UNISINOS

Programa de Pós-Graduação em

Computação Aplicada

Doutorado Acadêmico

Alexandre Stürmer Wolf

ADAPTTHING:

Modelo computacional para gerenciamento dinâmico e adaptativo de objetos da IoT utilizando histórico de contextos

São Leopoldo, 2019

Alexandre Stürmer Wolf

ADAPTTHING:

Modelo computacional para gerenciamento dinâmico e adaptativo de objetos da IoT utilizando histórico de contextos

Tese apresentada como requisito parcial para a obtenção do título de Doutor, pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Dr. Jorge Luis Victória Barbosa

São Leopoldo

2019

W853a Wolf, Alexandre Stürmer.
AdaptThing : modelo computacional para gerenciamento dinâmico e adaptativo de objetos da IoT utilizando histórico de contextos / por Alexandre Stürmer Wolf. – 2019.
130 f. : il. ; 30 cm.

Tese (doutorado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2019.
"Orientador: Dr. Jorge Luis Victória Barbosa".

1. Internet das Coisas. 2. Adaptação dinâmica. 3. Histórico de contextos. I. Título.

CDU: 004

Catalogação na Publicação (CIP):
Bibliotecário Alessandro Dietrich - CRB 10/2338

A todas as pessoas que fizeram um sacrifício ou foram literalmente sacrificadas para que este trabalho fosse concluído.

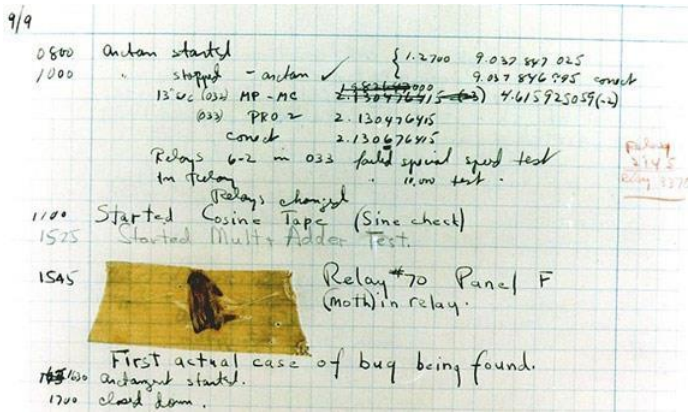
AGRADECIMENTOS

Aos meus pais, pelo carinho, paciência, incentivo e apoio em todas as horas. Agradecimento especial para minha filha Camila, que foi a maior motivação para a conclusão do processo de doutoramento, inclusive por manter o prumo na vida.

Agradeço as pessoas que "perdi" ao longo do processo, mas que mantenho com carinho a lembrança de todos os bons e maus momentos.

Um agradecimento especial ao meu orientador, o prof. Dr. Jorge Luis Victória Barbosa pelo empenho dedicado à elaboração deste trabalho, além dos ensinamentos, cobranças e principalmente motivação - *Avante*.

At 3:45 p.m., Grace Murray Hopper records 'the first computer bug' in the Harvard Mark II computer's log book. The problem was traced to a moth stuck between relay contacts in the computer, which Hopper duly taped into the Mark II's log book with the explanation: “First actual case of bug being found.” The bug was actually found by others but Hopper made the logbook entry (09/09/1947).



This Day in History: September 9
Computer History Museun

```
mov $1 , %rax
mov $1 , %rdi
mov $_ , %rsi
mov $30, %rdx
syscall
_: .ascii "The future of the education:\n"
```

RESUMO

Atualmente é esperado que os ambientes inteligentes tenham a capacidade de responder a eventos, principalmente em situações inesperadas. Os ambientes inteligentes são favorecidos pela evolução tecnológica, que possibilitou o surgimento do paradigma da Internet das Coisas – *Internet of Things* (IoT), que permite a conectividade entre diferentes sistemas e dispositivos, sejam físicos ou virtuais, através da Internet. Tanto os sistemas, quanto os dispositivos, podem ser adaptados conforme as necessidades dos ambientes, sejam através de monitoramento ou até mesmo ajustando parâmetros de funcionamento, respondendo assim, de forma dinâmica as mudanças existentes. Para isso, são necessários elementos sensoriais, chamados de objetos sensitivos, utilizados para coletarem informações dos ambientes. Além de coletar e obter os dados de fontes heterogêneas é necessário armazenar os dados de forma a constituir uma base histórica, para consulta e inferência, considerando as características do evento, localização, e momento em que ocorreu, gerando assim históricos de contextos. Com base nos históricos de contextos, aliado a novos eventos, é possível inferir a necessidade de reconfiguração do comportamento operacional dos objetos sensitivos, realizando até mesmo a realocação de recursos móveis para áreas menos densamente monitoradas, permitindo assim dados mais confiáveis e detalhados. Dessa forma, esta tese propõe o modelo computacional AdaptThing, que adapta dinamicamente dispositivos físicos e virtuais da IoT, através de uma rede de objetos sensitivos heterogêneos, com a capacidade de realizar a adaptação dinâmica do comportamento operacional dos elementos envolvidos, de forma a melhorar a resolução e detalhamento dos dados. O modelo computacional foi implementado e avaliado em dois cenários de aplicação. Um cenário foi educacional, onde o sistema forneceu questões de acordo com o conhecimento médio da turma onde aplicado, reduzindo o número de questões do mesmo assunto em 33,3 %. O outro cenário envolveu um conjunto de 25 estações climáticas profissionais, onde uma das estações teve seu funcionamento adaptado com base nas informações do contexto, reduzindo o seu consumo computacional em 67 %. Dessa forma considera-se que o modelo computacional AdaptThing possui a capacidade de gerenciar objetos sensitivos da IoT, adaptando dinamicamente o comportamento operacional de funcionamento, permitindo maior detalhamento de informações.

Palavras-Chave: Internet das Coisas. Adaptação dinâmica. Histórico de contextos.

ABSTRACT

It's currently expected that any environment will become an Intelligent Environment, capable of responding to events, especially in unexpected situations. Intelligent environments are favored by technological evolution, which has enabled the emergence of the Internet of Things (IoT) paradigm, which allows connectivity between different systems and devices, whether physical or virtual, through the Internet. Both systems and devices must adapt to the needs of environments, responding dynamically to existing changes. This requires sensory elements, called sensory objects, used to collect information from the environment. In addition to collecting and obtaining data from heterogeneous sources, it is necessary to store the data in such a way as to constitute a historical basis for consultation and inference, considering the characteristics of the event, location, and moment in which it occurred, thus generating a history of contexts. Based on contextual history, coupled with new events, it is possible to infer the need to reconfigure the operational behavior of sensitive objects, even relocating mobile resources to less densely monitored areas, thus enabling more reliable and detailed data. Thus, this thesis proposes the AdaptThing computational model, which supports heterogeneous sensitive object networks, with the ability to dynamically adapt the operational behavior of the elements involved, to improve data resolution and detail. The computational model was implemented and evaluated in two application scenarios. One scenario was educational, where the system provided questions according to the average knowledge of the class where applied, reducing the number of questions of the same subject by 33.3%. The other implementation scenario was applied to a set of 14 professional climate stations, where one of the stations had its operation adapted based on contextual information, reducing its computational consumption by 67%. Thus, it is considered that the AdaptThing computational model can manage IoT sensitive objects, dynamically adapting operational operating behavior, allowing for more detailed information.

Keywords: Internet of Things. Dynamic adaption. History contexts.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACM	Association for Computing Machinery
ANSI	American National Standards Institute
API	American National Standards Institute
ASCII	American Standard Code for Information Interchange
BASIS	Big Data Architecture for Smart Cities
BSD	Berkeley Software Distribution
CAN	Controller Area Network
CCN	Content Centric Networking
CoAP	Constrained Application Protocol
DAML	DARP Agent Markup Language
DARP	Defense Advanced Research Projects
DARPA	Defense Advanced Research Projects Agency
DLNA	Digital Living Network Alliance
DSL	Digital Subscriber Line;
DTD	Document Type Definition
DTLS	Datagram Transport Layer
EMML	Enterprise Mashup Markup Language
ESO	External Sensing Objects
FCAPS	Fault, Configuration, Accounting, Performance, Security
FIT	Failures in Time
FORTRAN	IBM Mathematical FORmula TRANslation System
HDFS	Hadoop Distributed File System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
IBM	International Business Machines
IEEE	Institute of Electrical and Electronics Engineers
IFTTT	If This Than That
IoT	Internet of Things
IP	Internet Protocol
IPC	Inter Process Communications
ISO	International Organization for Standardization
ITU	International Telegraph Union

ITU T	Telecommunication Standardization Sector
LTE	Long Term Evolution Networks
M2M	Machine to Machine
MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport Protocol
MSO	Mobile Sensing Objects
MTBF	Mean Time Between Failures
MTBR	Mean Time to Repair
MTTF	Mean Time to Failures
NSO	Network Sensing Objects
OIL	Ontology Inference Layer
OKBC	Open Knowledge Base Connectivity
OSI	Open System Interconnection
OWL	Web Ontology Language
P2P	Peer to Peer
PnP	Plug in Play
PSTN	Public Switched Telephone Network
QUIC	Quick UDP Internet Connections
RaaS	Robot as a Service
RDF	Resource Description Framework
REST	Representational State Transfer
RFC	Request for Comments
RFID	Radio Frequency Identification
ROLL	Routing Over Low power and Lossy networks
RPL	IPv6 Routing Protocol for Low
SHOE	Simple HTML Ontology Extensions
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer

LISTA DE SÍMBOLOS

B	Byte
EB	Exabyte
GB	Gigabyte
Gb	Gigabit
KB	Kilobyte
MB	Megabyte
Mb	Megabit
PB	Petabyte
RPM	Rotações por minuto
TB	Terabyte
Tb	Terabit
YB	Yottabyte
ZB	Zettabytes

SUMÁRIO

1 INTRODUÇÃO

Ambientes Inteligentes (CHEN, 2009), (REINISCH, 2011), (SKILLEN, 2012), (CASTILLEJO, 2014), (TELLES, 2016), (LI et al., 2018), (RIOS et al., 2018), procuram ter a capacidade de prever e responder a eventos. Nos últimos anos houve a disponibilização de uma miríade de dispositivos eletrônicos, que oferecem conectividade com a Internet, grande poder computacional, facilidade de utilização, aliado a um custo acessível, podendo ter seu funcionamento estático ou móvel.

Alguns desafios da atualidade apresentam-se na padronização, armazenamento e o processamento dos dados provenientes destes dispositivos, de forma a permitir a integração e inferência em tempo real. A disponibilidade das informações dos mais diversos dispositivos propiciou o cenário da computação ubíqua (BARBOSA, 2015).

Inicialmente vislumbrado por Weiser (1991), o paradigma da ubiquidade baseia-se em três pilares, que são a diversidade, conectividade e descentralização, que devem ocorrer de forma omnipresente e transparente, ou seja, sempre presente e sem que seja percebido. A Diversidade preconiza que cada dispositivo deve ter seu fim único, ou seja, deve atender à sua funcionalidade específica e prestar bem seu serviço único, sem a necessidade de realizar outras tarefas. A Conectividade traz a ideia de que cada dispositivo deve ser capaz de comunicar-se com os outros, independentemente de sua localização geográfica. A Descentralização garante que um dispositivo seja capaz de realizar suas funções, independente de outros dispositivos, ou seja, ele deve existir e funcionar, sem depender de outras entidades para operar.

Assim como Weiser, Ashton (1999) propôs o termo Internet das Coisas - *Internet of Things* (IoT), para tratar do uso na área de logística da identificação de objetos por meio de radiofrequência - *Radio frequency Identification* (RFID). O significado do termo rapidamente ampliou-se e passou a abranger a área de sensores e atuadores sem fio, de objetos conectáveis a redes que utilizam o protocolo de rede *Internet Protocol* (IP). A IoT (ATZORI; IERA; MORABITO, 2010), (RAVIDAS et al., 2019), (VIELITZ et al., 2019) considera a infraestrutura de rede com o objetivo de conectar e monitorar uma grande quantidade de dispositivos usando tecnologias modernas.

Em pesquisas relacionadas com a computação ubíqua, identifica-se o surgimento de oportunidades em diversas áreas, como, saúde e bem estar (VIANNA; BARBOSA, 2019), (VIANNA; BARBOSA; PITTOLI, 2017), (VIANNA; BARBOSA, 2014), educação (VIANNA; BARBOSA, 2019), (BARBOSA et al., 2011), (BARBOSA et al., 2014), jogos (BUZETO et al., 2013), (SEGATTO et al., 2008), comércio (BARBOSA et al., 2016), (FRANCO et al., 2011), acessibilidade (BARBOSA et al., 2018), (TAVARES et al., 2016), transportes (OLIVEIRA et al., 2015), agricultura (SOUZA et al., 2019), entre outros.

Junto à evolução das tecnologias, surge também a concorrência entre padrões e protocolos, onde cada desenvolvedor de software e fabricante de componentes eletrônicos propõe o seu padrão de funcionamento e comunicação, surgindo assim, uma grande heterogeneidade nos tipos e protocolos de comunicação (LIN et al., 2017).

Essa tese de doutorado propõe um modelo computacional que gerencia e adapta o comportamento operacional de objetos da IoT, sejam dispositivos físicos ou virtuais que realizam a comunicação via Internet, de forma a serem gerenciados de uma maneira centralizada em um

sistema com estrutura distribuída, que analisa o contexto passado (HONG, 2009), (WAGNER; BARBOSA; BARBOSA, 2014), (ROSA; BARSOSA; KICH; BRITO, 2015), (BARBOSA; TAVARES; CARDOSO; MOTA; MARTINI, 2018), avalia o contexto atual (SILVA et al., 2010), (ROSA et al., 2015), (WAGNER; BARBOSA; BARBOSA, 2014), tendo, como base, os dados históricos obtidos de diversos dispositivos e recursos computacionais, tratando a heterogeneidade de diferentes fontes de dados, inclusive, realizando a realocação de dispositivos móveis para locais menos densamente monitorados, de forma a detalhar eventos ocorridos ou que possam ocorrer.

Todo o processo de análise e inferência utilizado para a identificação de anormalidade, ou seja, acontecimentos fora do esperado, utiliza uma estrutura consolidada de dados, e sempre que ocorre um novo evento, ou seja, a recepção de novos dados, seja pela coleta ou aquisição, ou até mesmo a ausência de um evento esperado, o modelo avalia possíveis ações, podendo modificar o comportamento operacional dos objetos sensitivos envolvidos, formados pelos mais diversos tipos de dispositivos físicos ou virtuais. Eventos temporais também podem ser agendados ou disparados conforme regras previamente cadastradas. O modelo mantém dois históricos de contextos, onde um contém os dados recebidos ou adquiridos dos objetos sensitivos gerenciados, aliados à percepção, e outro com as alterações de comportamento operacional dos objetos, aliados à motivação que gerou esta alteração. Em um primeiro momento, a motivação é a ativação do sistema e a percepção inicial é a normalidade, no entanto, caso os valores processados estejam fora dos desvios de normalidade, que são calculados pelas médias históricas de períodos, isso tanto para valores numéricos quanto categorizados, a percepção pode ser especificada como anormal, sendo necessário ser analisada por um especialista, seja humano ou virtual. Os períodos são definidos de acordo com a necessidade de detalhamento e amostragem dos dados.

1.1 Motivação

A motivação para o desenvolvimento deste trabalho refere-se à possibilidade de integrar várias tecnologias, dispositivos e recursos da atualidade, gerando novas ferramentas de software, além de responder questões sobre desempenho e comportamento, quando aplicado a uma malha de sensoriamento de grande porte. Dentre as tecnologias que motivam este trabalho, destacam-se:

- Internet das Coisas - *Internet of Things*: representada por vários recursos que permitem a comunicação com a Internet (ALAN et al., 2013; PIRES, 2014; HANJO et al., 2017):
    - Dispositivos móveis: popularização de vários dispositivos móveis com acesso à Internet, como, *smartphones*, *drones*, *tablets*, veículos aéreos não tripulados (VANT), automóveis, dispositivos vestíveis entre outros;
    - Dispositivos estáticos: popularização de dispositivos eletrônicos com acessibilidade à Internet que permitem o sensoriamento de temperatura, umidade, iluminação, consumo de energia, radiação, movimento, vazamentos de gases, campos eletromagnéticos, análise corporal, localização, entre outros, além de recursos de software disponíveis;
    - Provedores de serviços: disponibilidade de vários provedores de serviços de dados, que fornecem informações de clima, radiação, mobilidade, imagens de satélite, entre outras funcionalidades;

- Grande conjunto de dados - *Bigdata*: disponibilidade de várias ferramentas que permitem armazenar e analisar grandes conjuntos de dados (GIRTELSCHMID, 2017; SANTOS, 2017; MAYER-SCHONB, 2014);
- Computação na Nuvem - *Cloud Computing*: possibilidade de utilizar um recurso de Internet que pode ser dinamicamente expandido (CHEE, 2013; MOLIRANI, 2018; TAURION, 2013; VERAS, 2015; VELTE &VELTE, 2013; NIST, 2018; BUTTA et al., 2011);
- Históricos de contextos: armazenamento de informações correspondente ao que aconteceu ou está acontecendo, considerando o evento, momento e local (HONG, 2009), (WAGNER; BARBOSA; BARBOSA, 2014), (ROSA; BARSOSA; KICH; BRITO, 2015), (BARBOSA; TAVARES; CARDOSO; MOTA; MARTINI, 2018);
- Computação Ubíqua: onde o usuário fornece e obtém dados sem saber que isso está ocorrendo (BARBOSA, 2015), (YAMIN, 2004), (YAMIN et al., 2005).

Além de integrar as tecnologias citadas, é necessário conhecer o estado da arte, para propor ideias atuais, como base no que já foi criado, bem como, propor ideias para continuar evoluindo. Dessa forma, a principal motivação para o desenvolvimento do AdaptThing é a integração e proposição de novas tecnologias, buscando por um nicho onde esta proposta possa ser considerada um diferencial, por consequência, poder auxiliar o meio em que for inserida.

1.2 Definição do problema e proposta

Esta tese propõe a definição de um modelo computacional chamado AdaptThing que utiliza a IoT como fonte fornecedora de dados, de forma a gerar históricos de contextos. Com base nos históricos de contextos, ter a capacidade de adaptar o comportamento operacional de dispositivos e recursos, de forma a obter dados mais precisos de eventos ocorridos, detalhando melhor a situação atual, e, possivelmente, poder prever situações futuras.

A adaptação de comportamento operacional de dispositivos e recursos corresponde aos parâmetros de configuração do dispositivo, como por exemplo, a frequência em que os dados são enviados para o sistema gerenciador, número de leituras que um dispositivo deve fazer em determinado intervalo de tempo, solicitação de alteração da posição geográfica de um dispositivo móvel, periodicidade de *sleep tim*e e *wake up*, intervalo de tempo em que determinado dispositivo deve verificar a conectividade com o sistema gerenciador, entre outras funcionalidades que dependem das características individuais de cada dispositivo.

Para isso, foi elaborado um modelo computacional que permite a intercomunicação de fontes heterogêneas de dados, possibilitando a comunicação entre si e seus elementos, enviando mensagens entre clientes e servidores de diversos tipos de redes, através de protocolos e padrões distintos, bem como, protocolos e padrões.

Assim, a presente Tese de Doutorado busca responder a seguinte questão de pesquisa:

A utilização de uma rede de dispositivos IoT com adaptação de comportamento operacional que altera os parâmetros de funcionamento de dispositivos, de acordo com a demanda e históricos de contextos, aliado a realocação de dispositivos móveis de forma dinâmica, permite coletar informações de forma mais efetiva, fornecendo assim informações mais precisas?

Propostas que buscam responder parcialmente essa questão podem ser encontradas na literatura. Os trabalhos relacionados são apresentados no capítulo 3. Para contribuir com o estado da arte e responder à questão de pesquisa é defendida a seguinte hipótese:

A utilização de dispositivos IoT com adaptação dinâmica de comportamento operacional, aliado a realocação geográfica de dispositivos móveis, pode ampliar a resolução de dados e fornecer uma visão mais ampla e detalhada, disponibilizando dados mais precisos para compreensão do que está acontecendo no local de estudo.

Baseado nessa hipótese foram definidas as seguintes questões complementares que visam guiar a investigação:

- Quais são os elementos, interações e requisitos existentes em um ambiente que utiliza uma rede de dispositivos da IoT?
- Quais os protocolos de comunicação que atendem a possibilidade de coleta e aquisição de dados, bem como, o gerenciamento das adaptações de comportamento operacional destes dispositivos da IoT?
- Como devem ser representados os dados de contexto para permitir a inferência e raciocínio através de máquinas de regras?
- Como inferir a chegada de um novo evento e então fazer projeção de ações futuras que deverão ser tomadas?
- Como deve ser projetada uma solução computacional que dê suporte aos requisitos de processamento dos dados dos dispositivos IoT?
- Como devem ser implementados os componentes de software que permitam a comunicação entre os dispositivos e o modelo computacional?
- Qual o comportamento e desempenho do modelo quando utilizado em uma aplicação real?

1.3 Objetivo geral

Este trabalho objetiva propor um modelo para gerenciamento de redes heterogêneas de dispositivos IoT, que além de coletar e obter os dados, possa também adaptar o comportamento operacional dos dispositivos, modificando seus parâmetros de funcionalidade, com base nos históricos de contextos.

O modelo deve agir frente a eventos, aumentando o detalhamento das informações, bem como, ter a capacidade de executar ações para novas situações potencialmente indesejadas, que venham a surgir nos ambientes em questão, com a capacidade de responder a eventos com base nos históricos de contextos, possibilitando assim a adaptação dinâmica e a realocação geográfica de dispositivos móveis.

1.4 Objetivos específicos

Os principais objetivos específicos dessa tese são:

- Criar um modelo computacional que permita realizar a coleta e obtenção de dados dos mais diversos dispositivos da IoT;
- Desenvolver um sistema para amparar o modelo computacional;
- Gerar históricos de contextos com dados organizados, otimizados e preprocessados de forma a agilizar a consulta futura;
- Criar uma máquina de regras que executa reconfiguração operacional e realocação geográfica de dispositivos da IoT móveis, dado a ocorrência de eventos;
- Aplicar e avaliar o funcionamento da arquitetura proposta pelo modelo computacional.

1.5 Metodologia

A metodologia para validação da hipótese, respeitando os objetivos propostos está dividida em cinco etapas:

1. Identificação do Estado da Arte, analisando o referencial teórico publicado por outros autores quanto ao uso de computação ubíqua para aplicações da *Internet of Things* (IoT), que permitam a adaptação e mobilidade geográfica de dispositivos e recursos;

2. Definição do modelo para suporte da hipótese;

3. Definição e codificação de uma plataforma computacional para suportar o modelo proposto;

4. Definição de experimentos a avaliação do modelo através da plataforma computacional proposta;

5. Avaliação dos resultados e definição da efetividade dos limites da plataforma desenvolvida.

A primeira etapa tem por objetivo compreender o estado da arte no que tange os problemas ao qual esta tese busca contribuir como parte da solução. Nesta etapa são avaliados modelos, algoritmos, técnicas, equipamentos e dispositivos computacionais, abordagens quanto ao tratamento e uso de dispositivos, aliados ao interesse da comunidade acadêmica na última década.

O interesse é identificado pelas publicações realizadas em mecanismos de disseminação de conhecimento das bases de dados acadêmicas através de revisão sistêmica.

A segunda etapa tem por objetivo definir um modelo capaz de suportar a hipótese, onde são analisadas e experimentados dispositivos da IoT, sensores, protocolos de comunicação e interoperabilidade, de forma a considerar os desafios aplicados na prática.

A terceira etapa é a concretização do modelo em forma de software, onde a arquitetura é especificada e codificada atendendo a requisitos de escalabilidade e extensibilidade, bem como são comparados e especificados os algoritmos necessários para suprir as necessidades do modelo.

A quarta etapa é a definição dos experimentos em conjunto com outras áreas de pesquisa afim, onde foram desenvolvidas aplicações em cenários diferentes.

A quinta etapa avaliou os resultados dos experimentos verificando o seu desempenho e a capacidade em aplicações de larga escala.

1.6 Estrutura da Tese

A tese está organizada em 6 capítulos, conforme descrito a seguir:

- No capítulo 2 é apresentada a fundamentação teórica, definindo termos e tecnologias da IoT, formas de armazenamento e análise de dados, protocolos e padrões de comunicação;
- O capítulo 3 apresenta o estudo dos trabalhos relacionados, que foram selecionados através de um mapeamento sistemático, mostrando o panorama atual de como estão sendo abordadas as tecnologias da IoT que possuem a capacidade de adaptar seu funcionamento com base em históricos de contextos;
- O capítulo 4 apresenta o modelo computacional AdaptThing, seu protocolo padronizado, abordando sua forma de funcionamento e as tecnologias e recursos utilizados;
- O capítulo 5 apresenta dois cenários onde foi aplicado o modelo computacional AdaptThing, sendo um no panorama de educação e outro no gerenciamento de plataformas climáticas;
- Por fim, o capítulo 6 apresenta as conclusões, descrevendo as contribuições da pesquisa, propondo trabalhos futuros.

2 REFERENCIAL TEÓRICO

Esse capítulo visa sistematizar os conceitos utilizados no desenvolvimento do trabalho, apresentando também, os trabalhos produzidos por outros autores. São abordados os fundamentos da computação ubíqua, demonstrando as possibilidades de integração com as tecnologias da Internet das Coisas - *Internet of Things* (IoT), que promovem a adaptação do comportamento operacional e a mobilidade de dispositivos. Também são abordadas técnicas e formas de comunicação, armazenamento e análise de dados.

2.1 Computação ubíqua

Na década de 1990, Mark Weiser publicou um artigo informando que os computadores pessoais poderiam desaparecer do olhar humano, passando a fazer parte de todos os objetos, de forma integrada e onipresente, assim surgindo o termo Computação Ubíqua (*Ubiquitous Computing* ou também chamado de Ubicomp). Weiser (1991) comparou este fenômeno ao desaparecimento dos motores, que encolheram até passarem a fazer parte dos objetos do dia-a-dia, porém sem estarem visíveis aos olhos humanos.

Através da computação ubíqua é possibilitado o surgimento de cenários que empregam o poder computacional em qualquer lugar ou momento, permitindo a acessibilidade com os mais variados dispositivos, permitindo a automatização e o aperfeiçoamento das soluções empregadas pelos usuários.

Conforme Barbosa (2015), a computação ubíqua veêm sendo explorada em diversos domínios de conhecimento, cobrindo áreas como saúde (*u-health*) (VIANNA; BARBOSA, 2019), (VIANNA; BARBOSA; PITTOLI, 2017), (VIANNA; BARBOSA, 2014), acessibilidade (*u-accessibility*), aprendizado (*u-learning*) (VIANNA; BARBOSA, 2019), (BARBOSA et al., 2011), (BARBOSA et al., 2014), comércio (*u-commerce*) (BARBOSA et al., 2016), (FRANCO et al., 2011), jogos (*u-games*) (BUZETO et al., 2013), (SEGATTO et al., 2008), transportes (OLIVEIRA et al., 2015) e agricultura (SOUZA et al., 2019), onde atualmente técnicas relacionadas a históricos de contextos e predição de contextos são tendências de pesquisa.

2.1.1 Definição de contexto

A primeira noção de contexto foi apresentada por Schilit, Adams e Want (1994), que descreveram contexto como sendo formado por locais, pessoas, objetos e as alterações que ocorrem nestes objetos. Uma das definições mais aceitas por pesquisadores é a sugerida por Dey et al. (1999):

> Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, um lugar ou um objeto que é considerado relevante para a interação entre o usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação.

A definição apresentada por Lieberman e Selker (2000), é tudo aquilo que afeta a computação, exceto as entradas e saídas explícitas. Para Chen e Kotz (2000), contexto é o conjunto de estados do ambiente e configurações, que determinam tanto o comportamento ou eventos de um aplicativo e são de interesse do usuário. Gray e Salber (2001) conceituam contexto como "propriedades que caracterizam um fenômeno, são sentidas e potencialmente relevantes para as tarefas executadas por uma aplicação e/ou os meios pelos quais essas tarefas são desempenhadas".

Para Yamin et al. (2003), contexto pode ser definido como toda informação relevante para a aplicação que pode ser obtida da infraestrutura computacional, cuja alteração em seu estado dispara um processo de adaptação na aplicação. Já Carvalho (2010) afirma que faz parte do contexto toda informação que pode descrever a situação das entidades e suas relações, envolvidas em uma ação que é considerada importante pelo sistema. Essas entidades são todos os conceitos abstratos e objetos físicos presentes na zona de observação do sistema num dado instante de observação.

O autor Abowd 2012, sugere o uso de cinco dimensões semânticas para definir o contexto, chamados de 5 "w". As cinco dimensões estão apresentadas na Tabela 1.

Tabela 1: As cinco dimensões semânticas definidas por Abowd (2012).

Dimensão	Descrição
Who	Quem realiza uma determinada atividade, quem pode alterar o contexto ou quem/o que pode ser notificado caso o contexto seja alterado.
Where	Onde o contexto está.
When	Quando, informação temporal para determinar quanto tempo de uma entidade que está dentro de um contexto.
What	O que o usuário está fazendo neste momento. Geralmente necessita de sensores para determinar qual é a atividade, o que torna isso uma tarefa complexa.
Why	Determinar o porquê o usuário está realizando determinada atividade, essa é umas das tarefas mais difíceis por envolver questões de inteligência artificial.

Fonte: Adaptado de Abowd (2012).

Segundo as afirmações dos autores citados na definição de contexto, pode-se constatar que sua abrangência engloba todas as informações instantâneas que caracterizam uma entidade. Na opinião do autor desta tese, contexto pode ser considerado uma "fotografia multidimensional do panorama em análise".

2.1.2 Sensibilidade ao contexto

Os sistemas sensíveis ao contexto exploram as informações que descrevem a situação do usuário com o objetivo de adaptar seus serviços e trazer benefícios à usabilidade e ao desempenho do sistema (CARVALHO et al., 2009).

Para Dey et al. (1999), um sistema é sensível ao contexto, caso use o contexto para proporcionar informações ou serviços ao usuário, de acordo as suas tarefas. Já para Rocha (2010), um sistema sensível ao contexto, é um sistema que utiliza o contexto (ou históricos de contextos) para garantir a realização da sua função, podendo para isso, alterar a sua interface de serviço ou modificar sua estrutura, de forma a gerar um novo comportamento.

A construção do suporte à sensibilidade ao contexto para as aplicações apresenta inúmeros desafios, dentre eles (VENECIAN, 2010):

- Determinação e caracterização dos elementos de contexto para uso na aplicação;
- Aquisição do contexto a partir de fontes heterogêneas, tais como dispositivos físicos, base de dados, agentes, serviços e aplicações;
- Representação de um modelo semântico formal;
- Processamento e interpretação das informações adquiridas;
- Disseminação do contexto para entidades interessadas de forma distribuída e em tempo hábil;
- Tratamento da qualidade da informação;
- Predição de contextos futuros.

Segundo Baldauf et al. (2007), o desenvolvimento e a implantação de sistemas sensíveis ao contexto, ainda é considerado uma operação complexa, que é resultante principalmente de três pontos críticos, que são reafirmados por Hussein, Bertin e Frey (2017):

- Heterogeneidade dos dispositivos móveis de acesso;
- Dificuldade em capturar e gerir as informações que descrevem o contexto;
- Complexidade relacionada à implementação dos mecanismos de adaptação e de filtragem de conteúdo baseado no contexto.

A sensibilidade ao contexto é uma característica intrínseca das aplicações da computação ubíqua. Quando informações de contexto computacional, contexto do usuário e contexto físico são agrupados dentro de um período de tempo, é obtido o histórico de contextos.

2.1.3 Sensores de Contexto

Segundo Regazzi, Perreira e Silva (2015), sensores são dispositivos que sofrem mudança de comportamento ou das propriedades quando sujeitos a uma ação de grandeza física ou química, podendo fornecer direta ou indiretamente um sinal indicativo da grandeza.

Um sensor é um dispositivo do tipo de transdutor, que converte uma forma de energia ou quantidade física em outra (BALBINOT, A.; BRUSAMARELLO, V, 2010), (SEIPPEL, 1983), (IEEE, 1997), (IEEE, 1999). Os transdutores dividem-se em dois subconjuntos:

- Sensores: fornecem informações de entradas em nosso sistema a partir do mundo externo;
- Atuadores: que executam ações de saída para o mundo externo.

Por esta definição, sensores são transdutores cuja ação é dar entradas do mundo externo para o sistema, já atuador é responsável pelas ações de saída do sistema para o mundo externo. O autor informa que os sensores possuem as seguintes características:

- Linearidade: é o grau de proporcionalidade entre o sinal gerado e a grandeza física. Quanto maior, mais fiel é a resposta do sensor ao estímulo. Os sensores mais usados são os mais lineares, conferindo maior precisão. Os sensores não lineares são usados em faixas limitadas, em que os desvios são aceitáveis, ou com adaptadores especiais, que corrigem o sinal;
- Faixa de atuação: é o intervalo de valores da grandeza em que pode ser usado o sensor, sem destruição ou imprecisão.

Além da definição de sensores, existem outros termos relacionados (BALBINOT, A.; BRUSAMARELLO, V, 2010), (SEIPPEL, 1983):

- Detetores: são dispositivos usados para "sentir a presença de alguma coisa" tal como calor, radiação ou outro fenômeno físico;
- Sinal: geralmente é qualquer quantidade que pode ser representada como uma função do tempo, tais como excitação e resposta, também denominadas de entrada e saídas;
    - Quanto ao tempo:
        i. Contínuo: são sinais nos quais o tempo é uma variável contínua;
        ii. Discreto: são sinais nos quais o tempo é uma variável discreta, normalmente assumindo valores periódicos.
    - Quanto ao tipo:
        i. Sinais analógicos: são sinais cuja amplitude não é restrita, podendo assumir quaisquer valores;
        ii. Sinais digitais: são sinais cuja amplitude é restrita a uma classe de valores.
- Servomecanismos: são sistemas de controle com realimentação nos quais as saídas são posições mecânicas, velocidades ou acelerações;
- Função de Transferência: Em teoria de controle a função de transferência refere-se à relação entre a saída e a entrada dos sistemas.

Segundo Wuet al. (2004), o termo “transdutor inteligente” foi mencionado pela primeira vez por KO e Fung em 1982, que implicou na integração de um sensor físico com o condicionamento de um sinal junto a uma função de filtro.

Atualmente, o termo “transdutor inteligente” foi alterado, correspondendo à integração de um sensor analógico ou digital, ou objeto atuador a uma unidade de processamento, que contém os circuitos de interface, um processador, memória e um controlador de rede em uma única unidade. O sensor inteligente transforma o sinal do sensor para uma representação digital padronizada, verifica e calibra o sinal transmitindo através de um protocolo de comunicação padronizado ou reconhecido entre o transmissor e o receptor.

A tecnologia de transdutor inteligente implica no desenvolvimento de redes de transdutores que permitem a monitoração, configuração *plug-and-play* e a comunicação de dados digitalizados do transdutor (WUET al., 2004). A utilização de sensores e transdutores inteligentes é apresentado na seção 2.2, onde são tratados os dispositivos da IoT.

Dessa forma, pode-se definir um Sensor de Contexto, como um sensor ou transdutor que atua em determinado contexto. Os contextos podem ser observados como dados do passado - histórico, presente - perfil, ou futuro - predição (BARBOSA, 2019).

2.1.4 Histórico de contextos

Informações de contextos passados, relacionados a uma entidade, são chamadas de históricos de contextos (HONG et al., 2009), (CIARAMELLA et al., 2010). Também é utilizado o termo trilha para definir os contextos passados (BARBOSA et al., 2018), (BARBOSA et al., 2016), (SILVA et al., 2010), (CAMBRUZZI; RIGO; BARBOSA, 2015), (DRIVER; CLARKE, 2008). Trabalhos como (HONG et al., 2009), (CIARAMELLA et al., 2010), (BAUR et al., 2010), (WAGNER; BARBOSA; BARBOSA, 2014), (SILVA et al., 2010), utilizam históricos de contextos no processo de tomada de decisão.

Os autores Rosa et al. (2015), Silva et al. (2010) e Cambruzzi, Rigo e Barbosa (2015), consideram as ações executadas anteriormente pelos usuários, tais como as atividades realizadas, aplicações utilizadas, conteúdos acessados e quaisquer outros dados possíveis, para melhorar a distribuição de conteúdo e serviços em ambientes sensíveis a contexto.

2.1.5 Gerenciamento de Perfis

Segundo dicionário Michaelis, (2018) um perfil é “um texto conciso em que se salientam os traços característicos de uma pessoa”. Também pode ser expresso como um conjunto de informações pessoais que definem um usuário. Atualmente é utilizado como um conjunto de informações de uma entidade.

Através de gerenciamento de perfis é possível identificar as características das entidades, a forma como está se comportando. Trabalhos de gerenciamento de perfis (WAGNER; BARBOSA;

BARBOSA, 2014), (FISCHER, 2001), (VIVIANI; BENNANI; EGYED-ZSIGMOND, 2010), permitem identificar o comportamento atual com base nas informações do passado.

2.1.6 Predição de Contextos

A ideia de predição refere-se ao fato de uma tentativa de identificar algo futuro. Segundo Rosa et al. (2015), com base no perfil das entidades e seus históricos de contextos, é possível tomar decisões.

Pesquisas trabalham a predição de estados futuros, como, estoques (BALLINGS et al., 2015), falhas em software (BALA; CHANA, 2015), localização (BURBEY; MARTIN, 2012), predição generalizada (PEJOVIC; MUSOLESI, 2015), (ROSA; BARBOSA; RIBEIRO, 2016), predição na computação (AMEYED; MIRAOUI; TADJ, 2015).

Algumas características de predição de estados futuros podem ser importantes de forma a "supor" que determinado dispositivo deveria ter "tal" comportamento, para criar uma instância inicial ou um estado homogêneo. Assim sendo, com base no que já ocorreu "tal" situação no passado, qual o percentual de chance de que ocorra o mesmo evento no futuro. No caso desta Tese, este evento que pode ocorrer no futuro é uma normalidade ou uma anormalidade inconveniente.

2.2 Internet das Coisas - *Internet of Things* (IoT)

A Internet das Coisas - *Internet of Things* (IoT), refere-se a um conjunto de redes que possibilitam a interconexão de objetos variados. Esse conceito foi introduzido pelo Auto-ID Labs, no *Massachusetts Institute of Technology* (MIT), que ainda hoje é um dos líderes em pesquisas científicas no campo de redes em *Radio Frequency Identification* (RFID) (MÖLLER, 2016). Na IoT, qualquer objeto ou entidade existente no mundo físico ou virtual que possua a capacidade de realizar comunicação pela Internet, é um objeto da IoT.

Outra definição de maior abrangência, apresenta o conceito de IoT como a presença pervasiva (omnipresente) de uma variedade de "coisas" em um ambiente físico, que através de conexões de comunicação, juntamente ao seu sistema de endereçamento único, são capazes de interagir e cooperar entre si, com a finalidade de criar aplicações e serviços, de forma a alcançar objetivos comuns. O conceito dos objetos em IoT, remete a objetos de uso diário, que possibilitam a leitura de algum dado relevante, reconhecimento, local em que está localizado e são endereçáveis através de um objeto que apresente capacidades de sensoriamento e que possam ser controlados pela internet independente de seu meio de comunicação. Dentre esses objetos, podem ser citados não somente os que possuem tecnologia embarcada, como veículos e equipamentos eletrônicos, mas também aqueles que não apresentam tecnologia eletrônica como é o caso de alimentos, peças de roupa, cadeiras, árvores, entre outros (PATEL; PATEL, 2016).

Uma definição mais antiga e muito citada em artigos é de Casagras (2009), onde ele define a IoT como uma infraestrutura de rede global, que interconecta fisicamente e virtualmente objetos, com o objetivo de explorar dados capturados e suas capacidades de comunicação. Essa infraestrutura é formada pela Internet e as redes de comunicação também necessita de identificação

única de objetos, sensores e capacidade de conexão, como base para o desenvolvimento independente de serviços e aplicações. Ela é caracterizada pelo alto grau de captura autônoma de dados, transferência de eventos de rede, conectividade e interoperabilidade.

A IoT refere-se à próxima geração de Internet (BAHGA; MADISETTI, 2014), que possuirá grande quantidade de nós representados por pequenos dispositivos ubíquos ou embutidos nos diversos ambientes, dotados de sensores, interconectados a servidores web, supercomputadores ou *clusters*. Tecnologias como as redes de sensores, a identificação única de objetos, a comunicação móvel, a localização em tempo real, a computação ubíqua, o protocolo IPv6 integram essa nova infraestrutura de computação e comunicação. Em Greengard (2015), o termo IoT refere-se não somente à interconexão entre objetos do mundo real, mas, à contextualização de informações referentes às coisas do mundo real. A aplicação do termo Coisas inclui objetos que existem em grandes quantidades e de diferentes tipos que possuem uma identificação única na Internet.

O resultado do processamento por uma aplicação ou serviço é uma atuação no ambiente em que estão presentes os objetos e/ou uma atuação sobre eles (CASAGRAS, 2009).

Conforme Casagras (2009), são propostas três classes de dispositivos para IoT:

- Objetos puramente passivos com identificação e dados fixos;
- Objetos dotados de moderado poder computacional e percepção de contexto, que por meio de sensores podem gerar mensagens e variar a informação associadas a eles de acordo com o tempo e lugar;
- Objetos que possuem conectividade em rede, sem a intervenção humana, possibilitando a emergência de inteligência nos sistemas de rede.

Esse cenário possibilita a agregação de inteligência em diversos cenários, como educação, cidades inteligentes, energia, saúde, transporte, entre outros, trazendo o desenvolvimento de novos produtos e serviços baseados em ubiquidade, de alto nível de inovação e impacto na sociedade (MÖLLER, 2016).

2.2.1 Dispositivos de sensoriamento

James e Cooper (2009) discutem os diferentes tipos de dados que estão no contexto de IoT. Os dados oriundos dos dispositivos da IoT podem ser discretos ou contínuos, gerados automaticamente ou através de alguma ação humana.

Segundo Smith (2012), um dispositivo de sensoriamento pode possuir outras tecnologias além de sua finalidade, como por exemplo, a conectividade com a Internet, protocolos de comunicação em nível de *hardware* além de processamento próprio, assumindo funcionalidade ativa ou passiva. Os dispositivos que possuem processamento próprio e conectividade com a Internet são chamados de *smart devices* ou objetos inteligentes, que podem ser físicos ou virtuais.

- Dispositivo físico da IoT: pode ser qualquer transdutor físico, podendo ser um microcontrolador ligado a sensores diversos (SMITH, 2012), como temperatura,

radiação, umidade, gases, campos magnéticos, entre outros, que possua capacidades de conexão com a Internet;

- Dispositivo de sensoriamento virtual: pode ser considerado um produto temporal, espacial ou temático que transforma um dado bruto, produzindo uma informação, ou seja, dados coletados, agregados ou processados a partir de um conjunto de sensores, que criam um valor para determinado objeto virtual. Da mesma forma, um atuador virtual pode ser um único ponto para a distribuição de comandos em um conjunto de atuadores com conectividade à Internet (SMITH, 2012).

2.2.2 Ambientes Inteligentes

Segundo Wacks (1992), os conceitos de ambientes inteligentes sempre estiveram presentes em livros, TVs e filmes de ficção científica, com muito apelo a imagens. Eles foram fonte de inspiração para a criação de novos artefatos. Com a chegada da Internet, surgiram novos paradigmas para o convívio social (PRUTHVI, M.; KARTHIKA, S.; BHALAJI, N, 2019).

Os dispositivos de sensoriamento já são usados para as mais diversas funcionalidades, no entanto, com a popularização de dispositivos que possuem acesso à Internet, foi possível o desenvolvimento de projetos que estejam de acordo com as necessidades dos tempos atuais (BOLZAMI, 2010). A utilização dos termos associados a dispositivos inteligentes, como "coisas inteligentes", pode variar de autor para autor, segundo Bolzani (2010):

- Taxonomia: quando referido ao contexto de automação de ambientes não industriais, residencial e residências inteligentes, o termo *Smart Ho*me é o mais utilizado, já o termo "ambientes inteligentes" é mais abrangente. Segundo Bolzani (2010), há um equívoco quanto aos termos, automação residencial, residências inteligentes, domótica, casa do futuro, ambientes inteligentes, sendo estes banalizados e empregados como sinônimos. Uma proposta de desambiguação é proposta por Bolzani (2010) na Tabela 2;
- Domótica: surgiu do latim, *domus*, que significa casa, é considerada ciência de engenharia das instalações em sistemas prediais. Caracteriza-se por ser uma ciência multidisciplinar com foco de estudo na relação do "homem com a casa". A inserção de usuários sem prévia aprendizagem em relação ao contexto do ambiente apresenta a necessidade de técnicas para gerenciamento da complexidade da interação entre o ambiente e o usuário;
- Automação Residencial: é um ramo derivado da automação predial com apelo especial a operações no âmbito doméstico, gerenciando equipamentos elétricos e eletrônicos sem a necessidade de intervenção humana, através de algum sistema de controle. Comumente utiliza-se de sensores para tomada de decisão, disparando ações ou mudando o ambiente. Uma residência inteligente faz o uso de Automação Residencial para controle do ambiente, sendo assim a residência não é denominada inteligente apenas por possuir automatização de dispositivos (BOLZANI, 2010);
- Termo Inteligente: de acordo com Russel e Norving (2011), a Inteligência Artificial é o estudo de agentes. Um agente é tudo o que pode ser considerado capaz de perceber

seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores. Sendo agentes aqueles que podem tomar decisões de acordo com uma determinada situação e que venha a proporcionar maior desempenho desta, agregando um grau de autonomia e podendo ter seu estado inicial modificado. Sendo assim, utilizando conceitos baseados na Inteligência Artificial, desde que um ambiente atenda a estes requisitos, acredita-se que pode ser denominado inteligente (BOLZANI, 2010);

- Ambientes Inteligentes: segundo Del Rio et al. (2018), "o conceito de um ambiente inteligente, idealizado há várias décadas, tornou-se uma realidade na qual os computadores são indispensáveis e fundamentais, integrados ao dia a dia da pessoa comum, por vezes, de forma imperceptível". São definidos como requisitos mínimos, a possibilidade de proporcionar a maior comodidade aos seus habitantes, promover maior segurança, promover a execução de tarefas automaticamente, otimizar o uso de recursos naturais e não causar transtornos aos usuários quanto a detalhes tecnológicos de funcionamento. De acordo com *Programme Advisory Group* (ISTAG), a Inteligência de Ambientes deriva da convergência de três tecnologias chave, que são a "Computação Ubíqua", "Comunicação Ubíqua" e "Interface Amigável" com o usuário.

Tabela 2: Desambiguação do uso de Ambientes Inteligentes (BOLZAMI, 2010).

Ambiente	Características	Elementos
Eletrificado	Controle manual e local de iluminação e cargas.	Infraestrutura de energia elétrica. Eletrodomésticos.
Automatizado	Controle automático de iluminação e cargas (intra residencial).	Infraestrutura de redes doméstica. Dispositivos de comando remoto. Espaços físicos conectados.
Comandada	Reação. Controle remoto externo. Troca de dados remota.	Rede de acesso. Gateways. Dispositivos externos de controle. Pessoas conectadas. Acesso a serviços e informações. Espaço social, tecnológico e físico conectados.
Inteligente	Reação. Planejamento. Controle remoto externo. Troca de dados remota. Conhecimento. Eventos. Dispositivos autoconfiguráveis e autônomos.	Dispositivos que podem substituir o ser humano na execução de tarefas.

Fonte: Adaptado de Bolzani (2010).

A proposta para desambiguação de termos apresentado na Tabela 2 auxilia na categorização de funcionalidades, porém sonde existe comunicação, segundo Möller (2016), faz parte do contexto da IoT.

2.3 Bigdata

Para Smith (2012), *bigdata* refere-se ao processamento e análise de repositórios de dados extremamente grandes e que não seriam possíveis de processar ou analisar com as ferramentas convencionais de análise de dados, pois requerem grande poder computacional para processar eficientemente grandes quantidades de dados em intervalos de tempos toleráveis.

Mayer-Schonb e Cukier (2014) mencionam que *bigdata* refere-se a grandes conjuntos de que apresentam dificuldades no armazenamento, pesquisa, visualização e análise. Nathan e Warren (2015) informam que *bigdata* é usado comercialmente para análise de grandes quantidades de dados, auxiliando na tomada de decisões, compreensão de comportamentos e perfis.

No trabalho de Song, Fisher, Wang e Cui (2018) é realizado um estudo imersivo sobre teorias e métodos utilizados em aplicações de *bigdata*, onde são apresentados vários recursos e aplicações, permeando por desempenho, teorias e formas de avaliação, métodos de análise, armazenamento de dados. Essa tecnologia envolve *Massive Parallel Processing Databases* (MPP), considerados como *grids* de mineração de dados, sistemas de arquivos distribuídos, plataformas de computação em nuvem, redes de comunicação e sistemas de armazenamento escaláveis. Neste mesmo trabalho é apresentado uma estimativa que a quantidade de dados chegue a 40 ZB em todo o mundo em 2020 (SONG, FISCHER, WANG, CUI, 2018).

Hadoop e MapReduce são os termos mais destacados quando referido o termo *bigdata* (DHAMECHA, DOBARIA, PATALIA, 2019). Hadoop é o termo usado para referir-se a uma família de projetos relacionados, que compõe a infraestrutura para computação distribuída e de larga escala de processamento, que usa o conceito de *bigdata*. Para White (2015), Hadoop é a implementação para MapReduce e sistema de arquivos distribuído mais utilizado e conhecido. O MapReduce é um modelo de processamento de dados distribuído e um ambiente de execução em *clusters* de larga escala. Este modelo divide o processamento em mapas e o divide em fases, cada fase é baseado em um par de chave/valor usado como entrada e saída para o processo. O programador especifica duas funções, o mapa e as funções de redução, para serem usadas na implementação e execução específica (WHITE, 2015).

O projeto Hadoop possui o *Hadoop Distributed File System* (HDFS), que é um sistema de arquivos projetado para armazenar arquivos extremamente grandes com um padrão de fluxo de acesso, executar sob *clusters* de máquinas de commodities ou plataformas de *hardware* comuns (WHITE, 2015). O MapReduce e o HDFS possuem uma API para o desenvolvimento e o uso de suas funcionalidades (DHAMECHA, DOBARIA, PATALIA, 2019), que abstraem a implementação de aplicações baseadas em *bigdata*.

2.4 Machine Learning

Machine Learning (ML), ou aprendizado de máquina, possuem a capacidade de aprenderem automaticamente a partir de conjuntos de dados. O aprendizado de máquina pode ocorrer de três

formas, com treinamento supervisionado, ou com treinamento não supervisionado e aprendizado por reforço (DHAMECHA, DOBARIA, PATALIA, 2019).

O aprendizado supervisionado ocorre quando são apresentados exemplos de entradas e saídas desejadas. O aprendizado não supervisionado ocorre quando não são fornecidos conjuntos de entradas versus saídas, assim sendo, o algoritmo deve estabelecer padrões dados um conjunto de dados. Já o aprendizado por reforço é um processo que interage com um ambiente dinâmico, desempenhando determinado objetivo, quando é fornecido um *feedback*, premiando ou punindo, conforme o passar do tempo. Para definir a utilização de determinado tipo de aprendizado, é necessário ter à disposição dos dados de entrada e, dependendo da situação, os dados de saída. Caso não tenha disponibilidade dos dados de saída, é necessário utilizar um aprendizado não supervisionado. Para a escolha de algoritmos deve-se considerar as seguintes questões:

- Representação: é um classificador deve ser representado em alguma linguagem formal que o computador possa manipular, ou seja, escolher o conjunto de classificadores que ele possivelmente pode aprender, onde este conjunto de classificadores é chamado de espaço de hipóteses do aprendiz. Se um classificador não está no espaço da hipótese, não pode ser aprendido.;
- Avaliação: é uma função de avaliação ou função objetivo. Isso é necessário para separar os bons classificadores dos que não apresentam bons resultados;
- Otimização: é um método para analisar entre os classificadores, os que apresentam pontuação mais alta, ou seja, os que apresentam melhores resultados.

No caso deste trabalho é utilizado o treinamento não supervisionado para obter os objetivos a serem alcançados, mais precisamente a classificação em categorias, e o treinamento supervisionado para treinar árvores de decisão para confrontar e analisar os novos dados, apresentado no capítulo 5, pág. 105, onde são apresentadas, onde foram utilizados os algoritmos K-Means e Árvores de decisão (ONAL, SEZER, OZBAYOGLU, DOGDU, 2017).

O objetivo do algoritmo K-Means é fornecer a classificação de dados, de acordo com as características fornecidas por eles. Esta classificação é baseada na análise e comparações entre os valores numéricos, que vai fornecer a classificação automática, sem a necessidade de supervisão, ou seja, sem nenhuma pré-classificação existente reforço (DHAMECHA, DOBARIA, PATALIA, 2019). O algoritmo analisa todos os dados, identificando uma classe, no entanto, o usuário deve fornecer ao algoritmo a quantidade de classes que são desejadas. Para gerar as classes e classificar as ocorrências, o algoritmo faz a comparação entre cada valor de cada linha por meio da distância. Geralmente é utilizada a distância euclidiana para o cálculo. A forma de calcular a distância depende da quantidade de atributos fornecidos.

Após a realização do cálculo das distâncias, o algoritmo calcula centroides para cada uma das classes. Conforme o algoritmo vai iterando, o valor de cada centroide é refinado pela média dos valores de cada atributo de cada ocorrência que pertence a este centroide. Com isso, o algoritmo gera k centroides e coloca as ocorrências da tabela de acordo com sua distância dos centroides.

Árvores de decisão são modelos estatísticos que utilizam um treinamento supervisionado para a classificação e previsão de dados. Árvore de decisão é um mapa de possíveis resultados de uma série de escolhas relacionadas, que permite comparar possíveis ações com base em suas regras.

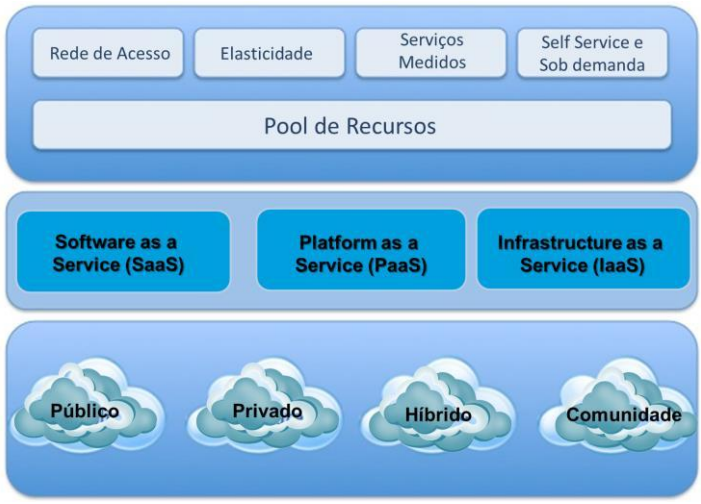
A árvore começa com um único nó, que se divide em possíveis resultados, que pode ter ramificações de acordo com possibilidades de resultados. Cada um desses resultados leva a nós adicionais, que se ramificam em outras possibilidades, criando-se uma forma de árvore (DHAMECHA, DOBARIA, PATALIA, 2019).

2.5 Computação nas nuvens - Cloud computing

A Computação em Nuvem é um modelo de computação no qual o usuário pode acessar uma grande variedade de aplicações e serviços em diferentes regiões, independente de plataforma, sem a necessidade de investimento em *hardware*. É possível ter um servidor, aplicativo ou serviço, que é executado em qualquer parte do mundo, podendo ainda dimensionar estes recursos sob demanda abstraindo a infraestrutura de *hardware*, possibilitando a expansão da IoT (BISWAS, GIAFFREDA, 2014), (KAUR, SINGH and SINGH, 2018).

O órgão Nacional de Padrões Tecnológicos dos EUA - NIST (2011) elaborou um modelo para computação em nuvem baseado em três camadas e cinco características essenciais, que ainda hoje são considerados como essenciais, apresentado na Figura 1.

Figura 1: Três camadas definidas por NIST (2011).



Fonte: Adaptado de NIST, 2011.

Os modelos de computação em nuvem mais adotados são formados por três camadas. Neste modelo é definida a arquitetura oferecida pelos fornecedores de soluções em relação à computação na nuvem, conforme NIST (2011):

- Infraestrutura com um Serviço (IaaS): oferece para o usuário adquirir armazenamento, instalação e configuração de servidores, dispositivo de rede e demais recursos computacionais. Parte do pressuposto, que o usuário não necessite

adquirir *hardware* ou software para dispor todos esses recursos no Data Center (MOLINARI, 2018);

- Software com um Serviço (SaaS): fornece um serviço ou um aluguel em que o desenvolvedor seja capaz de testar e implantar aplicações sem o custo completo de infraestrutura em *hardware* para isso (MOLINARI, 2018). Este modelo oferece uma plataforma de software em conjunto com serviços integrados, permitindo configurações e customizações;
- Plataforma com um Serviço (PasS): o usuário tem acesso as aplicações, sem a necessidade de instalar nenhum software adicional no seu computador. O aplicativo é oferecido como um serviço por um fornecedor de nuvem, accessível a partir de navegador web. O cliente não tem controle sobre a infraestrutura da aplicação, fica a cargo do fornecedor de nuvem o controle administrativo, gerenciamento da rede, sistema operacional, armazenamento e servidores (MOLINARI, 2018). Taurion (2013) esclarece que, caso a aplicação não esteja sendo usada, não necessita ser paga.

Ainda, conforme a Figura 1, os cinco elementos para descrever as características essenciais no qual a computação em nuvem é estabelecida são (NIST, 2011):

- Acesso Universal a Rede (*custom*): este recurso tem que estar acessível pela Internet, através de qualquer equipamento com acessibilidade;
- Elasticidade (*elastic*): Essa função deve prover agilidade e elasticidade para liberar recursos, permitindo assim, a percepção de recursos ilimitados;
- Medição de Serviço (*billing*): O fornecedor de nuvem deve prover para seu cliente a contabilização de seus gastos, de acordo com os recursos que foram adquiridos. Pode ser considerado um serviço de bilhetagem em formato de relatório, comprovando recursos que foram consumidos naquele período;
- Serviços sob Demanda (*self-service*): o consumidor pode, unilateralmente, requerer ou dispensar capacidades de computação, tais como o tempo do servidor, a capacidade de armazenamento, ou outros, conforme necessário e de forma automática. Tudo, sem necessidade de interação humana com o fornecedor de cada serviço. O fornecedor de nuvem deve transmitir a percepção que a nuvem possui recursos infinitos, a disposição de cliente, sem a necessidade de o cliente requisitar determinado recurso;
- Conjunto de Recursos (*pool*): o fornecedor de nuvem atende diversos clientes no mesmo espaço de tempo, compreendendo que esses recursos físicos ou virtuais são oferecidos de forma dinâmica. O pool deve compreender uma maneira transparente para o cliente final, dos recursos que foram contratados, como máquina virtual, armazenamento, memória, processador, rede e conexão, que são exclusivos para suas atividades.

Os modelos de nuvem são definidos a partir dos serviços que serão utilizados, segundo Molinari (2018), os mais significativos são quatro tipos: nuvem pública, privada, híbrida e comunitária. Conforme Molinari (2018):

- Nuvem Pública: possui estrutura é pública, mas possui mecanismo de controle para que a informação seja gerenciada. O usuário pode acessar a informação de qualquer lugar, contudo precisa saber a localização na informação. Este tipo de serviço não é cobrado;
- Nuvem Privada: é provida por fornecedor de serviço de nuvem ou alugada para um único cliente, sendo exclusivamente gerenciada por esse cliente. Não possui distinção geográfica, isto é, pode ser remota ou local. Possui o controle de acesso bastante restrito por questões de segurança, sendo protegida por *firewall*. Este tipo de serviço é cobrado;
- Nuvem Híbrida: composta por no mínimo dois tipos de nuvens, porém mantendo algumas características originais, mesclando duas tecnologias, permitindo o acesso a estes recursos. Alguns serviços são gratuitos até atingirem certos limites, após isso existe tarifação;
- Nuvem Comunitária: este tipo de nuvem é provida de uma nuvem pública ou privada, compartilha sua infraestrutura com uma ou mais organizações que tem algum propósito em comum. O objetivo da nuvem comunitária é alcançar a meta ou propósito em comum, na grande maioria este modelo é administrado por uma das organizações.

Virtualização em computação é a criação de uma versão virtual de alguma "coisa", como um sistema operacional, um servidor, um dispositivo de armazenamento ou recurso de rede (KUROSE, 2006). A virtualização pode ser aplicada a servidores, armazenamento, aplicativos e redes, sendo uma maneira eficaz de reduzir as despesas de TI e, ao mesmo tempo, aumentar a eficiência e a agilidade para empresas de todos os portes (KUROSE, 2006). A virtualização de *hardware* permite a multiplicação de máquinas virtuais sobre uma máquina física, criando uma arquitetura que permita dimensionar recursos sob demanda (BUYYA, 2011), possibilitando a implementação da computação nas nuvens.

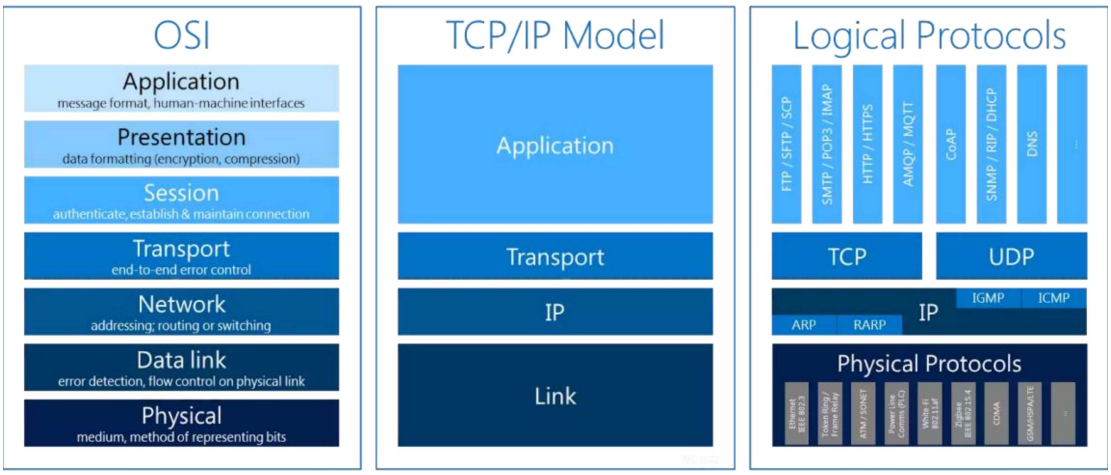
2.6 Protocolos de Comunicação e Interoperabilidade

Os protocolos de comunicação são parte importante na implantação de infraestruturas IoT, podendo contribuir para uma melhor utilização de recursos como os de processamento, memória, consumo energético e banda de comunicação dos dispositivos, considerando os fatores inerentes do universo IoT, tais como a heterogeneidade, mobilidade, intermitência de comunicação, extensibilidade, escalabilidade e capacidade de autoconfiguração (IoT-A, 2014).

Os protocolos de rede podem ser expressos de acordo com a sua funcionalidade, através da abstração em camadas. A abstração de rede mais conhecida é a do modelo *Open System Interconnection* (OSI), em que as funcionalidades de comunicação são categorizadas em sete

camadas. Um conjunto de protocolos relacionados, presentes em diferentes camadas, formam pilhas de protocolos, para prover a integração de funcionalidades, conforme a Figura 2.

Figura 2: Camadas do modelo OSI, Modelo da Internet, Modelo lógico



Fonte: Obtido de IOT-A (2014).

O Modelo lógico apresenta como fator crítico a interoperabilidade entre redes heterogêneas sendo suas camadas descritas como (IOT-A, 2014):

- Camada física: permanece conforme a definição OSI, de forma a não excluir qualquer tecnologia disponível, e possibilitar a integração de soluções emergentes. A convergência das diferentes soluções do Modelo de Comunicação é gerida na camada superior;
- Camada de ligação (enlace): lida com a heterogeneidade das tecnologias de rede disponíveis na IoT, para garantir a interoperabilidade e uma estrutura de segurança abrangente. Precisa tratar a diversidade tecnológica e ao mesmo tempo prover capacidades e interfaces uniformes para as camadas superiores;
- Camada de rede: mantém as características de sua correspondente no Modelo OSI. Mas, a fim de garantir gerenciamento, interoperabilidade e escalabilidade globais, a camada precisa fornecer um padrão de comunicação comum para cada solução de rede possível;
- Camada de identificação: é representada pelo Identificador de Entidade Virtual, o centro do primeiro ponto de convergência no Modelo de Comunicação. Aproveitando as interfaces uniformes fornecidas pela camada de ligação, a camada de identificação possibilita uma estrutura de resolução comum para a IoT. Os serviços de segurança, autenticação e de alta capacidade exploram essa camada para fornecer endereçamento uniforme aos diversos dispositivos e tecnologias das redes da IoT;
- Camada de transporte: provê funcionalidades de tradução, suporte a proxy e *gateways*, ajuste de parâmetros de configuração quando a comunicação atravessa ambientes de

rede diferentes. Construída acima das camadas de identificação e de rede, a camada de transporte fornece a peça final para alcançar um modelo de comunicação *Machine to Machine* global;

- Camada de dados: fica no topo do Modelo de Comunicação, sendo o ponto de entrada dos dados.

2.6.1 Identificação de dispositivos

Toda rede de dispositivos precisa de um identificador único, ou alguma forma que permita identificar um determinado elemento dentro de um, ou vários conjuntos. O uso de ID em redes baseadas em IP pode ser redundante, pois cada elemento da rede já possui um *Media Access Control* (MAC), porém quando utilizados vários elementos com tecnologias diferentes, pode ser necessário um identificador lógico.

Algumas tecnologias foram criadas para simplificar ou facilitar este acesso ao elemento, porém nem sempre estão disponíveis, como por exemplo o IPv6 que é um aprimoramento do IPv4. Mesmo que os dispositivos sejam ou não alcançáveis, é necessário saber de sua existência, para isso existem vários protocolos de descoberta de dispositivos e serviços.

A base de funcionamento da Internet é o endereçamento através de protocolos como o *Internet Protocol* (IP), que são combinações numéricas que estabelecem conexões entre computadores. IPv4 significa que é utilizando endereçamento IP na versão 4. Cada dispositivo que estiver online terá um código único em sua rede, para poder para enviar e receber dados de outros dispositivos que estiverem conectados (COMER, 2006).

Este identificador utiliza endereços de 32 bits, permitindo que aproximadamente 4,29 bilhões de IPs pelo mundo todo, o que atualmente pode ser considerado insuficiente pelo grande número de dispositivos conectados. Ainda hoje é utilizado endereços IPv4, porém gradativamente os equipamentos estão sendo substituídos por versões mais modernas que suportam IPv6. Como o IPv4 possui um número limitado de IPs, é utilizado uma técnica chamada de *Network Address Translation* (NAT), o qual permite que um IP válido (endereço de Internet), represente um IP não válido (endereço de Intranet), através de uma chave formada pelos IP origem e destino, porta origem e destino. Uma vez utilizado esta técnica, não é possível acessar diretamente um dispositivo da Intranet via Internet, pois o endereço sempre será ID do equipamento que possui acesso à Internet e não o dispositivo que está dentro da Intranet (KUROSE, 2007).

O IPv6 é uma evolução da versão 4 para a versão 6. Uma das principais características é a utilização de endereços de 64 bits, em vez de 32 bits (COMER, 2006). Esta modificação permite que o número de dispositivos conectados seja aumento exponencialmente, possibilitando a conexão de todos os dispositivos baseados em IP com um identificador válido (dado o volume de dispositivos existentes atualmente).

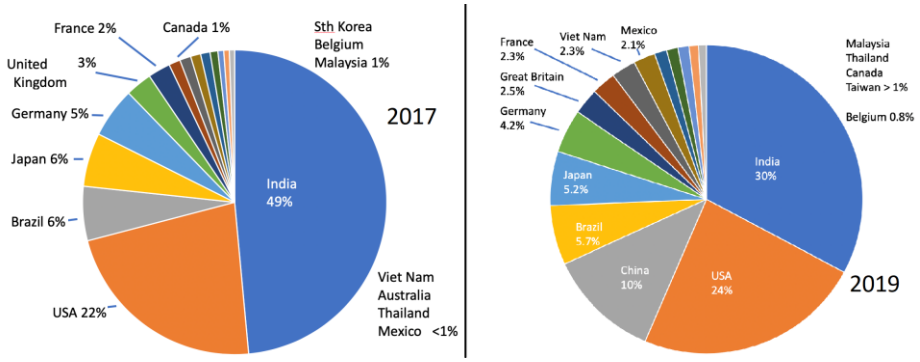
Segundo a RFC8200, atualiza o status do IPv6 para *Internet Standard*, conforme o site ipv6.br:

> No dia 14 de Julho de 2017, o grupo de trabalho "IPv6 Maintenance" (6man) do "Internet Engineering Task Force" (IETF) publicou a RFC8200 que atualiza o status do IPv6 para "Internet Standard" e torna obsoleta sua versão antiga (RFC2460).

Conforme o site regulador, a partir dessa RFC, o IPv6 passa do *status draft* para um padrão Internet oficial. Segundo o órgão regulador, a utilização de IPv6 em 2017, já alcançava 20 % do tráfego Internet. Em consulta no site ipv6.br em setembro de 2019, estima-se que quase 30 % do tráfego de Internet é IPv6.

Quanto ao consumo de endereços IPv6, no artigo "IPv6 in 2019: Where to from here?", publicado em 5 junho de 2019, apresenta o gráfico da Figura 3.

Figura 3: Espalhamento de endereços IPv6 pelo mundo.



Fonte: Adaptado de George Michaelson (2019).

Apesar desta tecnologia conseguir endereçar os dispositivos, ainda existem custos envolvidos, fazendo com que o IPv4 continue sendo utilizado, ou até mesmo que o IPv6 seja utilizado como identificador dentro de uma Intranet, sem acesso externo, ou que possua acesso unidirecional. Na Figura 3 é apresentado um teste em 02/09/2019, onde o provedor de Internet de fibra ótica não possui IPv6.

O IPv6 foi e é apresentado como uma solução para IoT, onde cada dispositivo e serviço possui um endereço específico, no entanto, essa ainda não é uma realidade, pois o IPv4 ainda é utilizado (MANIMOZHI S. and JAYANTHI J. G, 2018), sendo inclusive tema de pesquisa, de como adaptar formas de comunicação entre as tecnologias.

Uma solução baseada em IPv6 é o 6LoWPAN, que significa IPv6 over *Low power Wireless Personal Area Network*, que é o baixo consumo de energia em redes *wireless* pessoal (GEE K. M. Gee and COLLIER M, 2019). É uma adaptação do protocolo IPv6 com o protocolo 802.15.4 *Wireless Personal Area Network* (WPAN), nas três primeiras camadas (física, enlace e rede), do modelo OSI (Figura 2, página 40), para redes *wireless* de curto alcance. A RFC 4944, que foi publicada em setembro de 2007, descreve o formato do pacote. Ela é atualizada pela RFC 6282

que prevê melhorias na compressão dos cabeçalhos IPv6 e UDP. Este protocolo pode ser implementado em sistemas embarcados, simplificando a comunicação da IoT e/ou a *Internet of Everything* (IoE), termo utilizado por Shelby, 2009.

2.6.2 Descoberta de dispositivos

A descoberta de dispositivo de uma rede, é um protocolo de rede, que permite localizar outros computadores e dispositivos, bem como permitir que outros equipamentos da rede possam ver o dispositivo anunciante (DLNA, 2015). Existem vários protocolos, porém a maioria considera três estados de descoberta:

- Ativado: permite que o um dispositivo perceba outros dispositivos da rede, bem como permite que outros percebam a presença do dispositivo;
- Desativado: causa o impedimento de percepção;
- Personalizado: as configurações relacionadas à descoberta de rede estão habilitadas, mas com permissões limitadas.

A descoberta de rede solicita que o serviço *Cliente Domain Name Server* (DNS), Publicação de Recursos de Descoberta de Função, Descoberta *Simple Service Discovery Protocol* (SSDP) *Discovery* e *Host* de dispositivos *Universal Plug and Play* (UpnP) sejam iniciados. O processo de descoberta permite além de anunciar o dispositivo, também os seus recursos (DLNA, 2015).

2.7 Protocolos usados na IoT

A interoperabilidade de dados em dispositivos da IoT é considerada um dos maiores desafios da atualidade dado a heterogeneidade de padrões e protocolos de comunicação (COSTA, OLIVEIRA, MÓTA, 2018), onde cada desenvolvedor define a forma de comunicação, procurando formas de desenvolver aplicações de baixo custo. Conforme apresentado na seção 2.6o modelo OSI, buscou reduzir a incompatibilidade criando camadas de abstração. Nas próximas seções, são apresentadas algumas soluções e iniciativas adotados de forma a reduzir a heterogeneidade quanto à comunicação.

2.7.1 *Constrained Application Protocol* (CoAP)

Segundo Shelby et al. (2014), o CoAP é um protocolo que é executado na camada de aplicação de acordo com o modelo OSI (Figura 2, página 40). Foi criado para aplicações M2M com características similares ao protocolo *Hiper Text Markup Language* (HTTP). Ele realiza chamadas a GET, PUT, POST e DELETE, do modelo REST (BORMANN, 2016). Comporta-se como servidor e cliente durante a comunicação M2M.

O modelo de mensagem do CoAP é baseado na troca de mensagens, utilizando o protocolo UDP, que realiza a comunicação entre dois pontos. Um cliente requisita uma ação de um recurso, localizado em um servidor, que retorna uma resposta. O CoAP é dividido em duas camadas, uma que lida com as requisições e respostas, outra, para tratar as mensagens sendo transmitidas pelo UDP. Existem quatro possíveis tipos de mensagem (SHELBY et al., 2014):

- *Acknowledgement*, para sucesso;
- Reset, para rejeitar uma mensagem confirmável ou remover um observador;
- Confirmável, indica uma entrega confiável da mensagem;
- Não-Confirmável, não espera uma confirmação do envio.

Como está sobre o UDP, em que a entrega não é garantida, a transmissão pode exigir confirmação de entrega, por isso mensagens do tipo confirmável sempre retornam um ACK quando forem bem-sucedidas (ABDUL, 2015). Para a segurança, o CoAP utiliza do DTLS, uma variação do TLS para UDP, que transfere para a camada de transporte a manipulação de mecanismos de segurança. O protocolo provê quatro modos de segurança (GRANJAL, 2015):

- NoSec: nenhum mecanismo de segurança do DTLS é aplicado;
- PreSharedKey: utilizado com dispositivos que já são pré-programados com as chaves simétricas necessárias, onde cada chave possui uma lista de nós que podem comunicar-se;
- RawPublicKey: o dispositivo possui um par de chaves assimétricas sem a utilização de certificados, que é validado por um mecanismo *out-of-band*;
- Certificate: o protocolo faz o uso do DTLS com um certificado X.509, o dispositivo possui também uma lista de raízes confiáveis.

2.7.2 *Message Queuing Telemetry Transport Protocol* (MQTT)

Segundo Shubhangi (2016), o *Message Queuing Telemetry Transport Protocol* (MQTT) desenvolvido originalmente pela IBM, tornou-se um padrão aberto. Trata-se de um protocolo para o enfileiramento e transporte de mensagens, que utiliza o modelo *publish* e *subscribe*. Seus componentes principais são brokers, sessões, assinaturas (*subscriptions*) e assunto (*topic*).

O modelo *publish/subscribe* envolve a definição de um transmissor e diversos receptores (comunicadores/ouvintes), conectados em um *broker*, que organiza a troca de mensagens entre assinantes e publicantes. O assinante registra o interesse em determinado assunto e, assim que algum publicante disponibiliza conteúdo neste tópico, o *broker* direciona a mensagem para os assinantes registrados. A qualidade de serviço nesse processo é dividida em três categorias (SHUBHANGI, 2016):

- *At most once, Fire and Forget* (no máximo uma vez): utiliza do melhor esforço para enviar, caso não chegue em determinado envio, pode chegar no próximo;

- *At least once* (ao menos uma vez), garante que a mensagem chegue, mas pode ocorrer duplicatas;
- *Exactly once* (exatamente uma vez), garante que a mensagem chegará e não ocorrerá duplicatas.

O MQTT utiliza a porta TCP 8883 para realizar a comunicação, definida pela Internet *Assigned Numbers Authority* (IANA). O protocolo em si não oferece mecanismos de segurança, que normalmente são endereçados pela utilização de mecanismos como o TLS. Cabe ao implementador configurar no pacote de conexão os campos existentes para o nome de usuário e senha, que podem ser utilizados no processo de autenticação (SHUBHANGI, 2016). Assim como no CoAP, a segurança é endereçada por outros mecanismos externos, neste caso pode ser pelo TLS, que pode consumir muito poder computacional para os dispositivos que o utilizam.

2.7.3 *Extensible Messaging and Presence Protocol* (XMPP)

O protocolo faz a troca de pequenos trechos de mensagens. Suporta o modelo *client/server* e *publish/subscribe* para a troca de mensagens. No XMPP existem três principais funções para os integrantes da rede (RFC 6120):

- Cliente, realiza a conexão ao servidor via TCP, realiza o registro e faz o login;
- Servidor, é o serviço que permite a troca de mensagens;
- Gateway, é responsável pela interoperabilidade entre outras plataformas.

O XMPP utiliza da segurança provida pelo TLS. O documento provê também informações relativas ao que já está definido no padrão do protocolo (RFC 6120), em termos de segurança usa a obrigatoriedade de que clientes autentiquem-se em servidores, servidores autentiquem clientes e que servidores não devem autenticar outros servidores (SHUBHANGI, 2016).

2.7.4 *Universal Plug and Play* (UPnP)

As características de autoconfiguração e de descoberta automática de dispositivos inerentes ao UPnP, associado ao fato deste utilizar protocolos e padrões consolidados, tais como HTTP, SOAP e XML, é utilizado como um facilitador na gerência do crescente número de dispositivos conectados. Outro fator interessante é a quantidade significativa de dispositivos comerciais com a certificação Digital Living Network Alliance (DLNA) que tem como base o protocolo UPnP para comunicação (DLNA, 2015).

O protocolo permite a interoperabilidade entre diferentes dispositivos computacionais, em uma rede baseada no modelo *Plug and Play* (PnP), que simplifica a instalação e a configuração de dispositivos heterogêneos numa rede IP, sem a necessidade de configuração individualizada para

possibilitar os processos de obtenção de endereços IP, descoberta de dispositivos e chamada de serviços de rede (DLNA, 2015).

Sua arquitetura oferece uma conexão de rede denominada *Peer to Peer* (P_2P), onde cada um dos pontos ou nós da rede, funcionam tanto como cliente quanto como servidor. No caso de um dispositivo não possuir capacidade de suportar o padrão UPnP, um *gateway* UPnP pode ser usado para mapear os protocolos nos protocolos entendidos nativamente pelos dispositivos.

O UPnP permite o controle de dispositivos e o processamento de serviços de forma descentralizada, também permite a interoperabilidade entre pontos de controle. Além disso, a tecnologia UPnP inclui o conceito de descoberta e de descrição de dispositivos e serviços, permitindo assim que um dispositivo possa ser dinamicamente encontrado e compreendido, em termos de funcionalidade, por descrições XML (DLNA, 2015).

2.7.5 Ecossistema da IoT

De acordo com ITU-T (2012), o ecossistema de IoT é composto por uma variedade de atores de negócio. Os papéis podem ser descritos como:

- *Device provider*: O provedor de dispositivo é responsável por fornecer os dados brutos ou o conteúdo para o provedor de rede e provedor de aplicação, de acordo com a lógica de negócio;
- *Network provider*: O provedor de rede possui o papel central no ecossistema de IoT. Ele possui as seguintes funções principais:
    - Acesso e integração de recursos providos por outros provedores;
    - Suporte e controle das capacidades de infraestrutura de IoT;
    - Oferecer capacidades de IoT, incluindo capacidades de rede e exposição de recursos para outros provedores;
- *Platform provider*: O provedor de plataforma fornece capacidades de integração e interfaces abertas. Diferentes plataformas podem oferecer diferentes capacidades para provedores de aplicação. Capacidades de plataforma incluem tipicamente capacidades de integração, como por exemplo, armazenamento de dados, processamento de dados ou gerenciamento de dispositivos e suporte para diferentes tipos de aplicações de IoT;
- *Application provider*: O provedor de aplicação fornece e utiliza capacidades ou recursos providos por provedores de rede, de dispositivos e de plataformas, de modo a prover aplicações de IoT para aplicações clientes;
- *Application customer*: A aplicação cliente é o usuário das aplicações de IoT fornecidas pelos provedores de aplicação. Nesse caso, uma aplicação cliente pode representar múltiplas aplicações de usuário.

2.8 Considerações finais

Nesta seção foram apresentados os referenciais teóricos que embasam esta tese, apresentando conceitos e definições sobre histórico de contexto, IoT e protocolos e padrões de comunicação. No próximo capítulo é apresentado a metodologia adotada para seleção de trabalhos relacionados, bem como o estudo dos trabalhos mais relevantes.

3 TRABALHOS RELACIONADOS

Neste capítulo é apresentado o processo de pesquisa utilizado para mapear o estado da arte, dos assuntos relacionados com a tese de doutorado. Inicialmente foi realizado a busca em amplitude, para conhecer as principais pesquisas relacionadas, e uma vez mapeados os trabalhos foi realizada uma análise em profundidade, de forma a compreender a forma de atuação, bem como, poder comparar as principais características observadas no estudo.

Este capítulo está organizado em 7 seções, onde é apresentada a metodologia utilizada para a escolha dos trabalhos relacionados, os critérios de seleção e comparação. Esta pesquisa foi baseada nas diretrizes para o desenvolvimento de um mapeamento de estudo sistemático definido por Kitchenham (2004).

3.1 Planejamento do Mapeamento Sistemático

Segundo Kitchenham (2007), o mapeamento sistemático surgiu da necessidade de resumir toda informação existente sobre determinada área de estudo, com uma busca baseada em largura e não profundidade, de maneira completa e imparcial. Para tanto, é necessária a definição de um protocolo para guiar o desenvolvimento do mapeamento, que seja imparcial e passível de auditoria (TRAVASSOS, 2007). O protocolo de um mapeamento de estudo sistemático deve ser seguido rigorosamente de forma a qualquer usuário ter a possibilidade de reproduzir os mesmos resultados da pesquisa original, desde que seguido os mesmos passos. Conforme Kitchenham (2007), as atividades do protocolo a ser seguido, podem ser divididas em três etapas:

- Etapa de Planejamento:
    - Identificação da necessidade de uma revisão;
    - Definição do tema sobre o qual será executado o mapeamento;
    - Especificação das Questões de Pesquisa;
    - Desenvolvimento do protocolo de pesquisa;
    - Avaliação do protocolo de pesquisa.

- Etapa da Condução:
    - Identificação do tipo de pesquisa;
    - Seleção dos estudos primários;
    - Avaliação da qualidade da revisão;
    - Extração de dados e monitoramento;
    - Síntese dos dados.

- Etapa de Reporte:
    - Especificação dos mecanismos de reporte;
    - Formatação do Mapeamento Sistemático;
    - Avaliação final do Mapeamento Sistemático.

Ainda, segundo Kitchenham (2007), dentro da etapa de planejamento do mapeamento, a atividade mais importante é a criação das questões de pesquisa, que conduzem toda a metodologia do estudo, e é através delas que são identificados os estudos primários relacionados à pesquisa. Durante a extração de dados as questões de pesquisa atuam como fatores de inclusão e exclusão, visto que apenas os dados que estão relacionados com elas serão avaliados. Já a avaliação dos materiais selecionados, também é realizada tendo como base as questões de pesquisa.

3.2 Procedimento para o Mapeamento de Estudo Sistemático

Para orientar os objetivos do mapeamento de estudo sistemático proposto neste trabalho, foram elencadas as seguintes questões de pesquisa:

- QP1: Quais as principais aplicações desenvolvidas com base na Internet das Coisas (IoT)?
- QP2: Qual(is) a(s) relação (ões) entre a IoT e o armazenamento de grandes volumes de dados?
- QP3: Como a IoT é utilizada nas aplicações de computação ubíqua?
- QP4: Quais arquiteturas ou modelos estão sendo utilizados para coleta e armazenamento de grandes volumes de dados?
- QP5: Qual o tipo de modelagem está sendo utilizado para controlar a coleta, aquisição e o armazenamento de dados?
- QP6: Que tipo de aplicações está sendo desenvolvida de forma a realizar a adaptação das necessidades, conforme mudanças no contexto?

A definição do protocolo de busca, de acordo com Kitchenham (2007), é realizada após o desenvolvimento das questões de pesquisa. O próximo passo consiste em especificar as ferramentas de busca para a execução de consulta automatizada. As ferramentas de busca utilizadas para o desenvolvimento deste trabalho são apresentadas na Tabela 3. Os mecanismos de buscas foram definidos de acordo com as indicações de Dyba (2005), Kitchenham (2007), Travassos (2007) e Brereton (2007).

Tabela 3: Ferramentas de busca utilizadas.

Ferramenta de busca	Endereço na Internet
IEEE Xplorer	ieeexplore.ieee.org
ACM Digital Library	dl.acm.org
Science Direct	sciencedirect.com
Google Scholar	scholar.google.com.br
CiteSeerX	citeseer.uark.edu:8080
IET	digital-library.theiet.org
Scopus	scopus.com
Portal de Periódicos da Capes	periodicos.capes.gov.br
Wiley Online Library	onlinelibrary.wiley.com
Springer Link	link.springer.com

Fonte: do autor.

Foram excluídas algumas ferramentas de busca pela indisponibilidade de seus materiais, restrição de uso, erros e incompatibilidades ao aplicar estrutura de busca similar ou possuírem resultados muito inferiores em quantidade, em relação aos outros mecanismos de busca. A Tabela 4 apresenta os mecanismos de busca excluídos.

Tabela 4: Ferramentas excluídas no processo de busca.

Ferramenta de busca	Endereço na Internet
Science Direct	sciencedirect.com
CiteSeerX	citeseer.uark.edu:8080
IET	digital-library.theiet.org
Wiley Online Library	onlinelibrary.wiley.com

Fonte: do autor.

Suassuna (2011), sugere que seja incluída a busca por periódicos, além dos artigos, para compor os materiais a serem procurados. Uma vez definidos os mecanismos de busca manual e

automática, é preciso construir a "*String* de busca", de modo que represente o que será buscado na pesquisa conforme as questões levantadas no processo de planejamento (KITCHENHAM, 2007).

A "*String* de busca" é construída a partir da concatenação dos termos encontrados nas questões de pesquisa através de operadores booleanos do tipo "AND" e "OR", conforme a Tabela 5. Uma vez desenvolvida a "*String* de busca" é possível iniciar a pesquisa nas ferramentas de busca selecionadas, com alguns ajustes sintáticos, pois cada ferramenta possui uma estrutura própria, sendo necessárias algumas adequações conforme o mecanismo utilizado.

Tabela 5: *String* de busca definida para o mapeamento sistemático.

Funcionalidade	Termo principal	Termos alternativos
Tema base de consulta	Internet of Things	IoT
Forma de armazenamento	Bigdata	-
Contexto de atuação	Histor* Context	Ubiquitous, Context
Arquitetura ou Modelo da aplicação	Architecture, Framework, System, Model, Middleware	-
Tipo de aplicação	Distribuited, Centralized	-
Forma de adaptação	Adaptive, Dynamic	-

Fonte: do autor.

A *String* de busca que foi definida, de acordo com os termos considerados como fundamentais para a realização deste mapeamento é:

"("IoT" OR "Internet of Things") AND ("Big Data") AND ("Histor Context" OR "Ubiquitous" OR "Context") AND ("Architecture" OR "Framework" OR "System" OR "Model" OR "Middleware") AND ("Distributed" OR "Centralized") AND ("Adaptive" OR "Dynamic")"*

Algumas ferramentas de busca não suportam os parênteses externos como por exemplo, a ACM, já outras, o número de termos é limitado, como por exemplo a IEEExplore, que limita a busca a 15 termos. Foi acrescentado um * após a palavra "Histor", pelo motivo de existir flexões da palavra na literatura, como por exemplo: *historical*, *histories*, *history*, entre outras.

Para auxiliar a seleção dos estudos relevantes para o Estudo de Mapeamento Sistemático, é preciso que existam critérios de inclusão e exclusão (Suassuna, 2011). Os seguintes tipos de estudos foram incluídos nesta avaliação sistemática:

- Estudos completos publicados em revistas ou conferências, que tenham sido revisados, sobre a utilização da IoT que correspondam as questões de pesquisa;

- Estudos secundários, ou seja, que dependam de estudos primários, sob forma de trabalhos relacionados;
- Estudos teóricos com o objetivo de apresentar conceitos para o entendimento da área;
- Estudos experimentais relacionados à área;
- Somente trabalhos escritos em língua inglesa foram considerados.

De forma a limitar a abrangência do trabalho, os seguintes estudos foram excluídos:

- Estudos que não estejam claramente relacionados à área de IoT;
- Estudos que não respondem a nenhuma das questões de pesquisa;
- Artigos duplicados, que podem ser encontrados em mais de uma fonte da busca realizada;
- Artigos convidados, tutoriais, relatórios técnicos, teses, dissertações, relatórios e relatos de workshops ou eventos que estavam incompletos;
- Estudos não disponíveis para download ou sem acesso ao autor.

Com a conclusão dos passos da etapa de planejamento, cuja última atividade é a definição dos critérios de inclusão e exclusão dos trabalhos, foi iniciada a etapa de condução do mapeamento. Esta é caracterizada pela classificação da pesquisa, seleção e extração dos estudos relevantes para este trabalho.

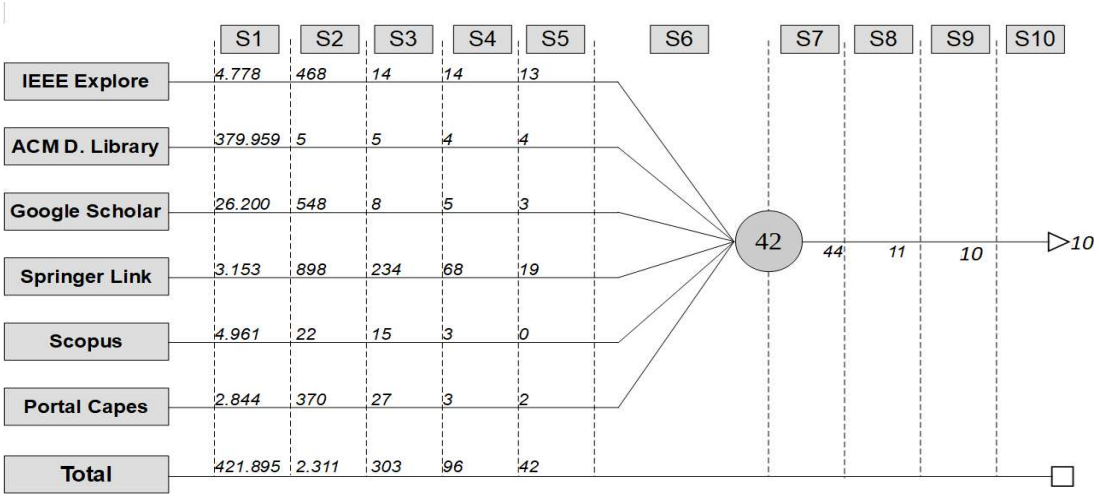
3.3 Classificação da pesquisa

Esta pesquisa é considerada exploratória empírica, pois conforme Cervo e Bervian (2002), a primeira etapa de um mapeamento é através de "Levantamento Bibliográfico", que tem por finalidade "levantar" todas as referências encontradas sobre um determinado tema, onde estas referências podem estar em qualquer formato, ou seja, livros, sites, revistas, vídeo, enfim, tudo que possa contribuir para um primeiro contato com o objeto de estudo investigado. Assim, observa-se que não existe nessa opção um critério detalhado e específico para a seleção da fonte material, basta tratar-se do tema investigado (CERVO e BERVIAN, 2002).

3.4 Resultados da Pesquisa

Conforme apresentado na seção 3.2, este trabalho realizou um Mapeamento de Estudo Sistemático, conforme apresentado na Figura 4. No lado esquerdo estão apresentados os mecanismos de busca utilizados neste trabalho, que estão definidos na Tabela 3 e restringidos na Tabela 4.

Figura 4: Mapeamento de Estudo Sistemático proposto.



Fonte: do autor.

Na Tabela 6 são apresentadas as legendas com as siglas utilizadas no Mapeamento de Estudo Sistemático apresentado na Figura 4.

Tabela 6: Siglas utilizadas no mapeamento sistemático.

Sigla	Função
S1	Busca Inicial com base na "String de busca", considerando todos metadados e conteúdos
S2	Delimitação por metadados
S3	Delimitação por título
S4	Remoção de materiais duplicados
S5	Remoção de impurezas
S6	Combinação de resultados
S7	Adição por heurísticas (outros trabalhos)
S8	Delimitação pelo conteúdo do texto (relevância)
S9	Trabalhos que foram analisados e documentados com maior profundidade (seção 3.5)
S10	Trabalhados que serão comparados (seção 3.6)

Fonte: do autor.

Conforme pode ser observado na Figura 4, após sucessivos filtros, a seleção de trabalhos foi restringida a 10, que serão apresentados na seção 3.5 e comparados na seção 3.6. O processo de eliminação e elencamento de materiais para base de estudo, procedeu-se da seguinte forma:

- IEEExplore: após filtros sucessivos foram definidos 13 trabalhos que foram analisados em profundidade;
- ACM Digital Library: na primeira busca, foi definido que as palavras chaves, poderiam estar em qualquer parte do texto, retornando um número muito grande de materiais (mais de 300.000 materiais), depois somente no *abstract*, por fim título. Após filtros sucessivos foram definidos 4 materiais que foram analisados em profundidade;
- Springer Link: pode ser considerado o repositório com mais materiais relevantes que passaram pelos filtros sucessivos, obtendo-se 19 materiais que foram analisados em profundidade, dentre estes documentos, todos atenderam aos requisitos do mapeamento de estudo sistemático;
- Google Scholar: foram eliminados os documentos em XML, trechos de livros sem acesso, materiais repetidos em outras bases, como por exemplo a Springer Link, obtendo-se assim, 3 materiais de estudo que serão analisados em profundidade;
- Scopus: após filtros sucessivos foram elencados 15 materiais, no entanto praticamente todos estavam contidos em Springer Link ou IEEExplore, após eliminar materiais repetidos, sobraram 3 materiais sem acessos do conteúdo na íntegra, totalizando 0 material nesta base;
- Portal da Capes: após realizado os filtros, foram apresentados resultados duplicados das mais diversas bases, no entanto houve o resultado de 2 materiais completos existentes em bases que não foram citadas como mecanismos de busca, como por exemplo Elsevier. Estes 2 materiais foram analisados em profundidade.

Alguns materiais também foram incluídos como materiais candidatos (S7), os quais foram analisados em conjunto com todos os materiais. Estes materiais não obedeceram a critérios específicos, no entanto, foram encontrados através de leituras em referências de outros trabalhos, até mesmo indicação de leitura por terceiros.

3.5 Seleção dos Estudos Relevantes

Nas próximas seções serão abordados os trabalhos selecionados em (S9), onde são documentados, apresentando também a percepção do objetivo principal dos trabalhos. Muitos dos trabalhos apresentados, não possuíam um processo de obtenção de resultados utilizáveis, a fim de comparação dos resultados, mas tinham a função de apresentar uma nova ideia, ou proposta de terminologias, realizando apenas um experimento básico de funcionamento.

3.5.1 Lightweight data interchange format

No trabalho intitulado *"A lightweight data interchange format for internet of things with applications in the PalCom middleware framework"* (NORDAHL, 2016), é apresentada uma notação textual para a comunicação de dispositivos da IoT, que suportam dados binários e textuais. Segundo o trabalho, PalCom é um *middleware framework* pervasivo, que foi implementado na linguagem Java, para criar redes dinâmicas entre os dispositivos da IoT. Este *framework* possui serviços de *discovery* (seção 2.6.2), e são gerenciados por um protocolo chamado de *service-interaction*.

O formato de troca de informações é baseado em JSON (NORDAHL, 2016), porém com manipulação de dados para poder realizar o tráfego de informações binárias, dado que JSON foi criado para manipular informações textuais. A inspiração para o desenvolvimento deste trabalho vem do formato de manipulação de *strings* da linguagem Fortran de 1966, que utiliza as constantes Hollerith, como apresentado no código da Figura 5.

Figura 5: Definição de *string* utilizando Fortran 77.

```
FORMAT(13H HELLO, WORLD)
```

Fonte: obtido de Nordhal (2016).

Neste formato, o programador informa ao compilador que existirão 13 caracteres, do tipo 'H', neste caso, textuais (*string*), existindo a vantagem de não precisar de delimitador. Partindo deste pressuposto, o formato dos dados sempre utilizará o padrão <Tamanho><Tipo><Dados>, onde o tamanho é a quantidade de bytes (inteiro), o tipo é representado por caracteres ASCII, já os dados dependem diretamente do tipo e tamanho.

Para poder trabalhar com objetos, *arrays* e outras estruturas, existem os tipos de dados "{" e "[", seguindo a mesma lógica já apresentada. De forma a regulamentar o formato de troca de dados, os autores definiram os seguintes formatos de dados, conforme a Tabela 7. Os tipos de dados são uma fusão entre os tipos primitivos da linguagem Java e Fortran, representados pelo JSON (NORDHAL, 2016).

Tabela 7: Descrição da sistemática de transformação e equivalência de dados.

Tipo	Exemplo	Equivalente com JSON
String	18sThis is a "quote".	"This is \"quite\"."
Integer	3i123	123
Float	10d123.456e78	123.456e78

Binary	8716y<binary data>	<base64 encoded string>
Char	cA	"A"
Byte	bA	"A"
Boolean	t ou f	true ou false
Null	n	null
Object	22{Key:<value>Key:<value>	{“Key”:<value>, “Key”:<value>}
Array	14[<value><value>	[<value>, <value>]

Fonte: obtido de Nordhal (2016).

Para validar o formato proposto, foram criados testes de performance, onde foi analisado a quantidade de *bytes* necessários para transmitir uma imagem (formato binário). Conforme relatos no trabalho, o uso neste formato permite uma diminuição de 25 % a 30 % na quantidade de *bytes*, já que não possui o *overhead* gerado durante o processo de transformação para base64, além do custo computacional para esta transformação, tornando este formato uma alternativa para transmissão de dados binários.

3.5.2 Rule-based recommendation system in IoT

No trabalho intitulado “Big data and rule-based recommendation system in Internet of Things” (HANJO et al., 2017), é proposto um sistema de recomendação utilizando *bigdata* (seção 2.3) em conjunto com uma máquina de regras para IoT, que preprocessa e analisa os padrões de dados em tempo “semi-real” e realiza recomendações em “tempo real”.

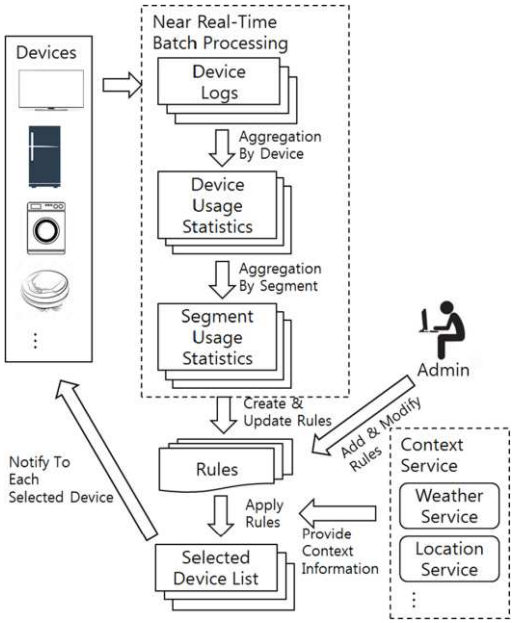
O trabalho utiliza “If This (triggering conditions), Than That (actions)” - IFTTT como trigger da máquina de regras. Na Figura 6 é apresentado um exemplo de como é disparado uma regra IFTTT. Neste trabalho é criado o termo “Contexto de Segmento”, onde um segmento é uma característica de lugares, por exemplo, locais úmidos, molhados, áridos, entre outras possibilidades.

Figura 6: Exemplo de regra IFTTT.

Fonte: Obtido de Hanjo et al. (2107).

O sistema permite que sejam coletados dados de dispositivos domésticos inteligentes, para fazer recomendações, contudo, pode ser adequado a outras implementações. Os dispositivos são identificados com base no endereço *Media Access Control* (MAC), endereço IPv6, e ainda um ID para cada usuário. Conforme Hanjo et al. (2017), em geral, os sistemas de recomendação utilizam regras pré-definidas ou as regras geradas, que analisam o uso e padrões de um usuário-alvo. Uma vez que os dados são coletados, os padrões de uso são analisados em cada contexto, onde são preprocessados, para resposta em tempo real. A ilustração do funcionamento da arquitetura proposta pelos autores pode ser visualizada na Figura 7.

Figura 7: Arquitetura de recomendação.



Fonte: Obtido de Hanjo et al. (2107).

Conforme pode ser observado na Figura 7, as informações meteorológicas providas por provedores de serviços externos, junto à localização do dispositivo, são necessárias para a máquina de regras ser executada. Também são necessários os dados da Tabela 8, para poder inferir uma recomendação com base na máquina de regras.

Tabela 8: Parâmetros de configuração e tipos de uso.

Campo	Descrição
segmentId	ID do segmento
areaCode	Código da área do segmento
basisDateType	Unidade de tempo

deviceType	Tipo do dispositivo
deviceUsageType	Tipo de uso
lastUpdateTime	Hora da criação

Fonte: Obtido de Hanjo et al. (2107).

O campo *deviceUsageType* da Tabela 8, representa o tipo de uso, que pode assumir os seguintes valores (utilizado para compreender a ideia):

- *lightUse*: comportamento de uso leve, por exemplo, limpando, comida rápida;
- *normalUse*: comportamento de uso normal, por exemplo, assistindo TV, usando máquina de lavar;
- *heavyUse*: comportamento de uso pesado, por exemplo, limpeza a vapor, comida especial;
- *clean*: limpando dispositivos, por exemplo, limpar forno;
- *normalCheck*: checar dispositivos, trocar partes, melhorar a performance;
- *safetyCheck*: inspeciona dispositivos, principalmente em termos de segurança.

Os dados dos dispositivos são enviados através de uma estrutura XML, podendo ser utilizado qualquer estrutura de linguagens de propósito geral, como por exemplo, Prolog, RDF, SWRL. Conforme Hanjo et al. (2017), é informado que não pode cobrir todos os cenários relacionados aos tipos de uso do dispositivo, como por exemplo, para adicionar um novo dispositivo em tempo de execução. Isso somente é possível, caso, seus atributos possam ser abstraídos em um nível superior.

O trabalho parte do pressuposto que os dispositivos da IoT informem os seus dados de utilização, em uma base de *bigdata*, o trabalho informa ainda, que podem ser utilizadas ferramentas como Jena, baseado em Web Semântica ou Pellet, baseado em Ontologias, para interpretar o conteúdo das regras (HANJO et al., 2017).

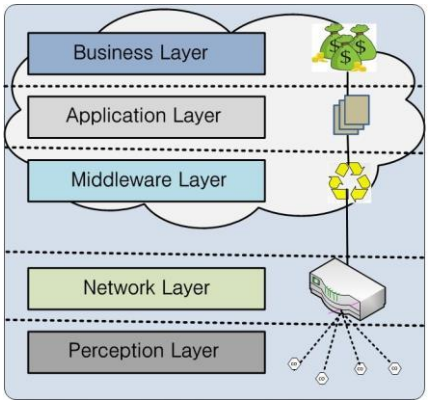
3.5.3 Fog Computing and Smart Gateway Based Communication for Cloud of Things

No trabalho "Fog Computing and Smart Gateway Based Communication for Cloud of Things" (AAZAM et al., 2014), é apresentado uma aplicação de *Cloud computing* (seção 2.5), onde ocorre a integração da IoT com a computação em nuvem, denominada *Cloud of Things* (CoT). São abordados os conceitos de *SmartGateway* e *FogComputing*.

Segundo os autores, esta integração da IoT com a *Cloud Computing* não é direta, pois existe a necessidade de redução de dados, já que a comunicação necessária, não apenas "invade" a rede principal, mas também o centro de dados na nuvem. Neste trabalho é apresentado uma arquitetura

IoT, que segundos os autores, é usualmente considerado em 3 camadas, mas que adicionaram mais duas conforme Figura 8. Neste caso, foram adicionadas as camadas *Business Layer* e *Middleware Layer*.

Figura 8: Arquitetura IoT com 5 camadas, proposta por Aazam et al. (2014).



Fonte: obtido de Aazam et al., 2014.

As cinco camadas são definidas por Aazam *et al.* (2014), como:

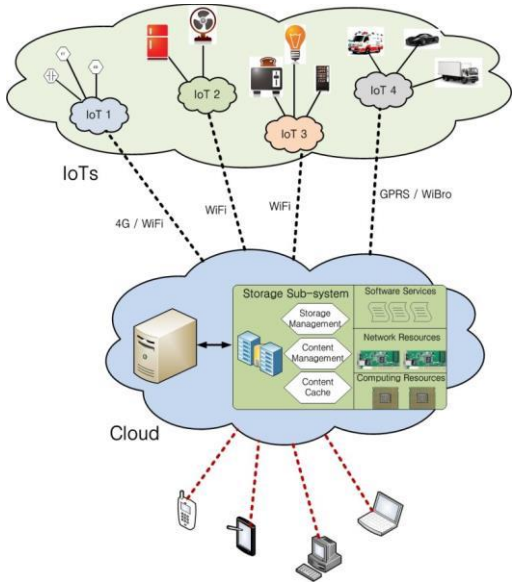
- *Perception Layer*: camada de percepção, é a camada mais baixa na arquitetura IoT. Seu objetivo é perceber os dados do ambiente. Toda a coleção de dados e a parte de detecção de dados é realizado nesta camada;
- *Network Layer*: camada de rede, responsável pela coleta dos dados percebidos pela camada inferior. A camada de rede é como o modelo OSI. A camada de rede só pode incluir um *gateway*, tendo uma interface conectada à rede do sensor e outra à Internet. Em alguns cenários, pode incluir um centro de gerenciamento de rede ou um centro de processamento de informações;
- *Middleware Layer*: camada que gerencia os serviços de armazenamento de dados. Também transmite processamento de informações e toma decisões de forma automática com base em resultados;
- *Application Layer*: camada que executa a apresentação final dos dados. A camada de aplicação recebe informações da *middleware* e fornece gerenciamento global da aplicação que apresenta as informações, com base nos dados processados pela camada anterior;
- *Business Layer*: camada de negócios, que atua no serviço ou o funcionamento do modelo. Segundo os autores, não é sobre ganhar dinheiro com o serviço prestado, no entanto, os esforços sem fins lucrativos e governamentais envolvidos na IoT, também podem ser parte disso. Os dados recebidos são moldados em serviços significativos.

Uma frase utilizada no trabalho de Aazam et al. (2014), é menção de ser citada é "… além disso, a informação é processada para fazer com que o conhecimento e os inúmeros meios de uso, o tornem sabedoria ...".

A proposta dos autores (CoT), conforme indicado no trabalho, ajuda a gerenciar os recursos da IoT, fornecendo meios mais econômicos e eficientes para produzir serviços. Os serviços a serem fornecidos estão na nuvem, isso proporciona um acesso ubíquo aos usuários, estendendo o escopo de uso dos serviços, facilitando o seu acesso, que por sua vez, ajuda a gerar mais dinheiro com os serviços.

A arquitetura proposta pelos autores pode ser observada na Figura 9, onde pode ser constatada uma visualização invertida, onde a camada mais baixa que é intitulada *perception*, está na parte superior da imagem.

Figura 9: Arquitetura CoT.



Fonte: obtido de Aazam et al. (2014).

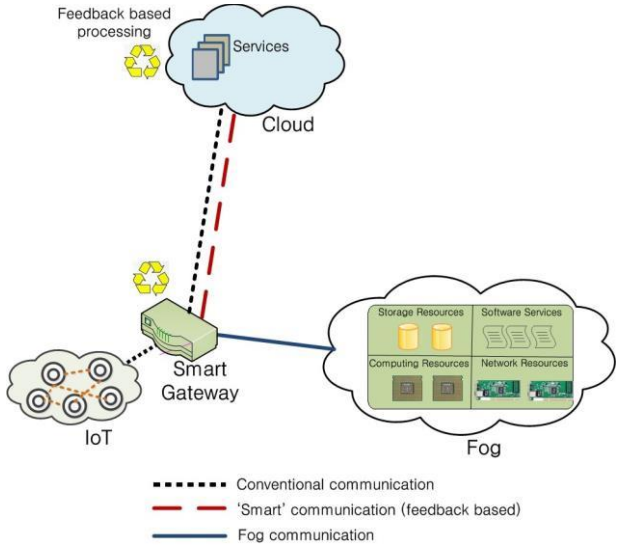
Conforme o trabalho apresentado por Aazam et al. (2014), são utilizadas algumas tecnologias e terminologias, apresentadas como (algumas já existentes, outras definidas pelos autores):

- *Cloud Computing*: a computação em nuvem leva a computação de *desktop* para toda a World Wide Web, sem que o usuário precise preocupar-se com manutenção e gerenciamento de todos os recursos;
- *Cloud of Things*: considera a computação web onipresente, segundo os autores, a IoT e computação em nuvem, quando trabalham juntas, criam novo paradigma, denominado *Cloud of Things* (CoT);

- *Smart Gateway*: é um dispositivo (proposta) de *gateway* que com base em *feedbacks* poderia decidir quanto um dispositivo deve ser interrompido de gerar e consumir recursos da rede, decidindo horários e tipos de dados que podem ser enviados;
- *Fog Computing*: refere-se ao fato de colocar recursos de rede perto das redes subjacentes. É uma rede entre redes subjacentes e a nuvem. A *Fog Computing* amplia o paradigma tradicional da Computação em nuvem para a borda da rede, permitindo a criação de aplicações ou serviços melhores.

No trabalho de Aazam et al. (2014), são referenciados outros trabalhos dos mesmos autores como embasamento teórico, onde são discutidos os conceitos e apresentado a arquitetura do *Smart Gateway* e *Fog Computing* como proposta.

Figura 10: Proposta de Smart Gateway e Fog Computing.



Fonte: obtido de Aazam et al. (2014).

O foco principal do trabalho não são os valores de desempenho, mas sim a proposta de S*mart Gateway*, no qual, conforme a Figura 10, pode omitir o tráfego da rede de acordo com o conhecimento ou regras de funcionamento.

3.5.4 BASIS: A Big Data Architecture for Smart Cities

No trabalho "*A Big Data Architecture for Smart Cities*" (COSTA C., et al., 2016), são apresentados os conceitos de *bigdata,* e como podem ser aplicados em *Smart Cities*. Segundo os autores, o termo Cidade Inteligente surge para conceituar a necessidade de compreender os cidadãos, a demanda de seus serviços e a sua relevância. As cidades inteligentes são conhecidas

por sua dinâmica humana, que faz o uso de dispositivos conectados através da IoT, gerando um vasto volume de dados.

O trabalho apresenta uma arquitetura de dados muito grande, aplicado em cidades inteligentes, intitulada BASIS, onde os atores apresentam a criação de múltiplas abstrações, desde o patamar mais conceitual até o mais tecnológico.

Conforme citado pelos autores, várias arquiteturas de *bigdata* para Cidades Inteligentes já foram propostas, exclusivamente focadas em sensores de dados e *Smart Grids* (GIRTELSCHMID et al., 2013; SUAKANTO et al., 2013). O modelo BASIS consiste em separar a complexidade em 3 camadas de abstração:

- Conceitual: encapsula vários componentes relevantes para um determinado papel, a partir da extração e análise de uma grande quantidade de dados;
- Tecnológica: ferramentas para mineração de dados e ferramentas para armazenamento não relacional;
- Infraestrutura: representação do nível físico, realiza a iteração do *hardware* com o mundo externo.

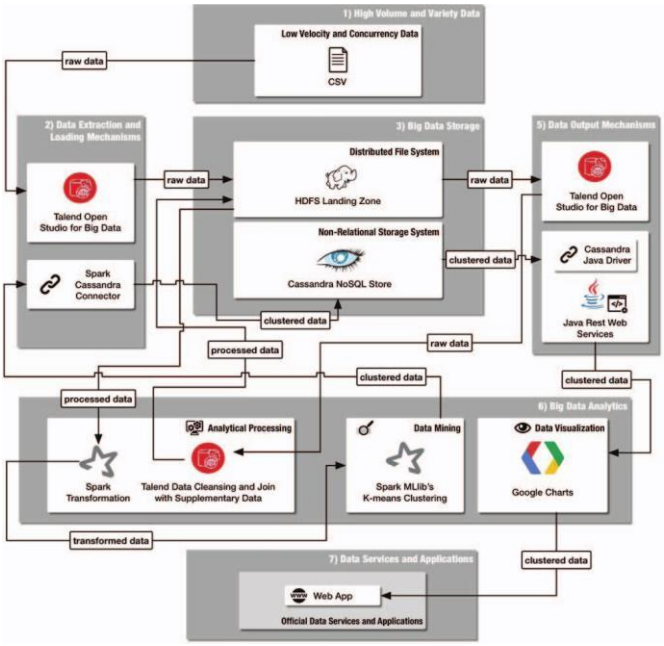
Conforme Costa et al. (2016), por mais eficiente e inovativo que a solução de *bigdata* pode ser, dados são o "combustível" da solução, pelo seu volume, variedade e velocidade, sendo fundamental existir os seguintes papéis para definição da infraestrutura:

- Especialista do Hadoop e armazenamento não relacional, que implementam e gerenciam as tecnologias associadas;
- Administrador que gerencia a performance do *hardware* e infraestrutura;
- Especialista de segurança e integridade.

Dessa forma, segundo os autores, nenhuma iniciativa de infraestrutura é suficientemente adequada, caso os dados que forem extraídos após a análise de *bigdata*, não possuírem sentido, assim, o conhecimento de cientistas de dados e analistas, é crucial para produzir informações com sentido.

Para demonstrar a aplicação do BASIS, foi criado uma instância (termo usado pelos desenvolvedores) utilizando várias tecnologias que envolvem *bigdata*, neste caso, Hadoop, Spark, Cassandra, Talend Open Studio for Big Data, entre outras. A aplicação desenvolvida foi utilizada para analisar atrasos de voos, com auxílio do algoritmo K-Means, ilustrado na Figura 11, onde é analisado o funcionamento da aplicação, como as diferentes ferramentas interagem, alcançando o objetivo proposto.

Figura 11: Instância tecnológica BASIS.



Fonte: obtido de Costa et al., 2016.

O artigo apresenta uma arquitetura *bigdata* para armazenar, processar, analisar e disponibilizar dados e serviços em um contexto de cidades inteligentes, propondo uma abordagem de múltiplas camadas, desde o mais conceitual até o mais tecnológico. Conforme apresentado por Costa et al. (2016), "as tecnologias de dados tradicionais podem parecer atraentes, devido à sua familiaridade, mas o armazenamento e o processamento distribuídos de volumes e dados variados, circulando em alta velocidade, sem problemas de desempenho, são necessários para conseguir a sustentabilidade dos serviços baseados em *Smart Cities*".

Os autores concluem que, o BASIS visa contribuir com abordagens e tecnologias bem estabelecidas, para entregar às cidades e aos seus cidadãos um estilo de vida mais inteligente baseado em grandes fluxos de dados.

3.5.5 Internet of the intelligent things

No trabalho "*Internet of intelligent things and robot as a service*" (YINONG, 2013), são discutidas as arquiteturas, interfaces e comportamentos de dispositivos inteligentes conectados ao ambiente de computação em nuvem.

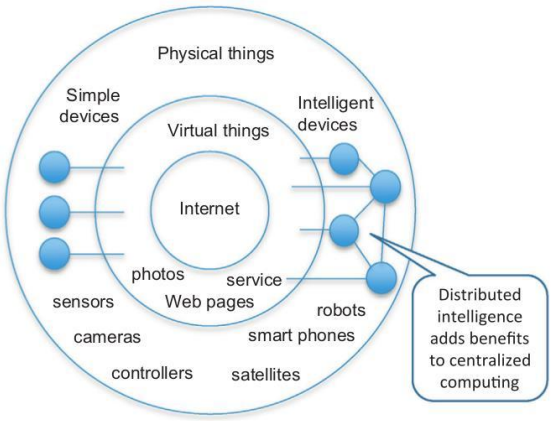
Neste trabalho, é informado que o desenvolvimento dos sistemas de computação e comunicação, passaram por um "ciclo espiral" de paradigmas de centralização e descentralização. Para elucidar este ciclo, é descrito que os primeiros sistemas informatizados eram os computadores *mainframe* com paradigma centralizado, que foram sendo movidos, para descentralizados, através de estações em rede, tornando-se mais confiáveis, extensíveis e econômicos.

Conforme Yinong (2013), os sistemas descentralizados possuem suas limitações e inconvenientes, no entanto com o uso de virtualização e computação em nuvem, cria-se um sistema centralizado, que para os usuários, aparenta estarem usando um sistema centralizado, onde os recursos de computação e comunicação não estão nos computadores clientes, mas em uma infraestrutura integrada acessível em qualquer lugar e a qualquer momento, já Internet das Coisas, consegue ampliar o conceito de computação em nuvem, indo além da computação e da comunicação para incluir tudo, particularmente, os dispositivos físicos.

Também é apresentado o estudo de caso *The Robot as a Service* (RaaS), que possui todas as características de ser autônomo, móvel, sensível e possuir ações, com o objetivo de ampliar mais o ambiente centralizado de computação em nuvem, em um sistema descentralizado, completando assim o "ciclo de desenvolvimento espiral". No projeto, foi criado uma base de conhecimento local de dispositivos inteligentes, que podem tomar decisões locais sem comunicarem-se com a nuvem. Essa base é formada pela coleta de informações de serviços físicos, móveis e de robôs autônomos e inteligentes, como provedores de serviços.

Na Figura 12 é apresentado a estrutura proposta por Yinong (2013). Nesta proposta, é informado que um robô é um exemplo perfeito de dispositivos físicos inteligentes. Geralmente é um sistema que sua aparência e movimentos, revelam que tem intenção ou gerência própria.

Figura 12: Arquitetura IoT.



Fonte: Adaptado de Yinong (2013).

O requisito básico para uma arquitetura ser considerada *Robot as a Service* (RaaS), deve possuir funções baseadas em *Service Oriented Architecture* (SOA), ou seja, *service provider*, *service broker*, *service client*, assim sendo:

- Uma unidade de nuvem RaaS é um *service provider*: cada unidade hospeda um repositório de serviços. Um desenvolvedor ou um cliente pode implantar novos serviços ou remover o serviço de um robô. Os serviços podem ser usados por esse robô, que também podem ser compartilhados com outros robôs;

- Uma nuvem RaaS contém um conjunto de aplicativos implantados: um desenvolvedor ou cliente pode compor uma nova aplicação ou função de acordo com os serviços disponíveis na unidade e fora da unidade;
- Uma unidade RaaS é um *service broker*: um cliente pode procurar os serviços e aplicativos disponíveis no diretório da unidade. Um cliente pode pesquisar e descobrir os aplicativos e serviços implantados no robô navegando na estrutura do diretório. Os serviços e aplicativos podem ser organizados em uma hierarquia de classes para facilitar a descoberta.

Existem dispositivos físicos simples e complexos. Os dispositivos simples não têm muita inteligência, e simplesmente enviam dados e/ou recebem dados da Internet. Este trabalho é baseado em dispositivos inteligentes com capacidade de comunicarem-se entre eles, podendo tomar certas decisões com base em informações locais e tomar ações autônomas e coordenadas, através de um robô como dispositivo.

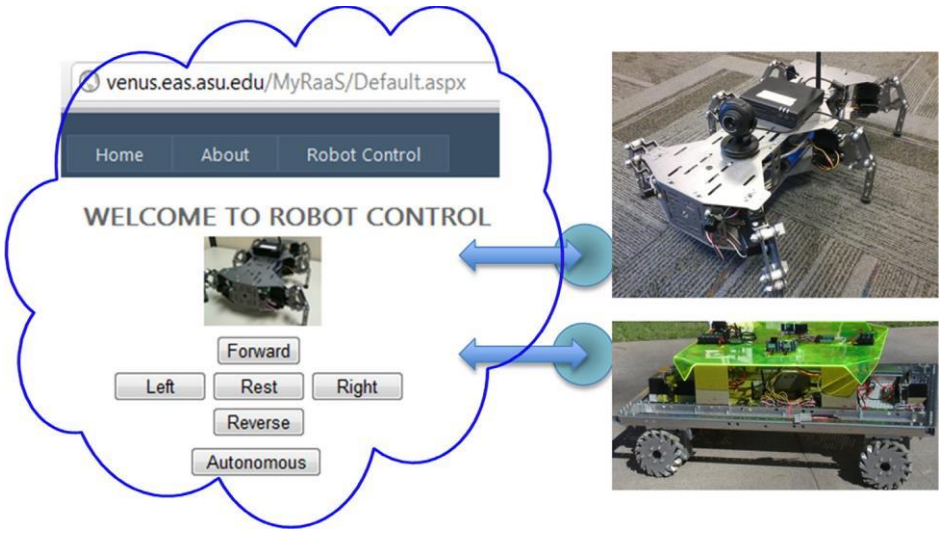
Um ponto importante deste trabalho é o *design* de tolerância a falhas, considerado com uma característica crítica do RaaS, pois o *single-point-of-failure,* ou seja, uma única instância de falha pode ocorrer entre o sistema de rede e o serviço físico, comprometendo sua funcionalidade.

Em um *design* típico de tolerância a falhas, as tarefas podem ser programadas para execução e comunicação redundantes, onde até mesmo os processos de decisão devem ser redundantes, obtendo-se um resultado, que deve ser elencado por um único processo de decisão e enviado para um único dispositivo. O design redundante proposta para as unidades RaaS, permite abordar o a falha entre a nuvem e Unidades RaaS, com os seguintes recursos:

- Possuem interfaces padrão para acessar a nuvem;
- Possuem interfaces padrão para o ambiente, assim como para os usuários finais, de forma que várias unidades RaaS possam ser compartilhados entre os usuários (vizinhos);
- Possui portas de conexão redundantes que podem conectar-se a múltiplos pontos de acesso da nuvem;
- As unidades RaaS vizinhas formam um conjunto de serviços de substituição para existir uma cópia de segurança de funcionalidades.

No caso deste trabalho, foram desenvolvidos os robôs conforme a Figura 13, utilizando-se dispositivos inteligentes, de capacidade de processamento superior e comunicação via Internet, aliado a dispositivos simples (sem inteligência embarcada, ou grande poder de processamento).

Figura 13: Robô utilizado no trabalho Robot as a Service.



Fonte: Obtido de Yinong (2013).

Para avaliação do trabalho, foram realizados um grande número de experimentos, analisando-se os dados de performance e o consumo de energia, que são coletados para diferentes processadores em diferentes taxas de *clock*. A configuração dos experimentos foram processadores Intel Core 2 Duo e Intel Atom N270 executando um número específico de *threads* em paralelo. Cada segmento representa uma instância de um serviço. Os experimentos foram realizados em serviços físicos que leem sensores e atuadores de controle, no entanto, com um número limitado de sensores (três sonares, três sensores de toque, um sensor de bússola e um sensor de luz) e atuadores (dois motores de acionamento e um servo para a rotação e *Webcam*).

O objetivo deste projeto, foi criar uma plataforma com processamento inteligente local, utilizando a Internet como meio de comunicação. Foram considerados problemas de comunicação com a Internet através da redundância de *hardware* e serviços.

3.5.6 Generic architecture for the future Internet of Things

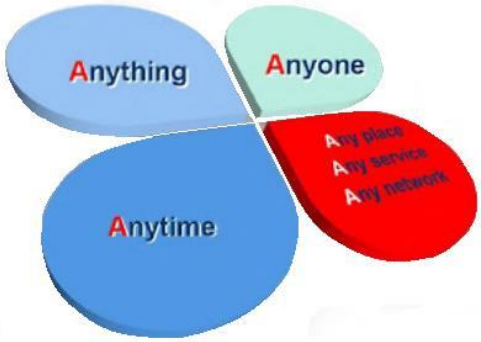
Para compreender o trabalho intitulado "*A global generic architecture for the future Internet of Things*" (WANG et al., 2017), é necessário visualizar uma proposta anterior, desenvolvida por Zheng *et al.* (2011), intitulado, "*Internet of Things - Global Technological and Societal Trends*".

Neste trabalho é apresentado uma estrutura chamada de 6A, que define as características futuras da IoT como *Anything, Anytime, Anyone, Any place, Any service, Any network*, apresentada na Figura 14. O 6A visa permitir pessoas e objetos serem conectados a qualquer hora, em qualquer lugar, com qualquer coisa, usando qualquer rede ou serviço.

Segundo Wang et al. (2017), a "IoT somente é limitada devido aos padrões de comunicação incompatíveis", ainda, informam que "foi aceito em todo mundo, que a maioria dos serviços para

a IoT utilizam serviços *web* para integrar serviços heterogêneos". Neste trabalho, é apresentado uma proposta para atender o que está previsto no modelo 6A, sendo independente de dispositivos específicos, plataformas, redes, domínios e aplicações, podendo minimizar a mensagem de transmissão no tamanho, para ser utilizado em dispositivos com capacidades mínimas de armazenamento e processamento, possibilitando menor tamanho físico, e redução de despesas gerais de rede.

Figura 14: 6A, Connectivity of the future Internet of Things.



Fonte: Adaptado de 6A, Zheng et al. (2011).

Segundo Zheng et al. (2011), uma arquitetura a ser proposta deve responder aos seguintes quesitos mínimos:

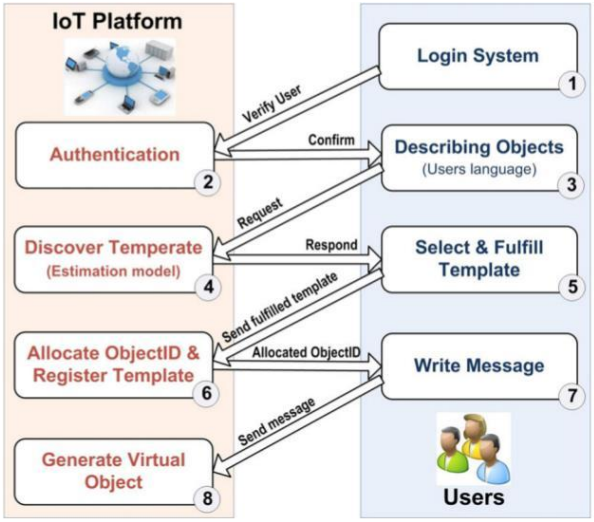
- Interoperabilidade: possibilidade de interagir e integrar de forma heterogênia com dispositivos, redes, sistemas e serviços;
- Implementar o princípio de SOA (arquitetura orientada a serviços, pode ser implementado com computação nas nuvens): permitir que implementações de terceiros possam ser utilizados, de forma independente do tipo de tecnologia ou produto;
- Serviço modularizado com baixo acoplamento: dado que os serviços e dispositivos da IoT são muito heterogêneos, muitas plataformas não podem ser reaproveitadas, custo que poderia ser reduzido, caso existisse uma interface padrão;
- Comunicação multiponto: um serviço precisa interagir de forma concorrente com múltiplos objetos, ou um objeto pode oferecer serviços para múltiplas entidades;
- Dinamicidade e reconfiguração em tempo real: dispositivos podem ser adicionados ou removidos de uma rede de forma dinâmica, a estrutura da topologia da rede pode ser alterada de forma dinâmica, alocando novos recursos de sistema para as demandas solicitadas;
- Iteração controlada e descentralizada: quando objetos físicos são movidos entre espaços, é difícil de manipular as incertezas e as iterações inesperadas entre os objetos, assim é necessário ter um controle maior sobre as iterações;

- Simplicidade de implementação: pode ser aprimorado através de um uso comum da IoT, deve ser *plug-in-play*, podendo conectar diversos dispositivos de maneira perfeitamente transparente, sem programação de forma virtual.

Conforme análise do material, é proposto que todos os dispositivos e serviços sejam virtualizados, de forma a existir um comportamento padrão, controlado por um *middleware,* que por sua vez, realize a comunicação de baixo nível com todos os componentes da IoT.

Uma vez definido que toda complexidade de comunicação com os dispositivos é realizada pelo *middleware*, a proposta é utilizar ontologias para definir o comportamento de determinado dispositivo em 8 passos, conforme Figura 15.

Figura 15: Processo de identificação de objeto IoT no sistema.



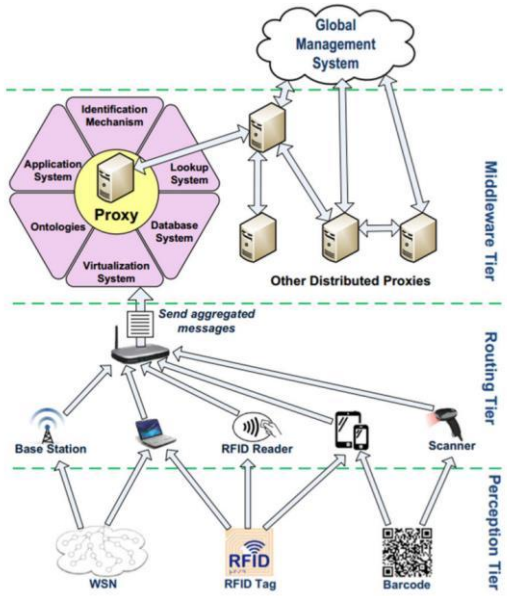
Fonte: Wang et al. (2017).

Nos passos seguintes, é apresentado a comunicação entre um usuário e o dispositivo:

- Realiza o login no dispositivo;
- Usa-se um método de autenticação para verificar o usuário;
- O usuário descreve o dispositivo (podendo ou não ter um *template*);
- A ontologia analisa as palavras chaves e busca por um *template* similar;
- Quando o usuário recebe as informações do objeto, ele pode selecionar um *template* que considera melhor atender os requisitos do objeto;
- Começa o processo de registro do objeto no sistema;
- Termina o processo de registro do objeto no sistema;
- O *middleware* recebe as mensagens do sensor.

A arquitetura proposta para o sistema é apresentada na Figura 16.

Figura 16: Arquitetura proposta por Wang et al. (2017).



Fonte: Fonte: Wang et al. (2017).

Para validar o modelo, foi criado uma simulação utilizando um "Arduino Uno" que transmite as informações via ZigBee para um computador PC. O *middleware* utilizado para o teste foi o LooCI. A aplicação foi composta de uma mensagem com três campos, *ObjetctID*, *Temperature Value* e *Sending Time.* Conforme os autores, os campos *Temperature Value* e *Sending time* são "propriedades dinâmicas".

No sistema proposto, as mensagens consistem em *ObjectID* e propriedades de com valores dinâmicos a serem coletados dos objetos. Outras descrições como esquema de dados, unidades de medidas e valores de propriedades estáticas, são separadas em partes pelo sensor de mensagens. Este método pode minimizar o tamanho das mensagens, de acordo com as capacidades dos dispositivos. Por fim, segundo os autores, o sistema proposto é restrito a iteração com tipos de objetos e serviços pré-definidos.

3.5.7 Integrating Physical Devices in IoT

No trabalho "*A Platform for Integrating Physical Devices in the Internet of Things*" (PIRES et al., 2014), apresentam uma plataforma *web* chamada de EcoDiF, onde a sigla remente a Ecossistema de Dispositivos Físicos.

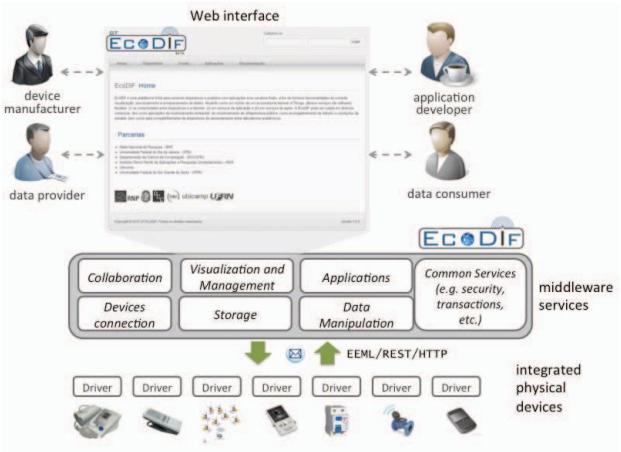
Conforme os autores, a IoT emergiu como um paradigma, em que “coisas” inteligentes colaboram ativamente entre si e com outros objetos, sejam físicos ou virtuais, disponíveis na *web*, para executar tarefas de alto nível. Assim, é necessário fornecer as abstrações para dispositivos físicos, serviços para aplicativos e usuários finais, bem como, meios para gerenciar a interoperabilidade de elementos heterogêneos, suportando o controle de dados em tempo real.

Segundo Pires et al. (2014), existem vários desafios a serem abordados para implementar o paradigma da IoT, entre eles estão a necessidade de uma arquitetura com capacidade de:

- Suportar de forma eficiente a heterogeneidade e dinâmica inerentes aos ambientes IoT;
- Fornecer as abstrações sobre dispositivos físicos e serviços para aplicativos e/ou usuários finais;
- Permitir a busca e a descoberta destes dispositivos (*discovery*);
- Permitir a conexão entre esses elementos através da rede;
- Monitor a localização e estado dos elementos conectados;
- Proporcionar e gerenciar a interoperabilidade entre tais elementos heterogêneos.

Foi destacado que é fundamental fornecer funcionalidades como controle, visualização, processamento e armazenamento de dados em tempo real, bem como, permitir a sua utilização em vários domínios diferentes de aplicação.

Figura 17: Arquitetura da plataforma EcoDiF.



Fonte: obtido de Pires et al. (2014).

A arquitetura proposta neste trabalho é apresentada na Figura 17, onde estão ilustrados o *middleware* e a comunicação com os dispositivos. A arquitetura é distribuída em camadas, onde cada camada possui suas funcionalidades, neste caso:

- *Devices Connection Module* - Módulo de Conexão de Dispositivos: visa simplificar a conexão de dispositivos físicos. Utiliza *drivers* personalizados que são desenvolvidos para cada tipo específico de plataforma de dispositivo, abstraindo a heterogeneidade. A comunicação é realizada através de solicitação HTTP do tipo PUT, são estruturados no formato EEML*drivers* podem ser:
    - Drivers ativos: obtêm os dados coletados pelos dispositivos fazendo solicitações periódicas para a API do dispositivo, mesmo que os valores de dados coletados permaneçam inalterados;
    - Drivers passivos: aguardam notificações da API do dispositivo, que são acionadas sempre que há mudanças nos valores de dados (ou conduzidos por eventos);
- *Visualization and Management Module* - O Módulo de Visualização e Gerenciamento: fornece abstrações sobre dispositivos físicos através de uma interface *web*, que permite aos usuários gerenciar dispositivos conectados ao EcoDiF. Permite monitorar o estado e a localização dos dispositivos, bem como, visualizar dados históricos armazenados na plataforma. Também é possível criar *triggers*, baseadas em eventos;
- *Collaboration Module* - O Módulo de Colaboração: utilizado para "facilitar" a colaboração entre os usuários do EcoDiF, permitindo-lhes realizar pesquisas de dispositivos e aplicativos a partir de seus respectivos metadados. No contexto da aplicação é compreendido que este módulo é utilizado para compartilhamento de informações entre usuários do sistema, acredita-se que o nome que melhor definiria a funcionalidade seria, *Sharing Module*;
- *Storage Module* - Módulo de Armazenamento: consiste em dois repositórios básicos:
    - Armazenamento de dados usando um banco de dados relacional;
    - Armazenamento de *scripts* de aplicativos em um sistema de arquivos;
- *Commons Service Module* - Módulo de Serviços Comuns: engloba os serviços de infraestrutura oferecidos pela plataforma, como a segurança, gerenciamento de ciclo de vida de aplicativos, transações, entre outras funcionalidades;
- *Applications Module* - Módulo de Aplicações: fornece um modelo e ambiente para programação e execução de *mashups,* que são pequenas aplicações *web*.
- Na proposta de Pires et al. (2014), as aplicações de *mashup* são implementadas como *scripts* escritos na linguagem EMML, que são executadas em uma *engine* chamada de Presto, que processa os *scripts*. Nesta proposta, o EcoDiF considera 4 tipos de usuários:
    - Fabricantes de dispositivos, que desenvolvem *drivers* para seus dispositivos para torná-los compatíveis com a API do EcoDiF;
    - Provedores de dados proprietários, que disponibilizam dados para o EcoDiF, através de *drivers* específicos para cada dispositivo;
    - Desenvolvedores de aplicativos, que criam aplicações ou serviços que podem fornecer dados ao EcoDiF;

- Consumidores de dados, que são usuários que interagem com EcoDiF para consultar informações disponíveis na plataforma.

Conforme Pires et al. (2014), constata-se que a proposta é aplicável para gerenciar vários tipos de dispositivos, tornando a plataforma uma ferramenta útil para monitoramento de atividades. Considerando a capacidade de integração de dispositivos sugeridas pelo modelo (sempre existem *drivers* disponíveis), é possível a criação de aplicativos baseados na integração de dados fornecidos por vários dispositivos heterogêneos e outros recursos da *web*.

3.5.8 A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing

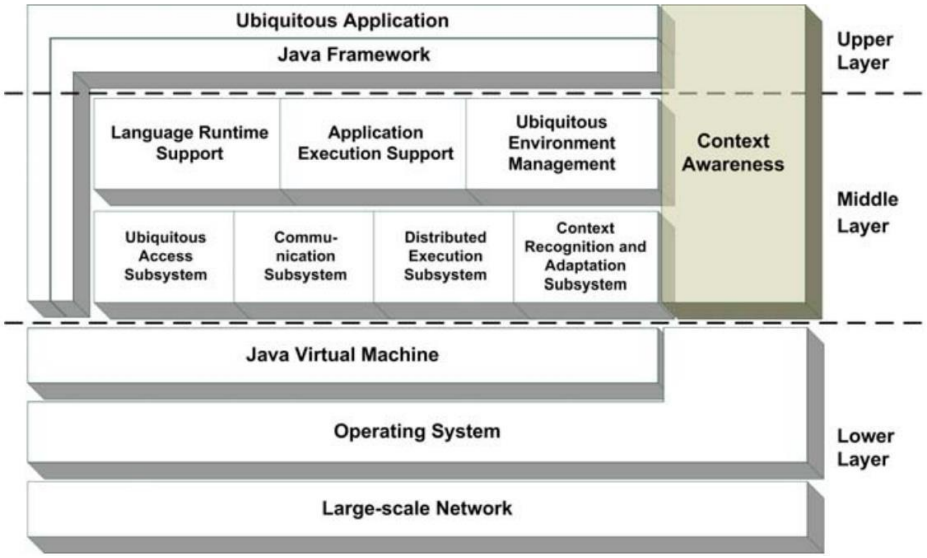
No trabalho "*A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing*" de Barbosa et al. (2014), apresentam a visão do *middleware* EXEHDA com um serviço para adaptação dinâmica.

Segundo os autores, o EXEHDA é um *middleware* adaptativo ao contexto e baseado em serviços, que visa criar e gerenciar um ambiente ubíquo, bem como, promover a execução de aplicações que expressam a semântica "follow-me" (siga-me). As aplicações são distribuídas, móveis e adaptativas ao contexto em que seu processamento ocorre, estando disponíveis a partir de qualquer lugar todo o tempo. O EXEHDA emprega uma estratégia colaborativa entre aplicação e ambiente de execução, através da qual é facultado ao programador individualizar políticas de adaptação para reger o comportamento de cada um dos componentes que constituem o software da aplicação (YAMIN, 2004).

No *middleware* EXEHDA, as condições de contexto são proativamente monitoradas, o suporte à execução deve permitir que, tanto a aplicação como ele próprio, utilizem as informações na gerência da adaptação. O processo de tradução dos dados de sensores para contextualizados é feito por algoritmo de estrutura de dados personalizado por tipo de aplicação.

Para atender a elevada flutuação na disponibilidade dos recursos, inerente à computação ubíqua, o EXEHDA é estruturado em um núcleo mínimo e em serviços carregados sob demanda. Os principais serviços fornecidos estão organizados em subsistemas que gerenciam a execução distribuída, a comunicação, o reconhecimento do contexto, a adaptação, o acesso ubíquo aos recursos e serviços, bem como a descoberta e o gerenciamento de recursos. Na Figura 18 é apresentado a arquitetura do middleware.

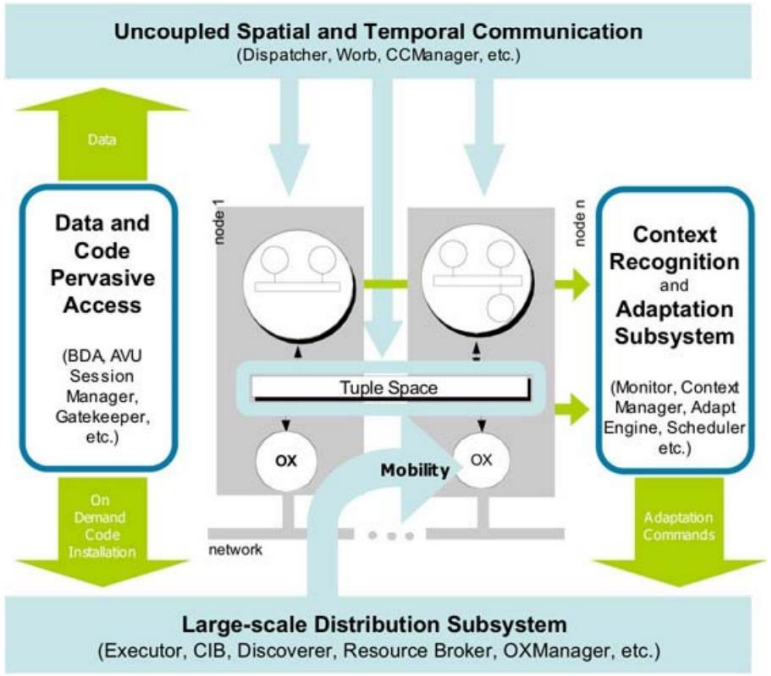
Figura 18: Arquitetura do middleware EXEHDA.



Fonte: obtido de Barbosa et al. (2014).

O EXEHDA é composto de vários serviços integrados, inclusive um serviço para adaptação dinâmica, chamado de *Dynamic Adaptation service* (DA service), conforme Figura 19.

Figura 19: EXEHDA Dynamic Adaptation Service.



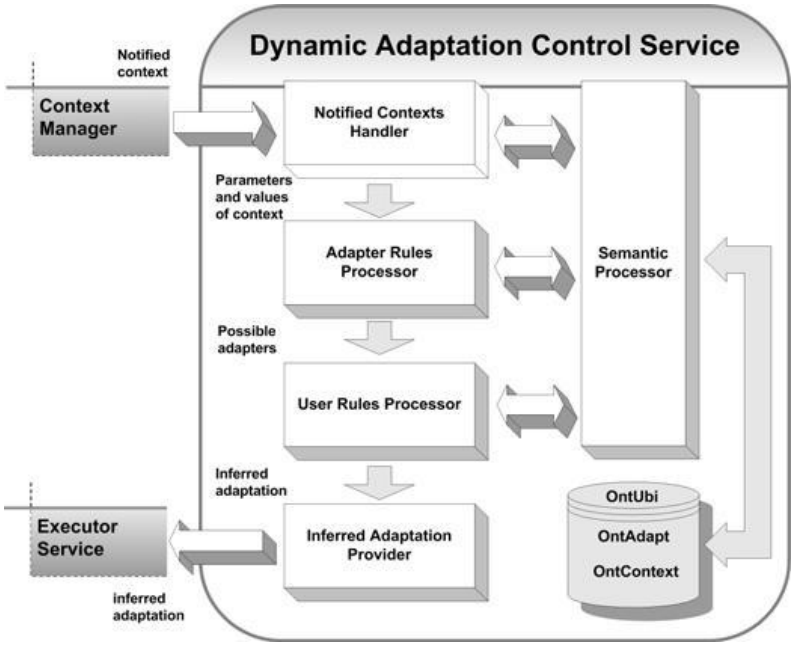
Fonte: obtido de Barbosa et al. (2014).

Os serviços são conceitualmente organizados em subsistemas (YAMIN et al., 2005b), com dados e códigos de acesso ubíquo, com reconhecimento de contexto e adaptação.

Sob o ponto de vista do *middleware*, existem 2 tipos de recursos físicos, que são nodos de processo e recursos especializados. Os nodos de processo são os que executam e gerenciam o *middleware*, já os recursos especializados são dispositivos externos, executados por bibliotecas específicas. Também existe o *discoverer service*, que busca por recursos especializados no ambiente e estudo, o próprio serviço disponibiliza toda a instância para usar determinado dispositivo.

Segundo Barbosa et al. (2014), para aumentar a flexibilidade para adaptações específicas, o *Dynamic Adaption Service* (DA Service) foi considerado muitos elementos de software para a manutenção, qualidade e execução, sempre que uma mudança no contexto ocorre. O formato proposto permite o desenvolvimento incremental, com inclusão de regras, parâmetros, políticas, restrições e ações. A funcionalidade do DA calcula a diferença entre os pesos dos parâmetros passados e os parâmetros obtidos do contexto, sendo que o peso corresponde a prioridade de adaptação. Dessa forma, o sistema pode garantir a todo momento que a adaptação de maior prioridade ocorra e não viole as restrições impostas pelo ambiente. A arquitetura proposta pelos autores é apresentada na Figura 20.

Figura 20: Dynamic Adaption Control Service.



Fonte: obtido de Barbosa et al. (2014).

Conforme a Figura 20, a arquitetura possui o recebimento dos dados do contexto, são preprocessados, passam por uma máquina de regras que dado um processo de inferência, podem executar ou não algum serviço.

A arquitetura utiliza um modelo semântico, neste trabalho chamado e OntUbi, que representa o ambiente gerenciado pelo EXEHDA. O modelo semântico é composto pelas seguintes ontologias:

- OntContext: situação do contexto, ela representa a coleção de contextos;
- OntAdapt: políticas de adaptação, regras parâmetros, operações, preferências, restrições e ações.

O objetivo deste trabalho foi apresentar o *middleware* EXEHDA e a forma proposta de adaptação, onde os autores consideram que o modelo semântico introduz aspectos relacionadas ao formalismo e expressividade, possibilitando relações dinâmicas e inferência entre as informações, permitindo um modelo com políticas de adaptação de alto nível. Entre outras características, os autores informam, que este modelo permite a manutenção, reutilização, padronização e compartilhamento de informações, através de um modelo ontológico entre os vários serviços oferecidos pelo *middleware*.

3.5.9 Modular Multicore Wireless Sensor Network

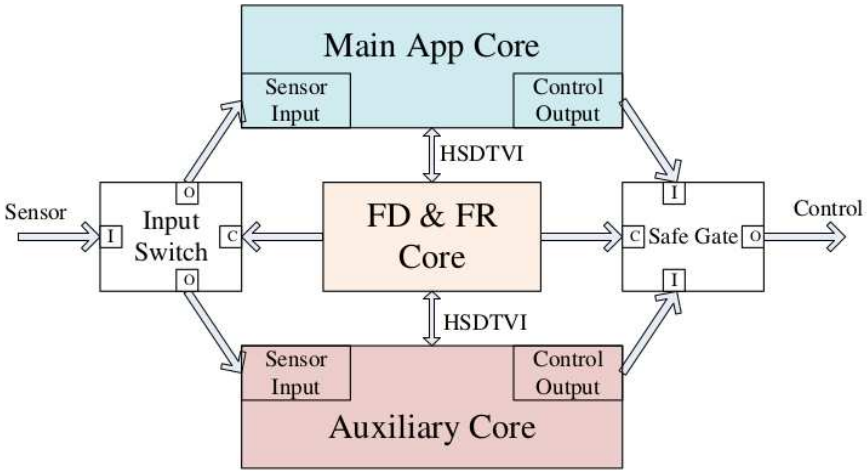
No trabalho "*Development of an energy efficient, robust and modular multicore wireless sensor network*" de Hong-Ling (2014), são apresentadas várias tecnologias baseadas em *Wireless Sensor Network* (WSN), que conforme o autor, é considerado uma tecnologia chave do século XXI, pois é fundamentada na "Computação Ubíqua", "Computação Móvel", "Computação Vestível" e "Internet de Coisas (IoT)".

Segundo a definição usada por Hong-Ling (2014), O WSN é composto por um conjunto de nós de rede com dispositivos sem fio, equipados com diferentes tipos de sensores, permitindo que seja implantado de forma fácil e econômica em áreas de interesse, que normalmente são muito difíceis ou até mesmo, impossíveis de acessar. Este tipo de dispositivo pode servir como uma interface para o mundo real, preenchendo a lacuna entre o mundo real e os sistemas de informação, através de pequenos nódulos inteligentes, que podem "sentir o ambiente" através de diferentes tipos de sensores (HONG-LING, 2014).

A proposta de Hong-Ling (2014) é baseada na redundância de *hardware* e tolerância a erros, onde são criados vários módulos de *hardware* que podem ser conectados a plataformas existentes no mercado. A ideia fundamental é o desenvolvimento de um *hardware* adicional (*slave*) faz a leitura de dados e envia via *wireless* para uma aplicação centralizadora e ao "mesmo tempo" para o *hardware* principal (*master*), que também envia para a aplicação centralizadora.

O *hardware* principal pode fazer checagens com a aplicação centralizadora, de forma a verificar os dados enviados pelo *hardware* adicional, que também enviou os mesmos dados lidos de sensores (é possível identificar ocorrência de falhas antes mesmo e enviar os dados para a aplicação centralizadora, pois o *master* recebe os dados do *slave*), conforme Figura 21.

Figura 21: Redundância de hardware.



Fonte: Obtida de Hong-Ling, 2014.

Na Figura 21 é apresentado a arquitetura do projeto proposta, a ênfase de experimentação deste trabalho, é determinado na tolerância a falhas, onde vários tipos de dispositivos inteligentes são utilizados para obter dados de sensores como, umidade, temperatura, consumo de energia, e enviados a uma aplicação centralizadora.

De forma a testar o desempenho do modelo foram utilizadas as seguintes as métricas algumas métricas, por seguinte, apresentadas. Para cálculo do número de falhas do sistema (Dovich, 1990). Uma vez de posse do número de falhas esperado, é possível calcular a confiabilidade.

Segundo o trabalho de Hong-Ling (2014), o uso mais comum é o *Failures in Time Failure Rate in Parts per Billions Hours* (FIT), sendo que esta métrica é utilizada para taxas de falhas de componentes individuais, pelo fato das falhas individuais serem muito baixas. Para calcular a ocorrência da primeira falha do sistema (DUBROVA, 2013), é usado *Mean Time to Failure* (MTTF). Já o cálculo do *Mean Time to Repair* (MTTR), ou seja, é o tempo médio necessário para reparar o sistema (DUBROVA, 2013).

Outra métrica utilizada é o Mean Time Between Failures (MTBF), que é o tempo médio entre as falhas do sistema, também é considerado ao inverso da taxa de falhas. No trabalho de Hong-Ling (2014), também são utilizadas outras métricas como formas de analisar o desempenho do modelo proposto.

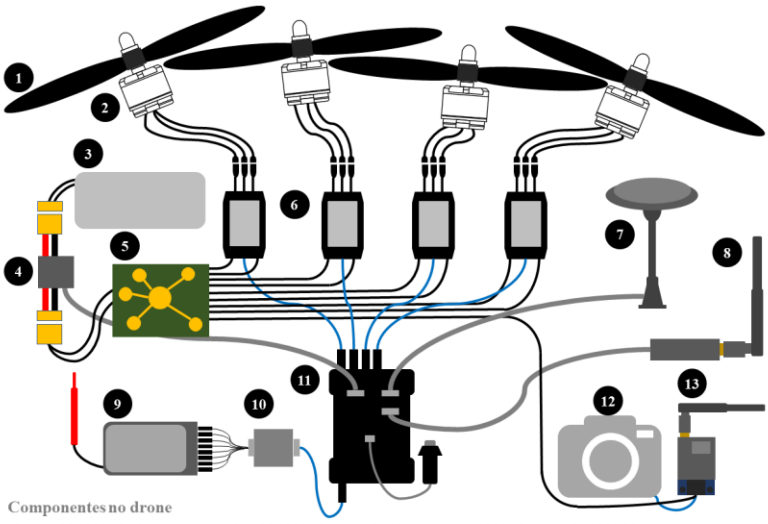
3.5.10 Uso de drone autônomo para auxílio na comprovação de alarmes

No trabalho de "Uso de drone autônomo para auxílio na comprovação de alarmes", (CONCEIÇÃO, 2018), foi desenvolvido um drone para verificação de alarmes em locais previamente cadastrados. O trabalho consistiu na construção de um Veículo Autônomo Não

Tripulado (VANT) do tipo drone, que possui a capacidade de realizar voos autônomos, fornecendo meios de verificar e validar incidentes através do disparo de alarmes.

Neste trabalho foi realizado uma análise das tecnologias disponíveis no mercado, juntamente a um referencial teórico e desenvolvimento de um protótipo de *hardware* e software para monitoramento e controle. Na Figura 22 é apresentada a arquitetura de *hardware* utilizada no projeto de drone autônomo.

Figura 22: Arquitetura de hardware utilizada para gerenciamento do drone.



LEGENDA		
1 – Hélices	7 – Módulo GPS e bússola	13 – Transmissor de vídeo
2 – Motores	8 – Rádio de telemetria	14 – Rádio transmissor
3 – Bateria	9 – Rádio receptor	15 – Receptor de vídeo
4 – Módulo de Potência	10 – PPM *Encoder*	16 – Placa de captura de vídeo
5 – Placa de distribuição	11 – Controladora de voo	17 – Notebook
6 – ESCs	12 – Câmera	18 – Rádio de telemetria

Fonte: Obtida de Conceição, 2018.

Para o desenvolvimento deste projeto, foi utilizado a controladora de voo Pixhawk com o *firmware* Copter da plataforma ArduPilot. Todo o software utilizado para o voo autônomo foi desenvolvido utilizando a biblioteca DroneKit, através da linguagem Python. Os comandos de missão são enviados via dispositivos de telemetria, conforme Figura 23.

Figura 23: Estação terrestre responsável por gerenciar o drone.

Fonte: Obtida de Conceição, 2018.

Uma vez cadastrados os pontos de verificação e os dados enviados ao drone, o mesmo realiza a missão de forma automatizada, indo até os pontos definidos. É importante salientar que as altitudes e pontos de ação devem ser cadastrados previamente, no entanto, caso o usuário queira, é possível reprogramar uma missão ou controlar o drone em tempo de voo, desde que o drone esteja em campo de alcance da estação terrestre.

Figura 24: Drone utilizado na aplicação de verificação de alarmes.

Fonte: Obtida de Conceição, 2018.

No projeto de Conceição (2018), o drone possuía além da telemetria (responsável por enviar e receber configurações), a possibilidade de transmissão de imagens, conforme pode ser visualizado na Figura 25.

Figura 25: Navegação autônoma via Mission Planer.

Fonte: Obtida de Conceição, 2018.

Como conclusão do trabalho, Conceição (2018), construiu o drone, gerou pontos de alerta, também realizou voos autônomos, conforme apresentado na Figura 25.

3.6 Comparativo entre os trabalhos

Para realizar o comparativo dos trabalhos analisados em profundidade, foi criado a Tabela 9, com uma referência numerária (1-10) para simplificar o apontamento dos trabalhos.

Tabela 9: Legenda com identificadores dos trabalhos analisados.

ID / seção	Título do trabalho
(1) - 3.5.1	*Lightweight data interchange format*
(2) - 3.5.2	*Rule-based recommendation system in IoT*
(3) - 3.5.3	*Fog Computing and Smart Gateway Based Communication for Cloud of Things*
(4) - 3.5.4	BASIS: *A Big Data Architecture for Smart Cities*
(5) - 3.5.5	*Internet of the intelligent things*
(6) - 3.5.6	*Generic architecture for the future Internet of Things*

(7) - 3.5.7	*Integrating Physical Devices in IoT*
(8) - 3.5.8	*A Middleware Architecture for Dynamic Adaptation in Ubiquitous Computing*
(9) - 3.5.9	*Modular Multicore Wireless Sensor Network*
(10) - 3.5.10	Uso de drone autônomo para auxílio na comprovação de alarmes

Fonte: do autor.

A Tabela 10 foi desenvolvida com os critérios mais relevantes apresentados nos trabalhos.

Tabela 10: Legenda com os critérios observados.

ID	**Objetivo observado**
O1	Define um protocolo e/ou padrão para transmissão de dados
O2	Propõe formas para redução no *overhead* de dados
O3	Definição de um *middleware* para IoT
O4	Recomendação de ferramentas ou arquitetura para IoT
O5	Define uma arquitetura computacional
O6	Utiliza web semântica
O7	Utiliza ontologias
O8	Propõe o uso de ferramentas externas para interpretação de regras
O9	Utiliza redundância de *hardware* para garantir a integridade de dados
O10	Define novos termos de tecnologias já existentes, porém com outras utilidades
O11	Define métricas de desempenho para avaliar arquiteturas IoT
O12	Usa histórico de contextos
O13	Usa adaptação de comportamento
O14	Utiliza dispositivos móveis autônomos

Fonte: do autor.

A percepção é empírica, mesmo não sendo expressa pelos autores dos trabalhos. Conforme a Tabela 10, os termos ou tecnologias mais recorrentes são referenciados na Tabela 11, são onde apresentadas as percepções dos objetivos dos trabalhos analisados.

Tabela 11: Comparativo entre os trabalhos analisados e objetivos observados.

Trab.	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14
1	x	x	x											
2				x	x	x	x	x					x	
3									x	x		x	x	
4				x	x							x		
5									x	x		x		x
6			x		x		x			x		x		
7			x		x								x	
8			x	x	x		x					x	x	
9									x		x		x	
10														x

Fonte: do autor.

Todos os trabalhos possuem alguma relação com esta proposta de doutoramento, seja, sob forma de arquitetura, terminologia, métricas de desempenho, forma de comunicação, redução de dados, padrão de dados, redundância de *hardware*, uso de histórico de contextos, adaptação de funcionalidades, entre outras especificidades.

A terminologia utilizada por Wang et al. (2017), "propriedades dinâmicas" é utilizado para definir os dados provenientes de dispositivos, como por exemplo, "um sensor de temperatura possui dados dinâmicos", pois varia de acordo com a temperatura, assim, este trabalho foi selecionado no Estudo Sistemático pelo termo "dinâmico", no entanto, esta terminologia não está de acordo com o que está sendo proposto nesta tese de doutoramento, que considera um sistema dinâmico, como um sistema com a capacidade de adaptar-se conforme a necessidade. De qualquer forma o trabalho de Wang et al. (2017), possui outros pontos que estão alinhados a proposta, no entanto seu maior objetivo foi introduzir a terminologia 6A da Figura 14 na seção 3.5.6.

As métricas utilizadas pelo trabalho da seção 3.5.9 podem ser utilizadas para verificar erros de comunicação, verificar pontos fortes e fracos do sistema proposto. No trabalho apresentado na seção 3.5.2 são apresentados dois *softwares* que podem ser utilizados para interpretação de instruções para máquina de regras, baseadas em ontologias.

No trabalho de Conceição, 2018 (seção 3.5.10) é desenvolvido um drone que realiza voos autônomos para locais previamente cadastrados, onde as missões de voo são configuradas através de telemetria. No trabalho os trajetos de voo, neste caso, locais de alerta, com altitude e local são informados através de um software chamado *Mission Planer* que possui um kit de desenvolvimento escrito em Python, o qual permite enviar missões para o drone através de *scripts*.

De um modo geral, praticamente todos os trabalhos estudados em profundidade, apresentam o requisito de serviços de *discover* como um ponto fundamental em qualquer implementação de uma arquitetura que suporte IoT. Vários deles defendem a utilização de ontologias para definição aberta de dados, inclusive apresentando ferramentas para utilização na prática. O uso de *bigdata* e *cloud computing* já é requisito básico para computação ubíqua, dado o grande volume de dados.

3.7 Considerações finais

Neste capítulo, foi apresentado a revisão sistemática, de acordo com os assuntos propostos por esta tese de doutoramento. Inicialmente foi definido uma *String* de busca, apresentada no início deste capítulo, com intuito de direcionar os assuntos resultantes dos mecanismos de busca utilizados.

Como retorno, ocorreram muitos trabalhos fora do escopo, repetidos, que após sucessivos filtros, foram restringidos a 44 trabalhos, posteriormente, após análise em profundidade foram selecionados 10 trabalhos. Todos os 44 trabalhos, de alguma forma contribuíram com ideias ou inspiração para o desenvolvimento da proposta desta tese.

Conclui-se que o uso de *bigdata* e *cloud computing* são requisitos funcionais para desenvolvimento da computação ubíqua na sociedade moderna, que, além do sensoriamento através de dispositivos de uso geral, quando os dados são mais relevantes para a aplicação, alguns autores propõem a redundância de *hardware*, aliadas a métricas de desempenho da aplicação.

O trabalho de Conceição (2018) possui aderência direta ao que é proposto nesta tese, pois realiza o voo autônomo para locais de interesse, dados disparo de alertas. O dispositivo utilizado foi um drone de construção própria, gerenciado pelo *software* DroneKit, que permite enviar missões para o drone.

No próximo capítulo será apresentada a proposta de um modelo computacional, sua arquitetura e funcionamento, juntamente como os diferenciais em relação aos trabalhos analisados. Observa-se que, entre os diferenciais, pode-se destacar que a estrutura proposta, permite obter dados de diferentes fontes com ou sem uso de *middlewares*, porém com o ajuste de configurações funcionais, bem como, realocação geográfica de dispositivos móveis, com base em um histórico de contextos.

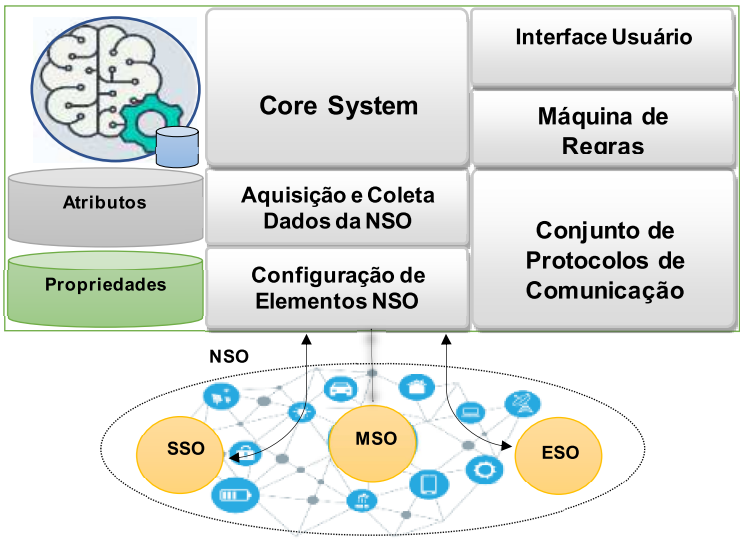
4 MODELO COMPUTACIONAL

No capítulo anterior foram apresentados os trabalhos relacionados com o modelo proposto nesta tese de doutoramento. As tecnologias apresentadas pelos autores no capítulo 3 servem como inspiração e métricas de comparação, verificando o desempenho desta proposta. Neste capítulo é apresentado o modelo computacional, em conjunto com suas funcionalidades e limitações.

4.1 O Modelo AdaptThing

O modelo AdaptThing (IoT com funcionalidades adaptativas), apresentado na Figura 26, tem como principal função gerenciar dispositivos IoT, adaptando dinamicamente suas funcionalidades, de acordo com os valores obtidos e análise de seus históricos de contextos. Além de gerenciar a coleta e obtenção de dados, o modelo responde de forma adaptativa, ajustando o comportamento operacional de funcionamento dos dispositivos gerenciados, realocando também, dispositivos móveis, conforme a necessidade, com base na inferência dos dados e eventos recebidos.

Figura 26: Representação do modelo Computacional AdaptThing.



Fonte: do autor.

Os parâmetros de comportamento operacional podem ser periodicidade, precisão, número de coletas, entre outras possibilidades que dependem de cada dispositivo. O modelo permite ainda, modificar a finalidade do dispositivo, caso o mesmo suporte alteração de *firmware*. Como exemplo, um dispositivo sensor de temperatura pode estar programado para enviar os valores lidos a cada 5 minutos para um servidor, uma alteração de comportamento operacional seria a possibilidade de modificar este tempo de envio de dados para mais ou para menos, assim como outras

possibilidades, que dependem dos recursos disponibilizados pelo dispositivo. Dependendo da capacidade dos recursos envolvidos, até mesmo a funcionalidade do dispositivo pode ser alterada, como exemplo, um microcontrolador ligado a um alto-falante com conectividade a um servidor, dependendo de suas características e constituição, é possível alterar o *firmware* através da conectividade, possibilitando usar o alto-falante para ouvir ruídos do ambiente, modificando o propósito inicial do projeto. A adaptação dinâmica do comportamento operacional dos dispositivos pode ser desde a alteração de parâmetros de funcionamento até a finalidade do dispositivo.

A possibilidade de adaptação dinâmica dos parâmetros de comportamento operacionais de um dispositivo possibilita maior economia de energia em momentos de ociosidade, bem como, coleta mais efetiva de informações dado as necessidades do contexto, inclusive podendo realocar recursos móveis para locais onde são necessárias informações mais precisas.

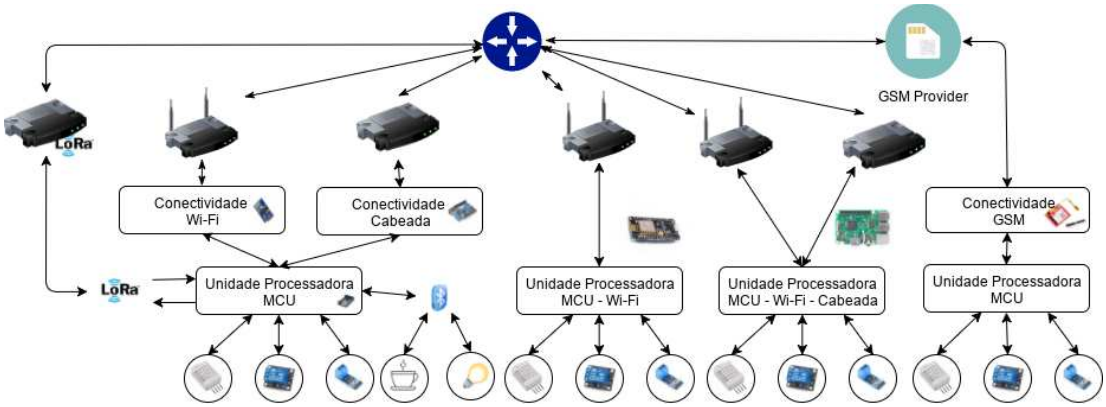
Pelo fato do modelo computacional obter dados de diferentes tipos de dispositivos, sejam físicos ou virtuais, é proposto o uso de um sistema centralizador de informações (que utiliza uma estrutura distribuída) pelo fato de um único dispositivo não conseguir analisar o contexto como um todo, porém com um conjunto de dispositivos, é possível analisar o cenário de forma mais abrangente, possibilitando melhores tomadas de decisões, bem como, compreender melhor o que está acontecendo em determinado momento.

Conforme a Figura 26, o modelo computacional possui sua estrutura em nuvem (seção 2.5), porém é possível ser implementado em estruturas computacionais mais simples. Além disso, o modelo possui o gerenciamento de três categorias de sensoriamento formando uma rede sensorial denominada de *Network Sensing Objects* (NSO), formada por:

- *Mobile Sensing Objects* (MSO): composto por dispositivos sensoriais móveis, ou seja, possuem sua localização alterada ao longo do tempo, por exemplo, *smartphones*, robôs, drones, entre outros;
- *Static Sensing Objects* (SSO): composto por dispositivos sensoriais fixos (estáticos), ou seja, sua localização não varia ao longo do tempo, por exemplo, sensores de temperatura, sensores de umidade, presença, radiação, entre outros;
- *External Sensing Objects* (ESO): composto por provedores de serviços, fisicamente podem estar fora do contexto, mas fornecem dados importantes para o contexto, por exemplo, serviços de dados de clima, imagens de satélite, entre outros.

A NSO é composta por todo e qualquer tipo de dispositivo que pode ser sensoriado. Cada dispositivo individual é chamado de Objeto Sensitivo – *Sensing Object* (SO), e cada SO pode ter um funcionamento completamente distinto. Um SO pode ser um *smartphone*, *tablet*, um circuito com um microcontrolador com vários sensores, um computador executando um ou vários serviços, entre outras possibilidades, conforme apresentado na Figura 27.

Figura 27: Network Sensing Objects (NSO).



Fonte: do autor.

O modelo AdaptThing possui um processo de inferência, baseado em máquina de regras, que é responsável pela adaptação de comportamento dos SOs pertencentes a NSO, representado na Figura 27, representada pelo cérebro. A máquina de regras sempre está funcionando em segundo plano, no entanto possui gatilhos que são disparados sempre que houver a chegada de um novo dado proveniente da NSO (novo evento).

A máquina de regra pode ser formada por qualquer tipo de linguagem ou recurso, desde que seja possível passar informações e coletar informações. Por padrão as regras são *scripts,* que podem ser feitos em qualquer linguagem, como Python, Java, Shell, PHP, inclusive chamadas de Sistema Operacional e recursos de códigos executáveis ou binários. Este tipo de funcionalidade permite executar recursos de outros sistemas, *midllewares*, recursos de *machine learning*, estando dentro ou fora do sistema implementado.

O modelo também é composto por duas estruturas de dados distintas, responsáveis por gerar dois repositórios de históricos de contextos, que estão interligados, porém um corresponde aos dados provenientes dos dispositivos SOs – *Attributes* - Atributos, e outro sobre as informações de configurações e parametrização dos SOs – *Properties* – Propriedades, conforme apresentado na Figura 26:

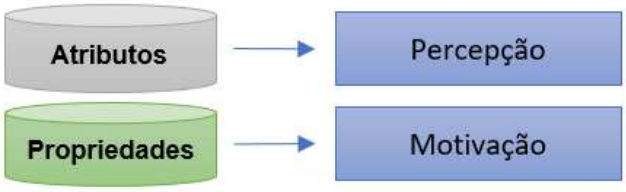
- Base de dados *Attributes*: formado por toda e qualquer informação que pode ser disponibilizada pelos SOs. Utilizado para formar o histórico de contextos de cada elemento SO. São compostos por valores, como por exemplo, temperatura, umidade, localização, corrente, iluminação, velocidade, situação, entre outros, possui também uma informação chamada de Percepção. A Percepção é um rótulo categorizador para cada situação identificada ou percebida pelo especialista, seja humano ou virtual, como exemplo, “normal”, “tempestade”, “fora da rota”, “alta velocidade”. Os dados sempre são preprocessados antes do armazenamento, organizando em tabelas apropriadas de acordo com sua categoria de NSO:
    - SSO: a localização é informada no cadastro ou entrada do SO na NSO, após somente é armazenado o instante do evento e os conteúdos de acordo com sua configuração e disponibilidade de recursos;

- MSO: sua localização varia com o tempo, assim, são armazenados os conteúdos conforme sua configuração e disponibilidade de recursos, localização e instante do evento;
- ESO: sua localização propriamente dita não é relevante, no entanto a localização de ocorrência dos dados é importante. Fornece informações de determinadas localizações, como pode exemplo a temperatura obtida pelo satélite *x* na região *y* no instante *z*. São armazenados conteúdos conforme configuração, instante do evento, a localização e abrangência da localização;

- Base de dados *Properties*: formado pelos parâmetros de configuração dos SOs. Forma o histórico de contextos sobre o comportamento de cada elemento SO. São compostos por valores como, taxa de leitura, localização, limites, e uma informação fundamental que é a motivação da alteração deste parâmetro, sendo que a primeira motivação é ativação, com os seus parâmetros de funcionalidade operacional padrão.

Conforme a Figura 28, na base de dados Atributos a percepção padrão é "normal", porém pode ser "temporal", "queda de granizo", "incêndio", "chegada de um novo integrante ao grupo", variando de acordo com suas possibilidades. A percepção são valores conceituais para categorizar as condições de funcionamento. Na base de dados Propriedades, a motivação padrão é "inicialização", e ocorrem conforme as situações dos eventos, que mudam as condições de funcionamento.

Figura 28: Tipo de histórico gerado em cada base de dados.



Fonte: do autor.

A base de dados Atributos armazena somente os valores obtidos dos SOs, assim, uma variação entre tempos de leitura (mudança na configuração do comportamento operacional do SO), por exemplo, pode gerar um erro ou ocasionar uma interpretação errada. Este tipo de erro é evitado ou reduzido pelo fato de existir o registro das alterações destas configurações.

Um exemplo de alteração de parâmetros no tempo de leitura de dados é apresentado na Figura 30. No exemplo foi modificado o tempo entre as leituras do SO de um minuto para uma hora. Os dados correspondem a temperatura interna de funcionamento do dispositivo, sendo utilizado para exemplificação. O gráfico foi adaptado de forma a manter a proporcionalidade visual.

Figura 29: Alteração no intervalo de tempo de leitura de dados.

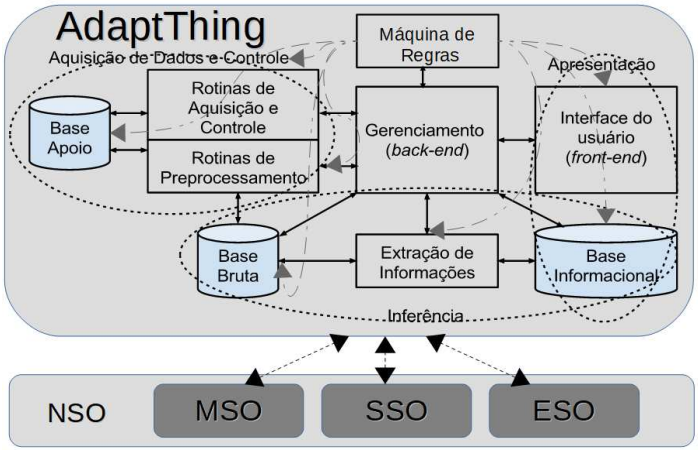
Fonte: do autor.

Tanto a inferência da máquina de regras quanto um especialista físico ou virtual, podem alterar os parâmetros de comportamento operacional dos SOs, porém, sempre que houver alguma alteração, é realizado a inserção de um registro na base Propriedades com a motivação que gerou esta alteração. Caso seja realizado uma modificação proveniente de um conjunto de regras, a categoria pode ser classificada como "desconhecido", solicitando ao especialista para denominar esta ocorrência.

4.2 Estrutura funcional do modelo AdaptThing

De forma genérica, o modelo computacional tem a finalidade de coletar os dados das mais diversas fontes de dados com a finalidade de criar um histórico de contextos do ambiente em estudo, obtendo e coletando dados de SOs a partir de uma NSO. A arquitetura do modelo é apresentada na Figura 30.

Figura 30: Estrutura funcional do modelo computacional AdaptThing.



Fonte: do autor.

O sensoriamento é utilizado na literatura para as mais diversas funcionalidades, no entanto, a periodicidade da coleta e obtenção de dados, geralmente ocorre através de agendamento ou disponibilidade de informações. A utilização de dispositivos com maior poder computacional, neste caso os SOs, permitem que estes dispositivos coletem e enviem os dados, e muitas vezes, realizem processamento local e possam ter suas funcionalidades operacionais modificadas.

Um exemplo hipotético pode ser: "um dispositivo realiza a leitura de um valor analógico a cada 2 segundos e armazena o valor em um *buffer* temporário. O servidor de aplicações solicita as informações a cada 5 minutos". Outro exemplo pode ser: "um dispositivo obtém dados de temperatura a cada 15 minutos e envia os mesmos para o servidor de aplicações". Os dois exemplos hipotéticos mencionados remetem a mesma ideia fundamental, que é existir a temporização predefinida, que muitas vezes atendem as demandas deliberadas, mas que poderiam ter seu funcionamento aprimorado.

De forma a proporcionar respostas adaptativas às necessidades do contexto foi criado o modelo computacional para gerenciar a Rede Sensorial Inteligente, chamada de Rede de Objetos Sensitivos (*Network Sensing Objects* - NSO). A atuação do sistema é baseada nas funcionalidades que um sistema em nuvem pode proporcionar, como:

- Disponibilidade: o sistema deve estar sempre disponível para que possa coletar ou obter dados dos mais diversos dispositivos da IoT;
- Escalabilidade: o sistema deve ter a possibilidade de alocar mais recursos, de forma transparente, conforme a necessidade de armazenamento, processamento, memória para atender a demanda do sistema, tanto para obtenção, coleta, processamento de regras;
- Acessibilidade: o sistema deve possibilitar o acesso de qualquer tipo de dispositivo que tenha conectividade com a Internet, tanto pelos dispositivos da IoT, quanto ao usuário operando sua interface do sistema.

4.3 Arquitetura do modelo computacional

A arquitetura para suportar a proposta do modelo computacional é apresentada na Figura 30, onde são apresentadas as principais funcionalidades do AdaptThing. O modelo delega, obtém e coleta dados dos mais diversos SOs, através de uma rede da NSO, composta por vários dispositivos da IoT, como apresentado na Figura 27. Uma NSO é composta pelos seguintes objetos sensitivos:

- Os dados dos SOs são utilizados para formar duas bases de histórico de contextos, uma formada pelos "atributos" e outra pelas "propriedades". Onde, os atributos são os dados obtidos dos SOs como por exemplo o valor medido de temperatura, o valor medido de umidade. As propriedades são os valores de configuração do comportamento operacional dos SO, como por exemplo, intervalo entre leituras, valores de referência, limiares de ativação, parâmetros de acordo com o tipo de SO;

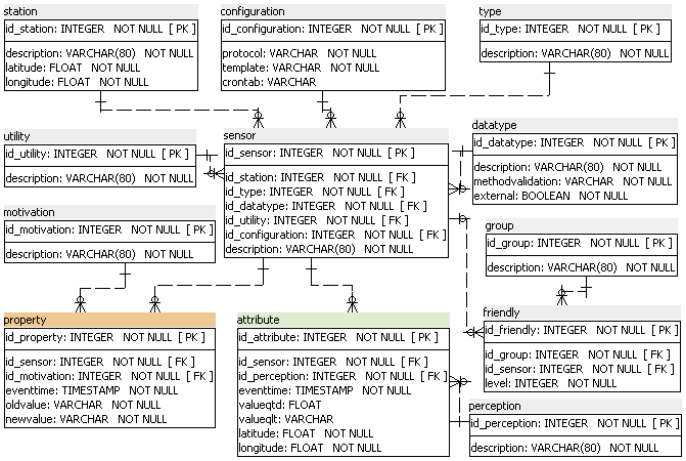
- A análise do histórico de contextos é realizada através de um conjunto de regras definidas por especialista físico ou virtual, que permite ao modelo analisar e adaptar determinados objetos sensitivos, a forma como devem atuar, permitindo assim informações mais precisas sobre os fatos de interesse.

Observando-se o modelo da estrutura funcional do AdaptThing, modelo é dividido nos seguintes macros segmentos internos:

- Delegação, obtenção e coleta de dados: responsável por coletar e receber dados dos SOs, é subdividido em duas categorias:
    - SOs "Padronizados" (*padronized*): utilizam o protocolo proposto nesta tese de doutoramento para gerenciar os dispositivos, apresentado na seção 4.4;
    - SOs "não Padronizados" (*not padronized*): qualquer tipo de protocolo que pode ser implementado e/ou possui especificação de uso;
- Geração de "Histórico de Contextos": a aplicação gera histórico de contextos tanto dos dados quanto das configurações dos SOs;
- Máquina de Regras: formado por um conjunto de regras, que com base no histórico de contextos e novos dados, pode atuar no comportamento dos SOs;
- Adaptação de SOs: responsável por informar os ajustes e configurações que devem realizados nos objetos para atender as necessidades;
- Interface com Usuário: interface de comunicação com o usuário.

Também é proposta uma estrutura de dados em sistema de arquivos, fora da base de dados, de forma a armazenar dados de maior tamanho ou com formato não previamente estruturado. Para simplificar a implementação, o mesmo pode ser implementado com um banco de dados relacional, auxiliado por um sistema de arquivos. Na Figura 31 é apresentada um trecho da estrutura de dados, com a finalidade de facilitar a compreensão do modelo proposto.

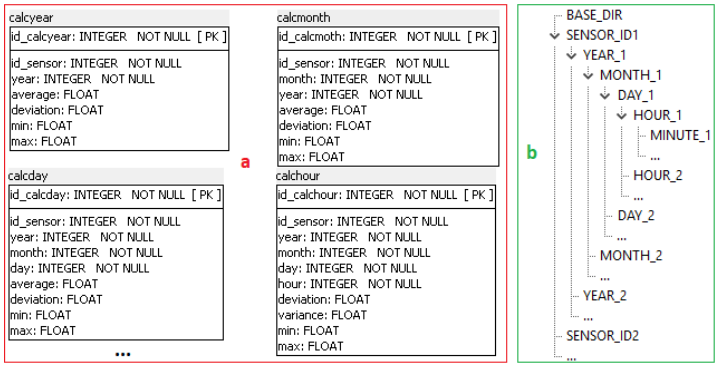
Figura 31: Estrutura de dados usando um modelo relacional de dados.



Fonte: do autor.

Conforme os dados são recebidos, são pré-avaliados, em seguida analisados por uma lógica de execução rápida, baseada em média e desvio padrão (minuto, hora, dia, média dos últimos "n" valores lidos), caso esteja fora dos padrões, os dados são avaliados por algoritmos mais complexos, baseados em *machine learming*, disponibilizados em estruturas de diretórios indexados.

Figura 32: Estrutura de dados com diretórios indexados.



Fonte: do autor.

Na Figura 32 são apresentadas as estruturas de dados consolidadas através de um diagrama simplificado da estrutura de armazenamento de dados do AdaptThing. A tabela "attribute" é responsável pelo armazenamento do histórico de contextos dos SOs, ou seja, valores lidos dos SOs. A tabela "property" armazena o histórico de contextos das modificações das funcionalidades operacionais dos SOs. Este segundo histórico é necessário, pois o processo de adaptação dos SOs pode afetar os resultados obtidos, gerando anormalidades.

Para o armazenamento de informações complexas, ou seja, caso um SO possuir o "datatype" do tipo "external", os dados são armazenados em uma estrutura de diretórios distribuídos, com

estruturação indexada. Seus dados são processados por métodos específicos, do próprio "datatype", como por exemplo, uma imagem de satélite, um texto, entre outras informações com maior volume ou estruturação variada.

Propõem-se ainda a utilização de uma tabela na qual são relacionados SOs que possuem proximidade física e/ou similaridade de funcionamento, de forma que, quando um SO possuir dados fora do esperado, possa verificar a variação de comportamento de outros SOs "friendly". Este recurso pode ser explorado, para modificar o comportamento dos SOs " friendly ", preparando os mesmos para um evento que poderá chegar até eles.

No caso de coleta de informações qualitativas ou textuais, como por exemplo os usuários em determinado local e momento, o e-mail enviado de um usuário para outro, neste caso, é realizado um processo de agrupamento para transformação destes dados em valores quantitativos, ou então preprocessados com ferramentas específicas como o Apache Spark (HAZARIKA, RAM and JAIN, 2017; ONAL, SEZER, OZBAYOGLU, DOGDU, 2017) para mineração de texto, análise social, classificação de texto, análise de sentimentos, agrupamentos, no entanto, os dados brutos continuam sendo armazenados em tabelas e/ou diretórios indexados.

Os dados recebidos podem ter erros ou inconsistências, estarem fora dos limites, ou até mesmo não serem recebidos no momento esperado. Sempre que houver um evento inesperado, é gerado um alerta, e caso ocorra de forma recorrente, de acordo com configurações prévias de contagem de eventos, é iniciado o processo de adaptação, de forma a corrigir o funcionamento do SO, ou detalhar de forma mais precisa o evento.

O processo de adaptação do AdaptThing ocorre conforme o fluxo da Figura 33, onde primeiro são observados os valores da base de dados já consolidada, se os dados forem considerados anormais, ou seja, fora da média e desvio padrão dos valores já consolidados de forma temporal, são disparadas as análises mais complexas baseadas em técnicas inteligentes ou *machine learning*, que estão armazenadas nos diretórios indexados.

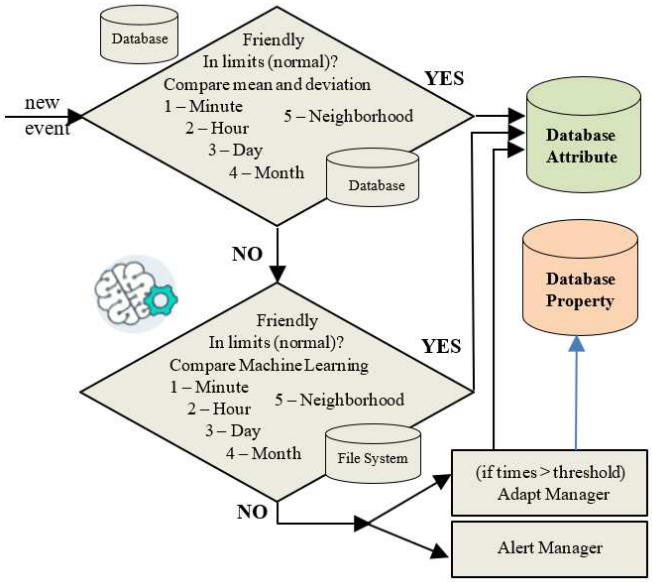
Como exemplo, é recebido a medida de temperatura de um SO, onde primeiramente é verificado se a temperatura está dentro da variação média histórica do dispositivo na base consolidada para o intervalo de tempo “minuto” (mês, dia, hora, minuto), se estiver dentro da média e desvio padrão, a nova informação é armazenada na base Attribute e o processo termina. No caso contrário, é verificado a média dentro do período de “hora” (mês, dia, hora) e assim sucessivamente até chegar ao mês. Caso ainda esteja fora do intervalo supostamente normal, é disparado o processo inteligente de verificação. Caso no processo inteligente de verificação ainda for considerado anormal, então é realizado um alerta e ao mesmo tempo chamado o processo de adaptação, que se for solicitado mais que *n* vezes dentro de um intervalo de tempo, realizará a alteração no comportamento do SO conforme foi previamente programado de forma a detalhar melhor o evento, como por exemplo, solicitar ao dispositivo para aumentar o número de leituras por intervalo de tempo.

É possível programar o sistema para ter o comportamento de forma inversa para identificação da anormalidade, no entanto, dependendo da situação, o sistema poderá ficar disparando alertas a todo instante, bem como, no caso aqui proposto, não disparar o alerta e a adaptação em momentos que deveria ser realizado.

O ideal é possuir uma base histórica de forma a reduzir os erros, onde quanto mais informação histórica, menor a possibilidade de erros. Caso a base de dados não possua dados

históricos, os erros serão maiores, no entanto, o escopo deste trabalho parte do pressuposto de uma base de históricos de contextos com mais de uma passagem de periódica.

Figura 33: Processo de adaptação.



Fonte: do autor.

Dessa forma, presume-se que na maioria dos casos os valores recebidos dos dispositivos são normais, ou seja, dentro de intervalos previamente conhecidos, somente realizando verificações mais complexas em situações possivelmente anormais, pois consomem maior poder computacional para serem realizadas. Outro ponto a favor deste tipo de abordagem é que as consultas nas tabelas já consolidadas possuem tempo de resposta mais baixo do que a execução de um método complexo.

O processo de adaptação, uma vez disparado, altera o comportamento funcional do SO, modificando os parâmetros de funcionamento. Cada SO pode ter funções completamente distintas um do outro, como por exemplo, taxa de amostragem, intervalo entre transmissões, destino geográfico, limites, entre outras possibilidades.

4.4 Protocolo proposto

O sensoriamento é utilizado na literatura para as mais diversas funcionalidades, no entanto, a periodicidade dos dados enviados para as aplicações, sejam elas centralizadoras ou distribuídas, geralmente ocorrem através de agendamento ou disponibilidade de informações.

Com a maior capacidade de processamento disponível aos dispositivos da IoT, aliado com a possibilidade de conexão com a Internet, foi criado o conceito denominado de Redes de Sensores Inteligentes. As Redes de Sensores Inteligentes permitem que os sensores, não somente coletem os dados, como também, realizem processamento local (SALVADORI, 2003).

Se um sistema que armazena os dados for sempre alcançável pelos demais elementos da rede, já é possível ter uma rede de sensoriamento, com informações relevantes, sendo enviados pelos elementos da rede de tempos em tempos, para modificarem sua periodicidade ou parâmetros.

No entanto, caso seja necessária uma lógica mais elaborada, com informações de um conjunto de elementos da rede, pode ser muito processamento para um elemento sensorial da rede. Nesse caso o sistema onde os dados são armazenados e processados, que possui maior poder de processamento, pode analisar o conjunto como um todo, informando o novo comportamento que os dispositivos deverão ter, caso necessário.

O problema ocorre quando o sistema principal quer alcançar o elemento sensorial, por muitas vezes possuir um IP não verdadeiro e por isso não é alcançável, tornando-se um percalço para os sistemas que precisam de adaptação instantânea ou mais rápido o possível.

O processo de comunicação entre o sistema principal e os elementos sensoriais é possível, caso todos os elementos sensoriais tiverem IP's verdadeiros, mas isso ainda não é uma realidade para todas as situações, ou é uma realidade com um custo bem mais elevado, o que torna o sensoriamento de várias situações inviáveis ou deixando a desejar no quesito tempo de resposta. As principais soluções adotadas para poder ajustar e adaptar o funcionamento dos elementos remotos são:

- O elemento sensorial envia as informações de tempos em tempos, momento em que é realizado alterações no elemento (somente quando é iniciada a comunicação pelo elemento remoto);
- Todos os elementos possuem IPs verdadeiros;
- O sistema centralizador está na mesma rede dos elementos sensoriais;
- É utilizado *hardware* adicional, rodando uma aplicação, *framework* ou serviço adicional para coletar informações da Intranet e realizar comunicação com a Internet;
- É realizado um mapeamento de comunicação nos roteadores informando a rota para cada elemento da rede.

Para possibilitar a coleta e obtenção de dados, bem como a adaptação de comportamento dos SOs, foi criado um sistema de gerenciamento e um protocolo para realizar a coleta e aquisição de dados, bem como a adaptação do comportamento operacional dos dispositivos.

O protocolo proposto é uma arquitetura de software que permite obter, coletar e adaptar o comportamento de SOs de uma NSO, possuindo compatibilidade com vários protocolos de interoperabilidade para aquisição de dados, porém com maiores funcionalidades utilizando o protocolo proposto.

Para que seja possível obter os dados quando necessário, foi criado um protocolo de comunicação para objetos da IoT padronizados, ou seja, quando existe a possibilidade de programar o *software* ou *firmware* do SO. No caso dos SOs não padronizados, a arquitetura proposta suporta ou é compatível com vários outros protocolos, no entanto, caso o funcionamento seja muito específico, a arquitetura tem suporte a implementação e chamadas de *call-backs*, protocolos ou serviços externos.

4.4.1 Protocolo de interoperabilidade

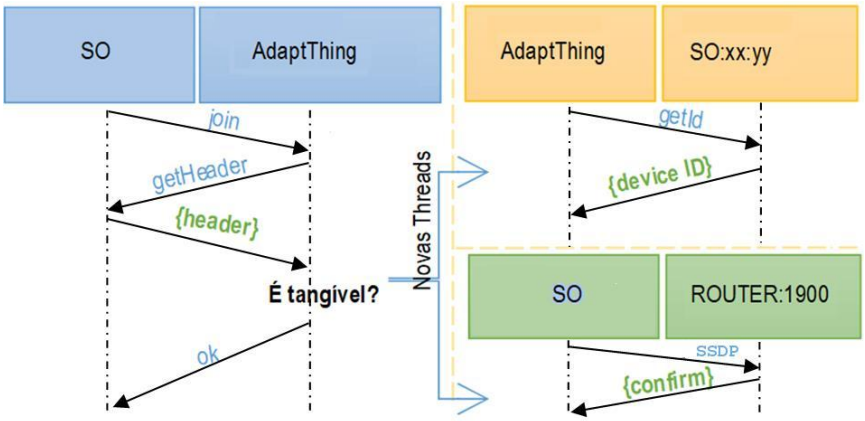
A qualidade dos dados depende diretamente da efetividade das regras, que por sua vez dependem de um especialista que conhece o comportamento do contexto em que a rede está inserida. A quantidade de dados não necessariamente corresponde à qualidade, inclusive podendo ser esta uma restrição. Existem situações onde o consumo de energia demandada pelos SOs é restritivo, assim, a arquitetura do sistema pode demandar dados em intervalos maiores, e dado a ocorrência de eventos no contexto, pode modificar o seu comportamento operacional obtendo dados com maior efetividade, e posteriormente voltar a ter um funcionamento com maior economia de energia e menor volume de dados.

O sistema é implementado utilizando-se a linguagem Java e Python e desenvolvido de forma modular, permitindo a inclusão de novos módulos, compatibilizando com SOs dos mais diversos fabricantes. O sistema trabalha com duas categorias de SOs, os "padronizados" e "não padronizados", onde os "padronizados" referem-se aos objetos sensitivos que podem ser programados, ou seja, que o usuário possui gerência sobre *firmware* ou *software*, já os "não padronizados" referem-se a dispositivos que não podem ter sua forma de funcionamento especificada.

Os dispositivos sensitivos padronizados utilizam um protocolo próprio para fazer parte da arquitetura do sistema com base no modelo computacional AdaptThing, bem como a coleta ou aquisição de dados. O uso deste protocolo permite a obtenção de dados, bem como, a alteração na forma de funcionamento, permitindo a adaptabilidade às necessidades de contexto.

Quando um novo objeto sensitivo é colocado em funcionamento, primeiramente o mesmo realiza uma solicitação para fazer parte da rede sensitiva, conforme Figura 34, momento em que é verificado se é ou não possível acessar o mesmo a qualquer instante, ou seja, vai ter a capacidade de realizar uma adaptação de comportamento operacional no momento que precisar ou usuário solicitar.

Figura 34: Protocolo Padronizado AdaptThing.



Fonte: do autor.

Conforme a Figura 34, inicialmente o SO deve solicitar ao sistema para fazer parte da NSO, para isso, o dispositivo começa comunicação (TCP) solicitando "join", o sistema responde com "getHeader", momento em que o dispositivo devolve um cabeçalho dividido em 3 grupos, formados pela identificação, atributos e propriedades, conforme Figura 35. Caso a troca de informações não siga o padrão de comunicação ou a estrutura dos cabeçalhos, o sistema descarta a conexão.

Figura 35: Expressão regular de validação da comunicação do protocolo padronizado.

```
1  ({)([\w|-]+):(\w+):([\w+|-]+):([0-9]{4,5}):(\w+)\[(.*)\],\[(.*)\](})
2  {([\w|-]+):([\w|-]+)}
3  {([\w|-]+):([\w|-]+),([\w|-]+):\[([\w|,|-]*)\]}
```

Fonte: do autor.

A validação inicial do cabeçalho é dada pela expressão regular da Figura 35, onde a linha 1 valida a identificação do SO, a linha 2 valida os atributos, já a linha 3 as propriedades. Por fim um *parser* analisa e desmembra os dados para validação final.

Figura 36: Exemplo de cabeçalho de resposta obtido por *getHeader()*.

```
{ESP-DS3231-0001:SSO:mod-sen-udp:2412:Funcao:Padronizador[
    {current:int},{led:int},{temp:float},{luz:int},{interval:int},{scan-ap:text}
],[
    {interval:int,int:[]},{led:int,int:[0,1]}
]}
```

Fonte: do autor.

A primeira parte da Figura 36 em vermelho corresponde a identificação do SO, assim como outras informações necessárias para o sistema gerenciador poder realizar a comunicação:

- Id: identificador do dispositivo (no exemplo, ESP-DS3231-0001);
- Tipo de dispositivo: estático (SSO), móvel (MSO) ou externo (ESO);
- Protocolo de comunicação: qual dos protocolos de comunicação será utilizado para a troca de informações, pode ser específico, MQTT, AMPQP, baseado em *webservices*, *request-response*, com estrutura definida, *get-set*, entre outras possibilidades. O sistema gerenciador possui suporte a vários protocolos de comunicação sobre a camada TCP e UDP, tendo a possibilidade de implementação de novos protocolos;

- Porta de comunicação: a porta será definida de acordo com a possibilidade de comunicação com o sistema gerenciador. Essa porta será definida através de SSDP e verificação de sobreposição no sistema gerenciador;
- Função do dispositivo: texto que define a função primordial do dispositivo;
- Padronizador: é de uso opcional, é utilizado para definir o significado de cada um dos identificadores utilizados tanto na parte dos atributos, quanto na parte das propriedades. Por exemplo, na Figura 36 foi utilizado o termo "current", mas nada impediria de usar apenas um mnêmico como "c", "cur", um padronizador é quem define o que significa cada sigla, isso simplifica o uso de termos mais explicativos para cada um desses parâmetros. Outra vantagem que este recurso oferece é a diminuição de *overhead* na comunicação de dados, onde são transmitidos conteúdos mais curtos.

A segunda parte da Figura 36, em verde, é utilizada para definir os atributos. Um atributo é uma informação que pode ser lida do SO. O atributo sempre é acompanhado do seu tipo de dado. Dentre os principais tipos de dados, cita-se:

- int: valor inteiro (caracteres *American Standard Code for Information Interchange* - ASCII, de 48 a 57) - formado por cadeia de caracteres;
- float: valor com ponto flutuante (caracteres ASCII, de 48 a 57 e um 46 (ponto));
- text: é possível qualquer valor de 0 a 255 (ASCII), no entanto por questões de visualização e compatibilidade internacional de *code pages*, sugere-se utilizar somente os caracteres 32 a 127 (ASCII);
- image: arquivo binário que é armazenado fora do banco de dados. No banco de dados somente é armazenado um *hash* que identifica a mesma em uma estrutura de diretórios;
- video: mesmo esquema de uma imagem;
- vstream: somente é informado as informações necessárias para poder obter o *stream*, seja áudio ou vídeo;
- sound: mesmo esquema de uma imagem;
- sstream: mesmo esquema do vstream;
- binary: os códigos serão armazenados em *little endian*, caso seja necessário, é possível criar um tipo de dado, organizado estruturalmente em *big endian* para definição do alinhamento de bits.

A terceira parte da Figura 36, em azul, é utilizada para informar as propriedades do SO, as propriedades são termos ou parâmetros que podem ter seus valores alterados, permitindo assim ao sistema gerenciador poder modificar seu comportamento operacional dos SO e a adaptar sua funcionalidade de acordo com as necessidades. Os tipos de dados são os mesmos já citados nos atributos. A estrutura de uma propriedade é:

- Nome da propriedade: o termo utilizado para fazer a alteração de seu valor;

- Tipo de dado da propriedade: qual tipo de dado da propriedade;
- Tipo de dado de validação: forma de validação dos dados recebidos;
- Domínio da propriedade: quais são os valores válidos ou aceitos. É possível separar valores usando “,”, ou definir intervalos usando “-“, ou ambos. Da mesma forma como é possível usar mnêmicos para os termos dos atributos e propriedades, é possível usar no domínio (desde que siga um padrão).

Eventualmente um objeto sensorial pode deixar de ser tangível, dessa forma, foi acrescentada no protocolo uma propriedade de temporização, onde o SO de tempos em tempos realiza uma requisição ao sistema gerenciador, momento em que, caso seja necessário, novos procedimentos podem ser definidos, como por exemplo, a reprogramação de propriedades ou até mesmo do dispositivo (quando possível), através da nova definição de *firmware*. Caso o dispositivo não seja alcançável, é disparado o procedimento de auto mapeamento de rede do protocolo, apresentado na seção 4.4.3.

4.4.2 Sistema de gerenciamento IoT

Para a criação de um sistema que permita coletar informações de dispositivos da IoT, bem como, poder adaptar o comportamento dos mesmos a qualquer momento, é necessária uma infraestrutura que permita tal comunicação.

Nos casos em que os dispositivos possuem IP real, a comunicação é transparente, no entanto, em muitos casos isso não ocorre, principalmente onde é considerado o menor custo possível. Dependendo da situação o custo de equipamentos adicionais pode ser inviável financeiramente ou até mesmo impossibilitado por vários motivos, como por exemplo, relevo, clima, alimentação energética, entre outros.

No mercado existem componentes computacionais como por exemplo, o ESP8266 ou módulos baseados no mesmo, que possuem baixo custo, dado as capacidades de conectividade Wi-Fi e processamento oferecidas. O modulo ESP8266, possui limitações como o número máximo de conexões simultâneas, logo, pode não ser uma boa solução como *gateway* de rede, principalmente quando for utilizado maior número de elementos. Apesar de sua limitação em relação a quantidade de conexões simultâneas, é uma opção para oferecer conectividade Wi-Fi para dispositivos sensoriais que não possuem acesso à Internet. Um dispositivo de baixo custo que poderia ser usado como *gateway* é o Raspberry Pi (CALDAS-CALLE L., JARA, J., HUERTA M. and GALLEGOS P, 2017; CHATTERJEE N. R., NETHRA U. and SUMA V, 2018) ou similar, apesar de ser um dispositivo de baixo custo, quando comparado a outros dispositivos com processamento similar, seria um custo a mais para gerenciar cada sub rede. Um dispositivo como Raspberry Pi, é uma boa solução de baixo custo para implementação de um gateway para dispositivos da IoT, quando necessário a utilização de *hardware* adicional como concentrador de rede ou middleware entre o sistema gerenciador e a NSO.

Outra solução poderia ser utilizar roteadores com ou sem Wi-Fi, executando o OpenWRT (KIM J. and LEE, L, 2016; PALAZZI C., E., BRUNATI, M. and ROCCETTI M, 2010), que é uma

possibilidade de customização de *firmware* para roteadores, ou similar, criando um software na camada de aplicação de forma a rotear os pacotes para os dispositivos. Esta postura é possível, no entanto, os equipamentos precisariam ter mais capacidade de armazenamento, processamento e memória, logo, equipamentos com custo mais elevado.

A solução proposta nesta tese, é utilizar a infraestrutura já existente, seja com equipamentos de ponta, até mesmo equipamentos de uso doméstico, para prover a conectividade do sistema gerenciador, utilizando o protocolo proposto, atuando na parte lógica, sem infringir os padrões do mercado, resolvendo automaticamente o mapeamento dos dispositivos da IoT. Esta solução resolve o problema de conectividade do sistema gerenciador como os SOs da NSO, ao mesmo tempo, reduz o problema de heterogeneidade, pois utiliza uma estrutura padronizada quanto os atributos e propriedade dos elementos envolvidos na rede.

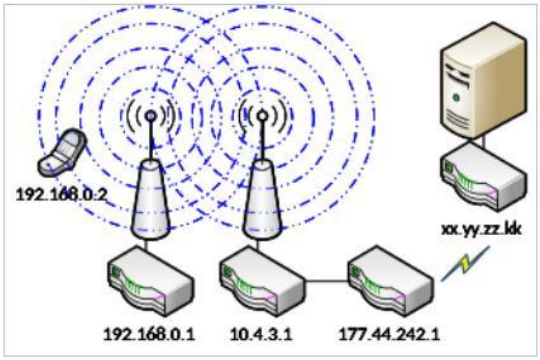
4.4.3 Recursos de Rede

O uso ou criação de um protocolo que permite o acesso de um sistema externo até aos SOs é necessário para poder realizar a adaptação dos SOs no momento que for necessário ou desejado. Devem ser utilizados padrões de comunicação regulamentados e homologados, caso contrário, os equipamentos de rede podem descartar os pacotes de comunicação ou não realizarem os procedimentos necessários.

Quando o SO precisa realizar a comunicação com um sistema na Internet, sua estrutura pode ser abstraída, onde o importante é conseguir fazer a comunicação. No entanto, para que seja possível realizar a configuração de dispositivos automaticamente, de forma a garantir a comunicação do servidor com os SOs, é imperativo conhecer a arquitetura de Intranet, e estar preparado para reconfigurar suas funcionalidades, conforme a necessidade.

De forma a compreender a necessidade do reconhecimento da estrutura da rede, na Figura 37 é apresentada uma estrutura de Intranet como de exemplo, onde um SO está alocado. A estrutura de Internet é desprezível, no entanto os saltos (conexões entre roteadores) internos até a máquina de borda são importantes, pois nos mesmos serão criadas regras de redirecionamento de portas usando UPnP - PCP (seção 2.6.2).

Figura 37: Visualização da estrutura da rede.



Fonte: do autor.

Um SO, uma vez ativado, já está programado para buscar pela arquitetura do sistema gerenciador ou outro que suporte o protocolo, que está hospedado na Internet e possui IP verdadeiro. Assim sendo, após ter estabelecido conexão com a Internet, é obtido a estrutura de rede, através de um procedimento similar ao serviço de rede *tracert* ou *traceroute*, porém sem a necessidade do tempo de resposta. A Figura 38 apresenta o *tracert* partindo do objeto sensitivo.

Figura 38: Obtenção da estrutura da rede.

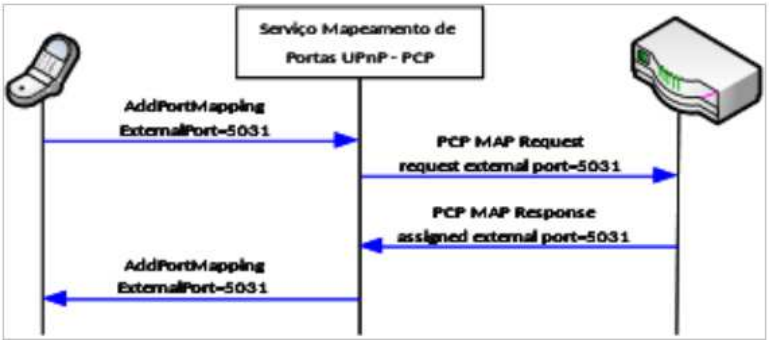
```
  1 ms  192.168.0.1
  1 ms  10.4.3.1
  1 ms  177-44-242-1 [177.44.242.1]
  1 ms  bbbbbb.br [177.44.240.36]
 14 ms  llllll.br [177.44.240.254]
  4 ms  xxx.tche.br [200.19.240.225]
  4 ms  cccc.rnp.br [200.143.255.161]
162 ms  200.143.254.122
155 ms  lll.rnp.br [200.143.252.25]
  ... continua
```

Fonte: do autor.

No caso da Figura 37 e Figura 38, o roteador que possui IP 188.44.242.1 e deve criar uma regra que, sempre que receber uma requisição na porta xy, deverá redirecionar para o IP 10.4.3.1 na porta xy1, que por usa vez, deverá direcionar para a IP 192.168.0.1 na porta xy2, que finalmente, deverá redirecionar para o IP 192.168.0.2 na porta xy3. As portas podem ser diferentes, desde que exista a possibilidade de mapeamento do ponto de entrada até o SO.

O *Port Control Protocol* (PCP), apresentado na seção 2.6.2, é utilizado para realizar o mapeamento de portas de forma automatizada. Os roteadores devem ter esta opção ativada. Por padrão esta opção vem desativada por questões de segurança. Na Figura 39 é apresentado o caso de sucesso quando um SO solicita o mapeamento de portas para o roteador. Também é possível informar a porta interna, além da porta externa.

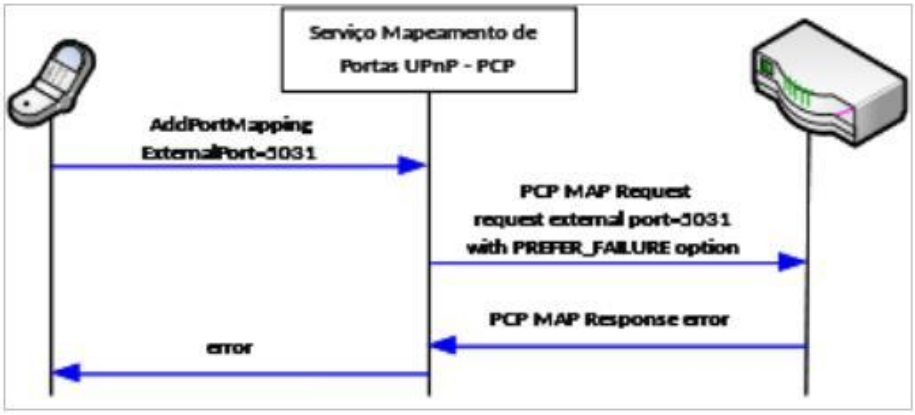
Figura 39: Mapeamento de portas automatizado, realizado com sucesso.



Fonte: do autor.

Na Figura 40 é apresentado o funcionamento do protocolo UPnP - PCP sem sucesso, que pode ser ocasionado por várias situações, dentre elas, porta já em uso. Como o controle de portas é realizado pela arquitetura do sistema gerenciador, supostamente não poderia ocorrer conflito de portas, no entanto, pelo fato dos equipamentos de rede utilizarem a "primeira porta" acima de 1024 para solicitar e receber conexões, pode ocorrer algum conflito, que será resolvido por outra rodada de reconfiguração automatizada.

Figura 40: Mapeamento de portas automatizado, sem sucesso.



Fonte: do autor.

O funcionamento do protocolo proposto possibilita o mapeamento automático de toda a NSO. O protocolo utiliza como base o protocolo SDDP existente na suíte de protocolos *discovery* do UPnP, para configuração de regras de redirecionamento de pacotes. Também utiliza o serviço *traceroute* para descoberta da estrutura de rede, ambos implementados nos SOs, quando possível o uso de protocolos padronizados.

4.4.4 Segurança do protocolo

A segurança do protocolo é baseada em outras camadas de rede, no entanto, implementa alguns recursos como *tokens* de execução baseados em *hash* (segurança por obscuridade), quando possível recursos baseados em SSL. O SSL não está sempre disponível, pois alguns dispositivos não possuem poder de processamento o suficiente para isso.

Como o protocolo é padronizado, sabe-se a forma de comunicação, dado a troca de mensagens com a estrutura de comunicação, assim sendo, sempre que possível é implementado técnicas de criptografia de dados, de acordo com o poder de processamento do SO.

4.4.5 Cenários de Aplicação do protocolo

Os cenários de aplicação pode ser qualquer tipo de aplicação, no entanto, os maiores benefícios de seu uso, ocorrem em situações onde, por padrão, não exista a possibilidade de ter acesso ao SO diretamente. Situações onde todos os SOs possuem IPs verdadeiros, ou possuem acesso via *hardware* adicional, também podem são contemplados, pois além de mapeamento de redes automatizado, que neste caso não será utilizado, o protocolo possui todo o suporte para aquisição e coleta de dados, além do processo de reconfiguração de SO.

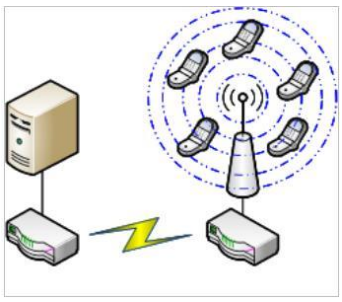
O principal objetivo deste protocolo é não modificar a infraestrutura básica, procurando por uma solução simples e de baixo custo, atuando na lógica da infraestrutura, ao invés da parte física, inclusive podendo utilizar equipamentos de uso doméstico para realizarem automaticamente a comunicação entre o sistema gerenciador e os SOs.

Cenário 1 - Arquitetura Básica

Neste cenário existe apenas um roteador ou equipamento com IP verdadeiro (celular, roteador Wi-Fi doméstico, entre outros). A rede também possui conexão Wi-Fi (podendo ser um roteador conectado como *bridge* a um rádio Wi-Fi), podendo ser utilizado uma rede cabeada da mesma forma.

Conforme apresentado na Figura 41, os SOs abrangidos pelo Wi-Fi possuem IPs não verdadeiros, assim sendo, a arquitetura do sistema gerenciador não consegue acessar os mesmos em "tempo real", pois deve esperar um dispositivo solicitar a comunicação para poder modificar os parâmetros de funcionamento, isso pode ocasionar *overhead* de comunicação, caso seja desejado o menor tempo de resposta, ou poderá ocorrer a impossibilidade de acesso a qualquer momento.

Figura 41: Cenário com um roteador Wi-Fi.



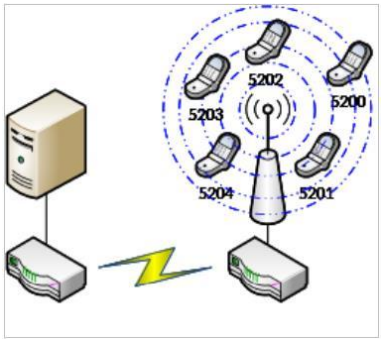
Fonte: do autor.

Outra possibilidade, seria o servidor estar na Intranet usando um equipamento adicional na infraestrutura existente, ou todos SOs possuírem IP válido. Esta possibilidade aumenta custos pelo uso de equipamentos adicionais ou dificulta o uso em estruturas maiores, onde o servidor precisa controlar o conjunto de redes distintas espalhadas pela Internet.

A solução proposta baseia-se na utilização do protocolo UPnP - PCP, para permitir o mapeamento de portas no roteador da NSO. Assim os SOs, com um pequeno código adicional, podem solicitar o mapeamento de portas para o roteador. Uma vez que exista a possibilidade do mapeamento de portas, cada um dos dispositivos precisa informar a porta que deseja representar a mesmo, sem a possibilidade de conflito de portas.

O dispositivo pode solicitar portas aleatoriamente, até encontrar uma liberada, no entanto, com o uso de um sistema centralizador este esforço é minimizado, pois o dispositivo escolhe uma porta e informa para o sistema centralizador, em caso de conflito, o sistema centralizador informa uma possível porta, caso o mapeamento tenha sucesso, ou seja, o sistema centralizador consegue chegar ao SO, o sistema armazena esta configuração como forma de acesso ao dispositivo, até que o mesmo venha a ter nova configuração (*lease time*).

Figura 42: Estrutura de rede com portas mapeadas.



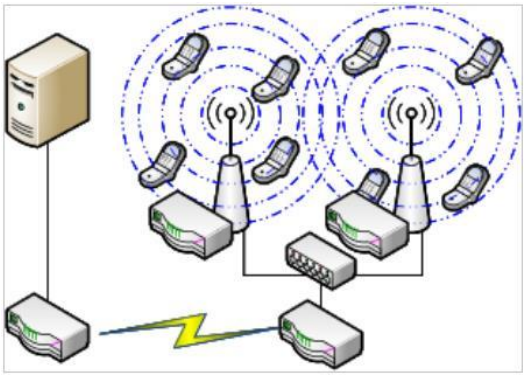
Fonte: do autor.

No caso da Figura 42, cada SO possui uma porta de conexão, neste caso, o roteador de borda da NSO possui uma tabela que faz o redirecionamento de pacotes, de acordo com a porta solicitada.

Cenário 2 – Múltiplos níveis de roteamento

O esquema apresentado na Figura 43 apresenta um roteador de borda com IP verdadeiro, que é interligado a outros dois roteadores que, em uma estrutura simples, já possuem IPs não válidos, logo abaixo, outros dispositivos que também possuem IPs não válidos.

Figura 43: Múltiplos níveis de roteamento.



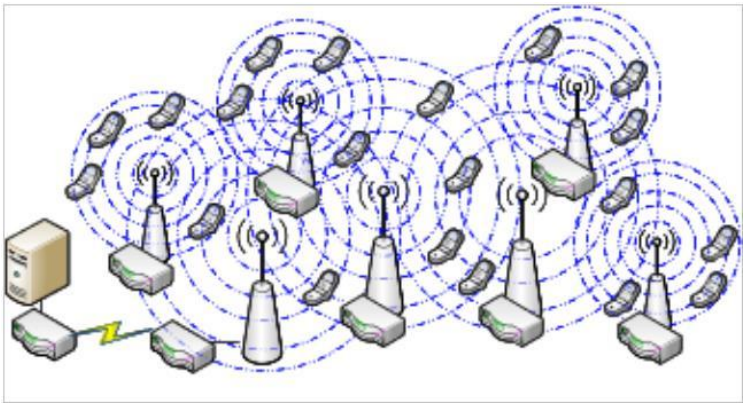
Fonte: do autor.

Nesta situação o uso do UPnP torna-se mais complexo, pois o SO somente alcança o roteador o qual está alocado. Nesta situação é necessário primeiramente conhecer a estrutura a qual o SO está estruturado, para isso é acrescentado outro pequeno código recursivo que vai sequencialmente buscando os saltos de rede (*router hops*), utilizando-se a lógica de mapeamento de rede, apresentado no início da seção 4.4.3.

Uma vez que é conhecida a estrutura de rede, é feito a solicitação de mapeamento de portas, de forma a redirecionar o fluxo de portas entre um roteador até o outro, chegando até o SO. Este é um processo que demanda tempo, porém é realizado somente quando um novo SO é acrescido na rede, quando não é possível chegar até o sistema centralizador ou ocorre uma situação de *lease time* (liberação de portas pelo roteador).

Este mesmo procedimento é utilizado em redes mais complexas, como por exemplo, na Figura 44, onde existem vários roteadores com IP não válidos, e apenas um IP verdadeiro. A comunicação entre os roteadores do exemplo é realizada via Wi-Fi, no entanto é possível utilizar-se de qualquer meio físico.

Figura 44: Estrutura de rede de maior complexidade.



Fonte: do autor.

Uma vez mapeado a estrutura de rede e mapeado as portas de redirecionamento, o sistema centralizador pode acessar os dispositivos a qualquer instante, somente ajustando a lógica de funcionamento da rede, sem a necessidade de *hardware* adicional.

No caso dos dispositivos móveis, sejam eles autônomos ou controlados, podem ser redirecionados com base na inteligência de um sistema centralizador, assim como os objetos estáticos, indiferente dos locais onde estejam. Podem ter momentos de menor ou maior demanda de processamento, dado as características do contexto em que estão inseridos, uma vez que os SO são alcançáveis, o sistema gerenciador poderá modificar a qualquer momento os seus comportamentos.

4.5 Considerações finais

Este capítulo apresentou as características do AdaptThing, bem como um sistema gerenciador que suporta o modelo proposto. Para que modelo contemple os requisitos de adaptação dinâmica, coleta e aquisição de dados de dispositivos sensoriais, aqui chamados de objetos sensitivos - *Sensing Object* (SO) é proposto a arquitetura de software para um sistema que gerencia:

- Históricos de contextos para utilização em processos de inferência, mantendo informações sobre os atributos (dados) e propriedades (configurações) dos SOs;
- Procedimento utilizado para realizar ações de acordo com eventos temporais e recepção de dados;
- Suporte a vários protocolos de coleta e aquisição de dados de uso geral. Caso o sistema não possua suporte nativo, é disponibilizado recursos de *call-back* e interfaces para possibilitar o desenvolvimento e compatibilidade com outros protocolos, middlewares e aplicações;
- Suporte ao protocolo padronizado que possibilita a aquisição, coleta, adaptação de comportamento operacional de SOs;
- Suporte de interface como o usuário.

Todos os SOs que utilizam o protocolo padronizado, mesmo que não seja possível o mapeamento automatizado de rota, seja por limitações de *hardware* ou infraestrutura, possuem um código extra, onde sempre que for enviado um dado ou sinalizado que está ativo e funcionando, o SO aguarda uma resposta do sistema gerenciador, que pode ser um “Ok” ou solicitação de alguma alteração de suas propriedades, momento este em que é realizado a adaptação de comportamento operacional.

5 AVALIAÇÕES E RESULTADOS

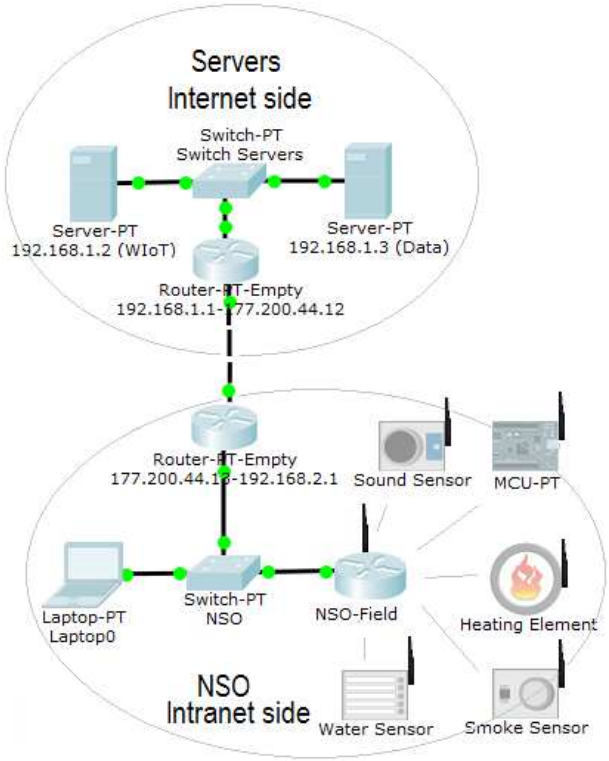
No capítulo anterior foi apresentado o modelo AdaptThing, a arquitetura de software para amparar a sua implementação e o protocolo padronizado para possibilitar a adaptação comportamental dos dispositivos da IoT, intitulados de SOs. Neste capítulo será apresentado a implementação de um sistema gerenciador para validar o modelo proposto.

5.1 Testes iniciais

Para testar o modelo AdaptThing foi desenvolvido um sistema gerenciador que pode ser disponibilizado em uma plataforma computacional que possua suporte à linguagem Java e/ou Python. A escolha pelas duas linguagens é baseada no interesse em verificar qual delas possui maior desempenho e simplicidade no desenvolvimento.

O sistema operacional pode ser Windows ou Linux, podendo ser executado em um computador pessoal, servidor ou nuvem PaaS, desde que possua IP válido ou ambiente simulado. Os testes de mapeamento automatizado foram realizados na ferramenta GNS3, que é um simulador de redes, onde todas as arquiteturas de rede apresentadas na seção 4.4.3 foram simuladas e obtiveram sucesso. Na Figura 45 encontra-se uma das arquiteturas simuladas no GNS3.

Figura 45: Simulação do AdaptThing no software GNS3.



Fonte: do autor.

Os testes iniciais de viabilidade do sistema gerenciador, protocolo padronizado e protocolos não padronizados, foram realizados em plataformas diferentes, porém com ênfase no Sistema Operacional Linux, neste caso:

- Computador Pessoal de alto desempenho: Asus Republic of Gamers (ROG) 8 cores, 16 Gb RAM, HDs 7200 RPM de 1 Tb, 1 porta *Giga Ethernet*, IP restrito, uso em rede local;
- Servidor Dell Blade, usando setup mínimo: 8 cores, 16 Gb RAM, HDs 15.000 RPM de 1 Tb, 4 portas *Giga Ethernet*, IP válido, *link* de 100 Mb. Com setup mínimo, o desempenho é aproximadamente 200 % maior do que o computador pessoal de alto;
- Servidor Dell Blade, composto por 9 lâminas com 32 cores (cada), 128 Gb RAM e *storage* de alto desempenho.

Foram testados a coleta, aquisição e adaptação de comportamento operacional de SOs em uma NSO de pequena escala, onde o banco de dados possuía aproximadamente 37 Gb de informações, proveniente de estações de monitoramento e dispositivos instalados em pontos estratégicos. Com este volume de dados, caso não seja realizado processos de consolidação de informações, o banco de dados leva aproximadamente 25 segundos para obter o *count* da tabela de dados que possui 90.539.158 registros.

O banco de dados utilizado foi Postgresql, a interface com o usuário utilizou Bootstrap 3.0 com recursos de apresentação de gráficos e *dashboard*, amparado pelo *framework* Flask Python e JSP no Java. Ambas implementações disponibilizam um console textual para o usuário via *sockets*, com a finalidade de simplificar o processo de configuração em servidores remotos.

As regras da máquina de regras foram escritas em uma linguagem *script* de autoria própria, que dispara um processo de parser, conforme:

- Na linguagem Java, compila com tools.jar e disponibiliza a classe para ser executada no próprio ambiente, quando necessário (*class loader* e *reflection*);
- No caso do Python, o script é disponibilizado em diretório apropriado e automaticamente pré-compilado para execução.

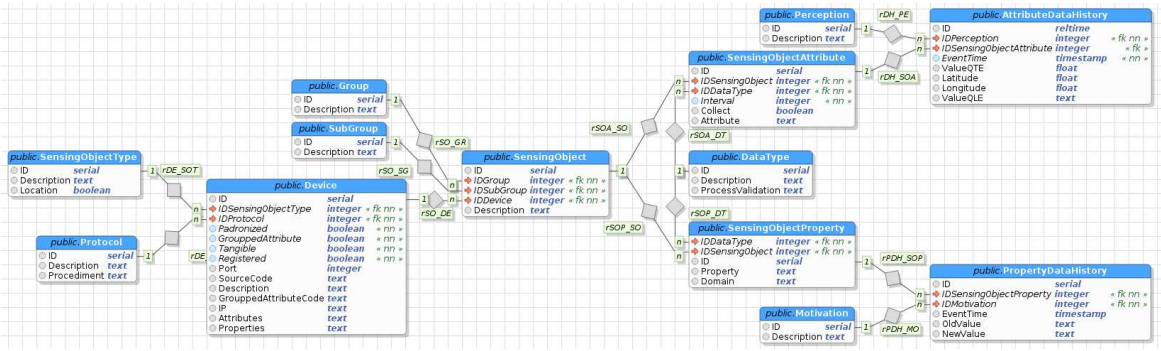
Sempre que um novo evento ocorre, como por exemplo, a chegada de um dado, é feito uma verificação se existe uma instrução para este evento, esta instrução é armazenada em uma tabela *hash* que pode ser fisicamente armazenada em banco de dados ou arquivo de lista indexada, conforme configuração do usuário.

A implementação do modelo AdaptThing, respeitando a arquitetura proposta na Figura 26 é dividida em várias partes, que são armazenamento, máquina de regras e inferência, processo de coleta e aquisição, adaptação de SOs.

5.1.1 Estrutura de dados

Os dados foram estruturados em um banco de dados relacional Postgresql. O arranjo proposto permite armazenar e obter de forma agrupada os dados, para posterior inferência. A estrutura parcial é apresentada na Figura 46.

Figura 46: Estrutura de banco de dados em modelo relacional.



Fonte: do autor.

Analisando a Figura 46 da direita para a esquerda, observa-se as tabelas responsáveis por armazenar o histórico dos atributos, intitulada de *AttributeDataHistory*, e o histórico das propriedades, intitulada de *PropertyDataHistory*. Os campos que armazenam os valores são do tipo "text", no entanto possuem outra tabela que informa o processo de inferência e armazenamento dos dados. Caso o tipo de dado for binário, o campo texto é utilizado para informar o local onde o conteúdo será armazenado.

Os dados dos atributos dos SOs quando chegam ao sistema gerenciador, recebem uma sinalização "Perception", que informa qual a percepção atual dos dados, que por padrão é "normal". No caso das propriedades, também existe uma sinalização chamada de "Motivation", que informa "o porquê" da alteração ou adaptação de parâmetros de configuração.

No meio da imagem é apresentado a tabela "SensingObject", que é responsável pela relação do "Device", que é o SO propriamente dito, com o grupo e subgrupo que ele pertence. Por fim, na parte da esquerda da Figura 46 são observadas características descritivas do tipo de dispositivo e protocolo utilizado para a comunicação.

Além das tabelas apresentadas na Figura 46, existem bases de apoio, que são responsáveis por armazenar temporariamente os dados, de forma a responder rapidamente as chamadas de Aquisição e Coleta.

5.1.2 Rotinas de Aquisição e Coleta de dados

Uma vez que os dados foram adquiridos de um determinado SO através de um protocolo, seja padronizado ou não, é disparada a rotina de pré-processamento que armazena os dados de forma organizada no histórico de contextos. Uma base de apoio auxiliar é necessária para rotinas mais complexas, ou que possuem um conjunto de arquivos, como por exemplo, imagens de satélite, voz, imagens.

As rotinas de "Aquisição e Coleta" podem armazenar diretamente na base de dados principal, desde que os dados já estejam ajustados, ou que demandem pouco poder computacional para sua estruturação e validação. A "base de apoio" armazena os dados em arquivos e de modo a sup0rtar qualquer tipo de dado não suportado por uma base de dados relacional.

Assim que o processo de coleta é concluído são disparadas as rotinas de "pré-processamento", que são responsáveis por extrair e organizar informações sobre os dados adquiridos. O sistema utiliza um agendador que observa periodicamente o diretório raiz da "base de apoio", sempre que possuir um arquivo (arquivo com o nome correspondente ao *hash*). Assim que os dados forem preprocessados, os mesmos são excluídos da "base de apoio", sendo armazenados em uma estrutura específica. Mesmo que o dado seja armazenado preprocessados em local específico, os dados brutos ficam armazenados em uma estrutura chamada de "bruta", que pode ser excluída manualmente. Esse procedimento é importante, pois, como exemplo, um conjunto de informações pode ser extraído a partir de uma inferência, no entanto, informações adicionais podem existir nos dados brutos, que sendo revisitados, podem fornecer mais informações do que originalmente extraídas.

A "base informacional" é utilizada para armazenar os dados resultantes do processo de Extração de Informações (dados otimizados). A Interface com o Usuário permite ver e analisar as informações inferidas. O sistema gerenciador utiliza a base informacional para evitar o reprocessamento de informações de médio e alto custo computacional, que uma vez processados, podem ser recorrentemente consultados, como por exemplo, a extração de informações de uma imagem de satélite, a temperatura média diária de uma região, a umidade média mensal de um SO, a extração de informações de um conjunto SO agrupado.

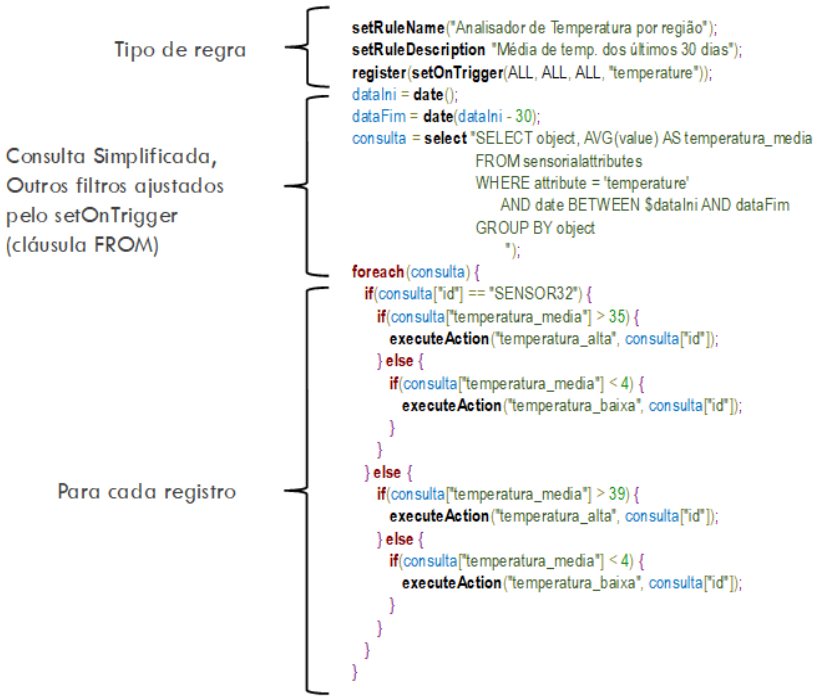
Toda a carga de dados para a interface do usuário é baseada em *lazy load* (carga conforme demanda) e consultas aproximadas, dado o volume de informações. Como já mencionado no início da seção 5.1, a contagem de número de registros cadastrados pode exceder muitos minutos.

5.1.3 Máquina de regras

A máquina de regras é responsável por analisar o histórico de contextos e correlacionar com os dados que estão sendo obtidos pelo sistema gerenciador. Também é utilizada para realizar a extração de informações, possibilitando a existência de dados preprocessados, evitando processamento de grandes massas de informações de forma repetitiva.

No sistema gerenciador, uma regra é formada por um *script* de alto nível, que é preprocessada por um *parser* e compilado para a linguagem Java ou disponibilizado para o Python. Um exemplo de regra é apresentado na Figura 47.

Figura 47: Exemplo de regra executada na máquina de regras.



Fonte: do autor.

O *script* de mais alto nível abstrai a necessidade de conexão com banco de dados, bem como sua estrutura. Todos os recursos da linguagem estão disponíveis, aliados a recursos adicionais. As regras podem ser:

- Trigger: a regra é disparada quando determinada leitura ocorre, considera quatro parâmetros: o id do SO, o id do grupo, id do subgrupo, e o nome do atributo, sendo satisfeito os quatro parâmetros, o *script* da regra é executado;
- Temporal: a regra é agendada, de tempos em tempos é disparado o *script*. O esquema de agendamento utiliza mesma sintaxe da *cron* do Linux;
- Rule: é uma regra que pode ser disparada por outra regra, não possui agendamento ou trigger atrelada, é utilizado como um script de sub-rotinas. Também pode ser um processamento de extração de informações.

5.1.4 Adaptação de SOs

O processo de adaptação depende dos resultados da inferência nos históricos de contextos, ou conforme desejo do usuário administrador. A adaptação dinâmica ocorre quando uma regra é disparada, momento em que o sistema considera as propriedades do SO, e com base nelas é realizado um processo de alteração de valores conforme especificado no domínio da propriedade.

A variação pode ser percentual ou específica. Sempre que ocorre o processo de adaptação, é realizado uma nova leitura dos atributos, caso ocorra uma alternância entra ajuste e leitura em *loop*, o sistema gera uma alerta ao usuário sobre um possível erro ou problemas na configuração.

O processo de adaptação ocorre de forma transparente para os SOs que são padronizados, no entanto é possível definir procedimentos para compatibilizar a adaptação de SOs não padronizados.

5.1.5 Desempenho e ajustes da aplicação inicial

O exemplo utilizado no início da seção 5.1 permitiu verificar as limitações complexidade de implementação do modelo AdaptThing. Os testes de auto mapeamento de rotas foram executados com sucesso. Os maiores problemas foram observados quanto à utilização do banco de dados relacional e na forma de realizar as requisições dado o grande volume de dados.

Na implementação original do sistema *web*, de forma a exibir uma paginação de tela, era solicitado a contagem de número de registros (*count*), para depois realizar chamadas com *offset* e *limit*. Neste procedimento o tempo para realizar o *count* no banco de dados levava mais de um minuto, o que exigiria uma tabela extra para armazenar a quantidade de registros, ou realizar *counts* aproximados.

Para aplicar recursos de análise dos dados através de regras, o tempo de carga dos dados para a memória RAM a partir de um banco de dados relacional, usando *storage* de alto desempenho, levava tempo superior ao processo de inferência em memória, fazendo com que o desempenho do projeto fosse muito inferior ao desejado, sempre que ocorria o processo de consolidação de dados temporal, ou seja, geração de tabelas com base em minutos, horas, dias, semanas e meses.

Para adequar o sistema com base no modelo computacional AdaptThing para funcionamento em *bigdata*, o armazenamento de dados foi realizado utilizando-se recursos da ferramenta Apache Spark com banco de dados Hive com armazenamento de dados em memória. A entrada de novos dados e a consolidação de dados foram realizadas em *stream mode*, ou seja, os dados são reavaliados e os modelos retreinados conforme os dados são recebidos. Utilizando-se destes recursos, análise de dados com métodos inteligentes, como por exemplo, K-Means e outros, tornaram-se mais rápidos e eficientes do que a execução de uma comparação de um valor em relação aos seus limites previamente cadastrados em banco de dados.

5.2 Aplicação em plataformas climáticas

Neste cenário foi implementado um sistema seguindo os preceitos do modelo AdaptThing. O sistema foi implementado utilizando a linguagem Python e Apache Spark (HAZARIKA, RAM and JAIN, 2017), configurado para execução em *cluster*, dado o desempenho da aplicação inicial (seção 5.1.5).

O cenário de aplicação do sistema foi um conjunto de 24 plataformas climatológicas industriais e mais uma para testes de adaptação, espalhadas em várias regiões do Brasil. Dentre o conjunto de sensores SOs, foram utilizados 14 que eram comuns entre as plataformas de monitoramento, neste caso, *inside temperature, inside humidity, outside temperature, outside humidity, pressure, rainrate, rainfall, wind speed, wind direction, wind gust, wind gust direction, wind chill, dew point, cloudheight*.

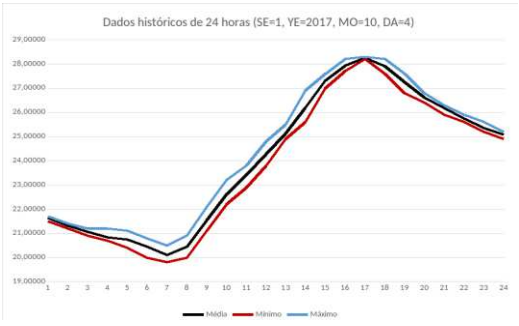
Cada plataforma de monitoramento é considerada pelo sistema uma "station" e os SOs um "sensor". Os dados históricos das plataformas foram importados para o sistema, gerando um histórico de contextos com informações desde 2017, sendo sincronizados com a coleta atual.

O gerenciamento do *cluster* é realizado com Apache Spark e diretórios distribuídos HDFS (HAZARIKA, RAM and JAIN, 2017). Os processos de consolidação temporais de banco de dados geram pequenos picos de execução, mas que não são perceptíveis na arquitetura utilizada, mesmo forçando uma situação de virada de ano, momento em que ocorre o maior número de consolidações de banco de dados juntas, bem como, métodos inteligentes armazenados nos diretórios indexados.

Os dados dos eventos de chegada são recebidos pelo sistema e distribuídos para banco de dados e para o Apache Spark em *stream mode* (HAZARIKA, RAM and JAIN, 2017), que gera a consolidação dos dados para gravação dos modelos inteligentes (ONAL, SEZER, OZBAYOGLU, DOGDU, 2017) em estrutura de diretórios indexados. Quanto à consolidação de média e desvio padrão, é realizado pelo próprio banco de dados, gerando *inserts* de *selects* em um processo rápido e de baixo custo computacional.

Na Figura 48 são apresentados os valores de um SO (*outside temperature*) de 24 horas. Os dados foram lidos em intervalos de 5 segundos, totalizando 17.280 valores. Neste caso, o SO estava programado a enviar os dados neste intervalo para poder gerar um teste visual, de forma a comparar o processo normal com o processo adaptado.

Figura 48: Leitura de dados de 24 horas de um SO.



Fonte: do autor.

Na linha central são apresentados os valores médios lidos no período, e as linhas laterais os extremos lidos. Assim como no exemplo da Figura 48, é possível calcular a média e os desvios em períodos mais curtos, como minutos, horas, dias, meses, anos, poderia ainda, ser calculado em períodos de segundos e milésimos, no entanto, para o cenário proposto não foi necessário.

Caso um valor lido tenha um desvio padrão acima ou abaixo do calculado, ou até mesmo o valor não exista no histórico para o mesmo período, é analisado a média e o desvio padrão no minuto, não estando normal, na hora, dia, mês, de acordo com as configurações do sistema, neste caso específico, é analisado a média e desvio no minuto. Caso ainda esteja fora dos padrões, é analisado as 5 últimas leituras e verificado o desvio padrão entre elas. Por fim, caso ainda esteja fora do padrão, é disparado um método inteligente para analisar o evento.

Como pode ser percebido, é esperado que os valores sempre estejam normais, realizando a adaptação somente quando for necessário.

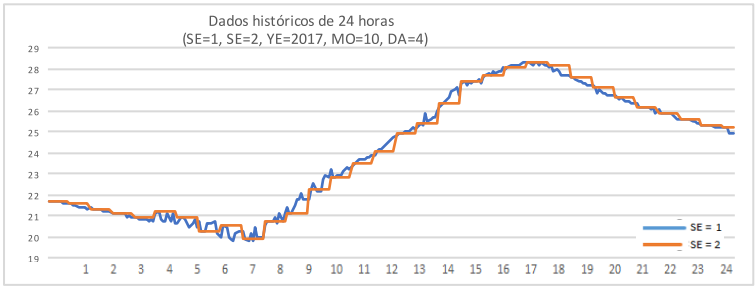
Neste caso, os métodos inteligentes implementados para todos os SOs foram o K-Means e Árvore de Decisão utilizando-se os recursos da MLlib do Apache Spark (ONAL, SEZER, OZBAYOGLU, DOGDU, 2017), porém sua ativação não apresentou impactos na arquitetura ou melhorias nos resultados. Provavelmente em aplicações mais complexas onde a média a desvio padrão não sejam o suficiente, as técnicas inteligentes devem fornecer melhores resultados.

O tempo utilizado para verificação de normalidade e anormalidade dos dados, seja por métodos estatísticos ou métodos inteligentes não impacta na arquitetura, porém a primeira consolidação dos dados históricos, afeta o desempenho do sistema, nas duas modalidades.

Destaca-se com uma vantagem usar os métodos inteligentes sobre o método estatístico, quando um utilizado os recursos do Apache Spark - Hive, comparado com banco de dados relacional. No caso do Apache Spark - Hive, ele trabalha em *stream mode*, onde os modelos vão sendo retreinados conforme os dados são inseridos. No caso da estatística no banco de dados relacional, os dados devem ser recalculados de tempos em tempos, existindo momentos onde o consumo computacional é maior.

Para o experimento, foram posicionados 3 SOs próximos (SO 1, SO 2 e SO 3). O tempo entre leituras do SO 2, foi programado para ser a cada 5 minutos, já o SO 1 a cada um minuto. Na Figura 49 pode ser observado a leitura de ambos sem adaptação, com os valores medidos sincronizadas por horários, repetindo os valores de SO 2 em 5 vezes para cada 1 de SO 1.

Figura 49: Leitura de dados de 24 horas de um SO sem adaptação.



Fonte: do autor.

Na Figura 50 são apresentados os valores do SO 3, o qual estava habilitado para realizar o processo de adaptação sempre que houvesse uma variação acima do desvio padrão histórico ou desvio padrão dos SOs “friendly”. O processo de adaptação modifica automaticamente as propriedades do SO, neste caso, a temporização de envio será de um minuto, porém poderia ser uma proporcionalidade como definido por Ku, Park and Choi (2017).

Conforme a Figura 50, marcados com círculos, foram realizadas intervenções do processo de adaptação em SO 3. Uma vez disparado o processo de adaptação, foi modificado o intervalo de leituras de 5 minutos para 1 minuto, e posteriormente, depois de "n" leituras sem estar fora do desvio padrão histórico, juntamente a outros processos de identificação de anormalidade, o SO é adaptado novamente voltando para seu funcionamento padrão com intervalos de 5 minutos.

Figura 50: Leitura de dados de 24 horas com adaptação.

Fonte: do autor.

Conforme a Figura 50, o SO 3 consome menos recursos em momentos de normalidade, por outro lado mais recursos em momentos de anormalidade, permitindo assim informações mais detalhadas do evento, com menor consumo de recursos, sejam de comunicação, energia ou processo computacional.

5.2.1 Resultados obtidos

Para apresentar os resultados da contribuição deste trabalho, foi analisado somente um caso específico em um intervalo de 24.

Foi observado que sempre que um conjunto de SOs possuía um SO amigado, todos demais poderiam trabalhar na forma mais econômica possível, no entanto, se um SO do contexto identificasse uma anormalidade, todos os SOs eram adaptados para verificar se a anormalidade era confirmada, uma vez sendo confirmada, todos alteravam seu funcionamento até que os valores estivessem dentro do desvio padrão histórico. Os dados que geraram a Figura 50 permitiram criar a análise da Tabela 12.

Tabela 12: Comparativo do desempenho dos SOs.

	TL	HMax	HMin	ErrM t	Cc
Com adaptação	1 s	04:00-08:00 08:30-12:00 14:30-15:00	08:00-04:00 08:00-80:30 12:00-14:30 15:00-24:00	0 %	33 %
Sem adaptação	5 s	---	00:00-24:00	11 %	8 %

Fonte: do autor.

Onde, *TL* corresponde ao tempo de leitura, *HMax* os horários de adaptação, *HMin* os horários de normalidade, *ErrM t* percentual do maior erro de leitura obtido em relação ao valor detalhado, *Cc* percentual de consumo computacional.

Os resultados indicam que o SO com adaptação consumiu aproximadamente 33% a mais de recursos computacionais em relação ao SO sem adaptação, no entanto houve um ganho de detalhamento de até 11%, partindo do pressuposto que a leitura do SO deveria ser o valor medido pelo SO de menor intervalo.

Caso os valores sejam entendidos de forma inversa, ou seja, o ideal é ter mais leituras pelo fato de necessitar dados mais detalhados, pode ser considerado uma economia de 67% de recursos computacionais, pois em 33% do tempo houve mais leituras, e 67% do tempo uma maior ociosidade.

A situação considera somente dois dispositivos, situação que pode ser replicada para todos os SOs amigados, aumentando o percentual de economia proporcionalmente de acordo com a quantidade envolvida.

O ideal é realizar com que os SOs amigados possuam o processo de coleta e aquisição de dados em momentos diferentes dentro de um intervalo definido, permitindo perceber melhor o ambiente, assim o detalhamento de informações não é tão afetado, e sempre que um perceber alguma alteração, pode adaptar o funcionamento dos demais.

5.2.2 Conclusão da aplicação

Para coleta e aquisição dos dados, foram utilizados protocolos baseados em *gets* e *sets* e o protocolo específico apresentado na seção 4.4.1 da pág. 94. O uso do protocolo não forneceu ganhos no sentido de otimização ou velocidade de processamento, mas em facilidades e simplificação no processo de definição das funcionalidades e parâmetros de comportamento funcional.

A possibilidade da verificação de SOs amigados, ou seja, elementos do mesmo contexto, com proximidade geográfica e/ou variação de valores similares, é algo que pode ser mais explorado, sendo mais uma forma de verificar a normalidade de funcionamento de elementos da

IoT. Propõem-se para trabalhos futuros a busca por formas de automatizar quais SOs podem ser considerados amigados e em que momentos.

O processo com maior custo computacional envolvido nesta implementação, refere-se ao processo de consolidação de dados, que pode ocorrer conforme configuração do usuário. O tempo e consumo de recursos varia conforme o volume de dados, granularidade dos períodos de consolidação, métodos inteligentes utilizados e *hardware* disponibilizado para o processamento.

Considera-se que o modelo é adequado para a aplicação proposta, podendo ser aplicado a outras situações onde dados de históricos de contextos podem ser levados em consideração para a verificação de novos eventos, bem como aumentar o detalhamento dos contextos.

5.3 Aplicação em sala de aula

O processo de ensino e aprendizagem é um processo contínuo que demanda empenho e dedicação, tanto do educador quanto do estudante. Para que este processo tenha maior êxito, é importante o uso de ferramentas modernas e mais atrativas, que possam facilitar a compreensão de conteúdos necessários para a formação, bem como avaliar o rendimento dos estudantes.

O Exame Nacional de Desempenho de Estudantes (ENADE) avalia o rendimento dos estudantes concluintes dos cursos superiores de graduação, através de materiais avaliativos que são desenvolvidos pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), pertencente ao Ministério da Educação (MEC). Segundo o INEP (2019), o ENADE avalia o rendimento dos concluintes dos cursos de graduação, em relação aos conteúdos programáticos, habilidades e competências adquiridas em sua formação. O exame é obrigatório e a situação de regularidade do estudante deve constar em seu histórico escolar (INEP, 2019). Segundo Oliveira e Silva (2018), o desempenho dos estudantes é um dos fatores associados ao sucesso do ensino-aprendizagem, e ressalta-se a importância da verificação do que influencia para a formação de um diagnóstico sobre as condições enfrentadas nos ambientes de ensino.

O ENADE é um exame aplicado em papel e caneta, mais precisamente, segundo o edital número 40 de 2018, item 16.8, “As respostas da prova objetiva e da prova discursiva deverão ser transcritas com caneta esferográfica de tinta preta, fabricada com material transparente, nos Cartões-Resposta”.

Dentre as possibilidades de estudos para o ENADE, a investigação de alternativas viáveis de administração e redução da prova são particularmente interessantes, pois tendem a reduzir o custo, agilizar o processo e tornar mais fiáveis as avaliações. Neste contexto, os testes informatizados adquirem relevância, já que possibilitam uma avaliação reduzida e com estimativas das habilidades dos estudantes. Conforme Santana et al. (2017), a redução é possível pela adaptação do questionário através de Testes Adaptativos de Itens (TAI).

Os TAIs são administrados via dispositivos computacionais, no caso deste artigo, os itens são apresentados como questões, que adaptam-se ao grupo que está respondendo ao teste, buscando encontrar um teste adequado, ou seja, um teste aplicado com o mínimo de perguntas possível, de acordo como conhecimento médio do grupo. Para isso, faz-se necessário estimar a proficiência dos indivíduos de forma sincronizada, para então ter a média do grupo, e, assim, selecionar os itens que mensurem eficientemente a proficiência do grupo examinado.

Segundo Costa (2009), a noção básica de um teste adaptativo é imitar automaticamente o que um examinador faria, ou seja, um TAI tem por intenção administrar itens, de um banco de itens previamente calibrados, que satisfaçam ao nível de capacidade do examinando. Conforme Santana et al. (2017), os itens do TAI são escolhidos de acordo com o modelo da Teoria de Resposta ao Item (TRI), que é adotado para descrever o comportamento da resposta do indivíduo.

Neste cenário de aplicação do AdaptThing foi criado um *software* responsável por adaptar dinamicamente o comportamento operacional elementos da (IoT) com base no seu histórico de contextos, com a finalidade de gerenciar o questionário do TAI considerando o TRI em grupo. Os dispositivos computacionais utilizados para responder o questionário, foram considerados como dispositivos da IoT virtuais.

O *software* por padrão estava programado para fornecer 6 questões do ENADE sobre assuntos específicos que versavam sobre sistemas operacionais, programação paralela e distribuída e *software* básico, porém, sempre que a média da turma fosse menor que a média nacional do ENADE, o *software* fornecia outras questões similares, para aprofundar o assunto tratado. Indiferente do resultado das questões terem sido certas ou erradas, o *software* apresentava um *feedback* do porquê da resposta, auxiliando assim o processo de aprendizado.

O *software* do *backend* foi implementado para demonstrar as capacidades do modelo computacional AdaptThing, que monitora e gerencia diferentes elementos da IoT, neste caso, as questões respondidas pelos estudantes.

A validação do *software* foi realizada através da aplicação do mesmo em dois grupos de estudantes, um formado por 15 e outro com 23 integrantes. Foram criadas 24 questões de assuntos variados, através de buscas em assuntos diversos, sendo que foram consideradas 6 questões "chave", simulando as questões do ENADE, e para cada uma das questões "chave", outras 3 questões complementares. Para a média de corte, ou seja, equivalência da média nacional, foram atribuídos valores aleatórios entre 4 e 9. Os valores de corte foram modificados entre o primeiro e segundo grupo de validação. Houve a necessidade de adequações, principalmente quanto ao processo de liberação de questões de forma sincronizada, onde somente poderia ser apresentada a próxima questão, uma vez que todos haviam respondido à questão atual.

Para a pesquisa de campo, foi aplicado o *software* adaptativo para um conjunto de 22 estudantes de cursos de TI, que já haviam realizado as disciplinas de sistemas operacionais, *software* básico e em fase final da disciplina de programação paralela e distribuída. O *software* visou obter as respostas da turma e adaptar o questionário, fornecendo outras questões de assunto similar, com três finalidades, primeiramente obter subsídios para medir o conhecimento médio da turma, por segundo, direcionar as questões complementares, e terceiro, permitir que os estudantes tivessem maior familiaridade com a estrutura das questões do ENADE possibilitando aprofundar seus conhecimentos.

Conforme apresentado no capítulo 4 o AdaptThing é um modelo computacional e não um *software*, no entanto é um modelo que pode ser adequado para qualquer aplicação que envolve IoT. Assim sendo, foi desenvolvido dois *softwares*, um para o gerenciamento dos SOs e outro para fornecer os SOs.

O *software* que gerencia os SOs possui a necessidade de analisar o contexto, ou seja, os dados do grupo, para chegar a uma medida de comparação com o histórico, que nesse caso é a média nacional do ENADE. Para isso foi necessário gerenciar o acesso e liberação de questões de

forma sincronizada, similar a leitura de dados de um conjunto de sensores. No primeiro momento, o *software* aguarda por um tempo para conhecer quais elementos, neste caso os integrantes farão parte do grupo, encerrado este procedimento, sempre é aguardado pela resposta dos integrantes. Existe uma funcionalidade no *software*, que ignora o participante caso não consiga a comunicação com mesmo ou extrapole um tempo pré-definido. A condução das respostas, deriva da avaliação média do grupo, apesar de ficarem armazenados os dados de cada integrante.

Para a aplicação do usuário, foi criado um *software* multiplataforma, onde eram respondidas as questões de forma sincronizada, sempre aguardando pelo próximo passo, iniciado pela construção do grupo e sucessivamente a aplicação do questionário.

5.3.1 Estudo Proposto

O objetivo desse trabalho foi disponibilizar questões do ENADE para um grupo de estudantes de forma dinâmica e adaptativa de acordo com o conhecimento médio do grupo em estudo. Buscou-se familiarizar os estudantes com a estrutura das questões do ENADE, e através da análise das respostas, sugerir questões complementares para compreenderem melhor o assunto tratado. Para atingir os objetivos, foi desenvolvido um *software* considerando as premissas do AdaptThing, onde um conjunto de questões foram disponibilizadas de forma sincronizada.

Na Tabela 13 é apresentado a origem das questões do ENADE e as médias nacionais de alunos concluintes, que foram obtidas do Relatório INEP de Desempenho de Cursos, que são disponibilizadas para as Instituições de Ensino Superior (IES). Também são apresentadas as médias alcançadas pelo grupo de estudantes que responderam o questionário. As questões foram extraídas de provas já realizadas que versavam sobre 3 assuntos, sistemas operacionais, programação paralela e distribuída e *software* básico. Uma vez escolhidas as questões, a ordem de apresentação foi aleatória para o grupo, porém todos respondiam de forma sincronizada a cada questão.

Tabela 13: Origem das questões do ENADE e médias de desempenho.

	Questões ENADE Componente Específico: Objetivas	**Ano**	**Num.**	**Média Nacional**	**Média da Turma**
Q1	Ciência da Computação - Bacharelado	2017	31	33,3 %	59,1 %
Q2	Engenharia da Computação	2017	22	52,5 %	59,9 %
Q3	Ciência da Computação - Sistemas de Informação - Bacharelado	2005	22	23,5 %	40,9 %
Q4	Computação - Ciência da Computação - Engenharia da Computação	2005	41	*90,0 %	36,4 %
Q5	Computação - Ciência da Computação - Engenharia da Computação	2005	42	34,5 %	36,4 %
Q6	Ciência da Computação - Bacharelado	2017	35	31,0 %	22,7 %

Fonte: do autor.

A questão Q4, número 41 de 2005, apresentada Tabela 13, por algum motivo foi desconsiderada no ENADE, assim, para fins de forçar a apresentação de questões complementares foi colocado sua média nacional em 90,0 %.

No experimento, foram apresentadas um total de 6 questões do ENADE e mais 6 questões complementares, pois nas questões Q4 e Q6, o grupo obteve a média inferior à média nacional. O tempo médio mensurado para cada questão foi de 4,2 minutos, tanto para as questões do ENADE, quanto as questões complementares. A questão seguinte era apresentada assim que o último estudante finalizada a questão. O sistema verificava a média do grupo, e dado a comparação com o valor histórico, neste caso a média nacional, era disponibilizado uma nova questão do ENADE ou então 3 questões complementares.

As questões complementares obtiveram uma média de acerto superior ao das questões do ENADE, no entanto, existem duas constatações, uma que os estudantes não estão acostumados a este tipo de questão, e realmente confundem conceitos.

5.3.2 Conclusão

O segundo cenário apresentou uma aplicação de um *software* educacional, que segue as especificações do AdaptThing, que permite adaptar dinamicamente o comportamento operacional de elementos da IoT, sejam eles físicos ou virtuais.

A aplicação teve como finalidade disponibilizar questões do ENADE para um grupo de estudantes de forma sincronizada, considerando as respostas como valores obtidos de elementos virtuais da IoT. O *software* aguardava a resposta de todos inscritos para determinada questão, somente liberava a próxima, após todos terem concluído a questão atual. Para cada questão resolvida, os resultados eram comparados com a média nacional dos estudantes, ou seja, com valores históricos já conhecidos. Sempre que a média nacional foi maior que a média da turma, eram ofertadas mais três questões de assuntos similares, de forma a compreender melhor o assunto tratado. Indiferente de o estudante acertar ou errar as questões, sejam do ENADE ou as questões complementares, sempre foi fornecido um *feedback* explicando o porquê da resposta certa e das questões erradas.

O *software* adapta a prova conforme a média da turma, de forma a não ficar repetindo certo tipo de questão que estatisticamente está bem compreendido, ou seja, possui compreensão estatística maior que a média nacional. Trabalhos futuros podem explorar as possibilidades de resposta erradas, no entanto, pelo fato de *software* informar um *feedback*, não foi realizado maiores avaliações sobre a distribuição do erro.

Os estudantes consideraram a lógica do *software* uma forma de evitar a repetição de questões que possuíam o conteúdo já dominado pela maioria da turma (supostamente). Outro ponto positivo que foi salientado é a possibilidade de resolução das questões com qualquer dispositivo que tivesse conectividade com a Internet, no entanto a maioria preferiu usar um computador, principalmente pelas imagens existentes nas questões.

Considera-se que o modelo AdaptThing é genérico o suficiente para ser aplicado a diferentes cenários de atuação, inclusive o cenário da educação. A aplicação considerou os dados

dos respondentes como objetos da IoT, adaptando o cenário de questões, como adaptaria o funcionamento de qualquer dispositivo, seja físico ou virtual.

Os resultados do *software* produzido podem ser utilizados para estudantes concluintes, ou até mesmo em disciplinas dos cursos superiores como ferramenta auxiliar para familiaridade com a estrutura de questões do ENADE, além de permitir aos estudantes compreender melhor os assuntos tratados, dados o *feedback* após cada questão.

6 CONCLUSÃO

Esta tese propôs um modelo computacional intitulado de AdaptThing que realiza a adaptação do comportamento operacional de funcionamento de dispositivos da IoT com base no histórico de contextos. O modelo utiliza duas estruturas de dados, uma chamada de Atributos, que armazena dos dados provenientes dos dispositivos da IoT e outra estrutura chamada de Propriedades, que armazena os dados referentes a configuração dos dispositivos.

Foi proposto a utilização dos termos *Sensing Objects* (SO) para designar qualquer tipo de dispositivo da IoT, assim como *Network Sensing Objects* (NSO) uma rede que compreende o agrupamento de todos estes objetos. A NSO gerencia três categorias de SOs, que são os *Static Sensing Objects* (SSO), que são compostos por objetos da IoT estáticos, ou seja, que não modificam sua posição geográfica ao longo do tempo; *Mobile Sensing Objects* (MSO), constituída por SOs que possuem sua posição geográfica modificada ao longo do tempo; *External Sensing Objects* (EXO), formados por provedores de serviços externos ao sistema.

O modelo computacional foi aplicado a diferentes formas de aplicação, mostrando-se versátil e cobrindo um diferente leque de aplicações, como testes *in door* através de simulações computacionais, através da ferramenta GNS3 e aplicações de *software*, também foi aplicado no gerenciamento de estações climáticas industriais e um caso educacional.

As principais contribuições desta tese são a definição de um modelo computacional genérico para realizar a adaptação de dispositivos da IoT baseados em históricos de contextos, e um protocolo padronizado para coleta, aquisição e adaptação de dispositivos da IoT.

6.1 Trabalhos futuros

Sugere-se como trabalhos futuros a documentação dos dispositivos da IoT, abordando os padrões de comunicação e os novos modelos de comunicação. Dentre outras possibilidades de estudos, sugere-se:

- Aprofundar a análise de dados proporcionada pelos SOs "friendly", ou seja, SOs que possuem variação similar, estando dentro de uma área geográfica próxima;
- Identificar automaticamente o que é uma área geográfica próxima e quais e quando os SOs são "friendly", questão de temporalidade;
- Aprofundar os experimentos de identificação de normalidade, anormalidade ou intervalos com o uso de *Machine Learning*;
- Experimentar outras ferramentas de *Bigdata* e *Machine Learning* além do Apache Spark;
- Criar um protocolo de roteamento multi nível, similar ao desenvolvido nesta Tese, porém para endereços IPv6 que são utilizados em Intranet (redução de custos), possibilitando o alcance de dispositivos de forma bidirecional.

6.2 Artigos publicados relacionados à Tese

VIELITZ, Felipe L.; MARTINS, Márcio G.; BARBOSA, J.; OLIVEIRA, Kleinner F.; DIAS, Lucas P. S.; WOLF, Alexandre. CMFRAME: a framework for managing dynamic and hierarchical context histories. CLEI 2019 - XLV CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA.

WOLF, A., & BARBOSA, J. L. V. (2019). An IoT dynamic and adaptive study management system for collective knowledge that uses the ENADE's questions. International Journal for Innovation Education and Research, 7(9), 241-250. Disponível em: https://doi.org/10.31686/ijier.Vol7.Iss9.1741.

ARANDA, J., A., S.; DIAS, L. P. S.; WOLF, A. S; CARVALHO, J., V.; TAVARES, M., C.; YAMIN, A., C., and BARBOSA, J., L., V. 2019. Towards a model to optimized collect of vital signs through adaptive strategies. In Proceedings of the 25th Brazillian Symposium on Multimedia and the Web (WebMedia '19). ACM, New York, NY, USA, 97-100. Disponível em: https://doi.org/10.1145/3323503.3360645.

SOUZA, Rodrigo; LOPES, João L. B.; CARDOZO, Anderson; CARVALHO, T. R.; DAVET, P.; WOLF, Alexandre Stürmer; BARBOSA, Jorge; YAMIN, Adenauer Correa; GEYER, Cláudio Fernando Resin. Uma Arquitetura para IoT Direcionada à Ciência do Contexto Baseada em Eventos Distribuídos. In: Simpósio Brasileiro de Computação Ubíqua e Pervasiva, 2016, Porto Alegra. Anais do XXXVI congresso da sociedade brasileira de computação. Porto Alegre: Sociedade Brasileira de Computação, 2016. p. 1206-1215.

REFERÊNCIAS

AAZAM M.; NUH E. N. Fog Computing and Smart Gateway Based Communication for Cloud of Things. International Conference on Future Internet of Things and Cloud, Barcelona, 2014, pp. 464-470. Disponível em: http://doi.org/10.1109/FiCloud.2014.83.

ABOWD, G. D. What Next, Ubicomp?: celebrating an intellectual disappearing act. In: ACM CONFERENCE ON UBIQUITOUS COMPUTING, 2012., 2012, New York, NY, USA. Proceedings. . . ACM, 2012. p. 31–40. (UbiComp '12).

ALAM M.; NIELSEN, R. H.; PRASAD N. R., The evolution of M2M into IoT. First International Black Sea Conference on Communications and Networking (BlackSeaCom), Batumi, 2013, pp. 112-115. Disponível em: https://doi.org/10.1109/BlackSeaCom.2013.6623392.

AMEYED, D.; MIRAOUI, M.; TADJ, C. A Survey of Prediction Approach in Pervasive Computing. International Journal of Scientific & Engineering Research, [S.l.], v. 6, n. 5, p. 1–11, 2015.

ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: a survey. Computer Networks, [S.l.], v. 54, n. 15, p. 2787 – 2805, 2010.

BAHGA A and MASDISETTU V. Internet of Things: A Hands-On Approach. VPT/Create-Space Inc, Atlanta.

BALA, A.; CHANA, I. Intelligent Failure Prediction Models for Scientific Workflows. Expert Syst. Appl., Tarrytown, NY, USA, v. 42, n. 3, p. 980–989, Feb. 2015.

BALBINOT, A., BRUSAMARELLO, V. Instrumentação e fundamentos de medidas - vol.1. Rio de Janeiro, Brasil: LTC, 2010.

BALLINGS, M.; POEL, D. Van den; HESPEELS, N.; GRYP, R. Evaluating Multiple Classifiers for Stock Price Direction Prediction. Expert Syst. Appl., Tarrytown, NY, USA, v. 42, n. 20, p. 7046–7056, Nov. 2015.

BARBOSA, J. L. V. Multi-Temporal Aspects on Contextual Variability Modeling. In: XI Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP), 2019, Belém. Anais do SBCUP 2019. Porto Alegre: SBC, 2019. p. 1-10.

BARBOSA, J. L. V. Ubiquitous computing: applications and research opportunities. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTATIONAL INTELLIGENCE AND COMPUTING RESEARCH (ICCIC), 2015, 2015. Anais. [S.l.: s.n.], 2015. p. 1-8.

BARBOSA, J., TAVARES, J., CARDOSO, I., MOTA, B., MARTINI, B. TrailCare: an indoor and Outdoor Context-aware System to Assist Wheelchair Users. International Journal of Human-Computer Studies, vol. 116, pp. 1-14, 2018. Disponível em: https://doi.org/10.1016/j.ijhcs.2018.04.001.

BAUR, D.; SEIFFERT, F.; SEDLMAIR, M.; BORING, S. The Streams of Our Lives: visualizing listening histories in context. IEEE Transactions on Visualization and Computer Graphics, [S.l.], v. 16, n. 6, p. 1119–1128, Nov 2010.

BISWAS, A. R. and GIAFFREDA, R. IoT and cloud convergence: Opportunities and challenges. IEEE World Forum on Internet of Things (WF-IoT), Seoul, 2014, pp. 375-376. Disponível em http://www.doi.org/10.1109/WF-IoT.2014.6803194.

BOLZANI A. M. Análise de Arquitetura e Desenvolvimento de uma Plataforma para Residências inteligentes.

BOLZANI, C. A. M. Análise de Arquitetura e Desenvolvimento de uma Plataforma para Residências inteligentes. 2010.

BRERETON, P.; KITCHENHAM, B., A.; BUDGEN, D., T.; MOHAMED, M.; MOHAMED, K. Lessons from applying the systematic literature review process within the software engineering domain. JSS 80, 571-583, 2007.

BUZETO, F. N.; CAPRETZ, M. A. M.; CASTANHO, C. D.; JACOBI, R. P. uOS: a resource rerouting middleware for ubiquitous games. In: IEEE 10TH INTERNATIONAL CONFERENCE ON UBIQUITOUS INTELLIGENCE AND COMPUTING AND 2013 IEEE 10TH INTERNATIONAL CONFERENCE ON AUTONOMIC AND TRUSTED COMPUTING, 2013, 2013. Anais [S.l.: s.n.], 2013. p. 88–95.

CAMBRUZZI, W. L.; RIGO, S. J.; BARBOSA, J. L. V. Dropout Prediction and Reduction in Distance Education Courses with the Learning Analytics Multitrail Approach. JUCS, [S.l.], v. 21, n. 1, p. 23–47, 2015.

CASAGARAS. Final report. European Technology Platform on Smart Systems Integration. Internet of things in 2020, report of Beyond RFID - the Internet of Things, Brussels

CASTILLEJO, E. Modeling users, context and devices for Ambient Assisted Living environments. Sensors, v. 14, n. 3, p. 5354-5391, 2014.

CERVO, A. L.; BERVIAN, P. A. Metodologia científica. 5. ed. São Paulo: Prentice Hall, 2002.

CHEE, B., J., S.; FRANKLIN J., C. Computação em Nuvem – Cloud Computing. São Paulo: M. Books, 2013.

CHEN, L. Semantic Smart Homes: Towards Knowledge Rich Assisted Living Environments. Studies in Computational Intelligence, v.189, p.279-296. 2009.

CIARAMELLA, A.; CIMINO, M. G. C. A.; LAZZERINI, B.; MARCELLONI, F. Using context history to personalize a resource recommender via a genetic algorithm. In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS DESIGN AND APPLICATIONS, 2010, 2010. Anais. [S.l.: s.n.], 2010. p. 965–970.

COMER, D. E. Redes de Computadores e a Internet, Bookman, 2007.

CONCEIÇÃO, D., M. Uso de Drone autônomo para auxílio na comprovação de alarmes. 2018. Monografia (Graduação em Engenharia da Computação) – Universidade do Vale do Taquari - Univates. 2018. Disponível em: http://hdl.handle.net/10737/2324.

COSTA, C. L., OLIVEIRA L., MÓTA, L. M. S. Internet of Things (IOT): um estudo exploratório em agronegócios. VI Simpósio da Ciência do Agronegócio, 2018.

DA ROSA, J. H.; BARBOSA, J. L. V; RIBEIRO, G. D. ORACON: An adaptive model for context prediction. Expert Systems with Applications, 2016.

DEL RIO, L. S., SILVA, J. O. B., AZEVEDO, M. S., PEREIRA E. P., FISCHER I. A., MEDINA, R. D. Proposta de ambientes inteligentes IoT sob a ótica da eficiência energética. Anais do EATI - Encontro Anual de Tecnologia da Informação, 2018. Disponível em: http://eati.info/eati/anais-2018/Longos/L10.pdf.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. Hum.-Comput. Interact., Hillsdale, NJ, USA, v. 16, n. 2, p. 97–166, Dec. 2001.

DEY, A.; HIGHTOWER, J.; LARA, E. de; DAVIES, N. Location-Based Services. IEEE Pervasive Computing, [S.l.], v. 9, n. 1, p. 11–12, Jan 2010.

DHAMECHA, M., DOBARIA, K. PATALIA T. A Survey on Recommendation System for Bigdata using MapReduce Technology. 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2019, pp. 54-58. Disponível em: http://dx.doi.org/10.1109/ICCMC.2019.8819856.

DRIVER, C.; CLARKE, S. An application framework for mobile, context-aware trails. Pervasive and Mobile Computing, [S.l.], v. 4, n. 5, p. 719 – 736, 2008.

DUBROVA, E. Fault-Tolerant Design: Springer, 2013.

DYBA, T.; JORGENSEN, M.; KINTCHENHAM, B. Evidence-based software engineering for practitioners. IEEE Software, p. 58 – 65. Janeiro, 2005.

FISCHER, G. User modeling in human-computer interaction. User Modeling and User-Adapted Interaction, [S.l.], v. 11, n. 1-2, p. 65–86, 2001. cited By 432.

FRANCO, L. K.; ROSA, J. H.; BARBOSA, J. L.; COSTA, C. A.; YAMIN, A. C. MUCS: a model for ubiquitous commerce support. Electronic Commerce Research and Applications, [S.l.], v. 10, n. 2, p. 237 – 246, 2011. Special Issue on Electronic Auctions: Strategies and Methods.

GEE K. M. Gee and COLLIER M. 6LoWPAN Forwarding Techniques for IoT. IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 888-893. Disponível em: http://www.doi.org/10.1109/WF-IoT.2019.8767174.

GIRTELSCHMID S.; STEINBAUER M.; KUMAR V.; FENSEL A.; KOTSIS G. Big Data in Large Scale Intelligent Smart City Installations. Proceedings of International Conference on

Information Integration and Web-based Applications; Services, New York, NY, USA, 2013, pp. 428:428–428:432.

HANJO, J; BYEONGHWA P.; MINWOO, P.; KI-BON, K; KISEOK C. Big data and rule-based recommendation system in Internet of Things. Cluster Computing, 2017. Disponível em: https://doi.org/10.1007/s10586-017-1078-y.

HARAZIKA A. V., RAM, G. J. S. R. and JAIN E. Performance comparision of Hadoop and spark engine. 2017. International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). Palladam. 2017. 671-674. Disponível em: https://dx.doi.org/10.1109/I-SMAC.2017.8058263.

HIGHTOWER, J.; BORRIELLO, G. Location Systems for Ubiquitous Computing. Computer, Los Alamitos, CA, USA, v. 34, n. 8, p. 57–66, Aug. 2001.

HIGHTOWER, J.; LAMARCA, A.; SMITH, I. E. Practical Lessons from Place Lab. IEEE Pervasive Computing, Piscataway, NJ, USA, v. 5, n. 3, p. 32–39, July 2006.

HONG, J.; SUH, E.-H.; KIM, J.; KIM, S. Context-aware system for proactive personalized service based on context history. Expert Systems with Applications, [S.l.], v. 36, n. 4, p. 7448 – 7457, 2009.

HONG-LING, S. Development of an energy efficient, robust and modular multicore wireless sensor network. Université Blaise Pascal - Clermont-Ferrand II, 2014. Disponível em: https://tel.archives-ouvertes.fr/tel-00968069.

HUSSEIN, D., BERTIN E. and FREY V. A Community-Driven Access Control Approach in Distributed IoT Environments, in IEEE Communications Magazine, vol. 55, no. 3, pp. 146-153, March 2017. Disponível em http://dx.doi.org/10.1109/MCOM.2017.1600611CM.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Instrumentation and Measurement Society. IEEE Standard for a Smart Transducer Interface for Sensor and Actuators - Network Capable Application Processor (NCAP) Information Model (Std. 1451.1). Standards Board. NY: IEEE, 1999. 341 p.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Instrumentation and Measurement Society. IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Data Sheet (TEDS) Formats, (Std. 1451.2). Standards Board. NY: IEEE, 1997. 114 p.

JAMES, A. J, COOPEER, J. and JEFFERY K. Research Directions in Database Architectures for the Internet of Things: A Communication of the First International Workshop on Database Architectures from the Internet of Things.

KAUR, A., SINGH V. P. and SINGH S. G. The Future of Cloud Computing: Opportunities, Challenges and Research Trends. 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)

(I-SMAC), 2018 2nd International Conference on, Palladam, India, 2018, pp. 213-219. Disponível em: http://www.doi.org/10.1109/I-SMAC.2018.8653731.

KITCHENHAM, B. Guidelines for performing Systematic Literature Reviews in Software Engineering. Vol 2.3 EBSB Technical Report, EBSE-2007-01, 2007.

KITCHENHAM, B.; DYBA, T.; JORGENSEN, M. Evidence-based software engineering. In ICSE'04: Procendings of the 26 th International Conference on Software Engineering, p. 273-281, Washington, DC, USA, 2004

KU, T., PARK, W. CHOI H. Choi. Energy information collection mechanism using big data correlation map. 2017. IEEE International Conference on Big Data (Big Data). Boston, MA. 4774-4776. Disponível em: https://dx.doi.org/10.1109/BigData.2017.8258538.

KUROSE, J. Redes de Computadores e a Internet, Addison-Wesley, 2006.

LI, J., XY, X., Xu, CAO, J., DAI, W., ZHANG, J. Indoor Environment Intelligent Monitoring System 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, 2018, pp. 1446-1451. Disponível em: http://dx.doi.org/10.1109/ICMA.2018.8484267.

MANIMOZHI S. and JAYANTHI J. G. IPv6 Mobility Architecture in IPv4 MANET. 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on, Palladam, India, 2018, pp. 443-447. Disponível em: http://www.doi.org/10.1109/I-SMAC.2018.8653730.

MICHAELIS. Moderno Dicionário da Língua Portuguesa. On-line. Disponível em: http://michaelis.uol.com.br/moderno/portugues/index.php. Acesso em outubro, 2018.

MOLIRANI, L. Cloud Computing. A inteligência da nuvem e seu valor em TI. São Paulo: Saraiva, 2018.

MÖLLER, D. P. Introduces the fundamental concepts and design methods in the field of cyber-physical systems. [S.l.]: Springer International Publishing, 2016.

NIST. The NIST Definition of Cloud Computing. 2011. Disponível em: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf . Acessado em outubro 2019.

NORDAHL, M.; MAGNUSSON, B. J. A lightweight data interchange format for internet of things with applications in the PalCom middleware framework. Ambient Intell Human Comput. (2016) 7: 523. Springer-Verlag. Berlin Heidelberg, 2016. Disponível em: https://doi.org/10.1007/s12652-016-0382-3.

NORVIG, P., RUSSELL S. Artificial Intelligence: Pearson New International Edition: A Modern New Approach. 3 ed. Pearson, 2011.

OLIVEIRA, R. R.; CARDOSO, I. M.; BARBOSA, J. L.; COSTA, C. A. da; PRADO, M. P. An intelligent model for logistics management based on geofencing algorithms and RFID technology. Expert Systems with Applications, [S.l.], v. 42, n. 15, p. 6082 – 6097, 2015.

ONAL, A., C., SEZER O. B., OZBAYOGLU M. and DOGDU E. Weather data analysis and sensor fault detection using an extended IoT framework with semantics, big data, and machine learning. 2017. IEEE International Conference on Big Data (Big Data). Boston, MA. 2037-2046. Disponível em: https://dx.doi.org/10.1109/BigData.2017.8258150.

PATEL, K. K.; PATEL, S. M. Internet of Things - IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. International Journal of Engineering Science and Computing, [S.l.], v. 6, n. 5, p. 6122–6131, 2016.

PEJOVIC, V.; MUSOLESI, M. Anticipatory Mobile Computing: a survey of the state of the art and research challenges. ACM Comput. Surv., New York, NY, USA, v. 47, n. 3, p. 47:1–47:29, Apr. 2015.

PIRES, P. F.; CAVALCANTE E.; BARROS T.; DELICATO, F. C; BATISTA T; COSTA B. A Platform for Integrating Physical Devices in the Internet of Things, 2014 12th IEEE International Conference on Embedded and Ubiquitous Computing, Milano, pp. 234-241. 2014. Disponível em: https://doi.org/doi: 10.1109/EUC.2014.42.

PRUTHVI, M., KARTHIKA, S. and BHALAJI, N. "SMART COLLEGE"- Study of Social Network and IoT Convergence. 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on, Palladam, India, 2018, pp. 100-103. Disponível em: http://dx.doi.org/10.1109/I-SMAC.2018.8653787.

REINISCH, C. Thinkhome energy efficiency in future smart homes. EURASIP Journal on Embedded Systems, v. 2011, p. 1-18, January, 2011.

RIOS, F., FLORES, R. A., SOLOIU, V. "Design of an Intelligent Vehicle for Industrial, Office and Home Environments Applications," SoutheastCon 2018, St. Petersburg, FL, 2018, pp. 1-3. Disponível em: http://dx.doi.org/10.1109/SECON.2018.8478862.

ROSA, J. H., BARBOSA, J. L. V., KICH, M. R, and BRITO, L. K. A Multi-Temporal Context-aware System for Competences Management. International Journal of Artificial Intelligence in Education, vol. 25, pp. 455–492, 2015. Disponível em: http://dx.doi.org/10.1007/s40593-015-0047-y.

ROSA, J. H.; BARBOSA, J. L. V.; KICH, M.; BRITO, L. A Multi-Temporal Context-aware System for Competences Management. International Journal of Artificial Intelligence in Education, [S.l.], v. 25, n. 4, p. 455–492, Dec 2015.

SALVADORI, F.; SAUSEN, P. S.; HARTMANN, L. V.; CAMPOS, M.; PADOIN E. L.; LEANDRO G. V. Acquisition and transmission data monitoring system applied to energy

substation. Industrial Informatics, INDIN 2003. Proceedings. IEEE International Conference on, 2003, pp. 60-64. Disponível em: http://dx.doi.org/10.1109/INDIN.2003.1300204.

SANTOS, M. Y.; OLIVEIRA J.; COSTA C., GALVÃO J; ANDRADE C.; MARTINHO B.; LIMA F. V.; COSTA E. Advances in Intelligent Systems and Computing, vol. 570, pp. 175, 2017, ISSN 2194-5357, ISBN 978-3-319-56537-8. Disponível em: https://doi.org/10.1109/SAI.2016.7556139.

SATYANARAYANAN, M. Pervasive computing: vision and challenges. IEEE Personal Communications, [S.l.], v. 8, n. 4, p. 10–17, Aug 2001.

SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. The Case for VM-Based Cloudlets in Mobile Computing. IEEE Pervasive Computing, [S.l.], v. 8, n. 4, p. 14–23, Oct 2009.

SEGATTO, W.; HERZER, E.; MAZZOTTI, C. L.; BITTENCOURT, J. a. R.; BARBOSA, J. Mobio Threat: a mobile game based on the integration of wireless technologies. Comput. Entertain., New York, NY, USA, v. 6, n. 3, p. 39:1–39:14, Nov. 2008.

SEIPPEL, R. G. Transducters, sensors and detectors. Reston publishing company Inc, 1983.

SHELBY, Z.; BORMANN, C. 6LoWPAN: the wireless embedded internet. Edition first, United Kingdom: John Wiley and Sons, Ltd, Publication, 2009.

SONG, M., FISHER R., WANG J., CUI L. Environmental performance evaluation with big data: theories and methods. Annals of Operations Research, 2018, p. 459--472. Disponível em: https://doi.org/10.1007/s10479-016-2158-8.

TAVARES, J.; BARBOSA, J.; CARDOSO, I.; COSTA, C.; YAMIN, A.; REAL, R. Hefestos: an intelligent system applied to ubiquitous accessibility. Universal Access in the Information Society, [S.l.], v. 15, n. 4, p. 589–607, Nov 2016.

TELLES, M. J. MASC: um modelo computacional para cidades inteligentes assistivas. Unisinos. Dissertação de Mestrado, 2016. Disponível em: http://www.repositorio.jesuita.org.br/handle/UNISINOS/5281.

VIANNA, H. D.; BARBOSA, J. L. V. A Model for Ubiquitous Care of Noncommunicable Diseases. IEEE Journal of Biomedical and Health Informatics, [S.l.], v. 18, n. 5, p. 1597–1606, Sept 2014.

VIANNA, H. D.; BARBOSA, J. L. V. A scalable model for building context-aware applications for noncommunicable diseases prevention. Information Processing Letters, [S.l.], v. 148, p. 1 – 6, 2019.

VIANNA, H. D.; BARBOSA, J. L. V.; PITTOLI, F. In the Pursuit of Hygge Software. IEEE Software, [S.l.], v. 34, n. 6, p. 48–52, November 2017.

VIELITZ, F.; MARTINS, M.; BARBOSA, J.; OLIVEIRA, K. F.; DIAS, L.; WOLF, A. CMFRAME: a framework for managing dynamic and hierarchical context histories. CLEI 2019 - XLV CONFERENCIA LATINOAMERICANA DE INFORMÁTICA.

VIVIANI, M.; BENNANI, N.; EGYED-ZSIGMOND, E. A survey on user modeling in multi-application environments

WAGNER, A.; BARBOSA, J. L. V.; BARBOSA, D. N. F. A model for profile management applied to ubiquitous learning environments. Expert Systems with Applications, [S.l.], v. 41, n. 4, p. 2023 – 2034, 2014.

WANG, W.; LEE, K.; MURRAY, D. A global generic architecture for the future Internet of Things. Service Oriented Computing and Applications. p. 329-344, 2017. Disponível em: https://doi.org/10.1007/s11761-017-0213-1.

WEISER, M. The Computer for the 21st Century. Scientific American, [S.l.], v. 265, n. 3, p. 66–75, January 1991

YAMIN A. Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva.

YAMIN, A. Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva. 2004. 195p. Tese (Doutorado em Ciência da Computação). Instituto de Informática, UFRGS, Porto Alegre, RS.

YAMIN, A.; AUGUSTIN, I.; GEYER, C. Exehda middleware: Aspects to manage the isam pervasive environment. XXV International Conference of the Chilean Computer Science Society, 2005.

YINONG, C.; HUALIANG, H. Internet of intelligent things and robot as a service. Simulation Modelling Practice and Theory, p. 159 - 171, 2013. Disponível em: https://doi.org/10.1016/j.simpat.2012.03.006.

ZHENG, L.; ZHANG H.; HAN W.; ZHOU, X.; HE J.; ZANG, Z.; GU Y.; WANG J. Technologies, Applications, and Governance in the Internet of Things, Internet of Things - Global Technological and Societal Trends. Denmark, River Publishers, 2011.

CALDAS-CALLE L., JARA, J., HUERTA M. and GALLEGOS P. QoS evaluation of VPN in a Raspberry Pi devices over wireless network. International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), Cozumel. 2017. Disponível em https://doi.org/10.1109/ICCDCS.2017.7959718.

CHATTERJEE N. R., NETHRA U. and SUMA V. SMARISA: A Raspberry Pi Based Smart Ring for Women Safety Using IoT. International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore. 2018. Disponível em https://doi.org/10.1109/ICIRCA.2018.8597424.

KIM J. and LEE, L. Constructing Infrastructure wireless network using open source. Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna. 2016. Disponível em https://doi.org/10.1109/ICUFN.2016.7537140.

PALAZZI C., E., BRUNATI, M. and ROCCETTI M. An OpenWRT solution for future wireless homes. IEEE International Conference on Multimedia and Expo, Suntec City. 2010. Disponível em https://doi.org/10.1109/ICME.2010.5583223.