INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Descoberta de Objetos Inteligentes em dispositivos móveis

PEDRO DINIS RICARDO
Outubro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

DESCOBERTA DE OBJETOS INTELIGENTES EM DISPOSITIVOS MÓVEIS

Pedro Dinis Ricardo

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Computacionais

Orientador: Nuno Alexandre Magalhães Pereira

Porto, Outubro 2016

Resumo

Objetos inteligentes são objetos normais do quotidiano que têm a particularidade de estarem equipados com componentes de *hardware* e *software*. Estes componentes podem ser processadores para processamento de tarefas, sensores para obtenção de dados do mundo que os rodeia e rádios para comunicação. Os objetos inteligentes têm como principal objetivo comunicar e estabelecer conexões com outros dispositivos, partilhando informação acerca deles mesmos. É por isso necessário que os seus serviços estejam disponíveis de alguma forma com vista a poderem ser acedidos e utilizados. No entanto, nem sempre é tarefa fácil descobri-los, saber que objetos estão disponíveis e que serviços têm à disposição. Dessa forma, torna-se essencial que existam mecanismos que consigam garantir a descoberta ao utilizador sem que este tenha qualquer ação perante isso.

O problema que irá ser apresentado neste projeto centra-se na forma como é possível descobrir os serviços destes objetos inteligentes no ambiente que nos rodeia através de um dispositivo móvel.

O objetivo deste trabalho passa por estudar e avaliar, tendo em conta aspetos relevantes no contexto dos objetos inteligentes, alguns dos protocolos existentes para a descoberta de serviços numa rede. Após esse estudo e respetiva avaliação serão implementados os protocolos que mais vantagens apresentem tendo em conta diversos fatores. As implementações serão a base para uma simulação que terá como objetivo a obtenção de métricas, tais como o gasto energético e o tempo. Esses dados ajudarão a formar conclusões sobre que protocolo é mais simples e eficaz no processo de descoberta de serviços, dando a melhor resposta ao problema apresentado.

Palavras-chave: Objetos inteligentes, dispositivos móveis, protocolos, serviços, sensores, rede, simulação

Abstract

Smart objects are normal objects of the quotidian that have the particularity of being equipped with components of hardware and software. These components can be processors for processing tasks, sensors to collect data of the world around them and radios to communicate. The main goal of the smart objects is to communicate and establish connections with other devices, sharing information about themselves. That´s why is highly required that their own services are available to be accessed and used in some way. However, it´s not always an easy task to find them, knowing which objects are available and which services are accessible. Thus, it's essential that mechanisms exist that can guarantee the discovery to the user without him having any kind of effort.

The problem that will be presented in this project focuses in finding the services of these smart objects in the environment around us through a mobile device.

That said this work will study and evaluate, considering the relevant aspects in the environment of the smart objects, some of the existent protocols that can find services in a network. After these study and evaluation, the protocols that present more advantages, considering different factors, will be implemented. These implementations will be the basis to a simulation that will aim to obtain data such as energy waste and time. Such values will help forming conclusions about which protocol is more simple and effective in the process of find the services, giving the best answer to the problem presented.

Keywords: Smart objects, mobile devices, protocols, services, sensors, network, simulation

Agradecimentos

Um agradecimento especial ao meu orientador Prof. Nuno Pereira pela ideia inicial do projeto e por toda a disponibilidade e paciência que demonstrou ter para que este trabalho fosse possível. Apesar dos vários obstáculos, conseguiu sempre encaminhar o trabalho para novas ideias e soluções de forma simples contribuindo para uma motivação adicional em momentos de problemáticos.

Adicionalmente, um agradecimento à minha família por todo o apoio que mostrou assim como à minha namorada, pela paciência e ajuda no que à elaboração deste trabalho diz respeito.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

Lista de Acrónimos

API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
IoT	*Internet of Things*
IP	*Internet Protocol*
LAN	*Local Area Network*
RAM	*Random Access Memory*
ROM	*Read-only Memory*
SOAP	*Simple Object Access Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
URL	*Uniform Resource Locator*
WLAN	*Wireless Local Area Network*
XML	*eXtensible Markup Language*

1 Introdução

1.1 Contexto

O tema desta dissertação insere-se no contexto de Computação Ubíqua, mais propriamente numa das suas áreas, a Internet das Coisas (*Internet of Things*).

Apesar de este termo já existir no mercado tecnológico há mais de uma década, é nestes últimos anos que temos vindo a observar um crescimento brutal ao nível dos dispositivos e respetivo *software.* A Internet das Coisas assenta, no fundo, no tornar inteligente e autónomo muitos dos objetos que são já habituais no nosso quotidiano, adicionando-lhe para isso componentes de *hardware,* tais como processadores, sensores, atuadores, comunicações sem fios, entre outros. Pretende-se sobretudo que estes nos exibam informações acerca do seu funcionamento e que sejam controláveis remotamente. Para isso é necessário que estes estabeleçam conexões com outros dispositivos. Os *smartphones* têm tido um papel preponderante nessa comunicação, seja através da instalação de aplicações dos fabricantes, seja através de funcionalidades existentes nos respetivos sistemas operativos. No entanto, nem sempre é trivial saber se existem objetos inteligentes à nossa volta e de que tipo são. Ainda mais difícil se torna se cada fabricante implementar a sua forma de comunicação e utilização dos serviços. Empresas como a Google têm sido impulsionadoras nesta área, desenvolvendo não só sistemas operativos abertos para o efeito, mas também formas de comunicação que já vem integradas nos seus sistemas operativos móveis, como é o caso do Android.

Existem atualmente milhões de dispositivos conectados à Internet. Prevê-se que este número suba exponencialmente nos próximos anos tendo em conta o crescimento dos chamados *wearables*, veículos inteligentes e sem condutor, sensores inteligentes, beacons e outros demais que irão aparecer.

1.2 Problema

O problema que irá ser abordado e ao qual se pretende apresentar respostas terá como base a descoberta de objetos inteligentes através de um dispositivo móvel (*smartphone*).

Tal como foi dito no ponto anterior, atualmente os *smartphones* têm tido um papel decisivo na descoberta e comunicação com objetos inteligentes. Porém, maior parte das vezes não sabemos o que existe à nossa volta, nomeadamente que serviços estão disponíveis para serem utilizados.

Existem diversos protocolos que permitem tanto aos objetos inteligentes publicar os seus serviços numa rede, como a outros dispositivos descobrir esses mesmos serviços. Esses protocolos deverão ser simples, eficazes e ter em atenção a mínima interação com o utilizador, uma vez que estamos a lidar com dispositivos que muitas vezes não apresentam qualquer interface disponível para qualquer tipo de configuração.

Em suma, a comunicação dos serviços que cada objeto inteligente oferece numa rede de forma a serem acessíveis através de um *smartphone* é o principal problema.

1.3 Motivação

Com o número cada vez maior de objetos inteligentes existentes há cada vez mais a necessidade de saber quais os mecanismos que são mais eficazes e eficientes na descoberta dos serviços desses objetos, principalmente através de *smartphones*.

O trabalho que será apresentado que conta com o estudo, desenho, implementação e avaliação dos mecanismos de descoberta de serviços existentes tem como relevância apresentar qual o melhor para ser implementado numa rede de objetos inteligentes tendo em conta a análise de métricas que passam tanto pelo gasto de energia como pela latência das comunicações.

1.4 Abordagem Preconizada

Este trabalho irá apresentar as melhores soluções ao problema introduzido através de avaliações e simulações de protocolos e tecnologias que se focam, essencialmente, na descoberta de objetos inteligentes e seus serviços na rede através de um dispositivo móvel.

Dessa forma, pode-se dizer que o trabalho irá estar dividido nas seguintes fases para dar resposta a este problema:

- Conhecer e estudar que protocolos permitem a descoberta de serviços numa rede;

- Avaliar esses mesmos protocolos tendo em conta as características únicas dos *smartphones* e dos objetos inteligentes;

- Desenvolver um caso de estudo que compare alguns desses protocolos e permita obter dados relevantes.

1.5 Estrutura do Documento

O documento está divido em quatro capítulos:

- Introdução – onde está presente o contexto deste trabalho, os objetivos, a motivação para o desenvolver, entre outros;
- Estado da Arte – onde é apresentada as tecnologias existentes ao redor dos objetos inteligentes assim como implementações relevantes no que à descoberta de serviços diz respeito;
- Implementação – onde é desenvolvida a simulação dos protocolos em causa e analisado os valores obtidos na mesma;
- Conclusão – onde é feita a conclusão do documento e apresentado algum do trabalho que poderia ser elaborado no futuro.

A estrutura do documento tem esta forma por ser a que mais sentido faz tendo em conta o trabalho desenvolvido e foi aprovada tanto pelo autor do documento como pelo seu respetivo orientador.

Por isso, relativamente à disciplina TMDEI e aos seus *Outcomes*, a Tabela 1 apresenta os respetivos capítulos em que cada *Outcome* se encontra abordado.

Tabela 1 – Integração dos *Outcomes* nos capítulos existentes do documento

Outcome	**Capítulo**
Outcome 1	Introdução
Outcome 2	Introdução, Estado da Arte
Outcome 3	Estado da Arte
Outcome 4	Implementação
Outcome 5	Implementação
Outcome 6	Implementação

2 Estado da Arte

Nos próximos pontos serão abordados temas relacionados com o contexto deste trabalho, nomeadamente objetos inteligentes, suas tecnologias e funções e a descoberta de serviços. Será também abordado as tecnologias atuais mais utilizadas e qual a sua relevância para a comunicação destes dispositivos na rede.

2.1 Objetos Inteligentes

Os objetos inteligentes têm tido cada vez mais atenção derivado do crescimento da Internet das Coisas. Este é um conceito que descreve exatamente o que estes objetos são: objetos do dia-a-dia ligados à Internet. O fato de estarem ligados à Internet é o que lhes proporciona uma das grandes diferenças perante os objetos comuns, nomeadamente a capacidade de serem identificados e conseguirem comunicar com outros dispositivos. Aliado a isso, ao estarem habilitados com componentes de computação, sensores, atuadores e *software* embutido, são capazes de fazer o ambiente ao seu redor mais "inteligente" ao mesmo tempo que possibilitam que o utilizador possa estar no controlo desse meio.

Os avanços tecnológicos ao nível da construção de sensores têm levado a que cada vez sejam mais pequenos e baratos e estejam preparados para processos de computação e de comunicação sem fios (Farooq & Kunz, 2011).

Existem variadíssimos tipos de implementações de objetos inteligentes, de seguida serão apresentados alguns dos mais conhecidos.

2.1.1 Beacons

Os Beacons são dispositivos de localização que atuam normalmente como uma espécie de posicionamento dentro de um espaço interior. Comunicam com *smartphones* e *tablets* através da tecnologia Bluetooth Low Energy e servem para apresentar ofertas altamente personalizadas ao utilizador dependendo da posição do mesmo, dos seus gostos, do seu

histórico com uma determinada marca, entre outras coisas. Estas informações são obtidas através das aplicações que são instaladas nos *smartphones* ou *tablets*.

Figura 1 – Exemplo de Beacon (McWilliams, s.d.)

2.1.2 Wearables

Os *wearables* são as chamadas tecnologias vestíveis, uma vez que são dispositivos que podem ser utilizados exatamente como uma peça de vestuário. Óculos, relógios, pulseiras, sapatilhas, são alguns dos exemplos de onde componentes tecnológicos podem ser inseridos para aumentar a fonte de informação daquilo que fazemos ou vemos.

Os mais conhecidos talvez sejam os chamados *fitness trackers,* aqueles que têm como objetivo a obtenção de dados aquando da prática de exercício físico. Normalmente estes dispositivos são pulseiras ou relógios que podem ou não ter um ecrã, mas que certamente terão sensores para obtenção de dados de batimento cardíaco, GPS e rádios para realizar comunicações com outros dispositivos, nomeadamente *smartphones*.

Figura 2 – Exemplo de *fitness tracker* (Fitbit)

2.1.3 Wireless Sensor Networks

Algo que é comum se referir quando se menciona o ambiente dos objetos inteligentes é a chamada Rede de Sensores Sem Fios (Wireless Sensor Networks - WSN).

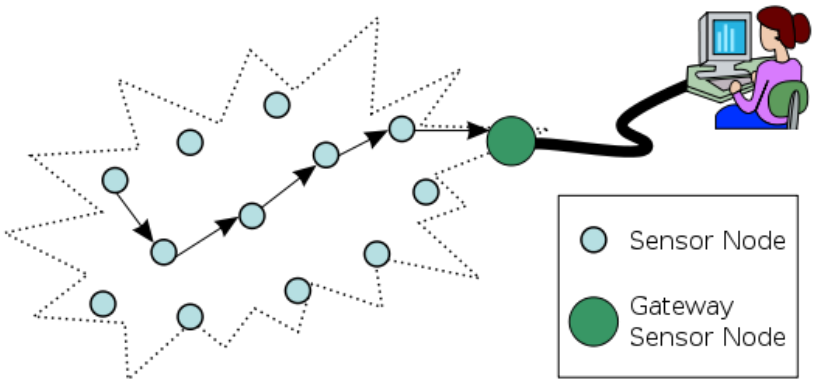

Figura 3 – Rede de sensores sem fios (Mallikarjuna Reddy V, s.d.)

Como o nome indica, é uma rede de sensores sem fios constituída por sensores autónomos distribuídos de forma a monitorizar condições do seu ambiente e transmitir essa informação para uma certa localização na rede (Farooq & Kunz, 2011).

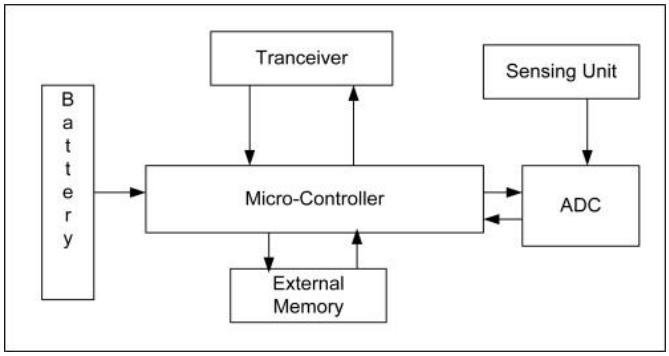

Figura 4 – Exemplo típico da arquitetura de um sensor (Farooq & Kunz, 2011)

Na Figura 4 é possível observar a típica arquitetura de um nó numa rede de sensores. É composto por uma bateria, um microcontrolador, memória externa, um recetor/emissor que faz a comunicação na rede, uma unidade sensorial e o conversor de analógico para digital responsável por converter os dados da unidade sensorial.

2.2 Tecnologia Relevante

Nesta subsecção irão ser apresentadas várias tecnologias, protocolos e conceitos relevantes para o estudo ao nível das comunicações dos dispositivos inteligente. Serão abordagens que têm como finalidade mostrar como podem comunicar estes objetos entre si ou pela Internet.

Uma das principais preocupações quando se fala em dispositivos que estão constantemente disponíveis e prontos a estabelecer conexões é a do gasto energético. Ainda mais importante se torna quando estes estão equipados por baterias que necessitam da maior autonomia possível. É por isso essencial estudar e implementar formas de conseguir ao máximo estender a autonomia possível de cada dispositivo, seja ao nível de *hardware* ou *software*, seja ao nível das comunicações *wireless* que cada dispositivo necessita.

No que diz respeito às comunicações *wireless*, existem diversos protocolos e *standards* que se focam especialmente no ambiente criado ao redor da Internet das Coisas e da Rede de Sensores Sem Fios.

2.2.1 Camada Física

Começando pela camada física, os padrões mais utilizados recaem no já conhecido IEEE 802.11, redes Wi-Fi, no também conhecido IEEE 802.15.1, em que é baseado o Bluetooth, e no IEEE 802.15.4. Os três funcionam como mecanismos de rádio sem fios, no entanto, apresentam diferenças no consumo energético e na qualidade do serviço. O IEEE 802.15.4 é um mecanismo de transmissão de curto alcance ao contrário do IEEE 802.11 que além de ter um maior alcance, é especialmente desenhado para suportar comunicações de alta velocidade.

O *standard* IEEE 802.11 é já amplamente conhecido e utilizado, consegue na sua última versão "n" ter uma taxa de transferência de 600 Megabits por segundo com uma largura de banda de 40 Mhz.

No entanto, tem um consumo energético elevado, uma vez que foi desenhado para ser uma forma de transporte de alta velocidade para computadores e não com a ideia de ser utilizado em pequenos dispositivos inteligentes. Daí que se tenha procurado conseguir diminuir a energia necessária para o funcionamento deste padrão e atualmente existam já muitos módulos que trabalham com transmissões de dados a um consumo baixo. Prevê-se até que até ao final do ano de 2016 outra versão deste *standard* esteja disponível no mercado, o chamado 802.11ah ou Wi-Fi HaLow, especialmente pensado para os dispositivos da Internet das Coisas. Anunciado em janeiro de 2016, esta versão irá operar numa largura de banda de 900 Mhz que ajuda ao baixo consumo energético e ao maior alcance na transmissão de dados. É expectável que o raio de alcance de um dispositivo Wi-Fi HaLow seja exatamente o dobro do atualmente conseguido numa rede IEEE 802.11 (Suthers, s.d.).

Figura 5 – IEEE 802.11ah Wi-Fi HaLow (Why The New 802.11ah Wi-Fi Standard Will Give Z-Wave A Run For It's Money, 2016)

Dito isto, no contexto dos objetos inteligentes, o 802.11 tem vários aspetos positivos. Citando um livro (Vasseur & Dunkels, 2010) que apresenta algumas dessas vantagens:

[…] A adoção alargada do 802.11 possibilita a implementação de objetos inteligentes de forma fácil. Em locais em que uma rede 802.11 exista, não é necessária infraestrutura adicional para suportar uma rede de objetos inteligentes. Além disso, a larga disponibilidade de placas de rede, routers e outros acessórios que utilizem o 802.11, reduz o custo de hardware para a criação de dispositivos que suportem esta tecnologia. Ainda para mais, a adoção alargada e a disponibilidade existente leva a que exista um elevado conhecimento sobre a mesma. No negócio dos objetos inteligentes, isto faz com que subsista no mercado bastantes arquitetos e engenheiros com habilidades para tal.

De seguida, o padrão IEEE 802.15.1. Este é mais conhecido por ser a especificação para a tecnologia Bluetooth.

Tem como principal objetivo a partilha de informações entre dispositivos e o baixo consumo de energia, de 1 mW a 100 mW, e num alcance pequeno, normalmente até 100 metros, dependendo da classe da tecnologia. A sua taxa de transmissão na versão 3 é de 24 Megabits por segundo. No entanto, foi na versão 4.0 que foi introduzido o chamado Bluetooth Smart, ou mais conhecido por Bluetooth LE (Low Energy).

Comparado com o Bluetooth tradicional, o Smart tem como ideia disponibilizar um consumo de energia ainda mais pequeno e um custo baixo, mantendo um alcance similar. Todos os sistemas operativos mais populares no mercado suportam esta tecnologia nas suas últimas versões. Além disso, é a utilizada nos dispositivos Beacon que já foram mencionados neste documento.

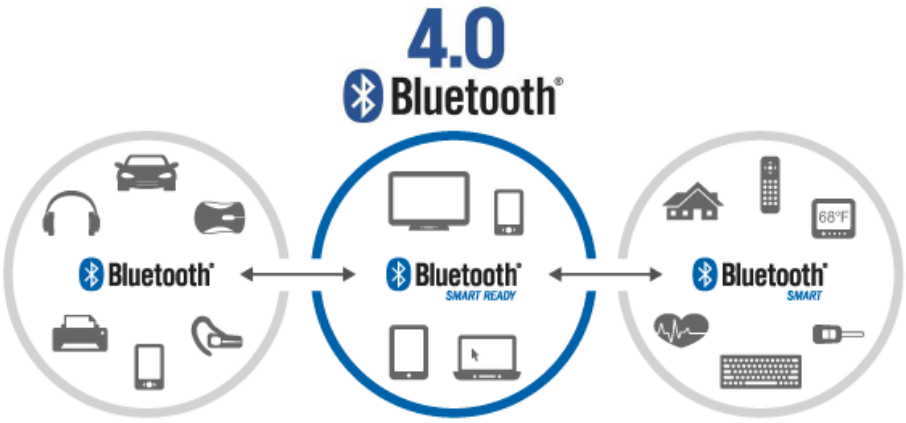

Figura 6 - IEEE 802.15.1 Bluetooth Low Energy (www.bluetooth.com)

Na versão 4.2 foi implementada a capacidade de conetividade através de IP através do IPSP (Internet Protocol Support Profile) que adiciona a tecnologia IPv6 ao Bluetooth Smart, com o intuito de ser implementado em dispositivos inteligentes contemplados na Internet das Coisas. No entanto, maior parte dos dispositivos no mercado com esta tecnologia não suporta ainda esta versão, o que faz com que uma implementação desta tecnologia em objetos inteligentes obrigue a que qualquer comunicação com o próprio seja feita através de um dispositivo que suporte Bluetooth e que esteja emparelhado com o mesmo. Por exemplo, numa suposta atualização de *firmware* que seja necessária no objeto inteligente, esta terá que ser feita recorrendo a um segundo dispositivo que deverá descarregar os ficheiros necessários e envia-los para o objeto, uma vez que este não terá acesso direto à Internet.

Relativamente ao padrão IEEE 802.15.4, como já foi mencionado, este é um *standard* especificamente desenvolvido para comunicações de baixa energia. Dessa forma, possui um curto alcance de transmissão. Isto torna necessário que todos os nós da rede estejam preparados para distribuir tráfego uns pelos outros, já que um nó pode não conseguir chegar a todos os restantes. Este *standard* tem uma taxa de transferência de dados de 250 *kilobits* por segundo e um consumo máximo de 1 mW, ajudando a que seja uma escolha bastante comum na construção de objetos inteligentes com baixas taxas de transmissão e complexidade e sem grandes custos (Vasseur & Dunkels, 2010).

No que diz respeito às camadas, o IEEE 802.15.4 especifica duas, a camada física e a camada MAC (Media Access Control). Cada pacote transmitido tem um tamanho de 127 *bytes* mas como a camada MAC adiciona um cabeçalho de tamanho variável, o espaço para a transmissão de dados fica-se pelos 86 a 116 *bytes*. As redes utilizando este padrão são divididas em redes PAN (Personal Area Network) em que cada uma tem um coordenador e um conjunto de membros. Cada pacote enviado neste tipo de rede transporta um identificador de 16 *bit* que especifica a que PAN o pacote é direcionado.

O IEEE 802.15.4 é normalmente implementado numa combinação de *hardware* e *software*, isto é, as partes de baixo nível são implementadas no *hardware* e as de alto nível no *software*. Diversas implementações deste *standard* existem, tais como, ZigBee, IPv6,

WirelessHart, Isa100.11a, entre outros (Vasseur & Dunkels, 2010). A mais comum será talvez aquela que é baseada em IPv6, a 6LoWPAN que é basicamente um acrónimo para IPv6 sobre áreas de redes *wireless* pessoais de baixo de consumo de energia. Basicamente este protocolo ajuda a que o IEEE 802.15.4 consiga estabelecer ligações à Internet, baseando-se para isso na ideia que cada dispositivo deverá ter atribuído um IP tornando-se uma parte da referida Internet das Coisas (Shelby & Bornmann, 6LoWPAN: The Wireless Embedded Internet, 2009).

Este *standard* apresenta desvantagens em comparação com o 802.11 no caso dos objetos inteligentes num importante aspeto: não é largamente utilizado e não existem muitos dispositivos que o suportem. Maior parte dos *smartphones* existentes no mercado não trazem *hardware* para suportar esta tecnologia. Mesmo os sistemas operativos móveis, não têm de raiz qualquer implementação deste *standard* (por exemplo o ZigBee).

Por fim, na camada física, além dos três *standards* apresentados, podemos também incluir o chamado PLC, que é a comunicação via rede elétrica.

É uma tecnologia que utiliza a rede elétrica existente e por isso não usa qualquer rede sem fios, sendo possível ser agregada uma rede TCP/IP de camada superior. Isto requer que os objetos inteligentes estejam ligados à corrente elétrica existente, o que põe logo de parte todos aqueles objetos inteligentes que têm como finalidade serem móveis ou sem fios.

2.2.2 Camada de Rede

Estando analisado o estado da arte na camada física para as comunicações dos objetos inteligentes, podemos concluir que em todos os *standards* é possível dotar o dispositivo de um endereço IP para comunicações. É por isso importante analisar que implementações existem que aproveitem este fator e tenham novamente em conta a taxa de transmissão e o baixo consumo, uma vez que o conjunto de protocolos TCP/IP pode ser considerado complexo ou até pesado para ser usado em objetos inteligentes que têm sérias limitações ao nível do tamanho de memória disponível e no poder de processamento.

Para combater essa complexidade existe a tecnologia 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks).

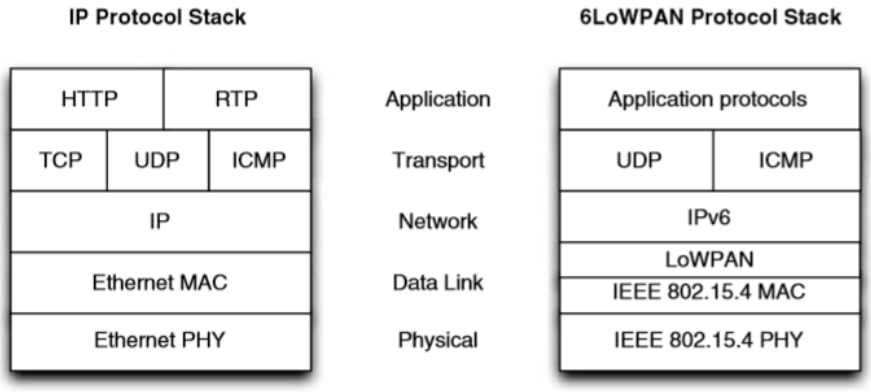

Figura 7 – Pilha IP e 6LoWPAN (Shelby & Bornmann, 6LoWPAN: The Wireless Embedded Internet, 2009)

Este protocolo tem como objetivo suportar IPv6 em redes 802.15.4, como é possível observar na Figura 7. Uma definição técnica deste *standard* pode ser descrita desta forma (Shelby & Bornmann, 6LoWPAN: The Wireless Embedded Internet, 2009):

[…] O standard 6LoWPAN torna possível o uso de IPv6 sobre redes wireless de baixa energia e baixa velocidade de transmissão em dispositivos embebidos através de uma camada de adaptação e da otimização dos protocolos relacionados.

Uma das características desta tecnologia é o fato de comprimir os cabeçalhos IP e UDP. O 6LoWPAN cria uma camada de adaptação entre o protocolo 802.15.4 e o IPv6 com cabeçalhos específicos que podem ser adicionados ou removidos consoante seja necessário. Isto permite que seja enviado apenas o que é útil e descartar o que não seja. Existe implementações deste *standard* para sistemas como o Contiki, TinyOS e FreeRTOS.

Outra implementação da pilha TCP/IP e que tem em conta os dispositivos da Internet das Coisas é o uIP (micro IP). O uIP é considerado por muitos (Vasseur & Dunkels, 2010) como o conjunto de protocolos ideal para os objetos inteligentes. É *open source* e tem a particularidade de ser especificamente desenhado para dar resposta aos requisitos dos objetos inteligentes: baixa memória e consumo de energia baixo. Na sua configuração por defeito requer apenas 1 *kilobyte* de RAM e muito pouco espaço de ROM. Muitos sistemas operativos utilizam esta tecnologia, com destaque para o Contiki, FreeRTOS e TinyOS. O uIP suporta tanto IP versão 4 como versão 6.

Existe também o NanoIP desenhado especialmente para os dispositivos embebidos (Shelby, Huuskonen, Mahonen, Riihijarvi, & Raivio, 2003). O conceito da criação deste protocolo foi o mesmo, trazer para dispositivos embebidos ou sensores uma rede de Internet sem o elevado peso que a pilha TCP/IP tem. O objetivo do NanoIP é ser o mais compacto e rápido possível tendo em mente as redes sem fios e o endereçamento local.

2.2.3 Camada de Aplicação

Ao nível dos protocolos de aplicação para a transmissão de informação, temos o já conhecido protocolo http e outro particularmente pensado para as redes de sensores *wireless*, o CoAP (Constrained Application Protocol).

Mais uma vez, a criação deste protocolo foi considerada para ser utilizado em dispositivos simples e mais limitados, de forma a que estes consigam comunicar na Internet e entre si (máquina-a-máquina).

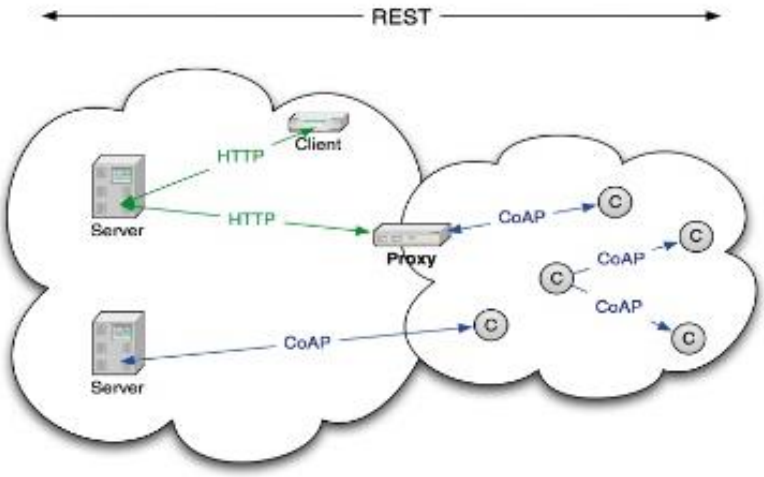

Figura 8 – Exemplo de uma implementação do CoAP (Selby)

O CoAP foi desenhado para facilmente ser traduzido para http, o que faz com que suporte *multicast*, uma baixa sobrecarga e seja simples. É um protocolo de desenvolvimento aberto, está embebido no sistema operativo Contiki e até pode ser suportado em dispositivos Arduino.

2.2.4 Sistemas Operativos

Para a criação de um sistema operativo para um objeto inteligente, deverão ser tidos em conta vários fatores. Um estudo (Baccelli, Hahm, Gunes, Wahlisch, & Schmidt, 2013) apresentou vários pontos bastantes importantes a ter em conta:

[…] ***Requisitos de Memória****: os dispositivos da Internet das Coisas têm pouca memória (tipicamente entre 5 kilobytes e algumas centenas de megabytes) e por isso os requisitos mínimos de memória do software deverão ser baixos. Isto afeta tanto a memória RAM como o espaço disponível para os programas.*

Requisitos de Processamento: *a complexidade das operações deverá ser baixa porque alguns dos microcontroladores dos sistemas IoT trabalham a um ciclo de relógio bastante baixo.*

Funcionalidades Limitadas: *o software deverá ser capaz de correr em hardware limitado sem componentes avançados como uma unidade de gestão de memória ou uma unidade de floating-point.*

Suporte de Plataformas: *o software para IoT terá que suportar uma variedade de plataformas, isto é, não só correr em plataformas mais limitadas, mas também ser capaz de aproveitar as capacidades das plataformas menos restritas.*

Eficiência Energética: *o software deverá explorar as funcionalidades do hardware para a gestão eficiente de energia e conseguir o máximo possível entrar em ciclos grandes de paragem total (sleep).*

Pilha de rede adaptada: *o software deverá disponibilizar implementações TCP/IP assim como 6LoWPAN de forma a suportar dispositivos mais limitados. Deverá também ser modular de uma forma que os protocolos em cada camada sejam fáceis de serem trocados.*

Fiabilidade: *os sistemas IoT são muitas vezes implementados em aplicações críticas em que o acesso físico é difícil e pode ter elevados custos. Por causa disso, é importante que o sistema seja robusto e tenha uma elevada taxa de fiabilidade.*

Standard API*: de forma a simplificar o desenvolvimento de software e a portabilidade de software existente, o sistema deverá disponibilizar uma interface de programação do tipo standard, tais como POSIX (Portable Operating System Interface) ou STL (Standard Template Library).*

Linguagens de Programação Standard: *suporte para linguagens de programação de alto nível, tais como C++, é fundamental.*

Partindo destes princípios genéricos relativos ao *software* que deverá ser executado em objetos inteligentes, é possível chegar à conclusão que os sistemas operativos para estes dispositivos deverão conseguir o balanço entre as limitações do *hardware* e a usabilidade para os desenvolvedores. Ou seja, por um lado, deverá ter um bom desempenho em diversas configurações com diferentes tipos de capacidades e por outro deverá ser capaz de explorar ao máximo as funcionalidades da plataforma em si.

Relativamente aos sistemas operativos existentes que têm conta estes fatores e que apontam sobretudo para a Internet das Coisas, temos os seguintes: Contiki, FreeRTOS, TinyOS e o mais recente Brillo da Google.

2.2.5 Contiki

Começando pelo Contiki, este é um sistema aberto especialmente desenhado para sistemas embebidos e redes de sensores sem fios. É altamente compacto e oferece uma configuração de protocolos bastante configurável. Uma configuração típica ocupa cerca de 2 *kilobytes* de RAM e 40 *kilobytes* de espaço para programas. Uma configuração completa incluí

funcionalidades como multitarefa, IPv6, ambiente gráfico, *web browser*, entre outras (Farooq & Kunz, 2011).

O Contiki segue uma arquitetura modular como é possível visualizar na Figura 9.

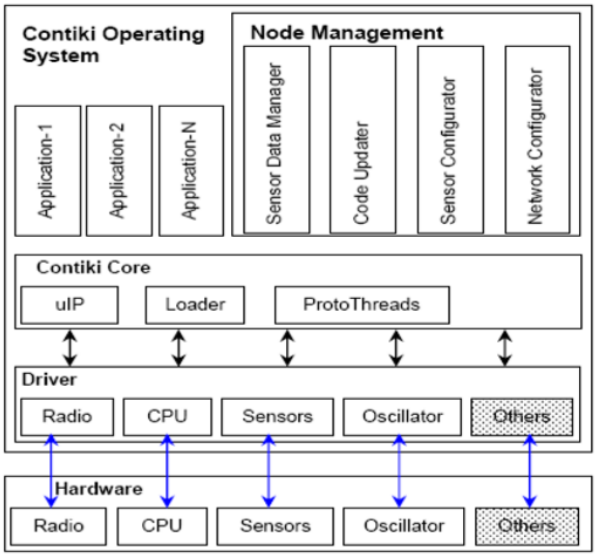

Figura 9 – Arquitetura do Sistema Operativo Contiki (Farooq & Kunz, 2011)

No que toca aos *standards* de Internet, o Contiki suporta a pilha TCP/IP e uIP e também as tecnologias 6LoWPAN e CoAP.

2.2.6 FreeRTOS

O FreeRTOS tem como objetivo ser um sistema para dispositivos embebidos e é conhecido por ser um sistema com uma *kernel* que funciona em tempo real. É aberto, desenhado para ser pequeno e simples e a sua *kernel* consiste em cerca de três ficheiros na linguagem C. Uma típica imagem binária do sistema ocupa menos do que 10 *kilobytes*. É de fácil leitura e manutenção e a sua portabilidade é simples. Consegue ser altamente configurável uma vez que tanto pode ser criado para ser executado num processador único suportando apenas algumas tarefas como pode ser criado para ser altamente funcional com processamento *multicore*, pilha TCP/IP ou uIP, sistema de ficheiros e USB (Svec, 2012).

Este sistema operativo está dividido em três áreas gerais (Svec, 2012):

- **Tarefas**: quase metade do código do sistema é responsável pelo tratamento das tarefas. Os ficheiros “tasks.c” e “tasks.h” tratam de criar, escalonar e manter as tarefas;
- **Comunicações:** perto de 40% do código do sistema trata das comunicações entre tarefas. Os ficheiros “queue.c” e “queue.h” são responsáveis por fazer as tarefas comunicarem entre si e assinalar o uso de recursos críticos utilizando semáforos ou outros;

- **Equipamentos:** as cerca de 9000 linhas de código são independentes de *hardware*. À volta de 6% do sistema age como uma ponte entre o código independente do *hardware* e o código dependente do *hardware*.

2.2.7 TinyOS

Quanto ao TinyOS, é também ele de desenvolvimento aberto e desenhado para dispositivos sem fios de baixo consumo normalmente na área da computação ubíqua. É flexível, orientado aos eventos, tem uma arquitetura baseada em componentes e usa uma linguagem baseada em C com extensões e componentes, a linguagem NesC. Suporta programas concorrentes com requisitos muito baixos de memória e consegue encaixar-se num bloco com 400 *bytes* de memória. Tem presente componentes para protocolos de rede (suporta a pilha TCP/IP e uIP e também a tecnologia 6LoWPAN), serviços distribuídos, *drivers* de sensores e ferramentas para aquisição de dados.

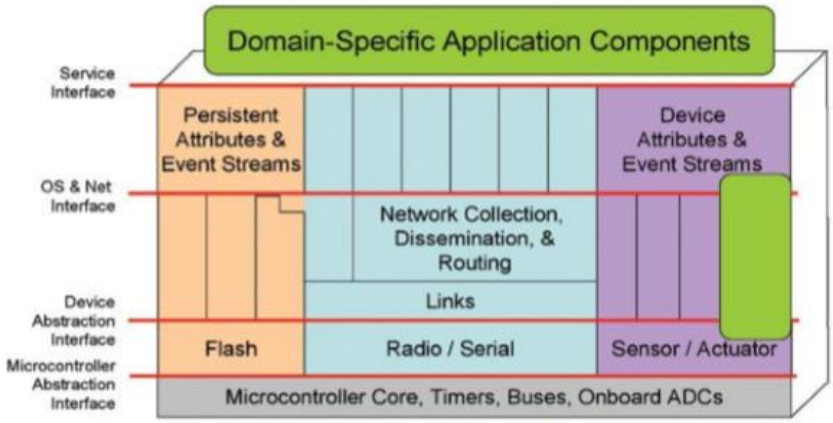

Figura 10 – Arquitetura do Sistema Operativo TinyOS (Farooq & Kunz, 2011)

2.2.8 Brillo

Por fim, o Brillo da Google, que é ainda um sistema operativo em fase de desenvolvimento e que foi anunciado em 2015 na conferência anual da empresa. Apesar disso, a Google tem já no mercado um dos sistemas operativos móveis mais utilizados no mundo, o Android, e juntando ao fato de ter nos últimos anos feito uma grande aposta seja em *software* (APIs no pacote Google Play Services presente no sistema operativo), seja em *hardware* (produtos NEST), prevê-se que o Brillo venha a ser a principal escolha no mercado assim que estiver disponível. Tecnicamente, e baseando-se num artigo disponibilizado pela Intel (Beare, 2016) sabe-se que é altamente baseado no Android mas tem um ambiente à volta de C/C++ e não possui qualquer aplicação em Java. Não terá suporte para gráficos e só precisará de 35 *megabytes* de memória para correr.

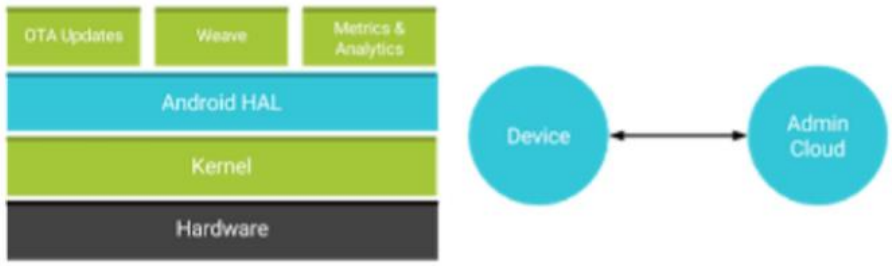

Figura 11 – Arquitetura do Sistema Operativo Brillo (Beare, 2016)

Irá comunicar através de um novo protocolo de comunicações introduzido também pela Google, que espera que seja também adotado por outros sistemas operativos para além do Brillo, o Weave. Este protocolo tem como objetivo a comunicações entre sistemas IoT e suportará como *standard* de transporte o 802.15.4 (por exemplo, ZigBee), o 802.15.1 (Bluetooth Low Energy), o 802.11 (WiFi) e o 802.3 (Ethernet).

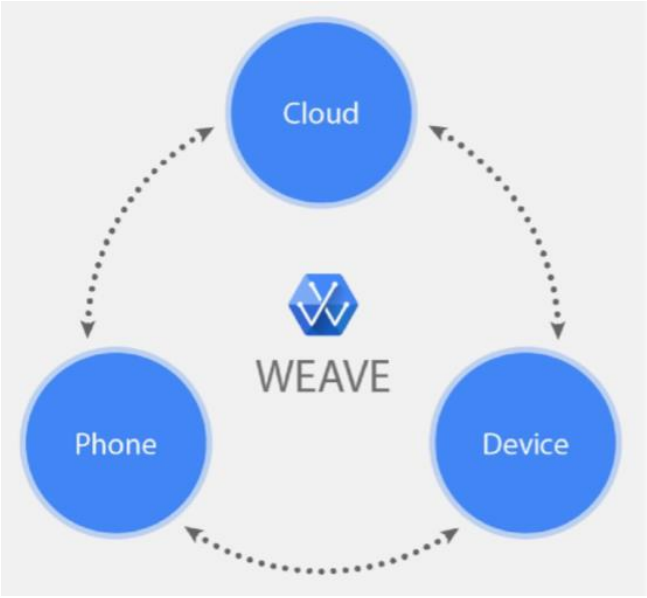

Figura 12 – Configuração da comunicação por meio do Weave (Beare, 2016)

2.2.9 Conclusão

Tendo em conta o que foi apresentado nos últimos capítulos e um estudo apresentado num artigo de IoT (Baccelli, Hahm, Gunes, Wahlisch, & Schmidt, 2013), é possível desenvolver uma tabela que faça a comparação entre uma implementação tradicional TCP/IP, utilizada por maior parte dos nós na Internet, com os protocolos correspondentes, numa implementação para redes de objetos inteligentes.

Tabela 2 - Comparação entre de pilhas de protocolos de Internet e IoT

	Internet	**IoT**
Camada de Aplicação	HTTP	CoAP
Camada de Transporte	TCP/UDP	UDP
Camada de Rede	IPv4/IPv6	6LoWPAN
Camada Física	802.3 (Ethernet), 802.11	802.15.4

2.3 Descoberta de Serviços

Nesta seção irá ser apresentado o estado da arte de soluções existentes para a descoberta de serviços numa rede, assim como algumas das suas vantagens e desvantagens relativamente à implementação em redes de objetos inteligentes.

A descoberta de serviços numa rede é fundamental não só para certos dispositivos funcionarem corretamente como também para se saber que serviços estão disponíveis para serem utilizados. Se pensarmos num caso de automação em que existe um controlador central que é responsável por ligar ou desligar certos aparelhos, este controlador terá que de alguma forma conhecer que aparelhos existem e que ações são possíveis de serem realizadas nesses aparelhos. Esta descoberta mais importante se torna quando se fala em objetos inteligentes já que estes, em particular, não têm ou então têm de forma bastante limitada, interação com o utilizador.

Os protocolos que implementem uma descoberta de serviços devem ter pelo menos dois componentes chave: o **cliente,** que tem um conjunto de requisitos que deverão ser satisfeitos pelo serviço, e o **dispositivo,** pelo qual é oferecido os respetivos serviços aos clientes que o requisitem. Cada elemento numa rede de serviços deverá poder ser um cliente, um dispositivo ou ambos ao mesmo tempo.

Os processos de descoberta de serviços estão muitas vezes associados à autoconfiguração. Esta é a responsável pelo método de um dispositivo se configurar a si mesmo com endereços de rede e outras informações relevantes para o seu funcionamento. A autoconfiguração não é responsável, no entanto, por configurar opções relativas à camada de aplicação. Essa tarefa é atribuída ao dispositivo em si.

A descoberta de serviços não está ainda estandardizada concretamente para objetos inteligentes. Além disso, a arquitetura IP, apesar de suportar a autoconfiguração de endereços, não possui ferramentas por defeito que façam a descoberta propriamente dita dos serviços existentes. Dessa forma, irão ser apresentados de seguida os mecanismos existentes mais

conhecidos e usados que dotam o protocolo IP de ferramentas capazes de fazer a descoberta. Apesar de alguns terem sido desenhados para descoberta de serviços de dispositivos tradicionais, como computadores e impressoras, as funcionalidades que contêm são vastas o suficiente para serem tidas em conta em objetos inteligentes.

2.3.1 SLP – Service Location Protocol

Como o nome indica, o SLP consiste num protocolo de descoberta de serviços. Pode-se definir este protocolo da seguinte forma (Vasseur & Dunkels, 2010):

[…] O SLP é um protocolo com um serviço leve de anúncios e pedidos que permite aos dispositivos anunciarem os seus serviços para outros dispositivos na rede e também para os próprios dispositivos pesquisarem na rede por certos serviços.

No SLP, os dispositivos podem ter uma de três funções: Service Agent (SA), User Agent (UA) ou Directory Agent (DA).

A função SA é indicada para aqueles dispositivos que queiram publicar os seus serviços. Têm também a responsabilidade de lidar e responder às tentativas de comunicação aos seus serviços.

Os dispositivos do tipo UA não disponibilizam qualquer serviço mas conseguem procurar serviços existentes na rede.

A função DA é responsável por manter uma agregação de dados de todos os serviços existentes na rede e dar resposta a eventuais questões lançadas pela função SA.

Relativamente à representação dos serviços neste protocolo, esta é feita através de URLs. Quando um dispositivo com a função UA pede por um serviço na rede, está a disponibilizar um URL contendo parte do tipo de serviço em que está interessado. É expetável que as respostas ao seu pedido venham parar a esse mesmo URL que foi gerado. Para exemplificar este processo, suponhamos a existência de uma impressora na rede. Se um dispositivo UA lançar um pedido na rede com o URL *service:printer*, a impressora que responda terá um URL que começará obrigatoriamente com *printer:*. Além dos URLs, um dispositivo SA pode também disponibilizar um conjunto de configurações através de parâmetros que informam o UA acerca do seu serviço. No exemplo dado para a impressora, estes parâmetros poderiam ser a possibilidade ou não de imprimir a cores, as dimensões do papel, entre outros (Vasseur & Dunkels, 2010).

As mensagens SLP são transportadas na rede por UDP porém, se uma mensagem atingir um limite de tamanho e tiver como destino um endereço *unicast*, esta será transmitida por TCP. Por causa da potencial falha de fiabilidade existentes no transporte por UDP, o SLP repete todos os pedidos *multicast* em intervalos crescentes de tempo até que uma resposta seja recebida. Todos os dispositivos deverão estar à escuta na porta 427.

Na figura seguinte, é demonstrado o comportamento básico numa rede com o SLP ativo.

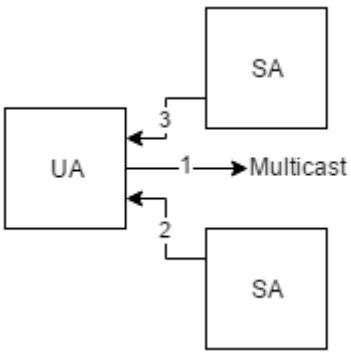

Figura 13 - Troca de informações básicas no SLP (Vasseur & Dunkels, 2010)

Como é possível observar, o processo inicia com o dispositivo UA a enviar uma mensagem com um pedido de um serviço em *multicast* para todos os dispositivos SA. Em resposta, estes enviam o seu serviço para o UA, ou seja, em *unicast*. As respostas dos SA têm uma particularidade de serem enviadas em intervalos com tempos aleatórios a partir do pedido do UA, com o intuito de não sobrecarrega-lo.

O comportamento do SLP é alterado assim que entra na rede um dispositivo do tipo DA. Como os dispositivos DA armazenam informação dos serviços existentes na rede, são estes que dão as respostas ao invés dos SA.

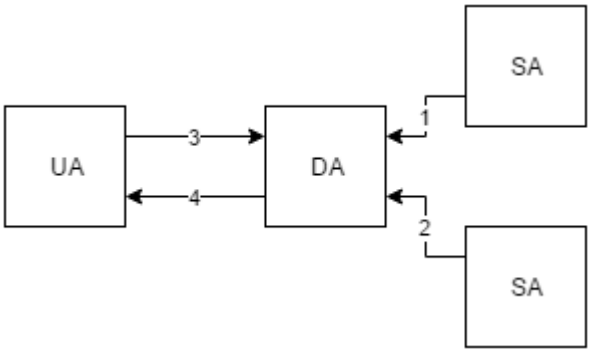

Figura 14 - Troca de informação com um dispositivo DA no SLP (Vasseur & Dunkels, 2010)

Os dispositivos SA descobrem o DA inicialmente através de mensagens específicas enviadas em *multicast.* Se o DA receber a mensagem responde com a sua disponibilidade. De salientar que o conhecimento do SA perante o DA tem uma validade especificada pelo SA quando este se regista. Quanto aos dispositivos UA, estes sempre que enviam um pedido na rede à procura de serviços, tentam sempre previamente descobrir se existem dispositivos DA e em caso afirmativo, é com eles que fazem a troca de informação acerca dos serviços existentes.

Este protocolo não vem incluído em nenhum dos sistemas operativos que foram referidos anteriormente como ideais para os objetos inteligentes. É, porém, suportado em outros sistemas mais utilizados como o Linux. A sua implementação não é difícil e com uma

breve pesquisa na Internet é possível encontrar algumas implementações (por exemplo, SLP Algorithm for TinyOS).

Para concluir, no que toca aos objetos inteligentes e tendo em conta alguns estudos (Vasseur & Dunkels, 2010), as vantagens do SLP são as seguintes:

- As mensagens trocadas são leves;
- Os URLs dos dispositivos podem ser compactados e codificados sem requerer nenhuma alteração ao SLP;
- A utilização de dispositivos com a função DA torna o SLP escalável o que é importante numa rede de objetos inteligentes em que podem existir centenas de nós ligados;
- Os mecanismos do SLP são simples, necessitam de pouca memória e têm um baixo consumo de energia.

2.3.2 Zeroconf

O Zeroconf é um conjunto de protocolos que além de disponibilizar a descoberta de serviços, consegue também a autoconfiguração de endereços e a resolução de nomes dos nós sem a presença de um servidor de nomes de domínio (DNS Server). Basicamente, consegue de forma automática criar uma rede baseada em IP quando os dispositivos são ligados, não requerendo intervenção nem qualquer configuração especial.

A Apple tem uma implementação desta tecnologia sobre o nome Bonjour, que vem integrada nos sistemas operativos da marca, o macOS e o iOS. Pode também ser instalada em sistemas Windows e vem incluída em *softwares* da marca tais como o iTunes e o Safari.

A parte de autoconfiguração de endereços é exatamente a mesma que a presente na tecnologia IPv4 e a descoberta de serviços é feita através dos protocolos IP e DNS com extensões de forma a não ser necessário a existência de um servidor de nomes de domínios.

O mDNS ou *multicast* DNS é uma dessas extensões. Este protocolo funciona ao realizar pedidos DNS num grupo *multicast*. Todos os nós da rede são considerados membros do grupo e todos recebem esses pedidos. Se um nó receber um desses pedidos e tiver o nome igual ao que esse pedido indica, esse nó responde ao nó origem que criou esse pedido.

A descoberta de serviços é feita através do DNS Service Discovery (DNS-SD) sobre o mDNS. Basicamente, o protocolo de descoberta de serviços adiciona aos nomes DNS uma descrição dos serviços existentes desse nó. O mDNS funciona na porta 5353.

Citando um estudo (Vasseur & Dunkels, 2010):

[…] O protocolo de descoberta de serviços no Zeroconf não está estandardizado, isto é, várias empresas implementam da forma que entendem. Por exemplo, a Microsoft utiliza o chamado Link-Local Multicast Name Resolution (LLMNR).

O Zeroconf também não é suportado de raiz pelos sistemas operativos para objetos inteligentes, apesar de o ser para todos os mais utilizados como Windows, macOS e Linux. Uma vez que este protocolo funciona através da resolução de nomes (DNS), algo que os sistemas operativos suportam, não será também difícil a sua implementação nos mesmos.

Novamente, no que toca aos objetos inteligentes e tendo como base certos estudos (Vasseur & Dunkels, 2010), pode-se afirmar que as vantagens do Zeroconf são:

- Os protocolos utilizados são simples;
- A implementação ocupa pouca memória;
- O peso das trocas de mensagens é baixo, o que é importante para dispositivos que comunicam por *wireless*.

No entanto, existem algumas desvantagens:

- A arquitetura da tecnologia não prevê um agente que sirva como *cache*. Dessa forma é necessário enviar mensagens a todos os nós da rede;
- A escalabilidade poderá ser um problema em redes com muitos nós.

2.3.3 UPnP – Universal Plug and Play

O UPnP é considerado como um sistema completo de configuração e de descoberta de serviços, tal como o Zeroconf. Foi originalmente desenvolvido pela Microsoft mas o trabalho continuou a ser desenvolvido pelo fórum UPnP (Vasseur & Dunkels, 2010).

É implementado sobre IP e utiliza os *standards* do protocolo IP, tais como HTTP e HTTP-over-UDP. Utiliza também o formato SOAP para o encapsulamento dos dados e XML para o formato dos dados.

Como foi mencionado, além da descoberta de serviços, o UPnP oferece também autoconfiguração de endereços numa rede, controlo de dispositivos e a apresentação dos mesmos, sem a ajuda de um servidor DHCP. Tendo em conta os mecanismos de controlo que esta tecnologia oferece, um dispositivo pode enviar um pedido a outro para este realizar certos tipos de ações. Com os mecanismos de apresentação, é possível obter informação visual sobre, por exemplo, o estado de um dispositivo.

Para a configuração de endereços, o UPnP utiliza as funcionalidades por defeito do protocolo IP. Para a descoberta de serviços, é utilizado o protocolo chamado Simple Service Discovery Protocol (SSDP).

No SSDP, os serviços são descritos por URLs e são utilizadas mensagens http transferidas pelo protocolo de transporte UDP. Essas mensagens consistem num pedido com dados descritos em SOAP em que é mencionado o tipo de serviço a ser descoberto. As mensagens são enviadas em *multicast* com destino a um grupo específico. Os nós que queiram

participar na descoberta de serviços entrar no suposto grupo para terem também acesso às mensagens.

O SSDP suporta anúncios e descoberta de serviços. Um qualquer dispositivo pode, repetidamente, anunciar a sua presença e os serviços que disponibiliza de forma a que novos dispositivos o possam encontrar. Porém, os novos dispositivos podem também enviar um pedido no grupo para conhecer todos os serviços ativos.

Ao dispor de suporte tanto para anúncios, como para descoberta dos serviços, o SSDP reduz de forma significativa o carregamento em geral na rede sempre que um dispositivo novo se junta.

Uma vez que alguns dos sistemas operativos para os objetos inteligentes utilizam CoAP ao invés de http na camada de aplicação, e o fato de o UPnP ser baseado em http, é normal que este protocolo não seja suportado em qualquer sistema. É, no entanto, possível de implementar utilizando CoAP ao invés de http nesses sistemas.

Aplicando aos objetos inteligentes, existem muitas desvantagens na utilização da arquitetura UPnP (Vasseur & Dunkels, 2010):

- Complexidade da implementação dos protocolos;
- Grande utilização de largura de banda e energia;
- Utilização de XML e SOAP que não foram desenhados para representarem os dados de forma compacta;
- Normalmente, as mensagens enviadas por UDP são pesadas e a especificação da arquitetura obriga a que seja tudo enviado num só pacote.

2.3.4 Jini (Apache River)

Originalmente desenvolvido pela Sun, esta tecnologia faz parte agora da Apache e tem como objetivo disponibilizar uma infraestrutura para a criação de arquiteturas orientadas a Objetos e Serviços (Service-object-oriented architecture).

Os seus protocolos para descoberta de serviços foram desenhados para descobrir e invocar serviços em dispositivos compatíveis com a linguagem Java. A sua arquitetura assenta em três componentes:

- Serviço de pesquisa (Lookup Service) – é um elemento central onde os fornecedores de serviços registam os seus serviços e age como um repositório;
- Fornecedores de Serviços – são elementos na rede que dispõe de serviços;
- Cliente – é um elemento que requisita um determinado serviço.

Os serviços de pesquisa enviam mensagens através de *multicast* de forma a anunciarem a sua presença em resposta aos pedidos que são enviados também em *multicast* pelos clientes e pelos fornecedores de serviços.

Os fornecedores de serviços enviam objetos do serviço com os seus atributos de forma a registarem-se no elemento central. Estes objetos são escritos em Java e atuam como interfaces para os clientes poderem chamar um serviço remoto.

Uma característica desta tecnologia é que os clientes, ao subscreverem perante os serviços, poderão ser notificados aquando de alterações nos estados dos mesmos.

O Jini suporta também períodos de alocação, ou seja, o registo de um fornecedor de serviços no elemento central deverá ser periodicamente renovado ou de outra forma será excluído do repositório.

No contexto dos objetos inteligentes, o Jini tem como desvantagens:

- Complexidade da arquitetura;
- Obrigatoriedade de ter a correr uma máquina virtual Java (JVM).

No caso dos sistemas operativos que foram já referidos seria necessário que estes tivessem instalado de alguma forma a tecnologia Java.

2.3.5 Bluetooth SDP – Bluetooth Service Discovery Protocol

O objetivo do Bluetooth SDP é disponibilizar formas dos dispositivos através de Bluetooth se encontrarem e trocarem informações tendo em conta as suas características. Por exemplo, um telemóvel ao ligar-se a um auricular Bluetooth, utiliza o Bluetooth SDP para determinar que funções são suportadas pelo auricular e as configurações necessárias para conseguir utilizar cada uma.

Cada serviço é identificado por um UUID (Universally Unique Identifier) de 128 *bits*. No caso de ser um serviço reconhecido oficialmente, este UUID terá o tamanho de 16 *bits.*

O modo de funcionamento assenta no modelo Cliente-Servidor, em que o dispositivo que inicia a comunicação é considerado como o SDP Client e o dispositivo destino o SDP Server. É neste que os dispositivos registam os serviços que fornecem.

A vantagem do Bluetooth SDP no que diz respeito aos objetos inteligentes:

- Funciona em Bluetooth, e por isso poderá ser usado em tecnologias Bluetooth Smart que têm como objetivo o baixo consumo de energia.

No entanto, pode ter como desvantagem:

- Apenas funciona em objetos com Bluetooth e estes não conseguirão descobrir outros que utilizem outros protocolos de rede.

A tecnologia Bluetooth é suportada tanto no sistema operativo Contiki como no TinyOS.

2.3.6 AllJoyn

O AllJoyn foi apresentado em 2011 pela empresa Qualcomm e é um sistema que tem como objetivo essencialmente disponibilizar serviços que tornem possível a comunicação entre dispositivos próximos e/ou aplicações com vista a criar redes dinâmicas.

É de desenvolvimento aberto e é pensado para redes de objetos inteligentes. Oferece descoberta, comunicação, controlo e partilha de recursos com outras aplicações e dispositivos inteligentes na rede.

O sistema utiliza uma espécie de modelo Cliente-Servidor para se organizar. Por exemplo, uma luz inteligente numa rede poderá ser considerada como o servidor e o interruptor como o cliente.

Cada servidor existente na rede possui um ficheiro XML de nome "introspection" que é utilizado para anunciar as capacidades do dispositivo e aquilo que pode fazer.

Este sistema é capaz de funcionar em basicamente todas as plataformas populares no mercado, como é o caso do Linux, Windows, iOS, Android e FreeRTOS. Não é conhecida qualquer compatibilidade com Contiki ou TinyOS.

As vantagens deste *standard* no contexto dos objetos inteligentes são:

- Orientado e focado para os objetos inteligentes;
- Utiliza mecanismos simples.

De momento, tem como desvantagem:

- Ao nível das redes *wireless,* apenas são suportadas as redes Wi-Fi (802.11).

2.3.7 Conclusões

A descoberta de serviços é bastante importante para que uma aplicação conheça que serviços estão disponíveis na rede para serem utilizados. No contexto dos objetos inteligentes, a descoberta dos serviços que estes oferecem é um importante mecanismo para conhecermos a presença de cada um, o que é possível obter deles e até permitir controlar e gerar informações a partir de dispositivos móveis.

Como foi visto nos pontos anteriores, não existe um *standard* aceite globalmente no que diz respeito aos protocolos para descoberta de serviços. Tendo em conta a evolução que se prevê para os objetos inteligentes, este fator ainda mais preocupante se torna.

2.4 Implementações

Analisados os protocolos e arquiteturas existentes para a descoberta de serviços numa rede, é importante mencionar as tecnologias existentes mais conhecidas e utilizadas no mercado que utilizem esses protocolos.

2.4.1.1 DIAL – Discovery And Launch

O DIAL é um protocolo desenvolvido pela Netflix e o Youtube para descobrir e abrir aplicações numa rede. Utiliza os protocolos UPnP e http.

Não requer qualquer tipo de emparelhamento entre dispositivos e é conhecido por implementar o chamado segundo ecrã (*2nd Screen*), como *tablets, smartphones,* computadores, entre outros. Normalmente os dispositivos de primeiro ecrã são televisões, dispositivos de reprodução de vídeo, etc.

Este protocolo foi inicialmente utilizado pelo aparelho de multimédia Chromecast da Google quando foi lançado, em 2013.

2.4.1.2 DLNA – Digital Living Network Alliance

O DLNA é um grupo que tem como objetivo promover um conjunto de orientações para dispositivos multimédia. As mais relevantes incluem a utilização de UPnP para gestão de *media* e descoberta de serviços e controlo de dispositivos. Todos os dispositivos que estejam em conformidade com esta tecnologia, são classificados como "DLNA Certified" através de um logotipo na sua caixa.

Atualmente, maior parte dos *smartphones* existentes no mercado vêm com esta certificação.

2.4.1.3 Google Cast

O Google Cast é uma tecnologia que permite partilhar conteúdo de um telemóvel, *tablet*, ou computador, para uma televisão, colunas, ou outros dispositivos permitidos.

Esta tecnologia é a base de funcionamento do Chromecast da Google, dispositivo esse que se liga através de HDMI a uma televisão ou através de uma entrada de som.

O Google Cast utiliza Zeroconf para a descoberta de serviços na rede, mais propriamente o mDNS.

2.5 Avaliação dos Mecanismos

Nesta seção irá ser avaliado todos os mecanismos de descoberta de serviços de forma a serem escolhidos aqueles que farão parte da simulação que se pretende elaborar.

Os mecanismos a avaliar são aqueles que foram estudados na seção de Estado da Arte, nomeadamente o **SLP,** o **Zeroconf**, o **UPnP**, o **Jini**, o **Bluetooth SDP** e o **AllJoyn**. Tendo por base as abordagens aí descritas, é apresentada de seguida uma tabela que resume as vantagens e

desvantagens de cada um desses mecanismos e outra tabela em que são postos em causa fatores importantes para a implementação dos protocolos e respetivos casos de estudo desses mecanismos.

Tabela 3 - Vantagens e Desvantagens de cada protocolo de descoberta de serviços

Mecanismo	**Vantagens**	**Desvantagens**
SLP	• Mensagens trocadas são leves; • URLs dos dispositivos podem ser compactados; • É escalável; • Mecanismos simples; • Utilização baixa de memória; • Baixo consumo energético.	
Zeroconf	• Protocolos simples; • Utilização baixa de memória; • Mensagens trocadas são leves.	• Não é facilmente escalável; • Não há mecanismos de cache.
UPnP		• Protocolos complexos; • Grande utilização de energia e largura de banda; • Estrutura de dados não é compacta; • Mensagens trocadas são pesadas.
Jini		• Arquitetura complexa; • Necessidade de utilização de uma máquina virtual Java.
Bluetooth SDP	• Funciona em tecnologias Bluetooth Smart com o objetivo de ter um baixo consumo energético.	• Necessidade de uma rede Bluetooth.
AllJoyn	• Orientado a objetos inteligentes; • Protocolos simples;	• Apenas suporta redes Wi-Fi (802.11).

Tabela 4 - Avaliação de cada protocolo de descoberta de serviços

Protocolo	**Facilidade de Implementação**	**Documentação Existente**	**Utilização Geral**
SLP	Fácil	Grande	Média utilização
Zeroconf	Complexa	Grande	Muita utilização
UPnP	Complexa	Grande	Muita utilização
Jini	Complexa	Baixa	Pouca utilização
Bluetooth SDP	Complexa	Baixa	Pouca utilização
AllJoyn	Complexa	Média	Pouca utilização

Com estas duas tabelas é possível avaliar e escolher quais são os mecanismos que fazem sentido e são possíveis de avaliar e de desenvolver casos de estudo.

Os escolhidos foram o **SLP** e o **Zeroconf**. A escolha destes recaiu no fato de ambos apresentarem várias vantagens quando aplicados aos objetos inteligentes e terem uma vasta documentação na Internet, o que ajuda à sua implementação. No caso do SLP, apesar de não ser um mecanismo muito utilizado, é um protocolo simples e a sua implementação é também fácil, o que será importante no desenvolvimento dos casos de estudo. O Zeroconf é mais complexo de implementar, mas é amplamente utilizado em dispositivos tais como o Google Chromecast e o Android, assim como em *software* tanto da Apple como Microsoft

O fato do SLP não ter capacidade para as funcionalidades de autoconfiguração de endereços e resolução de nomes não se tornou um fator importante uma vez que são funções que não são relevantes para a ideia principal de descoberta de serviços.

Relativamente aos restantes e às razões para as quais foram postos de parte, no caso do AllJoyn, apesar de oferecer um *kit* de desenvolvimento que permite criar aplicações para essa tecnologia, é um mecanismo recente e muito pouco utilizado.

Tanto o Jini e o UPnP foram postos de parte devido às limitações e desvantagens que têm ao nível das características necessárias para os objetos inteligentes. No caso do Bluetooth SDP, não foi possível encontrar um simulador que oferecesse facilmente modos de teste e a obrigatoriedade de uma rede Bluetooth também constitui um entrave ao seu estudo.

3 Implementação

Neste capítulo irá ser apresentado o trabalho desenvolvido relativamente ao estudo e desenvolvimento da simulação para a obtenção dos dados que possibilite conclusões acerca do protocolo mais vantajoso para a descoberta de serviços em objetos inteligentes.

Será referido o design da solução, as métricas que irão ser retiradas da simulação, a implementação dos protocolos no *software* de simulação, os dados obtidos, as validações desses dados e as conclusões sobre os mesmos.

3.1.1 Design

Existem dois grupos de fatores importantes que irão ser considerados no design da solução: métricas e simuladores.

No que diz respeito às métricas, existem critérios para avaliar a descoberta de serviços na rede, sendo alguns deles a latência e o consumo energético. A latência será o tempo até dois dispositivos se reconhecerem e estabelecerem uma ligação entre os seus serviços, assumindo que estão os dois na mesma rede. O consumo energético será a energia necessária para que esses dois dispositivos se descubram e troquem informações.

Para a obtenção destas métricas, será necessário realizar simulações dos mecanismos de descoberta previamente escolhidos. Serão, por isso, utilizadas implementações dos respetivos protocolos em estudo. O *software* responsável pela emulação será o Network Simulator 3, instalado e configurado num ambiente Linux, mais propriamente o Ubuntu 16.04. Este programa grátis com uma licença pública de utilização permite construir, desenhar e testar uma rede de forma virtual sem recurso a qualquer tipo de *hardware*. Dos diversos módulos e aplicações que o constituem, podem-se destacar as ferramentas que permitem a construção de redes LAN e WLAN, o envio de pacotes através de protocolos de transporte como o UDP e TCP, e também a análise energética de cada nó nas suas comunicações na rede.

Pretende-se, portanto, que o simulador permita que em cada instância que seja executado, exista a possibilidade de simular o respetivo mecanismo e obter as várias métricas. Para cada caso de teste, serão feitas várias simulações em diferentes ambientes com vista a simular utilizações reais. Também serão feitas iterações da mesma simulação nos mesmos ambientes e os seus resultados serão comparados de forma a garantir que os dados obtidos são constantes.

A simulação será construída a partir de uma rede Wireless 802.11. Esta escolha foi feita tendo em conta que é uma rede bem suportada no ambiente de simulação e que, como foi demonstrado no capítulo de Estado da Arte, será especialmente utilizada para a Internet das Coisas assim que sair a versão Wi-Fi HaLow. Estará a funcionar através de um ponto de acesso que por sua vez terá uma ligação com uma rede local com fios. O número de nós será variável o que significa que em diferentes simulações poderão existir mais ou menos nós que representam tanto objetos inteligentes como nós normais na rede. Isto enriquecerá o estudo uma vez que permitirá obter métricas em diferentes configurações e ambientes.

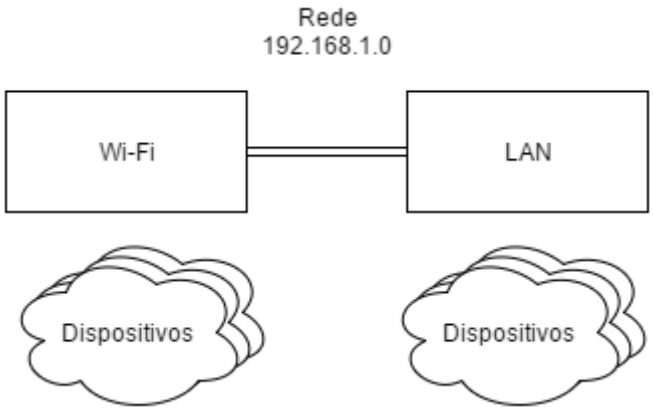

Figura 15 – Topologia por defeito da rede da simulação

Os nós presentes na rede sem fios serão os que farão parte da simulação que se pretende elaborar, é com eles que serão realizados os testes. Os restantes nós na rede com fios servirão apenas para emular um ambiente real em que outros dispositivos estejam também ligados na rede sem qualquer interferência direta na descoberta em si.

Em qualquer simulação, pelo menos um nó será emulado como sendo um dispositivo à descoberta, ou seja, será considerado como um dispositivo móvel que pretende descobrir que serviços de objetos inteligentes existem à sua volta. Os restantes nós serão objetos inteligentes que disponibilizam os seus serviços na rede de forma a poderem ser descobertos e utilizados.

Cada nó terá um endereço IP e no caso de ser um objeto inteligente terá também o respetivo serviço único que disponibiliza. Existirão no máximo seis nós que irão agir como objetos inteligentes e estarão sempre presentes na zona Wi-Fi.

O nó responsável por emular o dispositivo móvel terá a função de descobrir os nós que contenham serviços disponíveis e também ele será parte da zona Wi-Fi.

A partir do instante em que a simulação é iniciada, será posto em prática o respetivo protocolo de forma a obter os serviços disponíveis. Assim que todos sejam encontrados e tenha sido obtida a forma de ligação ao serviço propriamente dito, será finalizada a simulação. Aquando da simulação serão obtidas as métricas de tempo e gasto energético.

Na simulação não será contemplado o fato de existirem múltiplos serviços em cada nó, uma vez que normalmente os objetos inteligentes apenas contêm um único serviço.

3.1.2 Métricas

As métricas que irão ser utilizadas para avaliar o caso de estudo sobre os mecanismos de descoberta de objetos inteligentes e respetivos serviços serão as seguintes:

- Performance / Latência – É necessário que a descoberta seja feita de forma rápida. Esta métrica estará expressa em segundos;

- Consumo Energético – É um fator muito importante de ser avaliado na descoberta e comunicação principalmente em objetos inteligentes uma vez que estão muitas vezes limitados por uma bateria. É calculado tendo em conta o fator tempo e será expresso em Joules. O consumo energético calculado será o do conjunto dos nós da simulação, isto é, a energia gasta por cada nó ativo seja no processo de descoberta, seja no fato de apenas estar disponível na rede. Esta métrica é obtida através de funções provenientes do *software* em utilização da simulação.

3.1.3 Simulação

A simulação foi construída com base nas ferramentas disponibilizadas pelo programa Network Simulator 3.

O *software* é bastante modular e para além dos vários módulos que incorpora, tem a particularidade de ter processos simples para a adição de novas aplicações. No entanto, mesmo na vasta gama de aplicações desenvolvidas de raiz, o programa não possuía nenhuma implementação que recriasse os protocolos de descoberta pretendidos, por isso esse trabalho teve que ser desenvolvido à parte.

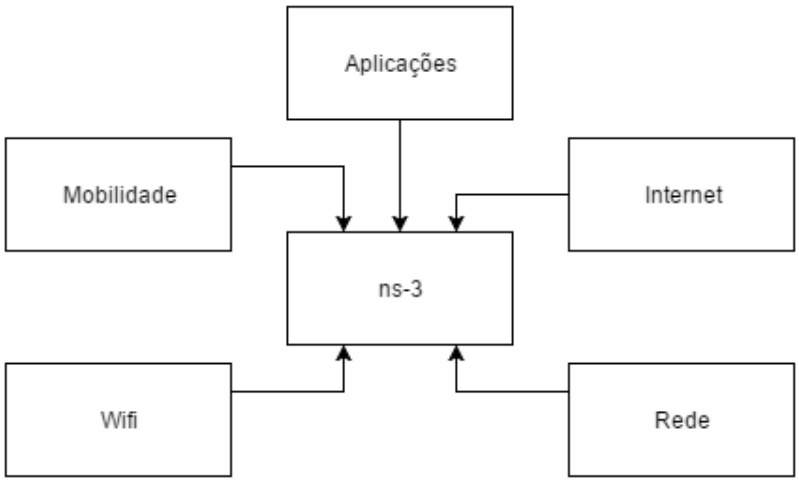

Figura 16 – Módulos obrigatórios para a implementação da simulação

No que toca ao desenvolvimento, na Figura 16 é possível observar os módulos necessários do *software* para uma simulação deste tipo. Estes módulos foram implementados em conjunto e este modelo seria seguido para a implementação de qualquer outro protocolo que tenha sido apresentado neste trabalho.

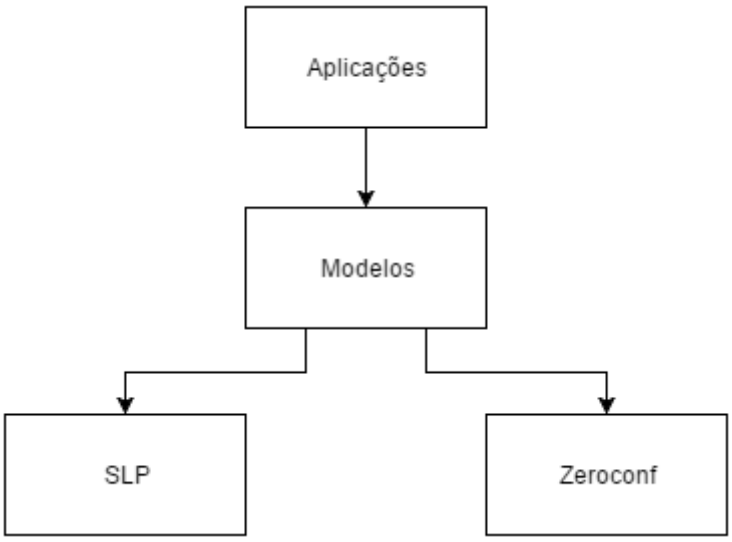

Figura 17 – Implementação dos protocolos no módulo de Aplicações

Quanto à lógica de cada protocolo, essa foi implementada no módulo de aplicações, como é possível observar na Figura 17. Porém, apenas foram desenvolvidas as funções básicas acerca do mecanismo, para que estes consigam que a descoberta dos serviços seja possível. Foram postas de parte todas as funcionalidades pertencentes aos mecanismos que não sejam relevantes para o estudo que se pretende demonstrar. Isto permitiu minimizar o tempo despendido na implementação de cada um e concentrar os esforços fundamentalmente na obtenção dos dados para a análise. Devido à similaridade dos protocolos, a implementação de cada um foi bastante idêntica.

A simulação foi construída de forma a implementar principalmente uma rede Wifi com base em IPv4 numa rede com o endereço IP 192.168.1.0/24. É baseada no *standard* 802.11n a uma frequência de 2.4GHz e tem fatores de mobilidade que permite que certos nós da rede se movam.

Estas configurações assim como a construção da topologia da rede e as ações que irão ser realizadas na simulação, estão presentes num ficheiro principal que é o responsável por lançar a simulação. Este ficheiro é então compilado e todos os módulos de que depende são também compilados e ligados ao mesmo. Se não existir nenhum erro, a simulação é iniciada.

Vários pontos técnicos da implementação de cada protocolo serão referidos de seguida.

3.1.3.1 SLP

No que ao protocolo SLP diz respeito, foi desenvolvida a lógica básica por detrás do mesmo. Cada nó da rede que implemente este protocolo tem um URL associado que corresponde ao serviço que disponibiliza.

A troca de informações começa pelo cliente que lança um pedido através de UDP na rede para o endereço de *broadcast.* A porta utilizada é a 427. Assim que o pedido é enviado, o cliente ficará à escuta de respostas que confirmem a existência de serviços e a respetiva forma de conexão aos mesmos, que será através do respetivo URL de cada serviço.

3.1.3.2 Zeroconf

Relativamente à implementação do Zeroconf, esta incidiu apenas sobre o mDNS e o DNS-SD. O primeiro é responsável por resolver nomes de nós na rede através de *multicast* e o segundo faz a descoberta dos serviços de cada nó. Por isso, no que à implementação diz respeito, foram criadas classes que contêm campos para o endereço *multicast* do grupo e para os nomes de domínio com os respetivos serviços.

Essas classes formam uma aplicação que é associada a cada nó na rede que se queira que tenha este protocolo implementado.

Sempre que um dispositivo lance um pedido na rede sobre os serviços existentes, esse pedido é enviado para todos os nós da rede que estejam associados a esse grupo *multicast,* através da porta 5353. Todos os pedidos são feitos em UDP. Cada nó no grupo irá responder com o seu nome de domínio.

De seguida, o cliente fará um novo pedido a esse domínio para obter a localização do respetivo serviço. A resposta será a forma para o cliente se ligar ao mesmo.

3.1.4 Dados

Nesta subsecção irão ser apresentados os dados obtidos de cada caso de simulação para cada protocolo.

Cada caso de simulação foi repetido várias vezes e os valores apresentados correspondem à media de todas essas repetições.

No primeiro caso, existe um serviço na rede ligado por Wi-Fi e um dispositivo à procura desse serviço.

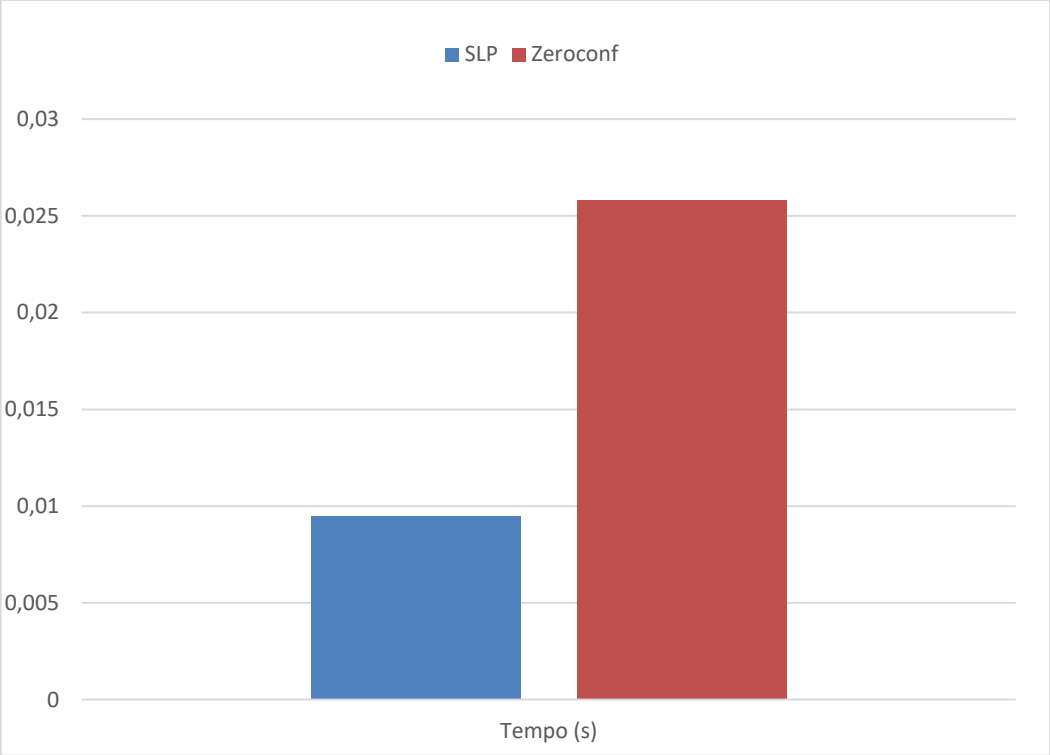

Figura 18 - Tempo total até à descoberta de um serviço por cada protocolo

Detalhes da configuração desta simulação:

- 3 nós Wi-Fi - um nó pesquisa serviços na rede, outro disponibiliza um serviço na rede;
- 5 nós Ethernet.

Para o único serviço existente ser descoberto, o protocolo SLP necessitou de cerca de 0,009 segundos enquanto o Zeroconf levou perto de 0,026 segundos. É mais do dobro a diferença entre cada um. Tendo isso em conta, no próximo caso, será aumentado o número de serviços disponíveis na rede para três.

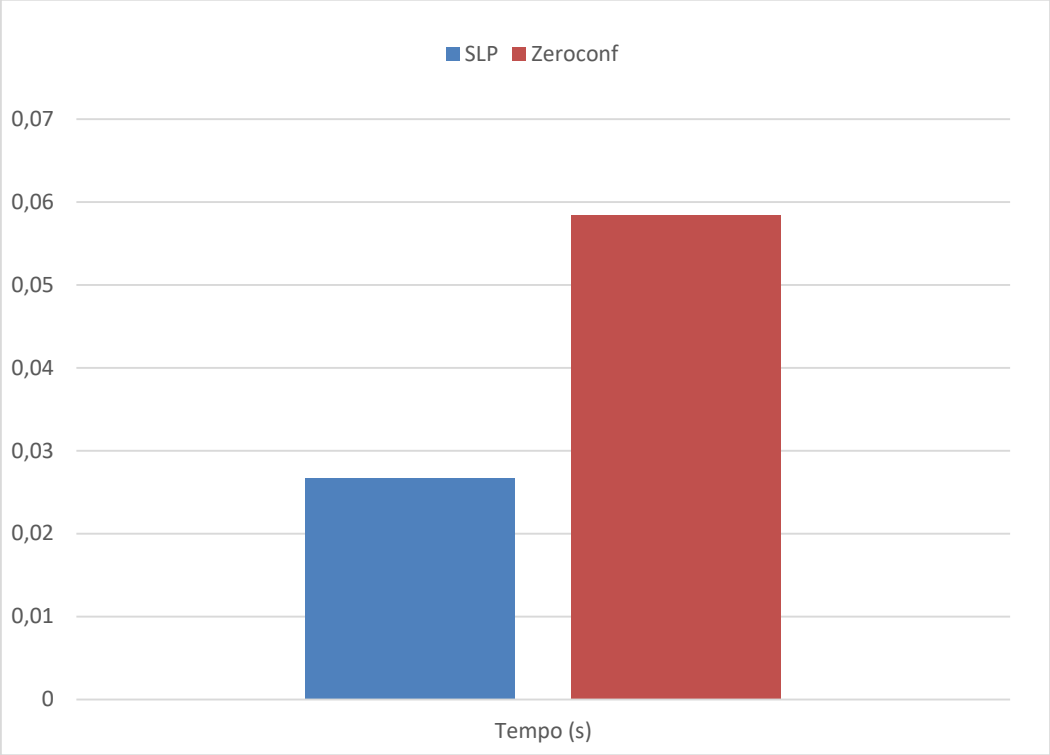

Figura 19 - Tempo total até à descoberta de 3 serviços por cada protocolo

Detalhes da configuração desta simulação:

- 5 nós Wi-Fi - um nó pesquisa serviços na rede, outros três disponibilizam serviços na rede;
- 5 nós Ethernet.

Nesta simulação é possível observar que o protocolo SLP levou perto de metade do tempo que o Zeroconf para encontrar todos os serviços disponíveis na rede. Porém, a diferença já não é tão considerável como no primeiro caso.

Aumentando-se novamente o número de serviços presentes assim como o número de nós na zona Wi-Fi obtém-se o gráfico seguinte.

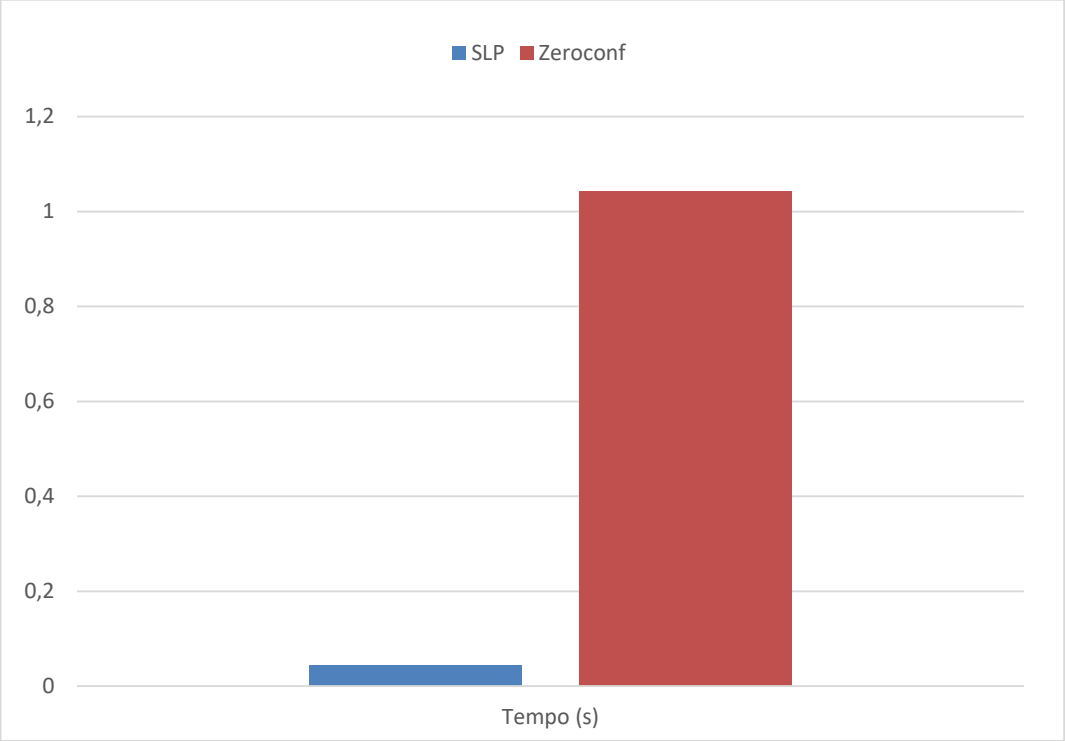

Figura 20 - Tempo total até à descoberta de 6 serviços por cada protocolo

Detalhes da configuração desta simulação:

- 10 nós Wi-Fi - um nó pesquisa serviços na rede, outros seis disponibilizam serviços na rede;
- 5 nós Ethernet.

A diferença entre cada protocolo é bastante mais acentuada agora. Para o Zeroconf encontrar todos os serviços disponíveis na rede levou pouco mais de um segundo, o que é mais do dobro do que na anterior simulação. Já o SLP manteve-se em valores abaixo de 0,1 segundos.

As conclusões que se retiram destes três casos de simulação é que no caso do Zeroconf, à medida que o número de serviços aumenta na rede, o tempo para a descoberta aumenta de forma exponencial, ao contrário do SLP, que tem aumentos mais lineares. No entanto, a diferença do primeiro caso para o segundo não foi tão significativa como do segundo para o terceiro, no caso do Zeroconf.

No que diz respeito ao gasto energético, replicando a primeira simulação obtém-se o seguinte gráfico.

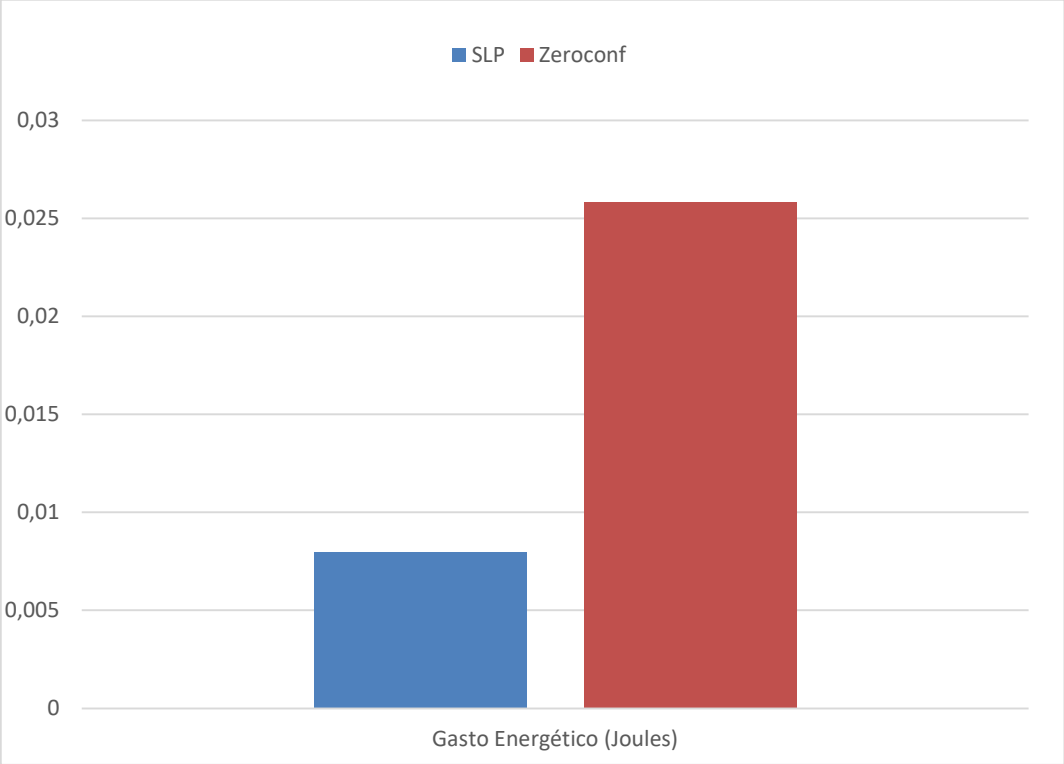

Figura 21 - Gasto de energia até à descoberta de um serviço por cada protocolo

Tal como o tempo de descoberta, o gasto energético é maior no protocolo Zeroconf, chegando perto dos 0,025 Joules de energia consumida, contra os 0,007 Joules do SLP.

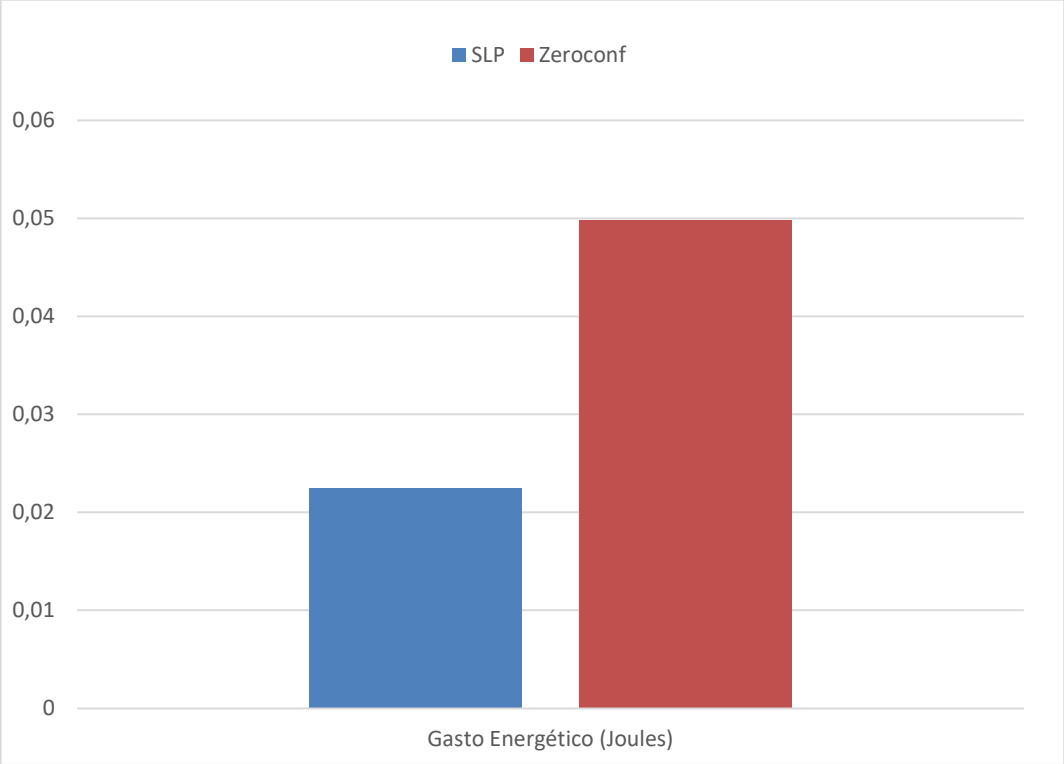

Figura 22 - Gasto de energia até à descoberta de 3 serviços por cada protocolo

Detalhes da configuração desta simulação:

- 5 nós Wi-Fi - um nó pesquisa serviços na rede, outros três disponibilizam serviços na rede;
- 5 nós Ethernet.

Mais uma vez, no segundo caso de simulação, o gasto energético é maior no protocolo Zeroconf, chegando perto dos 0,05 Joules de energia consumida.

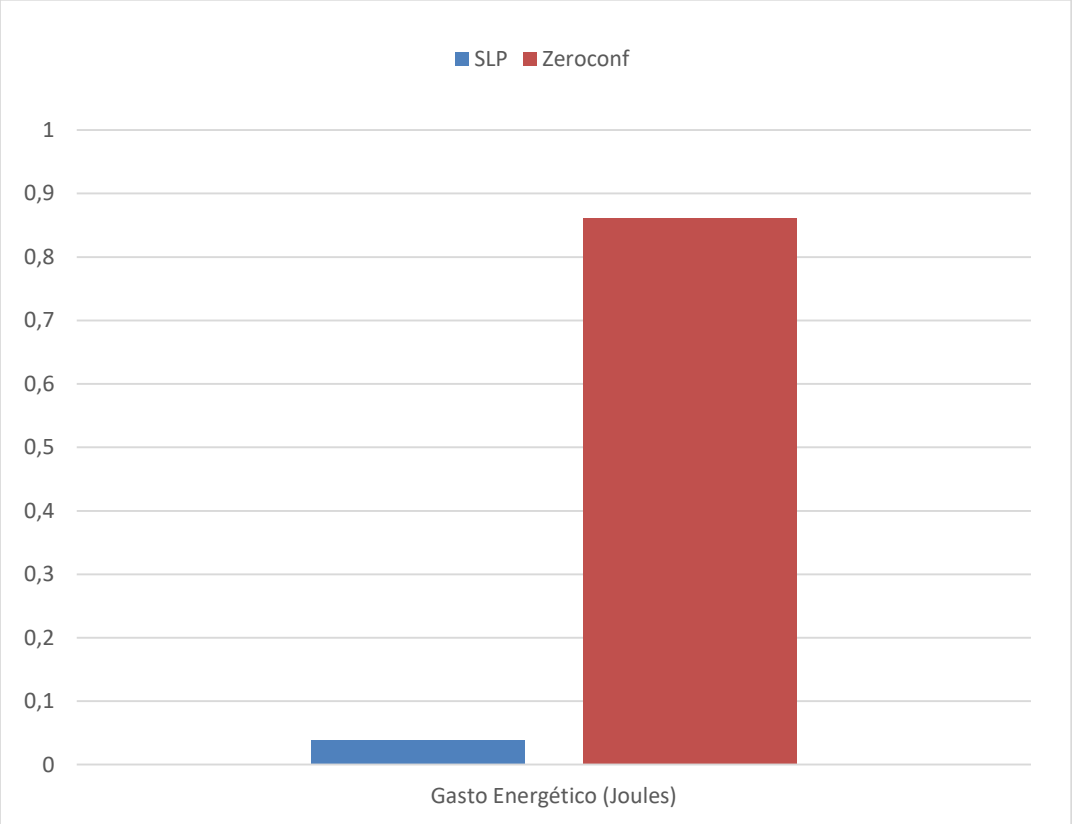

Figura 23 – Gasto de energia até à descoberta de 6 serviços por cada protocolo

Detalhes da configuração desta simulação:

- 10 nós Wi-Fi - um nó pesquisa serviços na rede, outros seis disponibilizam serviços na rede;
- 5 nós Ethernet.

Quando se duplica o número de serviços e nós na zona Wi-Fi da simulação anterior, a diferença no gasto energético é ainda mais acentuada, chegando, no caso do Zeroconf, aos 0,86 Joules de energia consumida. O SLP, porém, não teve um aumento significativo face à simulação anterior.

O aumento de energia entre casos de simulação é proporcional ao aumento do tempo de descoberta em cada protocolo. Mais uma vez, o Zeroconf tem aumentos exponenciais entre cada simulação, ao contrário do SLP.

Para finalizar, será criado um serviço na rede LAN de forma a ser descoberto por um nó da zona Wi-Fi.

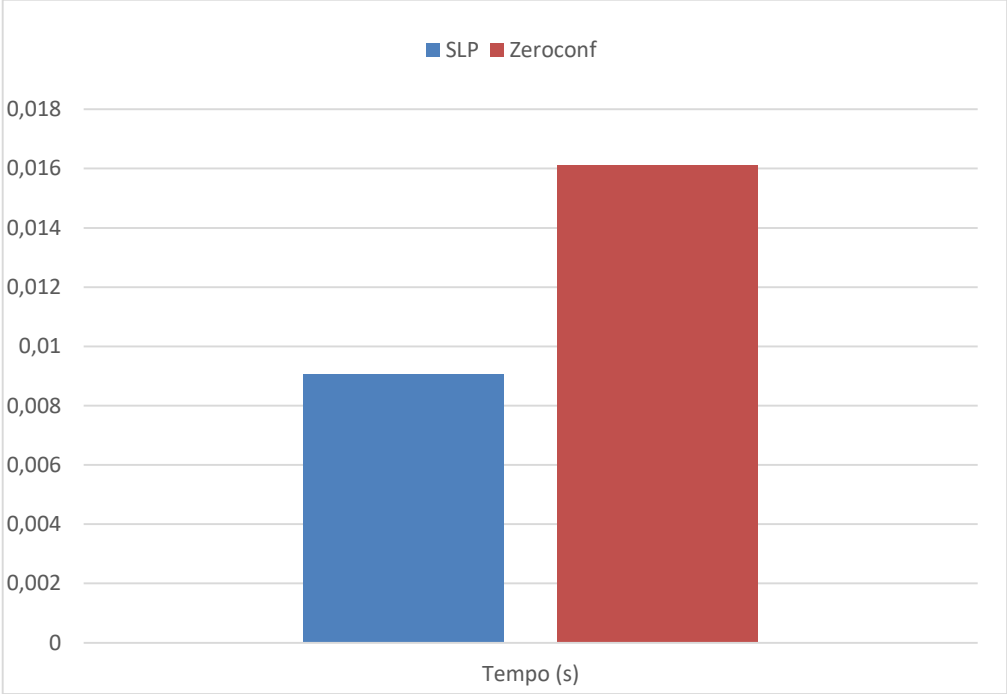

Figura 24 - Tempo total até à descoberta de um serviço na rede LAN

Detalhes da configuração desta simulação:

- 5 nós Wi-Fi - um nó pesquisa serviços na rede;
- 5 nós Ethernet – um nó disponibiliza um serviço na rede.

Neste último caso, temos um serviço disponível ligado à rede por cabo Ethernet. A diferença no tempo total da descoberta nos dois protocolos segue o mesmo padrão do que se já tinha visto nos casos anteriores. O protocolo Zeroconf leva perto do dobro do tempo para encontrar o serviço na rede em comparação com o SLP.

3.1.5 Validação

De forma a suportar a veracidade dos resultados obtidos através da simulação realizada, foram tidos em conta outros trabalhos idênticos na mesma área.

O trabalho de (Al-Mejibli & Colley, 2010) simulou e comparou cinco protocolos de descoberta entre os quais estão incluídos o SLP e o Zeroconf (referido como Bonjour).

Apesar da topologia da rede ser diferente e dos dados que apresentam estarem inteiramente relacionados com outros fatores, tais como os tamanhos das mensagens enviadas em cada pedido de descoberta, consegue-se perceber que os valores obtidos para o tempo que leva a que todos os serviços na rede sejam descobertos são bastante idênticos aos obtidos na simulação deste trabalho.

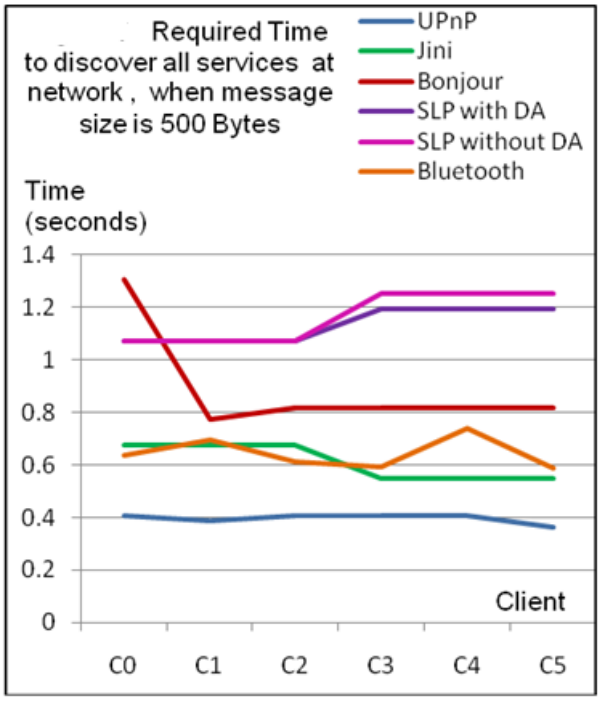

Figura 25 – Gráfico retirado de uma simulação de outro trabalho (Al-Mejibli & Colley, 2010)

A Figura 25 foi retirada do trabalho referido. Nela verifica-se que o tempo total para a descoberta de todos os serviços na rede ronda os 0,3 e 1,3 segundos. Partindo do pressuposto que a implementação de cada protocolo deverá ter sido mais complexa e que existem outras variáveis na simulação, como o tamanho da mensagem, é possível dizer que os valores são idênticos aos obtidos na simulação realizada neste trabalho.

3.1.6 Conclusões

Com base nas diferentes configurações para cada simulação, é possível dizer que o protocolo SLP foi o que mais vantagens demonstrou, não só pelo menor tempo de descoberta dos serviços na rede, como também pelo menor gasto energético, que está inteiramente relacionado com o tempo que é necessário para a dita descoberta chegar ao fim.

A diferença entre o SLP e o Zeroconf nesta simulação chega a ser significativa tendo em conta as características que normalmente os objetos inteligentes possuem: baixo processamento e baterias com pouca duração. A razão para esta diferença entre valores deve-se ao fato de na simulação, o mDNS, presente no Zeroconf, efetuar mais pedidos para a obtenção da informação necessária dos serviços existentes. O SLP é mais eficiente nesse aspeto já que não requer tantas trocas de mensagens.

Contudo, é importante voltar a referir que a implementação dos protocolos foi muito básica e que os resultados obtidos seriam certamente diferentes caso esses protocolos fossem totalmente implementados com todas as funcionalidades que os caracterizam.

Para dar resposta ao problema deste trabalho, tendo em conta a simulação realizada, o SLP seria o protocolo a utilizar pelos dispositivos móveis na descoberta de objetos inteligentes numa rede.

4 Conclusão

Neste trabalho foram abordados os mecanismos existentes que permitem a descoberta de serviços de objetos inteligentes a partir de dispositivos móveis. Foram apresentados os protocolos que permitiam essa descoberta e foram avaliados através de fatores relevantes para uma implementação nesse conjunto de dispositivos. Dessa avaliação resultaram dois protocolos que foram implementados de forma básica com o intuito de serem simulados para obter dados de análise. Feita e validada a análise, pode-se afirmar que os objetivos a que este trabalho se propôs foram cumpridos.

A simulação e avaliação de mais protocolos assim como a implementação mais aprofundada dos mesmos e utilizando diferentes simuladores são fatores que deveriam ser tidos em conta num trabalho futuro sobre o tema.

Referências

Al-Mejibli, I., & Colley, M. (2010). *Evaluating Transmission Time of Service Discovery Protocols by using NS2 Simulator.* Essex, UK.

Baccelli, E., Hahm, O., Gunes, M., Wahlisch, M., & Schmidt, T. (2013). Operating Systems for the IoT – Goals, Challenges, and Solutions. 6.

Beare, B. (2016). *Brillo/Weave Part 1: High Level Introduction.* Open IoT Summit.

bluetooth.com. (s.d.). Obtido de https://www.bluetooth.com/

Farooq, M. O., & Kunz, T. (2011). Operating Systems for Wireless Sensor Networks: A Survey. *Sensors*, 31.

Mallikarjuna Reddy V, A. (s.d.). *Wireless Sensor Network Image*. Obtido de https://commons.wikimedia.org/wiki/File%3AWSN.svg

McWilliams, A. (s.d.). *5 Minute Overview - What is iBeacon?* Obtido de ThoughtWorks: https://www.thoughtworks.com/insights/blog/what-is-ibeacon-in-5-minutes

Selby, Z. (s.d.). CoAP: The Web of Things Protocol.

Shelby, Z., & Bornmann, C. (2009). *6LoWPAN: The Wireless Embedded Internet.* John Wiley and Sons, Ltd.

Shelby, Z., Huuskonen, P., Mahonen, P., Riihijarvi, J., & Raivio, O. (2003). NanoIP: The Zen of Embedded Networking. 5.

Suthers, E. (s.d.). *Wi-Fi Alliance® introduces low power, long range Wi-Fi HaLow™*. Obtido de wi-fi: http://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-low-power-long-range-wi-fi-halow

Svec, C. (2012). FreeRTOS. Em *The Architecture of Open Source Applications.*

Vasseur, J.-P., & Dunkels, A. (2010). *Interconnecting Smart Objects With IP: The next Internet.* Morgan Kaufmann Publishers / Elsevier.

Why The New 802.11ah Wi-Fi Standard Will Give Z-Wave A Run For It's Money. (13 de 04 de 2016). Obtido de https://planetechusa.com/blog/802-11ah-wi-fi-halow-vs-z-wave/