Pedro Paulo Vicente Arquiteturas e Protocolos para IoT

UMinho | 2020

Universidade do Minho
Escola de Engenharia

Pedro Paulo Vicente

Arquiteturas e Protocolos para IoT

abril de 2020

Universidade do Minho
Escola de Engenharia

Pedro Paulo Vicente

Arquiteturas e Protocolos para IoT

Dissertação de Mestrado
Mestrado em Engenharia de Redes e Serviços Telemáticos

Trabalho efetuado sob a orientação do
Professor Doutor João Marco Silva
e da
Professora Doutora Solange Rito Lima

abril de 2020

DIREITOS DO AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Agradecimentos

A elaboração desta dissertação para obtenção do grau de mestrado teve apoios e incentivos que permitiram a sua realização. Nesta perspetiva, este espaço é reservado para dedicar aqueles que direta ou indiretamente foram contribuindo para a sua execução.

Agradeço a Deus por me ter dado força e coragem nesta peregrinação científica quando ficava sem forças para continuar. Quando me invocar, hei de responder-lhe; estarei ao seu lado na tribulação para encher de honras [*Salmos* 91:15]

A meu orientador, professor Doutor João Marco Silva, que incansavelmente contribuiu durante a realização desta dissertação. A partilha de conhecimento, experiência, empenho e confiança depositada em mim para que a realização deste trabalho chegasse a fase terminal. Gratidão a minha coorientadora professora Doutora Solange Rito Lima pelo calor depositado no exercício desta peregrinação científica.

Aos meus pais e irmãos, Inês Filomena Losso, Ambrósio D´Oliveira, Rosa Filomena, Anita, Gervásio Biano, Love Joy, Schnneider, Yolanda, Gugas e Mila, obrigado por estarem presentes nos momentos críticos da minha vida, pelo apoio, amizade e afeto demonstrado durante este período vivido geograficamente separado deles.

Agradeço à minha família, começando pela minha querida esposa, Domingas Madalena Suami, pelo grande apoio, amizade e amor incondicional demonstrado e por ser um pilar primordial em cada jornada vivida e obstáculos a serem ultrapassados. Aos meus filhos (Carlos, Arcânjo, Miroslav e Paulo) que ficaram privados do calor do pai por esta aventura abraçada pela ciência.

Aos meus amigos, António Puindi, Paulina Suquina, Francisco dos Santos, Arão de Sousa, Maria Tomás e Gerson Benjamim, que contribuíram com o seu apoio psicológico nas circunstâncias que a vida me foi proporcionando. Sem a vossa disponibilidade e compreensão, acima de tudo, não seria possível ter finalizado este trabalho. Com a peregrinação científica que hoje termino, espero que possa corresponder com esta entrega a dedicação que me foram dando nesta trajetória. A todos vocês, dedico este trabalho.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

A *Internet* das Coisas (*Internet of Things - IoT*) é um conceito inovador, que assenta na utilização de diversas tecnologias, tendo sido particularmente impulsionada pelo aparecimento das Redes de Sensores Sem Fios (*RSSF*) e pelos cenários práticos em relação à sua aplicabilidade. São exemplos destes cenários a monitorização ambiental, em ambientes domésticos, na indústria, na saúde, entre outros.

As *RSSF* encontram-se em constante desenvolvimento para contextos de aplicação onde diversos dispositivos estarão interligados à *Internet*, dando assim suporte ao conceito de diversidade subjacente à *IoT*. No entanto, os dispositivos sensores envolvidos em *RSSF* dispõem normalmente de capacidades limitadas, nomeadamente em termos de memória, processamento e restrições energéticas para o exercício das suas funções. Desta forma, a comunidade científica e a indústria têm proposto arquiteturas e protocolos específicos para operar em redes envolvendo estes dispositivos de baixa capacidade e promover a sua interligação à *Internet*.

Neste contexto, este trabalho tem por objetivo fazer um estudo detalhado das arquiteturas e protocolos subjacentes à *IoT*, relacionando-os com os protocolos convencionais da *Internet* e estudando os respetivos cenários de convergência. Assim, apresenta uma abordagem histórica relativamente aos modelos e arquiteturas de *Internet* e à influência destes na *IoT*, contextualiza os modelos e arquiteturas especificamente propostos para *IoT* e, em particular, explora os aspetos de convergência entre os protocolos das distintas tecnologias de rede nos diferentes níveis ou camadas arquiteturais.

Palavras-Chave: Arquiteturas protocolares, Convergência entre protocolos, Modelos arquiteturais, Redes *IoT*, RSSF

ABSTRACT

The Internet of Things (IoT) is an innovative concept supported by distinct technologies, which has been particularly driven by the emergence of Wireless Sensor Networks (WSNs) and corresponding applicability scenarios. Examples of these scenarios include monitoring in environmental, home, industry and health contexts, among others.

WSNs are constantly evolving for application areas where several devices need to be interconnected to the Internet, thus supporting the concept of diversity underlying IoT. However, the sensor devices involved in WSNs have usually limited capacities, namely in memory and processing, and have energy restrictions for the exercise of their functions. In this way, the scientific community and the industry have proposed specific architectures and protocols to operate in networks involving these low-capacity devices and to promote their interconnection to the Internet.

In this context, this master work aims to provide a detailed study of the architectures and protocols supporting IoT, relating them to the conventional Internet protocols and studying the corresponding convergence solutions. To achieve these goals, this work presents a historical perspective of Internet models and architectures influencing IoT, discusses the architectures and protocols specifically proposed to sustain IoT operation and, in particular, explores the convergence scenarios between protocols of distinct network technologies at different architectural layers.

Keywords: Architectural Models, Protocol Architectures, Protocol Convergence, *IoT* Networks, WSN

Índice Geral

Lista de Figuras

Lista de Símbolos e Abreviaturas

3GPP	3rd Generation Partenership Project
6LoWPAN	IPv6 over Low Power Wireless Personal Area Networks
AMQP	Advanced Massage Queueing Protocol
ARP	Address Resolution Protocol
AS	Autonomous System
BLE	Bluetooth Low Energy
BSD	Berkley Software Distribuition
CoAP	Constrained Application protocol
CSMA-CA	Carrier Sense Multiple Access with Collision Advoidance
DDS	Data Distribuition Service
DIO	Destination Information Object
DODAG	Destination-Oriented Directed Acyclic Graph
DNS	Domain Name System
DAO	Destination Advertisement Object
DAO-ACK	Destination Advertisement Object Acknowledgment
DPWS	Devices Profiles for Web Services
DTLS	Datagram Transport Layer Security

DV	Distance Vector
EIGP	Enhanced Interior Gateway Protocol
EPC	Envolved Packet Core
FFD	Full-function Devices
FTP	File Transfer Protocol
HSPDA	High-Speed Downlink Packet Access
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol Version 6
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management protocol
IGP	Interior Gateway Protocol
IOS	International Organization for Standardization
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
ITU	International Telecommunication Union
LCC	Link Control Protocol
LLN	Low-power and Lossy Network
LTE	Long-Term Evaluation
LTE-A	Long Term-Evaluation Advanced
M2M	Machine-To-Machine

MAC	Medium Access Control
MIB	Management Information Base
MIMO	Multiple-input and Multiple-output
MoM	Massage Oriented Middleware
MLD	Multicast Listener Discovery
MQTT	Message Queue Telemetry Transport
MQTT-SN	Message Queue Telemetry Transport for Sensor Network
NCPs	Network Control Protocols
NAT	Network Address Translation
NVT	Network Virtual Terminal
OFDMA	Orthogonal Frequency Division Multiple Access
OSI	Open System Interconnect
OSPF	Open Shortest Path First
PAN-ID	Personal Area Network Identifier
PDU	Protocol Data Unit
PLC	Power Line Communication
POP3	Post Office Protocol Version 3
PPP	Point-to-Point Protocol
PPPoA	Point-to-Point Protocol Over ATM
PPPoE	Point-to-Point Protocol Over Ethernet
QUIC	Quick UPD Internet Connections
RLE	Run-Length Encoding
REST	Representational State Transfer

RFC	Request for Comments
RIP	Routing Information Protocol
RPC	Remote Procedure Call
RPL	Routing over Low Power and Lossy Network
SMI	Structure of Management Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SSL 3.0	Secure Sockets Layer Version 3.0
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
W-CDMA	Wideband Code Division Multiple Access
WiMax	Worldwide Interoperability for Microwave Access
WSSOAP	Wireless Sensor SOAP
WTLS	Wireless Transport Layer Security
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
XNS	Xerox Network System

Capítulo 1

INTRODUÇÃO

A *Internet* das coisas (*Internet of Things - IoT*) é constituída por objetos inteligentes, equipados com sensores e com capacidade de comunicação entre si, recolhendo informação do exterior por meio de sensores neles existentes. A tendência das Redes de Sensores Sem Fio (*RSSF*) começam a ser cada vez mais comuns, estendendo as fronteiras da *Internet* até ao mundo físico.

A interligação entre os dois mundos pode ser feita por *proxies* responsáveis pela interpretação dos protocolos proprietários para *IP* ou por soluções em que os dispositivos estão ligados diretamente à rede *IP*. A pilha protocolar *TCP/IP* obrigaram os objetos sensores a possuir mecanismos que permitam a interligação (Stankovic, 2008). Fazendo o uso da tecnologia *IP* acrescentará vantagens em utilizar uma infraestrutura já existente, com protocolos abertos com aplicações para gestão e diagnóstico, uma vez que esta tecnologia disponibiliza conectividade integral de forma heterogénea e sem grande esforço para sua implementação.

Para que haja um potencial crescimento por parte da *IoT* há um enorme trabalho a ser feito. A interligação das *RSSF* à *Internet* não será apenas este um dos principais objetivos, para tal, esperam-se outras funcionalidades complexas para trabalhar em conjunto para um fim específico, de forma automática (Stankovic, 2008).

Um dos grandes desafios da *IoT* é a capacidade de comunicação com dispositivos menores, estes últimos com menor capacidade computacional, estando alguns dependentes de bateria, sendo capazes de interagir entre eles por meio de aplicação (Neves and Rodrigues, 2009). A mobilidade, confiabilidade, interoperabilidade, disponibilidade, gerenciamento, segurança e privacidade não foge da lista de prioridades da *IoT*.

A etiqueta TAG usada na tecnologia *Radio Frequency Identification* (*RFID*) possui

informação de identificação (*Electronic Product Code-EPC*) de um determinado produto por exemplo, o leitor emite um sinal de rádio e aproveita a proximidade das TAGs que transmitem suas informações de identificação por meio de um processo de indução magnética entre o Leitor e as Tags, este é um dos exemplos da aplicação do paradigma de *IoT*, onde diferentes aplicações podem ser usadas para o controlo de produtos armazenados; os sensores colocados nos automóveis que indicam aproximação de um obstáculo para prevenir dos possíveis acidentes; o termóstato que comunica com o sensor de ar condicionado consoante a temperatura do ambiente, e entre outras utilizações. Este exercício só é feito por meio de protocolos específicos.

Em torno desta problemática, a comunidade científica centrou-se na criação de protocolos de encaminhamento e transporte para *IoT*, tais como o *Constrained Application Protocol* (*CoAP*), *IPv6 over Low-power Wireless Personal Area Networks* (*6LoWPAN*), *Message Queuing Telemetry Transport* (*MQTT*), entre outros (Ko et al., 2011). Cada um destes protocolos possui aplicabilidade restrita que precisa de ser compreendida, pois não existe um único protocolo eficiente para todas as aplicações (Neves and Rodrigues, 2009). Por esta razão, os protocolos *IoT* foram e continuam a ser desenvolvidos para resolver questões como a diminuição do cabeçalho, a fragmentação e a remontagem de pacotes, entre outros, dando origem a novas arquiteturas de rede (Stankovic, 2008).

No sentido lato da palavra, a arquitetura de rede é caracterizada por um conjunto de componentes com caraterísticas especificas que definem interacções entre si (Loureiro, 1998).

Quanto às arquiteturas, é evidente que não existe uma arquitetura de referência única como modelo para todas as implementações concretas possíveis. Enquanto um modelo de referência pode provavelmente ser identificado, é provável que várias arquiteturas de referência irão coexistir na *IoT* (Khan et al., 2012).

A arquitetura, neste contexto, é definida como um quadro de especificação de componentes físicos de uma rede, a sua organização funcional, configuração, os seus princípios, procedimentos operacionais e formatos de dados usados na sua operação (Severi et al., 2014).

1.1 Motivação

O desenvolvimento dos dispositivos sensores e as limitações, principalmente a nível da capacidade de processamento, armazenamento e energia, influenciados pelo seu tamanho,

obrigou à criação de protocolos e aplicações para as *RSSF*. A necessidade de integrar as *RSSF* com a *Internet* motivou fazer estudos sobre a influência dos modelos de *Internet* na *IoT*, observando a convergência entre os protocolos das diferentes arquiteturas face à finalidade das funções protocolares por camadas, visto que os dispositivos sensores recolhem informações para serem analisadas pelos dispositivos com maior capacidade de processamento.

Assim sendo, a análise da convergência entre os protocolos nas arquiteturas é, por isso, muito importante, e a necessidade para integrar as redes de sensores é cada vez maior, já que os protocolos de *Internet* possuem características específicas que aumentam a complexidade para a criação de protocolos convergentes entre as redes *IoT*.

Deste modo, a contribuição de ideias que ajudem na solução para criação de protocolos convergentes, poderá cumprir um papel fundamental no contexto da futura integração de *RSSF* e a *Internet*, de forma semelhante ao que se pode verificar, atualmente, na arquitetura de comunicações de *Internet*.

1.2 Objetivos

De acordo com a finalidade identificada no enquadramento, estabelece-se para este trabalho de mestrado o seguinte conjunto de objetivos:

(i) descrever os modelos arquiteturais e arquiteturas protocolares de *Internet* e *IoT*;

(ii) estudar os principais protocolos *IoT* nas arquiteturas a nível das suas camadas;

(iii) identificar a convergência entre os protocolos *IoT* e de *Internet* nas camadas arquiteturais, expondo vantagens, possíveis limitações e apresenta a sua importância no contexto de integração com a *Internet*.

1.3 Estrutura da Dissertação

O trabalho está estruturado em cinco capítulos, antecedidos por uma introdução ao tema, no qual são apresentados o enquadramento, os objetivos e a estrutura do trabalho.

No capítulo 2 é feita uma descrição do modelo e arquitetura de *Internet* existentes na literatura, apresentando os protocolos por camadas.

No capítulo 3 apresentam-se os modelos *IoT* identificando a sua influência. Por outro lado, impõem-se reflexões de vários autores sobre a correspondência entre as arquiteturas e modelos arquiteturais, bem como a descrição das principais funções dos protocolos por camada.

No penúltimo capítulo, seguido das considerações finais e trabalho futuro, é estabelecida a comparação entre protocolos a nível das funções, nas diferentes arquiteturas, tendo em atenção a convergência dos mesmos, face as limitações, apresentando-se a importância no contexto da integração com a *Internet*.

Capítulo 2

MODELO *OSI* E ARQUITETURA *TCP/IP*: VISÃO GERAL E OS PRINCIPAIS PROTOCOLOS POR CAMADAS

O objetivo deste capítulo é apresentar uma visão geral do modelo *OSI* e arquitetura *TCP/IP*. Primeiro é feita uma apresentação sobre o modelo *OSI* e as funções por camadas e em seguida a introdução da arquitetura *TCP/IP*, relacionar a correspondência entre os dois modelos, descrevendo os principais protocolos por camadas.

2.1 Modelo *OSI* e arquitetura *TCP/IP*

Nos primórdios dos anos 70, as arquiteturas de comunicação eram quase dominadas por grandes fabricantes de computadores. Entre estes fabricantes, havia uma grande dificuldade de definir arquiteturas coerentes para os seus sistemas de comunicação, na maioria dos casos, tinham linhas de produtos de comunicação incompatíveis entre si. Nesta inconformidade, a Organização Internacional para Padronização (*ISO - International Organization for Standardization*) no âmbito de um esforço mais abrangente, cujo objetivo era definir uma arquitetura aberta, definiu o modelo de referência *OSI* (*Open System Interconnect*) (Marques and Guedes, 1998) tal como é apresentado na Figura 2.1.

Este modelo de referência definiu os princípios básicos para o Sistema Aberto como sendo aquele que segue as normas *OSI* nas sete camadas. O modelo *OSI* por si só não constitui uma arquitetura de rede, visto que o modelo não especifica serviços e protocolos a serem usados em cada uma das camadas. Este modelo, define os principais conceitos e divide a tarefa de comunicação em sete camadas funcionais, especificando as funções a serem desempenhadas

Figura 2.1: Modelo da arquitetura *OSI*
(Marques and Guedes, 1998)

por cada camada. No entanto, a *ISO* passou a projetar, especificar, implementar e testar os protocolos das várias camadas definidas pelo modelo *OSI* da *ISO*, originando deste modo o modelo da Arquitetura *OSI* (Marques and Guedes, 1998); (Tanenbaum, 2003).

- As principais funções das camadas do modelo *OSI* são:

1. **Camada Física**: é a mais baixo do modelo *OSI*. Ela constitui a interface com o meio físico de comunicação, e define a representação lógica da informação (*bits*) e como estes valores lógicos de (0 ou 1) são transformados em símbolos físicos, tensões ou correntes eléctricas, ondas electromagnéticas, sinais ópticas que viajarão no meio físico utilizado (Gouveia and Magalhães, 2005); (Tanenbaum, 2003). A propagação destes sinais pelo meio físico possui características, como a velocidade de propagação, atenuação, imunidade ao ruido, entre outros, são os principais elementos para a definição deste nível (Marques and Guedes, 1998) .

   Esta camada, parece ser dividida em duas sub-camadas, a primeira é relativa aos aspetos dependentes do meio físico como, por exemplo, conectores, transmissão e receção de sinais físicos, e o segundo é relativo a aspetos independentes do meio físico como, por exemplo, codificação e descodificação de conjunto de *bits* que são transmitidos e recebidos (Boavida and Monteiro, 2011); (Loureiro, 1998).

2. **Camada de Enlace de Dados**: é a segunda camada do modelo *OSI*, um dos grandes objetivos desta camada é garantia da comunicação, fornecendo mecanismos locais de controlo de fluxo de informações e de controlo de erros. Ela lida com conjunto de *bits*,

que são organizados em quadros ou tramas, enviados entre sistemas adjacentes da rede. Esta camada, parece dividida em duas sub-camada: a **sub-camada de controlo de acesso ao meio físico**, que determina quando é que uma dada estação de rede pode transmitir informação; e a **sub-camada de controlo de ligações lógicas**, esta poderá lidar com aspetos como controlo de fluxo e sequência (Boavida and Monteiro, 2011) (Gouveia and Magalhães, 2005).

3. **Camada de Rede**: é camada que garante a interligação entre quaisquer sistemas terminais, independentemente da localização desses sistemas e o tipo de sub-redes atravessado. Umas das grandes funções desta camada relaciona-se com o encaminhamento da informação por meio da rede, assegurado por mecanismos e protocolos. Nesta camada, é identificada apenas uma forma entre todos os sistemas terminais e encaminhadores de rede através da utilização da função do endereçamento lógico de modo a efetuar a transição de nomes lógicos para os endereços físicos (Boavida and Monteiro, 2011); (Loureiro, 1998).

   O modo operando deste nível foi motivo de muita controvérsia. Duas filosofias de encarar a rede podem ser consideradas: a definida pelas operadoras de telecomunicações pretender que o nível de rede assegure uma ligação fiável que garanta a sequência e o controlo de fluxo e, por outro, está a visão associada à Arpanet e depois à Internet que considera que o nível de sub-rede é colocá-lo no transporte extremo a extremo. Nesta vertente o nível de rede deve assegurar o encaminhamento eficiente dos pacotes e não implementar funcionalidade redundante (Marques and Guedes, 1998).

4. **Camada de Transporte**: é a que garante a comunicação ponto-a-ponto entre as estações na rede. A independência entre os sistemas terminais relativamente ao tipo e qualidade das sub-redes utilizadas, por meio de mecanismos de detenção de erro, controlo de fluxo e o controlo de sequência, são dependentes desta camada. Nela, são utilizados dois serviços de ligação para correcção de erro e o controlo sobre o fluxo, a chamada ligação orientada *conection-oriented* e a ligação livre *conectionless* (Loureiro, 1998); (Boavida and Monteiro, 2011).

   As mensagens que chegam a nível de transporte são uma abstração lógica com correspondência nas estruturas de dados manipulados pelos protocolos de nível superior. Estas mensagens têm, dimensão variável, não obrigando os utilizadores a preocupar-se com as limitações que podem ser impostas pelo tamanho dos pacotes dos níveis

inferiores. Isolar o utilizador das limitações físicas da rede significa que o transporte efetue a fragmentação das mensagens e, o emissor tem de decompor as mensagens em pacotes com tamanho aceitável pelo nível da rede e, numerá-los para recompor adequadamente a mensagem. Por outro lado, o recetor deve reconstruir a mensagem na ordem certa, mesmo que haja atrasos ou repetições provocadas pelos mecanismos dos níveis inferiores (Marques and Guedes, 1998).

5. **Camada de Sessão**: Uma das importantes tarefas desta camada, são os mecanismos para o controlo e sincronização da conversa entre as entidades da aplicação comunicante, ou seja, o nível tem por função a multiplexagem de várias instâncias de comunicação sobre a mesma ligação de transporte (Boavida and Monteiro, 2011); (Marques and Guedes, 1998). Os pontos de sincronismo e de recuperação de fluxo de dados são estabelecidas nesta camada, o que é bastante importante para certo tipo de aplicações como, por exemplo, aplicações com ambientes tolerantes a falha que, normalmente se baseiam na realização de acções atómicas, aplicações de acesso a base de dados distribuída, entre outros (Boavida and Monteiro, 2011); (Loureiro, 1998).

6. **Camada de Apresentação**: este nível controla a troca de informação entre sistemas heterogéneos e, exige que seja adotada na representação comum para os dados, isto é, uma representação de dados que seja compreendida por todos os sistemas envolvidos no diálogo. Esta camada fornece meios para o desenvolvimento e utilização de sintaxes - abstractas e de transferências - que possibilitam essa troca de informação. A necessidade de um protocolo de representação de dados existe na maioria dos sistemas, ela dificilmente aparece na arquitetura na forma estritamente hierárquica que o modelo *OSI* pressupõe (Marques and Guedes, 1998); (Boavida and Monteiro, 2011).

7. **Camada de Aplicação**: é a camada que fornece mecanismos de comunicação de alto nível, orientados para aplicações, isto é, orientado para processos de utilizador. Esses mecanismos poderão ser comuns a várias aplicações entre entidade de aplicações como, por exemplo, mecanismos orientados para a transferência de ficheiros ou para aplicações de terminal virtual, correio eletrónico, serviço de gestão, entre outros, todas elas definidas pelo modelo *OSI*. Esta camada pode ser encarada como a componente de comunicação dos processos aplicacionais, sendo, muitas vezes, confundida os próprios processos de aplicação (Marques and Guedes, 1998); (Boavida and Monteiro, 2011) .

A proliferação de arquiteturas de comunicação incompatíveis registadas nas décadas de

1970 e 1980 foi uma das principais motivações para o desenvolvimento do modelo *OSI*. A abertura e universalidade inicialmente reservada ao modelo *OSI* veio a ser desempenhada pela arquitetura *TCP/IP*. Quer o modelo *OSI* quer a arquitetura *TCP/IP* utilizam o conceito de estratificação, na qual as funções implementadas internamente a uma dada camada não são visíveis nas outras camadas(Boavida and Bernardes, 2012).

Estas sete camadas ou níveis que constituem o modelo *OSI*, teve como base a comunicação entre computadores, pelos as quais a informação têm de transitar, desde que é gerada por uma aplicação, até chegar ao meio físico de transporte (Loureiro, 1998); (Gouveia and Magalhães, 2005). No caso do modelo *TCP/IP*, privilegiou-se uma abordagem simples, pragmática e, normalmente, precedida de experimentações e comprovação em ambiente real (Boavida and Bernardes, 2012).

Neste sentido, a arquitetura protocolar *TCP/IP* apresenta as seguintes características (Boavida and Monteiro, 2011):

(i) conjunto de protocolos disponíveis livremente, independentes de *hardware* específico, sistemas operativos ou fabricantes, o que torna os protocolos totalmente abertos;

(ii) os protocolos suportados por, praticamente, todo o tipo de fabricante e equipamentos, o que os torna nos protocolos de comunicação mais utilizados atualmente;

(iii) possui um grupo de protocolos de aplicação dirigidas a necessidades específicas dos utilizadores;

(iv) possui esquema de endereçamento universal, o que permite a identificação unívoca dos dispositivos na rede e encaminhamento simples e eficiente.

A aqruitetura *TCP/IP* conduziu a soluções claras de complexidade que não fossem justificadas por necessidades concretas, em relação ao problema de comunicação fiável entre computadores. Esta realidade comprovada é patente na arquitetura protocolar resultante, que é constituida por apenas quatro camadas, em vez das sete camadas apresentadas pelo modelo *OSI* da *ISO*. A Figura 2.2 representa a arquitetura protocolar *TCP/IP* paralelamente com a arquitetura *OSI* , sendo claramente visível a correspondência entre as camadas (Boavida and Monteiro, 2011); (Marques and Guedes, 1998).

As camadas (físicas e enlace de dados) do modelo *OSI* correspondem a camada mais inferior de (interface de rede) da arquitetura *TCP/IP*. As camadas de (transporte e de rede) do modelo *OSI* correspondem as mesmas camadas de (transporte e de rede) da arquitetura

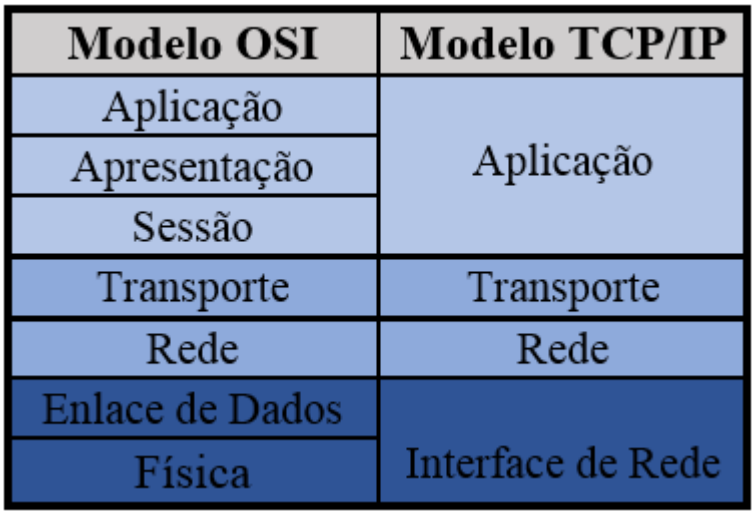

Figura 2.2: Relação entre a *OSI* e arquitetura protocolar *TCP/IP* (Boavida and Monteiro, 2011)

TCP/IP e, finalmente, as três últimas camadas do modelo *OSI* (sessão, apresentação e aplicação) corresponde a última camada de aplicação da arquitetura *TCP/IP*.

2.1.1 Principais protocolos da arquitetura *TCP/IP* por camadas

Na *Internet*, existe um conjunto de protocolos específicos para a resolução de um determinado problema em função do tipo de comunicação que é efectuada a nível da rede. Neste sentido, as arquiteturas protocolares, ou de rede, são constituídas por um conjunto de protocolos nas diferentes camadas ou níveis (Boavida and Monteiro, 2011); (Gouveia and Magalhães, 2005).

A arquitetura *TCP/IP*, possui um conjunto de protocolos orientados para necessidade concreta e suportados por aplicações existentes, sendo esta uma condição indispensável para que estas aplicações tenham a adesão por parte dos utilizadores (Boavida and Monteiro, 2011); (Gouveia and Magalhães, 2005). A Figura 2.3 representa as camadas protocolares, sendo visíveis alguns dos protocolos comuns que constituem a arquitetura protocolar *TCP/IP* nas respetivas camadas.

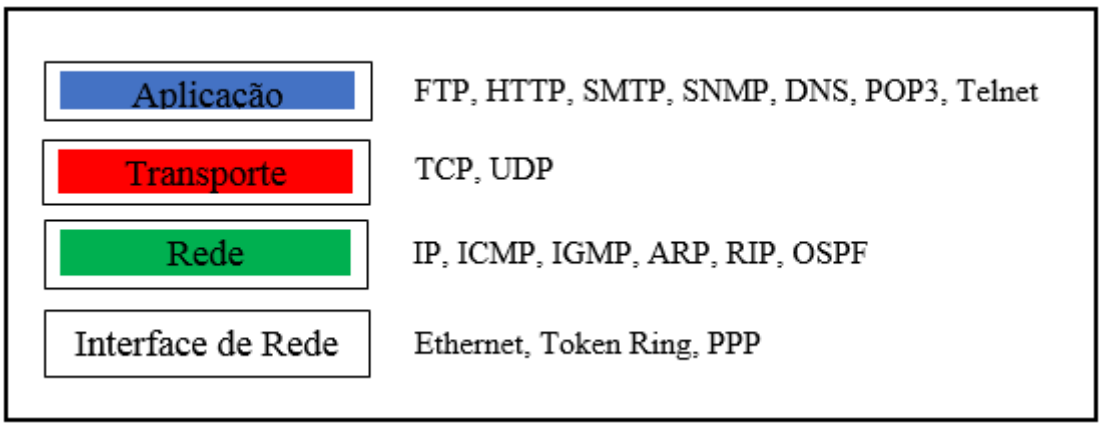

Figura 2.3: Principais protocolos por camada na arquitetura *TCP/IP*

2.2 Protocolos da Camada de Interface de Rede

2.2.1 O protocolo *Ethernet*

O protocolo *Ethernet* é um conjunto de tecnologia para redes locais, este protocolo funciona na camada de interface de rede da arquitetura *TCP/IP*, que corresponde as duas primeiras camadas do modelo *OSI* da ISO (camada Física e Enlace de dados). Um dos principais objetivos deste protocolo é efetuar a transferência de pacotes entre os *host* na mesma rede. A maior parte das redes usam esta tecnologia, sendo que os seus endereços são atribuídos por uma entidade central aos construtores do seu equipamento, pelo que não existem dois dispositivos de rede com o mesmo endereço, sendo suficientemente grandes com 48 *bites* e 6 *bytes* (Boavida and Monteiro, 2011); (Boavida and Bernardes, 2012).

2.2.2 O protocolo *Token-Ring*

O *Token-Ring* é um protocolo de redes que funciona na camada física do modelo *TCP/IP*, que são agregados no modelo *OSI* da ISO, desenvolvido pela *IBM*. A topologia anelar, usada pelo protocolo, é praticamente protegida de colisões de pacotes, permitindo um diagnóstico e soluções simples, dada a obrigatoriedade de utilizar nestas redes *hubs* inteligentes. Tal como a *Ethernet*, o *Token-ring* utiliza vários tipos de cabo para a comunicação, sendo os cabos de pares entrançado os mais utilizados. A escassez das redes ***Token-Ring*** aumenta a cada dia passa, visto que o protocolo apresenta grandes desvantagens em relação às redes *Ethernet*, devido aos custos elevados da implementação do *Token-ring* serem bastantes dispendiosos e a velocidade de transmissão limitada (16 Mbps), uma vez que *Ethernet* oferece uma velocidade

de 100 Mbps. O processo da transmissão é feito durante uma pequena janela de tempo, e apenas por quem detém o *Token* (Gouveia and Magalhães, 2005).

Nas redes *Token-ring*, é utilizado um disposetivo central denominado *Multistation Access Unit* (*MAU*) onde todos os computadores vão ligar. Uma outra diferença entre as redes *Token-ring* e as outras tem a ver com o processo de regeneração do sinal, os dados são amplificados e repetidos por cada computador da rede de forma a evitar degradação do sinal (Loureiro, 1998);(Gouveia and Magalhães, 2005); (Boavida and Monteiro, 2011).

2.2.3 O protocolo *PPP*

O *Point-to-Point Protocol* (*PPP*) tem como objetivo estabelecer a ligação entre duas redes ou máquinas a través de um modem ou um encaminhador, utilizando conexão física serial (cabo serial, ligações de fibras ópticas, etc) única e *full-duplex* (Kurose and Ross, 2013).

O protocolo suporta linhas síncronas e assíncronas, embora seja um protocolo que se encontra nas linhas de interfaces. Na prática, a interface deste é implementada através de conexões físicas do tipo RS-232 ou *modems*. Hoje em dia, esse tipo de conexões pode ser feito sobre o *Ethernet (PPPoE)*. O *PPP* é basicamente constituído por três partes, obedecendo um diagrama de três fases no processo de interacção, sendo um deles o encapsulamento de datagramas, o *Link Control Protocol* (*LCC*) e *Network Control Protocols* (*NCPs*) (Kurose and Ross, 2013) (Loureiro, 1998). Este protocolo também é suportado pelo *Windows NT*, como cliente ou servidor, usando-o no acesso aos servidores dos *Internet Service Provider* (*ISP*) (Loureiro, 1998).

2.3 Protocolos da Camada de Rede

2.3.1 O protocolo *IP*

O *Internet Protocol* (*IP*) é um dos mais populares protocolos atualmente em uso na *Internet*, desenvolvido nos finais dos anos 70 e foi publicado em Setembro de 1981. O protocolo pertence à camada de rede e é responsável pelo o encaminhamento dos dados a nível rede, prestando todos os serviços de rede para as camadas superiores. Ele é a base da arquitetura de *Internet* e é utilizado por todos os serviços de aplicação, tal como o gerenciamento de

redes, transferência de arquivos, resolução de nomes, dentre outros (Boavida and Monteiro, 2011); (Boavida and Bernardes, 2012). Por outro lado, também oferece um serviço de dados não confiável designado por melhor esforço. Isto quer dizer que o pacote chega quase sem nenhuma garantia, podendo também chegar de forma desordenada em comparação com os demais pacotes transmitidos pelos mesmos dispositivos ou nós na rede. Podem, de igual forma, perder-se na sua totalidade assim como os pacotes que chegarem duplicados nos nós finais (Boavida and Bernardes, 2012).

Caso o protocolo requerer confiabilidade, o processo é realizado na camada de transporte, onde encontramos protocolos específicos que exercem estas funções. Devido aos problemas que o protocolo *IP* da versão quatro (*IPv4*) não conseguia resolver, outras versões protocolares foram criadas no sentido de colmatar estes problemas. A título de exemplo temos o protocolo *IPv5* que não teve sucesso, sendo o seu principal objetivo permitir o tráfego de voz e de vídeo sobe o *multicast*. O *IPv5* não foi criado para substituir o *IPv4*, mas sim para operar junto do *IPv4* (Boavida and Bernardes, 2012); (Kurose and Ross, 2013).

O rápido crescimento de *Internet* e o desperdício do endereçamento do esquema de protocolos aplicado pelo *IPv4*, foram os motivos que levaram à criação da nova versão do protocolo *IP*, o chamado *IPv6* (Boavida and Monteiro, 2011). Este Procolo não resolveu somente o número de endereços disponíveis, também ofereceu novos serviços que o protocolo *IPv4* não oferecia, ou que não eram utilizadas de forma otimizada, dentre as quais o longo espaço de endereçamento para alcance global e escalabilidade. Este fornece uma arquitetura hierárquica para um encaminhamento eficiente, bem como serviços de outo configuração, permitindo um crescimento brutal de número de endereços *multicast*, implementações para qualidade de serviços, o formato de cabeçalho simplificado para a entrega de pacotes de forma otimizada, entre tantos outros benéficos (Boavida and Monteiro, 2011); (Boavida and Bernardes, 2012); (Kurose and Ross, 2013).

2.3.2 O protocolo *ARP*

O *Address Resolution Protocol* (*ARP*) é um dos protocolo bastante fundamental entre os protocolos da camada de rede. Este tem uma grande importância na comunicação de informação, porque permite obter endereço físico (*MAC*) de uma placa de rede, utilizando o endereço lógico (*IP*) correspondente (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

O protocolo questiona todas as máquinas na rede para conhecer o seu endereço físico e de seguida cria uma tabela com a correspondência entre os endereços lógicos e os endereços físicos numa memória (*cache*) de armazenamento. Sempre que uma máquina inicia a comunicação com outra, a tabela de *ARP* é consultada. Caso o endereço não for encontrado na tabela, o protocolo emite um pedido na rede (*ARP Request*) e os clientes na rede vão comparando o endereço lógico do pedido ao seu. Caso um dos clientes reconhecer o seu endereço *IP* no pedido (*ARP Request*), a mesma reponde enviando uma (*ARP Reply*) (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

2.3.3 O protocolo *ICMP*

O *Internet Control Message Protocol* (*ICMP*) funciona juntamente com o protocolo *IP*, ou seja, o *ICMP* é o protocolo integrante do protocolo *IP*. É utilizado para fornecer relatórios de erros aos clientes, isto é, envia uma mensagem que menciona os pacotes de informação que não foram entregues corretamente. Desta forma é enviada uma mensagem *ICMP* de volta para ser enviado o pacote de informação não recebido (Gouveia and Magalhães, 2005).

Estas mensagens são enviadas geralmente nas seguintes situações: quando um pacote *IP* não chega ao seu destino; quando o *gateway* não consegue retransmitir os pacotes na frequência adequada; finalmente, quando um encaminhador indica uma rota melhor para um cliente enviar pacotes (Gouveia and Magalhães, 2005); (Boavida and Bernardes, 2012). Tal como os outros protocolos que foram sofrendo atualizações em função das exigências que os serviços requeriam, também o *ICMP* sofreu atualização. Dai que o *ICMPv6* surge justamente para melhorar o *ICMP*, sendo usado por dispositivos que suportam o protocolo *IPv6* para relatar os erros encontrados no processamento de pacotes e para executar outras funções da camada da *Internet*. Estas mensagens são agrupadas em duas classes, mensagens de erros e mensagens de informação (Wu et al., 2009)

2.3.4 O protocolo *IGMP*

O *Internet Group Management Protocol* (*IGMP*) é o protocolo responsável pela gestão de informação que circula pela *Internet* e *Intranet* através do protocolo *TCP/IP*. Tal como o *ICMP*, o *IGMP* é uma parte integrante do *IP* e é um dos requisitos básicos de implementações a todos os clientes que desejam receber e enviar pacotes *multicast*, ou seja, é um mecanismo

para troca de informação entre um cliente e um encaminhador *multicast*. As suas mensagens são encapsuladas em datagramas *IP* (Gouveia and Magalhães, 2005).

Tal como os demais protocolos, o *IGMP* também sofreu várias atualizações em função das lacunas das versões anteriores e das necessidades exigidas nas aplicações. A versão *IGMPv1* tinha problemas devido ao grande período de latência associada ao termino de sessões *multicast*. O *IGMPv2* veio reduzir o *overhead* e, posteriormente o *IGMPv3* foi desenvolvida já com grandes ambições: para além da redução do *overhead*, a largura de banda é conservada pelas mensagens de grupo (*Group-Source Report*). Em consideração com as atuais atualizações recentes do *IGMP*, podem reduzir o tráfego desnecessário (Gouveia and Magalhães, 2005).

2.3.5 O protocolo *RIP*

O *Routing Information Protocol* (*RIP*) é um protocolo de rede de encaminhamento interno criado pela *Xerox Corporation* no inicio dos anos 80 para ser utilizado nas redes *Xerox Network System* (*XNS*). O protocolo ganhou mercado no domínio da rede pelas suas particularidades e começou a ser utilizado pelos fabricantes de dispositivos de redes, nomeadamente os grandes fabricantes de encaminhadores de rede. Este protocolo, utiliza algoritmo de encaminhamento interno entre os sistemas autónomos, do Inglês (*AS - Autonomous System*) (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013) .

No que diz respeito a funcionalidade, o protocolo *RIP* define dois tipos de operações:

(i) mensagem de requisição (*Request Message*);

(ii) mensagem de resposta (*Response Message*).

Os encaminhadores transmitem mensagens de requisição em *broacast* para todas as interfaces do protocolo *RIP* e as mensagens de resposta são retornadas caso os encaminhadores vizinhos recebam as mensagens de requisição (contendo as mensagens das tabelas de encaminhamento). Desta forma, o encaminhador recebe a tabela de encaminhamento e realiza o processamento de cada entrada da tabela cumprindo com as suas regras (Gouveia and Magalhães, 2005) (Kurose and Ross, 2013).

Para indicar a distância até à rede de destino, o protocolo utiliza a métrica de contagem de saltos da rota indicando os encaminhadores vizinhos. Os encaminhadores mantêm somente a melhor rota (a rota com o valor métrico mais baixo) até um destino. As tabelas de

encaminhamento periodicamente são atualizados por encaminhadores. O encaminhador principal atualiza os demais encaminhadores de rede sobre a mudanças existentes. Estas atualizações são emitidas independentemente das atualizações regulares programadas na rede (Gouveia and Magalhães, 2005) (Kurose and Ross, 2013).

Uma das grandes vantagens do *RIP* é impedir que os enlaces do encaminhamento comecem a transmitir de forma infinita as mensagens entre eles, de forma a evitar *loops* na rede. Para tal, o protocolo possui limite no número de saltos durante o percurso de origem até ao destino. Para além destas métricas, o protocolo possui características em termos de estabilidade que é uma das características comuns vista em muitos protocolos de encaminhamento, características estas que são desenhadas com a finalidade de dar estabilidade adequando-se às mudanças constantes que podem existir numa topologia de rede (Kurose and Ross, 2013).

O *RIP* também possui características semelhantes ao protocolo *TCP*, que tem o mecanismo de temporização para melhorar o seu desempenho. Estes caracterizam-se por:

(i) temporizador de atualização de encaminhamento;

(ii) temporizador do intervalo de distribuição;

(iii) temporizador para o nível de distribuição.

Este processo é utilizado de formas a impedir o congestionamento que possa surgir caso todos os encaminhadores utilizarem e atualizar suas tabelas ao mesmo tempo. Cada entrada da tabela de encaminhamento tem um temporizador de intervalo de distribuição associada a ela (Gouveia and Magalhães, 2005) (Kurose and Ross, 2013). Em virtude protocolo *RIP* não resolver todos os problemas, o mesmo sofreu atualizações com a versão *RIP-V2*, melhorando os aspetos como o suporte de autenticação, encaminhamento pós sub-rede, e entre outros (Gouveia and Magalhães, 2005) (Kurose and Ross, 2013).

Resumindo, o *RIP* é primeiro protocolo de encaminhamento padrão desenvolvido para ambientes *TCP/IP*. Este possui encaminhamento dinâmico, que implementa o algoritmo vetor-distância e é caracterizado pela simplicidade e facilidade de solução de problemas. Os seus pacotes são enviados usando o *UDP* (Kurose and Ross, 2013).

2.3.6 O protocolo *OSPF*

O *Open Shortest Path First* (*OSPF*) é um protocolo projetado especialmente para ambiente *TPC/IP*, utilizado no interior de sistemas autónomos (*Interior Gateway Protocol - IGP*) para troca de informações de rotas dos pacotes *IP*. Surgiu em substituição do protocolo *RIP*, mas é diferente deste na medida em que o OSPF pode obedecer uma hierarquia. O *OSPF* é um protocolo *link-state*, que procura o melhor caminho, usando o algoritmo *Shortest Path First*. Em termos de algoritmo funciona de modo diferente do vetor-distância. Ao contrário de ter na tabela os melhores caminhos ou rotas, os nós todos possuem todos os *links* da rede. Os encaminhadores funcionam com este protocolo trocando entre si informações sobre o estado dos enlaces de comunicação ligados às suas portas (Boavida and Bernardes, 2012); (Kurose and Ross, 2013). Alguns dos avanços incorporados neste protocolo são a segurança, caminhos múltiplos com os mesmos custos, o suporte integrado para o encaminhamento individual e em grupo *unicast* e *multicast* e, finalmente, o suporte para hierarquia dentro de um único domínio de encaminhamento (Kurose and Ross, 2013).

2.4 Camada de Transporte

2.4.1 O protocolo *TCP*

Transfer Control Protocol (*TCP*) é um dos principais protocolos da camada de transporte do *TCP/IP*. Este permite dar segurança a transferência de informação e verificar se a mesma foi bem recebida pelo recetor. Caso não seja, volta a retransmitir os dados. Desta forma, o protocolo tem o controlo de tráfego de informação entre o emissor e o recetor (Gouveia and Magalhães, 2005).

Dentro do pacote *TCP* contem também informação sobre as portas usados pelo emissor e pelo recetor das informações contidos na sua secção de dados. Não inclui, porém, as informações relativas aos endereços dos hosts envolvidos, já que essa informação estará contida nos pacotes *IP*, onde serão encapsulados os pacotes *TCP* ou *UDP*, depois de enviados para camada inferior. (Loureiro, 1998).

O *overhead* associado a este protocolo está relacionado não só com a grande complexidade e tamanho dos pacotes de informação, mas também com a possibilidade da informação ser sujeita a múltiplas transmissões ainda que parciais (Khan et al., 2012). O protocolo,

mediante as suas caraterísticas, permite efetuar o gerenciamento de pacotes, assim como a sua temporização, devido ao mecanismo de retransmissão e de controlo de tempo. O controlo de fluxos é também uma das características do protocolo devido a imprevisibilidade do tráfego (que é um grande problema) então os mecanismo de retransmissão e *buffers* são ativadas (Loureiro, 1998).

2.4.2 O protocolo *UDP*

User Datagram Protocol (*UDP*) é um protocolo simples da camada de transporte, não dirigido à conexão (não há necessidade de manter a conexão longa entre o cliente e o servidor) e não confiável. Permite que a aplicação encapsule um datagrama num pacote *IPv4* ou *IPv6* e posteriormente enviado ao destino. Por ser tão simples, o protocolo fornece apenas os serviços de endereçamento e fragmentação.O facto de não prover confiabilidade (controle de fluxo, congestionamento, erro) indica que o mesmo não adiciona novos serviços ao protocolo *IP* (Gouveia and Magalhães, 2005).

O protocolo por si só efetua simplesmente a ligação e envia os dados, o que o torna mais rápido e menos eficiente. É usado por aplicações de vídeo, videoconferência e em todas outras aplicações que não são demasiado exigentes em relação à confiabilidade. Um dos grandes exemplos que podemos citar é a perda de pacotes na transmissão de vídeo, que não é percetível ao utilizador. Este não reordena os pacotes que chegam fora de ordem e não realiza o *handshake* para estabelecer e finalizar uma ligação (Gouveia and Magalhães, 2005).

2.5 Camada de Aplicação

2.5.1 O protocolo *FTP*

File Transfer Protocol (*FTP*) é um protocolo de transferência de arquivo usado na *Internet*. Este define a maneira pela qual os dados devem ser transferidos numa rede *TCP/IP*, tendo como objetivo partilhar arquivos e ficheiros em dispositivos acessíveis através de uma rede. Isto significa que existe uma interdependência dos sistemas de arquivos das máquinas ou (dispositivos dos clientes) e do servidor, garantindo a eficácia na transferência destas informações ou dados (Loureiro, 1998); (Marques and Guedes, 1998); (Gouveia and Magalhães, 2005).

Quanto ao modelo de serviços, o protocolo inscreve-se a um modelo cliente-servidor, ou seja, uma entidade que requisita ao cliente e a outra espera a solicitação para posteriormente efetuar as ações ao servidor. Na conexão *FTP*, dois canais de transmissão estão abertos, um canal para dados e outro para comandos (canal de controlo) (Loureiro, 1998); (Boavida and Monteiro, 2011).

Para além deste protocolo de transferência de ficheiros em ambientes em rede, o *TFTP* também desempenha um papel primordial, vistos que a transferência de ficheiro é umas das primordiais tarefas e das mais importantes (assim como as aplicações que neles são explorados). O mesmo facilita operações remotas por meio de acesso à rede para o envio e a receção (*upload*) *put* e o *download* (*get*) em máquinas remotamente acessíveis, assim como P2P que é também um protocolo de transferência de ficheiros (Boavida and Bernardes, 2012).

Este protocolo é uma aplicação elástica que funciona em arquitetura Cliente/ Servidor. Os clientes ficam conectados ao servidor a partir do programa cliente, após cumprir os procedimentos requeridos na aplicação, para facilitar a comunicação, e depois de obterem credencias de autenticidade (ou seja, autenticação do utilizador) (Boavida and Bernardes, 2012). Atualmente existem várias aplicações clientes, cada qual com sua interface para utilização do *FTP*. Para além das interfaces usadas por cada aplicação, o *FTP* faz a utilização de duas interfaces, nomeadamente a interface gráfica, que para além da linha de comando, facilitando o processo de acesso remoto os ficheiros de forma visível sem ter que conhecer os comandos da aplicação (Boavida and Bernardes, 2012).

2.5.2 O protocolo *HTTP*

HiperText Transfer Protocol (*HTTP*) é o protocolo de comunicação simples entre sistemas de comunicação usado para ter acesso a todas as aplicações via *Web*. Foi originalmente desenvolvido para recuperação de recursos estáticos baseados em texto, progredindo posteriormente para imagens, vídeos e entre outros (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

O protocolo define como os navegadores *Web* (clientes) requisitam páginas de servidores *Web*. Quando um utilizador requisita um objeto, através da referência de uma página *Web*, o navegador solicita uma mensagem de requisição HTTP ao servidor *Web*. Este por sua vez, recebe a requisição e responde com uma mensagem de resposta *HTTP* que contém os objetos solicitados (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

Para requisitar uma página *Web*, o cliente *HTTP* abre uma conexão *TCP* com o servidor. Assim que a conexão *TCP* se estabelecer é realizada a troca de mensagens entre o cliente e o servidor através das portas de interface (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013). Existem duas versões do *HTTP* implementadas pelos navegadores: o (*HTTP/1.0* e *HTTP/1.1*). Ambas as versões usam *TCP* como protocolo de transporte devido as suas características especificas (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

O *HTTP/1.0* utiliza as conexões não persistentes após a requisição de cada objeto, o servidor responde e encerra a conexão *TCP*. Por exemplo, uma página *web* é composta de um arquivo base *HTML* e algumas imagens JPEG. Logo após a receção de cada arquivo, a conexão *TCP* é encerrada e deverá ser reaberta por cada novo objeto requisitado, sendo que este processo é realizado automaticamente da parte do utilizador (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

A versão *HTTP/1.1* permitiu melhorar o desempenho dos navegadores *Web* através da utilização de conexões persistentes onde o servidor mantém a conexão *TCP* aberta após o envio da resposta. Deste modo, as requisições e as respostas subsequentes entre o mesmo par cliente/servidor pode utilizar a mesma conexão já aberta, eliminando o tempo de abertura de conexão. Se eventualmente esta mesma conexão deixar de ser utilizada por um certo tempo, o servidor encarrega-se de libertar (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013). Este protocolo é baseado no paradigma solicitação/resposta, e dois tipos de mensagens: Mensagens de requisição e de resposta, que consiste num ou mais cabeçalhos, cada um em uma linha em branco separada e um corpo de mensagem opcional (Loureiro, 1998); (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

2.5.3 O protocolo *SMTP*

Simple Mail Transfer Protocol (*SMTP*) é o protocolo de aplicação mais simples para o funcionamento do correio eletrónico. Este protocolo usa o serviço de transferência de dados confiável do *TCP* para transferir uma mensagem desde o remetente até ao destinatário. O *SMTP* possui um paradigma semelhante aos outros protocolos de aplicação: possui dois lados, o do cliente e o do servidor. Quem envia uma mensagem desempenha o papel do cliente e quem recebe desempenha o papel do servidor, no entanto, ambos os lados do *SMTP* devem ser implementados em cada servidor de *e-mail*. O protocolo funciona juntamente com o *POP3*,

que também é um dos protocolos usados para o intercâmbio de email através de Internet (Loureiro, 1998); (Kurose and Ross, 2013).

As mensagens estabelecidas pelo *SMTP* são mensagens em caracteres *ASCII*. O processo de transferência de mensagens é feito quando o cliente *SMTP* estabelece uma Conexão *TCP* com o servidor *SMTP*. Depois de estabelecida a conexão *TCP*, clientes e servidores de *e-mail* trocam informações (endereços de *e-mail* do emissor e do destinatário) antes de enviar a mensagem eletrónica entre si (Boavida and Monteiro, 2011); (Kurose and Ross, 2013); (Boavida and Bernardes, 2012). O *SMTP* foi desenvolvida com um único propósito, a transmissão de mensagens de texto simples, vistos que na altura da criação do protocolo não havia recursos capazes, isto é, circuitos com capacidade de transmissão muito baixa (Boavida and Monteiro, 2011); (Boavida and Bernardes, 2012).

2.5.4 O protocolo *POP*

Post Office Protocol (*POP*) é um protocolo que permite ao cliente correr um programa de manipulação de correio eletrónico, ou seja, as mensagens que lhe são destinadas são depositadas numa caixa de correio num servidor *SMTP*. Este também suporta o *POP3*, é um protocolo de recolha, é instalado quando se faz a instalação de clientes ou servidores de *e-mail* e pode ser usado via *Telnet* como o *SMTP* (Loureiro, 1998); (Boavida and Monteiro, 2011). Um outro protocolo de leitura semelhante ao *POP3* e o *Interactive Mail Acess Protocol* (*IMAP*) (Kurose and Ross, 2013).

O *POP3* permite que o usuário do correio eletrónico possa visualizar as mensagens de e-mail sem que esteja ligado à *Internet* (*offline*). Esta caraterística do protocolo estar em modo (*offline*) é de particular importância para os utilizadores que se ligam à *Internet* através de redes públicas comutadas, onde o custo da ligação é proporcional ao tempo de ligação (a título de exemplo temos a rede telefónica convencional). Com este protocolo a ligação precisa apenas estar ativa durante o processo de transferência das mensagens. A leitura e o processamento destas mensagens podem posteriormente ser feita com a ligação inativa. O *POP3* é a versão mais recente, e em comparação com as versões anteriores funciona com auxilio de comandos de texto totalmente diferentes. Os comandos: *HELLO, FOLDER, READ, DELET, SAVE*, entre outros, são pertencentes ao *POP2* e, os comandos como *User Login, Stat* e tantos outros pertencem ao *POP3* (Boavida and Monteiro, 2011); (Kurose and Ross, 2013).

2.5.5 O protocolo *SNMP*

Simple Network Management Protocol (*SNMP*) é um protocolo simples de gerência de rede. Este tem como premissa a flexibilidade e facilidade de implementação, desenvolvido por um grupo de pesquisadores de *Internet Engineering Task Force* (*IETF*). Definido a nível da camada da aplicação, foi criado com a finalidade de facilitar a monitorização e a gestão da rede. O protocolo é utilizado para obter informações de servidores *SNMP*, agentes espalhados numa rede baseada na pilha de protocolos *TCP/IP*. Os dados são obtidos através de requisição de um agente, ou mais agentes, usando os serviços do protocolo de transporte *UDP* para o envio e receção de mensagens através da rede (Kurose and Ross, 2013).

Os computadores e dispositivos de rede, como *routers, hubs, switches,* ou outros desde que suportam o *SNMP*, enviam informação quanto ao seu comportamento para um computador designado por SNMP *manager*, permitindo deste modo que os administradores de rede possam gerir remotamente os componentes envolvidos na rede informática, prevenindo e resolvendo os possíveis problemas na rede (Loureiro, 1998)

Em função das necessidades, e com as exigências que começaram a surgir, o *IETF* fez a publicação da versão *SNMPv1* que foi justamente desenvolvido para dar respostas às necessidades que o *SNMP* não conseguia resolver. Em função desta, e com a abrangência do problema, foi necessário o desenvolvimento da versão *SNMPv2*, que foi uma versão que trouxe varias divergências nas suas variações (tais como: *SNMPv2p, SNMPv2c, SNMPv2u*, entre outros). Nestas variantes, as operações protocolares são as mesmas para todas as variantes do *SNMPv2*. O *SNMPv3* surge para colmatar os problemas e melhorar a performance, definições de segurança e comunicação entre agentes, tornando-se assim um padrão de protocolo para *Internet* (Loureiro, 1998).

O gerenciamento de uma rede pelo protocolo *SNMP* é constituído por três elementos primordiais, nomeadamente dispositivos geridos, o agente e o gerente. Este protocolo define sete (7) tipos de mensagens ou *Protocol Data Unit* (*PDU*) entre eles temos: *GetRequest, GetNext Request, GetBulkRequest, SetRequest, InformRequest, ResponsePDU* e *Trap*. Estas mensagens geralmente são transportadas pelo pacotes UDP na porta 161, exceto as *Traps* que faz o uso da porta 162 (Zeltserman and Zeltserman, 1999); (De Oliveira et al., 2010).

2.5.6 DNS *(Domain Name System)*

A sigla *DNS* surge das iniciais de *Domain Name System*, podendo também ter a designação de *Server* ou *Service*. A principal função do *DNS* é a de converter nomes em *IPs* e vice-versa, sendo que essas informações são conservadas e partilhadas. A necessidade do *DNS* vem pelo facto de os computadores na *Internet* não serem identificados por nomes, mas sim por endereços de *IP*, assim, quando se escreve, por exemplo, o endereço **www.uminho.pt**, se na realidade aceder a uma máquina na *Internet* terá um endereço de *IP* específico, neste caso é o 192.22.2.66. O servidor de *DNS* vai tentar descobrir a que endereço corresponde o determinado nome e, caso não consiga, encaminha o pedido para outro servidor DNS. Caso o mesmo assim não consiga, encaminha-o para o outro servidor e assim sucessivamente até o endereço *IP* ser encontrado (Gouveia and Magalhães, 2005). Em termos gerais diria que o *DNS* opera principalmente em duas funções, examina e atualiza o banco de dados e resolve sobre tudo os nomes de domínios em endereços de rede (Gouveia and Magalhães, 2005); (Boavida and Monteiro, 2011); (Kurose and Ross, 2013).

Caso o *DNS* não esteja a funcionar correctamente corre-se o risco dos nossos pedidos serem mal encaminhados, o que irá afetar não só a navegação na *Internet*, mas também o envio de *e-mail*. Para garantir a entrega de mensagens o *DNS* faz o uso dos protocolos *UDP* e o *TCP*. O *UDP* é utilizado na porta 53 quando as mensagens *DNS* entre o cliente e o servidor são encaminhadas como mensagens de aplicação. O *TCP* é usado quando é feita comunicação entre um servidor *DNS* primário e um servidor *DNS* secundário, isto é, quando o *DNS* primário faz a transferência de uma cópia da sua zona para o servidor secundário (Gouveia and Magalhães, 2005).

2.5.7 O protocolo *Telnet*

O *Telnet* é um protocolo padrão da *Internet* que permite obter uma interface de terminais e de aplicações através desta. Este utiliza o protocolo *TCP* para enviar dados em formato *ASCII* codificados em 8 *bits* entre os quais se intercalam sequências de controlo *Telnet* (Kurose and Ross, 2013). É um protocolo básico, no qual outros protocolos *TCP/IP* (*FTP, SMTP, POP3*, entre outros) se apoiam. O *Telnet* não menciona autenticação, porque é um protocolo totalmente separado dos aplicativos que o utilizam. O *FTP* é um protocolo que define uma sequência de autenticação acima do *Telnet* (Kurose and Ross, 2013).

Uma das desvantagens do *Telnet* é realizar transferência de dados sem protecção, o que quer dizer que os dados circulam abertamente na rede, ou seja, não implementa o mecanismo de criptografia. O protocolo foi construído considerando três conceitos fundamentais: o paradigma do Terminal de Rede Virtual (*NVT*), o princípio de opções negociadas e uma visão simétrica de terminais e processos (Kurose and Ross, 2013).

Para estabelecer o serviço de emulação de terminal com um cliente denominado *Telnet-Server* que ocorre serviço *Telnet*, o protocolo usa o *TCP* na porta 23, ao estabelecer a ligação, o utilizador é convidado a identificar-se através de um nome e respetiva *password*, correspondentes à sua conta no servidor *Telnet* (Loureiro, 1998).

Capítulo 3

ESTUDO DO MODELO *IoT*: DESCRIÇÃO DAS PRINCIPAIS FUNÇÕES PROTOCOLARES POR CAMADAS

O objetivo deste capítulo é apresentar uma visão geral sobre os modelos *IoT* identificando a sua influência. Por outro lado, traz reflexões discutidas por vários autores, apresentando a correspondência entre as arquiteturas e modelos arquiteturais, bem como a descrição das principais funções protocolares por camada.

3.1 Modelo *IoT*

Partindo de pressupostos das arquiteturas de *Internet*, é evidente que as arquiteturas *IoT* nascem da mesma forma como nasceram as arquiteturas *TCP/IP*, fruto do desenvolvimento tecnológico, da miniaturização dos dispositivos e das redes de sensores, em que as limitações energéticas foram as principais preocupações da comunidade científica. Assim sendo, a nova abordagem relativamente à comunicação começou a ser pensada à volta das redes de sensores, integrando-as nas redes de *Internet* e dando origem a *Internet* das coisas (Said and Masud, 2013).

O aspeto sensorial foi a parte inovadora incorporada nos dispositivos físicos, o que permitiu, por um lado, a criação de protocolos e aplicações específicas para os respetivos sensores em função das suas características e necessidades. Um dos exemplos são os órgãos sensoriais dos seres racionais (homem) e não racionais. Estes têm a função de captar os estímulos e ambientes do meio circundante e do seu próprio corpo, estes estímulos são

transmitidos na forma de impulsos elétricos até ao sistema nervoso central, que por sua vez, processa informações, traduzindo sensações e gerando respostas (Seeley, 2005).

Neste âmbito, os dispositivos sensores possuem características semelhantes ao exemplo acima mencionado, recolhendo informações do exterior por meio de sensores neles incorporados que atuam como estímulos, transportando os dados para o mundo virtual, isto é, aplicações (Seeley, 2005).

Na perspetiva desta reflexão, pesquisadores criaram modelos arquiteturais para *IoT* baseados nas arquiteturas de *Internet* e observando o modelo *OSI*. Procederam à adequação da realidade das redes de sensores em função das suas características (Said and Masud, 2013). Dada a sua complexidade, várias reflexões foram feitas por pesquisadores que tentaram criar modelos de referência, dividindo entre três a cinco camadas funcionais (Said and Masud, 2013).

Segundo o Khan (Khan et al., 2012), apresenta o modelo arquitetural de *IoT* dividido em cinco camadas funcionais: a camada de perceção, de rede, de *middleware*, de aplicação e a de negócios, especificando deste modo as tecnologias usadas por cada uma. Estas camadas são visíveis na Figura 3.1.

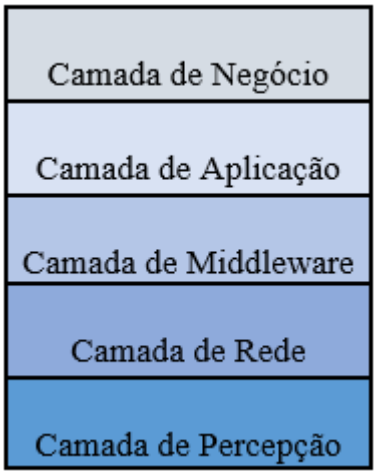

Figura 3.1: Modelo arquitetural de *IoT*
(Khan et al., 2012)

O Agrawal (Agrawal and Das, 2011), observando as características dos disposetivos sensores, em que os mesmos precisam estar conetados a rede, recolhendo informações e posteriormente serem processados, pela lógica fez a divisão do modelo arquitetural para *IoT* em três camadas: camada de perceção, de rede e de aplicação.

Na visão do Mari Carmen Domingo (Domingo, 2012), relativamente a esta matéria,

propõe uma arquitetura na perspetiva técnica, dividindo o modelo arquitetural para as *IoT* em três camadas principais: camada de perceção, de rede e de aplicação.

Segundo a reflexão do Xiaolin Jia (Jia et al., 2012), apresenta o modelo *IoT* dividido em três camadas fundamentais: camada de perceção, de rede e de serviço. Nesta vertente, o Jia comunga o mesmo pensamento dos outores que o antecederam, levando em consideração a ideia de que as arquiteturas *IoT* geralmente são dividas em três camadas, pese embora as nomenclaturas introduzidas serem diferentes do seu pensamento (Jia et al., 2012).

Segundo o Said (Said and Masud, 2013), apresenta afirmações de uma arquitetura adaptativa para *IoT* dividindo em quatro camadas: camada física, de interconexão, de dados e, por fim, de serviços. Este pensamento foi abraçado pelo Michele Mercaldi (Mercaldi et al., 2013) que introduziu o pensamento apresentando o modelo *IoT* divido em quatro camadas fundamentais a ter em consideração: camada física e Mac, de rede, de transporte e de aplicação.

É evidente, que as camadas protocolares dos modelos discutidos por estes autores definem apenas os conceitos, bem como as especificações das funções a serem desempenhadas por cada uma das camadas. No entanto, não especificam exatamente os protocolos a serem usados por estas camadas, vistos que em cada cenário é implementado um protocolo específico para resolver um determinado problema, originando assim uma arquitetura específica para cada situação (Boavida and Monteiro, 2011).

Os modelos arquiteturas mencionados acima por diversos autores, têm reflexões semelhantes relativamente a função que cada uma das camadas desempenham (apesar dos modelos apresentados não possuírem as mesmas divisões por camadas arquiteturais).

Os autores que apresentam as divisões dos modelos arquiteturais em três camadas, fazem reflexão mais direta, observando a parte concreta dos dispositivos sensores, nomeadamente a perceção onde os dispositivos sensores captam informação do exterior, a camada de rede que permite a ligação para partilhar a informação recolhida e a camada de aplicação que é responsável por prover serviços de aplicações. Esta camada separa a existência de comunicação em rede entre processos de diferentes máquinas, permitindo deste modo a integração dos protocolos de nível mais alto (Boavida and Monteiro, 2011).

3.2 Principais Funções por Camadas do Modelo *IoT*

3.2.1 Camada de Perceção

A camada de perceção é constituída por objetos físicos e os dispositivos sensores. Estes dispositivos usam tecnologias como *RFID*, sensores infravermelho e código de barras. As principais funções desta camada são identificar os objetos e captar os estímulos e ambientes do meio circundante através dos dispositivos sensores. A informação pode ser sobre a localização, temperatura, entre outras, dependendo do tipo dos dispositivos sensores (Khan et al., 2012).

3.2.2 Camada de Rede

A camada de rede ou de transmissão, é a camada responsável pela transferência de dados recolhidos nos dispositivos sensores para o sistema de processamento. A transmissão se dá de forma segura, transferindo as informações da camada de perceção para a camada de *middleware*, usando as tecnologias de transmissão com ou sem fio, nomeadamente as tecnologias 3G, *Wi-fi*, *Bluetooth*, *UMTS*, infravermelho, *ZigBee*, entre outras (Khan et al., 2012).

3.2.3 Camada de *Middleware*

Esta camada é responsável por gerenciar serviços, ou seja, é a camada adicional situada entre nível aplicacional e a camada de rede, a mesma possui uma ligação com o banco de dados. Esta camada recebe as informações da camada de rede e armazena-as no banco de dados. Cada dispositivo *IoT* conecta-se e comunica-se apenas com os dispositivos que implementam o mesmo tipo de serviço (Khan et al., 2012).

3.2.4 Camada de Aplicação

A primordial função desta camada é de fornecer o gerenciamento global em função da aplicação ou do aplicativo com base nos dados processados por objetos na camada de *middleware*. É nela onde se encontram as diversas aplicações implementadas pela *IoT* (Khan et al., 2012).

Noutros modelos arquiteturais, apresentado por vários autores, a camada de aplicação tem sido a última das arquiteturas protocolares assim como o modelo arquitetural *OSI* e a arquitetura *TCP/IP*. É nesta camada que se encontram os protocolos de nível mais alto, provendo serviços para aplicações de maneiras a separa a existência de comunicação em redes entre processos de diferentes computadores (Boavida and Monteiro, 2011).

3.2.5 Camada de Negócio

Esta camada é a responsável pelo gerenciamento do sistema global, bem como as aplicações *IoT* e serviços. Nela são construídos os modelos de negócios, os gráficos e fluxogramas, entre outros, com base nas informações ou dados recebidos (Khan et al., 2012).

As funções dos modelos apresentados acima, tal como o modelo *OSI* da *ISO* referenciado no capítulo anterior, não especificam exatamente os protocolos a serem usados por camadas dos modelos apresentados, mas sim as principais funções de cada camada. Dai que, os protocolos criados para resolução dos problemas específicos, mediante as características e necessidades de comunicação, deram origem as arquiteturas protocolares (Said and Masud, 2013); (Marques and Guedes, 1998); (Tanenbaum, 2003).

3.3 Arquiteturas *IoT* (*CoAP* e *6LoWPAN*)

.

As arquiteturas protocolares diferem dos modelos arquiteturais, também são organizadas por camadas, não existindo, contudo, uma estrutura formal como as definidas nos modelos arquiteturais. Geralmente, nas arquiteturas protocolares procura-se definir um protocolo próprio para cada camada, bem como a interface de comunicação entre as camadas adjacentes (Guth et al., 2016); (Gomez and Paradells, 2010).

Estabelecendo a correspondência entre o modelo arquitetural e arquiteturas protocolares apresentadas pelos autores já referidos, é notório que os níveis que constituem as arquiteturas protocolares, em determinados casos, nem sempre são iguais aos que constituem os modelos arquiteturais. Por exemplo, um modelo arquitetural pode ter quatro camadas ou níveis e uma arquitetura protocolar pode ter menos ou mais camadas protocolares em relação ao modelo

arquitetural e vice-versa. Em alguns casos, pode ser igual a correspondência a nível das camadas arquiteturais em relação ao modelo em causa. Este facto é visível na Figura 3.2

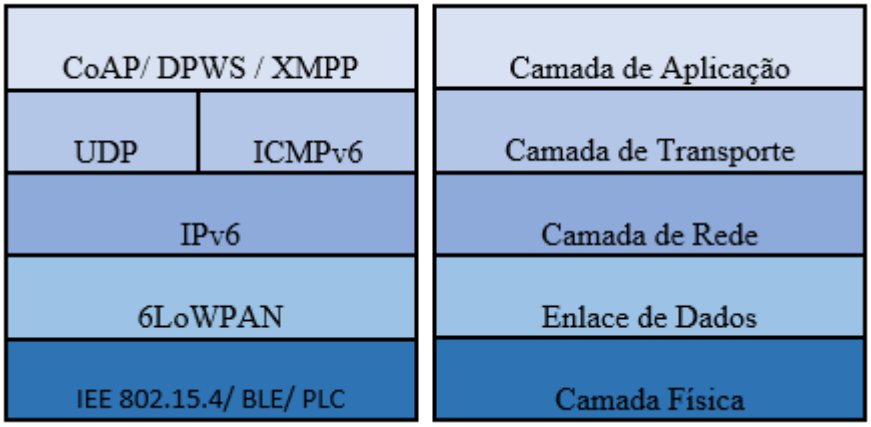

Figura 3.2: Arquitetura protocolar *6LoWPAN* e o modelo arquitetural *IoT* (Khan et al., 2012)

No caso dos modelos arquiteturais que são constituídas por três camadas ou níveis não há correspondência numérica com a arquitetura protocolar apresentada na Figura 3.2.

A reflexão feita por vários autores relativamente a este modelo arquitetural é bastante clara e heterogénea em termos de nomenclatura. O pensamento foi baseado em função das características dos dispositivos sensores, tais como os outros modelos mencionados anteriormente (Agrawal and Das, 2011); (Jia et al., 2012)..

A Figura 3.3 estabelece uma relação entre a arquitetura *CoAP* e o modelo *IoT* constituida por três camadas. É claramente visível os desníveis numéricos entre o modelo *IoT* e arquitetura *CoAP* apresentado e, este é constituído por cinco camadas funcionais, onde os protocolos que constituem os níveis arquitetural cumprem os requisitos funcionais estabelecidos pelo modelo *IoT* representado.

A camada de perceção do modelo *IoT* corresponde à primeira camada da arquitetura *CoAP*, a segunda camada do modelo *IoT* (camada de rede) correspondem à segunda e a terceira camada da arquitetura *CoAP* e, finalmente, a camada de aplicação do modelo *IoT* corresponde as duas últimas camadas da arquitetura *CoAP* representados pelos protocolos *UDP* e o *CoAP*.

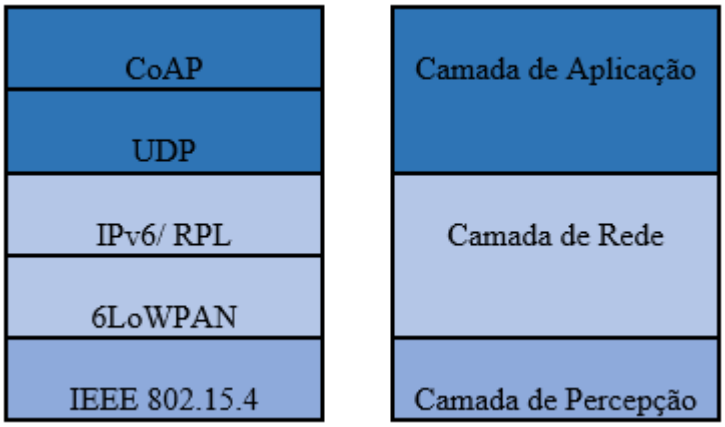

Figura 3.3: Relação entre arquitetura *CoAP* e o Modelo *IoT* (Agrawal and Das, 2011); (Jia et al., 2012)

3.4 Principais protocolos *IoT* e a sua organização a nível das camadas

Os protocolos *IoT* surgem pela crescente necessidade dos dispositivos sensores recolherem uma vasta gama de informação e pela necessidade de perceber determinados fenómenos que nos circundam, Além disto, é necessário ter o domínio e o controlo dos dispositivos do uso comum, nesta vertente, pesquisadores desenvolveram protocolos para integrar os dispositivos na *Internet*, visto que as suas características impossibilitam o uso total de protocolos das arquiteturas *TCP/IP* diretamente nos dispositivos sensores (Mercaldi et al., 2013).

Em função destas características, muitas adaptações têm sido feitas em torno das arquiteturas protocolares *IoT*, fazendo o uso de protocolos apropriados desenvolvidos por especialistas nesta área, dando origem as arquiteturas protocolares, cumprindo metodologias pouco diversas das metodologias usadas no desenvolvimento dos modelos acima mencionado, entre as quais as arquiteturas *6LoWPAN*, *CoAP*, entre outros (Boavida and Monteiro, 2011); (Said and Masud, 2013).

Uma vez que estes protocolos fazem o uso de um poder computacional bastante baixo, fruto das características dos dispositivos sensores desenvolvidos com esta finalidade, a sua miniaturização e a consequente integração na *Internet* é um dos fatores que torna estes dispositivos cada vez mais importantes. Tal como é representado a organização de protocolos por camadas na Figura 3.4 (Boavida and Monteiro, 2011); (Said and Masud, 2013); (Mercaldi et al., 2013).

Entre os principais protocolos *IoT* que constituem as arquiteturas protocolares *CoAP* e

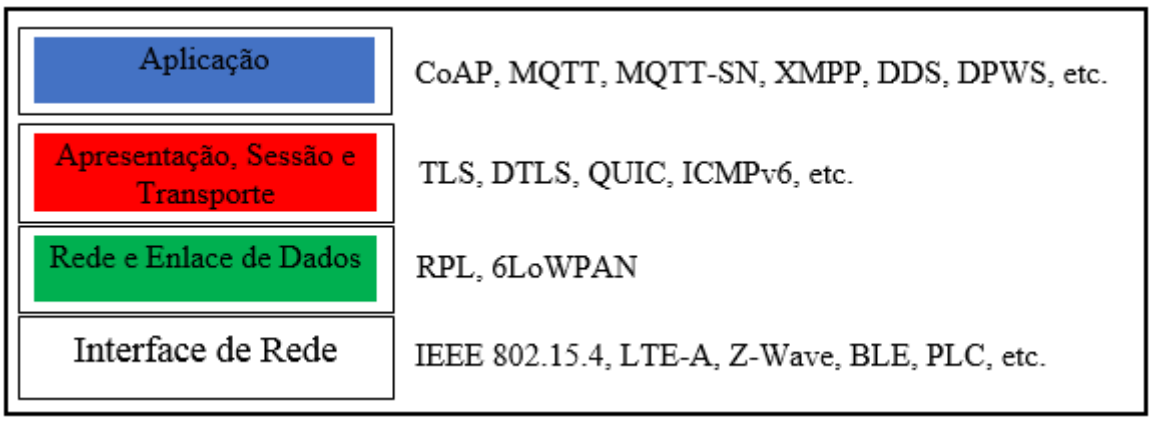

Figura 3.4: Protocolos *IoT* organizados por Camadas
(Said and Masud, 2013); (Khan et al., 2012)

6LoWPAN (representados na figura acima), existem outros protocolos *IoT* que deram origem ao surgimento de novas arquiteturas em função da sua complexidade e do problema que se pretendia (ou pretende) resolver (Said and Masud, 2013); (Al-Fuqaha et al., 2015); (Luzuriaga et al., 2015).

3.5 Camada de Interface de Rede

A camada de interface de rede é a uma das principais para qualquer que seja o sistema de comunicação, uma vez que funciona como interface entre as funções lógicas de comunicação e o meio físico. Sem esta camada não seria possível a comunicação entre sistemas. Ela possui múltiplas funções, como os dispositivos sensores que fazem a utilização de meios físicos sem fio. Possuem particular importância a modulação e desmodulação de sinais, os mecanismos de sobrevivência e interferência, a sincronização entre os dispositivos que constituem um sistema, a gestão de potências de transmissão, entre outros. Os dispositivos sensores dispõem de transcivers (mecanismos de transmissão e receção), cujo a função é o de adaptar o sinal lógico a transmitir (isto é, sequências de zero e um) (Weber et al., 2016); (Rahman and Shah, 2016); (Trelsmo et al., 2017).

3.5.1 Padrão *IEEE 802.15.4* (*Wi-fi*)

O *IEEE 802.15.4* foi desenvolvida pelo *Institute of Electrical and Electronics Engineers*, e é visto como um padrão de referência para automação industrial e doméstica, etc. É desenvolvido para redes de sensores sem fio de baixa potência, cujo tempo de vida útil das

baterias deve ser otimizado. A tecnologia tem sofrido várias atualizações e emendas, originando diferentes versões com base nas pilhas protocolares, nomeadamente o *ZigBee*, *Wireless HART*, etc (Weber et al., 2016); (Akbar et al., 2017); (Pawlowski et al., 2014).

O *IEEE 802.15.4* suporta dois tipos de topologias, em estrela e ponto-a-ponto. Na topologia em estrela, os nós comunicam-se diretamente com um nó central que assume o papel de coordenador. Por obrigatoriedade este nó deverá ser um *Full-function Devices* (*FFD*). Cada rede nesta topologia é uma estrutura independente e é reconhecida como identificador denominado *Personal Area Network Identifier* (*PAN-ID*), enquanto que a rede ponto-a-ponto o nó que recebe a função de coordenador passa necessariamente por um processo de seleção. O nó escolhido como nó coordenador será responsável pela adição dos demais na rede. Nesta topologia, os nós podem comunicar-se com qualquer outro nó na sua vizinhança, cada nó deverá ser um *FFD* (Weber et al., 2016); (Akbar et al., 2017); (Pawlowski et al., 2014).

3.5.2 O protocolo *LTE-A*

O *Long Term Evolution Advanced* (*LTE*) é um protocolo que foi desenvolvido para o sistema de telefonia móvel. Este é um padrão bastante avançado, diferenciando-se pela forma de acesso. As tecnologias *UMTS* e *HSPA* são baseadas no padrão *W-CDMA*, utilizando as especificações de Acesso Múltiplo por Divisão Ortogonal de Frequência, do inglês (*Orthogonal Frequency Division Multiple Access* - *OFDMA*). O protocolo não cumpre com todas as exigências técnicas necessárias para ser considerado um padrão da 4G, mas é aceite no âmbito comercial. O *LTE-A* é versão mais atualizada do *LTE* e é compatível com os requisitos da *International Telecommunication Union* (*ITU*) para uma tecnologia 4G, oferecendo taxas de até 1Gbps para *download* e 500 Mbps para *upload*. O protocolo já foi utilizado nas redes *IoT* devido à sua eficiência e baixo custo, e é um dos protocolos bastante promissora para as redes *IoT* (Kurose and Ross, 2013); (Zanella et al., 2014); (Houda et al., 2014).

O Padrão 4G *LTE* apresentado por (*3rd Generation Partnership Project* - 3GPP) por não apresentar as exigências 4G, apresenta duas inovações importantíssimas em relação aos sistemas 3G, nomeadamente o núcleo de pacote desenvolvido e a rede de acesso por rádio *LTE* (Kurose and Ross, 2013).

3.5.3 O protocolo *Z-Wave*

O *Z-Wave* é um protocolo de comunicação sem fio desenvolvido para ambientes domiciliares e domínios comercias de pequeno porte. Disponibiliza alcance em frequências bastante curtas, bem como a transmissão de dados em comunicação ponto-a-ponto, sendo ideal para as comunicações *IoT* em ambientes domiciliares (Zanella et al., 2014). Tendo sido publicado pelo *Z-Wave Alliance*, ele é compatível com todos os eletrodomésticos e outros dispositivos de ambiente domiciliar. O protocolo foi considerado uma boa escolha para as redes *LLNs* e *IoT* (Zanella et al., 2014).

Em relação à transferência de informação, este garante a confiabilidade na entrega de mensagens curtas da unidade de controlo para um ou mais dispositivos com um ruido bastante inferior durante o processo de comunicação. Os dispositivos na rede são considerados como um nó incluído na rede que faz a utilização do protocolo *Z-Wave*. Desta forma distinguem-se os dispositivos da mesma rede assim como a própria rede por meio de um *ID*, e os nós na vizinhança da rede (Yassein et al., 2016); (Al-Sarawi et al., 2017). Uma das grandes vantagens deste protocolo é permitir o uso da configuração da rede *Mesh*, isto significa, que cada dispositivo ou nó na rede envia e recebe comandos de controlo que ultrapassam obstáculos, permitindo a cobertura domiciliar, utilizando de igual modo nós intermediários e estabelecer rotas entre os dispositivos na rede para comunicar-se com o destinatário que receberá informação. Este protocolo tem a possibilidade de transcender a faixa de rádio utilizados nos dispositivos de rede (Yassein et al., 2016); (Al-Sarawi et al., 2017).

3.5.4 O protocolo *BLE*

A tecnologia *Bluetooth Low Energy* (*BLE*) ou *Bluetooth Smart*, faz utilização do poder energético bastante baixo, aumentando o tempo de vida dos dispositivos, podendo mesmo durar vários anos. Possui uma cobertura de 100 metros de alcance, isto é, dez vezes superior em termos de cobertura quando comparado ao *Bluetooth original*, quanto a latência é quinze vezes menor. Estabelecendo comparação com o *IEEE 802.15.4*, o *BLE* em dispositivos reais consideráveis é extremamente baixo e possui uma boa proporção energética consumida por *bit*. O protocolo pode funcionar entre faixas de transmissão que oscilam entre os 0.01 mW à 10 mW, esta é uma das características que torna o *BLE* uma forte tecnologia para aplicações *IoT*. O padrão foi desenvolvido rapidamente por fabricantes dos dispositivos móveis,

maioritariamente para os modelos *smartphone*. A tecnologia foi demonstrada como sendo viável para a comunicação veicular assim como nas redes de sensores sem fio, fornecendo uma grande solução para as redes sensores aplicando os modos de segurança (tais como a criptografia e assinatura de integridade de dados) (Al-Fuqaha et al., 2015).

3.5.5 O protocolo *PLC*

O *Power Line Communications* (*PLC*) é uma tecnologia que foi desenvolvida com a finalidade utilizar as redes de distribuição elétrica como meio físico para o transporte de sinais de informação. Esta ideia não é nova, dado que os grandes objetivos do protocolo é transmitir sinais de voz em banda larga pela rede de energia elétrica utilizando a infraestrutura já disponível. As redes *PLC* podem ser de comunicação interna ou externa (*indoor power line Communication*) e (*outdoor power line Communication*). A comunicação de linha elétrica interna de uma rede *PLC* existe em ambientes fechados, nomeadamente casas e edifícios comerciais. A sua configuração básica é constituida por um equipamento *Master* que é instalado próximo ao transformador de baixa tensão. Este têm como função gerenciar, distribuir ou concentrar a transmissão das informações aos equipamentos (modems) dos assinantes (De Oliveira et al., 2010); (Chauvenet et al., 2017).

Nestas redes internas, as fontes de interferência são causadas por dispositivos ou equipamentos de iluminação, pequenos geradores e pequenos equipamentos elétricos. Ao passo que nas redes de ambientes abertos, o sinal de rede *PLC* viaja através do sistema elétrico de transmissão e distribuição. O seu desempenho e prejudicado pelo relâmpago, comutação, transformadores, motores elétricos e bancos de capacitores. Os enlaces de comunicação de uma rede *PLC* são estabelecidas no segmento de rede de distribuição de energia elétrica. Como o principal desafio dos dispositivos de rede sem fio é funcionar com bateria, estas redes são vistas como meio de comunicação para as redes *IoT* (De Oliveira et al., 2010); (Chauvenet et al., 2017).

3.6 Camada de Rede e de Enlace de Dados

Esta sessão apresenta os principais protocolos da camada de rede e ligação de dados, uma vez que as funções de endereçamento, encaminhamento, controlo de fluxo, entre outros, são tratados por estas camadas de acordo com o modelo *OSI* da *ISO* (Weber et al., 2016).

Nas redes sensores e *IoT*, os protocolos têm sido desenhados com o objetivo de minimizar o consumo energético, existindo outros protocolos que suportam diferentes requisitos como o atraso e fiabilidade (Weber et al., 2016).

3.6.1 O protocolo *RPL*

O *Routing protocol for Low-Power and Lossy Network* (*RPL*) é um protocolo de encaminhamento de vetor de distâncias que foi desenvolvida para atender as necessidades das redes locais (*LLNs*). O protocolo define uma árvore de encaminhamento utilizando o conceito de grafos acíclicos direcionados (*Directed Acyclic Graph - DAG*) onde cada nó de rede pode associar mais de um nó de rede. Esta é uma das características fundamentais que difere o *RPL* dos demais protocolos baseados em árvores. A organização do protocolo é baseada no destino, ou seja, a rede possui um nó principal que concentra a receção dos dados ou informação dos outros nós, esta estrutura recebe o nome do (*Destination Oriented Directed Acyclic Graph - DODAG*) com outras características bastante importantes como a autoconfiguração, detenção de impedimentos de *loops*, o uso de múltiplos encaminhadores de borda e entre outros (Rahman and Shah, 2016).

O *RPL* funciona na camada *IP* de acordo com a arquitetura, permitindo o encaminhamento entre vários tipos de ligações como o *IEEE 802.15.4, IEEE 802.15.4G* e entre outros. Os encaminhadores que se conectam as redes *6LoWPAN* para outras redes *IP* normalmente funcionam como raízes *DODAG - RPL* (Rahman and Shah, 2016).

A arquitetura *route-over RPL* foi especificado pelo *IETF* em que o encaminhamento é implementado na camada de rede de acordo com a arquitetura *IP*. Esta abordagem implementa funções na camada de ligação para emular um único domínio de *broadcast*, onde todos os dispositivos aparecem como vizinhos para a camada de rede. Desta forma o *route-over* coloca todas as funções de encaminhamento na camada de rede (Rahman and Shah, 2016).

3.6.2 O protocolo *6LoWPAN*

O *IPv6 over Low Power Wireless Personal Area Networks* (*6LoWPAN*) faz a utilização do protocolo *IPv6* nas redes *IEEE 802.15.4*. As abordagens pertinentes desta arquitetura baseiam-se na fragmentação/desfragmentação e compressão de cabeçalhos *IPv6*. No *IPv6* a unidade máxima de transmissão é de 1280 octetos enquanto na rede *IEEE 802.15.4* é de

27 *bytes*, restando 81 octetos para os pacotes de dados, ou seja, possui uma baixa largura de banda (250 Kbps, 40 Kbps e 20 Kbps), permitindo que os dispositivos fiquem inativos por um período de tempo e, por outro, maximiza o tempo de vida de bateria, permitindo a fragmentação e compressão de cabeçalhos, assim como adaptação da camada *IPv6* e o *IEEE 802.15.4* nesta arquitetura. A utilização do protocolo *6LoWPAN* possibilita que os dispositivos *IoT* consigam ter um endereço *IP* devido as suas características, como por exemplo, o baixo consumo energético, o que faz com que a tecnologia *IEEE 802.15.4* seja a mais indicada para os dispositivos com estas características (Weber et al., 2016).

O *6LoWPAN* serve para traduzir os pacotes do tipo *IPv6* para as redes *IEEE 802.15.4* comprimindo o tamanho sem a necessidade de ação do utilizador e sem a perda de dados na comunicação. Dado a performance do protocolo, permite também a adaptação entre a camada de rede e ligação de dados (Weber et al., 2016).

3.7 Camada de Apresentação, Sessão e Transporte

Na arquitetura *IoT* as camadas de apresentação, sessão e transporte constitui uma única camada, diferente do modelo *OSI* da *ISO*. Para além da descrição destes protocolos existem outros protocolos *IoT* que completam esta camada. Neste caso, a reflexão é feita a volta dos protocolos que constituem as arquiteturas protocolares.

3.7.1 O protocolo *TLS*

O *Transport Layer Security* (*TLS*) é um protocolo que foi projetado para prover segurança através da criptografia, a interoperabilidade, a extensibilidade e eficiência. Tudo isto significa dizer que o protocolo é capaz de estabelecer uma conexão segura entre duas entidades, garantindo a comunicação entre ambas na rede. Protocolos como *CoAP, MQTT, XMPP, AMQP, DDS*, entre outros, fazem o uso do *TLS* para fornecer segurança na comunicação de redes *IoT*. Uma das caraterísticas do protocolo é encapsular os protocolos de aplicação como *HTTP* e o *FTP*, sendo que funciona por cima de um protocolo de transporte como *TCP* e o *UDP*. Para que haja confiabilidade na transmissão, o *TCP* deve ser usado uma vez que o *UDP* esta sujeito à perda de informação devido as suas características, nomeadamente o controlo de fluxo, a entrega ordenada de pacotes, a confiabilidade, orientação da conexão, permissão da conexão ponto-a-ponto, entre outros (Zanella et al., 2014); (Urien, 2013).

3.7.2 O protocolo *DTLS*

O *Datagram Transport Layer Security* (*DTLS*) é um protocolo com abordagens semelhantes ao protocolo *TLS*, não existindo novas versões para trabalhar em ambientes *IoT*. O mesmo fornece segurança para a camada de transporte para comunicações *UDP*. Tal como o *TLS*, o *DTLS* também é usado por protocolos *CoAP, MQTT, XMPP, AMQP, DDS*, entre outros, para fornecer segurança na comunicação de redes *IoT*. Os dispositivos inteligentes devem ser capazes de fazer o uso destas tecnologias quando usadas em aplicações regulares de *Internet*, que basicamente oferecem criptografia na conexão entre os nós, permitindo a troca de chaves durante o período em que a conexão é estabelecida (Zanella et al., 2014). Este protocolo de segurança foi concebido para desempenhar as funções semelhantes ao *TLS* de formas a maximizar a quantidade de código e permitir a reutilização da infra-estrutura, fornecendo segurança e por outro lado reduzindo a necessidade de utilizar *IPsec* ou estruturar um protocolo de segurança de camada das aplicações personalizadas (Zanella et al., 2014).

3.7.3 O protocolo *QUIC*

O *Quick UDP Internet Conections* (*QUIC*) é um protocolo de transporte experimental que foi construído por cima do protocolo *UDP*, anunciado pela *GOOGLE* como um protocolo moderno para ambientes *IoT*, o objetivo primordial deste protocolo é reduzir a latência fim-a-fim e fornecer aspetos de segurança equivalente ao protocolo *TLS/SSL*. Este protocolo possui características que o tornam importante para ambientes *IoT* tais como a migração de conexão, que é a qualidade que permite um nó *QUIC* usar o mesmo endereço *IP* de conexão mesmo quando se desloca de uma rede para outra. Este cenário só é possível pelo facto das conexões *QUIC* serem identificadas por *IDs* de conexão em vez de endereços de origem/destino e porta de origem/destino usada pelas conexões *TCP*. Um dos grandes problemas do protocolo *QUIC* aplicada em ambientes *IoT* é a utilização de recursos, pois não minimiza os mesmos recursos para os dispositivos com poder energético baixo (Zanella et al., 2014); (Fischlin and Gunther, 2017)

3.7.4 O protocolo *ICMPv6*

O *IPv6 Internet Control Message Protocol* (*ICMPv6*) é um protocolo que funciona na camada de transporte usado nos nós *IPv6* para fornecer mensagens (relatos) de erros do

nó de origem. As mensagens *ICMPv6* geralmente são enviadas em determinadas situações, nomeadamente quando um determinado pacote *IP* não consegue chegar ao destino, caso o tempo de vida do pacote expirar ou quando um *gatway* não consegue transmitir os pacotes na frequência adequada, como por exemplo o cenário de congestionamento. Por outro lado, temos também os casos em que tamanho dos pacotes forem bastantes grandes, destino inacessível, problemas de parâmetros, entre outros (Boavida and Bernardes, 2012); (Wu et al., 2009).

O *ICMPv6* usa abordagens semelhantes do *ICMP*, que é encarada como um protocolo pertencente à camada de rede, embora utilize protocolos *IP* como protocolos de suporte e a informação que veicula seja em regras passadas para as camadas superiores ou, até mesmo, aos processos de aplicação para o processamento e eventual ação. Um dos exemplos que se baseia neste protocolo são os comandos *ping* e a aplicação *traceroute*, utilizados por administradores de redes ou por utilizadores comuns(tendo sido definido no RFC 792 para o caso de *IPv4* e no *RFC* 2463 para o caso do *IPv6*) (Boavida and Bernardes, 2012); (Wu et al., 2009). A transição de *IPv4* a *IPV6* e a necessidade de interligar as redes sensores para *Internet*, permitiu que o *IETF* propusesse a integração do *IPv6* nas *RSSF*, dando a particular atenção às funcionalidades que se encontram relacionadas com os requisitos das *RSSF*, originando assim o *ICMPv6*, que é a versão mais avançada do *ICMPv4*, usado na arquitetura protocolar *6LoWPAN* (Weber et al., 2016).

3.8 Camada de Aplicação

Nesta sessão, são apresentados os principais protocolos *IoT* organizados em função das tarefas que exercem nas arquiteturas. Cada protocolo foi desenvolvido para resolve um determinado problema, as extensões protocolares surgem para colmatar o problema que um determinado protocolo não consegue resolver, (podendo mesmo haver a criação de um protocolo específico para resolução do problema em causa).

3.8.1 O protocolo *CoAP*

O *Constrained Applications Protocol* (*CoAP*) é um protocolo que utiliza o modelo cliente/servidor, desenvolvido por um grupo de pesquisadores de *IETF* com o objetivo de transferir dados na *Web* para os dispositivos com capacidade energética baixo. O *UDP* é utilizado pelo protocolo *CoAP*, tornando-o bastante leve e com menor consumo computacional

e energético. A sua arquitetura é semelhante o protocolo *HTTP* e faz uso dos métodos e princípios *RESTful*, sendo um dos primeiros ícones definidos pelo *CORE*. Os fabricantes de dispositivos *IoT* fazem o uso do protocolo *CoAP* devido às suas caraterísticas, suportando também a descoberta de recursos e serviços. Provê ainda um modelo de requisição/resposta entre os *endpoints* das aplicações (Agrawal and Das, 2011). O *CoAP* é um protocolo *Web* que permite a troca de mensagens assíncronas, a interligação segura para (*Datagrama Transport Layer Security - DTLS*), suportando os métodos *GET, POST, PUT e DELET*. Por outro lado, permite o mapeamento do *HTTP* de forma que os proxies possam prover acesso aos recursos do *CoAP* via *HTTP* de maneira uniforme (Rahman and Shah, 2016); (Zanella et al., 2014); (Karagiannis et al., 2015).

3.8.2 O protocolo *MQTT*

O *Message Queue Telemetry Transport* (*MQTT*) é um protocolo de mensagens bastante simples e totalmente leve. A sua simplicidade e código fonte aberto torna este protocolo adequado apenas para ambientes restritos, com baixa potência e capacidade de computação limitada (assim como a limitação de memória e a largura de banda) (Karagiannis et al., 2015). Este protocolo faz o uso do *TCP* e garante a confiabilidade na entrega de pacotes fornecendo um conjunto de recursos que inclui o suporte de comunicação *multicast* e a capacidade de estabelecer a comunicação entre os dispositivos remotos. Uma das principais características deste protocolo é a capacidade de minimizar o tráfego de rede de formas a reduzir a sobrecarga e intercâmbio de protocolos, permitindo a notificações quando há anomalias na rede (Al-Fuqaha et al., 2015); (Karagiannis et al., 2015).

O *MQTT* também oferece mecanismos para a melhoria de qualidade se serviço (QoS) dentre as quais o One delivery (*at last, at most*) e o *On delivery exactly*. O *at most* é o nível mais baixo da *QoS* e as mensagens são entregues de acordo com o melhor esforço de rede. Com o *at last*, as mensagens são enviadas pelo menos uma vez. Neste paradigma as mensagens duplicadas podem existir, assim sendo, é necessário uma mensagem de confirmação. Finalmente, *On delivery exactly* exige um protocolo adicional para garantir a entrega da mensagem apenas uma vez (Karagiannis et al., 2015); (Al-Fuqaha et al., 2015).

3.8.3 O protocolo *MQTT-SN*

O *Message Queue Telemetry Transpot for Sensor Network* (*MQTT-SN*) é um protocolo *publish/subscribe* para redes sensores sem fio. Este foi projetado para melhorar o protocolo *MQTT*. A sua melhoria tornaria o *MQTT-SN* mais adequado para as redes de baixa potência (do inglês *Low Power and Lossy Network - LLNs*) e para os dispositivos restritos. O *MQTT-SN* não é um novo protocolo, mas sim uma adaptação do protocolo *MQTT*, sendo que o protocolo *MQTT-SN* mantém as suas características mais próximo possível do *MQTT*. A diferença entre os dois é o facto de *MQTT-SN* não necessitar, ou não exigir o protocolo *TCP/IP* para o seu funcionamento. Este, por sua vez, permite agrupar um número mais considerável de nós na rede do que o protocolo *MQTT*, já que este não tem a preocupação de saber se o nó esta a fazer o uso do *TCP/IP*, *Bluetooth, ZigBee* ou qual quer outro protocolo que preencha o pré-requisito para estabelecer a comunicação atravéz do *MQTT-SN* Zanella et al. (2014); Karagiannis et al. (2015).

3.8.4 O protocolo *XMPP*

O *Extensible Messaging and Presence Protocol* (*XMPP*) é um dos protocolos mais comum na *IoT*, o protocolo é bastante conhecido e padronizado pelo *IETF* e foi usado amplamente em todas as redes. A necessidade da *IoT* fazer o uso deste protocolo vêm do facto do protocolo *XMPP* suportar pequenas e baixas latências. Esta é, sem sombras de dúvidas, uma das características primordiais que torna o protocolo *XMPP* uma das boas escolhas para comunicações e mensagens *IoT*. Ele é bastante flexível e pode ser usado em aplicações heterogéneas. Este protocolo é fornecido por arquitetura descentralizada, e há existência de muitas extensões deste protocolo, o que permite trabalhar num ambiente sem infraestrutura (Karagiannis et al., 2015). O *XMPP* usa o *XML* para comunicações de texto, isto deve-se à sobrecarga de tráfego de rede, mas pode ser resolvido através da compreensão do *XML* usando o *EX1* (Al-Fuqaha et al., 2015); (Karagiannis et al., 2015).

3.8.5 O protocolo *AMQP*

O *Advanced Message Queuing Protocol* (*AMQP*) é um protocolo de padrão aberto que funciona na camada de aplicação, desenvolvido por *Advancing Open Standards for the Information Society* (*OASIS*). O protocolo é orientado para entrega de mensagens, dependendo

da fila de mensagem confiável e eficiente, fazendo o uso de abordagem de assinatura/ publicação, tornando-se um protocolo de alta escalabilidade. Ele não funciona por si só, faz o uso de protocolo de transporte confiável, como protocolo de controle de transmissão (*TCP*). O seu principal objetivo é garantir a entrega confiável das mensagens usando abordagens semelhantes ao protocolo *MQTT*. O *AMQP* é um protocolo muito importante para as *IoT* devido às características de segurança aplicadas em fluxo de dados em tempo real e é amplamente usado em plataformas comerciais e industriais. A sua segurança é garantida pelos protocolos *TLS* e o *SASL* (*Simple Authentication and Security Layer*). No entanto, não é adequado para aplicações em tempo real (Zanella et al., 2014); (Karagiannis et al., 2015).

3.8.6 O protocolo *DDS*

O *Data Distribuition Service* (*DDS*) é um protocolo que foi criado em resposta à necessidade da indústria de padronizar os sistemas centrados em dados. É bastante importante para ambientes *IoT* por suporta aplicações *IoT* e comunicação máquina-a-máquina (*M2M*), que visa fornecer conectividade de dados de baixa latência, boa confiabilidade e uma arquitetura escalável (características que as aplicações comerciais e de cenários críticos da *Internet* de todas as coisas certamente precisarão). O protocolo não apresenta problemas mesmo com o aumento de nós nas redes de sensores sem fio. Uma das grandes características deste protocolo é a redução de tráfego desnecessário na rede, denominado fusão e filtragem (Zanella et al., 2014); (Karagiannis et al., 2015).

A fusão faz-se agrupando os dados num único pacote, reduzindo deste modo o tráfego da rede e exigindo menos largura de banda. Este é um dos fatores que melhora a eficiência na comunicação da rede. É relevante considerar que a filtragem é um recurso em que os dispositivos inteligentes filtram os conteúdos desejados. Por exemplo, sensores de temperatura em ambientes domiciliares automatizados que fazem recolha de dados e publicam valores de temperatura numa rede (Al-Fuqaha et al., 2015); (Zanella et al., 2014).

3.8.7 O protocolo *DPWS*

O *Device Profile for Web Services* (*DPWS*) é um protocolo que especifica um mecanismo de evento para o perfil de dispositivos sensores, baseando-se numa arquitetura orientada para serviços e usando tecnologias de serviços da *Web*. Esta tecnologia é dividida em dois princípios

fundamentais que são baseados na tecnologia de transporte de mensagens, como o *SOAP* e o *REST*. Os serviços *Web* baseados em *SOAP* são conhecidas como *WSSOAP* (*Wireless Sensor SOAP*) e *DPWS-WS*, enquanto que os serviços com base em *REST* são conhecidos como *RESTful-WS*. O protocolo de eventos *DPWS* fornece ferramentas para subscrição de eventos, sendo que as assinaturas podem ser canceladas, renovadas e expiradas ao longo do tempo, definindo assim modo de entrega único de mensagens e permitindo um mecanismo de mensagens assíncronas simples chamado modo de envio (Samaras et al., 2013); (Izaguirre et al., 2011).

Capítulo 4

COMPARAÇÃO ENTRE PROTOCOLOS A NÍVEL DAS FUNÇÕES POR CAMADAS ARQUITETURAIS (*TCP/IP* E *IoT*)

Nem todos os protocolos que se encontram na mesma camada arquitetural têm a mesma finalidade. Esta secção estabelece uma comparação entre alguns protocolos a nível das suas funções nas diferentes arquiteturas, tendo em atenção a convergência dos mesmos face às suas limitações e a importância que têm quando integrados com a *Internet*.

4.1 Camada Física

A camada física, ou interface de rede, tem como função estabelecer a ligação entre computadores ou dispositivos de rede ligados de forma adjacente. Nesta secção, abordar-se-a padrões de *Internet* e de *IoT*, onde os protocolos criados obedecem os aspetos da comunicação de acesso entre os principais protocolos nas distintas tecnologias, observando pormenores comuns, apesar das realidades tecnológicas serem diferentes.

4.1.1 Padrões de *Internet* e *IEEE 802.15.4*

O *IEEE 802.15.4*, desenvolvido pelo *Institute of Electrical and Electronics Engineers*, é visto como um padrão de referência para automação industrial, doméstica, entre outros. Este padrão, criado para redes sensores sem fios de baixa potência, cujo tempo de vida útil das baterias deve ser otimizado, é definido por um conjunto de regras, restrições e técnicas protocolares em redes de comunicação sem fios para a camada física e para a camada de

ligação de dados dos dispositivos com poder energético baixo. As tecnologias têm sofrido várias atualizações e emendas, originando diferentes versões com base nas pilhas protocolares, nomeadamente o *ZigBee, Wireless HART, 6LoWPAN*, etc (Akbar et al., 2017); (Pawlowski et al., 2014); (Yick et al., 2008).

A tecnologia *IEEE 802.15.4* suporta dois tipos de topologias, em estrela e ponto-a-ponto. Na topologia em estrela, os "nós" comunicam diretamente com um "nó" central que assume o papel de coordenador. Por obrigatoriedade, este "nó" deverá ser um FFD (*Full-function Device*). Cada rede nesta topologia é uma estrutura independente e é reconhecida pelo identificador denominado PAN-ID (*Personal Area Network Identifier*), enquanto na rede ponto-a-ponto o "nó" que recebe a função de coordenador passa necessariamente por um processo de seleção. Este por sua vez será responsável pela adição dos demais "nós" na rede. Nesta topologia, os "nós" podem comunicar com qualquer outro "nó" na sua vizinhança, sendo cada um deles um FFD (Weber et al., 2016); (Akbar et al., 2017); (Pawlowski et al., 2014).

4.1.2 Padrão *IEEE 802.3*

O protocolo *Ethernet* pertence ao padrão *IEEE 802.3* e é compatível com a tecnologia *IEEE 802.15.4*, tendo em consideração que as tecnologias fazem uso das topologias acima mencionadas, aproveitando o padrão *IEEE 802.15.4* para as redes sensoriais e protocolos padrão de *Internet*, oferecendo às camadas mais baixas da rede um tipo de rede WPAN com baixo custo e comunicação de baixa velocidade. O padrão consiste, fundamentalmente, em três elementos: o meio físico, as regras de controlo de acesso ao meio e finalmente o quadro *Ethernet*. Tanto o padrão *IEEE 802.15.4* como o *IEEE 802.3* são similares, tanto em relação à funcionalidade como pela forma como são definidas as informações que são transportadas pela rede através do meio físico e não só, agrupando os dados que são entregues pelos protocolos de alto nível, fazendo a sua inserção dentro dos quadros (*frames*) que serão enviados através da rede (Loureiro, 1998); (Severi et al., 2014); (Kurose and Ross, 2013).

O padrão *IEEE 802.3* possui uma arquitetura bastante utilizada pelas redes locais, pois facilita a sua montagem, a manutenção da rede a custos baixos e apresenta um desempenho considerável (Kurose and Ross, 2013). As redes de *Ethernet* fazem a utilização dos baseband e o controlo de emissão de informação sob forma de pacotes designados (frames), através de um mecanismo denominado CSMA/CD (*Carrier Sense Media Access with Collision Detection*).

Quando uma placa de rede pretende injetar um *frame* na cablagem, o mecanismo CSMA/CD faz a detenção de fluxo gerado por outras placas de rede, contudo, caso não detete os *frames* a circularem na rede, a placa transmite somente aquilo que há a transmitir. Geralmente, as colisões acontecem quando duas placas de rede fazem, em simultâneo, a injeção de *frames* na rede (Loureiro, 1998); (Kurose and Ross, 2013).

4.2 Protocolos de Comunicação

É importante salientar que os protocolos *Z-Wave, PLC, BLE,* etc, fazem parte da norma *IEEE 802.15.4*, e que os protocolos *Token-Ring* e *PPP* pertencem à norma *IEEE 802*. Um dos grandes objetivos desta última norma é o de especificar as funções da camada física e do controlo de acesso ao meio de uma rede sem fios. A norma *IEEE 802* pode ser aplicada aos mais variados tipos de rede nomeadamente (Kurose and Ross, 2013); (Tanenbaum, 2003):

(i) 802.3 - *Ethernet*;

(ii) 802.11 - Redes locais sem fios (*WLAN*);

(iii) 802.15 - Redes de área pessoal se fios (*WPAN*).

Ao passo que a norma *IEEE 802.15.4* define um conjunto de regras, restrições e técnicas a nível protocolar em redes de comunicação sem fios para a camada física e para a ligação de dados em dispositivos com poder energético baixo, tal como foi dito anteriormente acerca da definição dos tipos de rede que vão desde *802.15.1* até *802.15.8*, onde os protocolos IoT fazem parte da norma *IEEE 802.15.4*, e especificando para camada física as seguintes características (Antunes, 2012); (Tanenbaum, 2003):

(i) existência de três bandas de frequência em 27 canais, definindo as modulações nas respetivas bandas;

(ii) possibilidade de configurar vários níveis de segurança;

(iii) o endereçamento dos dispositivos na rede é feito de forma automática, vistos que o protocolo *IEEE 802.15.4* é responsável por permitir a transmissão das PDUs*Protocol Data Units*, uma que esta transmissão de dados é realizada através de ondas rádio;

(iv) permite que a configuração das mensagens sejam realizada por meio de tramas do tipo *beacon*;

(v) o acesso ao meio é feito por CSMA-CA (*Carrier Sense Multiple Access with Collision Avoidance*).

4.2.1 O protocolo *Z-Wave*

O *Z-Wave* é um protocolo de comunicação sem fios, que funciona em frequências curtas, aproximadamente de 868.42 MHz (Europa) e 908 MHz (EUA), fazendo uso de uma baixa largura de banda para enviar comando de controlo. Este protocolo, publicado pelo *Z-Wave Alliance* e considerado uma boa escolha para as redes *LLNs* e *IoT*, é compatível com as comunicações *IoT* em ambiente domiciliário, sendo igualmente aplicado nas áreas da indústria e medicina (Zanella et al., 2014); (Yick et al., 2008).

O protocolo *Z-Wave* garante a confiabilidade na entrega de informação, permitindo a transmissão de mensagens curtas da unidade de controlo para um ou mais dispositivos, com pouco ruído ou interferências de redes *Wireless*, sistemas *Bluetooth*, telefones sem fios ou dispositivos de transmissão áudio/vídeo durante o processo de comunicação. Os dispositivos na rede são considerados como um "nó" incluído na rede que faz a utilização do protocolo *Z-Wave*. A utilização de um ID no protocolo, permite que sejam identificados os dispositivos da mesma rede, a própria rede, bem como os dispositivos ou "nós" na vizinhança desta (Yassein et al., 2016); (Al-Sarawi et al., 2017); (Fouladi and Ghanoun, 2013).

Uma das grandes vantagens do *Z-Wave* é facilidade que permite a configuração da rede *Mesh*, o que significa que cada dispositivo ou "nó" na rede envia e recebe comandos de controlo que permite o alcance do sinal a longas distancias. Isto concede à cobertura domiciliar, fazer uso de "nós" intermediários para estabelecer rotas entre os dispositivos na rede, e fazer chegar os dados ou informação ao destinatário. Para além disto, a propagação do sinal utilizado pelo *Z-Wave* tem a possibilidade de transcender a faixa de rádio utilizada nos dispositivos de rede de *Internet* (Yassein et al., 2016); (Al-Sarawi et al., 2017); (Fouladi and Ghanoun, 2013).

4.2.2 O protocolo *BLE*

O protocolo (*Bluetooth Low Energy - BLE*) é uma tecnologia de transmissão sem fios, desenvolvido com a finalidade de melhorar o desempenho do consumo energético, aumentando assim o tempo de vida de dispositivos. Hoje em dia, vários dispositivos móveis, como *Smartphone*, estão a implementar o *BLE Hardware*, aumentando a capacidade em termo

de recursos, o que facilita o suporte do Sistema Operativo. Estas características permitem que o protocolo tenha uma taxa de transferência bastante reduzida (Chandan and Khairnar, 2018);.

O *BLE* é a versão melhorada do *Bluetooth* original (versão 4.0), que é totalmente limitado em termos de serviços e capacidades que oferece devido as suas caraterísticas, e não permite, em alguns casos, conexões entre os dispositivos *BLE*, esta versão cria somente a topologia em estrela, diferentemente das versões que sucedem a versão 4.0 com possibilidades de criar topologias em malha, estrela e ponto-a-ponto (Yassein et al., 2016); (Houda et al., 2014). Pelo contrário, a versão melhorada do *BLE* permite uma cobertura superior à do *Bluetooth* original em termos de alcance, possui uma cobertura dez vezes maior que este e uma latência quinze vezes inferior (Chandan and Khairnar, 2018); (Guo et al., 2015); (Zanella et al., 2014);. Este padrão pode ser usado em ambientes domiciliários, na medicina, no desporto, na comunicação veicular, entre outros, sendo uma grande solução para as redes de sensores e *IoT* ao aplicar modos de segurança, nomeadamente a criptografia e a assinatura de integridade dos dados (Chandan and Khairnar, 2018); (Al-Fuqaha et al., 2015); (Guo et al., 2015).

A versão 4.1 do *BLE* foi concebida no final de 2013 e, em relação ao *Bluetooth* original, é menos exigente em termos de requisitos ou recursos que oferece. Como exemplo, temos o seu modo inativo quando os dispositivos ficam afastados da conexão, voltando a conectar-se somente quando há necessidade de transmissão de dados. O modo funcional da primeira versão já existia no *Bluetooth* original, tendo sido melhorado na versão 4.1. Para além das versões acima mencionadas, outras versões foram criadas em função das necessidades de comunicação dos dispositivos e de recolha de informação. A versão 4.2 do *BLE* tem suporte para o *IPv6*, tornando a tecnológica mais importante para as *IoT*. Esta versão foi apresentada no final de 2014, sendo que a versão 5 foi apresentada oficialmente em 2016 e tem um raio de propagação superior de 40 metro. Esta versão possui técnicas para reduzir o risco de interferências em redes *Wi-Fi* ou *LTE*, podendo mesmo suportar vários dispositivos conectados ao mesmo tempo (Chandan and Khairnar, 2018); Al-Fuqaha et al. (2015); Guo et al. (2015).

Em relação à convergência entre as tecnologias *Internet* e *IoT*, existe uma semelhança na criação de algumas topologias de redes de *Internet*, nomeadamente a topologia anelar e *Mesh* que são implementadas pelo *BLE*, este protocolo de comunicação tinha como objetivo substituir a comunicação serial (Loureiro, 1998).

4.2.3 O protocolo *LTE-A*

O protocolo *LTE-A* é a versão atualizada do protocolo (*Long Term Evolution - LTE*). Este protocolo, desenvolvido especificamente para os serviços de telefones móveis, começou a ser implementado no 3G, sendo que os projetos do 4G já estavam a ser realizados. Este protocolo tem duas inovações importantes em relação ao *3GPP* (*3rd Generation Partnership Project*): o Núcleo de Pacote Desenvolvido (*Evolved Packet Core - EPC*) e a rede de acesso por rádio *LTE*, fornecendo uma velocidade até 150 Mbps de *downlink* e uma velocidade de ligação de *uplink* até 50 Mbps numa determinada área geográfica. Quanto à velocidade de 150 Mbps de *downlink*, a largura de banda de cada usuário dependerá de como as operadoras de telefones móveis implementam a sua rede, bem como a disponibilidade da largura de banda (Kurose and Ross, 2013); (Yassein et al., 2016) (Zanella et al., 2014); (Houda et al., 2014).

A *EPC* é uma rede de núcleo simplificado em *IP*, que unifica a comutação de voz da rede de telefones móveis por circuitos separados e a rede móvel de dados comutada por pacotes. Esta rede é baseada em *IP*, uma vez que os dados são transportados em datagramas *IP* (Kurose and Ross, 2013); (Yassein et al., 2016).

O modelo de serviço do *IP* não é totalmente adequado aos requisitos de desempenho exigentes do tráfego *VoIP* (*Voice over Internet Protocol*), a menos que estes recursos sejam cuidadosamente controlados para evitar congestionamento na rede. Uma das tarefas primordiais do *EPC* é controlar os recursos da rede para oferecer uma alta qualidade de serviço, observando de igual modo os aspetos de mobilidade e a transferência contínua que foram requisitos para o gerenciamento central para todos os dispositivos na rede, permitindo assim a inclusão de vários tipos de rede de acesso (2G e 3G) que se conectam ao núcleo da rede Kurose and Ross (2013).

A rede de acesso de rádio *LTE* utiliza as especificações de acesso múltiplo por Divisão Ortogonal de Frequência, do inglês (*Orthogonal Frequency Division Multiple Access - OFDMA*) que é uma técnica de transmissão de informação. Utiliza a sua banda em múltiplas portadoras ortogonais, que são denominadas de sub-portadoras para modulação, e não apresenta a sobreposição de frequências de forma a evitar a ocorrência de interferência de uma com a outra. Esta técnica tem uma forte vantagem em relação às que utilizam uma e única portadora, podendo alcançar a mesma taxa de transferência devido ao paralelismo de sub-portadoras de taxas baixas, já que tem uma maior resistência às diversas circunstâncias do meio ambiente. O *LTE* não cumpre com todas as exigências técnicas necessárias para ser considerado um padrão

do 4G, mas é aceite no âmbito comercial. Esta versão protocolar já foi utilizada nas redes *IoT* devido à sua eficiência e as caraterísticas que apresenta, que o torna compatível para as redes *IoT* (Zanella et al., 2014); (Kurose and Ross, 2013); (Houda et al., 2014).

A tecnologia utiliza múltiplas antenas de transmissão e receção, permitindo assim a multiplicidade da capacidade de transmissão e explorando a propagação de multicaminhos (*MIMO*), que é uma técnica incorporada em diversos padrões de comunicação, devido ao desempenho que elas proporcionam, como por exemplo *WiMax, HSPDA*, entre outros que são usadas pelo *LTE*. O *LTE-A* surgiu para aumentar o desempenho da rede, a velocidade e a utilização de frequências, permitindo a agregação de cinco portadoras no máximo (Zanella et al., 2014); (Kurose and Ross, 2013); (Houda et al., 2014).

Uma das limitações dos protocolos *IoT* e *Internet* que fazem uso das tecnologias sem fios nos dispositivos sensores em ambientes abertos, diz respeito justamente à continuidade de cobertura devido a determinados fatores, tais como: os edifícios espalhados pelas cidades, a configuração do meio geográfico, entre outros, são exemplos de alguns fatores que podem influenciar a má qualidade do sinal e consequentemente na comunicação, visto que as redes sem fios só se podem conectar a dispositivos de rede no raio de alcance das mesmas, dando assim continuidade ao sinal a ser transmitido, ao contrário dos dispositivos de rede móvel que fazem uso do protocolo *LTE-A* (Yassein et al., 2016); (Zanella et al., 2014); (Houda et al., 2014); (Kurose and Ross, 2013).

A utilização de placas *BLE* e dos seus protocolos incorporados em dispositivos móveis utilizando o protocolo *LTE-A*, facilitaria a integração das redes *IoT* na *Internet* devido às características específicas e às vantagens oferecidas por este protocolo. Neste sentido, o *LTE-A* permitiria manter os dispositivos conectados na rede e o *BLE* facilitaria a localização do dispositivo na troca de informação entre os dispositivos na rede (Kurose and Ross, 2013).

4.2.4 O protocolo *PLC*

O protocolo (*Power-Line Communication - PLC*) é uma tecnologia que foi desenvolvida com a finalidade de utilizar as redes de distribuição elétrica como meio físico para o transporte de sinais de informação. Esta reflexão já vem de longe, e o protocolo tem como um dos grandes propósitos a transmissão de sinal em banda larga pela rede de energia elétrica, utilizando a infraestrutura já disponível. A tecnologia faz a utilização de uma banda de frequência alocada para comunicações de ondas curtas e outra parte da faixa de frequência da rádio VHF, utilizada

pelo setor público (polícia, bombeiros, entre outros), através de um intervalo de frequência entre 1,7 MHZ e 30 MHZ (De Oliveira et al., 2010); (Chauvenet et al., 2017).

As redes *PLC* dividem-se em dois tipos: comunicação de linha interna (*indoor power-line communication*) e comunicação de linha externa (*outdoor power-line communication*). A comunicação interna existe em ambientes fechados, nomeadamente a casa e edifícios comerciais, uma vez que as infraestruturas domiciliares e industriais já possuem redes elétricas instaladas. A configuração da rede é feita por um equipamento *Master* que é instalado próximo ao transformador de baixa tensão que têm como função gerenciar, distribuir ou concentrar a transmissão das informações aos equipamentos (*modems*) dos assinantes De Oliveira et al. (2010); Chauvenet et al. (2017).

As desvantagens apresentadas por estas redes são as interferências causadas por dispositivos ou equipamentos de iluminação, pequenos geradores e pequenos equipamentos elétricos. Nas redes de ambientes abertos, o sinal de rede *PLC* viaja através do sistema elétrico de transmissão e distribuição, tal como foi referenciado no capítulo anterior. O seu desempenho é prejudicado por vários fatores (relâmpagos, comutação, transformadores, motores elétricos e bancos de capacitores). Os pontos ou enlaces de comunicação de uma rede *PLC* são estabelecidos no segmento de rede de distribuição de energia elétrica. Como o principal desafio dos dispositivos de rede sem fios é funcionar com bateria, estas redes (*PLC*) são um meio de comunicação para as redes *IoT*, uma vez estes aproveitam a infraestrutura já existente. Atualmente, os equipamentos de tecnologia *PLC* têm uma taxa de transmissão de 200 Mbps, processo este que teve o seu começo na primeira geração *PLC*, em que os equipamentos *BPL* (*Broadband over Power Lines*) forneciam uma taxa de transmissão de 14 Mbps. A principal função deste equipamento é a de permitir a comunicação entre linhas elétricas *PLC*, facilitando a transmissão de dados digitais a alta velocidade sobre os cabos elétricos, ou seja, o *BPL* funciona segundo um princípio semelhante ao *DSL*, que tem a função de transformar os dados da rede Internet que são transmitidos por meio de cabos que fazem o uso de gamas de frequências de sinalização mais alta do que as utilizadas para transmitir eletricidade ou voz, no caso de *DSL* (De Oliveira et al., 2010); (Chauvenet et al., 2017); (Tanenbaum, 2003).

4.2.5 Sumário

Concluindo, os protocolos *IoT*, nomeadamente o *Z-Wave, BLE, LTE-A* e *PLC*, foram construídos com o intuito de facilitar a comunicação entre os dispositivos sensores e a *Internet*, tendo em consideração as características dos dipositivos. O objetivo dos protocolos é ajudar a minimizar os custos e o consumo energético, dando maior tempo de vida às baterias e obedecendo a padrões de comunicação entre os dispositivos na rede, pois nestas tecnologias a largura de banda é baixa (Yassein et al., 2016).

Em relação ao que foi dito, o protocolo *LTE-A*, versão melhorada do *LTE* possui fortes vantagens relativamente aos protocolos *Z-Wave* e *BLE*, vistos que o *LTE-A* é a que possui maior cobertura de sinal, desempenho considerável e que a permite a utilização de várias portadoras, uma vez que o protocolo foi construído para dispositivos da rede de telefones móveis. Os *BLE Hardware* estão a ser utilizados nestes dispositivos, auxiliando-os em termos de propagação das mensagens curtas como publicidade, entre outros, uma vez que o *Hardware* e os protocolos *BLE* são adequados em determinados Sistemas Operativos usados nos dispositivos móveis como o *Smartphone* (Zanella et al., 2014); (Kurose and Ross, 2013); (Houda et al., 2014). A grande vantagem do *PLC* consiste na utilização da infraestrutura elétrica já existente em ambientes domiciliares, pois a sua implementação em relação às outras tecnologias de rede tem um custo bastante reduzido (Loureiro, 1998); (Zanella et al., 2014).

Tanto os protocolos de comunicação *IoT* como de *Internet* têm como objetivo primordial estabelecer a ligação entre as entidades na rede. Os protocolos *Token-Ring* e *PPP* são protocolos de *Internet* situados na camada de interface de rede, que permitem estabelecer a ligação entre as entidades da rede pelo meio físico (Zanella et al., 2014); (Kurose and Ross, 2013); (Houda et al., 2014).

4.2.6 O protocolo *Token-Ring*

O *Token-Ring* é um protocolo de comunicações para redes locais, que funciona na camada física (ligação de dados) e na camada de enlace do modelo *OSI*, dependendo da sua aplicação. O termo *Token-Ring* é utilizado para se referir ao padrão *IEEE 802.5* quando é implementada uma rede *LAN*, usando a topologia anelar e fazendo a utilização de trama de três *bytes* denominado *"Token"* que percorre em redor de uma rede lógica de estações de trabalho ou servidores. Esta tecnologia foi originalmente criada pela *IBM*, sendo que alguns

adotam a nomenclatura *Token-Ring IBM*, e é a segunda tecnologia mais popular depois da *Ethernet*. O funcionamento de uma rede deste tipo é totalmente diferente, quando comparada com as redes de *Ethernet*, e pode funcionar da seguinte forma (Gouveia and Magalhães, 2005); (Boavida and Monteiro, 2011); (Tanenbaum, 2003):

- O conjunto de nós que constituem a rede *Token Ring* estão conectados em forma de anel;

- As informações ou dados fluem na mesma direção, podendo o *Token* ser utilizado por um único computador de cada vez. É uma espécie de carruagem de produtos que passa por todas as estações;

- Cada dispositivo ou "nó" na rede recebe *frames* do "nó" que antecede. Posteriormente, este *frame* é enviado para o posterior "nó" da rede, caso este detete a chegada do *Token* que funciona em permanência, verificando se os dados coincidem com a sua identificação. Se coincidir, a informação é recolhida. Caso não, este envia para próxima estação.

O anel é um meio compartilhado: apenas um "nó" que possua o *Token* transmite quadros durante um certo período. O uso deste protocolo pela topologia anelar é praticamente protegido de colisões de pacotes, permitindo um diagnóstico e soluções simples, dada a obrigatoriedade destas redes fazerem o uso de *hubs* inteligentes ou *smart hubs* com um *chip* microprocessador interno. A desvantagem do *Token-Ring* em relação às redes de *Ethernet*, é que há um aumento de escassez, devido aos custos da implementação do *Token-Ring* serem bastantes dispendiosos e com uma velocidade de transmissão limitada (16 *Mbps*) face à velocidade que é transmitida na casa de *Gbps* (Gouveia and Magalhães, 2005); (Tanenbaum, 2003).

4.2.7 *PPP (Point-to-Point Protocol)*

O *PPP* é um protocolo de rede pertencente à pilha protocolar *TCP/IP*, que funciona na camada de enlace de dados. A criação do protocolo teve como finalidade transportar informação entre dois dispositivos na rede através de uma única conexão. Ele é usado em vários tipos de meios físicos, nomeadamente cabos seriais, dados de rede móvel, fibras óticas, linhas telefónicas comuns, entre outros (Kurose and Ross, 2013).

O protocolo *PPP* é constituído por três partes, que obedecem a um diagrama de três fases no processo de interação: encapsulamento de datagramas, *LCC* (*Link Control Protocol*)

e *NCP* (*Network Control Protocol*) (Kurose and Ross, 2013); (Loureiro, 1998). O protocolo também é suportado pelo *Windows NT*, enquanto cliente ou servidor, usado no acesso aos servidores do ISP Windows NT, como cliente ou servidor (Loureiro, 1998).

O protocolo também é usado em conexões de banda larga como o *PPPoE* ou o *PPPoA*. O protocolo *PPPoE* oferece um transporte de dados simples bem como a autenticação e alocação dinâmica de IP. O processo de autenticação é feito geralmente com uma chave de acesso, ao passo que na alocação dinâmica de *IP*, o provedor de acesso tem um número limitado de endereços *IP* em relação ao número de clientes na rede. É evidente que nem todos os clientes se conectam ao mesmo tempo, por isso é possível atribuir um endereço *IP* a cada cliente no momento em que estes se ligam ao provedor, utilizando NAT (*Network Address Translation*) estático. O endereço *IP* é mantido até que a conexão *PPP* termine, podendo este ser atribuído a outro cliente (Loureiro, 1998); (Kurose and Ross, 2013).

4.3 Camada de Rede e Enlace de Dados

A camada de rede é responsável pelo controlo das operações de rede. Uma das principais tarefas consiste na comutação e execução do encaminhamento dos pacotes entre a origem e o destino, principalmente quando existem caminhos diferentes para interligarem entre si dois hosts na rede. Nas redes *LoWPAN* (*Low-Power Wireless Personal Area Networks*), as restrições de memória para processamento de grandes pacotes de dados e a reserva da comunicação nos dispositivos sensores têm inviabilizado a utilização de alguns protocolos de *Internet*. Uma vez que na arquitetura *IoT* os protocolos de rede e enlace de dados pertencem a mesma camada, tal como estão representados na Fig. 3.4 serão discutidos na presente sessão, segundo a arquitetura da referida figura.

4.4 Protocolos de Encaminhamento

4.4.1 Comparação entre *RIP, OSPF e RPL*

Tanto o *RIP*, o *OSPF* como o *RPL* são protocolos de encaminhamento, apesar de terem sido construídos para trabalhar em tecnologias e redes diferentes, como é o caso do *RPL* que foi projetado para os dispositivos ou "nó" sensor que faz uso do poder energético

baixo e capacidade de memoria bastante inferior em relação aos protocolos de *Internet*. Estes dois protocolos de encaminhamento interno (*Interior Gateway Protocol - IGP*) são usados na comunicação interna entre os encaminhadores dentro de um mesmo sistema autónomo (*Autonomous System - AS*) para troca de informações de rotas dos pacotes *IP* (Kurose and Ross, 2013).

Os algoritmos usados pelos protocolos *RIP* e *OSPF* são completamente diferentes. O *OSPF* implementa o algoritmo de estado de enlace, que é, atualmente, o mais popular entre os protocolos de encaminhamento interno. O surgimento do *OSPF* deve-se principalmente a limitações dos demais protocolos tipo *IGP*, como é o caso do *RIP*. O protocolo *OSPF* utiliza o algoritmo *Shortest Path First*, também conhecido como *Dijkstra*, para calcular os melhores caminhos, contrariamente ao método vetor-distância utilizado pelo *RIP*. Cada rota contém o identificador de interface, o número do enlace e a distância ou métrica, o que permite aos encaminhadores descobrirem as melhores rotas. Um dos conceitos mais importantes diz respeito ao cálculo do custo de um determinado caminho para um destino, onde o custo de uma interface costuma ser inversamente proporcional à largura de banda da mesma, ou seja, o custo final de cada caminho está relacionado com a qualidade do enlace (Kurose and Ross, 2013). (Boavida and Bernardes, 2012);. Este protocolo possui características que o tornam flexível, permitindo-lhe adaptar-se melhor à rede em que é implementado, entre elas temos (Kurose and Ross, 2013);(Boavida and Bernardes, 2012):

(i) a não-existência de limite no custo máximo de uma rota;

(ii) as mensagens podem ser protegidas por um *checksum*;

(iii) o protocolo possui o encaminhamento por multicaminho que é caracterizado pelo balanceamento de carga;

(iv) suporta o encaminhamento *unicast* e *multicast*: a entrega de pacote ponto-a-ponto é realizada via *unicast*, mas também para a entrega de todos os *host* conectados à rede *broadcast*;

(v) cada entrada é protegida por um temporizador e é removida da tabela se um determinado pacote de atualização não chegar num determinado tempo;

(vi) quanto à segurança, a trocas entre encaminhadores OSPF exigem autenticação, evitando assim intrusos na rede;

(vii) em relação à estruturação hierárquica por áreas, é necessário, em redes de grande

porte, que os encaminhadores não consigam de conhecer todas as possíveis rotas dentro da rede.

4.4.2 O protocolo *RIP*

Contrariamente ao protocolo *OSPF*, o *RIP* utiliza o algoritmo vetor de distância. Este algoritmo é responsável pela construção de uma tabela que informa sobre as possíveis rotas dentro do sistema autónomo. Os protocolos baseados no vetor de distância partem do princípio de que cada encaminhador do sistema autónomo deve possuir uma tabela para atualização das possíveis rotas dentro do sistema autónomo. Com base nesta tabela, o algoritmo faz a escolha da melhor rota e ligação, ou enlace, a ser utilizada (Gouveia and Magalhães, 2005); (Kurose and Ross, 2013).

O conceito de *broadcast* utilizado pelo *RIP*, permite que um determinado encaminhador envie a sua tabela em intervalos de tempo de aproximadamente 30 segundos. As mensagens que constituem as tabelas são atualizadas pelos encaminhadores vizinhos e posteriormente enviadas a estes. Cada uma destas tabelas possui as seguintes informações (Gouveia and Magalhães, 2005): O endereço *IP* dos dispositivos de rede; o próximo encaminhador de destino com base na rota; a interface utilizada para alcançar o próximo encaminhador de destino; a métrica que indica a distância da rota; e o tempo que indica o último período em que a rota foi atualizada pela última vez (Gouveia and Magalhães, 2005).

Apesar da realidade dos protocolos *IoT* e *Internet* apresentarem algumas diferenças na conceção protocolar devido as características dos dispositivos ou "nós" sensores, os protocolos de encaminhamento dividem-se em dois grupos: *Distance Vector* e *Link-State*, conforme referenciado acima. O primeiro atribui valores de custo a cada "nó" da rede, baseados em métricas usuais como a contagem do número de saltos até ao "nó" de destino, fazendo uso do algoritmo *Bellman-Ford* para calcular os caminhos. O segundo atribui valores de custo de encaminhamento às ligações existentes entre os "nós". As métricas deste grupo baseiam-se na largura de banda e no ritmo de transmissão de dados, utilizando o algoritmo *Dijkstra*, que é mais rápido do que o *Bellman-Ford* (Kurose and Ross, 2013); (Gouveia and Magalhães, 2005).

4.4.3 O protocolo *RPL*

O *RPL* é um protocolo *IoT* baseado em *IPv6*, desenvolvido especificamente para redes de baixo consumo energético, restrições de memória, entre outros. Por ser um protocolo de encaminhamento por *Distance Vector – IPv6* para *LLN's* que especificam como construir (*Destination Oriented Directed Acyclic Graph - DODAG*), utiliza Função Objetiva (*Objective Function*) para traduzir as métricas e restrições estabelecidas para uma rede num valor de *rank*. A Função Objetiva também define como os "nós" ou dispositivos selecionam qual será o principal "nó" na rede, otimizam e selecionam as rotas numa instância *RPL*. A principal tarefa da Função Objetiva consiste em calcular o melhor caminho fazendo uso destas métricas e restrições. Nesse sentido, podem também existir diversas Funções Objetivas numa operação de um único dispositivo ou "nó" na mesma rede, uma vez que o processo de implementações varia em função dos diferentes objetivos numa única rede *Mesh*, de forma a que os requisitos de qualidade e transporte de tráfego sejam efetivados (Rahman and Shah, 2016); (Gaddour and Koubâa, 2012).

Em relação aos protocolos *OSPF, RIP* e *RPL* observam-se aspetos convergentes entre eles. Os protocolos *RIP* e *RPL* utilizam o mesmo tipo de algoritmo *Distance Vector* para definir as rotas, sendo que a diferença está na forma como as rotas são definidas. O *RIP* define as rotas usando tabelas de encaminhamento por meio do algoritmo *Distance Vector*, diferentemente do *RPL* que estabelece as rotas por meio de árvores de encaminhamento através do mesmo algoritmo, mas utilizando o conceito topológico de (*Direct Acyclic Graph - DAG*). Este conceito permite que um dispositivo de rede associe mais de um dispositivo ou "nó", que é uma das características que distingue o *RPL* dos demais protocolos de encaminhamento baseados em topologias de árvore, ao contrário do *RIP* em que as decisões de encaminhamento são realizadas tendo em conta o custo das rotas, seja qual for a métrica utilizada (no *RIP*, por exemplo, são saltos) (Rahman and Shah, 2016).

4.5 Protocolos de Controlo e Gestão de Mensagens

Entre os protocolos de comunicação de *Internet* e *IoT*, cada um possui uma função específica: tanto o *ICMP* e *IGMP* são protocolos de controlo que comunicam mensagens de erros e diagnósticos, assim como o *ICMPv6* que é a versão adaptada do *IPv6* utilizada com grande frequência nas camadas de apresentação, sessão e transporte para as redes *IoT*,

conforme é representado na Figura 3.4. A nomenclatura de protocolos de controlo é atribuída neste trabalho mediante finalidade especifica exercida por cada um destes protocolos.

4.5.1 O protocolo *ICMP* e *IGMP*

O ICMP é um protocolo de mensagens de controlo usado por encaminhadores (*router*), e por clientes e servidores para transmissão de mensagens de erros entre os dispositivos na rede, tal como foi mencionado na Sessão 2.3.3 (Gouveia and Magalhães, 2005).

Sendo um protocolo integrante do *IP*, além das funções associadas ao controlo do fluxo dos pacotes enviados e recebidos, também fornece dados estatísticos pormenorizados sobre o protocolo *IP* aquando da execução do comando *netstat-s*. Para além da troca de mensagens de controlo de erro entre os encaminhadores e computadores, também fornece informações com relação à escolha de caminhos e dão controlo de fluxo de informação. Um dos exemplos que podem ser mencionado diz respeito à escolha de um caminho ou rota alternativa por um determinado encaminhador entre o emissor e o recetor, e o envio de uma mensagem ao recetor, fazendo uso do *ICMP* para informar se existe um melhor caminho para o envio de pacotes *IP*. Este tipo de informação denomina-se *ICMP Redirect Messages*, e diz respeito ao comando *tracert* que verifica o caminho pelo qual os pacotes passaram no IP até ao "nó" de destino. Outro aspeto importante tem a ver com o facto de o ICMP efetuar apenas relatos dos erros encontrados durante o processamento de pacotes e execução de algumas funções na rede, não possuindo mecanismos de correção dos mesmos (Loureiro, 1998); (Wu et al., 2009); (Gouveia and Magalhães, 2005).

As versões protocolares do *ICMP* surgiram devido a exigências aplicacionais, daí que a versão *ICMPv6* surja para resolver restrições impostas pelo *ICMP*, salvaguardando por outro lado as funções originais do *ICMP*. O protocolo começou a ser usado em dispositivos que suportavam o *IPv6* ou em dispositivos com baixo poder energético, uma vez que o protocolo possuía características compatíveis para os dispositivos sensores (Boavida and Monteiro, 2011); (Wu et al., 2009).

4.5.2 O protocolo *IGMP*

Tal como o protocolo *ICMP*, o *IGMP* também é um protocolo integrante do *IP* e também é responsável pela gestão das mensagens que circulam na *Internet* através do *TCP/IP*. Uma

das principais tarefas do *IGMP* consiste em controlar os *host* membros do grupo *multicat-IP*, policiando os respetivos membros que se associam e aos que se desassociam ao grupo, informando os encaminhadores para enviarem por *multicat* apenas os *host* pertencentes ao grupo, nomeadamente os encaminhadores e os *host* adjacentes da rede *IP* (Gouveia and Magalhães, 2005). O protocolo sofreu várias atualizações, começando pela versão 1 até a versão 3. A versão *IGMPv1* tornou-se padrão em 1998: nesta versão protocolar, os *host* podiam unir-se ao grupo *multicast* mas não conseguiam abandonar o grupo *multicast*, fazendo uso do mecanismo *Time-out* nos encaminhadores para descobrir os *host* que já não estavam interessados em continuar no grupo. A versão *IGMPv2* conseguiu implementar o mecanismo de abandono do grupo por *low leave latency*. Finalmente, a versão *IGMPv3* permitiu que os *host* especificassem as fontes do tráfego *multicast* para receber dados ou informação, e posteriormente bloquear o tráfego de outras fontes (*source filtering*) (Gouveia and Magalhães, 2005).

As mensagens *IGMP* são encapsuladas em *datagrams IP*, com *datagrams IPTL=1*, com propriedades de controlo de *Internet*, incluindo o cabeçalho e opção de *IP Router* Alerta RFC-2113 citado por (Gouveia and Magalhães, 2005). Este protocolo possui três grupos de mensagens, nomeadamente (Gouveia and Magalhães, 2005):

(i) *Membership Queries*: Este tipo de mensagem de consulta de membros é enviado ao grupo multicast, e subdivide-se em três tipos:

- *General Queries*: Esta mensagem é para aprender que grupos têm membros numa determinada rede conectada;

- *Group-Specific Queries*: Para aprender se um grupo particular tem algum membro numa rede conectada, é enviada somente a direção do grupo;

- *Group-and-Source-Specific Queries*: Serve para saber se uma das redes conectadas deseja receber pacotes que foram enviados a um grupo *multicast* específico de qualquer fonte ou lista de fonte específica.

(ii) *Membership Report*: Envia relatórios aos *host* ou máquinas em resposta a uma consulta para se unir ao grupo.

(iii) *Leave Group Messages*: Mensagem enviada a todos os encaminhadores para abandono do grupo.

4.5.3 Sumário

Concluindo, os protocolos *ICMP* e *IGMP* convergem pelo facto de serem protocolos de mensagens e integrantes do *IP*, controlando as mensagens de circulam entre os encaminhadores e os clientes na rede. Estes diferenciam-se pelo facto de *ICMP* ser um protocolo de mensagem de erros, indicando a melhor rota e controlo de fluxo de informação entre os clientes na rede através de um encaminhador, contrariamente do *IGMP* que têm a função de controlar os clientes membros do grupo *multicast-IP*, informando aos encaminhadores para enviarem mensagens *multicast* apenas os clientes que associam-se ao grupo e não aos que se desassociam ao grupo, sendo esta uma das principais tarefas.

4.6 Protocolos de Endereçamento

Apesar de os protocolos da arquitetura *TCP/IP* dependerem uns dos outros, cada protocolo envolvido possui um papel fundamental para prover a comunicação entre os dispositivos de rede, sucedendo-se o mesmo nas arquiteturas *IoT*. Este ponto estabelece uma comparação entre os dois principais protocolos de endereçamento das tecnologias distintas: *IP* e *6LoWPAN* pertencentes a camada de rede e de enlace de dados como é o caso do protocolo *IoT*.

4.6.1 O protocolo *IP*

O *IP* é um protocolo de base, apesar de o mesmo ser complementado por outros protocolos, tais como o *ARP* , o *ICMP* e tantos outros protocolos auxiliares de controlo de encaminhamento. O *IP* é um dos mais importantes protocolos de rede, visto que trata da identificação dos dispositivos envolvidos, e é responsável pelos dados em que o *TCP/IP* se baseia para o processo de encaminhamento e entrega de pacotes de informação até ao destino, através de rede de *Internet*, que é constituído por vários componentes, como *host, routers*, sistemas telefónicos, cablagens, etc (Loureiro, 1998); (Boavida and Monteiro, 2011); (Boavida and Bernardes, 2012).

O protocolo *IP* permite que os pacotes circulem na rede por meio de um processo denominado encapsulamento, carregando no seu interior pacotes *TCP* (*Transmission Control Protocol*) ou *UDP* (*User Datagram Protocol*), processados pela camada superior. Os pacotes

IP (*Ethernet frame*) são transportados em redes *Ethernet*, através de um processo transparente realizado pelas placas de rede. Este processo de transporte depende muito das arquiteturas de rede usadas, o que influencia o tipo de revestimento ou invólucro de informação, como as células nas redes *ATM* (*Asynchronous Transfer Mode*) ou *Tokens* nas redes *Token-Ring*. Independentemente da rede que esteja a usar *TCP/IP*, os pacotes que são transportados entre os *hosts* na rede serão sempre pacotes *IP*. Os princípios básicos deste protocolo são (Loureiro, 1998); (Marques and Guedes, 1998):

- Oferecer um serviço não orientado a ligação fim-a-fim;

- Não garantir que os dados cheguem ao destino;

- Os endereços dos dispositivos de rede serem subdivididos em duas componentes: a identificação da sub-rede a que o dispositivo se encontra ligado, assim como a sua identificação na mesma rede;

- O processo de encaminhamento ser feito para cada pacote de forma independente, onde todos deverão possuir a identificação completa do destinatário;

- O encaminhamento ser orientado com base no identificador da sub-rede de destino, exceto nos dispositivos ligados de forma direta a essa sub-rede, em que o encaminhamento é realizado através do endereço do próprio dispositivo.

Um endereço *IPv4* consiste num número de 32*bits*, constituído por quatro octetos, o que em formato decimal significa: 255.255.255.255. Com o crescimento da *Internet* e dos dispositivos conectados à rede, rapidamente esta gama de endereços *IP* tornou-se insuficiente para a identificação de todos os dispositivos no universo, originando a introdução de conceito de endereços públicos e os endereços privados (Gouveia and Magalhães, 2005).

4.6.2 Protocolo *IPv6*

Tal como os demais protocolos de *Internet* que foram sujeitos a várias versões, o protocolo *IP* também sofreu atualizações, sendo o *IPv6* a versão mais recente deste protocolo que possui um mecanismo de identificação dos dispositivos em redes. Um dos objetivos primordiais do desenvolvimento do *IPv6* é o facto de este superar a escassez de endereços presentes na versão *IPv4*. O *IPv6* utiliza 128 *bits*, permitindo mais endereços, contrariamente ao *IPv4*, que utiliza 32 *bits*. Esta notação formada por quatro números separados por pontos, também conhecida por *dotted quad notation*. Atualmente, com o considerável

aumento do número de endereços é possível atribuir endereços *IP* a todos os dispositivos em rede presentes no mundo. Os endereços do *IPv6* são representados por oito grupos de quatro números hexadecimais separados por dois pontos, escritos da seguinte forma: 2001:0db8:85a3:0000:0000:0000:0000:7344. Se num determinado endereço *IPv6* existirem grupos de números formados apenas por 0000, estes podem ser omitidos, deixando somente os restantes grupos, representados por 2001:0db8:85a3::7344 (Loureiro, 1998); (Schumacher et al., 4919); (Boavida and Bernardes, 2012).

O *IPv6* não resolveu apenas a questão da escassez de endereços, também solucionou muitas questões, dispensando o uso do *NAT Network Address Translation* e introduzindo o protocolo de segurança *IPSec* como obrigatório, ao passo que no *IPv4*, o *IPSec* era opcional. Outro aspeto importante presente no *IPv6* diz respeito à não-existência de endereços *broadcast*, tendo sido substituída esta função pelos endereços *multicast*, com os seguintes modos de funcionalidades (Hinden and Deering, 1998); (Durdağı and Buldu, 2010):

- *Unicast*: Este modo tem a função de identificar apenas uma única interface de rede. Quando um determinado pacote é enviado para um endereço unicast, é entregue apenas a interface identificada por esse endereço;

- *Multicast*: Este modo tem a função de identificar um conjunto de interfaces. Se um determinado pacote for enviado para um endereço *multicast*, é entregue a todas as interfaces identificadas por esse endereço;

- *Anycast*: Este modo identifica de igual modo um conjunto de interfaces. Um pacote enviado para este tipo de endereço, é entregue à interface (mais próxima) identificada por esse endereço, de acordo com o protocolo de encaminhamento.

Estas características tornam o *IPv6* muito importante e compatível com as redes de sensores, o que fez com que o *6LoWPAN* fizesse uso do mesmo para prover a conectividade entre as redes *IoT* e *Internet* (Yibo et al., 2011).

4.6.3 O protocolo *6LoWPAN*

O *6LoWPAN*, enquanto protocolo da camada de rede pertencente as *RSSF* (Redes Sensores Sem Fio), desempenha um papel importante na integração na *Internet*, uma vez que faz uso do *IPv6* para permitir a comunicação entre as redes *IP* e as redes sensores ao nível da camada de rede, o que torna a comunicação visível e transparente, na visão dos

protocolos e aplicações que funcionam nas camadas mencionadas mais acima. Com o uso do *6LoWPAN* a comunicação ponto-a-ponto é garantida na camada de rede ou superior, entre os dispositivos sensores e os de *Internet* (Weber et al., 2016). Para que as funcionalidades sejam operacionais na íntegra e possam prover a comunicação entre as redes sensores e a *Internet*, foi necessária a introdução da camada de adaptação *6LoWPAN*. Este, por sua vez, criou uma camada de adaptação entre o *IEEE 802.15.4* e o *IPv6*, isto é, uma camada *LoWPAN*, com cabeçalhos específicos para cada uma destas funções, podendo adicionar ou remover, conforme a necessidade do cenário (Weber et al., 2016). A necessidade de compressão de cabeçalhos *IPv6* é de tamanha importância para as *RSSF*, devido a determinadas características do *IPv6* tais como: apresentação de um cabeçalho simplificado e métodos recentes de processamento de opções, deixando que o encaminhamento *Mesh*, a fragmentação e a comunicação *multicast* sejam tratadas na camada *LoWPAN*, isto é, no contexto da camada de adaptação *6LoWPAN* (Sherburne et al., 2014).

Devido a estas características, pode facilmente perceber-se que a escolha do *IPv6* para ser usado nas redes de baixo poder energético não é uma tarefa fácil, visto que o protocolo coloca uma série de exigências na otimização desta camada de adaptação. Um dos exemplos seria a não-fragmentação dos pacotes *IPv6* nas *IoT*, que seria muito grande para as redes de padrão *IEEE 802.15.4*, assim como o *overhead* que possui 40 *bytes* do cabeçalho *IPv6*, o que não seria benéfico devido à escassez de largura de banda disponível na camada física do *6LoWPAN* (Weber et al., 2016); (Sherburne et al., 2014), como por exemplo a compreensão de cabeçalhos *IPv6* é feita eliminado de forma parcial ou total os campos facilmente obtidos a partir de informação do endereçamento na camada de ligação, como é o caso de prefixo de 64 *bits* de ligação local que pode ser reduzido para 1 *bit*. Resumindo, o *6LoWPAN* tem a função de traduzir os pacotes do tipo *IPv6* para as redes *IEEE 802.15.4*, realizando algumas funções necessárias de modo a facilitar o processo de comunicação sem a necessidade de ação do utilizador e sem perda de dados na comunicação entre os dispositivos na rede, fazendo a utilização dos demais protocolos para garantir os serviços (Weber et al., 2016).

4.7 Camada de Transporte

Os protocolos *TCP* e *UDP* funcionam ao nível da camada de transporte nas redes de Internet, assim como alguns protocolos *IoT* (nomeadamente, *QUIC, DTLS,* etc.), presentes

numa única camada, isto é, a camada de apresentação, sessão e transporte conforme apresentados na Figura 3.4.

4.7.1 Os protocolos *TCP, UDP* e *QUIC*

Dentre os protocolos que fazem parte da família *TCP/IP*, é importante recordar que existe uma classificação que facilita a distinção entre os *connection-oriented communication* e os *connectionless communication*. Os protocolos orientados a conexão (*connection-oriented communication*) são aqueles que obrigam os dispositivos comunicantes numa determinada rede a estabelecerem sessão entre eles, este processo de autenticação que ocorre antes dos dados serem transmitidos por meio de detenção de erros pela verificação continua no processo de transmissão, contrariamente dos protocolos não-orientados a conexão (*connectionless communication*). Os dispositivos recetores têm a obrigatoriedade de manter informado quem transmite sobre os pacotes que vai recebendo, tornando o diálogo numa comunicação fiável entre as entidades na rede (Loureiro, 1998).

Sendo o *TCP* um protocolo *connection-oriented communication*, é penalizado pela taxa de transferência devido ao mecanismo de controlo e de retransmissão de segmentos, uma vez que o protocolo utiliza um temporizador para correção de erros e pode ter uma sobrecarga (*overhead*) dos dados a transmitir. Umas das principais funções é a de criar pacotes de informação transmitida pela camada superior, colocando dentro de pacotes a serem enviados alguma informação que facilitará no controlo de tráfego entre o emissor e o recetor. Todo este fluxo de dados é dividido por partes que criam um segmento *TCP*, que é então encapsulado por um datagrama *IP* para ser enviado pela rede (Loureiro, 1998); (Gouveia and Magalhães, 2005); (COELHO, 2017). Tal como os demais protocolos, o *TCP* suporta muitas das aplicações da *Internet*, tais como a navegação *Web*, o correio eletrónico, entre outros que fazem uso de *HTTP, SMTP* ou *FTP* e tantos outros protocolos de aplicação para prover os serviços requeridos, sendo a primeira versão *TCP* definida no RFC 793 (Comer, 1995). Algumas destas características acrescentam uma certa vantagem ao *TCP* em relação ao *UDP* através do mecanismo de retransmissão, garantido confiabilidade na entrega de informação. Estas características fazem com que o *UDP* tenha desvantagens em relação ao *TCP*, por outro lado, as desvantagens mencionadas sobre o *UDP* são as mesmas que dão vantagens devidas ao mecanismo implementado pelo protocolo, o que faz com que as aplicações multimédia deixem de parte o protocolo *TCP*, fazendo uso do *UDP*, apesar de os dois protocolos fazerem uso

de outros protocolos para garantia de alguns serviços (Kurose and Ross, 2013); (Tanenbaum, 2003).

4.7.2 Protocolo *UDP*

Ao contrário do *TCP*, o *UDP* é um protocolo não-orientado à conexão (*connectionless communication*), ou seja, não garante por si só a entrega de mensagens entre os *hosts* na rede. Em termos funcionais, os resultados que se podem obter fazendo uso do *UDP* são iguais aos resultados se utilizado o *TCP*, desde que a informação chegue corretamente ao destinatário, ou seja, ao "nó" de destino. Isto acontece não à custa do protocolo, mas sim, através do processamento feito a nível da aplicação que é utilizada e por meio de protocolos que são utilizados para garantir serviços em função do cenário em causa (Loureiro, 1998); (Comer, 1995). Sendo o *UDP* um protocolo que não oferece garantias, torna-se assim uma mais-valia na redução da latência, usada fortemente nas aplicações multimédia. Uma das diferenças apontadas entre o protocolo *UDP* e o protocolo *TCP* é a noção de sessão, que é um conceito inexistente no *UDP*, ao passo que no *TCP* a sessão é criada quando duas entidades na rede começam uma conversação, podendo esta ficar ativa até ser encerada. Estas características mencionadas do *UDP* tornam o protocolo simples em relação ao *TCP*, por dois motivos primordiais (Loureiro, 1998);:

(i) Não usa nenhuma sincronização entre a origem e destino, por não estabelecer a conexão.

(ii) O *UDP*, enquanto protocolo *connectionless communication*, não estabelece sessão entre cliente e servidor, nem os *hosts* que o utilizam são informados sobre os pacotes que vão chegando. Isto deve-se ao facto de o protocolo não possuir mecanismos de retransmissão de pacotes, que é um dos mecanismos usado pelo *TCP*;

(iii) Sendo o *UDP* um protocolo simples, possui apenas os dados, os *ports*, entre outros, diferentes do *TCP* que possui muita informação no cabeçalho dos seus pacotes.

(iv) Um pacote *UDP* pode ser fragmentado pelo protocolo *IP* para ser enviado em vários pacotes *IP* fragmentados se for necessário.

Uma das características do *UDP*, e que é bastante semelhante ao *TCP*, diz respeito à utilização dos campos, como por exemplo porta de origem e de destino, especificando que portas serão utilizadas na comunicação. O campo tamanho especifica quantos *bytes* terá um

pacote completo, ao passo que o *checksum* é opcional: faz a soma e verifica se os dados estão livres de erros (Kurose and Ross, 2013); (Gouveia and Magalhães, 2005).

Sendo um protocolo simples, as características acima mencionadas tornam-no num protocolo vulnerável aos ataques. Os invasores utilizam endereços falsos para não serem reconhecidos e utilizam portas para realizarem ataques por meio de pacotes *RIP* disfarçados de pacotes *UDP*, e portanto, não é um protocolo confiável porque não possui mecanismos de verificação e de retransmissão de pacotes, contrariamente ao protocolo *TCP* que possui estes mecanismos (Kurose and Ross, 2013); (Gouveia and Magalhães, 2005).

4.7.3 O protocolo *QUIC*

O *QUIC* é um novo protocolo de transporte desenvolvido pela *GOOGLE*, e é um protocolo rápido pois foi criado especificamente para ambientes *IoT*. Uma vez que o *UDP* é um dos protocolos de *Internet* usado nas redes *IoT* devido às características que apresenta, sendo uma delas os serviços de correção de erros menos eficientes que os do *TCP*, o *QUIC* foi criado para combinar algumas das melhores características que o *TCP* e o *UDP* apresentam, pelo facto do *QUIC* ser um protocolo de baixa latência ponto-a-ponto e acrescentar serviços de segurança equivalentes aos protocolos *TLS/SSL*, conforme o que foi dito na sessão 3.7.3. O protocolo está a ser implementado nos servidores da *GOOGLE*,assim como em navegadores *web* como o *GOOGLE Chrome* (De Coninck and Bonaventure, 2017); (Carlucci et al., 2015); (Zanella et al., 2014).

Um dos aspetos mais importantes que o protocolo *QUIC* possui é o facto de ter um processo de migração de conexão, sendo uma das qualidades que permite um dispositivo ou "nó" *QUIC* usar o mesmo identificador de conexão denominado *CID* (*Connetion Id*) mesmo quando se desloca de uma rede para outra (Zanella et al., 2014); (Fischlin and Gunther, 2017). Este procedimento é semelhante ao que acontece em redes móveis, quando um dispositivo móvel se desloca de uma área para outra usando a mesma estação base ou de uma região separada desta mesma estação base, passando de uma célula para outra na mesma portadora ou de uma célula diferente em portadoras diferentes. Este procedimento é denominado de *handover*. (Melikov, 2011); (Kurose and Ross, 2013); (Pinto, 2009).

A utilização de protocolos leves e simples para as redes *IoT* tem sido uma das prioridades da comunidade científica devido às características dos dispositivos sensores e da necessidade de integração com a *Internet*. Este facto faz com que a *GOOGLE* não trabalhe diretamente

com o *TCP* tradicional para realização de teste dos disposetivos fazendo o uso do *QUIC*, vistos que o *TCP* pode ser constituído diretamente no *Kernel* do sistema operativo, o que implicaria o não-controlo do mesmo, assim como as exigências que o protocolo apresenta devido às suas características específicas (De Coninck and Bonaventure, 2017); (Carlucci et al., 2015); (Zanella et al., 2014).

O *QUIC* é um protocolos com mais características semelhantes ao *UDP*, que foi criado especificamente para as *IoT* e que trouxe ganhos visíveis para estas redes, tal como a versão recente do *IP* (*IPv6*), assim como a vasta gama de endereços *IP* que permitiu a introdução da camada de adaptação *6LoWPAN*, o que criou uma camada de adaptação de forma torna a comunicação mais heterogénea entre as redes *IoT* e *Internet* (Weber et al., 2016); (Sherburne et al., 2014).

4.7.4 O protocolo *TLS*

O *TLS* (*Transport Layer Security*) é um protocolo de *Internet* que também é usado nas redes *IoT* para garantir aspetos de segurança devido as características que apresenta em termos de rapidez e latência quanto a encriptação. Na Figura 3.4 da arquitetura *IoT*, o protocolo esta representado nas camadas de apresentação, sessão e transporte. Como a questão da segurança estava prevista nas redes *IoT*, o *TLS* sofreu várias atualizações, apesar das primeiras versões serem usadas nas redes *TCP/IP* e da transferência de informações utilizar protocolos de transporte da *web TCP* e *UDP* para encapsular os protocolos de aplicação *HTTP* e *FTP*. Nas redes de sensores sem fios, os aspetos relacionados com a eficiência da largura de banda e com a capacidade computacional nos dispositivos que podem ser limitados, foram tidos em conta (Zanella et al., 2014); (Urien, 2013).

O protocolo *TLS*, baseado em (*Secure Sockets Layer - SSL*), é usado para garantir a segurança nos *Web-sites*. Diversos protocolos utilizam o *SSL* para cifrar as mensagens de sinalização na camada de transporte garantido a confiabilidade e a integridade. Um exemplo a mencionar é o protocolo *SIP* (*Session Initiation Protocol*) (Rosenberg and Schulzrinne, 2002). O *TLS* é composto por duas camadas: *TLS-Record* e *TLS-Handshake*. A primeira tem a função de garantir a segurança e o mecanismo de transporte, é fiável e privado. Neste mecanismo, as mensagens enviadas por esta camada são cifradas para não serem intercetadas por terceiros na conexão entre os dispositivos na rede, ao passo que o *TLS-Handshake* é a camada que permite negociar os parâmetros de segurança usados para estabelecer conexões

seguras, oferecendo métodos de autenticação (Zanella et al., 2014); (Sinnreich and Johnston, 2006); (Rosenberg and Schulzrinne, 2002).

A integridade dos dados também é um dos pontos que não foi deixado de fora, os protocolos que providenciam aspetos de segurança também foram tidos em conta, para além do *TLS*, o *DTLS* (*Datagram Transport Layer Security*) têm sido implementado nesta camada para cada um destes exercer funções específicas em função do cenário em causa (Boavida and Monteiro, 2011). Este protocolo também é utilizado em algumas aplicações multimédia apesar de este utilizar alguns mecanismos do *TCP*, ou seja, funcionar sobre o *TCP* que em muitos cenários pode acrescentar vantagens assim como desvantagens em relação ao *DTLS* pelo facto deste funcionar sobre o *UDP* (Urien, 2017).

4.7.5 O protocolo *DTLS*

O *DTLS* é um protocolo apresentado na Figura 3.4 pertencente à camada de apresentação, sessão e transporte. O *DTLS* utiliza o protocolo *TLS* para funcionar sobre o *UDP*, no entanto, o *TLS* apresenta uma semelhança em relação ao *DTLS* pelo facto destes partilharem os mesmos serviços. O *DTLS* possui uma ligeira diferença pelo facto de não apresentar mecanismos de controlo de congestionamento, o protocolo possui um ligeiro atraso por este herdar algumas características do *UDP*. Comparando com o *TLS*, o *DTLS* é vulnerável a vários ataques por meio de *IPs* falsos. Para colmatar esta situação, foi implementado um mecanismo chamado *Stateless Cookies* usado na melhoria da versão para lidar com pacotes não confiáveis, com a perda e o reordenamento de pacotes, assim como mecanismos de retransmissão na temporização, isto é, quando uma dada informação é enviada e não há retorno num determinado período de tempo, a informação é reenviada através da inclusão dos *Cookies* nas respostas geradas pelo servidor para as requisições que ele recebe (Zanella et al., 2014).

Desta forma, é possível averiguar se as mesmas mensagens têm origem nos verdadeiros clientes, evitando assim os ataques de negação de serviços em que os endereços são falsificados, enviando fluxo de dados como respostas do servidor. Estes protocolos estão a ser testados para serem implementados em ambientes *IoT* e espera-se que tenham sucesso. Assim como a transmissão de dados em redes móveis, que está a ser implementada com limitações (Zanella et al., 2014); (Urien, 2013)

4.8 Camada de Aplicação

A camada de aplicação, é também uma das importantes camadas, vistos que fornece interface de comunicação entre aplicações e serviços de rede, definindo processos de autenticação dos utilizadores nas aplicações, bem como a execução em ambientes de rede assim como a *Web*.

Este ponto descreve e estabelece a comparação a nível aplicacional entre alguns protocolos *IoT* e de *Internet*, visto que nem todos os protocolos *IoT* presentes no modelo arquitetural apresentados na Figura 3.4. têm a mesma semelhança em termos funcionais.

4.8.1 Os protocolos *HTTP*, *FTP* e o *CoAP*

O *HTTP* é um protocolo de rede da camada de aplicação desenhado especificamente para a *Internet*, para ter acesso a informações seguindo a estratégia requisição e resposta no paradigma cliente/servidor. O protocolo sofreu atualizações pela necessidade do tipo de informação que exigido entre os utilizadores na rede, tais como imagens e vídeos. Atualmente, a versão *HTTP/1.1* suporta a troca de mensagens entre os clientes e os servidores *HTTP*, definido o formato das mensagens, estas ligações que podem ser estabelecidas entre o cliente e o servidor *HTTP*, bem como as ações ou métodos a serem realizadas numa página *Web* ou num objeto genérico, nomeadamente o *GET, POST, PUT* e o *DELETE*, assim como o *LINK, UNLINK* e *HEAD*, efetuando assim a leitura dos cabeçalhos de uma página, a ligação entre recursos existentes assim como a quebra de ligação entre estes mesmos recursos (Boavida and Bernardes, 2012); (Kurose and Ross, 2013).

Nestas ligações não-persistentes entre o cliente e o servidor *HTTP* que são feitas numa determinada conexão ou ligação adequada para a transferência de informação, é necessário que o *browser* estabeleça uma sessão *TCP* com o servidor, enviando um pedido e a confirmação da ação para o terminar da ligação. No caso de utilizar uma ligação persistente, o servidor não termina a ligação *TCP* com o cliente, e neste caso, o cliente aguarda novos pedidos, que serão transmitidos na mesma ligação estabelecida anteriormente (Boavida and Bernardes, 2012); (Kurose and Ross, 2013).

Devido ao facto de o *HTTP* fazer a utilização do *TCP* que é um dos mais importantes protocolos da *Internet* e pelas características que o protocolo *TCP* apresenta, o *HTTP* é um protocolo de alto desempenho, por se tratar de um protocolo de pedido e resposta,

não se preocupando com a manutenção do estado dos clientes e efetuando várias trocas de mensagens (Boavida and Bernardes, 2012). Estas é uma das características que tornam o *HTTP* complexo e incompatível com ambientes *IoT*, daí que a comunidade científica se preocupe com o desenvolvimento de um novo protocolo *web* para ambientes *IoT* devido às características que os dispositivos sensores apresentam (Al-Sarawi et al., 2017).

4.8.2 O protocolo *FTP*

O *FTP* é um protocolo *Web* para transferência de ficheiros utilizando o *browser WWW* para ter acesso a servidores de ficheiros, o *FTP* é um protocolo baseado na aqruitetura cliente/servidor, utilizando as portas 20 e 21 para transferência de dados e controlo de informação entre os *hosts* na rede. Para efeitos de transferência de ficheiros de forma concorrente do servidor é estabelecida uma nova ligação de dados, fazendo com que o *FTP* utilize o *TCP* devido ao mecanismo de retransmissão para garantir a entrega confiável dos dados entre os clientes na rede, que é uma das características que torna o *FTP* mais complexo em cenários que seja necessário reduzir a quantidade de código e o consumo de recursos como a memória. Por exemplo, sistemas embebidos e sistemas sem disco (Boavida and Bernardes, 2012); (Kurose and Ross, 2013).

O *TFTP* (*Trivial File Transfer Protocol*) é a versão simplificada do *FTP*, surge para colmatar com estas lacunas, apresentando funcionalidades bastante limitadas em apenas quatro mensagens protocolares, nomeadamente o *read request, write request, data* e *acknowledgment*, utilizando sistemas de detenção e recuperação de erros nas mensagens de *acknowledgment* associadas ao mecanismo de retransmissão com temporizador *timeout* com funcionalidades arcaicas, sem comandos de consulta do diretório no servidor, obrigando o cliente a ter o conhecimento prévio do nome do ficheiro que pretende transferir (Boavida and Bernardes, 2012); (Kurose and Ross, 2013).

Uma das diferenças existentes entre o *FTP* e *HTTP* diz respeito ao facto de o armazenamento do ficheiro copiado realizar de forma automática ou movido a partir de um servidor de arquivo para dispositivo de armazenamento do computador local, e vice-versa, basicamente faze utilização do *upload* e *download* (Boavida and Bernardes, 2012). Esta é uma das características particular que faz com que o *FTP* não seja utilizado em ambientes *IoT*.

4.8.3 O protocolo *CoAP*

Tal como o *HTTP*, o *CoAP* é um protocolo da camada de aplicação, desenvolvido pelo *Constrained RESTful Environment* (*CoRE*) do *IETF*, orientado especificamente para ambientes *IoT*. Basicamente, o *CoAP* possui uma abordagem em termos funcionais semelhante à do *HTTP* que é um protocolo aplicacional para a transferência dos recursos da *Web*, visto que o protocolo utiliza uma arquitetura denominada *REST*, mapeado pelo *HTTP* de forma homogénea. Sendo o *CoAP* um protocolo para dispositivos que funcionam em ambientes com restrições energéticas, o mesmo tem como objetivo a redução do *overhead* original do *HTTP* através da compressão do cabeçalho de 127 *bytes* para apenas 4 *bytes*. Tal como o *HTTP*, o *CoAP* faz a utilização de quatros métodos tradicionais utilizados pelo *HTTP*, nomeadamente o *GET, PUT, POST* e *DELETE*. Este protocolo adota o modelo de requisição/resposta e a comunicação é feita de forma assíncrona, tendo quatro tipos de mensagens: a mensagem de confirmação (*CON – Confirmable*), a mensagem de não-confirmação (*NON – Non-Confirmable*), (*ACK – Acknowledgement*) e a mensagem de *RESET* (*NON – Non-Confirmable*) (Agrawal and Das, 2011); (Rahman and Shah, 2016); (Soldatos et al., 2012).

Desta forma, tal como o *HTTP* faz a utilização de alguns protocolos, o *CoAP* faz a utilização do *UDP*, cujas características são compatíveis com as redes *IoT*, nomeadamente o não-estabelecimento de conexão, a não-garantia na entrega de pacotes, entre outros. Uma das vantagens do *CoAP* consiste na capacidade da descoberta de recursos, isto é, o servidor fornece uma lista permitindo que os clientes tenham o conhecimento dos tipos de recursos a serem disponibilizados. O esquema *URI* é utilizado para identificar os recursos na rede, e é algo como: coap://site (Zanella et al., 2014);(Karagiannis et al., 2015); (Rahman and Shah, 2016). Relativamente aos aspetos de segurança, o *CoAP* utiliza o protocolo *DTLS* que é baseado no *TLS* sobre o *UDP*, em vez do *TCP* usado pelo *HTTP*. Comparativamente ao *HTTP*, o *CoAP* é mais eficiente em termos de custos porque realiza menor troca de mensagens entre o cliente e o servidor, o que leva a um menor consumo de potência em ambos os lados da conexão (Zanella et al., 2014);(Karagiannis et al., 2015); (Rahman and Shah, 2016).

4.8.4 O protocolo *SMTP* e o *XMPP*

O *SMTP* é um protocolo da camada aplicacional desenhado especificamente para transferir mensagens entre servidores de correio eletrónico, tendo uma filosofia funcional semelhante ao *HTTP* que opera no modelo cliente/servidor e fazendo uso de uma ligação persistente para transferir mensagens entre um cliente e um servidor. O *SMTP* apoia-se no *TCP* como protocolo de transporte, permitindo que os servidores de correio eletrónico que implementam o *SMTP* aceitem os pedidos de ligação *TCP* (Loureiro, 1998); (Boavida and Bernardes, 2012). Um dos aspetos que diferencia o *SMTP* do *XMPP* é falta de capacidade do primeiro protocolo em realizar o transporte de mensagens multimédia que incluam imagens, vídeos ou sons, visto que o mesmo tinha sido desenvolvido para a transmissão de mensagens de texto bastante simples e pequenas; por outro lado, os circuitos de transmissão tinham uma capacidade muito reduzida no período de desenvolvimento do protocolo. Este facto permitiu que as mensagens que eram transportadas pelo *SMTP* apenas possuíssem caracteres *ASCII* com 7 *bits* apenas. Para realizar o transporte de mensagens multimédia ou de caracteres *ASCII* com 7 *bits*, isto é, 8 *bits* em diante, era necessário uma série de mecanismos como a codificação de mensagens para *ASCII* com os primeiros 7 *bits* antes de efetuar a transmissão e em seguida a codificação logo após a receção (Boavida and Bernardes, 2012); (Kurose and Ross, 2013).

Ao passo que o *XMPP* foi projetado inicialmente para mensagens instantâneas, assim como para a troca de mensagens entre aplicações. Este protocolo de aplicação de padrão aberto foi desenvolvido por um grupo de pesquisadores do IETF (*Internet Engineering Task Force*). Dada a sua eficiência, o protocolo tornou-se bastante conhecido na Internet, funcionando em muitas aplicações para fazer chamadas de voz ou vídeo e teleconferências entre as partes envolvidas na comunicação. As aplicações comerciais fazem uso do *XMPP*, incluindo o *chat*, entre outros. Por outro lado, permite o transporte confiável de todos os dados *XML* estruturados entre indivíduos ou aplicações como chamadas *RPC* e *SOAP*, o que faz com que o protocolo se torne mais extensível. Tendo em consideração as suas características, ele oferece recursos de interoperabilidade, tais como a ligação *HTTP*, descobertas de serviço, transferência de arquivos, entre outros. Atualmente, espera-se que com as suas exigências o mesmo possa oferecer serviços criptográficos, alta privacidade e compatibilidade com outros protocolos em tempo real (Ramirez and Pedraza, 2017); (Xuefu and Ming, 2012); (Wang et al., 2017).

Sendo um protocolo projetado para aplicações em tempo real, atualmente é utilizado para aplicações IoT, visto que o XMPP apenas suporta mensagens curtas e de baixa latências, assim como as arquiteturas *publisher/subscriber* e *request/response*, em que o elemento principal para *publisher/subscriber* é o servidor, os editores enviam dados e os assinantes recebem a publicação de dados que lhes são enviados (Rahman and Shah, 2016); (Wang et al., 2017).

Este protocolo oferece uma série de vantagens como a flexibilidade, a extensibilidade, entre outros requisitos que tornam o protocolo cada vez mais importante para as *IoT*, assim como para a *Internet* por se adaptar ao padrão de comunicação (Soldatos et al., 2012). Os fundadores deste padrão protocolar desenvolveram uma série de extensões protocolares com base nas especificações do núcleo. Em relação à flexibilidade, o XMPP baseia-se no XML, que é uma linguagem de auto-descrição que fornece formulários de dados independentes de linguagem e plataforma. Neste sentido, existe uma abertura para as organizações definirem a sua própria linguagem em função das necessidades. O potencial que o XMPP herdou do núcleo protocolar XML torna o XMPP num protocolo robusto, que permite a troca de informação entre as redes heterogéneas e que tem algumas características como (Wang et al., 2017):

- **Extensível**: Usando as funcionalidades do XMPP, é possível construir funcionalidades personalizadas por cima do núcleo do protocolo, mantendo deste modo a interoperabilidade. As extensões são publicadas como XEP, sendo que algumas das principais extensões podem incluir o *Multi User Chat* (XEP-0045) para criação de grupos de conversação com múltiplos utilizadores implementado o modelo *publisher/subscriber*;

- **É seguro**: Porque qualquer servidor *XMPP* pode ser isolado a partir da rede pública, em redes empresariais, por exemplo, pode utilizar-se o *TLS* para criptografia de canais e *SSL* para autenticação;

- **Flexível**: Pela sua flexibilidade é possível construir várias aplicações utilizando recursos *XMPP*, nomeadamente ferramentas para gestão de rede, jogos, partilha de arquivos, gestão remota de sistemas, computação em nuvem, serviços *Web*, entre outros.

Estas características tornam o *XMPP* diferente do *SMTP*, visto que os protocolos apresentam conceitos e abordagens diferentes. O *SMTP* faz o uso dos protocolos *POP3* e *IMAP* para serviços e comprimento de certas tarefas, tais como o *download* de *e-mails* da máquina do cliente. Em suma, cada um destes protocolos é um conjunto específico de regras que se estabelece no processo de comunicação. O protocolo funciona em dois modo *online* e *Offline*, o modo online permite a transferência do *e-mail* da máquina do cliente, tão logo

que termine a transferência ficheiro o cliente pode visualizar as transferências feitas estando em modo *offline*. Ao passo que para o protocolo *IMAP* (*Internet Message Access Protocol*), utilizado para receção do *e-mail*, o principal objetivo é manter as mensagens no servidor para serem visualizadas em diferentes dispositivos. Com o *IMAP* também é possível manter em modo *offline* e fazer sincronização periódica, disponibilizando determinados serviços como (Corey et al., 2002); (Boavida and Bernardes, 2012).

- Criar, renomear e apagar pastas contidas no correio eletrónico;

- Observar novas mensagens;

- Remover mensagens definitivamente.

4.8.5 O protocolo *MQTT*

Um dos aspetos que poderia comparar o *MQTT* ao *XMPP* seria o facto de a interação entre as entidades na rede serem por meio de padrão *publisher/subscriber*, entre o cliente e o servidor. Tal como o *XMPP*, o *MQTT* também é um protocolo de troca de mensagens. Um dos principais objetivos do desenvolvimento do protocolo é a baixa largura de banda e a utilização do mínimo de recursos de *hardware*, garantido a confiabilidade na entrega de dados, tendo um baixo *overhead* em torno de 2 *bytes*, conectados ao servidor ou *broker* utilizando o protocolo *TCP* para garantir o serviço necessário. Os dispositivos de rede enviam vários tópicos para *broker* que é um serviço responsável que faz a gestão das publicações e as subscrições do protocolo *MQTT*. Tão logo que este serviço ou aplicação começa a funcionar, os clientes subscrevem os tópicos que necessitam, podendo também subscrever vários tópicos recebendo deste modo mensagens de outros clientes da rede que publicam neste tópico(COELHO, 2017); (Xiong and Fu, 2011).

O *MQTT* funciona na camada de integração (*middleware*) das aplicações ou entre as camadas aplicacionais. Um dos exemplos a mencionar são as várias implementações de softwares que fazem uso do *MQTT*, nomeadamente o *Mosquito*, *Facebook Apps* (*Messenger*), *Eclipse Paho RabbitMQ*, pelo facto de o protocolo não consumir muita bateria dos dispositivos assim como da largura de banda. Com o uso deste protocolo, a capacidade de entrega das mensagens de telefone para telefone diminuiu o atraso para centenas de milissegundos em vez de vários segundos (COELHO, 2017); (Xiong and Fu, 2011).

Devido a estas características, o protocolo é suficientemente utilizado em diversas

empresas, nomeadamente a *Cisco*, a *Credit Suisse*, o Banco Americano, entre outros, facilitando o envio e a receção das mensagens. Um dos aspetos em que difere, comparando com os demais protocolos, é o facto do *MQTT* especificar as mensagens que recebe e as que envia, implementando nestes moldes a segurança, o desempenho e confiabilidade (Xiong and Fu, 2011); (Wang et al., 2017).

Sendo o *MQTT* protocolo flexível, que também é uma dos aspetos importantes, que lhe permite suportar diversos cenários relativamente à baixa e alta latência e baixa largura de banda, tornando-o assim um protocolo compatível com os equipamentos de baixo poder energético. O *MQTT* possui duas características básicas: o *messsage Broker* e vários clientes. O *Broker* é uma aplicação servidor que tem a função de receber todas as mensagens dos *hosts* e enviá-las para os *hosts* de destino. O cliente realiza uma assinatura que é o tópico de mensagens e o servidor recebe estas mensagens que posteriormente são enviadas. Este é um dos aspetos que torna o *MQTT* contrário ao *XMPP*, o facto de este suportar mensagens curtas e baixas latências, assim como as arquiteturas *publisher/subscriber* e aspetos de segurança (Xiong and Fu, 2011).

Em virtude de determinados protocolos não resolverem todos os problemas tal como é esperado, devido às características dos dispositivos e das de mensagens que podem circular entre os dispositivos na rede, várias versões protocolares vão nascendo para colmatar determinados problemas não resolvidos por um determinado protocolo. O *MQTT-SN* (*Sensor Network*) é a versão protocolar do *MQTT* que não faz a utilização do *TCP* para a realização do transporte de informação entre os dispositivos sensores na rede, visto que o *TCP* é um protocolo pesado e pelas suas características acrescenta uma série de restrições. Quando comparado com o *MQTT*, podemos encontrar as seguintes diferenças (Xiong and Fu, 2011); (Wang et al., 2017):

- O *MQTT* possui uma comunicação *TCP*, ao passo que o *MQTT-SN* faz a utilização do protocolo *UDP*;

- O *MQTT* funciona em redes *Ethernet, Wi-Fi* e 3G, ao passo que o *MQTT-SN* funciona em redes *Zigbee, Bluetooth*, entre outras;

- O tamanho mínimo das mensagens no *MQTT* é de 2 *bytes* e máximo de 24 MB, ao passo que no *MQTT-SN* o tamanho mínimo é de 1 *byte* e máximo de 128 *bytes*, onde os clientes podem estar em *connection mode* e *connectionless mode*.

Capítulo 5

Conclusões e Trabalho Futuro

Com o desenvolvimento das redes de sensores, possíveis áreas e cenários de aplicação surgem cada vez mais. Devido a divergências nas tecnologias desenvolvidas de forma específica pelos respetivos fabricantes, a interoperabilidade entre diferentes tecnologias e dispositivos não resolve todos os problemas tal como esperado. Nesta vertente, a criação de protocolos *standard Internet* em ambientes *IoT* começa a desempenhar um papel fundamental, fazendo com que as redes de sensores deixassem de ser uma abordagem isolada.

Através do estudo descrito sobre o modelo *OSI* e arquitetura *TCP/IP* observou-se que o modelo define os principais conceitos e divide a tarefa de comunicação em sete camadas funcionais, especificando as funções a serem desempenhadas por cada uma delas. Este conceito também é extensivo para todos os modelos *IoT*, adequando-se em função das camadas que cada modelo apresenta, apesar das RSSF apresentar o aspeto sensorial como parte inovadora incorporada nos dispositivos físicos.

Estabeleceu-se uma relação entre o modelo *OSI* e *TCP/IP* para perceber a correspondência existente entre as camadas, onde é visível que o modelo apresentando não possui o mesmo nível de camadas arquiteturais, o mesmo acontece em alguns modelos *IoT*, que também apresentam claras evidências face aos níveis por camadas nos diferentes modelos aquiteturais apresentados por vários autores. Diferentemente dos modelos arquiteturais, as arquiteturas protocolares de *Internet* e *IoT* possuem um conjunto de protocolos orientados para necessidades específicas e suportados por dispositivos existentes, sendo estas uma das condições primordiais para que haja comunicação entre os dispositivos na rede, facilitando a adesão por parte dos utilizadores.

Através das reflexões feitas, percebeu-se que o desenho dos modelos arquiteturais e

arquiteturas protocolares *IoT*, surge em função das caraterísticas que os disposetivos sensores apresentam, o que originou, por outro lado, a criação de novos protocolos.

Na análise comparativa das funções protocolares ao nível das camadas entre as arquiteturas *Internet* e *IoT*, nota-se alguns aspetos convergentes nos protocolos desenhados com a mesma finalidade. O protocolo *HTTP* e o *CoAP*, são dois protocolos da camada de aplicação de *Internet* e *IoT* que apresentam caraterísticas semelhantes relativamente aos mecanismos de funcionamento. A transferência dos recursos da *Web* é uma das principais finalidades destes protocolos. O *CoAP* faz a utilização de quatro métodos tradicionais utilizados pelo *HTTP*, nomeadamente o *GET, PUT, POST* e *DELETE*, respetivamente, a comunicação é feita de forma assíncrona, assim como a utilização do esquema *URI* para identificar recursos na rede.

Uma das limitações é a não utilização de aplicações no trabalho para simulação de cenários em redes *IoT* para análise dos protocolos, observando a convergência entre os protocolos de *Internet* e *IoT*, adicionando mais o sustento dos pontos de vista e realidade baseada nos testes observados por vários autores, consoante descrito na literatura.

Como trabalho futuro, propõe-se: (i) sustentar as comparações realizadas, fazendo uso das aplicações para analisar diferentes cenários em ambientes *IoT*, implementado mecanismos para análise comportamental dos protocolos nas arquiteturas a serem usadas, em função dos cenários; (ii) avaliar aspetos de segurança, eficiência energética, entre outros; (iii) enriquecer a literatura por meio de pesquisas científicas e testes para execução em laboratório.

BIBLIOGRAFIA

Agrawal, S. and Das, M. L. (2011). Internet of things a paradigm shift of future internet applications. *Engineering (NUiCONE), 2011 Nirma University International Conference on*, pages 1–7.

Akbar, M. S., Yu, H., and Cang, S. (2017). Tmp: Tele-medicine protocol for slotted 802.15. 4 with duty-cycle optimization in wireless body area sensor networks. *IEEE Sensors Journal*, 17(6):1925–1936.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376.

Al-Sarawi, S., Anbar, M., Alieyan, K., and Alzubaidi, M. (2017). Internet of things (iot) communication protocols. *Information Technology (ICIT), 2017 8th International Conference on*, pages 685–690.

Antunes, L. P. d. C. (2012). Identificação de pessoas numa portaria virtual.

Boavida, F. and Bernardes, M. (2012). *TCP/IP–Teoria e Pratica*. FCA-Editora de Informatica Lda, Lisbon.

Boavida, F. and Monteiro, E. (2011). *Engenharia de Redes Informáticas*. FCA-Editora de Informática Lda, Lisbon.

Carlucci, G., De Cicco, L., and Mascolo, S. (2015). Http over udp: an experimental investigation of quic. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 609–614. ACM.

Chandan, A. R. and Khairnar, V. D. (2018). Bluetooth low energy (ble) crackdown using iot. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 1436–1441. IEEE.

Chauvenet, C., Etheve, G., Sedjai, M., and Sharma, M. (2017). G3-plc based iot sensor networks for smartgrid. *Power Line Communications and its Applications (ISPLC), 2017 IEEE International Symposium on*, pages 1–6.

COELHO, P. (2017). Internet das coisas: Introdução prática. *Lisboa: FCA*, pages 158–178.

Comer, D. E. (1995). Principles, protocols, and architecture. *Internetworking with tcp/ip*.

Corey, V., Peterman, C., Shearin, S., Greenberg, M. S., and Van Bokkelen, J. (2002). Network forensics analysis. *IEEE Internet Computing*, 6(6):60–66.

De Coninck, Q. and Bonaventure, O. (2017). Multipath quic: Design and evaluation. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 160–166. ACM.

De Oliveira, D. N., Vasques, T. L., Vieira, F. H. T., de Deus Junior, G. A., de Castro, M. S., de Araujo, S. G., de Souza, E. M., Vieira, J. G., de Oliveira Junior, O. C., Borges, A., et al. (2010). A management system for plc networks using snmp protocol. *Power Line Communications and Its Applications (ISPLC), 2010 IEEE International Symposium on*, pages 78–83.

Domingo, M. C. (2012). An overview of the internet of things for people with disabilities. *Journal of Network and Computer Applications*, 35(2):584–596.

Durdağı, E. and Buldu, A. (2010). Ipv4/ipv6 security and threat comparisons. *Procedia-Social and Behavioral Sciences*, 2(2):5285–5291.

Fischlin, M. and Gunther, F. (2017). Replay attacks on zero round-trip time: The case of the tls 1.3 handshake candidates. In *Security and Privacy (EuroS and P), 2017 IEEE European Symposium on*, pages 60–75. IEEE, IEEE.

Fouladi, B. and Ghanoun, S. (2013). Security evaluation of the z-wave wireless protocol. *Black hat USA*, 24:1–2.

Gaddour, O. and Koubâa, A. (2012). Rpl in a nutshell: A survey. *Computer Networks*, 56(14):3163–3178.

Gomez, C. and Paradells, J. (2010). Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6):92–101.

Gouveia, J. and Magalhães, A. (2005). *Redes de Computadores*. FCA-Editora de Informática Lda, Lisbon.

Guo, Z., Harris, I. G., Tsaur, L.-f., and Chen, X. (2015). An on-demand scatternet formation and multi-hop routing protocol for ble-based wireless sensor networks. *Wireless Communications and Networking Conference (WCNC), 2015 IEEE*, pages 1590–1595.

Guth, J., Breitenbücher, U., Falkenthal, M., Leymann, F., and Reinfurt, L. (2016). Comparison of iot platform architectures: A field study based on a reference architecture. In *2016 Cloudification of the Internet of Things (CIoT)*, pages 1–6. IEEE.

Hinden, R. M. and Deering, S. E. (1998). An ipv6 aggregatable global unicast address format.

Houda, M., Zarai, F., Obaidat, M. S., and Kamoun, L. (2014). Optimizing handover decision and target selection in lte-a network-based on mih protocol. pages 299–304.

Izaguirre, M. J. A. G., Lobov, A., and Lastra, J. L. M. (2011). Opc-ua and dpws interoperability for factory floor monitoring using complex event processing. pages 205–211.

Jia, X., Feng, Q., Fan, T., and Lei, Q. (2012). Rfid technology and its applications in internet of things (iot). *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 1282–1285.

Karagiannis, V., Chatzimisios, P., Vazquez-Gallego, F., and Alonso-Zarate, J. (2015). A survey on application layer protocols for the internet of things. *Transaction on IoT and Cloud Computing*, 3(1):11–17.

Khan, R., Khan, S. U., Zaheer, R., and Khan, S. (2012). Future internet: the internet of things architecture, possible applications and key challenges. *Frontiers of Information Technology (FIT), 2012 10th International Conference on*, pages 257–260.

Ko, J., Terzis, A., Dawson-Haggerty, S., Culler, D. E., Hui, J. W., and Levis, P. (2011). Connecting low-power and lossy networks to the internet. *IEEE Communications Magazine*, 49(4):96–101.

Kurose, J. F. and Ross, K. W. (2013). *Redes de Computadores e a Internet*. Pearson Education - Brasil, 6 edition.

Loureiro, P. (1998). *TCP-IP em redes Microsoft para profissionais*. FCA Online Library.

Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., and Manzoni, P. (2015). A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, pages 931–936.

Marques, J. A. and Guedes, P. (1998). *Tecnologia de Sistemas Distribuídos*. FCA-Editora de Informática Lda, Lisbon.

Melikov, A. (2011). *Cellular Networks: Positioning, Performance Analysis, Reliability*. BoD–Books on Demand.

Mercaldi, M., D'Oria, A., Murru, D., Liang, H.-N., Man, K. L., Lim, E. G., Hahanov, V., and Alexander, M. (2013). Internet of things: A practical implementation based on a wireless sensor network approach. pages 1–4. IEEE.

Neves, P. and Rodrigues, J. (2009). Motivaccao para utilização de ip em redes sensores sem fio. *9ª Conferencia Sobre Redes de Computador, Oeitas - Portugal, Covilhã*, 15(6):1–6.

Pawlowski, M. P., Jara, A. J., and Ogorzalek, M. J. (2014). Extending extensible authentication protocol over ieee 802.15. 4 networks. pages 340–345.

Pinto, S. (2009). *Redes Celulares*. FCA-Editora de Informática Lda, Lisbon.

Rahman, R. A. and Shah, B. (2016). Security analysis of iot protocols: A focus in coap. pages 1–7. IEEE.

Ramirez, J. and Pedraza, C. (2017). Performance analysis of communication protocols for internet of things platforms. *Communications and Computing (COLCOM), 2017 IEEE Colombian Conference on*, pages 1–7.

Rosenberg, J. and Schulzrinne, H. (2002). Session initiation protocol (sip): locating sip servers.

Said, O. and Masud, M. (2013). Towards internet of things: Survey and future vision. *International Journal of Computer Networks*, 5(1):1–17.

Samaras, I. K., Hassapis, G. D., and Gialelis, J. V. (2013). A modified dpws protocol stack for 6 lowpan-based wireless sensor networks. *IEEE Transactions on Industrial Informatics*, 9(1):209–217.

Schumacher, C. P. P., Kushalnagar, N., and Montenegro, G. (2007.rfc4919.). Ipv6 over low-power wireless personal area networks (6lowpans): overview, assumptions, problem statement, and goals.

Seeley, R. R. (2005). Anatomia e fisiologia. pages 384–439.

Severi, S., Sottile, F., Abreu, G., Pastrone, C., Spirito, M., and Berens, F. (2014). M2m technologies: Enablers for a pervasive internet of things. *Networks and Communications (EuCNC), 2014 European Conference on*, pages 1–5.

Sherburne, M., Marchany, R., and Tront, J. (2014). Implementing moving target ipv6 defense to secure 6lowpan in the internet of things and smart grid. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pages 37–40. ACM.

Sinnreich, H. and Johnston, A. (2006). Sip, p2p, and internet communications. *draft-johnston-sipping-p2p-ipcom-02 (work in progress)*.

Soldatos, J., Serrano, M., and Hauswirth, M. (2012). Convergence of utility computing with the internet-of-things. pages 874–879. IEEE.

Stankovic, J. A. (2008). When sensor and actuator networks cover the world. *ETRI journal*, 30(5):627–633.

Tanenbaum, A. S. (2003). Redes de computadores quarta edição. *Editora Campus*.

Trelsmo, P., Di Marco, P., Skillermark, P., Chirikov, R., and Ostman, J. (2017). Evaluating ipv6 connectivity for ieee 802.15. 4 and bluetooth low energy. *Wireless Communications and Networking Conference Workshops (WCNCW), 2017 IEEE*, pages 1–6.

Urien, P. (2013). Llcps: A new security framework based on tls for nfc p2p applications in the internet of things. pages 845–846.

Urien, P. (2017). Introducing tls/dtls secure access modules for iot frameworks: Concepts and experiments. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 220–227. IEEE.

Wang, H., Xiong, D., Wang, P., and Liu, Y. (2017). A lightweight xmpp publish/subscribe scheme for resource-constrained iot devices. *IEEE Access*, 5:16393–16405.

Weber, P., Jackle, D., Rahusen, D., and Sikora, A. (2016). Ipv6 over lorawan. In *Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), 2016 3rd International Symposium on*, pages 75–79. IEEE.

Wu, L., Hai-xin, D., Tao, L., Xing, L., and Jian-ping, W. (2009). H6proxy: Icmpv6 weakness analysis and implementation of ipv6 attacking test proxy. pages 519–524.

Xiong, X. and Fu, J. (2011). Active status certificate publish and subscribe based on amqp. pages 725–728. IEEE.

Xuefu, B. and Ming, Y. (2012). Design and implementation of web instant message system based on xmpp. *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, pages 83–88.

Yassein, M. B., Mardini, W., and Khalil, A. (2016). Smart homes automation using z-wave protocol. *Engineering and MIS (ICEMIS), International Conference on*, pages 1–6.

Yibo, C., Hou, K.-M., Zhou, H., Shi, H.-l., Liu, X., Diao, X., Ding, H., Li, J.-J., and De Vaulx, C. (2011). 6lowpan stacks: A survey. pages 1–4. IEEE.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32.

Zeltserman, D. and Zeltserman, D. (1999). *A practical guide to SNMPv3 and network management*. Prentice Hall Upper Saddle River, NJ.