim

de adequá-las aos objetivos da solução proposta. Essa solução não usa o SDDL como camada de distribuição, mas sim outra, projetada e avaliada neste trabalho.

A seguir, apresenta-se resumidamente a arquitetura e os componentes do M-Hub, referentes à versão original do projeto ContextNet. Por razões didáticas, as explicações sobre quais os componentes específicos do M-Hub foram reutilizados, bem como os detalhes sobre as extensões/modificações realizadas nesses componentes, são fornecidos no Capítulo 4, junto com a descrição da solução proposta.

2.3.1 Mobile Hub (M-HUB)

O *middleware* M-Hub [51] é um *middleware* de uso geral (executado em um dispositivo móvel pessoal convencional), responsável pela descoberta, conexão e coleta de dados dos *smart objects* (e.g., sensores, atuadores, relógios, robôs, veículos) próximos e acessíveis a partir de tecnologias WPAN como Bluetooth Classic, BLE, ZigBee, NFC, ANT+. Desse modo, o M-Hub é um *gateway* de IoMT, pois serve de ponte entre os *smart objects* e a Internet [20,26].

Esse suporte à mobilidade é bastante útil em cenários em que os *smart objects* ou o gateway (M-Hub) ou ambos são móveis. Há cenários em que o M-Hub está parado enquanto os *smart objects* se movimentam no ambiente, se aproximando ou se afastando do mesmo. Há também cenários onde os *smart objects* são estáticos enquanto o M-Hub se desloca no ambiente, realizando uma descoberta e comunicação de forma oportunísticas. Por fim, há cenários em que ambos estão em movimento [20,26].

A Figura 3.2 apresenta a arquitetura que compreende o M-Hub na qual é possível visualizar os principais componentes desse *middleware*.

A descoberta, monitoramento e registro de *smart objects* nas proximidades é feito pelo S2PA (*Short-range Sensing, Presence & Actuation Service*). Ele realiza varreduras periodicamente utilizando diferentes tecnologias WPAN, oferecendo assim suporte à heterogeneidade de protocolos de comunicação. Para minimizar o consumo de energia do dispositivo, o componente *Energy Manager* verifica o nível da bateria periodicamente, e altera a quantidade de consultas realizadas pelo S2PA quando a bateria estiver baixa, por exemplo.

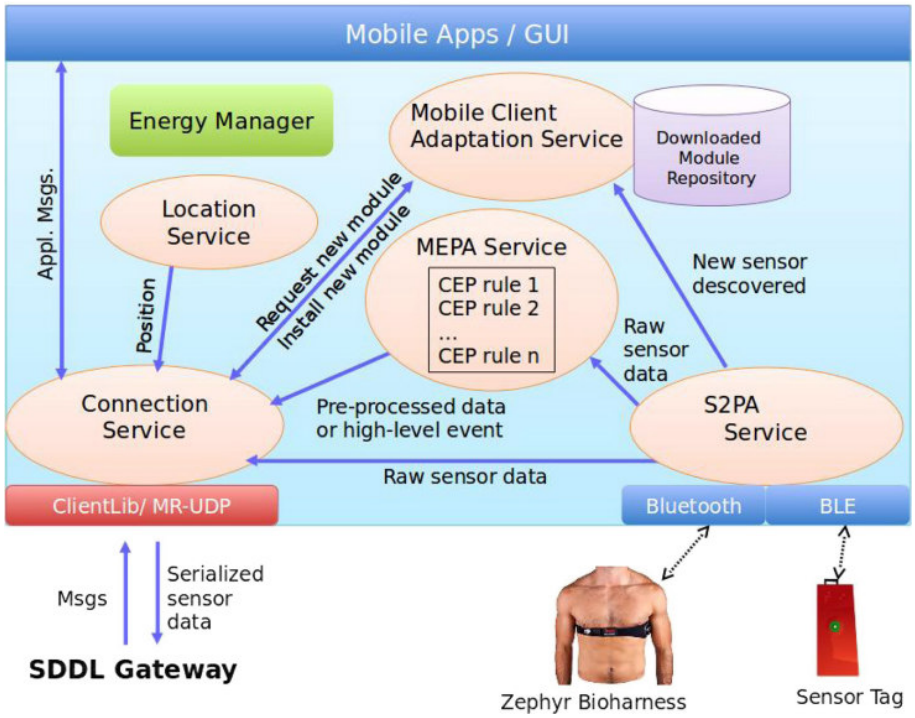

Figura 2.5: Principais componentes do M-Hub [26]

Para processar continuamente os fluxos gerados pelos *smart objects* afim de identificar padrões e detectar eventos complexos definidos pela aplicação móvel, foi desenvolvido o *Mobile Event Processing Agent* (MEPA)*Service*, que utiliza a linguagem EPL Esper da *engine* CEP Asper [51]. Já o componente *Location Service* é responsável por coletar a posição atual do M-Hub e incorpora-la em todos os dados coletados pelo M-Hub. O componente *Mobile Client Adaptation Service* é usado para implantação de novos módulos de sensores e/ou funcionalidades da aplicação, que são baixados a partir de um serviço em execução na nuvem SDDL. Para estabelecer uma comunicação com a Internet, foi desenvolvido o componente *Connection Service*.

2.4 Message Queuing Telemetry Transport (MQTT)

O *Message Queuing Telemetry Transport* (MQTT)[2] é um protocolo *publish/subscribe* implementado sobre o protocolo TCP, oferecendo diferentes níveis de confiabilidade [32, 34]. Nele são utilizados *brokers*, que atuam como intermediários entre os publicadores e subscritores, e tópicos(assuntos) para troca de mensagens entre eles, permitindo o desacoplamento das partes que o utilizam. O MQTT foi adotado

[2]MQTT: http://mqtt.org/

como protocolo de comunicação padrão entre os componentes da solução proposta. Por esse motivo, esse protocolo é explicado mais detalhadamente.

O protocolo MQTT foi projetado para ambientes de rede com alta latência, baixa largura de banda e pouco confiáveis, e para ser executado em dispositivos com recursos limitados em termo de memória, processamento, bateria e conectividade, como *smartphone* e *smart object*, por exemplo. Por apresentar baixo *overhead* de comunicação, o MQTT é considerado um protocolo leve [25], isto é, utiliza poucos recursos de rede e consome pouca energia na comunicação.

Cada cliente se conecta ao *broker* fornecendo seu próprio identificador exclusivo (ID do cliente) e a função do *broker* é gerenciar conexões dos clientes e transferir mensagens entre eles. Além disso, o *broker* pode armazenar os dados para enviá-los a todos os clientes que foram desconectados temporariamente, mas ficaram online novamente. Esse recuso de reter os dados é muito importante em ambientes da IoT, onde há redes não confiáveis com conexões frágeis. Opcionalmente pode-se utilizar o *Secure Sockets Layer* (SSL)/*Transport Layer Security* (TLS) para criptografar a conexão, ou restringir o acesso aos dados por meio de autenticação baseados em usuário e senha.

Quando um cliente não precisa enviar ou não recebe mensagens por um longo tempo, ele deve enviar periodicamente uma mensagem "*keep-alive*" ao *broker* para manter a conexão ativa, caso contrário o *broker* encerra a conexão após um tempo limite (que é especificado pelo cliente). Toda a arquitetura é baseada em TCP/IP, de modo que cada mensagem trocada entre os clientes é empacotada dentro de um pacote TCP.

Esse protocolo oferece três níveis de confiabilidade de entrega, que é a única política de *Quality of Service* (QoS) promovida pelo MQTT [34]. Essa qualidade é associada a mensagem, no qual o *broker* é responsável pela distribuição da mesma. Os níveis são:

- QoS 0: neste nível a mensagem é enviada no máximo uma vez. Pode ocorrer perda de mensagem.

- QoS 1: A mensagem chegará pelo menos uma vez considerando conectividade eventual, neste nível não corre o risco do receptor não receber a mensagem, porém pode ocorrer mensagens duplicadas.

- QoS 2: E o nível apesar de ser o mais lento, é o mais consistente, pois garante que a mensagem será entregue exatamente uma vez.

Um tópico, no protocolo MQTT, é a chave que identifica o canal de informações para o qual os dados de carga útil são publicados [34]. Essa chave é uma cadeia de caracteres UTF-8 que servem para filtrar mensagens trafegadas no *broker*. Um tópico consiste em um ou mais níveis hierárquicos, onde cada nível é separado por um barra inclinada para direita (“/”), por exemplo:

```
laboratorio/sala_servidor/temperatura
```

Nesse exemplo, o tópico permite filtrar com exatidão os dados do sensor de temperatura do laboratório. O primeiro nível ("laboratorio") representa o domínio. O segundo nível ("sala_servidor") indica a comodo do laboratório a ser monitorado. O terceiro nível (“temperatura”) indica o tipo de sensor específico.

Para auxiliar os subscritores nas filtragens das mensagens trafegadas no *broker*, o MQTT permite o uso de caracteres “curingas” (*wildcards*), que são: curinga de nível único (“+”) e curinga multinível (“#”). O primeiro é para substituir um único nível de tópico. No exemplo a seguir apresenta-se como usar esse tipo de curinga para filtrar os dados de sensores de temperatura de todos os cômodos do laboratório.

```
laboratorio/+/temperatura
```

Já o segundo permite que o subscritor receba todas as mensagens trafegadas no *broker* independente dos níveis posteriores ao curinga “#”, cobrindo assim diversos níveis de tópicos. Como por exemplo, filtrar dados de todos os sensores do laboratório, independente do cômodo:

```
laboratorio/#
```

Existem diversas implementações de *brokers* e clientes MQTT em diversos tipos de linguagens de programação como Javascript, Phython, Java, C# e C. Na

implementação da solução proposta, utilizou-se o broker Mosquitto[3] (versão para Windows e Linux), que tem suporte a criptografia e autenticação para acesso e os clientes MQTT Ecplipse Paho[4] (Windows e Linux) e MQTT Eclipse Paho Android Service (versão para Android), sendo todos livres e de código aberto.

2.5 Conclusão

Esse capítulo apresentou um breve fundamental teórico, abordando conceitos da Web Semântica, como ontologias, linguagem de ontologias (RDF, RDF-S e OWL) e tecnologias de consultas semânticas (SPARQL e C-SPARQL). Apresentou-se também o projeto ContextNet, detalhando o *middleware* M-Hub, que funciona como um *gateway* móvel da IoT, apresentando e explicando sua arquitetura e seus principais componentes. Dentre eles, o S2PA que é o componente responsável por realizar a descoberta e a comunicação com os *smart objects* próximos, utilizando tecnologias de curto alcance.

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

[3] http://mosquitto.org/

[4] www.eclipse.org/paho/

3 Solução Proposta

Este trabalho faz uso do processamento de fluxos semânticos, de forma a permitir uma descoberta dinâmica de recursos da IoT/IoMT e aumento da expressividade nas consultas, além de combinar o fluxos de dados semânticos com base de conhecimento para enriquecer a descoberta. Este capítulo apresenta inicialmente a ontologia desenvolvida. Em seguida é apresentado o *middleware Semantic Mobile Hub* (SM-Hub) e a arquitetura de *software* proposta. Ao final desse capítulo apresenta-se a discussão.

3.1 Ontologia MIoT-Ont

A fim de se obter uma ontologia de referência simples e expressiva para representar ambientes da IoT/IoMT, foi desenvolvida uma ontologia denominada MIoT-Ont. Para isso, foi utilizado o método *Development* 101 [36], que possui um guia de passos para desenvolver uma ontologia, que são: (i) determinar o domínio e o escopo da ontologia; (ii) considerar a reutilização de ontologias existentes; (iii) enumerar termos importantes para a ontologia; (iv) definir a estrutura hierárquica das classes; (v) definir as propriedades; (vi) definir as restrições; e (vii) gerar as instâncias.

Diante disso, foram considerados alguns conceitos necessários para ambientes IoT/IoMT, como: dispositivos, sensores e atuadores para identificar os *smart objects* pela sua semântica e não pelo seu identificador definido pelo fabricante; conceitos de localização tanto geográfica como relativa, a fim de gerenciar *smart objects* de forma mais precisa, permitindo descobrir *smart objects* que estão em uma sala, por exemplo.

Após realizar um levantamento bibliográfico sobre ontologias para IoT, foram reutilizados conceitos e suas relações, além das hierarquias da SSN [15], M3-lite [1] e IoT-Lite [9], para o desenvolvimento da MIoT-Ont. A Figura 3.1 apresenta a ontologia proposta em que é possível visualizar suas principais classes. A classe *Device* é uma abstração de um dispositivo que pode ser definido como um aparelho

ou um mecanismo que desenvolve determinadas ações (*e.g*, sensoriamento e atuação). *Sensing Device* e *Actuating Device* são subclasse de *Device* usadas para definir sensores e atuadores, respectivamente. A classe *Unit* define uma unidade de medida de um dado resultante de um sensor. Já a classe *Result* define o resultado de uma medição. O *QuantityKind* representa a essência de uma quantidade sem qualquer valor ou unidade numérica. Pode também ser definido como um tipo de dados de medição. Para descrever os conceitos de localização tanto geográfica quanto simbólica, empregou-se o termo *Point* do vocabulário Geospatial Ontologies utilizado pela IoT-Lite.

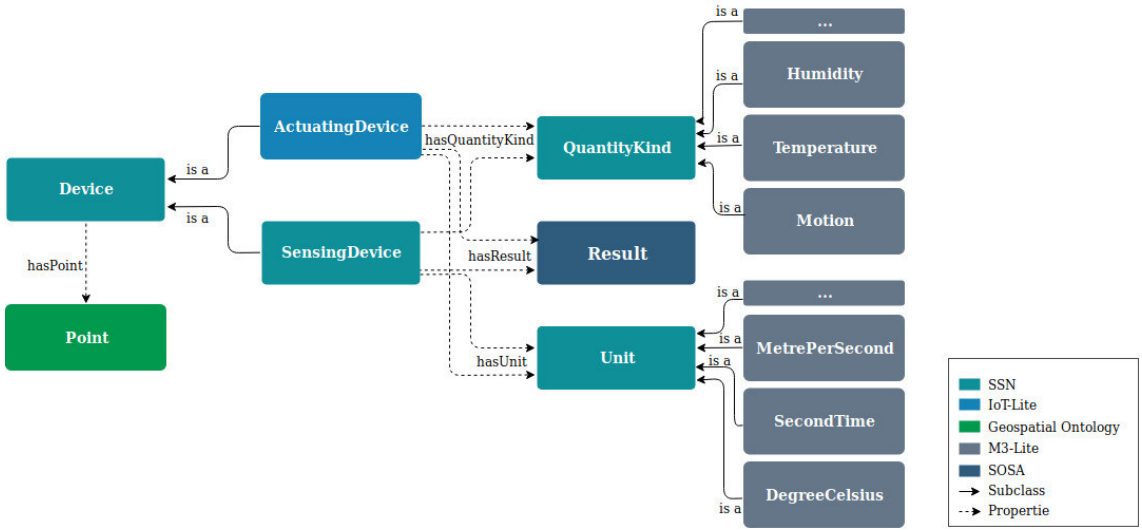

Figura 3.1: Visão geral da MIoT-Ont

Um exemplo de *smart object* que pode ser representado nesta ontologia é uma placa de arduino UNO 2 com um *shield bluetooth* e um sensor de temperatura acoplado nele. Sua modelagem de acordo com a ontologia é apresentada na Tabela 3.1.

Classe	Instância
Device	HmSoft
SensingDevice	TemperatureSensor
QuatityKind	Temperature
Unit	DegreeCelsius
Result	27
Point	Sala 05

Tabela 3.1: Modelagem de um *smart object* conforme a MIoT-Ont

.

Deste modo, com o uso da MIoT-Ont na proposta deste trabalho, é possível oferecer semântica em ambientes da IoT, permitindo descrever cenários nos quais se obtém os valores dos *smart objects* em um determinado cômodo de uma casa, por exemplo. Outros cenários podem explorar esse ambiente, como alterar o valor de *smart object* ou alterar todos os *smart objects* pertencentes ao mesmo tipo (*quatityKind*), como por exemplo, ligar todas as lâmpadas da sala independentemente do fabricante.

A MIoT-Ont permite a interoperação entre os *midlleware*, softwares, aplicações e *framework*, uma vez que é fornecido um vocabulário comum para troca de dados entre os mesmos por meio do seu uso. Ainda, fazendo uso dessa ontologia, é possível realizar anotações semânticas sobre os *smart objects* presentes no ambiente, oferecendo desta forma, suporte a heterogeneidade dos *smart objects*. Neste sentido, este trabalho de pesquisa contribuiu com uma ontologia leve para descrever ambientes da IoT.

3.2 Semantic Mobile Hub (SM-Hub)

Para a introdução de semântica no *midlleware* M-Hub, a fim de adequá-lo aos objetivos da solução proposta, viu-se a necessidade de proceder com um refatoramento [22], de forma a anotar semanticamente os dados obtidos dos *smart objects* a partir da ontologia de referência. Adicionalmente, os serviços providos por *smart objects* e atuadores descobertos dinamicamente no ambiente devem também ser representados através de uma ontologia. Alterou-se as funcionalidades responsáveis pela descoberta de serviços de sensores e atuadores do *middleware* para explorar a flexibilidade que uma busca por conteúdo anotado semanticamente proporciona.

3.2.1 Representação de serviços e dados de *smart objects*

No M-Hub os serviços e os dados de *smart objects* são representados através de um objeto do tipo `SensorData`. Este objeto contém informações de baixo nível dos *smart objects* como: endereço MAC do dispositivo, o indicador de potência do sinal recebido (*received signal strength indicator* - RSSI), a sua ação, que pode ser do tipo de descoberta, de leitura e de conexão ou desconexão, o nome do sensor, o seu valor em um *array* de *Double* e sua localização geográfica.

Para conter os valores semânticos dos *smart objects*, se fez necessário desenvolver uma nova classe denominada `SemanticSensorData`. Esta contém um *arrayList* de informações dos *smart objects* no formato de `RdfQuadruple`. Com isso, os serviços e os dados dos *smart objects* são transformados em triplas RDF de acordo com a ontologia MIoT-Ont, acrescida do tempo de instanciação.

Desse modo, as informações não se limitam em apenas aquela lista de atributos do objeto `SensorData`, pois o `SemanticSensorData` pode possuir, por exemplo, informações sobre o tipo e a unidade de medida do *smart object*, além de oferecer suporte a heterogeneidade do mesmo. O Código 3.1 mostra em JSON um exemplo de uma leitura de um sensor de temperatura através do objeto `SemanticSensorData`.

Código 3.1: JSON do array de Quadruplas

```
{
"rdfQuadruple":[
{
  "object":"http://purl.oclc.org/NET/ssnx/ssn#SensingDevice",
  "predicate":"http://www.w3.org/1999/02/22-rdf-syntax-ns#:Type",
  "subject":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#FFE1-00-1FB",
  "timestamp":1533318129378
},
{
  "object":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#DegreeCelsius",
  "predicate":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#:hasUnit",
  "subject":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#FFE1-00-1FB",
  "timestamp":1533318129378
},
{
  "object":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#Temperature",
  "predicate":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#:hasQuatityKind",
  "subject":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#FFE1-00-1FB",
  "timestamp":1533318129378
},
{
  "object":""25.0"^^http://www.w3.org/2001/XMLSchema/\#float",
  "predicate":"http://www.w3.org/ns/sosa/:hasResult",
```

```
24    "subject":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#FFE1-00-1FB",
25    "timestamp":1533318129378
26  }
27  }
```

3.2.2 Componentes do SM-Hub

A Figura 3.2 apresenta os componentes do M-Hub. Nessa arquitetura, o S2PA (*Short-Range Sensor, Presence and Actuation*) é o serviço responsável pelo gerenciamento das operações de descoberta, conexão e interação com os *smart objects*. Ele é capaz de se comunicar com dispositivos que utilizam diferentes tecnologias por meio da interface `TechnologyCommunication`. Graças a essa interface genérica, o S2PA não precisa conhecer detalhes do hardware e dos protocolos de comunicação.

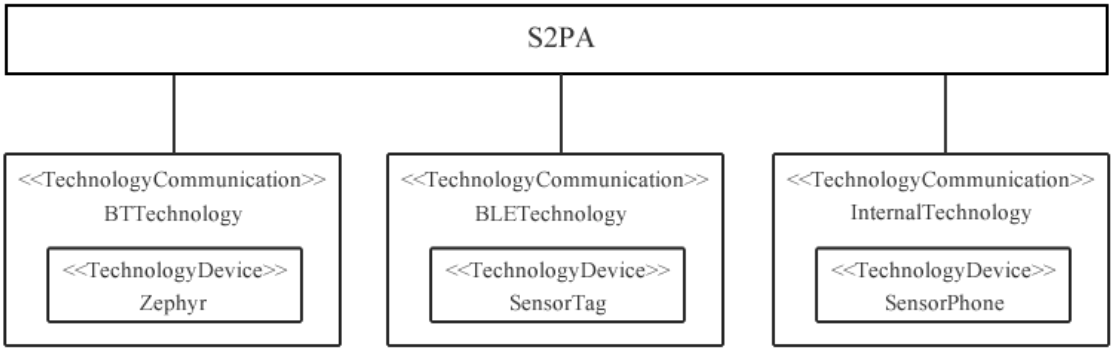

Figura 3.2: Componentes M-Hub

Os componentes BT Technology e BLE Technology implementam as funcionalidades necessárias para a interação com dispositivos que se comunicam por meio das tecnologias WPAN Bluetooth Classic (*e.g.*, Zephyr Bioharness) e Bluetooth Low Energy (*e.g*, Sensor Tag), respectivamente, tendo como base as operações definidas pela interface `Technology`. Já o Internal Technology implementa as funcionalidades de interação com sensores embutidos no mesmo dispositivo Android em que executa o M-Hub. Cada um deles estende a classe `TechnologyDevice`, que possui um método `convert()` que manipula a transformação dos dados brutos (*array* de bytes) recebidos dos *smart objects* para um *array* de *Double*.

A Figura 3.3 apresenta os componentes do SM-Hub, em que se destaca os componentes desenvolvidos no âmbito do trabalho da dissertação.

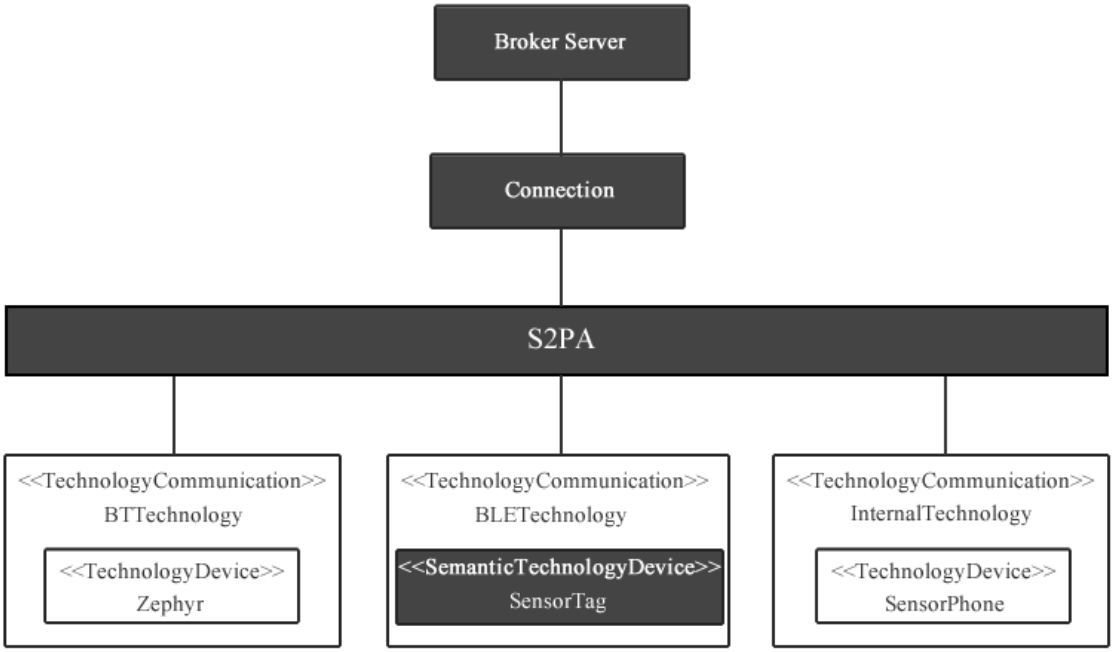

Figura 3.3: Componentes SM-Hub

O `SemanticTechnologyDevice` serve para converter o *array* de bytes recebidos dos *smart objects* em um *array* de quádruplas de acordo com a MIoT-Ont. O S2PA recebe esse *array* e instancia um objeto do tipo `SemanticSensorData`. Para a disponibilização desses dados, foi desenvolvido o componente Connection, que é responsável pelo gerenciamento da conexão e publicação dos dados no *Broker Server*. O protocolo de comunicação adotado foi o MQTT.

Como se vê, o *middleware* SM-Hub, que é uma extensão do *midlleware* Mobile Hub (M-Hub) [51], tem como principal funcionalidade a anotação semântica dos serviços e dos dados dos *smart objects* descobertos dinamicamente a partir da ontologia MIoT-Ont e distribuição dos fluxos semânticos através de uma camada de comunicação utilizando o MQTT.

3.3 Arquitetura de *Software* Proposta

O mecanismo de descoberta desenvolvido foi dividido em três partes: (i) uma biblioteca que fornece uma API (*Application Programming Interface*) para construção de sistemas de informação e aplicações de IoT/IoMT que pode ser executada tanto em *desktops* quanto por dispositivos móveis (*e.g.*, *smartphone*, *tablets*), permitindo o envio de consultas que possibilitam a descoberta de *smart objects*,

consumo de dados e o envio de comandos; (ii) um *middleware Semantic Mobile Hub* (SM-Hub) que atua como uma *gateway* móvel semântico para ambientes da IoT/IoMT; (iii) uma infraestrutura em nuvem desenvolvida para o processamento das consultas e dos fluxos de dados semânticos. Para a comunicação entre a API, infraestrutura em nuvem e o SM-Hub, foi adotado o protocolo *publish/subscribe* MQTT.

A Figura 3.4 mostra uma visão geral da arquitetura proposta, em que é possível visualizar todos os elementos que a compreende, que são:

- **SM-Hub**: *middleware* responsável pela aquisição e anotação semântica dos serviços e dados dos *smart objects* descobertos dinamicamente via WPAN e distribuição dos fluxos semânticos para a infraestrutura em nuvem proposta;
- **Clientes**: dispositivos móveis e desktops que fazem uso da API de descoberta para construir aplicações IoT/IoMT;
- ***Sub Stream and Query* (SSQ)**: componente da infraestrutura em nuvem responsável por manter uma conexão ativa com *broker*, com a finalidade de receber os fluxos eventos semânticos, as consultas e as bases de conhecimentos, e encaminhá-las aos componentes responsáveis;
- ***Response Query* (RQ)**: componente encarregado por receber a consulta do SSQ, instanciar ela no Semantic Processor, definir um *listenner* responsável por receber as respostas e publicar elas em um tópico especifico para que os clientes as recebam;
- ***Semantic Processor* (SP)**: componente responsável por processar consultas e fluxos e armazenar as bases de conhecimentos. Caso esteja especificado na consulta, o SP realiza a combinação dos fluxos de eventos semânticos com uma ou várias bases de conhecimento para fornecer a resposta. Por fim, ele envia as respostas para o RQ.

Todos estes componentes foram desenvolvidos no âmbito deste trabalho de mestrado, com exceção do *broker* MQTT.

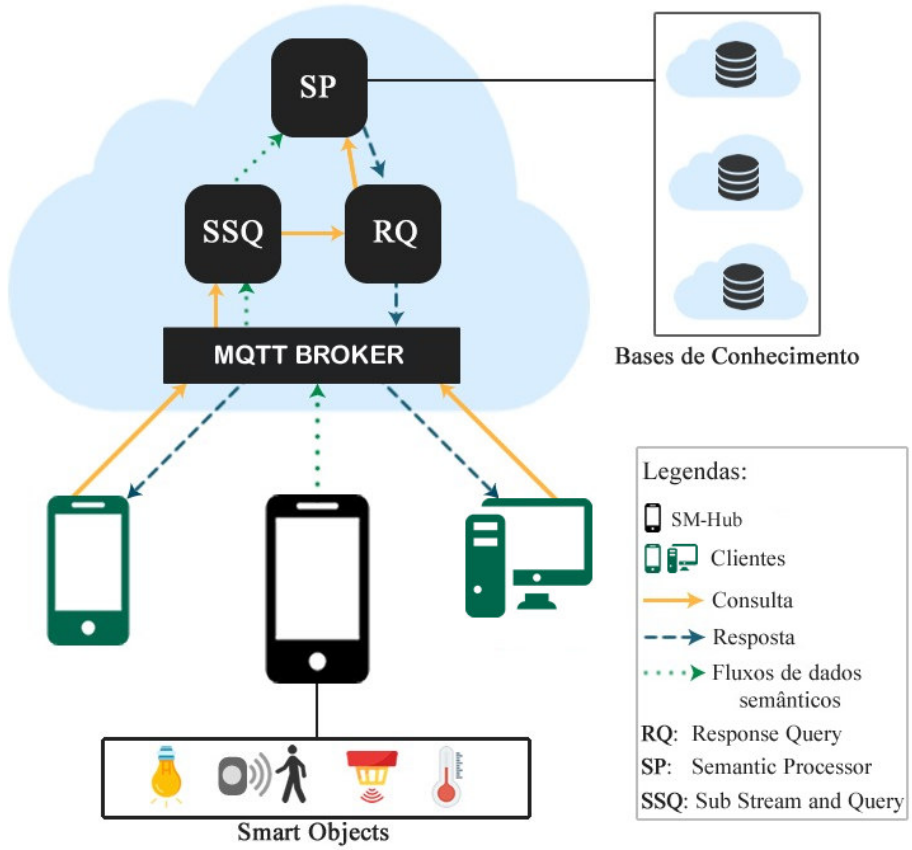

Figura 3.4: Arquitetura proposta

3.4 Registro de Fluxos de Eventos

A solução proposta faz uso de fluxos eventos semânticos e consultas para descoberta de *smart objects* em ambientes da IoT/IoMT. O processo de envio e recebimento de fluxos é ilustrado no diagrama de sequência da Figura 3.5.

Ele inicia-se com o SSQ estabelecendo uma conexão com o *broker* e subscrevendo-se no tópico específico para receber os fluxos. O SM-Hub também estabelece a conexão com o *broker* e em seguida publica os fluxos de eventos no tópico que o SSQ está registrado. O *broker* encaminha a mensagem para o SSQ, que por sua vez extrai cada quádrupla do *array* de quádruplas e envia para o componente SP, em que há uma instância da *engine* C-SPARQL.

3.5 Registro de Bases de Conhecimento

O C-SPARQL suporta bases de conhecimento no formato OWL, RDF-S e bases no modelo Jena. Para que ele possa realizar a combinação do fluxo com as bases,

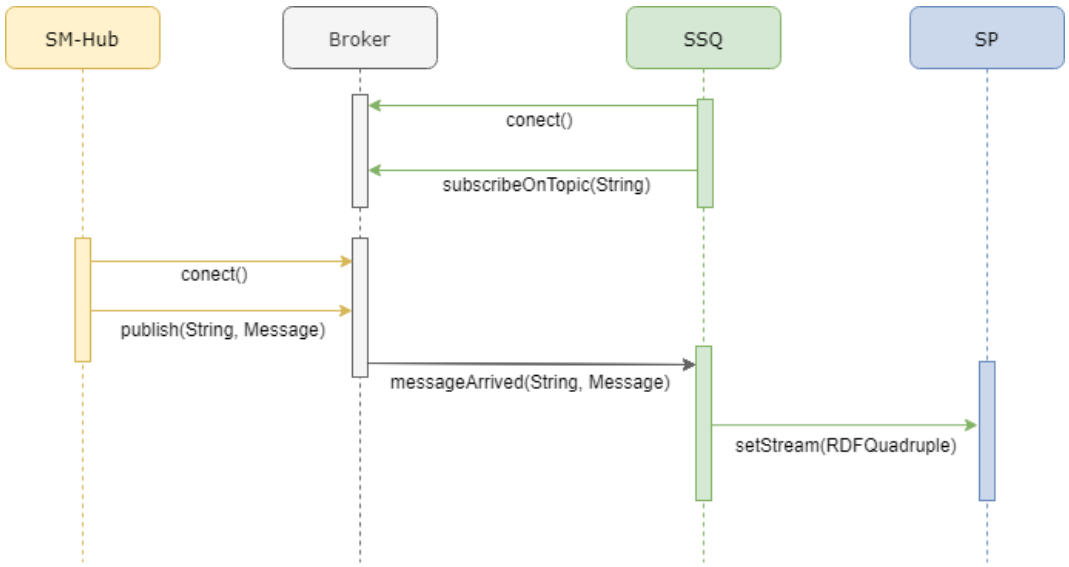

Figura 3.5: Diagrama de sequência do processo de registro de fluxos de eventos

.

é necessário que ele as receba. Para tanto, foi definido e implementado dois métodos na biblioteca desenvolvida: o `putStaticNamedModel` e o `putStaticJenaModel`. O primeiro é utilizado para enviar a base no formato OWL ou RDF-S. Já o segundo é para enviar a base no modelo Jena. Ambos recebem dois parâmetros, o primeiro é o nome da base para que possa ser utilizado na consulta e o outro é a própria base de conhecimento. O Código 3.2 mostra como enviar uma base no modelo Jena.

Código 3.2: Enviando uma base de conhecimento no modelo Jena

```
1    new StaticModel().putStaticJenaModel(String,Model);
```

A Figura 3.6 mostra o diagrama de sequência referente ao processo de envio de base de conhecimento. Pode-se observar que o processo é similar ao do fluxo de eventos, em que única diferença é que quem fornece a base é a aplicação ao invés do SM-Hub.

O processo inicia-se com o SSQ estabelecendo uma conexão com o *broker* e subscrevendo-se no tópico específico para receber a base. A aplicação também estabelece a conexão com o *broker* e em seguida publica a base de conhecimento no tópico que o SSQ está registrado. O *broker* encaminha a mensagem para o SSQ, que por sua vez extrai o modelo Jena e envia para o componente SP, em que há uma instância da *engine* C-SPARQL.

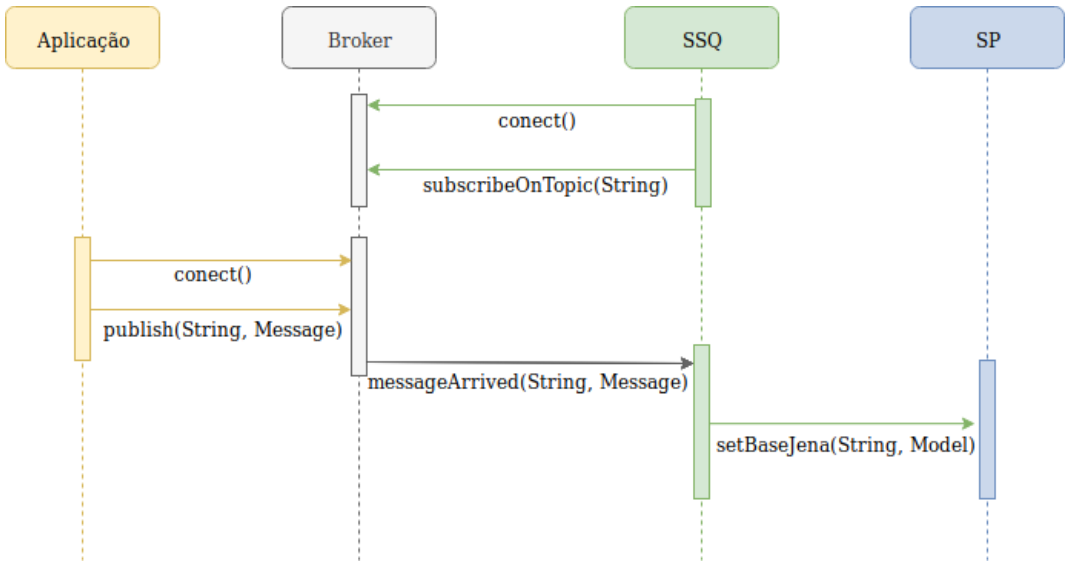

Figura 3.6: Diagrama de sequência do processo de registro de base de conhecimento

.

3.6 Descoberta baseada em Consultas

Nesta abordagem, o cliente só poderá descobrir os serviços disponíveis se definir e publicar uma consulta. Os critérios das consultas podem levar em consideração os dados proveniente do fluxo, da base de conhecimento ou até mesmo a combinação de ambos. A linguagem utilizada para a especificação da consulta é o C-SPARQL, o que garante uma grande expressividade na descoberta de dados semânticos. A Figura 3.7 mostra o diagrama de sequência referente ao processo de consulta.

O processo se inicia com o SSQ estabelecendo uma conexão com o *broker* e em seguida se registrando no tópico onde as consultas serão publicadas. A aplicação cliente se conecta com o *broker*, publica a consulta e logo após se registra no tópico onde as respostas serão publicadas. Na sequência, o SSQ recebe a consulta e envia para o RQ que instancia a consulta no SP. Este componente, por sua vez, realiza o processamento e envia a resposta da consulta para o RQ, que publica a resposta. Por fim, a aplicação cliente recebe a resposta.

É importante ressaltar que o componente SP, além de ter a *engine* C-SPARQL, também possui um método para transformar a resposta da *engine* que vem no formato de tabela RDF (RDFTable) em tuplas RDF (RDFTuple), fazendo com que a aplicação não precise se preocupar com essa transformação.

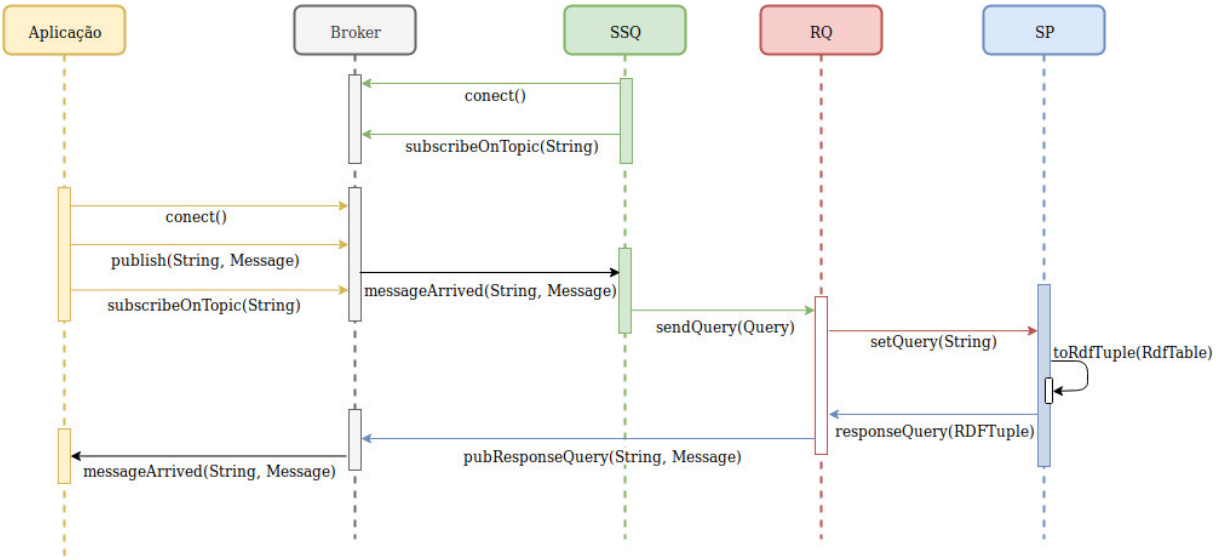

Figura 3.7: Diagrama de sequência do processo de descoberta de serviços

.

A fim de dar suporte a diferentes cenários de IoT, foi definido e implementado dois tipos de consultas: a instantânea e a contínua. As consultas instantâneas retornam os serviços e/ou dados disponíveis que satisfazem os critérios de busca no momento em que a consulta é realizada. Após a resposta ser entregue, a consulta é cancelada. Este modo é útil em situações onde a aplicação precisa descobrir imediatamente e não podem ou não precisam esperar os próximos fluxos de eventos.

No modo contínuo, a consulta é executada continuamente e toda vez que é descoberto um *smart object* cuja as características casem com a consultada especificada, a aplicação recebe a notificação. Este modo é particularmente útil em cenários de mobilidade, em que a disponibilidade dos provedores de serviço varia de acordo com a movimentação dos *smart objects* e/ou do próprio *gateway*. Ambas as consultas podem ser canceladas a qualquer momento pela aplicação.

O Código 3.3 mostra um exemplo de como uma aplicação publica consultas usando a biblioteca desenvolvida. Nesse exemplo, a aplicação constrói um objeto `Query` que contém atributos como: consulta em C-SPARQL, o identificador do publicador da consulta e um booleano para identificar o tipo de consulta (*false* = instantânea e *true* = contínua). Para tanto, foi adotado o padrão de projeto *Builder* [23], que permite separar os passos de construção de um objeto em pequenos métodos. A consulta definida no exemplo visa descobrir a temperatura disponível na sala de servidores. As respostas são recebidas pela aplicação por meio de um

listener específico. Como neste exemplo a consulta é contínua, a aplicação recebe constantemente os valores, possibilitando assim o monitoramento da temperatura da sala.

Código 3.3: Usando a API

```
1  //Consulta em C-SPARQL
2  String qCSPARQL = "SELECT ?data"+
3  "FROM STREAM <http://www.semanticprocessor.br/ontology/laboratory#>"+
4  "[RANGE 5s STEP 1s] FROM <http://mycSPARQL.com/lsdi> WHERE {"+
5  "?object sosa:hasResult ?data ."+
6  "?object iotlite:hasQuantityKind Temperature ."+
7  "?object geo:hasPoint serverRoom }"
8
9  // Instancia o objeto query
10 Query query = new Query.Builder().query(qCSPARQL) //Consulta
11                   .publisherID("exemplo@ex.com") //identificador do publicador
12                   .continuos(true) //tipo da consulta
13                   .build();
14 ResultReceiver result = new ResultReceiver();
15 result.addListener(query, new Listener() {
16 @Override
17 public void update(java.util.Observable o, ArrayList<RDFTuple> rdfTuples){}
```

A aplicação recebe como resultado um *arrayList* de tuplas RDF. Essas tuplas são formadas pela URI do vocabulário acrescido do resultado. Como algumas aplicações não necessitam da URI, desenvolveu-se uma função opcional para o programador que remove todo o vocabulário e deixa apenas o resultado, com intuito de otimizar o desenvolvimento de aplicações. Para isso, basta definir um *arrayList* do tipo *string* para receber o resultado, instanciar a classe e enviar por parâmetro o *arrayList* de tuplas RDF através do método `toString`(), como se verifica no Código 3.4. A Tabela 3.2 mostra o resultado com e sem a URI.

Código 3.4: Formatando o resultado

```
1     ArrayList<String> result = new Formatter().toString(rdfTuples);
```

Com URI	Sem URI
http://purl.oclc.org/NET/UNIS/fiware/iot-lite#Temperature	Temperature
http://purl.oclc.org/NET/UNIS/fiware/iot-lite#DegreeCelsius	DegreeCelsius
"25.0"^^http://www.w3.org/2001/XMLSchema/#float	25.0

Tabela 3.2: Resultados das consultas

3.7 Conclusão

Este capítulo apresentou o mecanismo desenvolvido para a descoberta dinâmica baseada em fluxos de eventos semânticos de *smart objects* para o desenvolvimento de aplicações de IoT/IoMT. Esse mecanismo é composto por uma API para construção de aplicações, um *middleware* que atua como uma gateway móvel semântico (SM-Hub) e uma infraestrutura em nuvem desenvolvida para o processamento das consultas e dos fluxos de eventos semânticos.

O mecanismo desenvolvido neste trabalho simplifica o processo de desenvolvimento de sistemas de informação e aplicações para IoT/IoMT, uma vez que abstrai a complexidade da descoberta de *smart objects* e seus serviços. Por exemplo, o programador não precisa se esforçar para implementar um novo meio de descobrir e interagir com objetos inteligentes, já que o SM-Hub faz essa função. O SM-Hub permite a interoperação entre *middleware* e aplicativos através do MIoT e suporta a heterogeneidade de *smart objects*, uma vez que fornece um vocabulário comum e anotação semântica.

Não é necessário se preocupar com a comunicação com a nuvem, bem como a conFiguração da mesma, pois tudo isso é feito pela API quando é enviado uma consulta. O mecanismo também simplifica as operações de descoberta, filtragem e monitoramento dos *smart objects* por meio do uso da linguagem C-SPARQL. O programador também utiliza as mesmas interfaces/métodos/abstrações independente da plataforma, fazendo com que o programador não adote modelos de programação diferentes. Por fim, considera-se que esse mecanismo soluciona os problemas apresentados para a descoberta, que são: mobilidade tanto de *smart objects* quanto de *gateways*; a grande heterogeneidade de *smart objects* e tecnologias de comunicação para acesso aos mesmos; necessidade de interoperabilidade em

ambientes IoT; necessidade de combinar dados dos *smart objects* com bases de conhecimento, de forma a permitir consultas complexas para a descoberta de serviços disponibilizados pelos *smart objects*.

4 Avaliação Qualitativa

A avaliação qualitativa da solução proposta consistiu no desenvolvimento de um cenário de uso, com objetivo de provar que o mecanismo proposto pode ser aplicado ao desenvolvimento de aplicações IoT/IoMT do mundo real. Em outras palavras, o cenário de uso permitiu observar o comportamento do mecanismo de descoberta por meio de sua aplicação em um domínio específico, bem como verificar se ele é realmente adequado ao problema para o qual foi projetado.

4.1 Cenário de Uso

O cenário de uso consiste em uma aplicação móvel para descoberta de forma contínua de vagas livres em um estacionamento inteligente (*smart parking*) com base no perfil do motorista. Como característica principal, o aplicativo mostra as vagas com base na preferência do motorista como: vagas para idosos, gestantes ou deficientes físicos, por exemplo. Além disso, ele orienta o motorista até uma vaga de interesse, diminuindo o tempo de procura, congestionamento e consumo de combustível.

Para isso, simulou-se um estacionamento, onde cada vaga possui um sensor de presença. O SM-Hub (simulado) após detectar os sensores, conecta-se e realiza a leitura dos dados a cada 5 segundos, anota-os semanticamente conforme a MIoT-Ont e disponibiliza-os para a infraestrutura em nuvem proposta. As aplicações móveis que fazem o uso da API desenvolvida enviam uma consulta de descoberta e recebem continuamente os resultados de vagas livres. A Figura 4.1 demonstra esse cenário.

O fluxo de eventos referentes às vagas seguem desta forma: <PresenceSensorID, "hasResult", *true* ou *false*, *Timestamp*>, levando em consideração o identificador da vaga e seu estado. O resultado "*true*"indica que a vaga está ocupada, por outro lado, o resultado "*false*"significa que a vaga está livre. A fim de simular um fluxo real de detecção de ocupação de vagas em um estacionamento, utilizou-se um *dataset* do CNRPark + EXT[1]. Este possui um conjunto de dados de detecção de

[1]http://cnrpark.it/

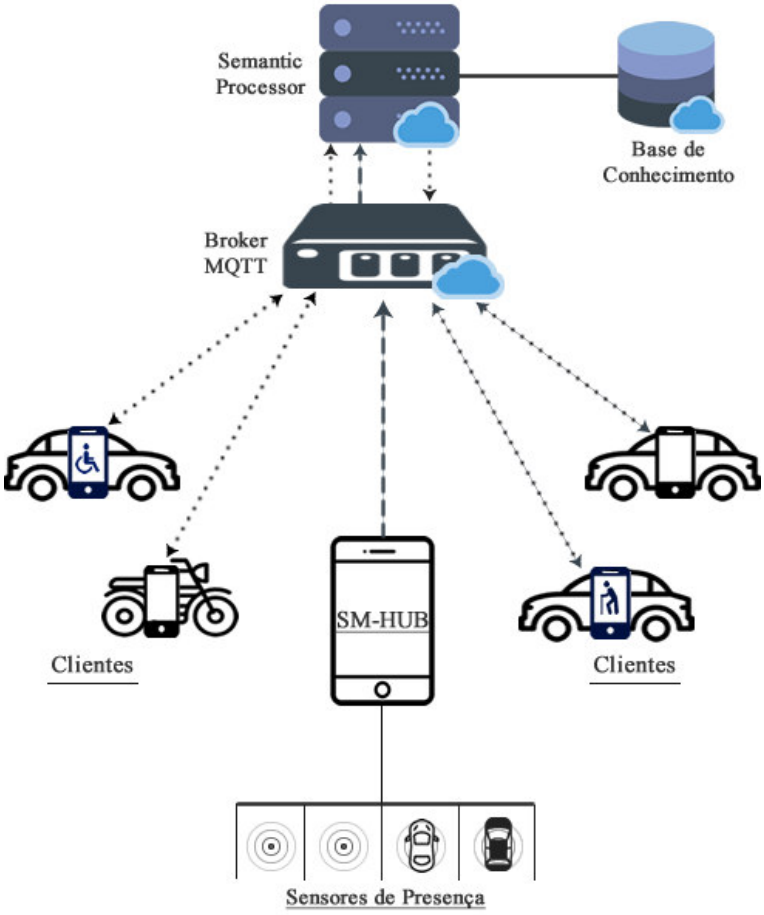

Figura 4.1: Arquitetura do LSDi Smart Parking

ocupação visual das vagas com certa de 12.000 de imagens coletadas em diferentes dias de julho de 2015 a partir de 2 câmeras. Os arquivos seguem esta organização: <CAMERA>/<CLASS>/YYYYMMDD_HHMM_<SLOT_ID>.jpg , onde:

- **<CAMERA>**: identificador da câmera que pode ser A ou B;
- **<CLASS>**: *status* de ocupação da vaga que pode estar livre (*free*) ou ocupada (*busy*);
- **YYYYMMDD_HHMM**: *datetime* de captura da imagem;
- **<SLOT_ID>**: identificador da vaga.

Um exemplo desse registro é: A/busy/20150703_1425_10.jpg. Como a solução faz uso de fluxos de dados semânticos, foi preciso transformar os registro do *dataset* em fluxos semânticos. O Código 4.1 mostra esse fluxo anotado de acordo com a MIoT.

Código 4.1: Fluxo semântico das vagas

```
1  "object":""false"^^http://www.w3.org/2001/XMLSchema/\#boolean",
2  "predicate":"http://www.w3.org/ns/sosa/:hasResult",
3  "subject":"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#10",
4  "timestamp":201507031425
```

Dessa forma, é possível detectar em tempo próximo ao real a ocupação de uma vaga, mas não é possível identificar outras informações importantes, como por exemplo, para qual tipo de veículo aquela vaga é destinada, qual a sua localização geográfica e relativa ou até mesmo identificar se o motorista que está procurando uma lugar, possui permissão para estacionar nela. Para esse fim, é necessário a combinação do fluxo de eventos das ocupações dos espaços com uma base de conhecimento, em que estará armazenado informações referente a vaga e as restrições do motorista.

4.1.1 Ontologia OntoParking

Com intuito de realizar descobertas mais complexas, como as supracitadas, desenvolveu-se uma ontologia para descrever ambientes de estacionamentos denominada OntoParking, seguindo a mesma metodologia de desenvolvimento da MIoT-Ont (*Developmente* 101). Diante disso, foram considerados alguns conceitos necessários para estes ambientes, como: estacionamento, vaga e sua localização geográfica e relativa; conceitos relacionados ao motorista como: perfil do motorista, veículo e sua preferência (*e.g.*, vaga de uso geral, para idoso ou para deficiente físico). Após o levantamento bibliográfico, analisou-se as ontologias para estacionamentos com o objetivo de reutilizar conceitos existentes, como não se encontrou ontologias que descrevessem esse ambiente de forma simples e expressiva para ser aplicado em ambientes da IoT, desenvolveu-se uma ontologia para estacionamento denominada Onto-Parking, adicionando conceitos referentes ao estacionamento e ao motorista. A Figura 4.2 descreve os conceitos da ontologia e as principais relações entre eles.

A classe *Parking* representa o estacionamento (*e.g.*, estacionamento do shopping), o *Space* é uma abstração da vaga e o *Status* é referente ao *status* da vaga, que pode ser livre ou ocupada. A mesma pode ser destinada para o estacionamento de diversos tipos de veículos, como carro, moto e até mesmo bicicleta, e para representá-la, foi definido o conceito *VehicleSpace*. A classe *PreferentialSpace* retrata a preferência

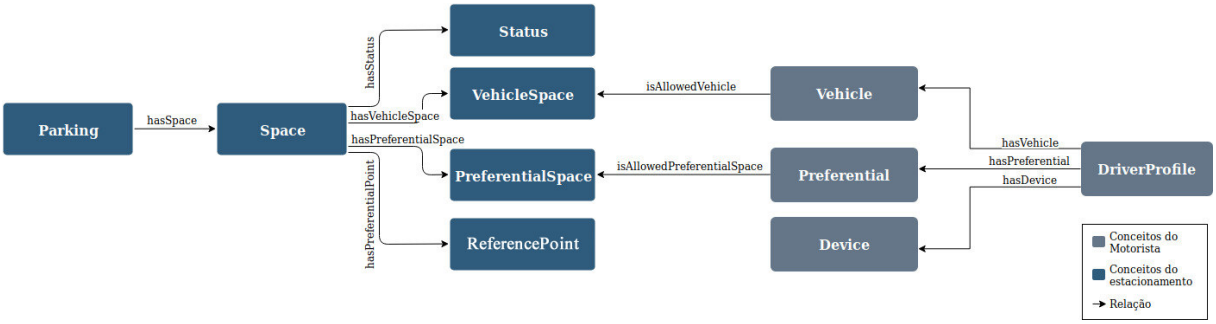

Figura 4.2: Visão geral da OntoParking

da vaga, que pode ser para idosos, gestantes ou deficientes físicos. Já o *ReferencePoint* representa um ponto de referência da vaga, como próximo a entrada, ao restaurante, ao mercado ou a saída, por exemplo. Para esse ambiente de estacionamento, criou-se conceitos referentes ao motorista, já que é ele quem procura as vagas. O *DriverProfile* é uma abstração do perfil do motorista, como nome, idade e telefone. Foi pressuposto que cada motorista possui um *Device* (*e.g., smartphone*), que será usado para sua identificação. O *Vehicle* simboliza o veículo do motorista (*e.g.*, carro ou moto) e o *Prefential* representa a preferência do motorista por vagas. Com isso, somente os idosos poderão estacionar nas vagas destinadas a idosos, por exemplo. Neste sentido, este trabalho de pesquisa contribuiu com uma ontologia leve para descrever estacionamentos.

4.1.2 Aplicativo Lsdi Smart Parking

A aplicação tem duas funcionalidades: (i) descobrir o local do estacionamento de interesse, para que seja traçado uma rota da posição atual do usuário até o estacionamento (Figura 4.3a). Essa consulta é executada em uma bases de dados RDF e não leva em consideração o fluxo de dados. (ii) Descobrir as vagas livres daquele estacionamento, de acordo com o perfil do usuário, como exemplo dessa descoberta: uma vaga livre para o motorista João que possui um carro e é idoso. As vagas livres são mostradas ao usuário através de um marcador verde na posição geográfica de cada vaga, são atualizadas a cada 5 segundos e traçada um nova rota até a vaga mais perto do usuário (Figura 4.3b). Caso esta seja ocupada antes do usuário chegar, é traçado uma nova rota até a próxima vaga livre. Essa função é executada continuamente enquanto o usuário se aproxima do estacionamento.

Figura 4.3: Telas

Para a segunda funcionalidade, é necessário levar em consideração os dados carregados nos fluxos e combinar o mesmo com o conhecimento armazenado de forma contínua. Para essa finalidade, o aplicativo envia uma consulta complexa em C-SPARQL através da API desenvolvida. Esta consulta pode ser vista no Código 4.2

Código 4.2: Consulta em C-SPARQL

```
1  SELECT ?space ?lat ?long
2  FROM STREAM <http://www.semanticprocessor.br/ontology/Parking#>
3  [RANGE 5s STEP 1s]
4  FROM <http://mycsparql.lsdi/smartParking>
5  WHERE {
6  ?space mIot:hasResult "false"^^s:boolean .
7  ?driver pk:hasDevice pk:DeviceID .
8  ?driver pk:hasVehicle ?vehicle .
9  ?driver pk:hasPreferential ?preferential .
10 ?vehicle pk:isAllowedVehicle ?vehicleSpace .
11 ?preferential pk:isAllowedPreferential ?preferentialSpace .
12 ?space pk:hasVehicleSpace ?vehicleSpace .
13 ?space pk:hasPreferentialSpace ?preferentialSpace .
14 ?space pk:Latitude ?lat .
15 ?space pk:Longitude ?long }
```

Na linha 1 são especificados quais os valores retornam caso a consulta seja satisfeita: identificador da vaga e posição geográfica (latitude e longitude). A identificação do fluxo e da janela de tempo são expressas por meio da cláusula *FROM STREAM* e *RANGER* (linha 2 e 3), respectivamente. Na linha 4, observa-se a cláusula *FROM*, que identifica qual base de dados deve ser consultada. Mediante a linha 6, é realizado a seleção de eventos do fluxo referentes às vagas que possuem em seu estado *false*, ou seja, estar livre. Da linha 7 à 15 são estabelecidos os padrões de triplas RDF a serem localizadas na base de dados, a fim de verificar se aquela vaga livre se encaixa no perfil do usuário, que por sua vez é identificado através do *DeviceID* (linha 6). O resultado é um conjunto de tuplas, como pode ser observado na Tabela 4.1.

Campo	Resultado
Vaga	<http://www.semanticprocessor.br/ontology/Parking/#Space18>
Latitude	“-2.557785"^^<http://www.w3.org/2001/XMLSchema/#double>
Longitude	“-44.308088"^^<http://www.w3.org/2001/XMLSchema/#double>

Tabela 4.1: Resultado da consulta

4.2 Conclusão

Este Capítulo apresentou uma avaliação qualitativa da solução proposta através de um cenário de uso. Este demonstra o uso do SM-Hub no qual se descobre, conecta e obtém os dados do sensor de presença de cada vaga constantemente e transforma-os em fluxos semânticos. Com a ontologia MIoT-OnT, foi possível anotar semanticamente os dados referentes à ocupação de vagas, a fim de tratar a heterogeneidade dos *smart objects* de presença e gerar uma maior interoperabilidade entre os *smart objects* e aplicação. A aplicação, através do mecanismo de descoberta semântica desenvolvido, pôde realizar buscas complexas e de alto nível através da infraestrutura em nuvem desenvolvida, que combina dados através dos valores semânticos dos mesmos com uma base de conhecimento.

A solução proposta é genérica o suficiente para permitir modelar dados de sensores utilizados nos mais diversos domínios, bem como permite a utilização de qualquer base de conhecimento baseada em RDF. Desta forma, a partir da

mesma pode-se construir aplicações que realizem descoberta semântica baseada na combinação de fluxos de dados de *smart objects* anotados semanticamente com bases de conhecimento para os mais variados domínios de aplicação, como da saúde, *e.g*, obter os dados dos sensores de monitoramento cardíaco dos pacientes, seja por meio do nome do paciente, da doença ou até mesmo os dados de todos os pacientes que possuam uma doença específica em um determinado ambiente físico; da indústria, *e;g.*, monitorar em tempo próximo do real a temperatura das suas máquinas e equipamentos, bem como o ambiente físico, entre outros.

5 Avaliação Quantitativa

O objetivo geral da avaliação quantitativa é medir o desempenho/eficiência do mecanismo de descoberta proposto, a partir da realização de experimentos controlados e aferição de métricas. Entretanto, avaliar quantitativamente todos os aspectos desse mecanismo, que agrega diversos conceitos e componentes não é uma tarefa trivial. Diante disso, os esforços de avaliação quantitativa voltaram-se para a realização da avaliação de desempenho dos componente desenvolvidos. A seguir, cada conjunto de experimento realizado, com os seus respectivos resultados e discussões serão apresentados em uma subseção a parte.

5.1 Avaliação de Desempenho do RQ, SSQ e da Aplicação

Tanto as consultas como as respostas, passam por diferentes etapas/componentes do mecanismo proposto: (1) aplicação cliente envia a consulta para o *broker* MQTT; (2) o *broker* encaminha para o subscritor SSQ; (4) o SSQ envia para o objeto para o RQ; (5) que por sua vez instancia a consulta no SP e define um *listerner* para receber as resposta e enviar para aplicação consumidora; (6) SP realiza o processamento semântico, podendo combinar o fluxo de dados com uma ou várias bases de conhecimentos e envia a resposta para o RQ.

O objetivo específico deste conjunto de experimento é avaliar a desempenho de cada componente. Não foi considerado o tempo de processamento do componente SP, pois existem fatores que influenciam no tempo do processamento, como o tamanho da base de conhecimento, por exemplo.

5.1.1 Descrição do Experimento

O experimento consistiu na: (i) execução de uma aplicação de teste que faz uso da API desenvolvida; (ii) simulação do SM-Hub; (iii) execução de um *Broker* MQTT

e da infraestrutura em nuvem proposta. Para tanto, utilizou-se um computador DELL que possui as seguintes especificações: 8 GB de RAM, processador intel Core i7 2.7 GHz e sistema operacional Ubuntu 18.04 LTS.

A Figura 5.1 demonstra como é realizada a anotação do *timestamp* de cada componente.

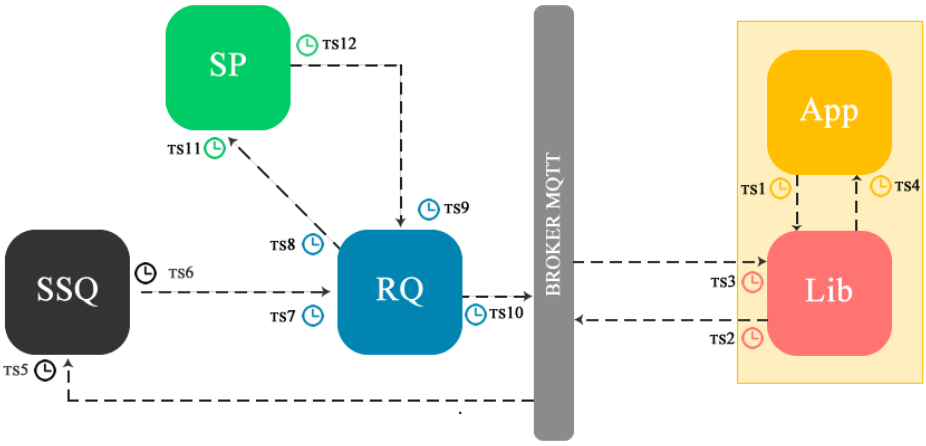

Figura 5.1: Anotação dO *timestamp*

O tempo de processamento do SSQ é calculado em milissegundos com base na diferença entre o *timestamp* de chegada no componente e o *timestamp* de saída do componente, conforme a equação a seguir:

$$TempoTotalSSQ = TS6 - TS5 \tag{5.1}$$

Ressalta-se que o tempo de processamento do componente RQ e da aplicação não são iguais ao SSQ, pois eles enviam e recebem dados duas vezes. Para isso, são utilizados as seguintes equações:

$$TempoTotalRQ = (TS8 - TS7) + (TS10 - TS9) \tag{5.2}$$

$$TempoTotalAplicao = (TS2 - TS1) + (TS4 - TS3) \tag{5.3}$$

Para esse tipo de análise, foi realizado baterias de testes com trinta execuções. Após isso, calculou-se o valor máximo, mínimo, desvio padrão, a margem de erro do intervalo de confiança de 95%, referente a cada um deles. Definiu-se que o desempenho representativo de cada componente seria dado através da média dos resultados obtidos.

5.1.2 Resultados e Discussões

A Tabela 5.1 apresenta os resultados obtidos nos experimentos em milissegundos. Ao invés de mostrar os valores do intervalo de confiança de cada componente, preferiu-se apresentar a margem de erro já que é a distância da estatística estimada para cada valor de intervalo de confiança.

Componente	Média (ms)	Máximo (ms)	Mínimo (ms)	Desvio Padrão (ms)	Margem de Erro (ms)
SSQ	12,76667	23	8	4,417264	+/- 1,64943
RQ	12,23333	25	9	3,158938	+/- 1,17957
API	147,8333	212	107	26,349413	+/- 9,89307

Tabela 5.1: Avaliação de desempenho

Esses resultados permitem mostrar que o desempenho de cada componente do mecanismo de descoberta é bastante baixo. Como esperado, a aplicação por conter mais etapas de processamento do que os outros componentes, apresentou-se um valor mais alto de desempenho. Os baixos desvios padrões dos experimentos e as margens de erros próximas a zero indicam que a tendência é que esses desempenhos se mantenham estáveis.

5.2 Avaliação de Desempenho do SP

O componente SP citado na na seção 4, é responsável por processar consultas e fluxos de dados semânticos, podendo levar em consideração bases de conhecimento. O tempo gasto para processar e realizar as combinações precisa ser ágil. Do contrário, caso ocorram mudança dos dados dos *smart objects*, estas serão percebidas muito tardiamente pela aplicação consumidora. Consequentemente, isso pode impedir que ela reaja a tais modificações em tempo hábil.

Diante disso, o objetivo deste conjunto de experimentos é avaliar a eficiência do processador semântico, além de identificar as influências dos fatores no tempo de resposta do processamento.

5.2.1 Descrição do Experimento

Para este experimento adotou-se a técnica denominada Planejamento de Experimentos [46]. Esta serve para planejar experimentos, ou seja, definir quais dados, quantidade e condições devem ser coletados durante um determinado experimento, com intuito de obter a maior precisão estatística possível na resposta e com um menor custo.

O tempo de processamento do SP é calculado em milissegundos com base na diferença entre o *timestamp* de chegada no componente e o *timestamp* de saída do componente, conforme a equação a seguir:

$$TempoTotalSP = TS2 - TS1 \tag{5.4}$$

Há uma série de fatores que impactam no tempo do processamento e para análise de desempenho utilizou-se três fatores analisados em dois níveis:

- Tamanho da base de conhecimento:
  - Mil instâncias (MI).
  - Cinco mil instâncias (CI).
- Volume do fluxos de eventos:
  - Mil eventos por segundo (ME).
  - Cinco mil eventos por segundo (CE).
- Complexidade da consulta:
  - Uma combinação com a base de conhecimento (1C).
  - Sete combinações com a base de conhecimento (7C).

Foram utilizados os mesmos recursos de hardware e softwares do teste anterior. Novamente, efetivou-se baterias de testes com trinta execuções para cada experimento. Calculou-se a média, mediana, desvio padrão, margem de erro do intervalo de confiança de 95%, primeiro e terceiro quartil, valores discrepantes de cada experimento e os resultados dos fatores de impacto. Definiu-se que o desempenho representativo seria dado através da média dos resultados obtidos.

Resultados e Discussões

Para representação gráfica dos resultados, optou-se pelo Boxplot, o qual permite avaliar rapidamente a dispersão dos dados, destacando também os valores discrepantes presentes [48]. Como mostra a Figura 5.2, o Boxplot é formado pelo primeiro e terceiro quartil, pela média, mediana e *outiliers*.

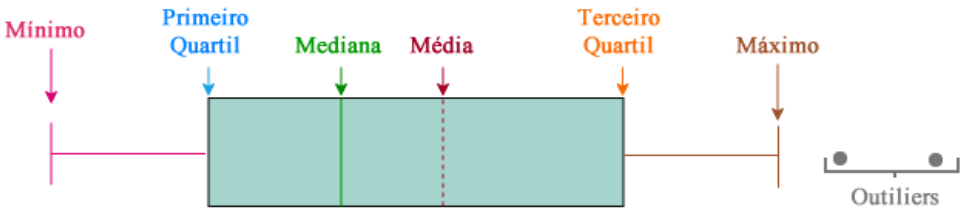

Figura 5.2: Boxsplot

No total foram realizados 8 experimentos, combinando de todas as formas todos os fatores. O resultado referentes a média, mediana, primeiro e terceiro quartil e os valores discrepantes são representados através do Bloxplot, como se verifica na Figura 5.3. Já os valores de desvios padrões e margens de erros, são mostrados na Tabela 5.2. No calculo desses valores, os resultados discrepantes foram descartados, já que têm efeito sobre as médias, desvios padrões e consequentemente nos intervalos de confianças [42].

Experimento	Média (ms)	Desvio Padrão (ms)	Margem de Erro (ms)
MI+ME+1C	11,68	3,10	+/- 1,18
MI+ME+7C	17,34	5,30	+/- 2,01
MI+CE+1C	12,75	6,28	+/- 2,39
MI+CE+7C	17,44	9,05	+/- 3,44
CI+ME+1C	42,24	6,62	+/- 2,51
CI+ME+7C	45	12,29	+/- 4,67
CI+CE+1C	44,62	5,98	+/- 2,27
CI+CE+7C	48,34	9,76	+/- 3,71

Tabela 5.2: Avaliação de desempenho do SP

É possível observar na Figura 5.3 que em todos os experimentos há valores discrepantes. Esses valores são referentes à primeira resposta e atribui-se essa

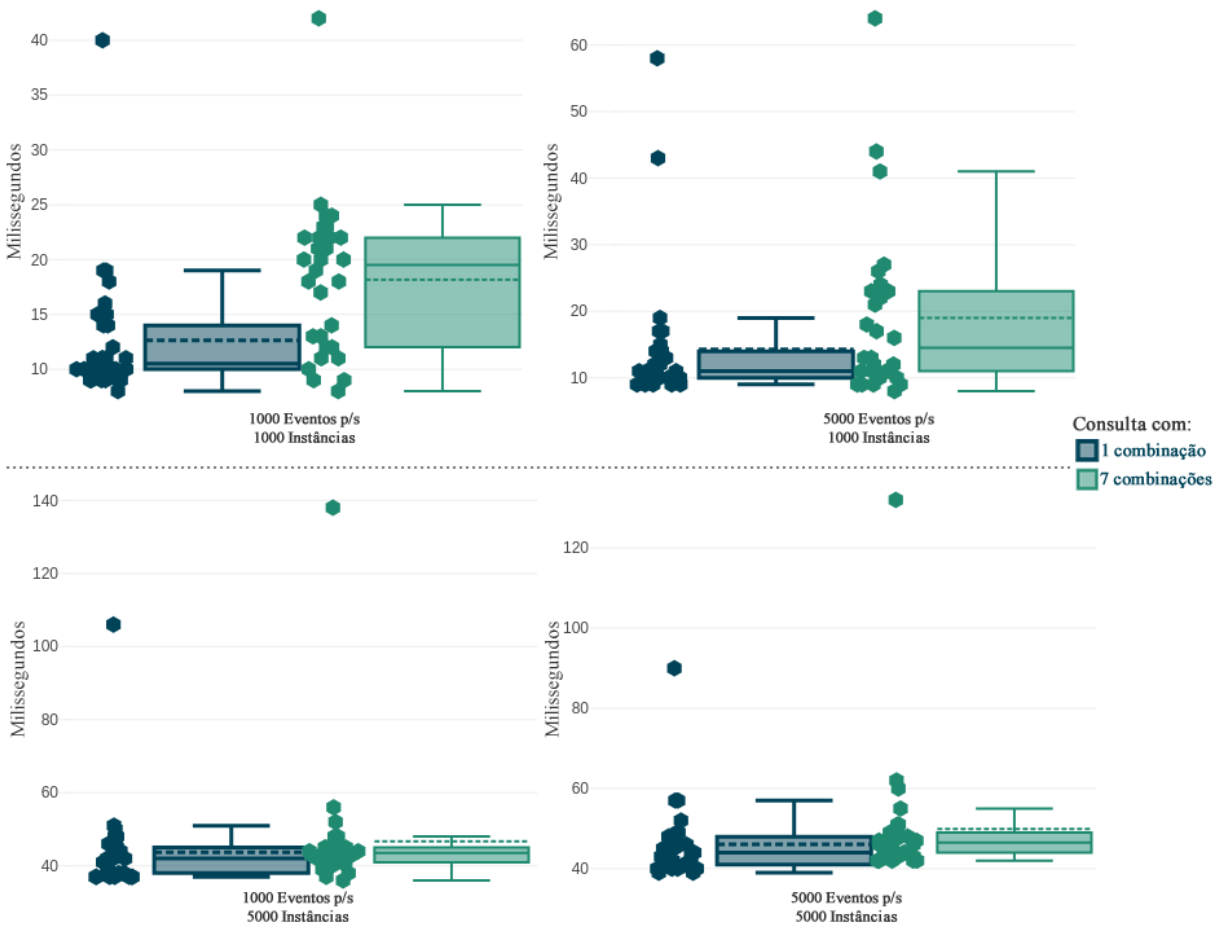

Figura 5.3: Resultado dos experimentos

diferença significativa em relação aos outros resultados aos seguintes fatos: (i) tempo de instanciação da consulta na *engine* C-SPARQL; (ii) janela deslizante utilizada na consulta.

Em relação aos experimentos com a base de conhecimento com mil instâncias, pode-se observar que o tempo médio de processamento com uma combinação na base de conhecimento e com mil eventos p/s é de 11,68 ms e de 12,75 ms mudando o volume do fluxo para cinco mil eventos p/s. Aumentando a complexidade da consulta (sete combinações), os valores médios foram 17,34 ms com mil eventos p/s e 17,44 ms com cinco mil eventos p/s. Como se vê, apesar de aumentar consideravelmente o volume do fluxo e os números de combinações com a base de conhecimento, não houver aumento significativo no tempo do processamento, que por sua vez, manteve-se baixo.

Em relação aos quatro últimos experimentos com a base de conhecimento com cinco mil instâncias, pode-se observar que o tempo médio do processamento

aumentou em média o quádruplo dos primeiros. Combinando essa base de conhecimento com uma combinação na base de conhecimento e com mil eventos p/s, obteve-se a média de 42,24 ms e de 44,62 ms com aumentando o fluxos de eventos para cinco mil. Já com uma consulta com sete combinações e mil eventos por segundo, a média foi 45 ms e de 48,34 ms no último com cinco mil eventos por segundo. Apesar desse aumento de tempo em relação aos primeiros experimentos, o valor se manteve baixo.

Com base na quantidade de experimento (QE), na variância (V), nos níveis dos fatores (NF) e na influência total dos fatores (IF), calculou-se o fator de influência de cada fator no tempo do processamento através da seguinte formula: (QE*(V^NF)/IF). Pode-se constatar que o tamanho da base influencia cerca de 97,53% no tempo de processamento, já a complexidade da consulta influencia 1,89% e o volume dos fluxos de eventos influencia apenas 0,32%. A influência da interação dos fatores ficou em média 0,06%. Com isso, pode-se verificar que quanto maior a base de conhecimento, maior será o tempo do processamento e que o volume dos fluxos de eventos não têm tanta influência nesse tempo, o que é essencial para ambientes da IoT/IoMT.

Por fim, os baixos desvios padrões dos experimentos e as baixas margens de erros apresentadas permitem afirmar que os resultados tendem a se manter estáveis com o uso de aplicações IoT/IoMT que executem descobertas em condições semelhantes às simuladas nesses experimentos. Constata-se também que o SP é bastante eficiente, pois conseguiu processar os dados em tempo próximo ao real, mesmo com grandes bases de dados e grandes quantidades de fluxos de eventos.

5.3 Conclusão

Este capítulo apresentou uma avaliação quantitativa para medir o desempenho/eficiência do mecanismo proposto. Para isso, executou-se avaliação de desempenho dos componentes desenvolvidos.

Os resultados obtidos mostraram que o desempenho de cada componentes é muito baixo e estável. No componente responsável pelo processamento semântico, foram realizados oito experimentos para analisar a eficiência do componente SP e

identificar as influências dos fatores no tempo de processamento. Os resultados mostraram que componente consegue processar em tempo próximo ao real, mesmo em condições diversas. Pode-se constatar que o fator que mais influência no tempo de processamento é o tamanho da base de conhecimento.

De modo geral, os resultados e as discussões apresentadas nessa seção mostram que o mecanismo proposto apresenta um desempenho muito satisfatório em relação às métricas estabelecidas, dentro das condições avaliadas. Considerando-se que o desempenho é um fator muito importante na escolha do mecanismo de descoberta a ser utilizado para desenvolver e executar aplicações de Iot/IoMT, a avaliação desempenho realizada demonstra o potencial de utilização do mecanismo desenvolvido em cenários em que as aplicações exigem baixo tempo de resposta. Diante do exposto, cumpre-se não apenas o objetivo de desenvolver um mecanismo de descoberta dinâmica de serviços em IoT/IoMT eficiente, mas também o objetivo de avaliá-la quantitativamente.

Cabe ressaltar que, de modo geral, os trabalhos relacionados a descoberta semântica de serviços em IoT encontrados na literatura (ver Seção 6) apresentam pouca ou nenhuma avaliação quantitativa. Destaca-se também, que a identificação dos fatores de influência no tempo do processamento foi uma contribuição bastante relevante, pois não foi encontrado nenhum trabalho na literatura referente a *engine* C-SPARQL que realizasse esse tipo de experimento. Um novo conjunto de avaliações quantitativas, incluindo por exemplo, experimentos de escalabilidade, será o objetivo de trabalhos futuros.

6 Trabalhos Relacionados

Para caracterizar o estado da arte da pesquisa e os principais trabalhos relacionados à descoberta semântica de serviços em ambientes da IoT foi realizada uma revisão sistemática da literatura. Este capítulo apresenta inicialmente a metodologia utilizada bem como os critérios para a comparação dos trabalhos. Em seguida é feito uma análise comparativa entre os mecanismos de descoberta. Ao final desse capítulo apresenta-se as discussões.

6.1 Revisão Sistemática da Literatura

A Revisão Sistemática da Literatura (RSL) é classificada como um método de investigação científica realizada de maneira formal, abrangente, detalhista e não tendenciosa, com o objetivo de reunir, avaliar criteriosamente e realizar uma síntese dos resultados dos estudos primários [11]. Os métodos sistemáticos para selecionar, avaliar, coletar e analisar as pesquisas relevantes devem ser expostos de modo que outros pesquisadores possam repetir o procedimento.

Os objetivos dessa RSL são: (i) identificar limitações do serviço de descoberta semântica descritas nos trabalhos; (ii) observar os estudos experimentais que têm sido realizados como forma de avaliar os mecanismo de descoberta semântica de serviço.

6.1.1 Metodologia

Para a realização dessa revisão sistemática, foi adotada como base as Diretrizes para Realização de Revisões de Literatura Sistemática em Engenharia de Software [11] que propõem a execução dos seguintes passos:

- Definir as questões de pesquisa;
- Definir os parâmetros de buscas nas bases científicas;

- Definir os critérios de inclusão e exclusão no processo de seleção dos trabalhos;
- Selecionar os trabalhos;
- Extrair os trabalhos;
- Extrair os resultados.

As questões de pesquisas definidas de acordo com o tema e os objetivos já mencionados, foram:

- Quais são as técnicas e tecnologias utilizadas para realizar descoberta semântica de serviços em ambientes da IoT?
- Como foram conduzidos os estudos experimentais que validam esses trabalhos?

As bases selecionadas para a busca dos artigos foram: IEEE, ACM, Scopus e Science Direct. Elas foram escolhidas porque estão entre as cinco maiores bases na área de ciência da computação, segundo a classificação da Capes[1]. Nelas, os parâmetros de busca foram aplicados para o título, resumo e palavras-chave. Foi aplicado em cada uma o filtro de ano de 2013 à 2018, a fim de se obter os trabalhos mais recentes. A Tabela 6.1 mostra os parâmetros de busca utilizado nas suas respectivas bases.

Base	**Parâmetros**
ACM e IEEE	("*Service Discovery*"AND (*ontology OR ontologies* OR *semantic*) AND (iot OR "*Internet of Things*"))
Science Direct	tak("*Service Discovery*"AND (*ontology* OR *ontologies* OR *semantic*) AND (iot OR "Internet of Things"))
Scopus	TITLE-ABS-KEY ("*Service Discovery*"AND (*ontology* OR *ontologies* OR *semantic*) AND (iot OR "*Internet of Things*"))

Tabela 6.1: Parâmetros de busca

Os termos utilizados para a busca em título, palavras-chaves e resumo foram: *Service Discovery*, para encontrar artigos relacionados a descoberta de serviços;

[1]Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) é uma fundação vinculada ao Ministério da Educação (MEC) do Brasil que atua na expansão e consolidação da pós-graduação stricto sensu (mestrado e doutorado) em todos os estados do Brasil

ontology, *ontologies* e *semantic* foram os termos utilizados para buscar por modelos e modelagens semânticas, que podem ser aplicados na descoberta de serviço; *Internet of Things* e sua sigla IoT, referem-se ao termo principal na busca de artigos.

A realização da busca foi executada em 03/08/2018 e teve como resultado 87 artigos. A base que retornou uma maior quantidade de trabalhos foi a Scopus com 49, pois ela possui o maior banco de dados de resumos e citações da literatura com revisão por pares[2]. Já a ACM retornou a menor quantidade, somente 3 trabalhos. Com a IEEE, conseguiu-se 27 trabalhos e apenas 8 na Science Direct como mostra a Figura 6.1.

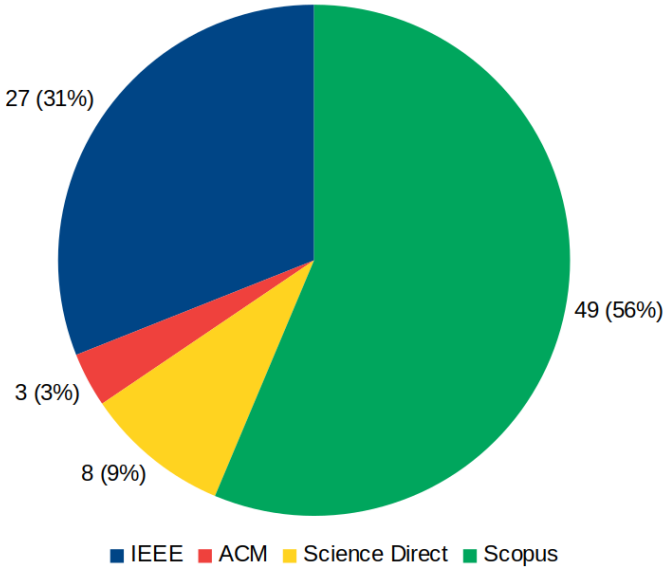

Figura 6.1: Quantidade de artigos por base científica

Na fase de seleção, em que se analisou o título e o resumo, foram identificados 22 artigos duplicados e 39 que estavam fora do escopo da pesquisa. Todos foram descartados da lista dos selecionados. Os principais critérios adotados para rejeição dos artigos foram: trabalhos que não envolviam IoT, como por exemplo, citavam esse termo apenas no resumo; que não utilizavam semântica; que apresentavam apenas ontologia ou proposta de vocabulários semânticos para IoT; artigos curtos (4 páginas ou menos), artigos de revisões e descrição de *workshops* e/ou conferências; por último, aqueles que não estavam disponíveis integralmente nas bases de dados pesquisadas. O resultado da fase de seleção pode ser visto na Figura 6.2.

[2]https://www.elsevier.com/pt-br/solutions/scopus

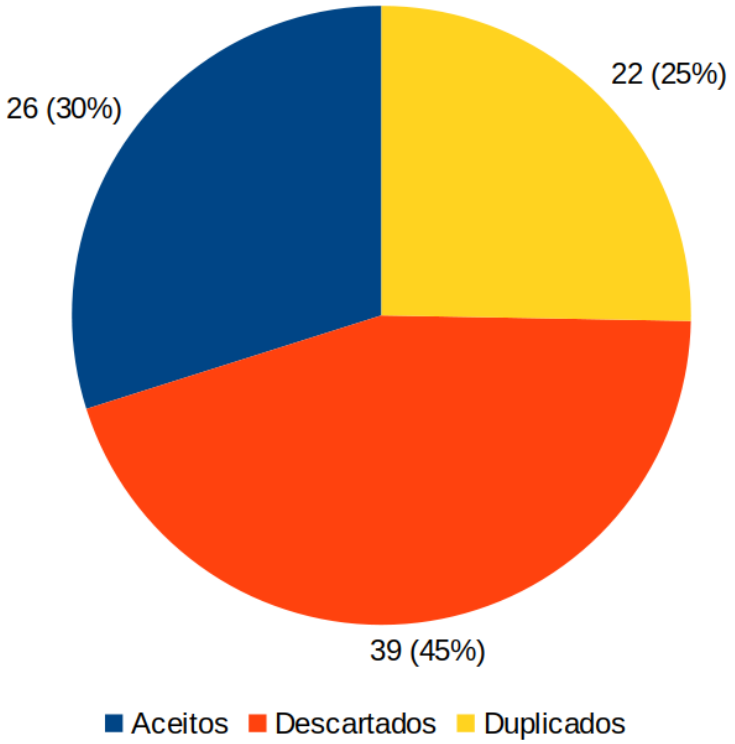

Figura 6.2: Resultado da fase de seleção

Logo após a seleção, iniciou-se a fase de extração de forma completa dos 26 trabalhos selecionados. Nessa etapa, analisou-se os trabalhos por completos, descartando 17 artigos que fugiam do escopo da pesquisa. Essa análise foi mais criteriosa do que na fase anterior e os principais motivos de rejeição foram: (i) trabalhos que não tratam de IoT ou que não usaram semântica para descoberta de serviço; (ii) que não apresentam estudos experimentais para validar o mecanismo proposto, ou não trazem detalhes suficientes para analisar os métodos ou formas de aplicação. (iii) que apresentam apenas algoritmos como solução para descoberta de serviços. Foi realizada uma leitura mais detalhada dos artigos aceitos, anotando suas características, tecnologias, métodos e resultados. No fim, foram aceitos 9 trabalhos dos 26 analisados na fase de extração.

Para execução desses passos, foram utilizados os seguintes softwares:

- Start[3]: utilizado nas fases de seleção e extração;

[3]Start: http://lapes.dc.ufscar.br/tools/start_tool

- Mendeley[4]: utilizado para o gerenciamento dos trabalhos selecionados, funcionando como um repositório compartilhável e sendo utilizado para anotação;
- LibreOffice Cal: usado para geração de gráficos e análise dos dados.

Disponibilizou-se no Apêndices A uma síntese dos trabalhos selecionados.

6.2 Resultado

Os critérios adotados para a comparação, que em parte são baseados em trabalhos que propõe critérios [25, 27, 44], são:

- **Consultas com múltiplos atributos**: Ter capacidade de encontrar recursos através de consultas baseadas no casamento de um ou mais atributos (e.g., buscar recurso que possuam um certo tipo e uma certa localização).
- **Consultas em intervalos:** Ter a capacidade de encontrar recursos a partir da especificação de limites inferiores e superiores para um ou mais atributos.
- **Descrição semântica de recursos**: Usar ontologias para modelar semanticamente recursos e os serviços exposto pelo mesmo;
- **Suporte a busca por localização:** Ter a capacidade de modelar e buscar informações de localização de recursos;
- **Suporte à mobilidade:** Lidar com a mobilidade dos *smart objects* e dos *gateways*;
- **Segurança:** Fornecer segurança na distribuição dos dados, tais como autenticação,autorização e criptografia
- **Processamento de fluxos semânticos:** Suportar a coleta, processamento e análise de fluxos de dados semânticos

[4]Mendeley: https://www.mendeley.com

6.2.1 Consultas com múltiplos atributos e consultas em intervalos

O uso de linguagens de consulta possibilita a criação de *queries* de busca contendo múltiplos atributos e/ou intervalo de valores. O SPARQL é uma delas, pois possibilita a realizar consultas em dados modelados semanticamente (*e.g* OWL, RDF) e é adotado em diversos mecanismo de descoberta semântico, como em [13, 27, 33, 52]. Nos trabalhos proposto por [35, 36] que são baseados em SkipNet e em [49] não é explicito a linguagem utilizada para a consulta, porém pode ser realizado consultas em intervalos ou até mesmo contendo vários atributos. Em [49], o cliente constrói uma consulta em SPARQL através de componentes gráficos, limitando-se a expressividade na consulta, porém é possível construir consulta com diversos atributos. Diferente dos demais, a proposta exposta em [43] utiliza requisições REST para obtenção dos dados.

6.2.2 Descrição semântica de recursos

Para modelar semanticamente os recursos e os serviços exposto pelo mesmo, frequentemente utiliza-se ontologia, que nada mais é do que um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre eles. Elas também permitem a troca de informações entre aplicativos e entre diferentes níveis de abstração, que é importante para a IoT, pois fornece interoperabilidade entre os provedores, consumidores e dispositivos e facilita a integração, descoberta de recursos e extração de conhecimento.

Alguns trabalhos [12, 13, 36] propõem seu próprio modelo ontológico; há ainda os sistemas que reutilizam e integram ontologias existentes [27, 33, 35, 52], como no caso do trabalho do Khodadadi e Sinnott [33] que se baseiam em uma ontologia derivada do Modelo de Serviço Mínimo (MSM). O Wang *et al.* [52] reutiliza uma ontologia desenvolvida no projeto IoT.est[5]. Já o modelo de informação do Gomes *et al.* [27] aproveita a ontologia SAN, uma extensão da ontologia SSN(*Semantic Sensor Network Ontology*) do W3C. Além disso, há uma parte da ontologia de SOUPA para descrever localizações espaciais de entidades e criando conceitos de QoC a partir do meta-modelo QoCIM. De fato, o reuso de ontologias padronizadas na literatura facilita

[5]IoT.est: http://www.ict-iotest.eu/

a evolução, extensão e interação com outras ontologias, além de facilitar a adoção de uma ontologia.

De acordo com os autores Paganelli e Parlanti [44], esse critério é atendido quando o trabalho utiliza a ontologia para modelar as informações, porém a ontologia é apenas uma forma para modelar os dados semanticamente. Os trabalhos apresentados em [43, 49] não utilizam ontologias para fornecer a descrição semântica de recursos. O mecanismo proposto por Niermirepo *et al*. [43] transformam os dispositivos, serviços e os dados em objetos *JavaScript Object Notation* (JSON) de acordo com a semântica definida pelos autores. A proposta de Quevedo *et al*. [49] utiliza uma tabela semântica para armazenar os serviços disponíveis. Como se vê, é uma forma diferente de modelar os dados semanticamente, porém não apresentam um modelo de conceitos, relacionamentos e hierarquia entre eles, coomo também não é possível compartilhar o domínio de conhecimento.

6.2.3 Suporte à busca por localização

A localização é um critério altamente seletivo para a descoberta de informações sobre o mundo físico [17]. Algumas propostas [27,35,52] fornecem suporte à descoberta de serviços através da localização modelada em uma ontologia. Essa modelagem leva em consideração tanto as coordenadas geográficas como a localização relativa, porém cada trabalho adota uma nomenclatura diferente para esse conceito, por exemplo, em Gomes et al. [27] é denominada *"PhysicalStructures"*, já o Wang et al. [52] adota o conceito "*geoLocation*". Além do relacionamento dos *smart objects* com a localização, Li et al. [35] propõe a relação de "co-localização"(*"co-location"*) entre os *smart objects*. Através do armazenamento dos dados dos *smart objects* junto com a localização em uma base da dados é possível rastrear e descobrir *smart objects* por meio da localização, *e.g.*, descobrir os *smart objects* presentes em uma sala.

6.2.4 Suporte à mobilidade

Em ambientes dinâmicos da IoT/IoMT, nos quais a disponibilidade dos provedores de serviços varia de acordo com a movimentação dos *smart objects* e/ou do próprio *gateway*, faz com que seja necessário adoção de um mecanismo de descoberta

que dê suporte à mobilidade. Para esse fim, é vantajoso utilizar consultas contínuas além de *gateways* móveis, pois elas ficam instanciadas e executadas continuamente sobre os eventos de descoberta e desconexão de *smart objects*. Em outras palavras, esse tipo de consulta permite a análise contínua de dados de conexão e desconexão de *smart objects* devido à mobilidade através de consultas armazenadas.

Os trabalhos [13, 27, 33, 52] não levam em consideração aspectos de mobilidade, lidando apenas com *gateways* estáticos e com consultas instantâneas através do uso da linguagem de consulta SPARQL. Nesses mecanismos, a consulta é executada em dados armazenados e recebem imediatamente a resposta contendo as informações que atendem aos critérios especificados. Com isso, não é possível receber as próximas atualizações dos dados.

O mecanismo proposto por Gomes et al. [27] atende esse quesito parcialmente. Apesar de não utilizar *gateway* móvel, a solução proposta oferece suporte a mobilidade dos *smart objects*. Ao invés de utilizar tecnologias que suportem consultas contínuas, ele utiliza as consultas instantâneas SPARQL, executando a mesma de tempos em tempos em uma base de dados RDF. Com isso, não é possível detectar em tempo real a mudança de estado dos *smart objects*, por exemplo.

6.2.5 Segurança

A comparação mostra que apenas um dos dez mecanismos de descoberta semântica lida com aspectos de segurança. A solução de Niemirepo et al. [43] possui mecanismo de segurança voltado ao controle de acesso aos dados, em que o componente *RequestBroker* verifica o direito de acesso do cliente, através dos atributos do parâmetro de um objeto JSON ou nos cabeçalhos de solicitações HTTPS. Se a verificação falhar, uma mensagem de erro será retornada, caso contrário, a chamada será mediada para o módulo responsável de consulta no servidor. O protocolo adotado no trabalho também permite que os dados sejam transmitidos por meio de uma conexão criptografada utilizando o protocolo SSL/TLS.

6.2.6 Processamento de fluxos semânticos

Diversas aplicações IoT/IoMT exigem que seus sistemas computacionais reajam rapidamente às mudanças do ambiente [25]. A análise do fluxo de eventos em tempo próximo ao real em ambientes da IoT/IoMT pode ser um fator crítico, visto a grande quantidade de dados coletados por um grande número de *smart objects*. Os resultados do processamento podem ser outros eventos, isto é, derivados de eventos de entrada (por exemplo, dados do sensor). Esses fluxos podem ser anotados semanticamente, gerando maior interoperabilidade e elevando o nível de abstração do aplicativo.

Várias situações requerem um processamento de fluxo semântico, *e.g.*, identificar a mudança nos valores das ações das empresas no mercado financeiro; identificar a alteração da temperatura em um local, pessoa ou ambiente específico, identificado através de sensores [4]; e a criação de filtros na qualidade dos dados doa *smart objects* (por exemplo, acurácia, precisão). No entanto, nenhum trabalho foi encontrado na literatura que usa o fluxo de dados semânticos para a descoberta de serviços em IoT/IoMT.

6.2.7 Questões de pesquisa

Após a comparação dos trabalhos, realizou-se um mapeamento das principais técnicas e tecnologias utilizadas, como ontologias e tecnologias semânticas, a fim de responder a primeira questão desta RSL. A Tabela 6.2 mostra esse mapeamento.

#	Quantidade
Novas ontologias	3
Reuso de ontologias	4
SPARQL	4
OWL	3
JSON	5

Tabela 6.2: Técnicas e tecnologias utilizadas nos mecanismos de descoberta semântica

Como se observa, a somatória da quantidade de técnica e tecnologia utilizada é superior ao número de trabalhos selecionados devido alguns deles utilizarem mais de uma técnica/tecnologia. Alguns trabalhos [12, 13, 36] propõem seu próprio modelo ontológico; há outros [27, 33, 35, 52] que reutilizam e integram ontologias existentes, como a SAN e SSN. Pode-se verificar que não há nenhuma ontologia padrão para a descrição de ambientes IoT, já que cada trabalho adota uma ontologia diferente dos demais.

Como esperado, por ser a linguagem padrão para construção de ontologias segundo a W3C, a OWL foi a mais utilizada. A principal tecnologia para acesso aos dados semânticos usada nos trabalhos foi o SPARQL. Para a troca de dados, como por exemplo, envio dos dados dos *smart objects* para uma infraestrutura em nuvem, foi utilizado pela maioria o JSON. Acredita-se que essa escolha se deu pelo fato dele ser um formato compacto, de padrão aberto independente, simples, leve e rápido, tornando-se ideal para ambientes da IoT. Alguns trabalhos não informaram as técnicas ou tecnologias utilizadas.

Referente a questão de pesquisa sobre o estudos experimentais que validam esses trabalhos, alguns apresentaram somente casos de uso [12, 33, 43] que serviram para provar que o mecanismo proposto pode ser aplicado ao desenvolvimento de aplicações IoT. Outros avaliaram apenas o desempenho da sua solução [13, 35, 36, 49] que serviu para verificar a eficiência do mecanismo de descoberta. Somente um trabalho analisou das duas formas [27].

Na avaliação qualitativa, os cenários dos casos de uso empregados pelos trabalhos são diversos, como: economia de energia e monitoramento da temperatura em um *data center*, monitoramento da poluição do ar e casas inteligentes. Na avaliação quantitativa, os desempenhos dos mecanismos são avaliados em termo de recuperação, precisão, tempo médio de respostas, tempo de processamento, variando o tamanho da rede, da base de dados e até mesmo os números de entidades na consulta.

Nenhum artigo avaliou sua solução junto aos clientes. Com isso, não é possível obter o grau de satisfação dos usuários do mecanismo de descoberta em relação a usabilidade, eficiência do serviço, abstrações e interfaces de programação, por exemplo.

6.2.8 Análise Comparativa

Uma comparação entre os trabalhos apresentados com a solução proposta nesta dissertação está exposta na Tabela 6.3. Nela, o símbolo ✓ denota que o requisito é completamente atendido, já o ◯ aponta que o requisito parcialmente atendido e o X mostra que o requisito não é atendido.

Trabalho	Consultas com múltiplos atributos	Consultas em intervalos	Descrição semântica de recursos	Suporte a busca por localização	Suporte à mobilidade	Segurança	Processamento de fluxos semânticos
Chun et al. [13]	✓	✓	✓	X	X	X	X
Li et al. [36]	✓	✓	✓	X	X	X	X
Khodadadi and Sinnott [33]	✓	✓	✓	X	X	X	X
Niemirepo et al. [43]	✓	✓	◯	X	X	✓	X
Chindenga et al. [12]	X	X	✓	X	X	X	X
Li et al. [35]	✓	✓	✓	✓	X	X	X
Wang et al. [52]	✓	✓	✓	✓	X	X	X
Quevedo et al. [49]	✓	✓	◯	X	X	X	X
Gomes et al. [27]	✓	✓	✓	✓	◯	X	X
Solução Proposta	✓	✓	✓	✓	✓	X	✓

Tabela 6.3: Comparação dos trabalhos relacionados

Apesar da existência de vários trabalhos que abordam o desenvolvimento de mecanismos de descoberta semântica dos recursos em IoT, nota-se que alguns critérios como suporte à buscar por localização, suporte à mobilidade, segurança e processamento de fluxos semânticos não são bem tratados. Em relação a busca por localização, a solução proposta, assim como nos trabalhos de Li et.al [35] e Gomes et.al [27], permite que as aplicações utilizem tanto informações geográficas quanto simbólicas para descobrir serviços em IoT. Já em relação ao suporte à mobilidade, o mecanismo proposto nesta dissertação se destaca dos demais, uma vez que ele fornece um *middleware* móvel capaz de descobrir, conectar e se comunicar de forma oportunística com *smart objects* próximos.

Embora todos apresentem uma modelagem semântica dos recursos, observa-se que nenhum deles fazem uso de fluxos de dados semânticos. Em contrapartida, solução proposta é única que utiliza esses fluxos semânticos para

descoberta de serviços, fazendo com que seja possível descobrir e monitorar *smart objects* em tempo próximo ao real, fornecer uma maior interoperabilidade entre diversas tecnologias, além de fazer o uso combinado dos fluxos com bases de conhecimento para enriquecer a descoberta.

Diante do exposto, pode-se considerar que as contribuições do mecanismo proposto nesta dissertação, dentre os que foram comparados, são: suporte a mobilidade dos *smart objects* e dos *gateways*; suporte a coleta, processamento e análise de fluxos semânticos de dados dos *smart objects*; e avaliação da solução bem mais abrangente que o da maioria das abordagens relacionadas.

6.3 Conclusão

Este capítulo apresentou uma RSL sobre a descoberta semântica de serviços em ambientes da IoT. Essa é uma das contribuições desta pesquisa, uma vez que, não há revisões na literatura que analisem os diversos aspectos de um mecanismo de descoberta semântica em IoT. Foi promovida então uma análise comparativa entre os mecanismos dos trabalhos relacionados e a solução proposta nesta dissertação, levando em consideração diversos critérios relevantes como consultas com múltiplos atributos e em intervalos, descrição semântica dos recursos, suporte à busca de *smart objects* através de sua localização, suporte à mobilidade e ao processamento de fluxos semânticos, além da segurança na distribuição dos dados.

Além de comparar os trabalhos, realizou-se um levantamento das principais técnicas e tecnologias utilizadas nesses mecanismos. Pode se observar que a maioria reutilizam ontologias já existentes na literatura e que a linguagem mais aplicada para defini-las foi a OWL. O SPARQL foi a linguagem de consulta mais usada para acessar os dados semânticos e para troca de dados, a maioria utilizou o JSON.

Como se observa, os trabalhos relacionados mostram apenas provas de conceitos ou avaliação de desempenho, deixando de avaliar as duas formas em conjunto, ou até mesmo deixando de averiguar o grau de satisfação dos usuários do mecanismo de descoberta em relação às abstrações e serviços, por exemplo.

Com a realização da analise, pode-se constatar que alguns critérios relevantes para descoberta de serviços em IoT impostos por diferentes aplicações não

são bem tratados, como: busca por localização dos recursos; suporte à mobilidade de *smart objects* e/ou *gateways*; e segurança na distribuição dos dados. Apesar de todos trabalhos utilizarem semântica nas suas soluções, apenas a solução proposta nesta dissertação faz uso de fluxos de dados semânticos, além de combiná-los com bases de conhecimentos.

7 Conclusões e Trabalhos Futuros

Este trabalho objetivou o desenvolvimento de um mecanismo de descoberta instantânea e contínua de *smart objects* e seus serviços em IoT/IoMT através do uso combinado de Processamento de Fluxos de Semânticos com técnicas de representação de conhecimento. No entanto, muitos desafios que circundam a descoberta de serviços em IoT foram apontados neste trabalho, a saber: a mobilidade tanto de *smart objects* quanto de *gateways*; grande heterogeneidade de *smart objects* e tecnologias de comunicação para acesso aos mesmos; a necessidade de interoperabilidade entre o mecanismo de descoberta de *smart objects*, diversas plataformas de *middleware*, *frameworks* e as aplicações da IoT/IoMT; diversas aplicações requerem combinação de informações dos *smart objects* com bases de conhecimento de forma a estabelecer critérios de combinação na consulta.

Com o intuito de tratar desses problemas, foi proposto um mecanismo integrado a um *middleware* móvel para descoberta, conexão, comunicação e anotação semântica de *smart objects*, um modelo semântico para ambientes IoT/IoMT, uma biblioteca que fornece uma API para construções de sistemas de informação e aplicações de IoT/IoMT e uma infraestrutura em nuvem para o processamento de fluxos semânticos de *smart objects* e de consultas instantâneas e contínuas.

Avaliou-se a solução proposta quantitativamente e qualitativamente, obtendo-se resultados muito satisfatórios. Em termos qualitativos, foi desenvolvido um cenário de uso utilizando a solução proposta que consistia em uma aplicação para detecção de vagas livres em um *Smart Parking* com base nos perfis dos usuários. Esta faz o uso consultas que combina o fluxo semântico de ocupação de vagas com conhecimento armazenado dos estacionamentos e dos motoristas, além de um modelo ontológico para superar a heterogeneidade dos *smart objects*, promovendo uma maior interoperabilidade entre *midlleware* e aplicações, explorando assim todos os requisitos da solução.

Em termos quantitativos, realizou-se uma série de experimentos controlados e aferição de métricas para medir o desempenho/eficiência da solução

proposta. Os resultados obtidos mostraram que o desempenho de cada componentes é muito baixo e estável. Pode-se identificar as influências dos fatores no tempo de processamento em que foi constatado que o fator que mais influência é o tamanho da base de conhecimento.

De modo geral, a avaliação qualitativa mostrou que o mecanismo proposto pode ser aplicado no desenvolvimento de aplicações IoT/IoMT do mundo real e que os resultados e as discussões da avaliação quantitativa mostraram que o mecanismo proposto apresenta um desempenho muito satisfatório em relação às métricas estabelecidas dentro das condições avaliadas.

7.1 Contribuições

Considera-se que as principais contribuições científicas desta pesquisa são:

- Revisão Sistemática da Literatura sobre a descoberta semântica de serviços em IoT, pois, de modo geral, as revisões existentes, embora analisem diversos aspectos de semântica em IoT, não tratam especificamente da descoberta semântica de serviço.
- A proposição de um mecanismo de descoberta semântica inovador, que permite uma descoberta dinâmica e em tempo próximo ao real de recursos da IoT/IoMT, com suporte a mobilidade, interoperabilidade, heterogeneidade dos *smart objects* e suas tecnologias de comunicação, as consultas complexas, ou seja, as consultas que combinem o fluxo eventos semânticos com bases de conhecimento.
- Identificação do fatores de influência no tempo do processamento semântico, haja vista que não há nenhuma avaliação desse tipo da *engine* C-SPARQL.

7.2 Trabalhos Futuros

Os próximos passos desta pesquisa vão em direção aos seguintes objetivos:

- Alocar o processamento semântico mais perto do limite da rede (computação de borda), a fim de reduzir a quantidade de dados transmitidos na rede e também a complexidade computacional necessária na nuvem.

- Ampliar o conjunto de avaliações quantitativas do mecanismo de descoberta, incluindo por exemplo, experimentos de escalabilidade.

7.3 Publicações

Para divulgação dos resultados desta pesquisa, foram escritos alguns artigos, que são apresentadas a seguir. Para cada publicação, informa-se o status em que ela se encontra e o Qualis do periódico ou conferência, se disponível.

- Costa, A.S; Alves R.S; Pereira, D.M.F; Santos, D.V.; da Silva e Silva, F. J; Endler, Markus. Dynamic Discovery of Services in IoT Based on Smart Objects Semantic Information Flows. Sensors (Basel), ISSN: 1424-8220, 2019.

  Qualis CAPES(2013-2016): A1 em Ciência da Computação e em Engenharia IV.

  Situação: Submetido (Em processo de avaliação).

- Costa, A.S; Alves R.S; da Silva e Silva, F. J; Endler, Markus. Descoberta Dinâmica de Serviços de IoT Baseado no Processamento Semântico de Fluxos de Eventos. Simpósio Brasileiro de Sistemas de Informação - SBSI 2019, Sergipe, Brasil.

  Qualis CAPES(2013-2016): B2 em Ciência da Computação.
  Situação: Aceito

- Costa, A.S.; Alves, R. S.; Silva, A. C.; da Silva e Silva, F. J. Uma abordagem de *Middleware* Para Geração de Fluxos de Eventos Semânticos em Ambientes de IoT. In: Jornada de Informática do Maranhão - JIM, 2018, São Luís, Maranhão, 2018.

  Situação: Publicado

Referências Bibliográficas

[1] The m3-lite taxonomy. `http://ontology.fiesta-iot.eu/ontologyDocs/m3-lite.html`. Acessado: 20/08/2018.

[2] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny. Unified iot ontology to enable interoperability and federation of testbeds. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016.

[3] H. G. Almeida and S. Siqueira. Uma revisão sistemática sobre descrição semântica na internet das coisas. *iSys-Revista Brasileira de Sistemas de Informação*, 11(2), 2018.

[4] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Stream reasoning and complex event processing in etalis. *Semantic Web*, 3(4):397–407, 2012.

[5] G. Antoniou and F. Van Harmelen. Web ontology language: Owl. In *Handbook on ontologies*, pages 67–92. Springer, 2004.

[6] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[7] S. Babu and J. Widom. Continuous queries over data streams. *ACM Sigmod Record*, 30(3):109–120, 2001.

[8] D. F. Barbieri, D. Braga, S. Ceri, E. D. VALLE, and M. Grossniklaus. C-sparql: a continuous query language for rdf data streams. *International Journal of Semantic Computing*, 4(01):3–25, 2010.

[9] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor. Iot-lite: a lightweight semantic model for the internet of things. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 90–97. IEEE, 2016.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

[11] D. Budgen and P. Brereton. Performing systematic literature reviews in software engineering. In *Proceedings of the 28th international conference on Software engineering*, pages 1051–1052. ACM, 2006.

[12] E. Chindenga, C. Gurajena, and M. Thinyane. Towards an adaptive ontology based model for interoperability in internet of things (iot). In *IST-Africa Week Conference, 2016*, pages 1–8. IEEE, 2016.

[13] S. Chun, S. Seo, B. Oh, and K.-H. Lee. Semantic description, discovery and integration for the internet of things. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 272–275. IEEE, 2015.

[14] A. Cockburn. *Agile software development*, volume 177. Addison-Wesley Boston, 2002.

[15] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Web semantics: science, services and agents on the World Wide Web*, 17:25–32, 2012.

[16] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor. The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25 – 32, 2012.

[17] O. G. Consortium. Location matters: Spatial standards for the internet of things. *ITU-T Technology Watch Report*, 2013.

[18] B. Daum and U. Merten. Arquitetura de sistemas com xml. *Rio de Janeiro: Campus*, 2002.

[19] S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker. Soda: Service oriented device architecture. *IEEE Pervasive Computing*, 5(3):94–96, 2006.

[20] M. Endler and F. e Silva. Past, present and future of the contextnet iomt middleware. *Open Journal of Internet Of Things (OJIOT)*, 4(1):7–23, 2018.

[21] D. Fensel. Ontologies. In *Ontologies*, pages 11–18. Springer, 2001.

[22] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.

[23] E. Freeman, E. Robson, B. Bates, and K. Sierra. *Head First Design Patterns: A Brain-Friendly Guide*. "O'Reilly Media, Inc.", 2004.

[24] B. Gomes, L. Muniz, F. J. d. S. e Silva, L. E. T. Ríos, and M. Endler. A comprehensive cloud-based iot software infrastructure for ambient assisted living. In *Cloud Technologies and Applications (CloudTech), 2015 International Conference on*, pages 1–8. IEEE, 2015.

[25] B. d. T. P. Gomes, L. C. M. Muniz, F. J. da Silva e Silva, D. V. dos Santos, R. F. Lopes, L. R. Coutinho, F. O. Carvalho, and M. Endler. A middleware with comprehensive quality of context support for the internet of things applications. *Sensors*, 17(12):2853, 2017.

[26] B. d. T. P. Gomes, L. C. M. Muniz, F. J. da Silva e Silva, L. E. T. Ríos, and M. Endler. A comprehensive and scalable middleware for ambient assisted living based on cloud computing and internet of things. *Concurrency and Computation: Practice and Experience*, 29(11):e4043, 2017.

[27] P. Gomes, E. Cavalcante, T. Batista, C. Taconet, S. Chabridon, D. Conan, F. C. Delicato, and P. F. Pires. A qoc-aware discovery service for the internet of things. In *Ubiquitous Computing and Ambient Intelligence*, pages 344–355. Springer, 2016.

[28] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[29] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[30] O. Hartig, C. Bizer, and J.-C. Freytag. Executing sparql queries over the web of linked data. In *International Semantic Web Conference*, pages 293–309. Springer, 2009.

[31] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, 1(1):7–26, 2003.

[32] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*, pages 791–798. IEEE, 2008.

[33] F. Khodadadi and R. O. Sinnott. A semantic-aware framework for service definition and discovery in the internet of things using coap. *Procedia Computer Science*, 113:146–153, 2017.

[34] S. Lee, H. Kim, D.-k. Hong, and H. Ju. Correlation analysis of mqtt loss and delay according to qos level. In *Information Networking (ICOIN), 2013 International Conference on*, pages 714–717. IEEE, 2013.

[35] J. Li, Y. Bai, N. Zaman, and V. C. Leung. A decentralized trustworthy context and qos-aware service discovery framework for the internet of things. *IEEE Access*, 5:19154–19166, 2017.

[36] J. Li, N. Zaman, and H. Li. A decentralized locality-preserving context-aware service discovery framework for internet of things. In *Services Computing (SCC), 2015 IEEE International Conference on*, pages 317–323. IEEE, 2015.

[37] M. Li, M. Liu, L. Ding, E. A. Rundensteiner, and M. Mani. Event stream processing with out-of-order data arrival. In *Distributed Computing Systems Workshops, 2007. ICDCSW'07. 27th International Conference on*, pages 67–67. IEEE, 2007.

[38] D. L. McGuinness, F. Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[39] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad hoc networks*, 10(7):1497–1516, 2012.

[40] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda. Task ontology for reuse of problem solving knowledge. *Towards Very Large Knowledge Bases: Knowledge Building & Knowledge Sharing*, 46:59, 1995.

[41] S. B. Mokhtar, D. Preuveneers, N. Georgantas, V. Issarny, and Y. Berbers. Easy: Efficient semantic service discovery in pervasive computing environments with qos and context support. *Journal of Systems and Software*, 81(5):785–808, 2008.

[42] B. Murteira, C. S. Ribeiro, J. A. e Silva, and C. Pimenta. *Introdução à estatística*. McGraw-Hill, 2007.

[43] T. Niemirepo, M. Sihvonen, V. Jordan, and J. Heinilä. Service platform for automated iot service provisioning. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on*, pages 325–329. IEEE, 2015.

[44] F. Paganelli and D. Parlanti. A dht-based discovery service for the internet of things. Vol. 2012, Article ID 107041, 10 2012.

[45] J. Z. Pan. Resource description framework. In *Handbook on ontologies*, pages 71–90. Springer, 2009.

[46] G.-J. Park. Design of experiments. *Analytic Methods for Design Practice*, pages 309–391, 2007.

[47] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1):414–454, 2014.

[48] K. Potter, H. Hagen, A. Kerren, and P. Dannenmann. Methods for presenting statistical information: The box plot. *Visualization of large and unstructured data sets*, 4:97–106, 2006.

[49] J. Quevedo, M. Antunes, D. Corujo, D. Gomes, and R. L. Aguiar. On the application of contextual iot service discovery in information centric networks. *Computer Communications*, 89:117–127, 2016.

[50] D. Robins. Complex event processing. In *Second International Workshop on Education Technology and Computer Science. Wuhan*, pages 1–10. Citeseer, 2010.

[51] L. E. Talavera, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. J. d. S. e Silva. The mobile hub concept: Enabling applications for the internet of mobile things. In *Pervasive computing and communication workshops (PerCom workshops), 2015 IEEE international conference on*, pages 123–128. IEEE, 2015.

[52] W. Wang, S. De, G. Cassar, and K. Moessner. An experimental study on geospatial indexing for sensor service discovery. *Expert Systems with Applications*, 42(7):3528–3538, 2015.

[53] R. S. Wazlawick. Uma reflexão sobre a pesquisa em ciência da computação à luz da classificação das ciências e do método científico. *Revista de Sistemas de Informação da FSMA*, 6:3–10, 2010.

A Apêndice A - Trabalhos Selecionados na RSL

A.1 Semantic description, discovery and integration for the Internet of Things

Este trabalho tem como objetivo utilizar diretório baseado em semântica para gerenciar os metadados e relacionamento dos *smart objects*. Para isso, foi desenvolvido um método eficiente para atualizar os metadados e realizar a descoberta de serviços através da sua semântica.

Primeiramente, foi introduzido um modelo semântico para estabelecer uma conceituação compartilhada, compreendendo as relações dinâmicas e estáticas entre os *smart objects*. O modelo proposto não se limita em apenas modelar dispositivos com sensores, mas também atuadores e dispositivos de etiquetas, com suas respectivas entidades e seus recursos. Para atualizar a descrição dos componentes loT e as mudanças em suas relações, cada componente contém um conceito denominado *Modeltype*, que pode ser estático ou dinâmico. Além disso, ele pode ser ativo ou passivo, que serve para decidir a interação entre o diretório e os *smart objects*. Esses dois valores devem ser utilizados juntos, por exemplo, estático e passivo.

Um componente com um atributo estático indica que os valores dos atributos não mudam, como por exemplo, a localização de um dispositivo estático. No caso de um componente definido como dinâmico, indica que seus valores variam com frequência, e uma nova relação entre os componentes pode ser criada ou uma relação existente pode ser removida. A interação entre os objetos IoT e o diretório, foi modelado de duas formas, ativo e passivo. No modelo passivo, indica que o diretório deve enviar um pedido ao objeto IoT para o obter suas informações atualizadas. Por outro lado, um modelo ativo indica que um objeto loT encaminha os meta-dados atualizados para o diretório sempre que seus meta-dados mudam.

O diretório foi desenvolvido baseado no modelo de componentes IoT proposto. Para armazenar os dados semânticos dos objetos IoT, como triplas RDF

dentro o diretório, foi utilizado o *Triple Store*. Para a descoberta dos objetos IoT heterogêneos, o sistema permite que o cliente busque os objetos de acordo com sua semântica, especificando consultas em SPARQL.

Para validação, foi implementado um portal web, onde é possível registrar objetos IoT e descobrir objetos. Através de uma interface gráfica baseada no modelo de componentes, o usuário pode facilmente publicar um objeto. Para o usuário final, o portal ajuda a gerar uma consulta SPARQL, pois já traz através de componentes como combo box, os componentes e suas relações.

A.2 A decentralized locality-preserving context-aware service discovery framework for internet of things

No ambiente da IoT, a localização de serviços desejáveis é desafiadora devido à considerável diversidade, grande número, comportamento dinâmico e distribuição geográfica dos serviços fornecidos por objetos físicos. Para abordar os desafios de descoberta em IoT, foi proposto um *framework* de descoberta baseado em semântica com reconhecimento de contexto denominado LOCA *(LOcation-preserving Context-Aware)*.

O LOCA usa ontologias para codificar as informações de contexto e combinar consulta com serviços para selecionar os serviços mais apropriados. O modelo ontológico visa representar conceitos referentes ao o contexto, usuários, dispositivos, serviços e ambientes. Foi modelado apenas o contexto mais fundamental, porém o modelo é flexível pois possibilita a adição de conceitos especifico em diferentes domínios de aplicação (por exemplo, casa), não limitando a um conjunto fixo de atributos ou propriedades.

Para descobrir serviços, foi proposto duas técnicas de filtragem de contexto: raciocínio de contexto baseado em lógica e correspondência de contexto baseada em semântica. A primeira utiliza a ontologia, que por sua vez está no formato OWL, para explorar o raciocínio lógico. Já na técnica de correspondência semântica, utiliza a ontologia de contexto para calcular a similaridade semântica dos serviços. Esse calculo leva em consideração a distância semântica das entidades em um grafo. Para suportar

a flexibilidade, é permitido que os usuários personalizem o fator de distância para refletir suas necessidades.

A fim de localizar eficientemente os serviços, foi proposto um mecanismos de descoberta descentralizado baseado em P2P com Tabelas hash distribuídas (*Distributed Hash Table* - DHT) para implementar serviços de descoberta distribuídos escaláveis e robustos, pois os DHTs fornecem boas propriedades de balanceamento de cargar, porém não pode controlar onde os dados são armazenados. Para isso, foi adotado a estrutura SkipNet. Ela, além de ser tolerante a falhas, permite um controle sobre a disponibilidade e armazenamento do dados.

Quando novos eventos/serviços são gerados na rede, eles são armazenados no repositório de informações local. Cada ponto expões um conjunto de APIs para descoberta que podem ser invocados pelos aplicativos clientes para procurar serviços desejáveis. Ao mesmo tempo, ele mantém uma tabela de roteamento para habilitar o roteamento de consulta, onde retorna os identificadores dos repositórios que manipulam informações e serviços sobre esses objetos.

Para avaliação foi realizada uma comparação desse mecanismo com um de correspondência baseada em palavras-chaves. Os desempenhos são comparados em termos de recuperação e precisão. Com isso foi possível observar que a combinação baseada em semântica pode melhorar drasticamente a taxa de recuperação de correspondência em relação a de correspondência por palavras-chave, pois encontra mais entidades de contexto que estão semanticamente relacionadas. Em comparação de precisão, as duas apresentaram resultado iguais, pode-se perceber que a combinação semântica não prejudicará a precisão. Outro experimento realizado foi a verificação da porcentagem de consultas locais e a examinação da sobrecarga de descoberta de serviços em termos de saltos físicos percorridos pela consulta de descoberta.

A.3 A Semantic-aware Framework for Service Definition and Discovery in the Internet of Things Using CoAP

Este trabalho tem como objetivo propor um sistema que facilite a definição do serviços aproveitando as linguagens de definição de API e a implementação de um componente de descoberta eficiente através serviços semânticos. A ideia proposta é modelar todos os recurso disponíveis como serviços, sejam eles sensores, dispositivos de ponta, recursos em nuvem ou serviços convencionais. Além disso, os seres humanos são modelados como serviço, permitindo-lhes interagir com os serviços expostos por outras entidades, criando assim um *marshup* IoT.

Os serviços são descritos usando TDD (*Thing Description Document*). Um TDD consiste em duas partes principais: propriedades das entidades e os serviços oferecidos por cada entidade, onde são serializados através de JSON-LD. Essa anotação é baseada em uma ontologia que foi derivada do Modelo de Serviço Mínimo (MSM), e são armazenados em formato RDF em um banco de dados *Apache Triple* e é utilizado o SPARQL para suportar a consultas nesses dados.

O módulo de descoberta baseado em semântica é implementado como um serviço da Web RESTful e pode ser chamado a partir de um ponto de final CoAP. O CoAP ajuda a reduzir os gastos gerais de usar pedidos e respostas HTTP normais. Um benefício de usar o CoAP é que os dispositivos IoT consomem menos energia e conseguem um melhor desempenho. Logo, um servidor proxy é introduzido para lidar com comunicações que usam esse protocolo.

A avaliação desse sistema foi realizado através de um estudo de caso focado na economia de energia em data center (DC). A aplicação desenvolvida usa os serviços disponíveis no DC para obter uma visão geral consumo de energia, ou ajustar a temperatura da sala do servidor seguindo alguma politica, ou calcular o consumo de energia. Este cenário só é possível se todos os recursos, desde sensores até *hosts* físicos reais, expuserem suas propriedades e serviços de tal maneira que os provedores da nuvem possam usar os recursos de composição de serviços e descoberta de recursos da estrutura para encontrar os serviços desejados. E com a semântica a integração e composição e descoberta do serviço podem ser aplicadas de forma mais efetiva.

O *framework* utiliza um *proxy* CoAP em vez do protocolo HTTP para reduzir a largura de banda e o consumo total de energia da rede. Por exemplo, quando 1000 novas entidades são registradas e o TDD para cada uma contém 3 serviços e 2 propriedades, executar uma consulta SPARQL sobre o ponto final do CoAP reduz a largura de banda em 15%.

A.4 Service Platform for Automated IoT Service Provisioning

Neste artigo, foi definido uma arquitetura de plataforma de serviço independente de domínio que possibilita o provisionamento automatizado de serviços IoT. Para isso, foi aplicado a abordagem de Arquitetura de Dispositivos Orientados a Serviços (*Service-Oriented Device Architecture* - SODA) [19], com o objetivo de permitir que os dispositivos sejam conectados a uma arquitetura orientada a serviços (*Service-Oriented Architecture* - SOA).

O serviço do dispositivo nada mais é do que o encapsulamento em formato JSON de um dispositivo como um serviço, transformando esses dados de acordo com a semântica definida, fazendo com que o formato dos dados sejam uniforme e independente de dispositivos que os produzem. Apesar de definir uma semântica para os dispositivos, este trabalho não faz uso de ontologias.

A comunicação com o servidor é feito através do protocolo HTTPS, que por sua vez é uma implementação do protocolo HTTP sobre uma camada adicional de segurança que utiliza o protocolo SSL/TLS. Essa camada adicional permite por exemplo, que os dados sejam transmitidos por meio de uma conexão criptografada.

Quando um usuário introduz um novo dispositivo, ele é modelado semanticamente e armazenado no servidor. As informações sobre os serviços da IoT e informações associadas ao dispositivo ficam armazenadas em um servido. O sistema é capaz de descobrir os serviços de IoT que exigem esse um certo tipo de dispositivo. Este serviço encontrado é exibido para o usuário. Essa descoberta automatizada só é possível se o sistema for capaz de identificar tanto os dispositivos requeridos pelo serviço IoT quanto os dispositivos disponíveis para o usuário.

A Figura A.1 demonstra um cenário de uso da solução proposta. O sistema suporta situações em que um dispositivo é conectado a dois ou mais dispositivos e também na qual serviço de IoT requer mais de um dispositivo, em que o usuário tem dois dispositivos à sua disposição que é o sensor de movimento e o sensor de porta. O provedor de serviços registrou dois serviços e seus manipuladores de dados que definem que tipo de dispositivos eles exigem. Um serviço pode exigir mais de um dispositivo. O usuário pode solicitar "Serviço de segurança"somente se tiver os dispositivos "sensor de movimento"e "Sensor de porta".

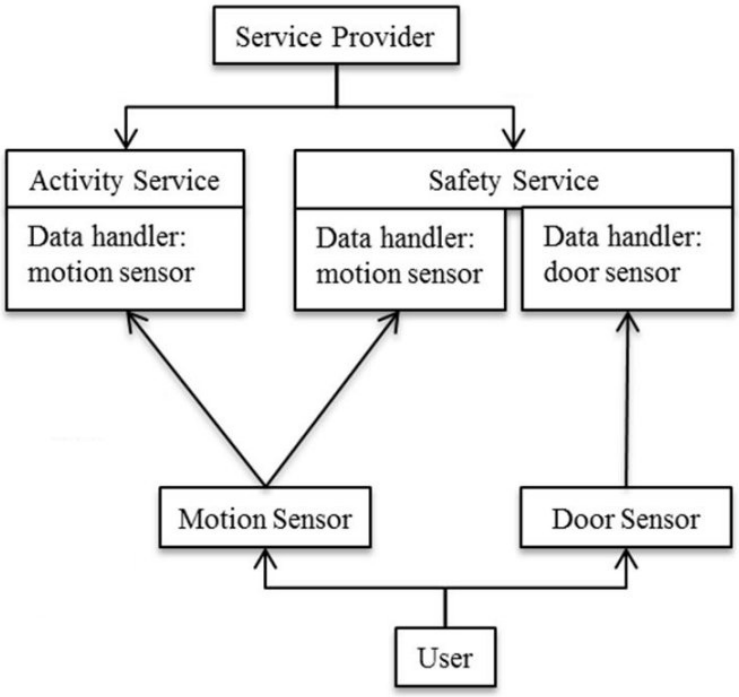

Figura A.1: Anexando dispositivos do usuário aos serviços da IoT. Fonte: [43]

A plataforma oferece aos provedores de serviços de IoT a possibilidade de se concentrar no conhecimento de domínio de sua própria aplicação, como algoritmos para analisar mudanças na pressão arterial ou oferecer serviços de segurança baseados em alarmes sem se preocupar com os dispositivos à disposição do usuário e sua engenharia.

A.5 Towards an adaptive ontology based model for interoperability in internet of things (IoT)

O objetivo deste artigo é investigar alguns dos requisitos para a interoperabilidade semântica, além disso, propor um modelo para descoberta de dispositivos e serviços semânticos em IoT. O modelo proposto é denominado Modelo de Descoberta de Dispositivo e Serviço (*Device and Service Discovery Model* - DSDM).

A ontologia da base de conhecimento é construída para interpretação semântica e armazenamento de informações de dispositivos e serviços. Ele descreve a semântica para interoperabilidade e relacionamentos de dispositivos e consiste em um repositório de dispositivos e repositório de serviço que é descritos usando OWL (*Web Ontology Language*).

A descoberta de dispositivos pode ser realizada de duas maneiras. A primeira sendo iniciada pelo dispositivo solicitando a conexão à rede. O dispositivo envia uma solicitação ao DSDM por meio de um *broadcast* e o DSDM responde à chamada por meio de um *unicast*. Como alternativa, um dispositivo pode ser descoberto quando responde ao *broadcast* do DSDM, isso significa que os dispositivos que estão em um estado *off* permanecem invisíveis para o DSDM. A descoberta é implementada usando o protocolo de descoberta UPnP *Simple Service Discovery Protocol* (SSDP).

Ao utilizar as informações armazenadas na ontologia, o DSDM é capaz de compor serviços conforme a solicitação do dispositivo. Também gerencia as solicitações de serviço. Mais importante ainda, fornece um serviço de contabilidade, controlando os serviços disponíveis e os solicitados. Também gerencia dispositivos conectados. Foi desenvolvido um componente de comunicação fornece uma interface para dispositivos para se comunicar com o DSDM. Com isso, facilita a interação heterogênea com outros dispositivos, abrangendo uma variedade de protocolos de comunicação.

O modelo também suporta a adaptação do sistema atualizando constantemente o repositório, mantendo o ambiente atualizado. O modelo de adaptação dinâmica garante que a rede tenha informações atualizadas do dispositivo

ao adicionar dinamicamente novas descrições de serviço à medida que novos dispositivos se adicionam à rede.

A.6 A Decentralized Trustworthy Context and QoS-Aware Service Discovery Framework for the Internet of Things

Neste artigo, foi proposto uma estrutura descentralizada de descoberta de serviços baseada em semântica, que pode localizar efetivamente serviços confiáveis com base nas exigências de qualidade de serviço (QoS) do solicitante e na alteração dos requisitos de contexto. A estrutura é baseada em SkipNet para permitir serviços de descoberta distribuída escalável e segura.

A ontologia adotada no sistema é fruto da reutilização, integração de ontologias de contexto e de QoS existente. É categorizado em três temas distintos mas que são relacionados: (a) ontologias sobre dispositivos físicos, (b) ontologias sobre agentes (agentes humanos e de software), (c) ontologias sobre o contexto ambiental dos agentes. Um fator importante dessa ontologia é as relações sociais entre dispositivos, pois através delas, o sistema de descoberta pode rastrear e descobrir os serviços IoT seguindo diferentes níveis de confiabilidade entre as entidades IoT.

Tanto as solicitações de serviços como os anúncios de serviços são definidos semanticamente usando conceitos da ontologia. Especificamente, todas as propriedades necessárias especificadas na descrição de serviço solicitada são comparadas com as propriedades fornecidas pelos serviços anunciados. Se dois ou mais serviços anunciados satisfizerem igualmente as propriedades funcionais, é selecionado o serviço que melhor satisfaça as propriedades não funcionais (por exemplo, contexto e QoS).

Para reduzir o custo do raciocínio semântico em ontologias, foi adotado o mecanismo de otimização de codificação numérica proposto por Mokhtar et al. [41]. Esta técnica de codificação é baseada em números primos, atribuindo esse número a cada classe na hierarquia ontológica. Com essa codificação pode identificar se dois

conceitos em uma ontologia estão relacionados sem a execução onerosa do raciocínio semântico.

Foi conduzido um conjunto de experimentos de simulação para medir a eficácia do mecanismos de descoberta de serviços. Nestes experimentos, foi gerado a topologia de rede usando um gerador de topologia popular GT-ITM. Na primeira parte do experimento, o desempenho do mecanismo de descoberta foi analisado de acordo com o tamanho da rede e a taxa de sucesso. Uma solicitação de serviço é satisfeita se o perfil do serviço corresponder à solicitação em propriedades. E pôde-se verificar que ele alcança alta taxa de sucesso (60%) mesmo em uma rede muito grande (ou seja, 8192 número de nós), mas só consegue satisfazer 100% a solicitação com 521 nós na rede.

Para verificar o desempenho, foi feito uma comparação do tempo de resposta em função do tamanho da rede, do sistema proposto de sobreposição de preservação de localidade com um de sobreposição DHT. Concluiu-se que usando a sobreposição de preservação de localidade, o tempo de resposta aumenta logaritmicamente à medida que o tamanho da rede cresce. Isso demonstra a boa escalabilidade da estrutura de sobreposição proposta, pois ela se adapta bem a uma grande rede, como uma rede DHT.

Foi realizada uma comparação da taxa de sucesso da correspondência baseada em semântica e da correspondência baseada em palavras-chave. E pôde-se verificar que correspondência baseada em semântica pode melhorar drasticamente a taxa de sucesso correspondente, encontrando mais entidades que são semanticamente relacionadas

A.7 An experimental study on geospatial indexing for sensor service discovery

Este trabalho tem como por objetivo desenvolver uma arquitetura de descoberta escalável usando técnicas de indexação geoespacial e tecnologias de serviços semânticos. Ele é centrado em dois componentes funcionais: *Gateway* Distribuído responsável por gerenciar a comunicação dos sensores e abstrair suas

funcionalidades em serviços RESTful, anotadas de acordo com um modelo semântico; e o Servidor de Descoberta (Discovery Server) responsável pelo serviço de descoberta.

No servidor de descoberta, o índice geoespacial ajuda o mecanismo de descoberta a reduzir o espaço de busca e localizar o(s) *gateway(s)* que provavelmente conterão os serviços em relação às consultas. Para isso é utilizado uma Árvore-R que organizar os *gateways* próximos em retângulos envolventes mínimos (*minimum bounding retangle* - MBR), e cada um mantém os dados dos sensores que estão dentro do seu retângulo. Esses retângulos delimitadores dos *gateways* permanecem inalterados. A Figura A.2 mostra Árvore-R de *gateways* adotada neste trabalho.

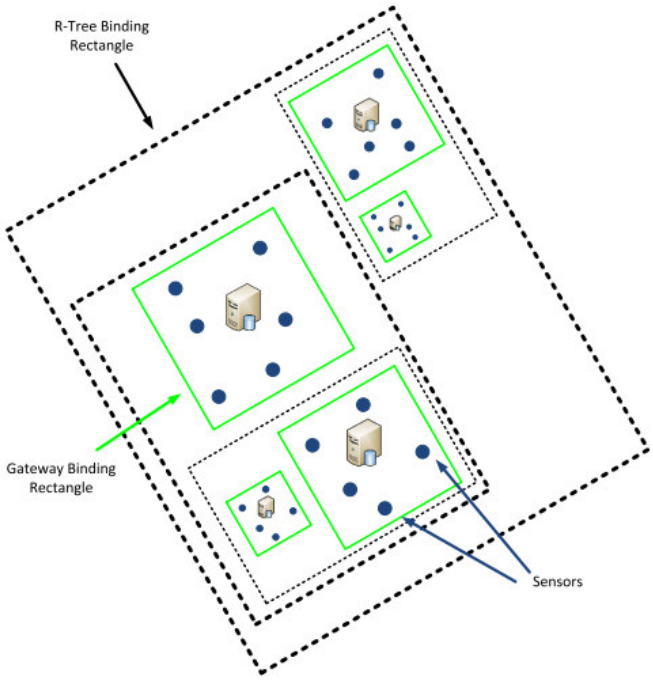

Figura A.2: Árvore-R de *gateways*. Fonte: [52]

Durante a descoberta, as informações de localização fornecidas na consulta são usadas primeiro para pesquisar o índice geoespacial para localizar os *gateways* que potencialmente contêm os serviços necessários. Em seguida, as consultas são encaminhadas para os repositórios semânticos nesses *gateways*, que posteriormente realizam pesquisas semânticas usando o SPARQL. Os serviços que fornecem todas as funcionalidades necessárias e tipos semânticos (por exemplo, temperatura ou umidade) são então recuperados e retornados ao solicitante

Foi reutilizado a ontologia desenvolvida no projeto IoT.est.[1], que define aspectos do serviço não funcional como nome do serviço, categoria, QoS e localização, aspectos funcionais, como as operações permitidas no serviços e o recursos Restful, como por exemplo, parâmetros de entrada/saída e as URLs).

Para a avaliação do método proposto, foi realizado uma comparação com outro método de indexação geoespacial chamado Indexação Geoespacial de Serviços de Sensores (GISS), que cria um índice com serviços de sensores individuais, como os nós folha. Relacionado ao tempo de resposta da consulta, os resultados mostram que o o método proposto responde a uma consulta muito mais rápido que o GISS.

Outra avaliação realizada foi referente a taxa de transferência, que pode-se observar que ambos os métodos são de fato escalonáveis, pois a taxa de transferência não diminui à medida que o número de consultas simultâneas aumenta (para até 100 consultas simultâneas).

A.8 On the application of contextual IoT service discovery in information centric networks

Neste artigo, é proposto um mecanismo de descoberta de serviços, baseado em Rede de Dados Nomeados (*Named Data Networking* - NDN), que aproveita o uso de um mecanismo de correspondência semântica para alcançar um processo de descoberta flexível. A solução considera, como mostrado na Figura A.3, quatro entidades básicas: Clientes, Provedores de Serviços, *Brokers* de Descoberta e Motor de Correspondência Semântica (*Semantic Matching Engines* - SME). As diferentes entidades interagem entre si através do uso de interfaces bem definidas.

O provedor de serviço é uma entidade que fornece um ou mais serviços (por exemplo, sensores, atuadores). Utiliza o protocolo NDN para se comunicar com o *Broker* de Descoberta e com os Clientes interessados através da interface Is e Ir. Ele envia uma solicitação ao *Broker* de Descoberta para adicionar/remover seus serviços e fornece aos clientes o conteúdo solicitado.

[1] http://www.ict-iotest.eu/

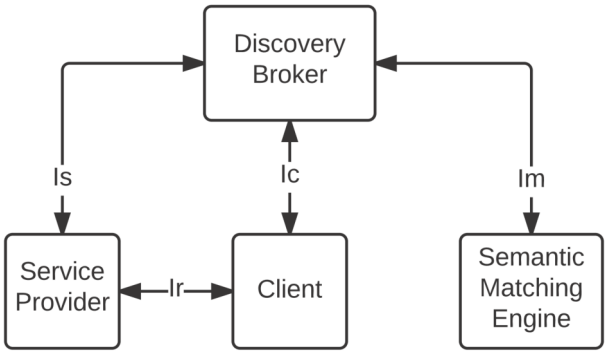

Figura A.3: Visão geral da solução: entidades e interfaces. Fonte: [49]

Já o *Broker* de Descoberta é responsável por manter as informações sobre os serviços disponíveis em uma tabela local e encaminha parte das informações recebidas para o SME, além de combinar as consultas recebidas com esses serviços. Com os Clientes e os Provedores de Serviço a comunicação é feita usando o protocolo NDN, já com o SME é através do protocolo de transporte (TPC).

O SME escuta as consultas provenientes do *Broker* de Descoberta, executa os diferentes algoritmos de correspondência e responde com uma lista dos serviços relevantes (ou seja, serviços para os quais há uma correspondência positiva entre os termos incluídos na consulta e as *tags* usadas para descrever o serviço). O uso de um comparador semântico como parte da solução de descoberta de serviço aumenta a flexibilidade, permitindo a correspondência correta de consultas e serviços em que nenhuma das palavras é uma correspondência exata, mas sim sinônimos.

Para descobrir os serviços disponíveis, os clientes devem enviar uma consulta para o *Broker* de Descoberta. A consulta inclui uma descrição semântica dos serviços desejados. O *Broker* encaminha a solicitação para o SME, que determina o conjunto de serviços relevantes e retorna os IDs correspondentes para o *Broker*. Ele processa esses IDs e retorna a descrição completa dos serviços para o cliente. O cliente pode solicitar diretamente o conteúdo aos provedores de serviços de acordo com os princípios da arquitetura de Redes Centrais de Informação (*Information Centric Networking* - ICN).

Foi avaliado o tempo de serviço para as três operações principais da solução: registro de serviço, cancelamento de registro de serviço e consulta de serviço. O tempo de descoberta e o número de serviços registrados exibem uma relação direta, não apenas devido ao aumento do tamanho da resposta, mas também devido ao aumento

do tempo de processamento no comparador semântico. Os resultados mostram que o tempo de serviço para procedimentos de cancelamento de registro é menor do que aqueles dos procedimentos de registro.

A.9 A QoC-aware discovery service for the Internet of Things

Este trabalho tem como objetivo desenvolver um serviço de descoberta ciente de contexto com consultas com múltiplos-atributos, intervalos e operações síncronas/assíncronas denominado QoDisco. Ele inclui um modelo de informação baseado em ontologia para descrever semanticamente recursos, serviços e informações relacionadas à QoC.

O modelo de informações QoDisco aproveita a ontologia SAN, uma extensão da ontologia SSN do W3C que fornece conceitos, atributos e propriedades para modelar sensores e atuadores. Além disso, há uma parte da ontologia de SOUPA para descrever localizações espaciais de entidades (*PhysicalStructure*) em termos de latitude, longitude, altitude, distância e superfície, bem como representações simbólicas de espaço. Para lidar com as preocupações relacionadas à QoC, foi incorporado parte do meta-modelo QoCIM.

A arquitetura do QoDisco compreende quatro módulos, cada um responsável por um conjunto de funcionalidade específicas providas pela plataforma. A Figura A.4 ilustra esses módulos.

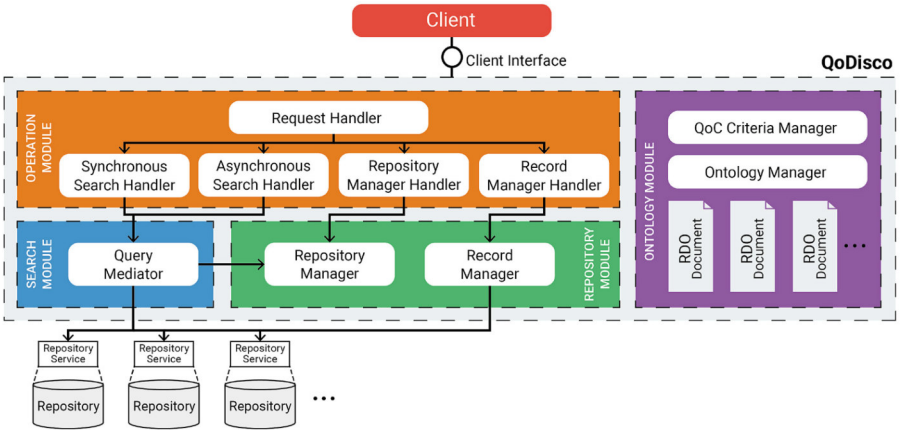

Figura A.4: Arquitetura QoDisco. Fonte: [27]

O *Módulo de Ontologias* consiste de documentos RDO *(Repository Domain Ontology)* que descrevem os conceitos que compõem o modelo de informação do QoDisco. Neste módulo, o componente *Ontology Manager* oferece operações para adicionar, remover e modificar elementos RDO. Já o componente *QoC Criteria Manager* é responsável gerenciar os critérios de QoC.

A principal funcionalidade do *Repository Module* é gerenciar repositórios e mapeá-los para o documento RDO presentes do *Ontology Module*. Por sua vez, o *Record Manager* é responsável por adicionar, remover e modificar registros nos repositórios.

O *Search Module* compreende o componente *Query Mediator*, que encaminha *queries* para os repositórios registrado no QoDisco. Para realizar a busca, esse componente fornece o nome de um documento RDO (i.e., nome de domínio) ao componente *Repository Manager*, que provê a URL dos repositórios mapeados para tal domínio.

O *Operation Module* é composto pelos cinco seguintes componentes:

- *Request Handler*: É o ponto de entrada do *Operation Module*, recebendo requisições síncronas e assíncronas para busca, além de requisições para o gerenciamento de repositórios e registros.
- *Synchronous Search Handler:* Realiza as consultas síncronas por registros usando o *Query Mediator*
- *Asynchronous Search Handler:* Lida com consultas assíncronas de registros.
- *Repository Manager Handler:* Interage com o componente *Repository Manager* para realizar as operações de adição e remoção de um registro.
- *Record Manager Handler:* Interage com o componente *Record Manager* para adição, remoção e modificação de registro.

A *Client Interface* é o ponto de entrada para o QoDisco, fornecendo as funcionalidades do *Operation Module* e *Ontology Module* aos clientes, obedecendo o estilo arquitetural REST (*Representation State Transfer*) através do protocolo HTTP.

Para avaliação do QoDisco, foi desenvolvido um cenário de uso para monitoramento da poluição do ar. Para analisar seu desempenho da busca por

recursos, foi medido o tempo gasto pelo QoDisco para responder a uma consulta do cliente síncrona à medida que o número de observações aumenta. Já a busca assíncrona foi medido o tempo médio (em milissegundos) para notificar os clientes, com um número crescente de solicitações do cliente.