FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Estudo de uma Plataforma Aberta para Comunicações M2M

Daniel de Oliveira Reis

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Doutora Ana Cristina Costa Aguiar (Prof. Auxiliar Convidada da FEUP)

18 de Julho de 2012

Estudo de uma Plataforma Aberta para Comunicações M2M

Daniel de Oliveira Reis

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Doutor Pedro Alexandre Guimarães Lobo Ferreira do Souto, Prof. Associado da FEUP

Arguente: Doutor José Maria Amaral Fernandes, Prof. Auxiliar da Universidade de Aveiro

Vogal: Doutora Ana Cristina Costa Aguiar, Prof. Auxiliar Convidada da FEUP

---

18 de Julho de 2012

Resumo

Atualmente assiste-se a um crescimento exponencial nas áreas de redes de comunicação, tanto pela sua amplitude e disseminação como também pelas inúmeras aplicações que surgiram aliadas a novas formas de comunicar. O que acontece hoje em dia é que a Internet - como rede de comunicação global que é - permitiu o desenvolvimento de tecnologias inovadoras que possibilitam a interação entre pessoas de variadas formas em vários ambientes, seja para fins lúdicos seja para ambientes colaborativos empresariais em que os vários intervenientes do processo comunicacional necessitam de estar conetados entre si, acedendo a informação atualizada disponibilizada na rede.

Nos próximos anos irá observar-se uma mudança de paradigma, que se traduzirá num número crescente de dispositivos que também irão estar conetados em rede e irão partilhar informação entre si disponibilizando-a de forma normalizada na rede. A partir dessa informação será possível o desenvolvimento de novos serviços que irão atuar em benefício da humanidade, seja em áreas como a prestação de cuidados de saúde ou em áreas como a telemática veícular.

O conceito de *Internet of Things* reflete claramente a intenção de não só pessoas como também dispositivos estarem conetados entre si em prol de um objetivo comum. As comunicações M2M apresentam-se como um meio para atingir esta finalidade.

Nesta disertação realiza-se um estudo de tecnologias e protocolos abertos e interoperáveis que podem ser utilizados em comunicações M2M e definem-se funcionalidades que devem ser implementadas por uma plataforma aberta e interoperável desenvolvida no âmbito desta dissertação. Definiram-se também métricas para a realização de uma comparação entre a plataforma desenvolvida no âmbito deste trabalho e outra já existente com base noutro protocolo de comunicação (XMPP).

Nesta dissertação conclui-se que nem sempre faz sentido a utilização de sistemas orientados a notificações (por exemplo através da adoção de protocolos como o AMQP e XMPP) dado a necessidade de estabelecimento de conexão ao nível da camada de aplicação sempre que a ligação à Internet é interrompida e em determinados cenários nos quais a publicação de dados realiza-se de forma periódica. Nestes casos é suficiente a utilização de um sistema ReST que apresenta-se como um estilo simples, de fácil compreensão e *stateless*. De qualquer forma, um serviço ReST pode representar o sistema de informação acerca de dispositivos a partir do qual se podem obter informações acerca dos seus recursos bem como informação para a realização de subscrições de recursos utilizando vários protocolos distintos.

Abstract

Nowadays we assist to an exponential growth of communication networks in terms of its wide and dissemination as well as numerous applications that emerged combined with new ways of communicate. The Internet - as a global communication network - allow the development on new technologies and enable new ways of communication which people use in a wide range of scenarios.

In the next few years there will be a paradigm shift that can be translated in a crescent number of devices connected in the network that will share information among them and dispose it in the network in a standard way. From this available information, it will be possible the development of new kinds of services acting in benefit of mankind in a wide range of vertical markets as for instance, healthcare or vehicular telematics.

The Internet of Things (IoT) concept clearly presents the purpose of having humans and also devices conected between them in order to achieve a common objective. Machine-to-Machine (M2M) communications are the means to reach the goals proposed by IoT.

This dissertation presents a study of involved technologies and protocols which can be used to perform M2M communications as well as the definition of basic funtionalities that must be implemented by the developed platform in an open and interoperable way. In addition, some metrics were defined to enable the comparison between the developed platform and another platform which relies in XMPP as protocol to enable communication between all the entities.

This dissertation concludes that not always make sense to use event driven architectures such as XMPP or AMQP given the need to establish connections at the application layer when the Internet connection fails and in some scenarios in which the publishing of data is done periodically or with less frequency. In these cases it is enough to use systems based on ReST style presented as a simple and understandable way of communicate besides of being stateless.

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

M2M	Machine-to-Machine
IoT	Internet of Things
SOAP	Simple Object Access Protocol
WSDL	Web Service Definition Language
HTTP	Hypertext Transfer Protocol
XML	Extensible Markup Language
WS	Web Service
SMTP	Simple Mail Transfer Protocol
FTP	File Transfer Protocol
ReST	Representational State Transfer
www	World Wide Web
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
TCP	Transmission Control Protocol
WADL	Web Application Description Language
MOM	Message-oriented middleware
XMPP	Extensible Messaging and Presence Protocol
STOMP	Streaming Text Oriented Messaging Protocol
DNS	Domain Name System
JID	Jabber ID
SMS	Short Message Service

Capítulo 1

Introdução

Atualmente vivemos numa sociedade onde a informação desempenha um papel fundamental na vida das pessoas. A utilização a nível mundial de redes de informação como a Internet, onde a informação é maioritariamente gerada por pessoas e para pessoas, desempenhou nos últimos anos um papel fundamental a vários níveis da sociedade. A quantidade de informação atualmente disponível para consulta em redes de informação é variada e engloba diferentes áreas da sociedade. O paradigma da Internet está prestes a sofrer uma alteração sem precedentes. Estima-se que num futuro próximo, dispositivos inteligentes farão também parte da Internet. Mais que isso, estima-se que o número de comunicações e informação associada gerada por dispositivos que comunicam entre si e com o meio em que se encontram inseridos, ultrapasse a quantidade de informação gerada pelas pessoas hoje em dia. Assim, surgiu a necessidade de desenvolver novas formas de comunicação que tenham em conta várias restrições associadas a novos dispositivos inteligentes, como é o caso da sua capacidade energética [SA11].

Nesta dissertação é abordado em pormenor algumas visões da "Internet do Futuro", com especial atenção à *Internet of Things* sobre a qual surgiram vários projetos e realizaram-se vários estudos que visam ultrapassar alguns obstáculos inerentes a esta nova imagem da Internet que irá surgir dentro de poucos anos. Existem no entanto, outros obstáculos inerentes a esta visão, sejam estes a falta de standards que forneçam um meio para uma ampla implementação de novos serviços, sejam as empresas que utilizam tecnologias proprietárias que não se encontram ao alcançe de todos. Estes serviços proprietários presentes em vários mercados verticais (automóveis e cuidados de saúde, por exemplo) representam um sério entrave ao desenvolvimento de novos standards e tecnologias abertas que permitam o desenvolvimento de serviços que estejam disponíveis à escala global, para que todos os possam utilizar e melhorar, permitindo assim um desenvolvimento da tecnologia que irá ao encontro das necessidades da população e não ao encontro dos objetivos financeiros de vários grupos empresariais.

As comunicações *Machine-to-Machine* representam um meio para se atingir a visão enunciada, uma vez que são estas comunicações que permitem aos dispositivos estarem acessíveis na rede para poderem ser utilizados por um vasto número de pessoas e outros dispositivos, na criação de mais e melhores serviços de informação que sirvam as sociedades a vários níveis. Esta

dissertação insere-se neste contexto, dando o seu contributo a partir de um estudo sobre algumas tecnologias e tendências que permitam construir um serviço que comunique com um dispositivo inteligente, baseado em standards abertos, permitindo que possa ser facilmente utilizado e reutilizado por outros serviços, dispositivos e pessoas. A utilização de standards abertos é um meio necessário para que a tecnologia possa ser adotada globalmente. Outro fator a ter em conta quando se trata do desenvolvimento de novos serviços, está relacionado com as capacidades físicas dos dispositivos inteligentes. A nanotecnologia tem dado passos muito importantes, particularmente na criação de novos materiais que permitirão alcançar novos patamares de interação com dispositivos que estarão ligados à rede em permanência, uma vez que estes serão cada vez mais robustos, mais compactos e energeticamente mais eficientes.

A falta de standards e tecnologias associadas em comunicações *Machine-to-Machine* representam um problema base na procura de soluções arquitetónicas para este tipo específico de comunicações. Existe também a falta de uma comparação objetiva entre várias soluções candidatas a resolver os problemas associados a estas comunicações. Este trabalho foca-se também neste último ponto, tentando contribuir com uma comparação objetiva entre soluções abertas e interoperáveis fundamentalmente diferentes entre si.

1.1 Contexto / Enquadramento

Como vimos na secção anterior, esta dissertação insere-se no âmbito de uma *Internet of Things*. Em concreto, primeiramente é realizado um estudo acerca de algumas tecnologias fundamentais para o desenvolvimento de uma solução que permita a interação com um dispositivo inteligente. Depois será desenvolvida uma plataforma que tenha em conta o maior número possível de caraterísticas associadas à visão enunciada. Após o desenvolvimento desta plataforma, esta será comparada com outra plataforma proprietária que incide sobre o mesmo cenário na realização de funcionalidades comuns, tornando possível a realização de um estudo comparativo entre essas soluções a vários níveis, quer seja através de métricas definidas (por exemplo, número de mensagens trocadas), quer seja através de outras características fundamentais para a aceitação e proliferação de tais soluções pela rede de informação (por exemplo, o esforço necessário para desenvolver determinado serviço).

O presente trabalho incide sobre um cenário genérico de cuidados de saúde, no qual um utilizador possuirá um sensor de batimentos cardíacos ligado ao seu *smartphone* que utilizará a Internet para publicar os dados relativos ao seu batimento cardíaco. Esta informação poderá ser consumida por vários atores no sistema, sejam estes fornecedores de serviços de cuidados de saúde ou familiares autorizados a receberem as notificações de tal utilizador. É um requisito obrigatório, que tal solução seja unicamente baseada em standards abertos e que possua um mecanismo de notificações eficiente. Este mecanismo surge aliado aos inúmeros cenários onde será possível interagir com um dispositivo inteligente que por sua vez realize uma qualquer tarefa ou ação na ocorrência de um determinado evento. Nesta dissertação não serão discutidos problemas relacionados com a autenticação e autorização de serviços.

O surgimento de serviços *web* que utilizam tecnologias SOAP e WSDL entre outras, desempenharam um papel fundamental no desenvolvimento de soluções interoperáveis que permitiram a integração de novos serviços com um reduzido custo assim como a comunicação entre aplicações diferentes concebidas para plataformas também diferentes. Uma desvantagem desta abordagem está presente quando se pretende desenvolver um serviço muito simples. Nestes casos, a solução apresenta-se como demasiado "pesada"e por vezes complexa.

Mais recentemente, Roy Thomas Fielding na sua tese de doutoramento (no ano 2000), sugeriu uma arquitetura à qual lhe chamou ReST. Esta arquitetura possui várias vantagens em relação a serviços tradicionais, pois permite disponibilizar os recursos fornecidos por dispositivos inteligentes na rede (através de hiperligações) de forma flexível, utilizando standards abertos e protocolos amplamente utilizados no âmbito da Internet. Aliada a esta inovação, surgiram também nos últimos anos vários desenvolvimentos em áreas como "*Middleware* orientado a mensagens", que permitem estabelecer comunicações e notificar eventos entre dispositivos diferentes que se encontram ligados em rede. Estas tecnologias associadas a modelos *Publish-Subscribe* representam também um importante passo no contexto das comunicações eficientes entre máquinas. Alguns exemplos destas tecnologias são o caso do AMQP e do XMPP que se apresentam como duas soluções diferentes que podem ser aplicadas ao cenário desta dissertação ou seja, à comunicação entre dispositivos diferentes num cenário de monitorização de batimentos cardíacos.

1.2 Estrutura da dissertação

Para além da introdução, esta dissertação contém mais 7 capítulos.

No Capítulo 2, são descritas várias tecnologias que podem ser utilizadas para a implementação de uma solução para o problema proposto, contendo no final do capítulo uma discussão acerca da aplicabilidade das mesmas. A Secção 2.6 apresenta algumas arquiteturas que têm vindo a ser propostas por alguns grupos cujo objetivo é a especificação de standards na área de comunicações M2M.

No Capítulo 3, apresenta-se uma descrição mais pormenorizada do problema a resolver, no qual se insere esta dissertação.

O Capítulo 4 é referente às metodologias utilizadas para a realização deste trabalho. Nele se definem as funcionalidade base a implementar numa solução assim como as métricas que permitem realizar uma comparação entre a plataforma desenvolvida e outra proprietária já existente. Serão também descritos os processos utilizados nas comparações efetuadas.

No Capítulo 5 é realizada uma primeira comparação entre protocolos para a observação de algumas caraterísticas que os distinguem na realização de algumas funcionalidades.

O Capítulo 6 descreve a plataforma desenvolvida no âmbito deste trabalho, estando presentes várias decisões de implementação assim como uma descrição pormenorizada dos módulos que constituem a plataforma desenvolvida.

A análise experimental diz respeito ao Capítulo 7 onde são comparadas as duas plataformas com recurso a medição de dados extraídos fruto da utilização das mesmas.

Por fim, no Capítulo 8 serão apresentadas as conclusões retiradas da elaboração desta dissertação e serão mencionadas as contribuições inerentes ao trabalho desenvolvido.

Capítulo 2

Revisão bibliográfica

No decorrer deste capítulo serão abordados com maior profundidade os conceitos e tecnologias referenciadas no Capítulo 1. Este capítulo representa a base epistemológica necessária para a compreensão do trabalho que foi desenvolvido durante esta dissertação para que seja possível optar informadamente entre as várias soluções possíveis o conjunto de protocolos necessários para o desenvolvimento da plataforma. A secção seguinte introduz alguns conceitos essenciais acerca do contexto no qual se insere a plataforma desenvolvida. Após a apresentação destes conceitos, segue-se uma apresentação das caraterísticas fundamentais de algumas tecnologias que podem ser utilizadas no contexto proposto.

2.1 IoT e M2M

> "And men got dreaming. Shouldn't there be a network that made all my devices collaborate at all times, converse spontaneously among themselves and with the rest of the world, and all together make up a kind of single virtual computer – the sum of their respective intelligence, knowledge and know how?" [Sun10]

A citação anterior pertence ao autor Rafi Haladjian decorria o ano 2005. Ao pensarmos não só no futuro das redes de comunicação mas também no futuro da sociedade, é quase impossível não pensar que existirão vários milhões de dispositivos inteligentes conetados entre si através de uma rede de comunicação com o objetivo de fornecerem informação útil e valiosa em áreas distintas como por exemplo, a saúde, transportes, gestão de tempo e automatização de processos. Quanto mais se pensa em possíveis aplicações destes dispositivos num amplo leque de cenários, mais rapidamente se conclui que este tipo de comunicações irão realmente fazer parte do futuro, permitindo a criação de serviços inovadores e inteligentes com benefícios para toda a humanidade. Além disso, recentes avanços em áreas como a nanotecnologia e arquiteturas de redes de comunicação tornam o "sonho" mais fácil de alcançar. A nanotecnologia poderá permitir a construção de uma nova Internet de alta velocidade além de permitir o surgimento de novos sensores consideravelmente mais pequenos e com capacidades de processamento muito superiores aos que existem atualmente. Apesar de já existirem vários trabalhos e projetos publicados sobre estes temas, ainda

existem alguns obstáculos para ultrapassar inerentes à visão enunciada por Haladjian e que serão apresentados nesta secção.

A visão da União Internacional de Telecomunicações (ITU) acerca de uma *Internet of Things* (IoT) é bastante clara: *"From anytime, anyplace connectivity for anyone, we will now have connectivity for anything"* [Uni05].

Segundo esta visão acerca de IoT, espera-se que dispositivos inteligentes possam tornar-se participantes ativos em processos de negócio ou de informação, onde podem interagir e comunicar entre eles e com o ambiente em que se encontram reagindo autonomamente a eventos do mundo real, influenciando-o através de processos que desencadeam ações e criam serviços com ou sem intervenção humana [GSB]. Estes serviços por sua vez, podem interagir com os dispositivos inteligentes associados, utilizando interfaces standardizadas que fornecem as ligações necessárias via Internet para que dessa forma seja possível interagir com dispositivos, tanto para recolher informações destes, como para atuar sobre os mesmos, tendo em conta alguns obstáculos de segurança e privacidade existentes [SGFW10].

Está previsto que por volta do ano 2020 existam cerca de 16 biliões de dispositivos inteligentes conetados. Para se ter uma noção do que este valor representa, alguns estudos apontam que por volta do mesmo ano existam no mundo cerca de 8 biliões de pessoas concluindo-se assim que num futuro próximo a Internet, que agora está completamente voltada para as pessoas, irá possuir mais dispositivos a comunicar na rede do que propriamente seres humanos. Existirá sem sombra de dúvida, uma mudança de paradigma acerca do conceito de Internet e das suas aplicações [Nyg94]. Uma das consequências diretas desta mudança de paradigma será a enorme quantidade de dados gerada por todo o tipo de dispositivos que possuam tais capacidades de comunicação e interação. Se considerarmos por exemplo, que num futuro próximo existirão cerca de meio milhão de automóveis e que cada um destes possui cerca de 100 sensores (que por sua vez enviem uma mensagem de 16 bytes a cada segundo), serão gerados cerca de 6.4 Tbps de informação apenas no mercado automóvel. Toda esta informação gerada não possui qualquer valor a não ser que possa ser facilmente acedida, analizada e interpretada por dispositivos inteligentes e seres humanos. Assim será possível combinar dados de diferentes dispositivos e processá-los em informação com valor adicional em benefício da sociedade em geral [Uni05].

Um conceito fortemente ligado à visão IoT é o de *Machine-to-Machine*, que se apresenta como um subconjunto de funcionalidades que compreendem a IoT, funcionando quase como um meio para atingir um fim, que neste caso será a implementação da visão IoT. Comunicações M2M inseridas no contexto IoT têm sido um tema sujeito a uma intensa discussão nos últimos anos. A siga M2M possui um vasto leque que semânticas, nomeadamente, *Man-to-Machine*, *Machine-to-Man*, *Machine-to-Mobile*, *Mobile-to-Machine* e ainda *Machine-to-Machine*, sendo esta última o significado geralmente mais utilizado. Nesta dissertação, M2M é utilizado segundo o conceito de *Machine-to-Machine*.

M2M é um termo utilizado para descrever tecnologias que permitam aos computadores, sensores inteligentes, atuadores e dispositivos móveis comunicarem entre si, extrair informação do ambiente que os circunda e tomarem decisões sobre o ambiente em que estão inseridos, mui-

tas vezes sem que para isso seja necessária a intervenção humana. O sucesso da IoT depende profundamente da evolução das comunicações M2M ao nível da conetividade entre dispositivos inteligentes e a rede de informação que os serve.

O crescimento exponencial das infra-estruturas de comunicação, assim como a proliferação de dispositivos inteligentes levou a que surgisse uma necessidade real de adaptação das aplicações existentes em alguns mercados verticais para que estas possam utilizar novas tecnologias que gerem mais valias para todos os intervenientes do processo, tanto máquinas como pessoas. As portas estão abertas para uma revolução na forma como comunicamos com as máquinas e na forma como novos dispositivos inteligentes podem fazer a diferença no dia-a-dia de uma qualquer pessoa. Estes desenvolvimentos tecnológicos na área M2M originaram uma força capaz de alterar determinados setores de mercado, especialmente nas aplicações relacionadas com a monitorização em tempo real, como em áreas de cuidados de saúde, domótica, monitorização do meio ambiente e automação industrial [LLLS11]. O desenvolvimento de novas tecnologias M2M e a sua posterior utilização em grande escala levanta vários desafios ao nível da eficiência energética e de outros componentes da qualidade do serviço, como a fiabilidade e segurança.

Os sistemas M2M do futuro serão complexos e encontrar-se-ão presentes em muitos segmentos de mercado, incluindo as telecomunicações e componentes eletrónicos. Para atingir um crescimento exponencial e ao contrário do que acontece com os mercados M2M atuais (demasiado segmentados e normalmente baseados em soluções proprietárias), os mercados M2M do futuro terão de ser necessariamente baseados em standards da indústria. Estes standards serão mais abrangentes do que simples especificações, uma vez que compreendem não só interfaces, mas também plataformas e serviços. A indústria M2M necessita de ultrapassar as soluções existentes em mercados verticais e desenvolver plataformas que horizontalizem os mesmos, assim como desenvolver tecnologias que otimizem a gestão dos dispositivos, arquiteturas de redes e segurança que permitam a massificação de dispositivos inteligentes ligados à rede [WTJ+11].

Uma vez apresentada a forma como IoT e comunicações M2M estão relacionadas resta também referir algumas das suas aplicações típicas no mundo real com recurso a alguns exemplos em diferentes áreas onde é importante inovar e evoluir para a obtenção de mais valias com um rápido retorno do investimento. Em áreas como a segurança, as comunicações M2M permitirão o desenvolvimento de sistemas de vigilância e o controlo de acesso em espaços físicos. No contexto da preservação ambiental, recorrendo a comunicações M2M será possível obter mais valias na monitorização do meio ambiente. Melhorias significativas nos processos de automatização e monitorização de redes de gás, eletricidade e aquecimento permitirão uma redução drástica dos consumos de energia. Na área da telemática veicular, poder-se-á realizar uma gestão geo-espacial dos veículos e mercadorias, implementar melhorias nos sistemas de apoio à navegação, obtenção de informações de tráfego mais precisas e ainda diagnosticar remotamente veículos. Na área da saúde, a monitorização à distância de sinais vitais, uma melhoria significativa nos sistemas de suporte de vida e suporte a pessoas idosas ou portadores de deficiências, telemedicina e mesmo diagnósticos remotos serão uma realidade ao alcance de muitos. Em suma, as comunicações M2M posuem inúmeras aplicações num vasto leque de áreas, desde as previamente referidas até às sim-

ples máquinas de *vending* que poderão por exemplo, notificar o fornecedor para a rutura de stock de um determinado produto.

M2M apresenta portanto várias oportunidades e alguns desafios que precisam de ser ultrapassados para que se consiga alcançar uma real visão de IoT. Embora existam motivações económicas e industriais significantes para que se aposte no desenvolvimento de novos standards abertos e de novas tecnologias associadas a arquiteturas de serviços, os mercados altamente fragmentados representam um obstáculo e um sério risco para o crescimento previsto de mercados M2M.

2.2 WS-*

Nos últimos anos verificou-se uma utilização em larga escala de tecnologias que permitem desenvolver serviços cujas funcionalidades servem certos aspectos enunciados pela visão IoT e comunicações M2M associadas. Estas tecnologias denominam-se por *Web Services*[1] (WS). Ainda não existe um amplo consenso acerca da definição de Serviço *Web*, no entanto, de seguida são apresentadas duas definições que servem os propósitos desta secção.

> *"A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to interprocess communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards"* [Poi]

> *"A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."* [BHM+04]

Na primeira citação apresentada, o autor define um serviço *web* como uma coleção de protocolos abertos utilizados para trocar dados entre várias aplicações comunicantes, destacando ainda a interoperabilidade dessas aplicações fruto da utilização destes protocolos abertos amplamente utilizados em diversas aplicações.

Por sua vez, o autor da segunda citação acima transcrita fornece uma descrição mais específica acerca do conceito de serviços *web*. Segundo este autor, este conceito surge associado a um sistema desenhado para suportar comunicações M2M através da rede, no qual são utilizadas interfaces (WSDL) em formatos facilmente interpretados pelas máquinas. Outros sistemas que desejem interagir com o serviço devem fazê-lo respeitando as descrições fornecidas pelo documento WSDL com recurso a mensagens SOAP transportadas normalmente por HTTP e estruturadas em forma de documentos XML.

Um WS é na sua essência, uma solução que permite a comunicação entre aplicações diferentes. Recorrendo a esta tecnologia, é possível que novas aplicações possam ser facilmente integradas com outras previamente existentes, ultrapassando os tradicionais problemas de interoperabilidade existentes quando as aplicações comunicantes encontram-se desenvolvidas para diferentes plataformas e/ou em diferentes linguagens de programação. A ubiquidade da Internet, indiferente a fronteiras empresariais ou tecnológicas, aliada à adoção de forma generalizada de linguagens para

[1] Ao longo desta dissertação é possível que apareçam várias referências a *Web Services* como Serviços *Web* ou simplesmente serviços

estruturação de documentos (como o é o caso da *eXtensible Markup Language* vulgarmente conhecido por XML), permitiu o desenvolvimento e aceitação generalizada destes componentes de integração. Além disso, a importância da utilização de tecnologias standard, permite que esta tecnologia seja considerada atrativa em vários meios empresariais, nos quais um serviço deste tipo pode agilizar os processos comunicacionais entre os vários intervenientes inseridos nesse meio de forma dinâmica e segura, uma vez que não requer a intervençao humana no processo comunicacional. Dito de uma outra forma, os serviços web fazem com que os recursos de uma aplicação estejam disponíveis na rede de uma forma normalizada, o que permite que uma aplicação cliente desenvolvida numa qualquer linguagem para uma determinada plataforma, possa operar e extrair a informação fornecida pelo WS, independentemente da linguagem ou plataforma em que este último foi desenvolvido.

O conceito de serviços *Web* possui um conjunto de especificações adicionais que adicionam várias funcionalidades de diferentes naturezas, como especificações de segurança para tornar o canal de comunicação seguro ou especificações relacionadas com a programação orientada a eventos, o que permite que um serviço possa facilmente subscrever outro serviço e vice-versa. Estas especificações são muitas vezes referidas como WS-* e desempenharam um papel fundamental durante os últimos anos tendo dado o seu contributo na crescente necessidade de distribuir aplicações pela *World Wide Web*, quer para o desenvolvimento de novas aplicações, quer para a reutilização de aplicações já existentes e criação de mais valias.

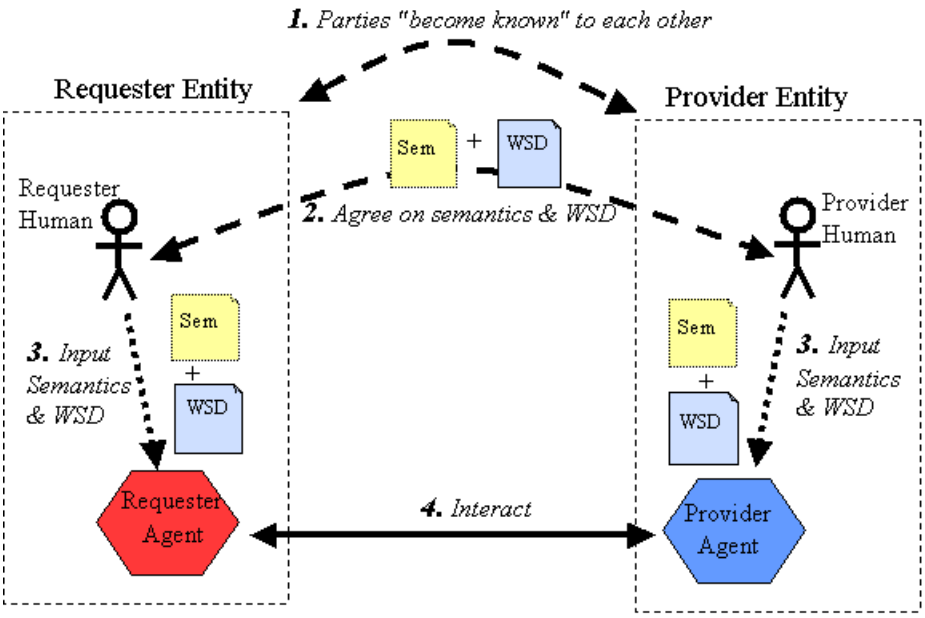


Figura 2.1: Funcionamento geral de serviços web [BHM+04]

A Figura 2.1 ilustra o funcionamento geral de um WS e ajuda a relacionar alguns conceitos-chave desta tecnologia. Utilizando esta tecnologia, existem várias formas para estabelecer a comu-

nicação entre o fornecedor de serviços e o consumidor dos mesmos no entanto, esta comunicação é estabelecida recorrendo aos quatro passos principais apresentados na figura. Primeiro, as entidades comunicantes tomam conhecimento da existência da outra parte. De seguida, as duas entidades concordam em relação às semânticas e descrições do serviço que irão utilizar durante o processo comunicacional e passam essa informação aos respetivos agentes. São estes agentes que finalmente realizam a trocam mensagens entre si e processam as tarefas específicas da entidade em que estão inseridos.

Em relação ao processo de desenvolvimento de um WS, podemos sumariar o processo da seguinte forma: Um programador constrói o seu serviço utilizando uma qualquer linguagem de programação e publica o serviço utilizando a interface *Web Service Definition Language* (WSDL). Esta interface define o ponto de contato para um fornecedor de serviços e possui também uma definição formal da interface do serviço (através de um conjunto de elementos comum) para que os seus clientes saibam como construir mensagens para comunicar com o serviço. Do ponto de vista do cliente, o serviço *web* é-lhe apresentado como um conjunto de operações que o cliente poderá invocar para obter os resultados pretendidos. Do lado do cliente gera-se um objeto remoto, o que permite aos clientes invocar as operações definidas do lado do servidor da mesma forma como se se tratasse de um objeto local.

Um dos fatores principais que permite a interoperabilidade de serviços *web* é o seu protocolo de mensagens amplamente utilizado e conhecido por SOAP (acrónimo de "*Simple Object Access Protocol*"). SOAP foi desenvolvido para que diferentes plataformas possam interoperar. Em vez de representar uma inovação tecnológica, SOAP apenas sugere uma forma para uniformizar comunicações distribuídas pela Internet. SOAP é normalmente transferido através de HTTP sendo esta uma forma eficiente para envio e receção destas mensagens (podem ser utilizados também outros protocolos como *Simple Mail Transfer Protocol* (SMTP) e o *File Transfer Protocol* (FTP). A função de SOAP é definir a formatação de uma mensagem e não como a mensagem é entregue. Este protocolo possui várias vantagens como a sua simplicidade, portabilidade, utilização de standards abertos, interoperabilidade e resistência a alterações. Alguns destes pontos são atingidos fruto da utilização do XML (documento bem estruturado e amplamente utilizado) e à utilização de HTTP que providencia a transmissão de mensagens sem que existam problemas relacionados com as *firewalls* uma vez que estas raramente monitorizam o porto 80 utilizado pelo protocolo HTTP.

Normalmente os WS estão registados num serviço diretório que funciona como ponto de informação para as aplicações clientes. Através deste diretório é possível que uma qualquer aplicação cliente encontre o serviço com o qual pretende comunicar. O *Universal Description Discovery and Integration* (UDDI) é um protocolo desenvolvido para a organização e registro de *Web Services* e fornece três funcionalidades principais, sendo estas a publicação (permite que uma organização publique os seus serviços), a descoberta (permite que o cliente do serviço possa procurar e encontrar um dado serviço) e finalmente a ligação, que permite ao cliente estabelecer uma ligação e interagir com o serviço desejado.

A utilização desta tecnologia oferece algumas vantagens, das quais é importante realçar a reusabilidade, uma vez que uma aplicação pode utilizar as funcionalidades de outras aplicações, a

interoperabilidade ao nível da plataforma e tecnologias, a utilização de standards abertos (garante uma redução nos custos e o aumento na qualidade do sistema) e ainda um processo de descoberta automático que permite uma fácil descoberta dos serviços e descrições associadas. Serviços *web* permitem a composição de aplicações através da invocação de serviços através da rede, o que permite uma utilização mais eficiente reduzindo assim a necessidade de criação de novas aplicações. Um conceito fortemente ligado a WS é o *loose coupling* o que permite que a aplicação cliente do serviço ignore completamente todos os detalhes técnicos e de implementação do serviço. Os serviços são invocados através de mensagens em vez de APIs ou formatos de ficheiros. Isto funciona graças à independência da interface em relação à componente da implementação.

Estes serviços são no presente, sujeitos a várias limitações que incluem uma baixa performance e um suporte insuficiente para semânticas empresariais. Além disso, existe uma clara falta de homogeneidade e coordenação em relação ao amplo conjunto de standards existentes e emergentes. O problema da utilização deste tipo de serviços é que no contexto M2M associado à visão IoT, no qual são esperados um número exorbitante de pacotes na rede, esta opção revela-se ser demasiado verbosa e complexa, especialmente quando se trata de objetos simples com os quais se deseja comunicar. Para driblar algumas destas desvantagens surgiu o conceito ReST que é apresentado na secção a seguir.

2.3 RESTful WS

Até agora, os projetos desenvolvidos sobre a alçada de IoT focaram-se essencialmente no estabelecimento da conetividade numa ampla variedade de ambientes e cenários. Um passo necessário e promissor passa agora pela construção de modelos de interação escaláveis construídos sobre a camada de rede, permitindo que os esforços sejam aplicados à camada de aplicação [DVFE].

Segundo o conceito *Web of Things* (WoT), dispositivos inteligentes e os seus serviços encontram-se integrados na *World Wide Web* reutilizando e adaptando tecnologias e padrões normalmente utilizados para conteúdo *web* tradicional [DVFE]. Ao invés da utilização da infra-estrutura da Internet apenas como mecanismo de transporte (como no caso de WS-*), os dispositivos devem constituir uma parte integrante da Internet, recorrendo às suas ferramentas e infra-estruturas através da utilização do protocolo HTTP como protocolo da camada de aplicação [DVFE].

Representative State Transfer (ReST) [Roy00] é um estilo de arquitetura que explica os detalhes técnicos responsáveis pelo enorme sucesso da *World Wide Web*, que por sua vez permitiu que dados e serviços sejam partilhados num crescimento descentralizado. Este estilo pode ser utilizado tanto por pessoas (por exemplo através de um *browser*) como por máquinas que o utilizem para trocar informação relevante. ReST é também menos complexo que outras abordagens (WS-*), sendo uma solução de fácil compreensão e implementação. O termo ReST foi apresentado na tese de doutoramento de Roy Fielding no ano de 2000. ReST não descreve protocolos específicos, formatos de dados ou sequências de interação, apenas apresenta os princípios de arquitetura e componentes que levaram ao sucesso e ampla utilização da *World Wide Web*.

2.3.1 Princípios

ReST é um estilo de arquitetura construído sobre determinados princípios básicos da Internet. Existem cinco princípios que devem ser utilizados para a criação de serviços ReST:

- Tudo são recursos: Um recurso é um qualquer componente de uma aplicação que valha a pena ser unicamente identificável e que possa ser transmitido na rede através das suas representações (binárias ou textuais). Em ReST o pensamento não está focado em ficheiros, mas sim em recursos;
- Todos os recursos são identificáveis através de um identificador único: Os identificadores são utilizados para permitir a fácil manipulação do recurso. Em ReST, utilizam-se *Uniform Resource Locators* (URLs) para identificar recursos;
- Utilização de uma interface simples e uniforme: ReST utiliza HTTP como protocolo base, sendo este um protocolo bem conhecido e aceite em todo o mundo;
- Comunicação efetuada através de representações: Cada recurso pode possuir várias representações diferentes de si próprio. Quando se realiza uma determinada operação sobre um recurso, o que acontece na realidade é uma troca de representações desse recurso;
- Interações *stateless*: Todas as interações realizadas sobre um recurso são independentes, isto é, o servidor não mantém informação acerca das interações realizadas.

ReST é o estilo de arquitetura da Web, implementado por URLs, HTTP, e vários tipos de dados standard. Um dos pontos chave deste estilo arquitetónico tem a ver com a utilização de URLs para identificação de recursos na *web*. Segundo este estilo, os serviços são abstraídos através de uma interface uniforme (HTTP e respetivos métodos) que fornece mecanismos para que as aplicações cliente possam escolher a melhor representação possível das respostas que recebem do serviço. Estes são alguns dos motivos que posicionam o estilo ReST no contexto IoT e consequentemente nas comunicações M2M para o desenvolvimento de uma arquitetura global e APIs para dispositivos inteligentes. O protocolo de comunicação base associado ao estilo ReST é, como referido anteriormente, o protocolo HTTP. A arquitetura base deste protocolo segue o modelo de comunicação cliente-servidor. Este modelo é amplamente utilizado na Internet e apresenta-se como um protocolo de fácil compreensão: um cliente que deseje consultar um determinado recurso envia ao servidor um pedido HTTP, ao qual o servidor responde enviando uma mensagem ao cliente com a respetiva resposta. Em relação ao formato da resposta enviada pelo servidor ao cliente e uma vez que os dispositivos inteligentes possuem recursos limitados, a informação pode ser enviada ao cliente tomando a forma de um documento estruturado XML ou ainda a forma de um documento JavaScript Object Notation (JSON). Por outro lado, caso um utilizador queira consultar informações relacionadas com um determinado recurso, é preferível obter essa informação por exemplo através de um *browser* que renderize a resposta dada pelo servidor na forma de uma página *web*. Uma vantagem inerente a estas várias representações reside na possibilidade de poderem ser interpretadas e processadas por máquinas e ainda por pessoas permitindo uma maior flexibilidade

e eficiência no processo comunicacional. Através de páginas *web* e de *tags* bem definidas nas representações é possível também documentar este tipo de serviços. Podem ser também fornecidas várias informações semânticas utilizando microformatos, que basicamente são um conjunto de formatos abertos construídos sobre standards existentes e bem conhecidos[2].

Dados dinâmicos acerca do mundo real podem ser apresentados em páginas *web* e depois processados com o auxílio de ferramentas *Web* 2.0. Por exemplo, os dispositivos podem ser indexados como páginas *web* através das suas representações para que os utilizadores possam aceder a estas páginas através de um *browser* com o auxílio de um motor de busca. Estes endereços podem também ser enviados para outros utilizadores (através de *email* por exemplo) além de também poderem ser adicionados aos favoritos do *browser* pelo utilizador final. A ideia base passa pela utilização da *web* como sistema de informação descentralizado para que seja fácil expôr novos serviços e aplicações, direta ou indiretamente por dispositivos inteligentes.

A ideia central de ReST está intimamente relacionada com a noção de recurso. São considerados recursos todos os componentes numa aplicação que valem a pena serem unicamente identificados. Na *web*, a identificação de recursos é realizada através de URLs. Para navegar entre os demais recursos são utilizadas ligações que permitem a interacção com recursos associados. Isto permite que os clientes explorem um serviço acedendo à representação por eles fornecida e através da qual estão presentes outros serviços associados ao mesmo.

Cada recurso possui um identificador global (URL em HTTP). Estes recursos são acedidos por componentes na rede que comunicam através de um protocolo (ex. HTTP) e trocam conteúdo (representações) desses recursos. Para interagir com um recurso, uma aplicação tem que possuir o identificador do recurso bem como o método que quer aplicar sobre o mesmo. Por outro lado, não é necessário conhecer a implementação e a configuração do sistema ou seja, não interessa saber se existem *proxies*, *firewalls* ou qualquer outro componente entre a aplicação e o servidor que aloja esses recursos. A aplicação deve ser capaz de interpretar os dados enviados (como resposta) pelo servidor, sejam eles texto simples, XML, imagens, documentos, etc. Para isso, é possível a um qualquer cliente especificar no pedido qual o formato da representação que deseja receber por parte do servidor.

De forma simples, o primeiro passo para permitir ligar dispositivos inteligentes na *web* passa por criar a rede de recursos. Dois aspetos centrais são o identificador de um recurso e as suas relações com outros recursos.

Recursos são entidades abstratas e não possuem apenas uma única representação de si mesmos, mas sim várias representações em formatos diferentes. Na *web*, o suporte de vários tipos de dados fornecidos por HTTP e HTML permite que os clientes navegem pelos recursos com a utilização de hiperligações. Para uma comunicação M2M, outros tipos de dados como XML e JSON têm sido amplamente utilizados para se atingir a interoperabilidade necessária a uma arquitetura global de serviços.

[2]Mais informações acerca de microformatos em http://www.microformats.org/about

Em ResT, a interacção com os recursos e a receção das representações por parte dos clientes acontece através de uma interface uniforme que especifica um contrato de serviço entre clientes e servidores. Existem 3 partes distintas e fundamentais para que se realize esta interacção: operações, negociação de conteúdo e estado da resposta.

Quanto às operações, ReST utiliza quatro métodos principais do protocolo HTTP para permitir a interação com recursos, sendo estes muitas vezes conhecidos como verbos HTTP: GET, PUT, POST e DELETE. O método GET é utilizado para obtenção de representações de recursos. O verbo PUT serve para atualizar o estado de um recurso ou para criar um recurso utilizando um identificador. O método POST cria um novo recurso sem ser necessário especificar o identificador. Por fim, o método DELETE serve para remover um recurso. Um último verbo porém menos conhecido é o OPTIONS que pode ser utilizado para devolver ao cliente as operações permitidas por um recurso (por exemplo operações GET e PUT).

Em relação à negociação de conteúdo, isto é, ao mecanismo que permite aos clientes e servidores comunicarem para chegarem a acordo sobre qual a representação a devolver pelo servidor, esta negociação está embutida na interface HTTP subjacente à arquitetura ReST.

No que diz respeito ao estado da resposta, são utilizados os códigos standard do protocolo HTTP. Existem vários códigos disponíveis (200 representa uma operação efetuada com sucesso enquanto que o 405 significa que o método utilizado não é permitido pelo recurso referenciado pelo URL).

ReST não descreve protocolos específicos, formatos de dados ou sequências para interação. ReST descreve sim, os princípios arquiteturais e componentes que permitem que a *World Wide Web* funcione com sucesso. A flexibilidade e facilidade de compreensão deste estilo, torna-o um estilo atrativo e de ampla utilização a nível mundial nos mais variados contextos. Devido às suas caraterísticas, o estilo ReST é muitas vezes apelidado de "*Lightweight alternative to web services*" uma vez que apresenta-se como uma arquitetura "stateless" baseada no modelo cliente-servidor permite "*cache*" o que permite aumentar a sua performance e no entanto consegue ultrapassar alguns obstáculos inerentes a serviços *web* tradicionais.

2.4 Protocolos de comunicação

> "We live in a connected world, and modern software has to navigate this world. So the building blocks for tomorrow's very largest solutions are connected and massively parallel. It's not enough for code to be "strong and silent" any more. Code has to talk to code. Code has to be chatty, sociable, well-connected. Code has to run like the human brain, trillions of individual neurons firing off messages to each other, a massively parallel network with no central control, no single point of failure, yet able to solve immensely difficult problems. And it's no accident that the future of code looks like the human brain, because the endpoints of every network are, at some level, human brains." [Hin]

Na citação transcrita, o autor evidencia a importância do código comunicar entre si para a resolução de determinado problema fazendo uma analogia ao funcionamento do corpo humano. No fundo, para que uma aplicação possa comunicar com outra aplicação desenvolvida noutra linguagem de programação inserida num ambiente de desenvolvimento também diferente, é necessário que ambas aplicações sejam interoperáveis. Só assim se conseguirá implementar uma arquitetura de rede que albergue um vasto conjunto de tecnologias e dispositivos diferentes inseridos num ambiente de comunicação através do qual comunicam eficientemente e a partir dos quais se extraem informações fomentando o desenvolvimento de serviços com mais valias.

2.4.1 XMPP

No final do século XX, já era prática comum as pessoas estarem ligadas à Internet e comunicarem entre si utilizando diversas aplicações clientes de *instant messaging* (IM). IM fornece capacidades de comunicação entre utilizadores quase em tempo real além de permitir visualizar a informação relacionada com a presença de um utilizador (se está *online*, *offline*, etc). No passado existiam muitas aplicações de clientes conetadas a diferentes servidores que implementavam também protocolos de comunicação proprietários distintos. Na prática, a utilização de protocolos proprietários distintos levam a que os utilizadores de uma aplicação de IM não possam comunicar com os utilizadores de outra aplicação caso esta utilize um protocolo distinto.

O *Extensible Messaging and Presence Protocol* (XMPP)[3] é um protocolo aberto que utiliza a Internet para realizar comunicações baseadas em troca de documentos XML. Apesar da sua popularidade como protocolo de *instant messaging* (IM), o XMPP pode também ser utilizado como um serviço de mensagens genérico no qual os utilizadores trocam vários formatos de mensagens entre si.

No ano 2007 estimaram-se entre 40 a 50 milhões de utilizadores de tecnologias XMPP. Fornecedores de serviços como a Apple, Google, IBM, Nokia, Sony e Sun Microsystems já possuem suporte para XMPP [Pet07].

2.4.1.1 Arquitetura

O propósito do XMPP é permitir a troca de pedaços de dados estruturados (denominados por "*XML stanzas*") através de uma rede de comunicação entre duas ou mais entidades. XMPP é normalmente implementado utilizando uma arquitetura cliente-servidor distribuída, na qual um cliente necessita de se conetar ao servidor para obter acesso à rede e assim poder trocar *stanzas* com outras entidades. De notar que estas entidades comunicantes podem estar conetadas a servidores distintos [SA11].

Na Figura 2.2 é apresentada a composição de uma rede XMPP. A partir da figura, pode-se observar que existem dois servidores XMPP que trocam mensagens entre si e com os respetivos clientes. Neste tipo de redes, os clientes conetam-se ao seu servidor XMPP através de um protocolo cliente-servidor que utiliza normalmente uma conexão TCP para comunicar. Os servidores

[3]Página principal: http://www.xmpp.org/

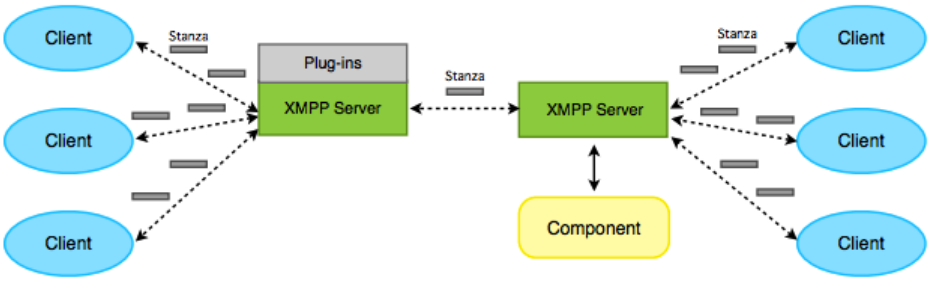


Figura 2.2: Aspeto geral de uma rede XMPP [Teh]

também podem comunicar entre si através de uma conexão TCP. Existe ainda a possibilidade de adicionar funcionalidades ao servidor acrescentando-lhe um novo serviço. Estes serviços possuem a sua própria identidade e endereço dentro do servidor mas executam externamente recorrendo a um "protocolo de componentes" (definido no XEP-0114 [4]) para comunicarem com o servidor. Desta forma é posível desligar um serviço sem que isso afete o servidor. Em IM, uma conversa entre vários utilizadores é um exemplo típico de um componente. A maior parte dos servidores pode ser também extendida através de *plug-ins*. Ao contrário dos componentes, os *plug-ins* funcionam dentro dos processos do servidor, são escritos normalmente na mesma linguagem do servidor e podem alterar o seu comportamento.

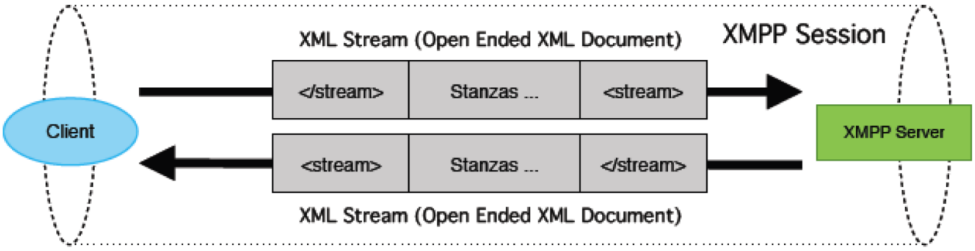


Figura 2.3: XML Stream [Teh]

Em XMPP os dados XML são organizados em *streams* (ver Figura 2.3), ocorrendo estes em pares, cada um orientado num sentido da comunicação. Cada *stream* consiste num elemento de abertura, seguido de um ou mais *XML Stanzas* e um elemento de fecho. Cada XMPP *stanza* é um filho de primeiro grau do *stream* com todos os seus elementos e atributos. Existem três tipos básicos de *Stanzas*: <presence>, <message> e <iq>. O de presença controla e reporta a disponibilidade de uma entidade na rede. O servidor de um utilizador envia automaticamente informação de presença para todos os contatos que possuam uma assinatura de presença do utilizador (a assinatura de presença é direcional, ou seja, possui um remetente e um destinatário explicitamente identificados). O *stanza* de mensagem possui o corpo da mensagem a ser entregue. Por último,

[4] disponível na página de internet http://xmpp.org/extensions/xep-0114.html

o "iq" que significa *Info/Query* prevê um mecanismo de requisição e resposta para comunicações XMPP.

Os clientes numa rede XMPP possuem um identificador único denominado por *Jabber Identifier* (JID). Este identificador segue o seguinte esquema: "utilizador@domínio/recurso". O "utilizador" identifica unicamente um utilizador e apenas é valido dentro do domínio especificado. O "domínio" é o nome DNS da entidade que depois de resolvido fornece a localização do servidor ao qual o cliente está ligado. O "recurso" identifica uma conexão do cliente e permite ao utilizador conetar-se ao mesmo servidor diversas vezes a partir de localizações distintas. Os endereços XMPP possuem semelhanças com os endereços de email, por isso, dentro de um servidor XMPP, as contas de utilizador são normalmente autenticadas com o mesmo sistema de autenticação utilizado pelo sistema de *email*.

2.4.1.2 Comunicação

Em XMPP, cada cliente coneta-se ao servidor que controla o seu domínio XMPP. Este servidor é responsável pela autenticação, entrega de mensagens e manutenção da informação de presença de todos os seus clientes. Se um utilizador pretende enviar uma mensagem instantânea para fora do seu domínio, o servidor XMPP contacta um servidor externo que controla o domínio XMPP do destinatário da mensagem e redireciona-lhe a mensagem. O servidor XMPP do destinatário é responsável pela entrega da mensagem ao utilizador dentro do seu domínio. O mesmo modelo servidor-servidor aplica-se a todos os dados transacionados incluindo as informações de presença.

O processo através do qual um cliente tem que passar para se conetar ao servidor, trocar *stanzas* e terminar a conexão é o seguinte [SA11]:

- Determinar o localização do servidor normalmente baseado na resolução DNS [5];
- Abrir um canal TCP;
- Abrir um *XML stream* sobre o canal TCP[6];
- Para encriptação do canal de comunicação, opcionalmente negociar *Transport Layer Security* (TLS)[7];
- Autenticação utilizando mecanismo *Simple Authentication and Security Layer* (SASL)[8];
- Ligar um recurso a um *stream*[9];
- Trocar *stanzas* XML com outras entidades na rede[10];

[5] documentação disponível em http://tools.ietf.org/html/rfc6120#section-3.2
[6] documentação disponível em http://tools.ietf.org/html/rfc6120#section-4.2
[7] documentação disponível em http://tools.ietf.org/html/rfc6120#section-5
[8] documentação disponível em http://tools.ietf.org/html/rfc6120#section-6
[9] documentação disponível em http://tools.ietf.org/html/rfc6120#section-7
[10] documentação disponível em http://tools.ietf.org/html/rfc6120#section-8

- Fechar o *stream* XML[11];
- Fechar a ligação TCP.

O protocolo XMPP utiliza comunicações TCP persistentes em dois cenários distintos: quando um utilizador liga-se a um servidor XMPP, utiliza uma ligação TCP no porto 5222 e os servidores utilizam a porta 5269 para comunicarem entre si.

2.4.1.3 Aplicações do protocolo

O XMPP fornece uma *framework* genérica para transporte de mensagens através da rede. Alguns cenários para a utilização deste protocolo, incluem salas de conversação e indústrias de jogos. Com a massificação de jogos *online*, este protocolo permite realizar a comunicação entre jogadores, incluindo informações de presença, funcionalidades de IM e outras formas de comunicação.

O protocolo XMPP é um protocolo aberto, flexível e extensível, tornando-o um dos protocolos de eleição para comunicações instantâneas na Internet. Este protocolo interoperável pode ser importante no contexto M2M, uma vez que os dispositivos inteligentes poderão possuir JIDs e comunicar na rede com outros dispositivos. O XMPP já possui uma vasta quantidade de extensões e funcionalidades que permitirão estabelecer comunicações seguras com outros dispositivos e interagir com eles. A informação de presença pode representar uma mais valia em determinados cenários nos quais se deseja monitorizar o estado de um dispositivo.

2.4.2 AMQP

O protocolo *Advanced Message Queue Protocol* (AMQP) surge como uma solução de *messaging* que permite que as aplicações se possam conetar umas às outras como componentes de uma aplicação maior. As mensagens são transferidas de forma assíncrona e distingue-se perfeitamente o envio e a receção de dados. O objetivo do AMQP é permitir o desenvolvimento e utilização por parte da indústria de *middleware* de mensagens para que as aplicações que utilizem este protocolo possam comunicar entre si utilizando standards abertos, resultando numa redução de custos em sistemas de integração empresariais. A partir de [Rab] foi possível analisar algumas caraterísticas deste protocolo enunciadas nas secções seguintes.

2.4.2.1 Arquitetura

O protocolo AMQP segue o modelo apresentado na figura 2.4. Segundo este modelo, as mensagens são publicadas nos *exchanges* pelos produtores. Os *exchanges* podem ser comparados a postos dos correios e são utilizados para distribuir cópias das mensagens pelas filas de mensagens utilizando determinadas regras, conhecidas por *bindings*. As filas de mensagens, os *exchanges* e os *bindings* são muitas vezes referidos como entidades AMQP.

Quando uma mensagem é publicada, os produtores podem especificar vários atributos nas mensagens. Alguns destes atributos podem ser utilizados pelos servidores (*brokers*) enquanto que

[11] documentação disponível em http://tools.ietf.org/html/rfc6120#section-4.4

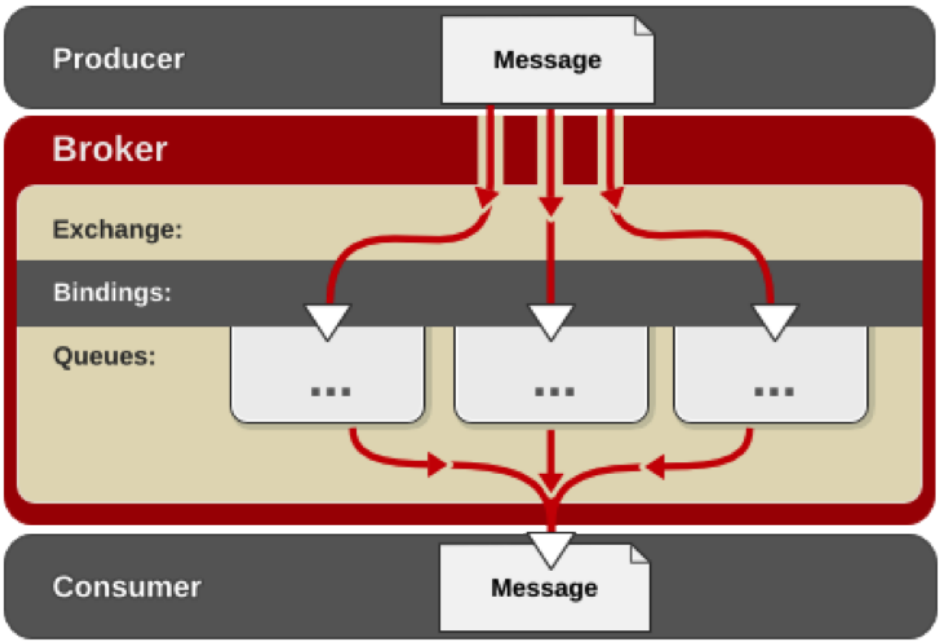


Figura 2.4: Entidades AMQP [YB11]

outros serão apenas utilizados pelas aplicações consumidoras da mensagem. Este protocolo possui a noção de *message acknowledgement* que permite aos consumidores notificarem o *broker* da chegada de uma mensagem, que por sua vez poderá retirar a mensagem da fila. Em determinadas situações (por exemplo caso uma mensagem não possa ser entregue), os produtores poderão definir um mecanismo que permite especificar a acção a realizar na ocorrência de tal situação.

O AMQP é um protocolo programável na medida em que as entidades e esquemas de roteamento são definidos pelas próprias aplicações e não pelo *broker*. Desta forma, é possível que as aplicações declarem que tipo de filas e *exchanges* pretendem, assim como algoritmos de roteamento e subscrições. Esta caraterística permite uma fácil personalização do serviço.

Podemos estabelecer uma analogia que nos permita identidicar facilmente cada entidade deste modelo: A fila representa a uma cidade, o *exchange* é o aeroporto dessa cidade e os *bindings* são o caminho do aeroporto para o destino pretendido. Podem existir zero ou mais caminhos para alcançar o destino final.

Para permitir que um único *broker* implemente vários ambientes isolados (grupos de utilizadores, exchanges, filas, etc), AMQP inclui o conceito de *virtual hosts* (*vhosts*). *Vhosts* são parecidos com os *virtual hosts* utilizados por muitos serviços *web* e fornecem um ambiente completamente isolado onde as entidades AMQP residem. Os clientes deste protocolo especificam quais os *vhosts* que pretendem utilizar durante a negociação da conexão. Alguns casos de utilização de *vhosts* compreendem a separação de entidades AMQP utilizadas por diferentes grupos de aplicações e ainda a separação de múltiplas instalações e ambientes (produção, desenvolvimento, testes) de uma ou mais aplicações.

Uma diferença chave que distingue o protocolo AMQP de outros da mesma área Message-oriented middleware (MOM) é que as mensagens não são enviadas diretamente para filas. Ao invés disso, são enviadas para exchanges que as direcionam de acordo com as regras chamadas bingings. Isto significa que o roteamento é em primeira mão tratado pelo broker e não pelas próprias aplicações.

2.4.2.2 Entidades AMQP

Como já vimos anteriormente, *exchanges* são entidades AMQP para onde as mensagens são enviadas. Estes redirecionam a mensagem para uma ou mais filas. Existem quatro tipos distintos de *exchanges*: "*Direct Exchange*", "*Fanout Exchange*", "*Topic Exchange*" e por último, o "*Headers Exchange*". Além do tipo, podem-se definir vários parâmetros adicionais como o nome e a durabilidade, entre outros.

Muito resumidamente, o "*Direct Exchange*" realiza a entrega de mensagens nas filas baseado numa chave de roteamento presente nas mensagens. É o *exchange* ideal para comunicações *unicast*. O "*Fanout Exchange*" redireciona as mensagens para todas as filas que se encontram ligadas a este tipo de *exchange*. Este tipo poderá ser utilizado por entidades que pretendam por exemplo enviar informações relativas a resultados desportivos para vários dispositivos móveis. O "*Topic Exchange*" realiza o roteamento de mensagens para uma ou várias filas baseado numa chave de roteamento assim como um padrão (que foi utilizado para ligar uma fila a um "*exchange*"). É bastante utilizado para implementar variações do modelo *Publish-Subscribe* por exemplo, para um dispositivo móvel receber informações relativas a um determinado desporto. Finalmente, o "*Headers Exchange*" é utilizado para redirecionar mensagens baseado nos *headers* das mensagens em vez da utilização da chave de roteamento, uma vez que em algumas soluções é mais fácil colocar informação nos *headers* do que utilizar a informação presente na chave de roteamento.

Filas no modelo AMQP são muito similares a outras filas de mensagens ou seja, guardam mensagens que são comsumidas por aplicações. Assim como os *exchanges*, também as filas possuem um nome e uma propriedade de durabilidade, mas podem ser também exclusivas (utilizadas apenas por uma conexão) e podem ser automaticamente apagadas quando o último consumidor cancela a sua subscrição.

Os *bindings* são regras que os *exhanges* utilizam para redirecionar mensagens para filas. Para ordenar que o *exchange* E roteie uma mensagem para a fila Q, Q tem que estar "ligado" a E. *Bindings* podem possuir uma chave de roteamento opcional que, como vimos anteriormente, são utilizados por alguns tipos de *exchanges*. O propósito de chaves de roteamento é permitir que uma mensagem seja entregue numa determinada fila. Por outras palavras, a chave de roteamento atua como um filtro. Mais detalhes sobre roteamento de mensagens no Capitúlo 6.

2.4.3 Outros protocolos

2.4.3.1 Atom Publishing Protocol

Muitas aplicações para dispositivos inteligentes requerem que a informação acerca de um recurso ou coleção de recursos seja atualizada. Com a utilização da tecnologia ATOM, a Internet possui um modelo standard e ReSTful para interagir com coleções. O protocolo Atom-Pub extende as interações apenas de leitura da tecnologia Atom para possuir permissões de escrita sobre os recursos. Uma vez que o protocolo Atom possui interfaces ReSTful, as interações com *feeds* de ATOM podem ser executadas através de simples operações GET. Atom permite que os clientes possam monitorizar dispositivos inteligentes subscrevendo os seus *feeds* através de um servidor remoto em vez de interagir diretamente com o dispositivo em si. É possível definir várias condições para que quando estas sejam atingidas, o servidor coloque uma mensagem JSON no servidor ATOM utilizando o protocolo AtomPub. Esta interacção permite que milhares de clientes possam monitorizar um único sensor delegando o processamento a um servidor intermediário e mais poderoso [DVFE].

2.4.3.2 Stomp

O protocolo *Streaming Text-Oriented Messaging Protocol* (STOMP) surgiu da necessidade de conetar *brokers* de mensagens empresariais a partir de linguagens de scripting como Ruby, Python e Perl. É uma alternativa a outros protocolos de mensagens abertos como o AMQP (definido na secção 2.4.2).

STOMP é um protocolo baseado em *frames* modelados em HTTP. Um *frame* consiste num comando, um conjunto de *headers* e um *body* opcionais. STOMP é baseado em texto mas também permite a transmissão de mensagens binárias. A codificação predefinida é o UTF-8 mas possui suporte para outras codificações.

Um servidor STOMP é modelado como um conjunto de destinos para o qual as mensagens podem ser entregues. O protocolo STOMP trata os destinos como *strings* opacas e a sua sintaxe é específica da implementação de cada servidor. A entrega, troca de mensagens e semânticas dos destinos podem variar de servidor para servidor e ainda de destino para destino. Isto permite que os servidores sejam "criativos"com as semânticas que irão suportar com o STOMP.

Um cliente STOMP é um agente utilizador que pode atuar de duas formas em simultâneo: como produtor de mensagens, enviando as mensagens para um destino no servidor via *frame* SEND; como consumidor enviando um *frame* SUBSCRIBE para determinado destino e recebendo mensagens do servidor em forma de *frames* MESSAGE.

Este protocolo é *human readable* e aparentemente simples de implementar. Já existem vários clientes desenvolvidos para mais de uma dezena de linguagens, que incluem *Java*, *C#*, *Python*, *Ruby*, *Perl*, *PHP* e *JavaScript*. Existem também um número crescente de *brokers* que suportam este protocolo.

Por outro lado, para um protocolo tão simples, as especificações oficiais não sao as melhores. O protocolo utiliza um delimitador específico que por sua vez é incompatível com o modo binário,

sendo assim necessário especificar o tamanho do conteúdo das mensagens em vez da utilização da string delimitadora.

2.4.3.3 R-Event

O projeto R-Event consiste numa *framework* para construção de serviços orientados a eventos. O sistema de notificação desenvolvido permite que utilizadores dentro de um grupo específico publiquem e subscrevam informações de presença. Os utilizadores podem utilizar a informação de presença para permitir por exemplo a elaboração colaborativa de um trabalho entre vários colaboradores. O protótipo foi escrito em *Java* e implementa um modelo MVC sobre comunicações seguras (HTTPS). Neste trabalho, os autores apresentam uma *framework* na qual os elementos pertencentes a uma arquitetura orientada a eventos são projetados e representados através de recursos ReST.

2.4.4 Técnicas Comet

O protocolo HTTP é um protocolo cliente-servidor *stateless*, onde as interações são sempre iniciadas do lado do cliente. Este modelo de interação é valioso quando se trata de aplicações orientadas ao controlo onde os clientes podem ler informações ou escrever dados nos dispositivos inteligentes. O problema surge quando se pretende desenvolver um sistema de comunicação bidirecional baseado em eventos, no qual os dados têm de ser enviados para os clientes de forma assíncrona mal sejam produzidos os conteúdos. Em aplicações relacionadas com a monitorização de um determinado ambiente, os servidores devem possuir capacidades de colocar os dados do lado do cliente sem que ele esteja periodicamente a realizar pedidos de informação.

Uma alternativa passa pela utilização de um modelo denominado por *Comet*, também conhecido como "*HTTP Streaming*" ou "*Server Push*". Este modelo permite que um serviço *web* envie dados para os clientes sem que o cliente faça essa requisição explicitamente. Uma vez que os *browsers* não foram implementados para este tipo de comunicações, várias abordagens têm sido propostas neste sentido. Uma ideia genérica, é que o servidor não termine uma ligação TCP depois da resposta ter sido devolvida ao cliente e utilize este canal de comunicação para enviar informações relacionadas com os eventos.

É claro que a WoT necessita de mais desenvolvimentos e standards nestas áreas no entanto, recentes desenvolvimentos em tecnologias como *HTML5* e os seus *WebSockets* são um sinal de que os esforços estão a ser orientados na direção da WoT.

2.5 Discussão

Como referido em 2.1, existem vários problemas associados à evolução de comunicações M2M, entre eles a quantidade de dados que serão gerados por novos dispositivos inteligentes que irão estar presentes na rede para que possam ser consultados e manipulados. Nos últimos

anos têm sido realizados vários esforços no âmbito do desenvolvimento de tecnologias e arquiteturas de redes que sejam interoperáveis e que permitam uma comunicação segura e eficiente entre dispositivos inteligentes. Foi também observado que existe uma falta de standards na área de comunicações M2M, não existindo ainda uma definição clara e objetiva de uma arquitetura (e tecnologias associadas) que possam servir de base para uma solução que inclua a comunicação entre dispositivos conetados em rede.

Um dos vários problemas que existem está relacionado com a multiplicidade de cenários que utilizam comunicações entre máquinas. Os dispositivos e suas comunicações irão formar um novo tipo de Internet. É necessário também uma especificação clara de quais os requisitos a cumprir por soluções desenvolvidas no âmbito de IoT, uma vez que representam um papel fundamental quando se pretende desenhar uma solução.

Esta revisão bibliográfica pretende analisar algumas soluções fundamentalmente diferentes na abordagem deste problema para que se compreenda melhor quais as soluções que embora tenham sido desenvolvidas originalmente com outro propósito, oferecem capacidades para a resoluçao de alguns dos problemas mencionados na secção 2.1.

Começando pelo estilo de arquitetura ReST, este estilo encontra-se intimamente ligado à arquitetura da Internet e ao seu sucesso como a maior rede de comunicação descentralizada a nível mundial. Ao adotar este estilo para comunicações M2M, usufrui-se das suas vantagens como por exemplo, a possibilidade de criação de aplicações "*loose coupled*", a escalabilidade associada a esta arquitetura, interações *stateless*, entre outros. No cenário de comunicações M2M uma arquitetura ReST poderá fornecer representações de recursos em vários formatos (JSON, XML, HTTP, JPEG, etc.), sendo que estes recursos podem constituir representações de dispositivos inteligentes e respetivas propriedades. A utilização de uma interface uniforme (HTTP) para interagir com os recursos, assim como o modelo cliente-servidor associado a este estilo representam uma forma de comunicar já bastante utilizada e que é facilmente compreendida. Resta mencionar que apesar do modelo cliente-servidor poder ser visto como uma vantagem ao nível da sua compreensão, em determinados cenários apresenta várias lacunas, uma vez que em cenários onde se monitorizam objetos, seria talvez preferível utilizar uma arquitetura orientada a eventos para que o servidor possa colocar informações do lado do cliente sem que este último necessite de realizar consultas periódicas ao servidor. Exemplos de técnicas *Comet* recentes, incluem desenvolvimentos como o *HTTP5* e "*WebSockets*" que permitirão a um servidor possuir tais capacidades de notificação. Para já, e uma vez que apenas alguns *browsers* possuem suporte para estas tecnologias, estas técnicas não serão abordadas. Outro problema existente consiste na falta de mecanismos de descoberta dos vários dispositivos ligados à rede de comunicação. Já existem alguns desenvolvimentos nesta área (WSDL 2.0 e WADL), porém ainda não existe um mecanismo generalizado para permitir tais funcionalidades de descoberta.

O AMQP como protocolo *Message Oriented Middleware* (MOM) fornece tais capacidades. O modelo associado *Publish-Subscribe* pode ser utilizado como sistema de notificações. Em determinados cenários, poderá ser relevante a utilização deste protocolo, uma vez que permite a configuração das comunicações do lado do clientes e não do lado do *broker*, tornando-o assim um

protocolo facilmente configurado e personalizado. Os *exchanges* já fornecem vários algoritmos de roteamento de mensagens que podem ser úteis para o desenvolvimento de uma determinada solução, traduzindo-se numa redução do tempo necessário para implementação de uma solução que utilize este protocolo.

Por outro lado, o protocolo XMPP (originalmente desenvolvido para IM), já possui embutidas várias funcionalidades relevantes no âmbito do comunicações M2M. Essas funcionalidades incluem a segurança das comunicações, a escalabilidade (fruto da utilização de *Server Federation*) e possui ainda vários mecanismos para difusão de informações de presença. Neste contexto, os dispositivos inteligentes possuirão um identificador e irão difundir informações da sua presença na rede, que permitirá a descoberta de dispositivos e estabelecimento de comunicações entre os dispositivos conetados à rede de informação. Neste trabalho, para além do desenvolvimento de uma plataforma para comunicações M2M, realiza-se também uma comparação entre essa plataforma e uma solução proprietária XMPP utilizando várias métricas num cenário idêntico.

2.6 Standardização

Nesta secção procura-se apresentar os esforços realizados por parte de organizações internacionais cujo objetivo é o desenvolvimento de standards para comunicações M2M.

Em 2009 foi criado um novo comité técnico (TC) do *European Telecommunications Standards Institute* (ETSI) cuja missão é o desenvolvimento de standards para M2M. A objetivo deste comité inclui a especificação de soluções baseadas em tecnologias diferentes e standards que possam ser interoperáveis. O TC é composto por membros de operadoras de redes europeias, americanas e asiáticas assim como de fabricantes de dispositivos a nível mundial que cooperam no desenvolvimento de atividades de standardização na área de comunicações M2M. O *3rd Generation Partnership Project* (3GPP) é um acordo de colaboração estabelecido em Dezembro de 1998 que visa unir vários grupos de atividades de standardização da área de telecomunicações incluindo o ETSI.

No âmbito do 3GPP, para além do acrónimo M2M para designar comunicações "Machine-to-Machine", também se utiliza a expressão *Machine Type Communication* (MTC).

Nos últimos anos verificou-se um aumento do número de atividades de standardização em áreas como as comunicações M2M. Existem vários fatores que justificam este esforço de desenvolvimento de standards:

- a procura do mercado;
- processos de standardização são essenciais para o desenvolvimento de tecnologia a longo prazo;
- criação de redes interoperáveis;
- devido à utilização em massa de dispositivos M2M que irão sobrecarregar as redes existentes.

Até agora os operadores móveis são especialistas em comunicações entre humanos. O mercado M2M é um novo mercado que requer uma mudança de mentalidade em áreas como as redes de comunicação e tecnologias associadas. A fragmentação e complexidade de aplicações, a falta de processos de standardização, a competição tecnológica, o baixo retorno por conexão e as mudanças que são necessárias na rede de comunicação aliados à falta de experiência por parte dos operadores móveis representam os desafios que estes operadores necessitam de ultrapassar [3GP11].

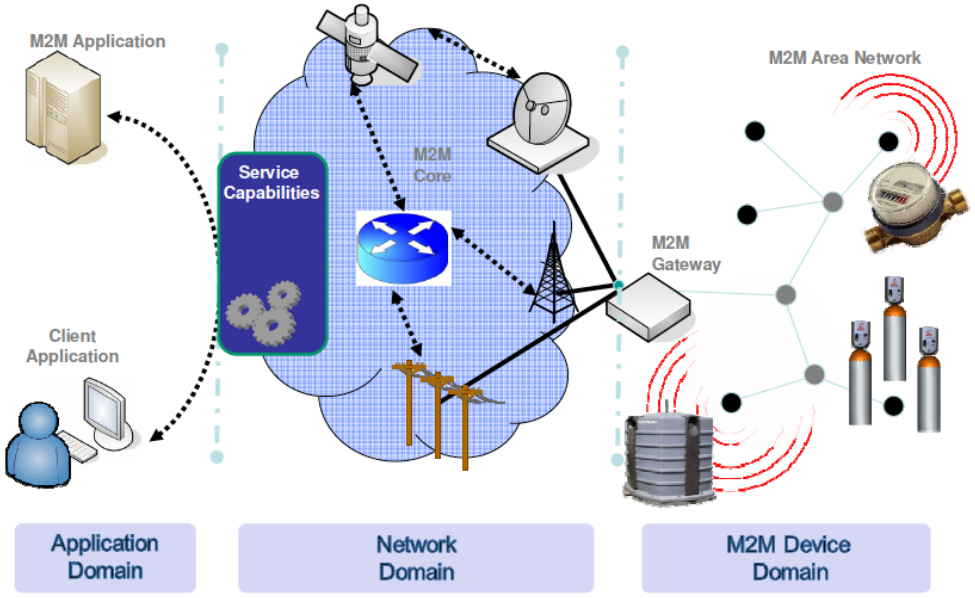


Figura 2.5: Visão geral de uma arquitetura M2M proposta pela ETSI

Na Figura 2.5 apresenta-se uma visão geral de uma arquitetura para realização de comunicações M2M onde estão presentes vários elementos chave. O *M2M Device* representa um dispositivo com capacidades de resposta a solicitações de dados. A *M2M Area Network* no domínio do dispositivo fornece capacidades de conetividade entre dispositivos M2M e *M2M Gateways* que por sua vez asseguram a interconexão entre dispositivos M2M e a rede de comunicação. As *M2M Communication Networks* ao nível da rede estabelecem comunicações entre os *gateways* e as aplicações M2M (através de tecnologias como WLAN, WiMAX, LTE, entre outras). Por fim, as aplicações M2M contém a camada de *middleware* onde os dados são transferidos entre serviços e aplicações.

O documento *Service Requirements for Machine-Type Communications (MTC)* [3GP11] identifica e especifica os requisitos para comunicações MTC. Além disso, identifica aspetos do serviço que necessitam de ser melhorados para que as redes de comunicação possam albergar também os dispositivos em vez de apenas humanos como acontece agora.

Segundo [3GP11], existem três tipos principais de comunicações ilustrados nas Figuras 2.6, 2.7 e 2.8.

A Figura 2.6 apresenta uma arquitetura seguida pela maior parte das aplicações de hoje, na

qual existem vários dispositivos conectados a um ou mais servidores. Estes servidores são operados pelo operador de rede que fornece conetividade para servidores MTC. A diferença entre a Figura 2.6 e a Figura 2.7 reside na entidade que controla o servidor MTC: na primeira figura o servidor MTC é controlado pelo operador de rede, ao passo que na segunda figura o servidor MTC não é controlado pelo operador de rede.

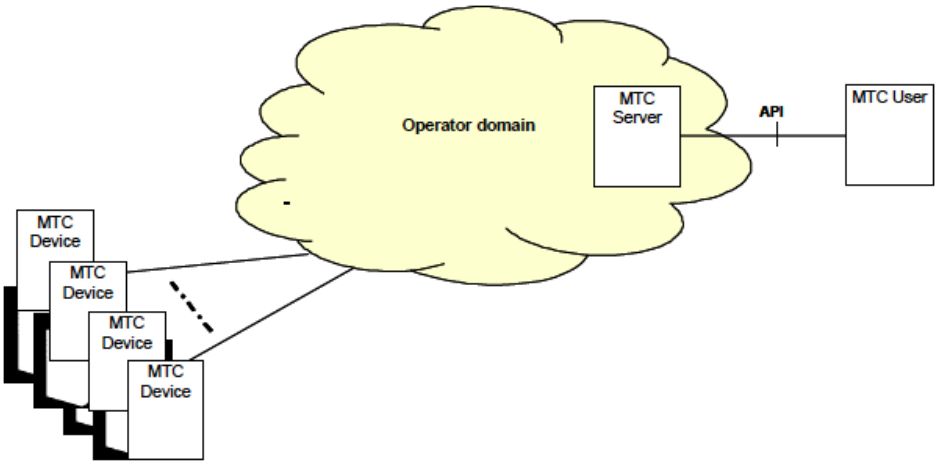


Figura 2.6: Cenário de comunicação com dispositivos MTC que comunicam com um servidor MTC que se encontra dentro do domínio do operador [3GP11]

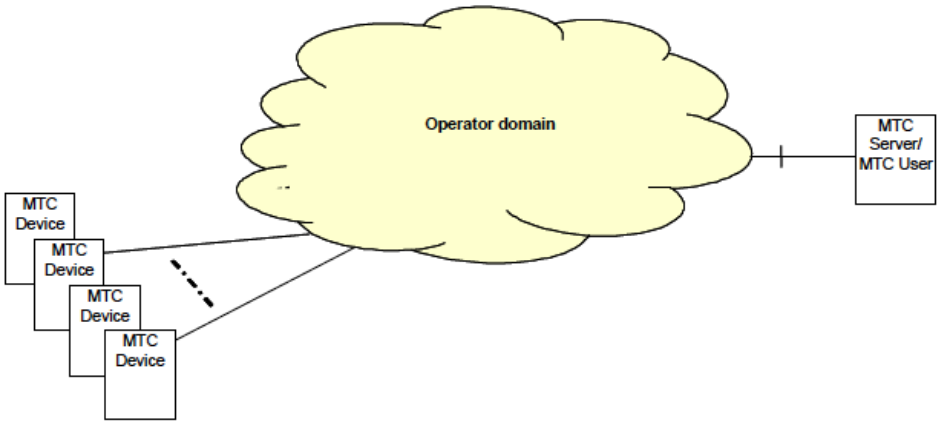


Figura 2.7: Cenário de comunicação com dispositivos MTC que comunicam com um servidor MTC que se encontra fora do domínio do operador [3GP11]

Na Figura 2.8, a comunicação entre dispositivos MTC acontece de forma direta sem a necessidade de um servidor MTC intermédio.

Comunicações M2M ou MTC são propostas para permitir aos operadores de redes fornecer serviços de comunicações MTC com custos operacionais reduzidos, além de reduzir o impacto e esforço necessário para lidar com grandes sistemas MTC de grupos de comunicação. A otimização

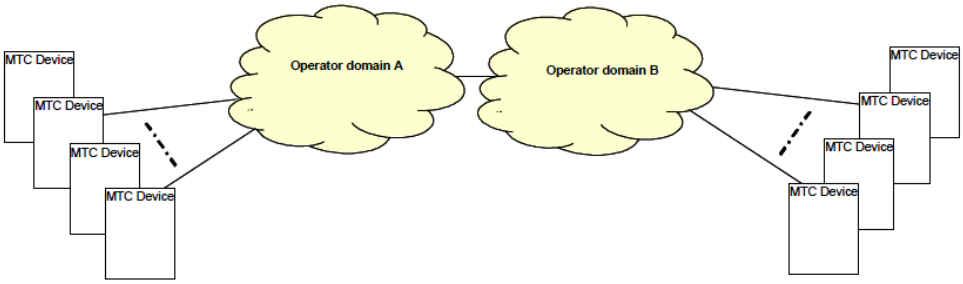


Figura 2.8: Comunicação entre dispositivos MTC que comunicam diretamente entre si [3GP11]

de operações de redes para minimizar o impacto energético destas comunicações em dispositivos que possuem várias restrições ao nível do seu consumo de energia aliado ao estímulo para o desenvolvimento de novas aplicações MTC representam também os objetivos da especificação de standards nesta área [3GP07].

Por último, de referir que são propostas arquiteturas de alto nível e também de arquiteturas funcionais para a realização de comunicações M2M no documento [ETS]. Este documento data de Novembro de 2011 e constitui um avanço no trabalho de standardização realizado pelas organizações referidas nesta secção.

Capítulo 3

Definição do problema

As comunicações M2M tratam de permitir que dispositivos inteligentes se descubram e comuniquem entre si sem necessitarem de intervenção humana, abstraindo a tecnologia subjacente. Várias soluções proprietárias foram desenvolvidas nos últimos anos na área de comunicações M2M, porém estas soluções previnem a utilização de tecnologias M2M em larga escala. Existe uma clara falta de soluções que forneçam uma proposta escalável e integrada para a gestão de dispositivos tendo em conta algumas caraterísticas associadas à visão IoT entre elas, a interoperabilidade das aplicações desenvolvidas. O problema com estas soluções proprietárias é o facto de não utilizarem protocolos abertos e como consequência, menos pessoas podem usufruir de tais soluções, seja no processo de desenvolvimento de *software*, sejam os utilizadores finais que ficam vinculados a uma plataforma sem poderem utilizar outros sistemas para usufruirem do serviço.

Várias atividades de standardização na área de comunicações M2M encontram-se ainda em fase de desenvolvimento faltando por isso, a existência de uma solução capaz de responder eficazmente aos requisitos propostos pela visão IoT. Mais que isso, ainda não existe nenhuma comparação objetiva entre várias soluções para prestação de serviços.

Esta dissertação tem como objetivo o estudo de várias tecnologias abertas que podem ser úteis no processo de comunicação M2M sobre a alçada IoT e inclui ainda o desenvolvimento de uma solução que vá de encontro ao maior número possível de caraterísticas enunciadas pela visão IoT (enunciadas na Secção 2.1).

Adicionalmente irá realizar-se uma análise comparativa de duas soluções baseadas em protocolos abertos num cenário específico. Uma das soluções a comparar está a ser desenvolvida no âmbito de uma dissertação a decorrer em paralelo [Pru12] e utiliza o protocolo XMPP como protocolo base de comunicação, que será comparada com a plataforma desenvolvida no âmbito deste trabalho que por sua vez utiliza outras tecnologias abertas e interoperáveis (ReST e AMQP). Para a realização desta análise, é necessário definir que métricas serão utilizadas para comparar as plataformas e como avaliar essas métricas em soluções diferentes. Para concretizar a comparação de tecnologias envolvidas nas plataformas surge a necessidade de especificação de um cenário-exemplo que inclua as interações típicas entre os vários elementos do sistema. O cenário escolhido está inserido no contexto de cuidados de saúde que como referido em 2.1, é uma das

principais motivações das inúmeras aplicações da IoT. Já existem algumas soluções proprietárias desenvolvidas nesta área, porém nenhuma delas é baseada nos princípios adjacentes à visão IoT, principalmente ao nível da interoperabilidade das aplicações e dispositivos.

No cenário proposto, pretende-se realizar uma monitorização cardíaca de vários utilizadores por parte de um ou mais utilizadores-monitores (ver diagrama de casos de uso 3.1). Um dispositivo de medição dos batimentos cardíacos encontra-se ligado ao *smartphone* do utilizador monitorizado via *bluetooth*. Os dados gerados pelo sensor de batimentos cardíacos são colocados num servidor com o auxílio do *smartphone* que se encontra ligado à Internet. Este conjunto sensor e *smartphone* simulam um sensor de batimentos cardíacos remoto. Poderão existir vários intervenientes neste contexto, sejam eles vários prestadores de serviços (que poderão monitorizar um utilizador) ou ainda familiares da pessoa monitorizada num outro sistema, remotamente. Esta plataforma permitirá melhorar a qualidade de vida dos utilizadores, permitindo uma redução drástica dos custos associados à monitorização dos mesmos, além de permitir que várias entidades possam consultar os dados relativos a uma determinada pessoa. Adicionalmente, existirá um sistema de notificações para enviar mensagens para os clientes que tenham subscrito as informações de um dado utilizador, por exemplo notificando-os que os batimentos cardíacos do utilizador monitorizado excedeream os 100 batimentos por minuto. No contexto desta dissertação não serão abordados problemas relacionados com a segurança e autenticação de atores no sistema, focando-se apenas no desenvolvimento de uma solução aberta que implemente as funcionalidades básicas de conetividade de um serviço neste contexto e ainda uma comparação efetiva entre duas soluções baseadas em protocolos abertos e interoperáveis.

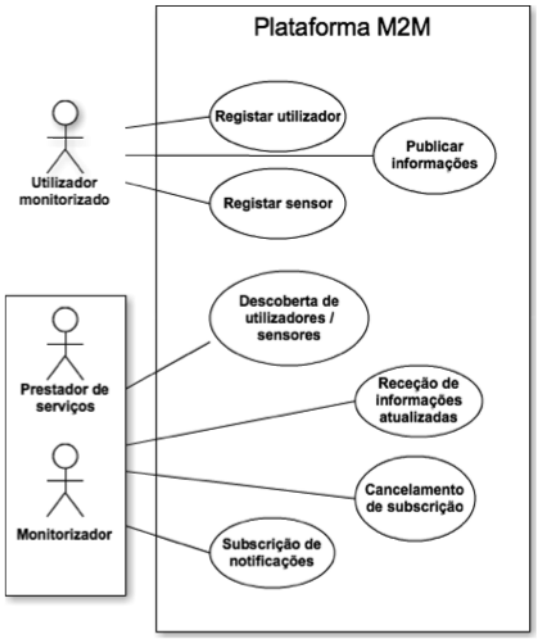


Figura 3.1: Diagrama de casos de uso associados ao cenário

Capítulo 4

Metodologia

Um vez definido o problema e o cenário que fornece o contexto da solução, neste capítulo será descrita a metodologia que será utilizada para a realização desta dissertação.

O primeiro passo passa pela revisão da literatura (Capítulo 2) que permite conhecer algumas tecnologias que podem vir a ser utilizadas no contexto de comunicações M2M. É através desta análise que se escolherá as tecnologias/protocolos a utilizar no desenvolvimento da plataforma.

A partir do cenário descrito no Capítulo 3, é possível definir algumas funcionalidades-base que incluam as interacções típicas entre elementos no sistema dentro do contexto de comunicações M2M. Tendo já definido um conjunto de funcionalidades-base do sistema bem como o cenário em que este se insere, desenvolveu-se uma plataforma M2M que implemente tais funcionalidades inserida no contexto de *healthcare* e que contempla todas as funcionalidades definidas para M2M.

De seguida é necessária uma definição clara de métricas relevantes utilizadas no processo de comparação a realizar entre a plataforma desenvolvida e uma plataforma proprietária cujo protocolo base é o XMPP. Ambas as plataformas operam no mesmo cenário de cuidados de saúde. A comparação que se irá realizar contempla métricas que necessitam ser medidas diretamente a partir das soluções sobre as quais incide como por exemplo, o tamanho das mensagens trocadas, além de uma comparação *a priori* entre os protocolos presentes nas duas plataformas sujeitas a comparação.

4.1 Funcionalidades base

A partir do contexto *healthcare* podem-se aferir algumas funcionalidades-base que por sua vez constituirão os objetos de estudo sobre os quais as diferentes tecnologias irão incidir. Mais concretamente, a plataforma desenvolvida tem obrigatoriamente que implementar estas funcionalidades que posteriormente serão também utilizadas para realizar uma comparação entre as plataformas no que diz respeito à troca de mensagens necessária para executar determinada tarefa. As funcionalidades a implementar e sujeitas a posterior comparação são as seguintes:

- anúncio de sensor/serviço - refere-se à forma como um sensor ou serviço disponibiliza a sua informação para que possa ser facilmente descoberto por uma entidade que deseje interagir com ele;
- descoberta de sensor/serviço - o anúncio de informação relacionada com um sensor ou serviço apenas faz sentido caso existam entidades que possam facilmente encontrá-los na rede de informação para que possam eventualmente interagir com eles. Esta funcionalidade refere-se à forma como é realizada essa descoberta;
- publicação de dados - um determinado sensor ou serviço necessitam de publicar dados referentes ao meio no qual se inserem para que todas os utilizadores que os tenham descoberto na rede à priori possam consultar os dados publicados por si;
- subscrição de sensor/serviço - a arquitetura M2M deve fornecer mecanismos que permitam a utilizadores subscrever as informações de outros sensores/serviços para que, na presença de um evento e consequente publicação de dados, os subscritores possam receber informação atualizada acerca de determinado recurso previamente registado no sistema.
- obtenção de dados - após a descoberta e subscrição de um determinado sensor ou serviço por parte de um utilizador, é fundamental analisar como se processa a recepção de informações subscritas relacionadas com determinado sensor ou serviço.

4.2 Métricas

As métricas definidas para este trabalho incluem a quantidade de informação trocada entre as entidades que fazem parte do sistema assim como o número de interações necessárias para a realização de determinada funcionalidade (descritas na Secção 4.1). A análise destas métricas é realizada ao nível do IP (camada de rede do modelo *Open Systems Interconnection* [DZ83]).

O número de interações é um fator importante uma vez que cada interação que necessite de uma mensagem de resposta, obriga a esperar duas vezes o tempo de percurso da mensagem. É também importante avaliar o número de pacotes transferidos bem como o seu tamanho, uma vez que é bastante diferente a nível de performance realizar um número baixo de transferências de grandes pacotes de dados ao invés de por exemplo, realizar um elevado número de transferências de pequenos pacotes de dados. Estas métricas são importantes devido ao número exorbitante de dados que surgem associados a comunicações M2M conforme referido na Secção 2.1.

Estas métricas serão analisadas com recurso a um programa de captura de pacotes de informação na rede, nomeadamente a aplicação *open-source* multi-plataforma *Wireshark*[1].

Por fim, e devido à experiência resultante do desenvolvimento de uma das soluções, irá realizar-se uma análise acerca do esforço de desenvolvimento necessário para implementar uma aplicação associada ao contexto M2M, uma vez que este esforço de desenvolvimento está intimamente ligado ao retorno do investimento de uma dada solução e pode ser decisivo na hora de entidades com

[1] O software *Wireshark* está disponível para *download* em http://www.wireshark.org/

poder de decisão optarem por uma determinada solução em detrimento de outra para responder a um dado problema inserido num determinado cenário.

4.3 Análise de soluções abertas para M2M

Esta secção apresenta a análise efetuada *a priori* e visa o estabelecimento de uma comparação entre os protocolos utilizados pela plataforma desenvolvida nesta dissertação e uma outra plataforma M2M já existente. O estudo incide sobre as funcionalidades-base apresentadas na Secção 4.1 e para cada uma destas, estuda-se a necessidade de estabelecimento de conexões, o modelo de transações utilizado e ainda o *overhead* de estabelecimento de ligações. Esta análise é fundamental para a compreensão do modo de funcionamento das várias tecnologias implementadas pelas duas plataformas sujeitas a comparação.

4.4 Resultados

Este capítulo pretende complementar a análise *a priori* descrita na secção anterior. A partir da utilização da plataforma desenvolvida e de uma outra plataforma já existente é possível analisar na realidade como se comportam estas plataformas na realização de determinada funcionalidade num cenário real. Os valores medidos são reais e serão capturados com o auxílio do software de captura de pacotes de dados referido na Secção 4.2.

4.5 Ferramentas

Para a realização deste trabalho utilizaram-se as seguintes ferramentas:

- *Latex* - para a escrita da presente dissertação;
- *Ruby on Rails* - plataforma utilizada para desenvolvimento de servidor ReST;
- *RabbitMQ* - servidor de mensagens *open-source* que implementa o protocolo AMQP;
- *Android* - plataforma móvel para desenvolvimento de aplicações cliente do serviço;
- *Wireshark* - *software* que permite realizar a captura de pacotes de informação na rede de comunicação;
- *Doxygen* - *software* que permite a geração de documentação automática a partir de código-fonte.

Capítulo 5

Análise de soluções abertas para M2M

Este capítulo apresenta uma análise prévia à adequação de alguns protocolos definidos no Capítulo 2 para a implementação de uma arquitetura M2M. Para cada tecnologia interessa estudar o seu comportamento na realização de um conjunto de funcionalidades já definidas na Secção 4.1. De forma sumária, as funcionalidades sobre as quais incide o estudo apresentado neste capítulo são:

1. o anúncio de um sensor ou serviço;
2. os mecanismos de descoberta de sensores ou serviços;
3. a publicação de dados;
4. a subscrição de dados;
5. a obtenção de dados.

Neste capítulo serão apresentadas soluções para implementar estas funcionalidades utilizando as tecnologias ReST, ReST conjugado com AMQP e ainda o XMPP, uma vez que o último é o protocolo base da solução proprietária com a qual se irá comparar a plataforma desenvolvida no âmbito desta dissertação.

A análise efetuada neste capítulo constitui uma importante base de decisão quando se trata de escolher uma forma de realizar comunicações M2M de modo eficiente.

5.1 ReST

Como referido no Capítulo 2, é possível implementar um serviço de comunicações M2M utilizando o estilo arquitetónico ReST. Ao especificar um sistema de informação utilizando este estilo, é fulcral que a estrutura de dados que suporta o serviço permita manter toda a informação necessária e relevante para que esta possa ser facilmente consultada e atualizada tanto por dispositivos como por utilizadores ou seja, é necessário especificar com rigor quais os recursos que serão necessários criar para guardar todas essas informações, assim como especificar as relações existentes entre os vários recursos implementados e disponíveis na rede de comunicação.

Nesta secção e para um serviço M2M genérico, apenas serão necessárias duas tabelas que permitirão guardar as informações essenciais acerca dos utilizadores e respetivos sensores (ilustrado pela Figura 5.1).

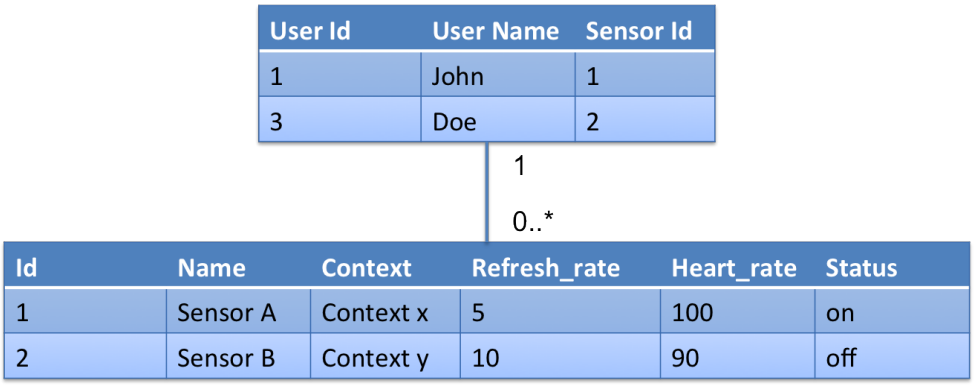


Figura 5.1: Exemplo de estrutura de dados para guardar informações acerca dos recursos (utilizadores e sensores)

Cada utilizador poderá possuir zero ou mais sensores associados, no entanto o contrário já não acontece, sendo que cada sensor apenas poderá pertencer a um dado utilizador num determinado instante. Nas subsecções seguintes será analisado o comportamento de um servidor ReST na execução das funcionalidades identificadas na Secção 4.1.

Para todas as interações ReST e sempre que exista uma resposta subjacente ao pedido efetuado pela aplicação cliente o tamanho necessário para executar as funcionalidades das secções seguintes segue a equação 5.1 onde as parcelas da soma correspondem ao tamanho do pedido e tamanho da resposta respetivamente.

$$L_{transaction} = L_{request} + L_{response} \tag{5.1}$$

5.1.1 Anúncio de sensor/serviço

Utilizando um sistema ReST, um sensor tem obrigatoriamente de se registar no sistema para que possa ser descoberto por outro utilizador. Esta secção trata de apresentar a troca de mensagens necessária para o registo de uma entidade num sistema ReST que posteriormente poderá ser descoberto tanto por sensores como por outros utilizadores.

Na Figura 5.2 observa-se o fluxo de mensagens entre uma aplicação cliente e um servidor que alberga o serviço. Uma vez que o protocolo HTTP é o protocolo base deste estilo arquitetónico, será possível realizar várias operações sobre os recursos disponíveis apenas utilizando métodos HTTP. Inicialmente o cliente realiza um pedido HTTP para a criação ou modificação de um recurso. Cabe ao programador do serviço optar entre um dos dois métodos HTTP disponíveis (PUT e POST) que permitem alterar a informação de um recurso existente.

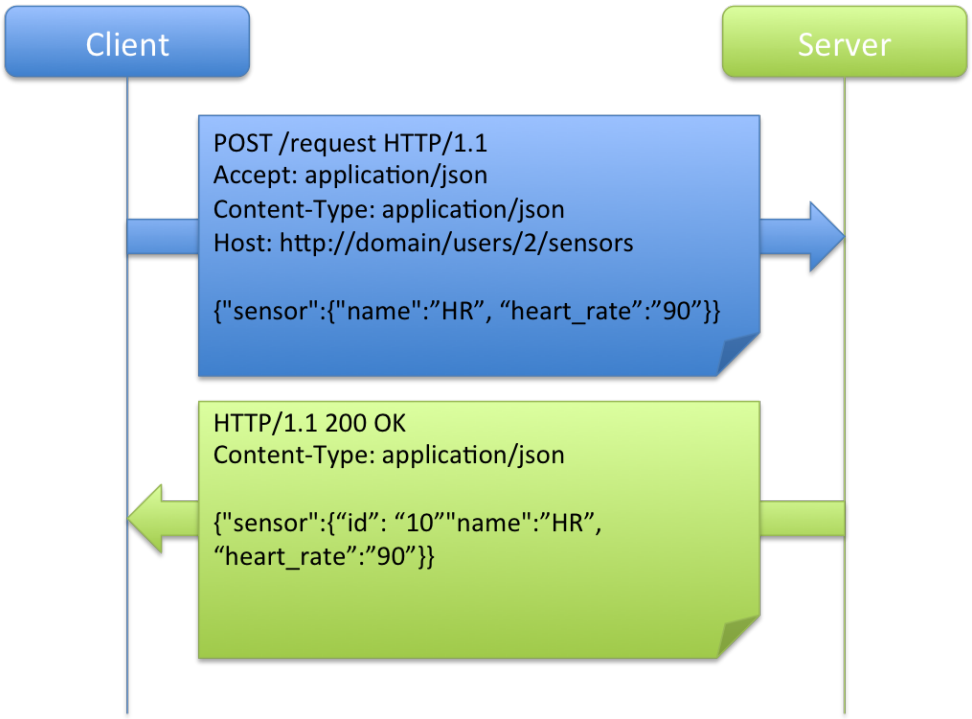


Figura 5.2: Registar informação associada a um sensor

Não existe ainda um consenso acerca da utilização dos métodos HTTP PUT e POST, no entanto é prática comum a utilização do método PUT quando se fornece toda a informação acerca de um recurso para a sua criação ou modificação (incluindo o identificador único).

O RFC 2616 [RIJ99] não identifica claramente as intenções do método POST, no entanto este método deve ser utilizado caso se pretenda criar um novo registo baseado numa representação parcial desse mesmo recurso. O servidor retorna ao cliente uma representação do recurso criado já com o identificador (único) gerado pelo serviço aquando do registo dessa informação no sistema.

Para efeitos deste trabalho, apenas se deseja manter um registo na tabela associada a cada sensor. Desta forma poder-se-ia optar pela utilização do método POST para o registo de um sensor no serviço ao qual este último responde com a informação completa acerca do recurso criado incluindo o identificador gerado por este tornando o serviço mais flexível uma vez que é mais fácil garantir identificadores únicos num domínio. Essa informação poderá ser guardada no sensor para que em interações futuras com o serviço, o sensor utilize esse identificador (fornecido pelo serviço) para interagir com ele utilizando o método HTTP PUT e apenas submetendo as alterações que pretenda efetuar sobre a informação já disponível para consulta.

5.1.2 Descoberta de sensor / serviço

O estilo de arquitetura ReST permite a navegação entre os recursos disponibilizados pelo servidor através de hiperligações que permitem a obtenção da informação desejada. Um exemplo deste mecanismo de descoberta de serviços é ilustrado pela Figura 5.3.

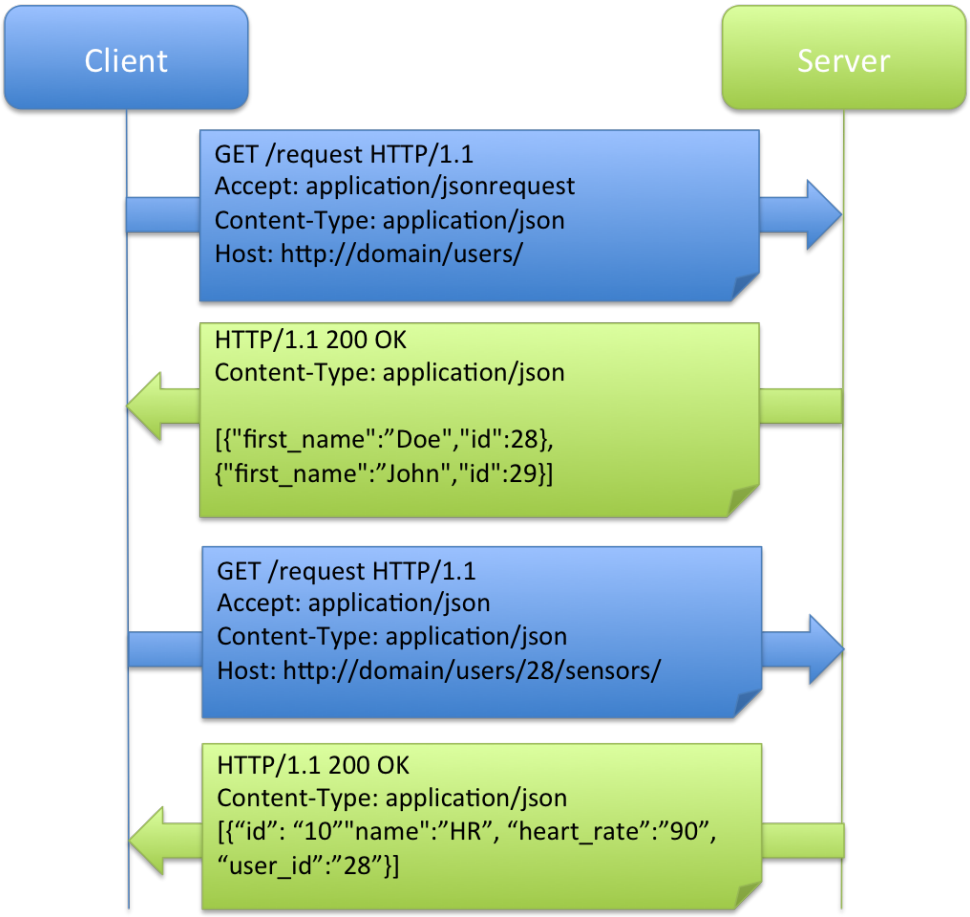


Figura 5.3: Fluxo de mensagens associado à descoberta de recursos num serviço ReST

Através da análise da Figura 5.3, pode observar-se que através de sucessivos pedidos HTTP GET, é possível navegar por entre os vários recursos disponibilizados para a obtenção de informação desejada acerca de um determinado recurso. Esta informação poderá ser utilizada no pedido GET seguinte. É desta forma que um utilizador pode navegar por entre os recursos disponibilizados por um serviço ReST, desde que estes estejam devidamente relacionados (através de hiperligações).

Na Figura 5.3, o cliente inicialmente realiza um pedido por informações acerca dos utilizadores registados no sistema. Após o servidor consultar na base de dados quais os utilizadores que constam registados no serviço, envia uma resposta ao cliente com essa informação formatada de acordo com o especificado pelo cliente no pedido que realizou. Após rececionar esta informação é-lhe possível extrair os identificadores de todos os clientes e com essa informação poderá realizar novamente um pedido HTTP GET utilizando um URL que especifica explicitamente o cliente alvo deste novo pedido e assim obter quais os sensores desse mesmo utilizador. Em suma, as informações rececionadas pelo cliente numa interação com o servidor, poderão ser utilizadas num pedido futuro, estando bem explícitias no URL (que fornece uma representação associada a um determinado recurso registado no sistema).

Um método HTTP que poderá ajudar a perceber quais os métodos que podem ser aplicados a um dado recurso, é o método *OPTIONS* que fornecerá ao cliente essa informação. Mais concretamente, ao utilizarmos o método *OPTIONS* sobre um dado recurso, o servidor poderá por exemplo, informar o cliente que os métodos GET, PUT e POST estão disponíveis para a interação com esse recurso.

A navegação entre recursos através de hiperligações apenas fornece um mecanismo de descoberta de recursos a partir de uma localização base (URL do primeiro pedido), não fornecendo uma forma de encontrar estes na rede de informação como acontece com o caso de serviços *web* tradicionais através das suas interfaces WSDL. No caso de serviços ReST, a resposta a este problema surge através da disponibilização de documentação associada a um serviço numa página *web*, que pode ser facilmente encontrada por um motor de busca.

5.1.3 Publicação de dados

Tendo em conta a utilização do método HTTP POST para o registo de um sensor inteligente no serviço ReST e conhecendo o identificador gerado pelo serviço que foi retornado ao utilizador, é então possível realizar alterações sobre um recurso apenas com a utilização do método PUT como apresenta a imagem 5.4.

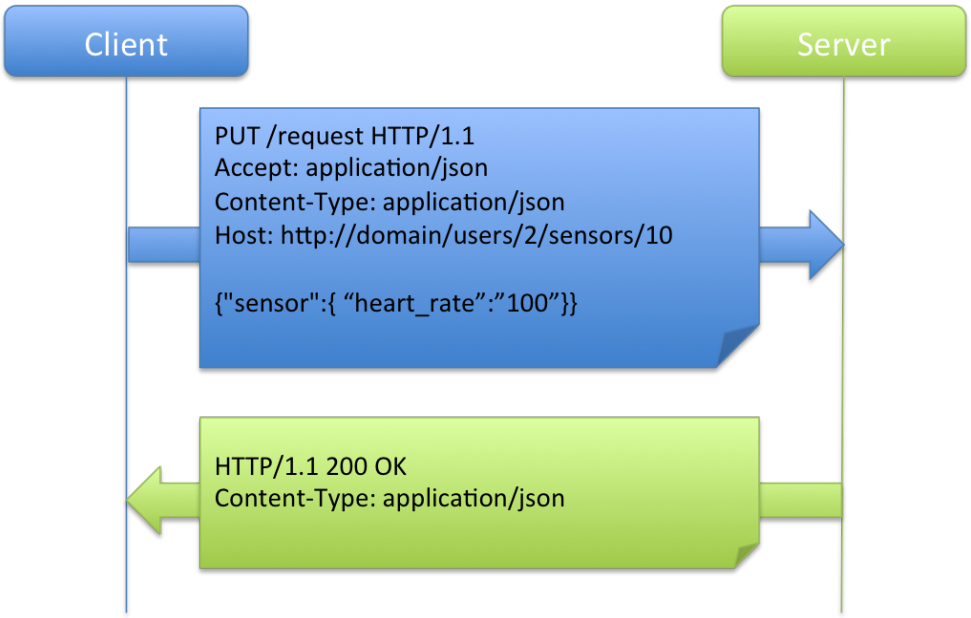


Figura 5.4: Utilização do método HTTP PUT para publicação de dados

A partir da imagem 5.4 é possível analisar um exemplo de uma interação genérica com o serviço, no qual um sensor pretende publicar dados acerca do ambiente em que se insere. Para a realização desta tarefa de forma eficiente, o sensor pode especificar o formato de dados que deseja utilizar para submeter as alterações bem como o formato de dados que pretende receber por parte do servidor para que ele possa facilmente interpretar essa informação. Em semelhança com o que

acontece com outros métodos HTTP, a resposta a este pedido contém um código que poderá ser utilizado pelo sensor para ter a certeza de que as alterações foram submetidas com sucesso.

5.1.4 Subscrição de sensor / serviço

Uma vez que a arquitetura ReST não é orientada a notificações de eventos, fruto da utilização da arquitetura cliente-servidor associada ao protocolo HTTP (e por conseguinte, *stateless*), não existe ainda uma forma consensual para a implementação de mecanismos de notificação que permitam aos clientes receber notificações acerca da alteração do estado de um determinado recurso de interesse. Este é um ponto de especial importância no contexto M2M, uma vez que a informação só tem valor se puder ser facilmente acedida e rececionada em tempo útil dentro de um determinado contexto. O fornecimento desta funcionalidade numa arquitetura ReST pode ser realizada de duas formas distintas: por *polling* ou utilizando *websockets* (Secção 5.1.5).

5.1.5 Obtenção de dados

Como vimos na secção 5.1.4 e uma vez que a arquitetura ReST não é orientada a eventos, a subscrição de dados pode ser realizada através pedidos HTTP GET periódicos. Por exemplo, no caso da aplicação incidir sobre um cenário onde os sensores submetem informação para o servidor de forma periódica, a informação relativa a essa periodicidade pode ser utilizada pelos clientes que assim saberão quando realizar um novo pedido com vista à obtenção de informação atualizada. Por outro lado, quando as comunicações M2M se inserem num contexto em que determinada acção toma lugar de forma periódica e a sua execução não depende dos valores sensoriados por outros sensores ou seja, não é importante a receção de informação atualizada, um atuador poderá consultar as informações que desejar e só depois realizar determinada acção. Um sistema ReST por si só é suficiente para dar resposta a este tipo de cenários.

Para a realização de interações mais eficientes entre aplicações cliente e servidores, pode utilizar-se o *header HTTP If-Modified-Since* no pedido GET seguido de uma determinada data e hora. O servidor somente fornecerá a informação sobre o recurso ao cliente que tenha efetuado tal pedido caso o recurso em questão tenha sofrido alguma alteração desde a data especificada no pedido. Caso contrário, o servidor responderá com o código *304 (not modified)* sem nenhum corpo de mensagem. O objetivo desta funcionalidade do protocolo HTTP é permitir atualizações eficientes com recurso a uma menor quantidade de dados comunicada.

Por último, outra alternativa a este sistema inclui a utilização de *websockets* [I. 11] que mantêm um canal de comunicação aberto entre servidor e cliente para que após um pedido por parte do cliente e respetiva resposta por parte do servidor, este último possa utilizar o canal de comunicação para notificar o cliente de possíveis alterações no recurso sobre o qual o cliente manifestou interesse. Após a receção da notificação por parte do cliente (utilizando o canal de comunicação criado pelo servidor), este poderá realizar outro pedido ao servidor e o processo repete-se quantas vezes for necessário.

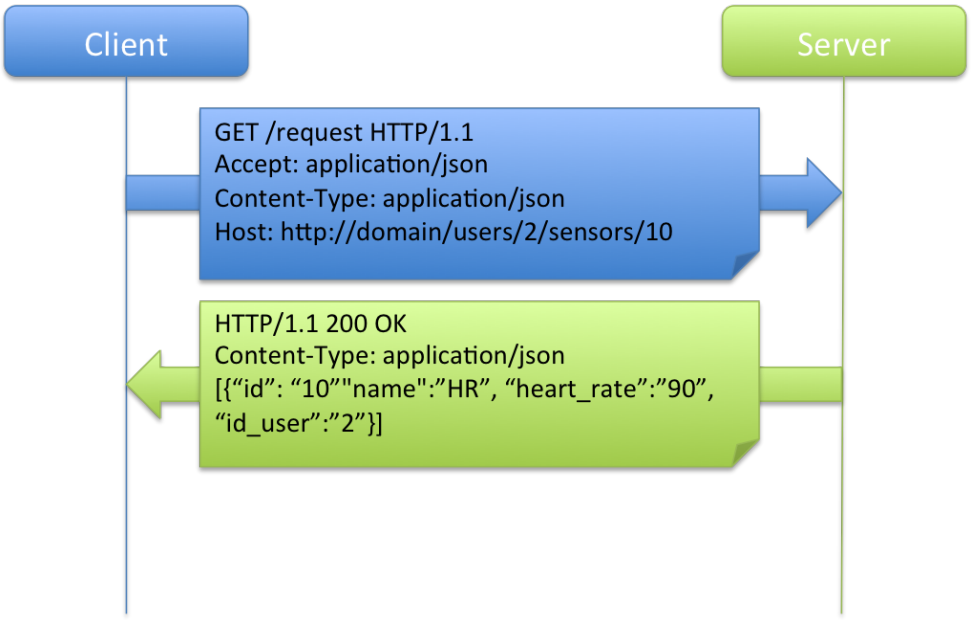


Figura 5.5: Trocas de mensagens relativas à obtenção de dados

A Figura 5.5 exemplifica a troca de mensagens realizada quando um cliente pretende obter informações acerca dos sensores registados num determinado serviço. O servidor responde com o código que informa o sucesso da operação seguido do corpo da mensagem (com as informações pedidas propriamente ditas).

5.2 Combinação ReST e AMQP

Nesta secção apresenta-se um sistema híbrido com arquitetura ReST e auxílio de um sistema de notificações que facilite a notificação de eventos, permitindo que um servidor ReST não fique sobrecarregado com funcionalidades orientadas a eventos, uma vez que os serviços poderão funcionar em máquinas/localizações diferentes. Escolheu-se o protocolo AMQP para funcionalidades de notificações. Sempre que seja necessário efetuar qualquer tipo de notificação, cabe ao servidor ReST publicar os dados no servidor AMQP para que este realize o trabalho relacionado com a distribuição de dados por todas entidades que demonstrarem interesse em rececionar tais informações.

Como mencionado anteriormente, o AMQP surge da necessidade de fornecer mecanismos de notificações para que todos os clientes que subscrevam determinada informação a possam receber de forma fiável e atualizada sem a necessidade de realizar pedidos HTTP GET periódicos. Na Secção 2.3 verificou-se a ausência destes mecanismos de notificações na arquitetura ReST e nesta secção utiliza-se o protocolo AMQP para a gestão de eventos associados a um ou mais recursos, tentando assim colmatar algumas lacunas patentes no estilo ReST.

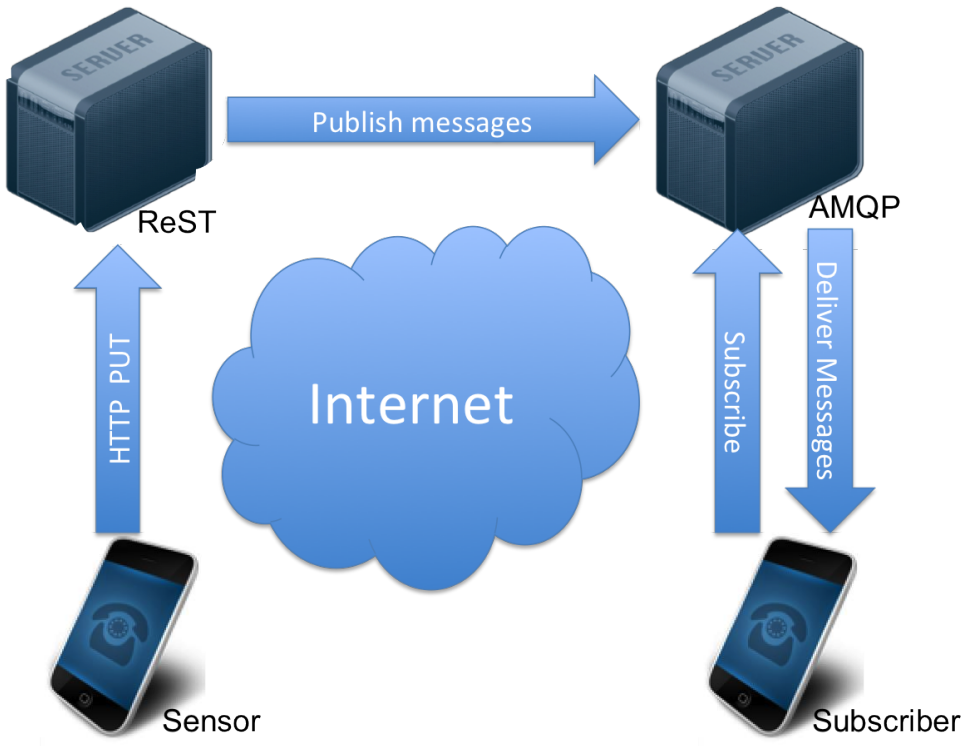


Figura 5.6: Arquitetura geral do sistema híbrido composto por servidor ReST e AMQP

No contexto M2M torna-se necessário adotar um mecanismo de notificações que permita a atualização em modo "*push*" de informação junto dos utilizadores. Assim o modelo ReST poderá ser utilizado como serviço de descoberta de sensores / serviços onde estão presentes as informações necessárias para a subscrição de determinado sensor, que no caso da utilização do protocolo AMQP, passa pelo fornecimento da informação que permita a um cliente subscrever as notificações associadas a um recurso, quer seja através do nome das filas de mensagens, quer seja através de chaves de roteamento (ver Secção 2.4.2). Uma aplicação cliente que deseje subscrever um determinado sensor necessita primeiro de o descobrir conforme descrito na secção 5.1.2. As informações fornecidas pelo servidor ReST acerca de um dado sensor são necessárias para que a aplicação cliente possa subscrever de facto o serviço ao invés de realizar pedidos HTTP GET periódicos como acontece no caso de apenas existir um servidor ReST.

Nas secções seguintes apresenta-se o comportmento de uma solução híbrida na realização das funcionalidades enunciadas no início deste capítulo. Os fluxos de troca de mensagens foram extraídos com o programa *Wireshark*[1] de forma a capturar os pacotes trocados na rede, fruto da utilização do protocolo AMQP, para uma publicação de uma mensagem e subscrição da mesma. Esta decisão surgiu da falta de documentação acerca das trocas de mensagens ao nível das camadas de rede e aplicação e serve para exemplificar a troca de mensagens segundo este protocolo. Convém notar que as informações apresentadas a seguir podem ser ligeiramente diferentes nou-

[1] O software *Wireshark* está disponível para *download* em http://www.wireshark.org/

tros casos uma vez que dependem de várias decisões de implementação (tipo de roteamento de mensagens por exemplo).

5.2.1 Anúncio de sensor / serviço

Em concordância com o que foi apresentado na secção 5.1, um sensor que deseje registar as suas informações num serviço, fá-lo-á através de um pedido HTTP POST. Desta forma, o servidor ReST possuirá todas as informações associadas a um determinado sensor e poderá fornecer às aplicações clientes várias representações (em diversos formatos) desse mesmo sensor, caso estas aplicações enviam um pedido HTTP GET ao URL que correspondente ao sensor no servidor ReST.

Em semelhança com o que acontece com a utilização apenas de tecnologia ReST, o número de bytes necessários para realizar esta funcionalidade segue a Equação 5.1, sendo que o tamanho do pedido corresponde ao tamanho de um pedido HTTP POST.

5.2.2 Descoberta de sensor / serviço

A utilização do modelo ReST permite a obtenção das informações associadas a um dado recurso. Em semelhança com o que se passa no caso de apenas se utilizar o modelo ReST, as informações acerca de um recurso podem ser acedidas através de um simples pedido HTTP GET ao recurso, que por sua vez fornecerá todas as informações disponíveis para subscrever um dado sistema de notificação (AMQP, XMPP ou outros), bastando para isso que essas informações estejam disponíveis na base de dados associada ao serviço. No caso do protocolo XMPP, o *JabberID* é suficiente para realizar a subscrição de determinado recurso, enquanto que no caso do protocolo AMQP podem fornecer-se o nome da fila de mensagens associada a cada sensor ou ainda o nome do *exchange* e respetiva chave de roteamento. Dependendo dos mecanismos de notificações associados ao servidor ReST, basta apenas disponibilizar as informações necessárias para a subscrição desses sistemas.

Como para a realização desta funcionalidade apenas se recorre a transações ReST, também a descoberta de sensores e serviços segue a Equação 5.1 para cada consulta efetuada ao sistema (através de um pedido HTTP GET).

5.2.3 Publicação de dados

A publicação de dados utilizando um modelo híbrido composto por um serviço ReST e um serviço que utilize o protocolo AMQP ganha especial atenção no caso de cenários em que se pretende monitorizar um dispositivo.

A Figura 5.7 exemplifica a troca de mensagens quando um produtor de mensagens decide enviar uma mensagem para o *broker*. As mensagens trocadas entre cliente e servidor sobre o fundo verde visam o estabelecimento da conexão com o servidor. Na zona de subscrição realiza-se a criação de canais e filas de mensagens para publicação de mensagens. Estas especificações são muito úteis pois graças à sua flexibilidade, o AMQP permite que o produtor decida qual o tipo de difusão de mensagens que deseja utilizar e tipos de filas, entre outros. Apenas as mensagens

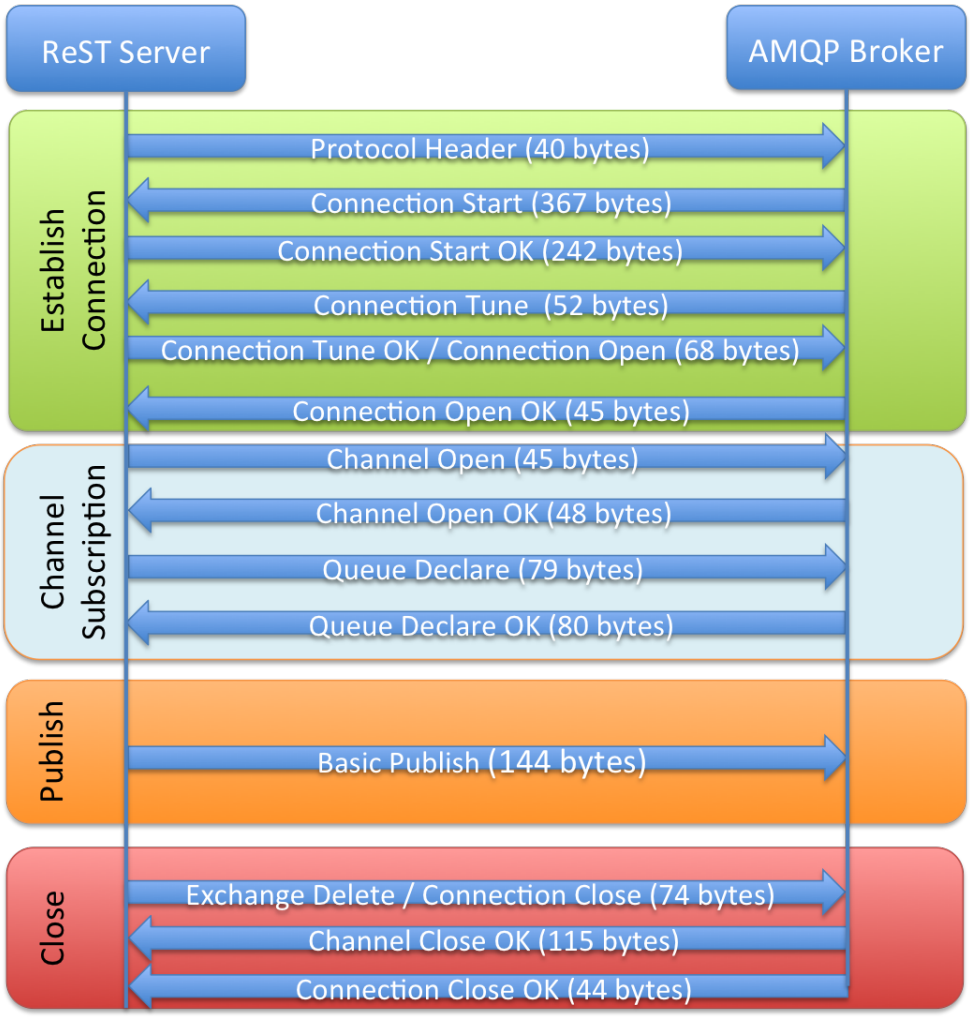


Figura 5.7: Troca de mensagens representativa da publicação de dados por parte do servidor ReST junto do *broker* AMQP

a zona *publish* é que transferem de facto os conteúdos para o servidor. As últimas mensagens trocadas visam o fecho da conexão. Cabe ao servidor ReST publicar os dados no servidor AMQP. Esta operação realiza-se o número de vezes que um sensor necessitar de atualizar os seus dados no servidor ReST. A publicação de dados de um serviço / sensor é igual ao caso ReST.

Uma vez que a publicação de dados no *broker* AMQP é realizada pelo servidor ReST (que desempenha o papel de *publisher*), o tamanho da transação necessária para a realização desta tarefa do ponto de vista da aplicação cliente segue mais uma vez a Equação 5.1, sendo que o tamanho do pedido corresponde ao tamanho de um pedido HTTP PUT.

5.2.4 Subscrição de sensor / serviço

Na Figura 5.7 realizou-se uma análise do fluxo de mensagens quando o servidor ReST deseja publicar informações no *broker* AMQP. De seguida irá analisar-se a troca de mensagens quando

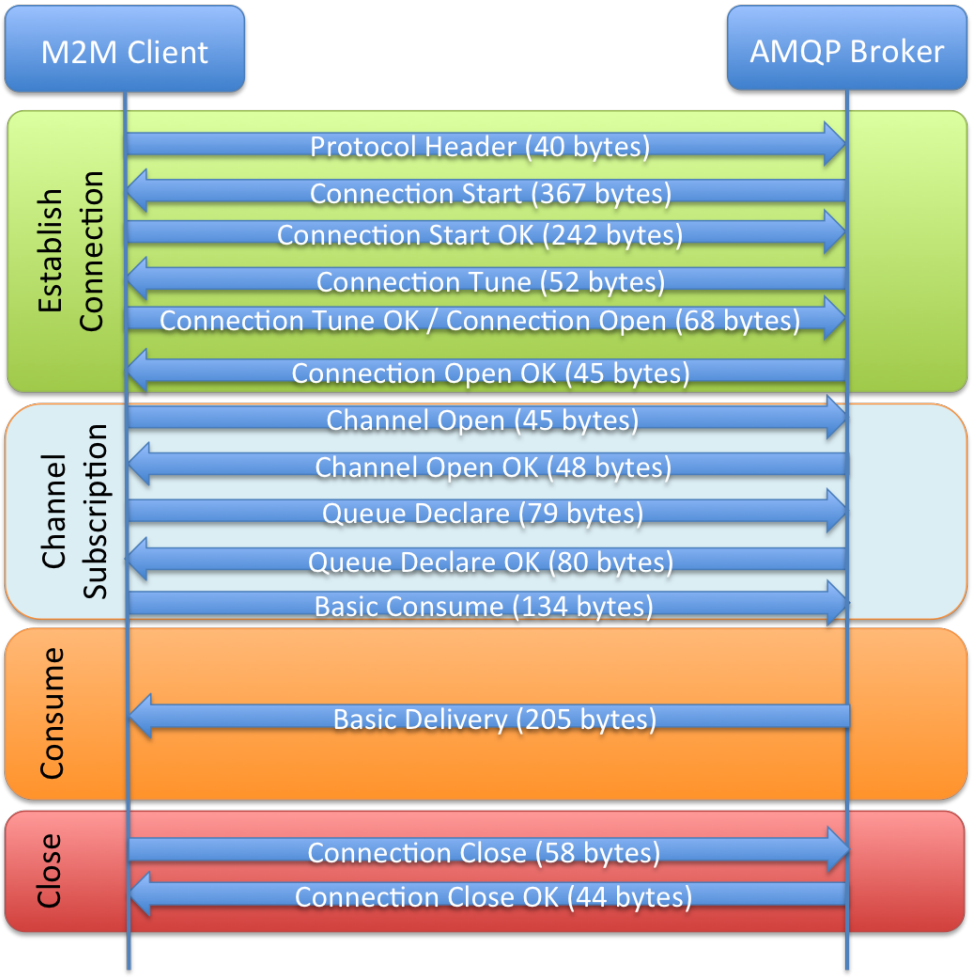


Figura 5.8: Fluxo de mensagens entre um *subscriber* e um *broker* AMQP

uma entidade subscreve as informações fornecidas por outra entidade.

Na Figura 5.8, apresentam-se as trocas de mensagens associadas a uma subscrição realizada por parte de uma aplicação cliente junto do *broker* AMQP. Da mesma forma como na análise da Figura 5.7 para a publicação de dados, são necessárias as fases de estabelecimento da conexão e estabelecimento de canais para realizar a subscrição. A diferença surge nas mensagens trocadas entre este estabelecimento e fecho de conexão para declarar a intenção de consumir dados. Neste caso, o cliente primeiro envia uma mensagem AMQP *Basic Consume* ao servidor que por sua vez envia um TCP ACK, estabelecendo um canal de comunicação que será utilizado pelo servidor para enviar notificações ao cliente. A mensagem enviada pelo servidor (*Basic Delivery*) contém os *headers* da mensagem assim como o seu corpo (com as informações propriamente ditas). Quando o cliente recebe a notificação, é ele próprio quem envia um TCP ACK ao servidor para aguardar por novas notificações. De notar que a nível da aplicação, utilizando o protocolo AMQP pode-se optar pela não realização de confirmações acerca as mensagens transferidas, baseando-se na entrega confiável do protocolo de transporte TCP caso seja desejado.

Uma vez que a subscrição é realizada pela aplicação cliente ao servidor AMQP, o número de bytes necessários para esta transação segue a Equação 5.2, sendo igual à soma dos bytes necessários para o estabelecimento da conexão (caso esta ainda não exista) e o tamanho da mensagem "*Basic Consume*" assim como a quantidade de dados necessária para a realização do fecho da conexão (omitido na Equação 5.2 uma vez que é boa prática mas não fundamental para a realização desta funcionalidade).

$$L_{transaction} = L_{connection+channelsubscription} + L_{basicconsume} \tag{5.2}$$

5.2.5 Obtenção de dados

A distribuição de dados segundo o protocolo AMQP é feita utilizando a ligação TCP estabelecida durante a fase "*consume*"(mensagens trocadas sobre fundo amarelo na Figura 5.8).

Em relação ao tipo de roteamento das mensagens e distribuição das mensagens para os possíveis destinatários, o protocolo AMQP pode realizar esta tarefa de diferentes formas, sendo estas decisões de implementação realizadas do lado do *publisher* (servidor ReST) e do lado do *subscriber*. Estas variantes nas entidades AMQP conferem ao protocolo uma elevada flexibilidade quando se trata de implementar e gerir a distribuição de dados ao nível da aplicação. Desta forma o roteamento de mensagens deve ser definido para cada implementação específica, de acordo com as necessidades.

Supondo que a conexão não foi estabelecida previamente ou se perdeu devido à mobilidade associada a um sensor, para a receção de dados é necessário adicionar uma parcela à Equação 5.2 referente à entrega das mensagens por parte do servidor AMQP junto do cliente. Assim, a equação que possibilita o cálculo dos bytes necessários para a obtenção de dados segue a Equação 5.3.

$$L_{transaction} = L_{connection+subscription} + L_{basicdelivery} \tag{5.3}$$

$$L_{transaction} = L_{basicdelivery} \tag{5.4}$$

Caso a conexão com o *broker* AMQP já esteja estabelecida e a subscrição efetuada, a Equação 5.3 dá lugar à Equação 5.4.

5.3 XMPP

A tecnologia XMPP funciona de forma fundamentalmente diferente das tecnologias/protocolos propostos nas secções anteriores deste capítulo. Esta tecnologia atribui identificadores únicos aos utilizadores (*Jabber ID*) que permitem estabelecer a comunicação entre todas as partes. Na sua utilização para suporte de comunicações M2M, os sensores associados a utilizadores comunicam com outros sensores e com serviços, enviando-lhes mensagens acerca do seu estado. Utilizando o protocolo XMPP, os dados são transportados em formato XML. Nesta secção serão identificadas

quais as trocas de mensagens necessárias para implementar cada uma das funcionalidades necessárias, em semelhança com o que aconteceu anteriormente neste capítulo com as outras soluções.

5.3.1 Anúncio de sensor / serviço

De forma análoga a sistemas de mensagens instantâneas nos quais é possível atribuir um estado à presença de cada entidade na rede e esta é difundida por todos os contatos dessa entidade, no caso dos sensores é utilizado o mesmo mecanismo de difusão de informações de presença na rede. Um sensor inteligente que utilize o protocolo XMPP para comunicar apenas necessita de estabelecer a comunicação com o *broker* de mensagens e enviar-lhe a informação que diz respeito ao seu estado. Por sua vez, o *broker* trata de realizar a difusão desta informação de presença para todas as entidades que registaram interesse subscrevendo esse sensor. Esta troca de mensagens encontra-se representada na Figura 5.9 (não é apresentada a receção da informação de presença por parte de terceiros). Para o estabelecimento da conexão são trocadas os dois *stanzas* iniciais e de seguida é enviado um *stanza* de presença.

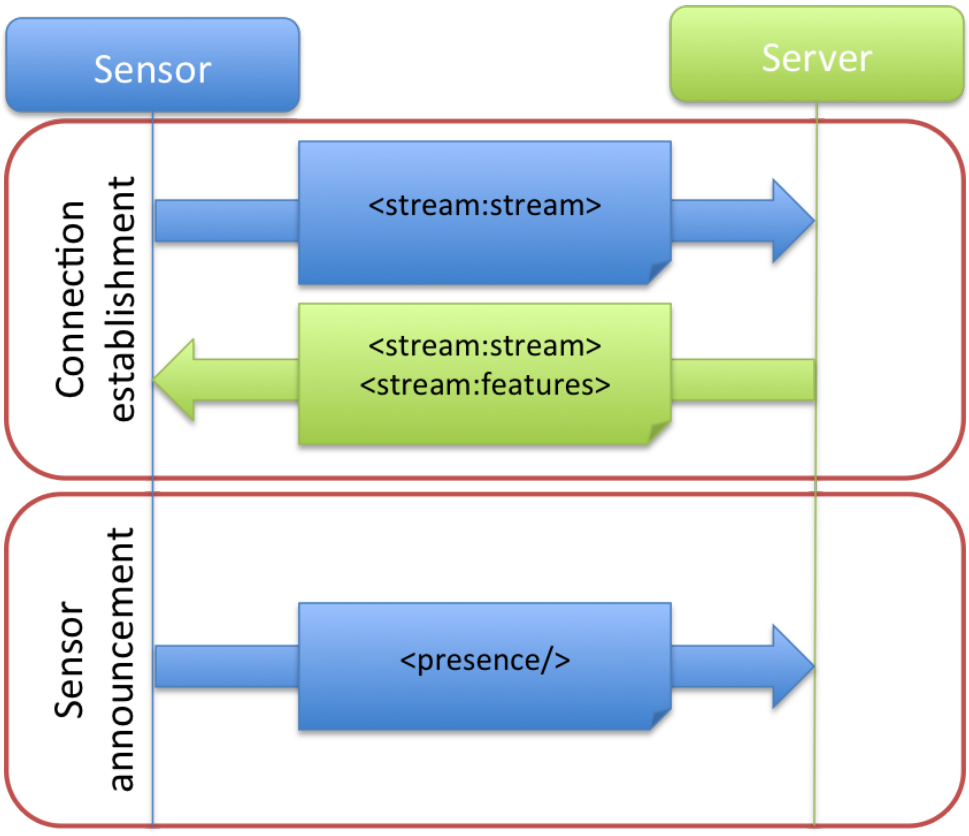


Figura 5.9: Anúncio de sensor utilizando XMPP

A equação que traduz esta funcionalidade em bytes relativos às transações associadas é a Equação 5.5 e corresponde ao tamanho dos pacotes utilizados para o estabelecimento da conexão somado ao tamanho do *stanza* de presença.

$$L_{transaction} = L_{connection} + L_{presence} \tag{5.5}$$

$$L_{transaction} = L_{presence} \tag{5.6}$$

Mais uma vez, caso a conexão com o *broker* XMPP já esteja estabelecida, a Equação 5.5 dá lugar à Equação 5.6.

5.3.2 Descoberta de sensor / serviço

Conforme referido em [DJ 03], não existe nenhuma parte na especificação do protocolo utilizado pelo XMPP que determina claramente como um potencial subscritor de informação de uma dada entidade pode encontrá-la.

5.3.3 Publicação de dados

Após a observação acerca do que teoricamente acontece quando se regista um sensor no sistema de comunicação, interessa analisar como se transmite uma mensagem entre duas entidades XMPP, exemplificado na Figura 5.10.

Após a fase inicial de estabelecimento da conexão (representada pelas duas primeiras trocas entre sensor e servidor) o sensor envia um *stanza* de mensagem ao *broker*. Este *stanza* especifica o remetente e o destinatário da mensagem, o tipo em que se enquadra e finalmente o corpo da mensagem. Quando o *broker* recebe a mensagem trata de a encaminhar para os destinatários.

A Equação 5.7 traduz o número de bytes necessários para a publicação de dados. Sempre que ainda não exista uma conexão com o *broker* XMPP, é necessário estabelecer essa conexão e depois enviar um *stanza message* com o destinatário dessa publicação.

$$L_{transaction} = L_{connection} + L_{message} \tag{5.7}$$

5.3.4 Subscrição de sensor / serviço

A mensagem apresentada nesta secção apresenta uma mensagem enviada ao serviço pela entidade que pretende subscrever um dado utilizador. Este pedido é um *stanza* iq onde existe um e só um elemento <*subscribe*> que por sua vez deve possuir um atributo nó que especifique qual entidade se deseja subscrever. Note-se que o parâmetro destinatário é *harcoded* uma vez que não existe um mecanismo de descoberta de recursos associado a este protocolo. Juntamente com a informação do nó, tem de ser enviado também o *JabberID* da entidade subscritora.

```
1 <iq type='set'
2   from='francisco@denmark.lit/barracks'
3   to='pubsub.shakespeare.lit'
4   id='sub1'>
```

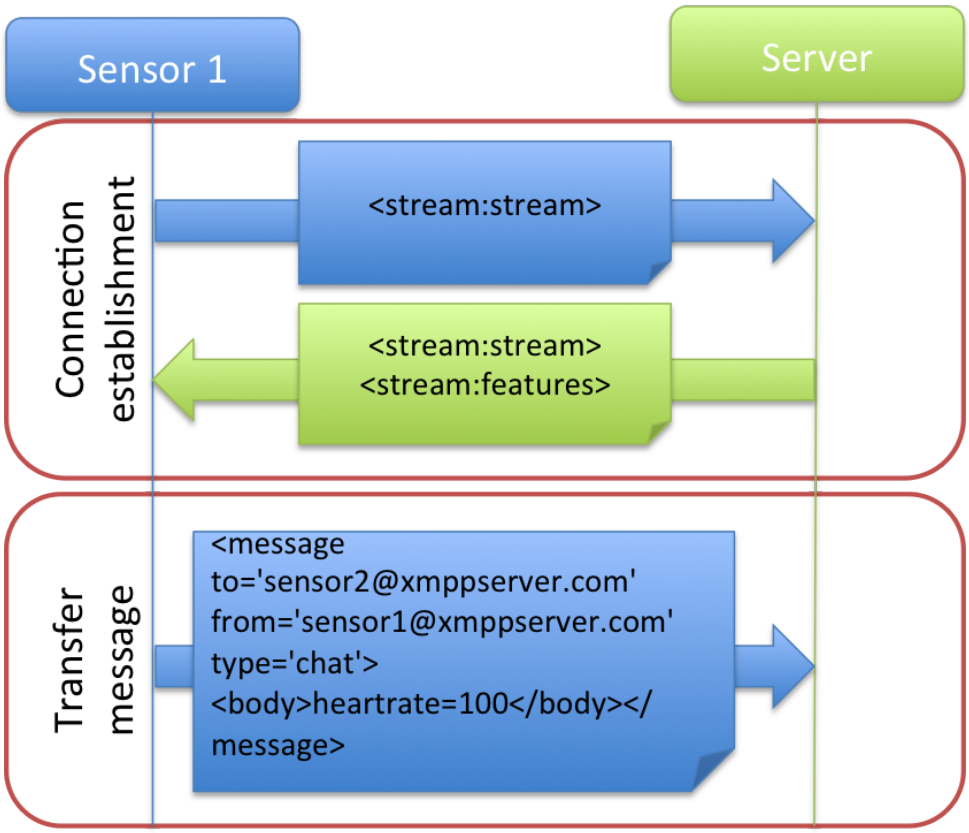


Figura 5.10: Publicação de dados por parte de uma entidade utilizando o protocolo XMPP

```
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscribe
      node='princely_musings'
      jid='francisco@denmark.lit'/>
  </pubsub>
</iq>
```

A mensagem que o servidor envia como resposta ao pedido inicial (em caso de sucesso) é apresentada apresentada a seguir. É função do servidor informar a entidade que realizou o pedido de subscrição que se encontra subscrita (o que pode incluir um sub identificador gerado pelo serviço). Para mais informações consultar [MSAM12].

```
<iq type='result'
  from='pubsub.shakespeare.lit'
  to='francisco@denmark.lit/barracks'
  id='sub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <subscription
      node='princely_musings'
      jid='francisco@denmark.lit'
```

```
9       subid='ba49252aaa4f5d320c24d3766f0bdcade78c78d3'
10      subscription='subscribed'/>
11    </pubsub>
12  </iq>
```

Supondo que não existe uma conexão estabelecida entre o cliente e o *broker* XMPP, a Equação 5.8 traduz o número de bytes necessários para a realização desta funcionalidade. Nela são somados o tamanho das mensagens relacionadas com o estabelecimento da conexão aos tamanhos das mensagens *iq request* e *result*.

$$L_{transaction} = L_{connection} + L_{iq-request} + L_{iq-result} \tag{5.8}$$

5.3.5 Obtenção de dados

Na secção anterior analisou-se o processo de envio de mensagens de uma entidade para um servidor de mensagens utilizando o protocolo XMPP. Nesta secção é possível perceber como funciona o mecanismo de distribuição de mensagens. Um exemplo ilustrativo deste mecanismos é apresentado na Figura 5.11.

Após a fase de conexão inicial o servidor reencaminha a mensagem para o seu destinatário. Este poderá estar conetado ao mesmo *broker* ou poderá estar ligado noutro servidor que utilize o protocolo XMPP. Isto não constitui um problema uma vez que os servidores comunicam entre si para que seja possível a comunicação entre duas entidades que estejam conetadas em servidores distintos (funcionamento em modo federado).

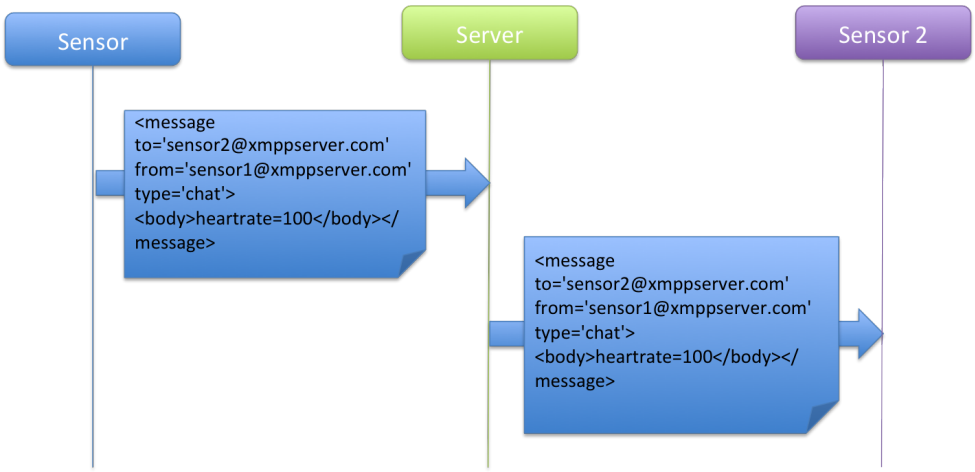


Figura 5.11: Exemplo da obtenção de mensagens utilizado pelo protocolo XMPP

Em semelhança com o que acontece na Secção 5.3.3 a obtenção de dados segue a mesma lógica do envio de dados, mas neste caso é a aplicação cliente que recebe os dados provenientes de outro utilizador. Assim, para esta funcionalidade as trocas de mensagens associadas seguem o definido pela Equação 5.7.

5.4 Análise quantitativa

Esta secção finaliza o presente capítulo apresentando um conjunto de caraterísticas e informações que permite estabelecer uma comparação entre as tecnologias objeto de estudo ao longo do capítulo.

Os protocolos estudados utilizam todos um canal TCP para comunicação. No caso do ReST, a utilização de HTTP (protocolo da camada de aplicação) pressupõe a existência de um canal ao nível da camada de transporte que seja fiável e garanta a entrega ordenada de pacotes, daí a predominância de canais TCP.

Quanto à necessidade de estabelecimento de ligações, ReST graças à utilização do protocolo HTTP não requer o estabelecimento de ligações na camada de aplicação. O mesmo já não acontece nos casos do protocolo AMQP e XMPP, uma vez que estes requerem uma quantidade significativa de trocas de mensagens para o estabelecimento do canal de comunicação ao nível da camada de aplicação.

O modelo de transações associado à tecnologia ReST é o modelo *request-response* no qual a aplicação cliente envia um pedido ao servidor, e o último envia-lhe as informações como resposta. Quanto aos protocolos AMQP e XMPP, o estabelecimento de ligação funciona também num modelo *request-response* ao passo que a publicação de dados e notificações funciona segundo o modelo *one way* ou seja, o pacote de informações viaja num só sentido e não existe uma resposta ao nível da aplicação, confiando nas garantias de entrega de pacotes de forma ordenada associado ao protocolo TCP.

As métricas definidas no âmbito deste trabalho influenciam diretamente a performance das soluções. No caso da quantidade de dados, estes valores influenciam o consumo de energia e a largura de banda disponível para outras funcionalidades. Quanto aos *round trips*, quanto mais existirem na execução de uma funcionalidade, maior é o atraso na resposta aos pedidos efetuados.

Funcionalidade	**Tamanho da transação**	**Round-Trips**
Anúncio de sensor	295 bytes	1
Descoberta de sensores	960 bytes	2
Publicação de dados	206 bytes	1
Subscriçao de sensor	-	-
Obtenção de dados	253 bytes	1

Tabela 5.1: Quantidade de dados e *round-trips* necessários para a realização de cada funcionalidade utilizando ReST.

A Tabela 5.1 apresenta os tamanhos das trocas de mensagens assim como os *round trips* necessários para a execução das funcionalidades identificadas. De notar que na descoberta de sensores realizaram-se dois pedidos distintos (2 *round trips*), sendo que o primeiro visa encontrar os utilizadores registados no sistema e o segundo visa a obtenção de informações relativas aos sensores associados a um determinado utilizador. Para a obtenção de dados utilizando o modelo ReST basta apenas efetuar um pedido HTTP GET ao recurso prentendido.

Funcionalidade	Tamanho da transação	Round-Trips
Anúncio de sensor	295 bytes	1
Descoberta de sensores	960 bytes	2
Publicação de dados	206 bytes	1
Subscriçao de sensor	1200 bytes	5
Obtenção de dados	1405 bytes	6

Tabela 5.2: Quantidade de dados e *round-trips* necessários para a realização de cada funcionalidade utilizando uma combinação ReST - AMQP.

A Tabela 5.2 é idêntica à Tabela 5.1 excetuando nas funcionalidades de subscrição e obtenção de dados. Os valores apresentados contemplam o estabelecimento de uma conexão e a obtenção de dados relativos a um determinado sensor, daí serem elevados.

Em relação ao protocolo XMPP, não foi possível identificar claramente as várias funcionalidades na aplicação objeto de estudo e fruto da segurança TLS não foi possível captar o conteúdo das mensagens envolvidas.

Tecnologias/ Funcionalidades	ReST	ReST + AMQP	XMPP
Anúncio de sensor	HTTP POST	HTTP POST	Est. conexão + stanza "presence"
Descoberta de sensor	HTTP GET	HTTP GET	-
Publicação de dados	HTTP PUT	HTTP PUT	Est. conexão + stanza "message"
Subscrição de dados	-	Est. conexão + subscrição	Est. conexão + stanza "iq request" + stanza "iq response"
Obtenção de dados	HTTP GET periódicos	Est. conexão + subscrição + obtenção	Est. conexão + stanza "message"

Figura 5.12: Tabela resumo da adaptação das tecnologias às funcionalidades definidas

Por último, a Figura 5.12 apresenta o conjunto de funcionalidades definidas assim como a forma como cada tecnologia (identificada nas colunas) pode realizar tal funcionalidade.

Capítulo 6

Desenvolvimento de plataforma M2M

Como já referido nesta dissertação, o objetivo principal do trabalho é o desenvolvimento de uma plataforma M2M que opera sobre o cenário de *healthcare*. Além disso, para realizar uma comparação objetiva e mensurável entre a plataforma desenvolvida no âmbito desta dissertação e uma outra plataforma proprietária já existente, surge a necessidade de desenvolvimento da primeira para que dela se possam extrair informações concretas acerca das mensagens envolvidas no processo comunicacional. Essas informações serão comparadas com outra plataforma já desenvolvida (que utiliza o XMPP como protocolo de comunicação) para que seja possível no final, estabelecer uma comparação entre ambas. No decorrer deste capítulo serão apresentados os vários módulos que constituem a plataforma desenvolvida e como funciona a comunicação entre os seus vários componentes.

6.1 Visão geral

Antes de uma descrição dos módulos que fazem parte da plataforma desenvolvida, é importante apresentar a arquitetura geral da aplicação assim como a interação entre os diversos módulos que a constituem.

Na imagem 6.1 é apresentada uma visão geral da arquitetura da solução a implementar. Esta solução foca-se no cenário de cuidados de saúde e utiliza fundamentalmente o estilo arquitetónico ReST. As caraterísticas intrínsecas a este estilo de arquitetura como a interoperabilidade, escalabilidade, extensibilidade, a utilização de standards abertos bem conhecidos, a possibilidade de obter várias representações para um determinado dispositivo (consoante seja mais útil) e a utilização de uma interface simples e uniforme (HTTP) levaram a que este estilo fosse escolhido para implementar uma arquitetura de serviços. Opcionalmente e para a implementação de mecanismos de notificação, utilizar-se-á o protocolo AMQP como solução MOM.

O servidor ReST permite guardar informações sobre os utilizadores e respetivos sensores numa base de dados para que possam ser posteriormente consultados e utilizados pela aplicação cliente, conforme referido na Secção 5.1. Além disso, estabelece uma conexão com o servidor AMQP que utiliza sempre que necessite de publicar mensagens que serão consumidas pelas aplicações

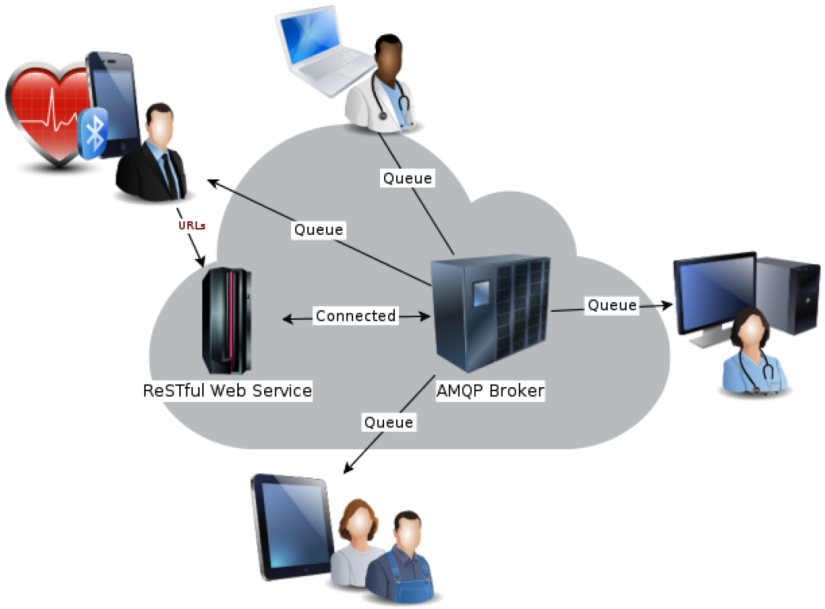


Figura 6.1: Arquitetura geral da solução a implementar

cliente que subscreverem notificações. O servidor AMQP fornece mecanismos de subscrição a aplicações clientes que desejem subscrever informações relacionadas com determinado utilizador e sensor. A aplicação cliente estabelece comunicação com o servidor ReST para o anúncio de um sensor, descoberta de utilizadores e sensores registados no servidor e ainda publicação de dados. Outra funcionalidade da aplicação cliente está relacionada com a subscrição de notificações do servidor AMQP. Estes componentes serão apresentados em maior detalhe nas próximas secções.

6.2 Servidor ReST

Este componente desempenha um papel central na plataforma desenvolvida. Além de beneficiar de todas as vantagens associadas ao estilo arquitetural ReST (já enumeradas no capítulo 2.3), o servidor também estabelece uma conexão com o servidor AMQP e utiliza-a sempre que quiser delegar tarefas relacionadas com notificação de eventos.

De seguida é apresentada uma lista das funcionalidades implementadas pelo servidor ReST que representa uma parte fundamental da plataforma, uma vez que é este servidor que guarda todos os dados necessários para uma interação entre os vários componentes constituintes da plataforma.

Funcionalidades do servidor:

- Descoberta de recursos segue o padrão ReST;
- Operações "*Create, Read, Update e Delete*" (CRUD) disponíveis para os recursos implementados.

- Interligação com o *broker* AMQP para notificação de eventos (apresentada na secção seguinte);
- Possibilidade do servidor receber pedidos e fornecer informação aos clientes em vários formatos (HTTP e JSON);
- Validações básicas de campos específicos associados a determinado recurso (presença e unicidade).

Das funcionalidades apresentadas, as primeiras 3 serão explicadas nas secções seguintes.

6.2.1 Recursos

O serviço implementado possui apenas dois recursos distintos: utilizadores e sensores. Quanto aos utilizadores, é possível guardar os dados pessoais associados a estes (nome e idade), existindo ainda um campo denominado *"exchange name"* que se encontra relacionado com o protocolo AMQP. De forma análoga, o serviço também permite registar informações básicas dos sensores (nome e contexto em que se insere) além de permitir guardar informações acerca dos dados percepcionados pelo sensor (no campo *"sensor info"*). Também neste recurso existe um campo de dados relacionado com a conexão ao servidor AMQP (a chave de roteamento). Ambos os recursos permitem ainda guardar a data de registo de determinado recurso assim como a data da última alteração, o que possibilita aos clientes saber quando foi realizada a última publicação de dados por parte de determinado sensor.

A relação existente entre os recursos enunciados é de "um-para-muitos" ou seja, cada utilizador pode possuir zero ou mais recursos associados, no entanto o contrário já não se verifica, pois cada sensor registado no sistema possui um e apenas um utilizador associado. Esta relação está presente no "*model*" (da arquitetura MVC) de ambos os recursos.

Foram implementadas também algumas validações básicas para o registo de dados no serviço. Quando se pretende registar um utilizador, o serviço verifica a presença do primeiro nome, não podendo este exceder 100 caractéres. Além disso e mais importante, o serviço verifica a presença e unicidade do *"exchange"* gerado pelo próprio serviço (fundamental para a integração com o serviço AMQP). No caso do registo de um novo sensor as mesmas validações são aplicadas aos campos nome e chave de roteamento (unicidade). Tanto o campo *"exchange"* como a chave de roteamento associada a cada sensor são gerados pelo servidor ReST no momento do anúncio de determinado utilizador ou sensor e são eles que permitem ao servidor ReST publicar mensagens no servidor AMQP. Em semelhança com o que acontece com as relações, também estas validações são especificadas no modelo de ambos os recursos.

Um outro ponto de especial importância está relacionado com a capacidade do servidor responder a pedidos em vários formatos ou seja, se for especificado no pedido o formato de dados que se pretende obter na resposta por parte do servidor, este responde em conformidade com o pedido. Se por exemplo, algum utilizador estiver a utilizar um browser e desejar consultar um determinado recurso, o servidor renderiza uma página *web* (HTML) com a informação relativa ao pedido

efetuado pelo utilizador. Se por outro lado, uma aplicação cliente realizar a mesma consulta, pode especificar no seu pedido que pretende receber as informações em formato JSON, e o servidor atenderá ao seu pedido. No processo de comunicação é fundamental reduzir o tamanho de dados rececionados, daí a utilização do formato JSON. Em vez do cliente receber todo o ficheiro HTML, é muito mais eficiente requerer os mesmos dados no formato JSON, uma vez que estes são apenas uma representação do(s) recurso(s) que a aplicação cliente pretende receber, e não incluem estilos nem elementos HTML desnecessários do ponto de vista da aplicação cliente. Além disso, optou-se pelo formato JSON en detrimento de outras linguagens estruturadas de dados (como o XML), uma vez que a plataforma Android já possui nativamente mecanismos para interpretar este formato de mensagens. Foram implementados dois formatos de resposta no servidor desenvolvido: HTML e JSON. Isto facilita bastante a interpretação de dados por parte dos clientes do serviço além de conferir uma maior flexibilidade ao serviço ReST.

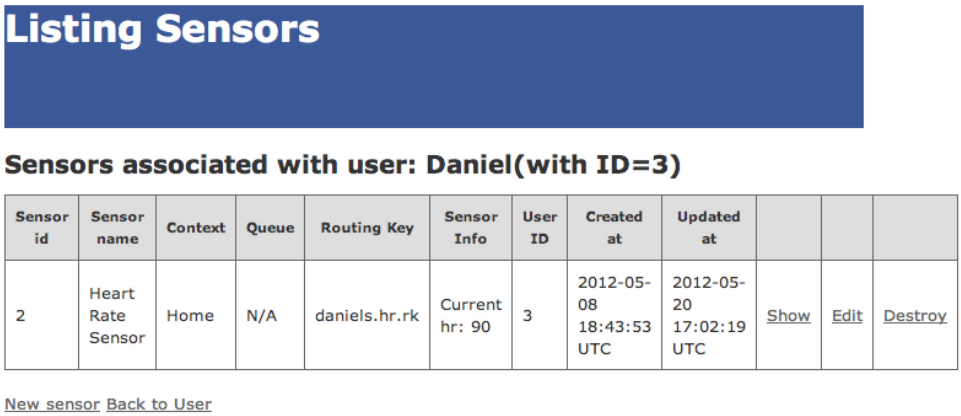


Figura 6.2: Tabela de sensores associados a um dado utilizador

Na Figura 6.2 apresenta-se uma tabela de sensores associados a um determinado utilizador. Observam-se várias colunas referentes aos sensores e seu contexto assim como campos que remetem para o sistema de subscrições implementado.

6.2.2 Descoberta de recursos

O desenvolvimento deste serviço foi de encontro ao sugerido pelo estilo arquitetónico ReST quanto à descoberta de recursos. Assim, utiliza-se o método HTTP GET para a consulta de recursos, sejam estes utilizadores ou sensores associados, como referido na Secção 5.1.2. Como mencionado anteriormente, é possível ao cliente especificar o formato de dados que pretende receber (HTTP ou JSON).

6.2.3 Integração com AMQP

Como já foi dito, a integração com um *broker* AMQP parte da necessidade de implementar um sistema de notificações eficiente que possa por si só, tratar de toda a lógica relacionada

com a publicação, distribuição e entrega de mensagens. Como observado em secções anteriores, utilizadores e sensores possuem campos específicos necessários para realizar tal integração com este sistema de notificação de eventos. No caso dos utilizadores, estes possuem o campo *"exchange name"* enquanto que os sensores apresentam o campo *"routing key"*. Este par *"exchange name/routing key"* permite ao servidor ReST publicar mensagens associadas a um determinado utilizador e sensor no *broker* (através do protocolo AMQP) na ocorrência de um determinado evento. No contexto AMQP, o servidor ReST representa os *"Publishers"* que publicam mensagens no servidor que depois tratará do seu roteamento e posterior entrega a subscritores dessas mesmas informações.

Na solução desenvolvida, quando o servidor ReST processa um pedido de modificação da informação associada a determinado sensor, este publica também uma mensagem no *broker* AMQP. Desta forma, os sensores apenas necessitam de enviar um pedido HTTP PUT ao servidor ReST, para que este realize o resto do trabalho. O servidor ReST quando inicia, cria uma conexão ao broker AMQP que utiliza sempre que necessite de publicar dados, evitando desta forma o estabelecimento de várias conexões com o *broker* AMQP. O código de estabelecimento de conexão deverá estar presente num ficheiro dentro da pasta "config/initializers". Todo o código colocado nessa pasta executa quando o servidor ReST inicia.

Existem inúmeras vantagens associadas a esta divisão de competências entre servidor ReST e AMQP no entanto, vale a pena destacar que do ponto de vista dos sensores, a troca de mensagens ilustrada na Figura 5.7 é substituída por um mero pedido HTTP PUT. Para além disso, é possível distribuir os vários compoentes da plataforma pela rede de comunicação, cada qual com as suas funcionalidades, o que permite uma mais rápida identificação e resolução de qualquer problema que surja na plataforma.

6.3 Servidor AMQP

Mais uma vez, a presença deste servidor de mensagens na plataforma desempenha o papel de gestão de notificações da plataforma. Além de ser um protocolo bastante flexível, que permite aos produtores de mensagens e seus subscritores definirem a forma como todo este sistema se comporta, isto é, como as mensagens são entregues aos clientes, com a sua utilização é também possível distribuir o processamento dos pedidos por vários servidores, além de permitir a construção de uma plataforma modular, na qual cada servidor realiza determinadas tarefas de forma isolada. Com este isolamento, é possível distribuir a comunicação por vários servidores, sendo possível também facilmente implementar funcionalidades adicionais sem para isso correr-se o risco de alterar inadvertidamente o comportamento de outras partes do sistema.

Como foi referido anteriormente, existem vários tipos de entidades AMQP: o servidor ReST representa o papel de *"publisher"* que envia mensagens para um determinado *"exchange"* com uma chave de roteamento para que o servidor possa reencmainhar as mensagens para as respetivas filas. Os clientes (desenvolvidos para a plataforma móvel Android) representam os subscritores

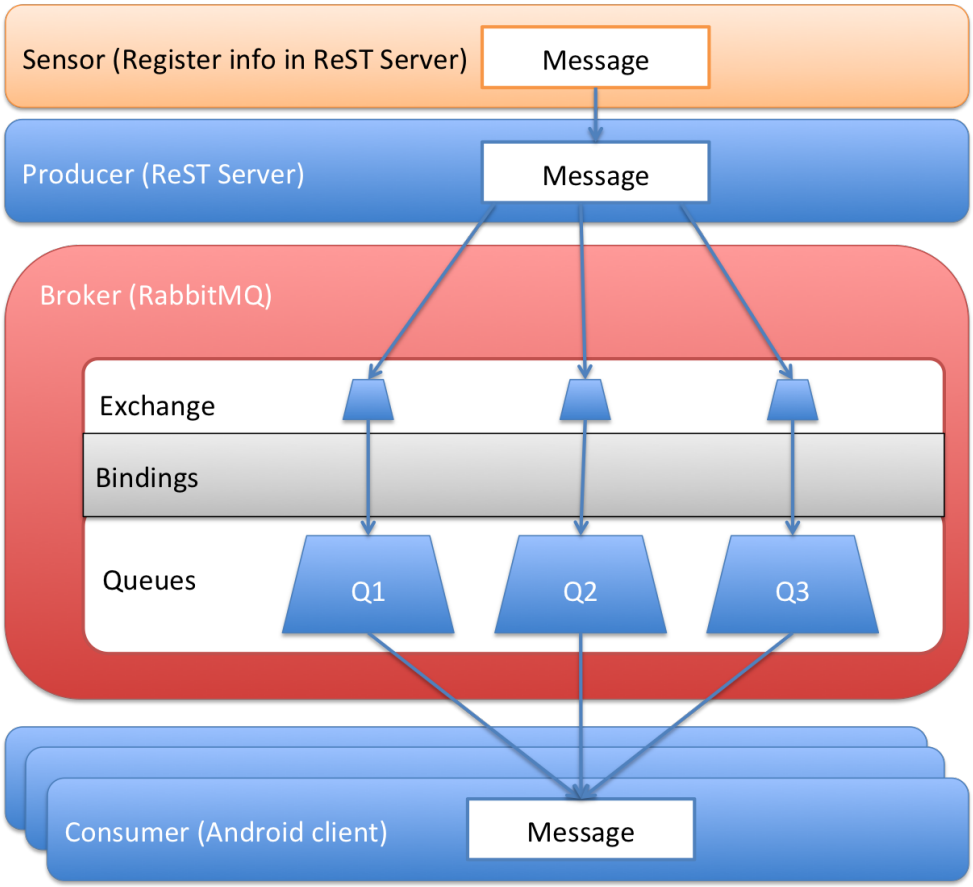


Figura 6.3: Identificação das entidades AMQP

que enviam ao servidor uma intenção de subsrever um conjunto de notificações referentes a determinado recurso (utilizando para isso os dados presentes no servidor ReST (*exchange name* e *routing key*)).

ID	Sensor	Producer-Exchange	Producer-Routing key
1	GPS	User_1	User.1.sensors.gps
2	Heart rate	User_1	User.1.sensors.hr
3	Others	User_1	User.1.others
4	GPS	User_2	User.2.sensors.gps
5	Heart rate	User_2	User.2.sensors.hr
6	Others	User_2	User.2.sensors.others

Consumer-Exchange	Consumer – Routing key	Producer ID subscribed
User_1	User.1.sensors.gps	1
User_1	User.1.sensors.*	1,2
User_1	User.1.*	1,2,3
User_2	User.2.*	4,5,6
User_2	User.*.sensors.*	4,5,6
User_2	User.*	4,5,6

Figura 6.4: Funcionamento do roteamento de mensagens tipo "*Topics*"

Como foi observado no capítulo 2.4.2, existem vários tipos de *"exchanges"*. Para o desenvolvimento desta plataforma foi utilizado o tipo "*topic exchange*"que realiza o roteamento de mensagens para uma ou mais filas de mensagens baseado na chave de roteamento e no nome do

"exchange". É bastante utilizado para implementar variações do modelo *Publish-Subscribe* por exemplo, para um dispositivo móvel receber informações relativas a um determinado utilizador.

Na figura 6.4 são apresentados alguns exemplos de "exchanges"e chaves de roteamento. A subscrição de recursos utilizando este método realiza-se com recurso a expressões regulares. Assim se o subscritor utilizar exatamente o mesmo par "*exchange*"e chave de roteamento que o produtor de mensagens, subscreve apenas esse recurso. A utilização do caractér "*"significa "uma ou mais expressões"e é a presença deste caractér numa chave de roteamento que permite a subscrição de vários recursos conforme apresentado na figura. Existem várias formas de subscrição de vários recursos utilizando este método e é importante a definição de quais são as expressões regulares que farão sentido utilizar no cenário que se deseja implementar.

6.4 Cliente Android

Para a representação de um sensor inteligente, optou-se pelo desenvolvimento de uma aplicação cliente para a plataforma móvel Android. Esta aplicação deve permitir interagir com o servidor ReST para a criação/modificação/consulta de dados relativos ao sistema, assim como fornecer a possibilidade de subscrição de vários utilizadores/sensores. Nesta secção é apresentada a arquitetura desta aplicação cliente e a forma como é realizada a comunicação entre os vários módulos.

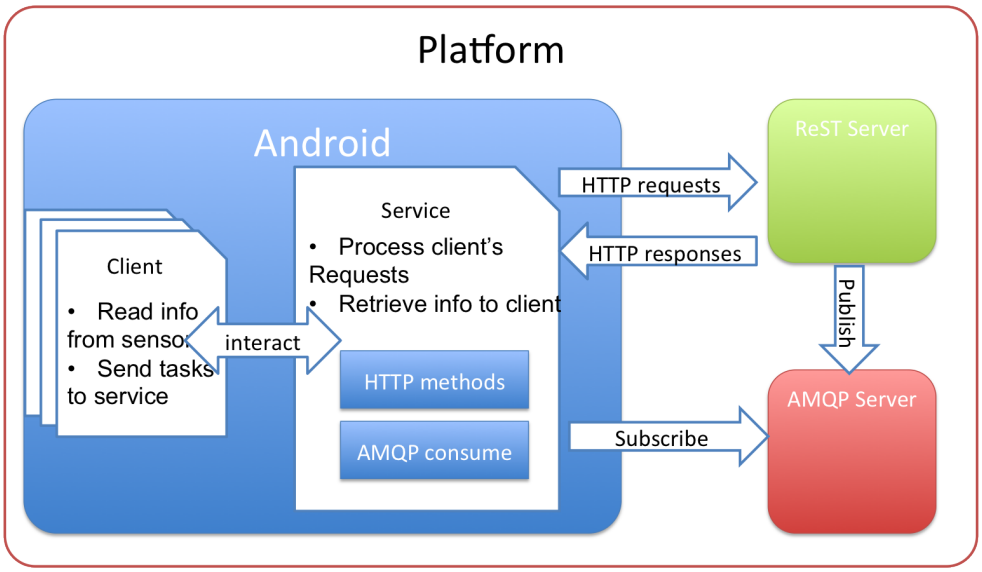


Figura 6.5: Interação entre componentes da plataforma

6.4.1 Arquitetura

No contexto da aplicação cliente, foram desenvolvidos dois módulos distintos: um serviço e uma aplicação que interage com esse serviço.

Um serviço em Android pode ser considerado todo e qualquer processo que não necessita de ter uma interface associada. Depois do serviço se encontrar em execução, este fica a correr num processo em segundo plano. O serviço deve possuir pelo menos uma forma de comunicar com outros processos que executem na mesma plataforma. É da responsabilidade do serviço toda a lógica de comunicação com o servidor ReST e o servidor AMQP. No fundo, o serviço desenvolvido, fornece todas as funcionalidades de comunicação M2M a uma aplicação cliente (em Android) que deseje utilizar as suas capacidades. Associado ao serviço foi desenvolvida uma interface Android que controla a execução do mesmo (apenas para ligar e desligar o serviço). É possível também especificar vários parâmetros essenciais à comunicação com os servidores ReST e AMQP como a sua localização na rede.

6.4.2 Comunicação inter-processos Android

Todas as aplicações cliente (em Android) que pretendam utilizar este serviço, têm de respeitar um conjunto de regras e códigos estipulados para que consigam realizar a comunicação com o mesmo. Através de códigos, é possível que qualquer aplicação desenvolvida para Android, utilize uma ou várias funcionalidades providenciadas pelo serviço abstraindo-se assim toda a lógica protocular e de comunicação, como se de uma API se tratasse. A título de exemplo, uma aplicação para se registar no serviço, necessita de lhe enviar uma mensagem com o código ”MSG_REGISTER_CLIENT”. Note-se que esta foi uma das formas possíveis para realizar a comunicação inter-processo dentro da plataforma Android. Existem outras formas de realizar a comunicação e deve ser utilizada a que melhor se adequar ao serviço implementado. Caso o serviço seja local (apenas pode ser utilizado por atividades no mesmo processo) pode ser implementado um serviço tipo "*LocalService*". Se por outro lado, o serviço comunica com aplicações fora do seu processo, pode-se utilizar a classe nativa Android "Messenger"ou ainda, um pouco mais complicado, a utilização de interfaces AIDL para descrição do serviço que também fornece outras funcionalidades para a realização de comunicação inter processos.

Na Figura 6.6 apresenta-se o aspeto da aplicação cliente Android com as suas funcionalidades implementadas. Durante esta dissertação não foi depositado um grande esforço no desenvolvimento da interface, e sim na eficiência das suas funcionalidades.

6.5 Ferramentas

Nesta secção serão apresentadas as ferramentas utilizadas para o desenvolvimento da aplicação no âmbito desta dissertação.

6.5.1 Servidor ReST

Para o desenvolvimento do servidor, foi utilizada a *framework Ruby on Rails* (RoR). A decisão de utilizar esta plataforma está associada às várias caraterísticas desta plataforma entre as quais, a facilidade de criação de serviços *ReSTful*, permitindo ainda técnicas como o *scaffolding* que

Figura 6.6: Interface da aplicação cliente desenvolvida

reduz drasticamente o tempo de desenvolvimento de soluções que utilizem o estilo arquitetónico ReST. Esta plataforma também possui um vasto conjunto de extensões que permitem personalizar facilmente o serviço de várias formas, adicionando-lhes inúmeras capacidades. RoR possui ainda vários conceitos associados como o *Don't Repeat Yourself* (DRY) fornecendo mecanismos que evitam a repetição de pedaços de código que por sua vez, permite uma maior compreensão do mesmo, para que este seja facilmente adaptado por outros programadores. O modelo utilizado por esta plataforma de desenvolvimento é o *Model-View-Controller* (MVC) que divide a camada de aplicação em três componentes distintas que comunicam entre si: as vistas que fornecem as representações dos dados, o modelo que poderá estar por exemplo, ligado a bases de dados, e ainda os controladores que são responsáveis pelo processamento dos pedidos, interrogação de modelos e transferência de dados para as vistas que serão recebidas pelo cliente.

6.5.2 Broker AMQP

Para o desenvolvimento do servidor de notificações foi utilizada uma implementação do protocolo AMQP *open-source* denominada *RabbitMQ*. Esta escolha revelou-se importante para a redução do tempo e esforço necessário para interligar os componentes da plataforma a este servidor (servidor ReST e aplicação cliente), uma vez que a sua utilização é bastante simples e permite a definição de múltiplos modelos de roteamento de mensagens.

6.5.3 Aplicação Android

A plataforma Android foi escolhida para o desenvolvimento da aplicação cliente uma vez que se apresenta como uma plataforma utilizada por vários fabricantes de *smartphones* a nível mundial além de ser um sistema operativo *open-source* desenvolvido pela empresa Google que corre em dispositivos eletrónicos como *smartphones* e *tablets*. O facto de ser um sistema operativo aberto, permite que os fabricantes de dispositivos eletrónicos possam modificar o sistema operativo sem restrições permitindo uma total liberdade de configuração e adaptação do sistema a um qualquer dispositivo. Para além disso, atualmente o sistema operativo Android conta com cerca de 400 milhões de utilizadores e 1 milhão de ativações diárias. O mercado de aplicações para este sistema operativo conta com a presença de 600 mil aplicações e jogos e 20 mil milhões de instalações. Estes são motivos mais do que suficientes para a utilização deste sistema operativo como plataforma base para o desenvolvimento de uma aplicação cliente que interaja com a plataforma desenvolvida no âmbito desta dissertação.

6.6 Documentação

Para além da informação apresentada neste capítulo acerca da plataforma desenvolvida, existe também documentação adicional da plataforma Android disponíveis a partir da página de Internet http://paginas.fe.up.pt/ ei05106/android-doc/.

A documentação da aplicação Android foi gerada com recurso ao *software* de geração automática de documentação *Doxygen*[1].

[1] Aplicação disponível a partir da página de Internet Doxygen

Capítulo 7

Resultados

Os dados apresentados ao longo deste capítulo são dados que foram efetivamente calculados com o auxílio do *software Wireshark* cuja função é a captura e disponibilização da informação relativa à transferência de pacotes de dados utilizando tanto a plataforma desenvolvida como a plataforma que utiliza XMPP sobre a qual também incide esta análise. De notar que a análise foi realizada ao nível da camada de rede do protocolo TCP/IP. No final deste capítulo, serão apresentadas algumas conclusões baseadas na informação aqui apresentada.

7.1 Plataforma M2M

A plataforma desenvolvida no âmbito desta dissertação utiliza um servidor ReST e ainda um *broker* AMQP como serviço orientado a notificações. Como referido no Capítulo 5, o estilo arquitetónico ReST funciona com o modelo de transações *request-response*. Ao longo desta secção serão apresentadas as mensagens trocadas na execução de cada funcionalidade e respetivos tamanhos de pacotes.

Na Figura 7.1 apresentam-se as quantidades de dados envolvidas na execução de um determinado método HTTP, fazendo uma clara distinção entre o pedido e a respetiva resposta. Note-se que os dados apresentados referem-se a pedidos que não contemplam dados específicos do cenário, mas sim pedidos genéricos ou seja, no mínimo cada pedido HTTP necessita de transacionar a quantidade de informação apresentada na figura referida. Cada transação possui uma parte constante (referente aos cabeçalhos da mensagem) e outra variável (referente aos dados específicos que as mensagens contêm) sendo que os dados apresentados na figura baseiam-se no tamanho apenas da parte constante.

7.1.1 Anúncio de sensor/serviço

O anúncio de um sensor / serviço na plataforma desenvolvida realiza-se entre duas entidades distintas: o servidor ReST e a aplicação cliente. Esta funcionalidade é implementada recorrendo a um pedido HTTP POST que regista o utilizador ou sensor na base de dados do servidor ReST. O servidor responde ao cliente com a informação do recurso criado, fornecendo-lhe o identificador

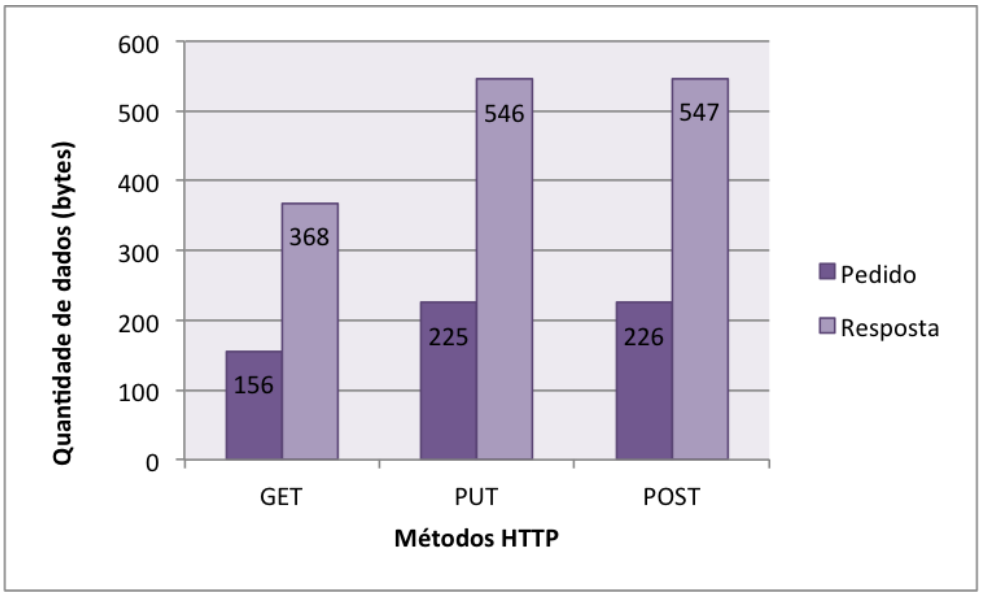


Figura 7.1: Tamanho dos cabeçalhos das mensagens associadas a pedidos HTTP.

associado ao registo do sensor no serviço. Esse identificador é utilizado sempre que seja necessário publicar informações relativas ao sensor.

As trocas de mensagens necessárias para a realização desta funcionalidade são praticamente as mesmas que as apresentadas na Figura 7.4, mudando apenas o método HTTP que neste caso será o método HTTP POST em vez de um HTTP PUT ilustrado na Figura 7.4. Os valores apresentados na Figura 7.1 são diferentes uma vez que não foram contabilizados no pedido as informações que se desejem registar no sistema e a resposta também não apresenta informações relativas ao pedido executado. Serão necessários no mínimo 773 bytes de informação para o anúncio de sensores e/ou serviços na plataforma desenvolvida.

7.1.2 Descoberta de sensor / serviço

A Figura 7.2 apresenta a troca de mensagens envolvida no processo de obtenção de informações acerca dos clientes registados no sistema. O *request* assinalado com fundo vermelho possui o tamanho de 154 bytes e segue-se a resposta por parte do servidor ReST com o tamanho de 703 bytes. No pedido pode observar-se a intenção do cliente receber a informação no formato JSON e a resposta surge de acordo com o especificado por ele. Note-se também que na resposta, os dados JSON apresentados são meramente exemplificativos e o seu tamanho pode variar. Neste modelo de transações não existem ligações ao nível da camada de aplicação e com pedidos HTTP GET é possível descobrir todos os recursos registados no sistema de forma bastante simples. A descoberta de um sensor (apresentada na Figura 7.3) processa-se da mesma forma, apenas variando o URL do pedido por parte do cliente e a resposta por parte do servidor.

Na Figura 7.1 observa-se que são necessárias trocas de no mínimo 524 bytes para cada pedido

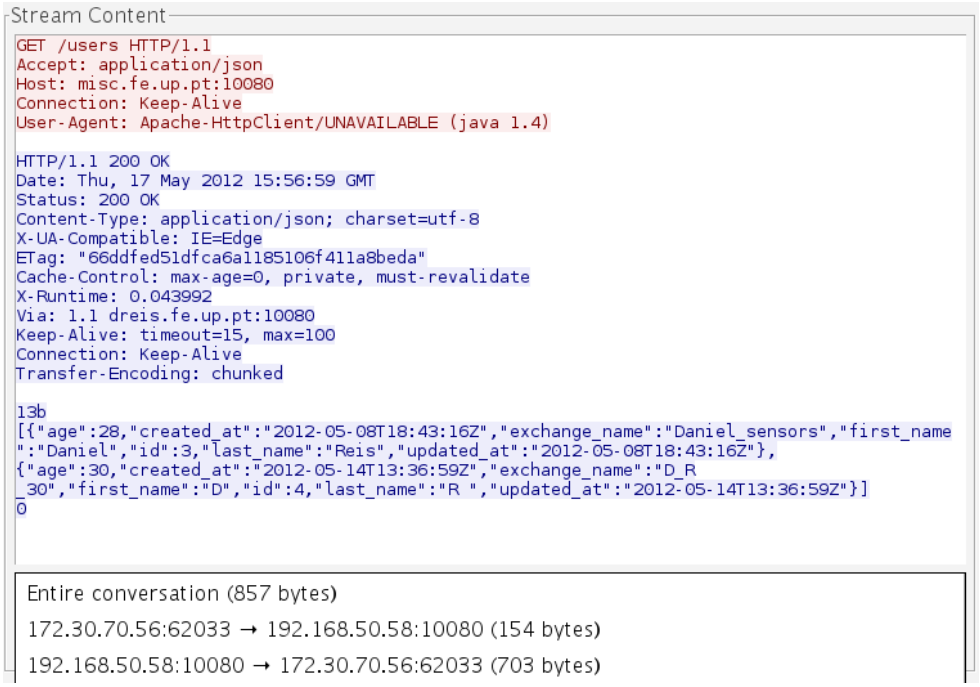


Figura 7.2: Pedido HTTP para realização de descoberta de utilizadores registados no sistema

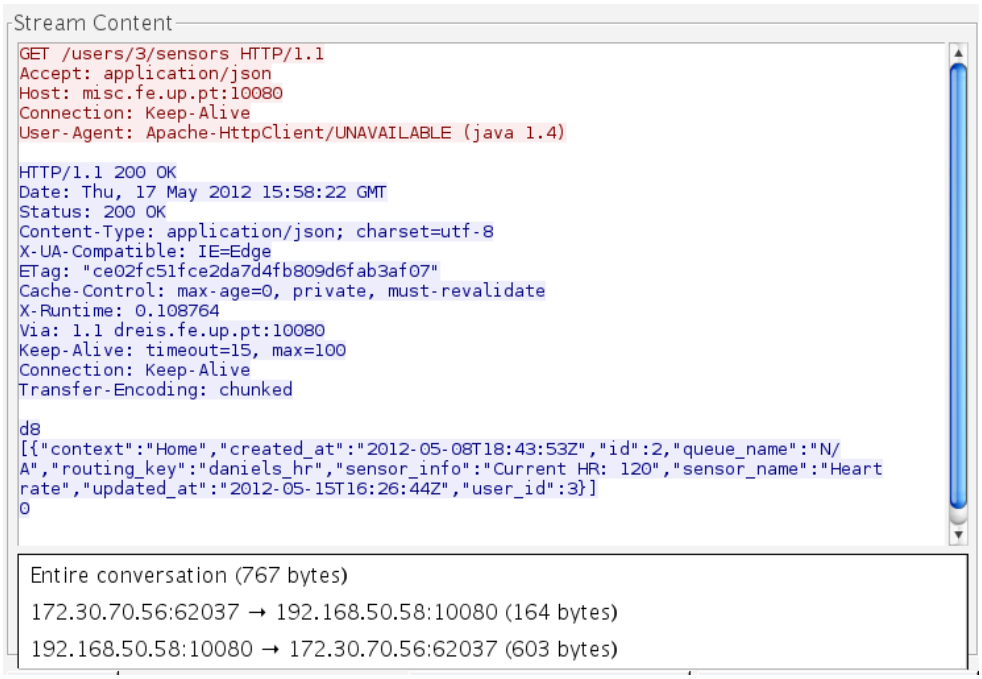


Figura 7.3: Pedido HTTP capturado para obtenção de informação relacionada com os sensores associados a um dado utilizador registado no sistema

HTTP GET e respetiva resposta isto é, a cada iteração. Note-se que na resposta não foram contabilizados os dados relativos à informação acerca de sensores ou utilizadores como apresentado na

Figura 7.2, daí os valores apresentados serem diferentes.

7.1.3 Publicação de dados

No que diz respeito à interação aplicação cliente-servidor ReST, sempre que um sensor necessitar de publicar informações, fá-lo através de um pedido HTTP PUT. Na Figura 7.4 são apresentadas as mensagens trocadas entre cliente e servidor. Como acontece na secção anterior, também aqui o pedido aparece sobre um fundo vermelho enquanto que a resposta do servidor aparece sobre um fundo azul. De notar que na resposta do servidor estão presentes as informações acerca do recurso atualizado, no entanto esta foi uma decisão de implementação. Caso fosse necessário reduzir o tamanho dos pacotes trocados, poder-se-iam omitir estas informações fornecendo apenas o estado do pedido ao cliente. Uma publicação de dados por parte de um sensor utiliza aproximadamente 338 bytes enquanto que a resposta apresenta o tamanho de 772 bytes, num total de 1110 bytes de conversação total para realização desta funcionalidade, isto se forem tidas em conta as informações acerca da publicação de dados. Ao considerar-se apenas os cabeçalhos das mensagens, isto é excetuando a parte variável da mensagem conclui-se que para a publicação de dados, são necessários no mínimo 771 bytes em cabeçalhos da mensagem para a realização de uma publicação de dados por parte de um recurso registado no serviço ReST.

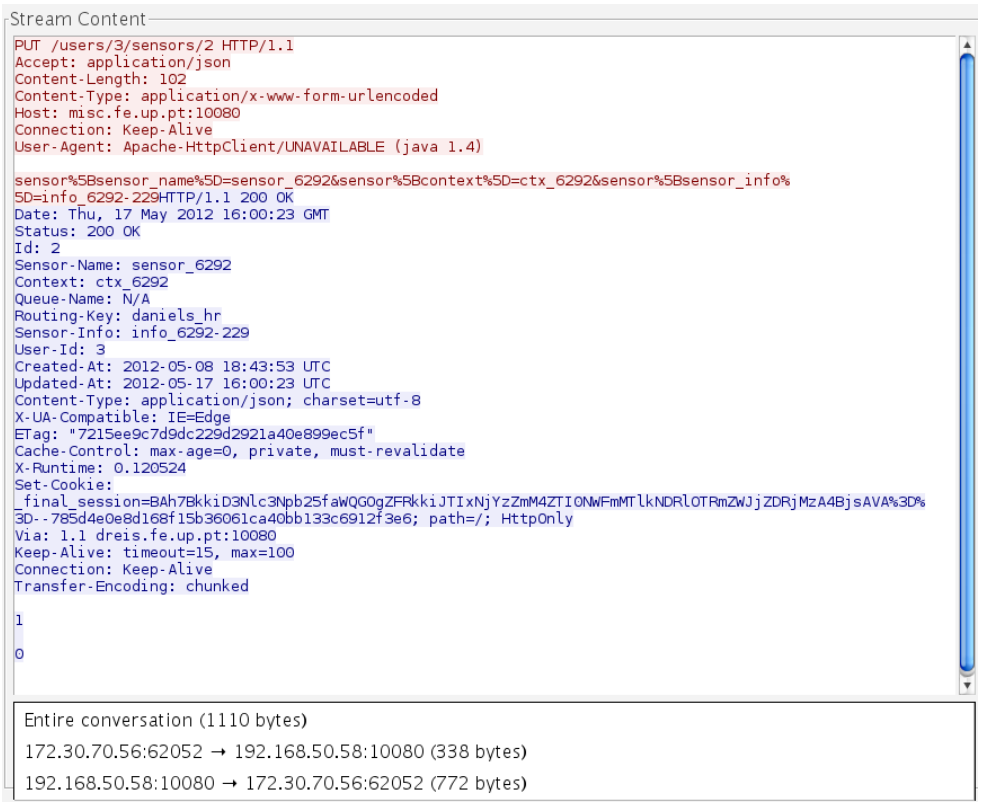
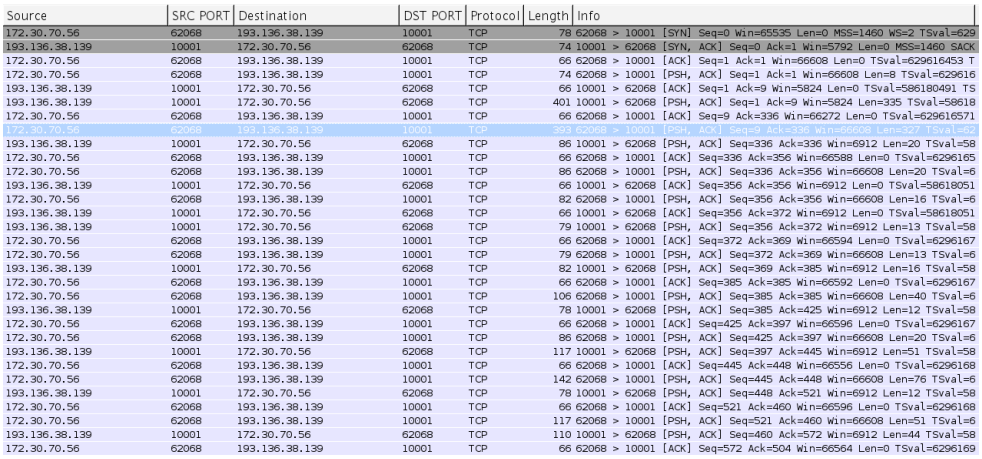


Figura 7.4: Pedido HTTP utilizado para a publicação de informações associadas a um sensor

7.1.4 Subscrição de sensor / serviço

Para a subscrição de notificações associadas a um determinado sensor, a aplicação cliente deve utilizar as informações fornecidas pelo servidor ReST (*exchange* e chave de roteamento) para estabelecer uma conexão com o servidor AMQP. Durante a fase de descoberta de um sensor, a aplicação cliente já consultou os utilizadores e sensores registados no sistema, daí já possuir todas as informações para se conetar ao *broker* AMQP e subscrever notificações.



Source	SRC PORT	Destination	DST PORT	Protocol	Length	Info
172.30.70.56	62068	193.136.38.139	10001	TCP	78	62068 > 10001 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=2 TSval=629
193.136.38.139	10001	172.30.70.56	62068	TCP	74	10001 > 62068 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=1 Ack=1 Win=66608 Len=0 TSval=629616453 T
172.30.70.56	62068	193.136.38.139	10001	TCP	74	62068 > 10001 [PSH, ACK] Seq=1 Ack=1 Win=66608 Len=8 TSval=629616
193.136.38.139	10001	172.30.70.56	62068	TCP	66	10001 > 62068 [ACK] Seq=1 Ack=9 Win=5824 Len=0 TSval=586180491 TS
193.136.38.139	10001	172.30.70.56	62068	TCP	401	10001 > 62068 [PSH, ACK] Seq=1 Ack=9 Win=5824 Len=335 TSval=58618
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=9 Ack=336 Win=66272 Len=0 TSval=629616571
172.30.70.56	62068	193.136.38.139	10001	TCP	393	62068 > 10001 [PSH, ACK] Seq=9 Ack=336 Win=66608 Len=327 TSval=62
193.136.38.139	10001	172.30.70.56	62068	TCP	86	10001 > 62068 [PSH, ACK] Seq=336 Ack=336 Win=6912 Len=20 TSval=58
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=336 Ack=356 Win=66588 Len=0 TSval=6296165
172.30.70.56	62068	193.136.38.139	10001	TCP	86	62068 > 10001 [PSH, ACK] Seq=336 Ack=356 Win=66608 Len=20 TSval=6
193.136.38.139	10001	172.30.70.56	62068	TCP	66	10001 > 62068 [ACK] Seq=356 Ack=356 Win=6912 Len=0 TSval=58618051
172.30.70.56	62068	193.136.38.139	10001	TCP	82	62068 > 10001 [PSH, ACK] Seq=356 Ack=356 Win=66608 Len=16 TSval=6
193.136.38.139	10001	172.30.70.56	62068	TCP	66	10001 > 62068 [ACK] Seq=356 Ack=372 Win=6912 Len=0 TSval=58618051
193.136.38.139	10001	172.30.70.56	62068	TCP	79	10001 > 62068 [PSH, ACK] Seq=356 Ack=372 Win=6912 Len=13 TSval=58
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=372 Ack=369 Win=66594 Len=0 TSval=6296167
172.30.70.56	62068	193.136.38.139	10001	TCP	79	62068 > 10001 [PSH, ACK] Seq=372 Ack=369 Win=66608 Len=13 TSval=6
193.136.38.139	10001	172.30.70.56	62068	TCP	82	10001 > 62068 [PSH, ACK] Seq=369 Ack=385 Win=6912 Len=16 TSval=58
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=385 Ack=385 Win=66592 Len=0 TSval=6296167
172.30.70.56	62068	193.136.38.139	10001	TCP	106	62068 > 10001 [PSH, ACK] Seq=385 Ack=385 Win=66608 Len=40 TSval=6
193.136.38.139	10001	172.30.70.56	62068	TCP	78	10001 > 62068 [PSH, ACK] Seq=385 Ack=425 Win=6912 Len=12 TSval=58
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=425 Ack=397 Win=66596 Len=0 TSval=6296167
172.30.70.56	62068	193.136.38.139	10001	TCP	86	62068 > 10001 [PSH, ACK] Seq=425 Ack=397 Win=66608 Len=20 TSval=6
193.136.38.139	10001	172.30.70.56	62068	TCP	117	10001 > 62068 [PSH, ACK] Seq=397 Ack=445 Win=6912 Len=51 TSval=58
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=445 Ack=448 Win=66556 Len=0 TSval=6296168
172.30.70.56	62068	193.136.38.139	10001	TCP	142	62068 > 10001 [PSH, ACK] Seq=445 Ack=448 Win=66608 Len=76 TSval=6
193.136.38.139	10001	172.30.70.56	62068	TCP	78	10001 > 62068 [PSH, ACK] Seq=448 Ack=521 Win=6912 Len=12 TSval=58
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=521 Ack=460 Win=66596 Len=0 TSval=6296168
172.30.70.56	62068	193.136.38.139	10001	TCP	117	62068 > 10001 [PSH, ACK] Seq=521 Ack=460 Win=66608 Len=51 TSval=6
193.136.38.139	10001	172.30.70.56	62068	TCP	110	10001 > 62068 [PSH, ACK] Seq=460 Ack=572 Win=6912 Len=44 TSval=58
172.30.70.56	62068	193.136.38.139	10001	TCP	66	62068 > 10001 [ACK] Seq=572 Ack=504 Win=66564 Len=0 TSval=6296169

Figura 7.5: Subscrição de notificações associadas a um determinado sensor de um dado utilizador

Na Figura 7.5 são apresentadas as mensagens capturadas durante a execução desta funcionalidade. A troca de mensagens entre o servidor AMQP e a aplicação cliente representam cerca de 1074 bytes, dos quais 571 bytes foram utilizados pelo cliente nas mensagens que enviou ao servidor ao qual o último respondeu um total de 503 bytes. Note-se que a subscrição de notificações não contempla a receção de dados subscritos pela aplicação cliente. Essa funcionalidade é apresentada na secção seguinte. Após a fase de estabelecimento de conexão, são necessários 893 bytes de informação trocada para a realização de uma subscrição por parte da aplicação cliente.

7.1.5 Obtenção de dados

Após a fase de subscrição de dados, o cliente será sempre notificado pelo servidor AMQP na ocorrência de uma publicação de informações no servidor ReST por parte do sensor. Na Figura 7.6 é apresentada a mensagem enviada pelo servidor para o cliente com a informação atualizada do sensor subscrito. Esta mensagem tem o tamanho de 395 bytes. Mais uma vez o tamanho dos pacotes varia com as informações que contêm. Foi uma decisão de implementação que o servidor ReST publique toda a informação relativa a um recurso no servidor AMQP que por sua vez realiza um "push notification" junto dos clientes interessados em tal informação.

Cada obtenção de dados representa trocas de no mínimo 166 bytes. A este valor adiciona-se o número de bytes relativos aos dados rececionados.

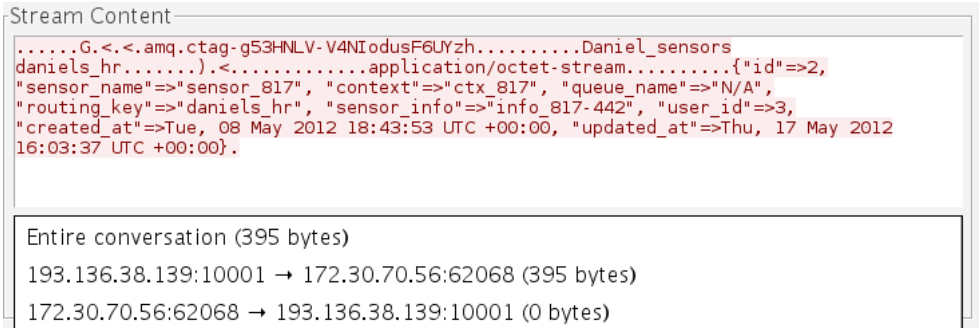


Figura 7.6: Obtenção de dados enviados pelo servidor AMQP para a aplicação cliente

7.2 Aplicação proprietária baseada em XMPP

Nesta secção será objeto de estudo a aplicação desenvolvida no âmbito de outra tese de mestrado a decorrer em paralelo [Pru12]. Os dados aqui apresentados foram também eles capturados com a mesma ferramenta que analisou as transferências de informação da plataforma desenvolvida nesta dissertação. Conforme sugerido na Secção 2.4.1.2 para a implementação de um serviço que utilize o protocolo XMPP, esta aplicação utiliza TLS para segurança do canal. A consequência dessa implementação é que o programa de captura reconhece as transferências de informação entre entidades, mas não permite visualizar as mensagens para a realização de um estudo mais profundo acerca do seu funcionamento. Mesmo assim utilizou-se o programa e nesta secção serão apresentadas algumas considerações que resultaram da experiência e da existência de transferências de pacotes na rede.

Ao nível da camada de aplicação, a fase de estabelecimento de conexão é requisito necessário para a execução das funcionalidades apresentadas a seguir. Na Figura 7.7 apresentam-se as mensagens trocadas durante essa fase inicial.

Na primeira iteração *request-response* são trocados 816 bytes para o estabelecimento do canal de conexão. Na iteração seguinte é realizada o estabelecimento de um canal seguro TLS (com um custo de 101 bytes) totalizando 917 bytes para o estabelecimento do canal de comunicação.

7.2.1 Anúncio de sensor / serviço

Esta funcionalidade não é suportada pela plataforma objeto de estudo. Os sensores encontram-se *hardcoded* na aplicação cliente, o que se traduz num sistema menos flexível. Seria necessário implementar uma funcionalidade de descoberta de sensores associados a um dado utilizador o que iria conferir um outro alcance desta plataforma e a possibilidade de ser extendida de outra forma que assim não permite. Utilizando o protocolo XMPP é possível descobrir recursos associados a uma dada entidade conforme descrito em [HMESA08]. Para a descoberta de utilizadores poderia recorrer-se a uma implementação de um servidor ReST como acontece no caso da plataforma desenvolvida no âmbito desta dissertação.

```
REQUEST:  (128 bytes)
------------------------------------------------------------------------
<stream:stream to="mycontext.ptinovacao.pt" xmlns="jabber:client" xmlns:stream="http://
etherx.jabber.org/streams" version="1.0">


RESPONSE: (688 bytes)
------------------------------------------------------------------------
<?xml version='1.0' encoding='UTF-8'?><stream:stream xmlns:stream="http://etherx.jabber.org/
streams" xmlns="jabber:client" from="mycontext.ptinovacao.pt" id="72ab89e9" xml:lang="en"
version="1.0"><stream:features><starttls xmlns="urn:ietf:params:xml:ns:xmpp-tls"></
starttls><mechanisms xmlns="urn:ietf:params:xml:ns:xmpp-sasl"><mechanism>DIGEST-MD5</
mechanism><mechanism>PLAIN</mechanism><mechanism>ANONYMOUS</mechanism><mechanism>CRAM-MD5</
mechanism></mechanisms><compression xmlns="http://jabber.org/features/
compress"><method>zlib</method></compression><auth xmlns="http://jabber.org/features/iq-
auth"/><register xmlns="http://jabber.org/features/iq-register"/></stream:features>

REQUEST: (51 bytes)
------------------------------------------------------------------------
<starttls xmlns="urn:ietf:params:xml:ns:xmpp-tls"/>

RESPONSE: (50 bytes)
------------------------------------------------------------------------
<proceed xmlns="urn:ietf:params:xml:ns:xmpp-tls"/>
```

Figura 7.7: Dados capturados relativos ao estabelecimento de conexão utilizando o protocolo XMPP

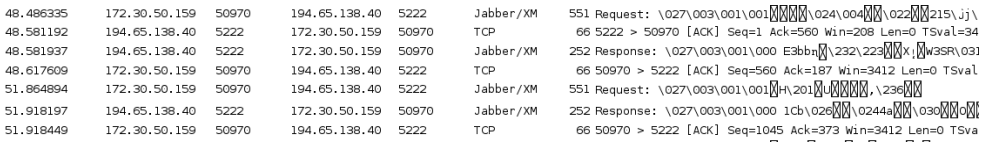
Durante a fase de conexão da aplicação cliente ao *broker* XMPP são trocados vários *stanzas* conforme descrito na apresentado na Figura 5.9. Como observado anteriormente, esta fase requer a transação de 917 bytes de informação. De seguida verificaram-se várias trocas de mensagens com tamanho idêntico e com um intervalo de 30 segundos entre elas. Fruto da utilização de ligações TLS, não foi possível interpretar o conteúdo destes pacotes transferidos, contudo levantam-se alguns problemas relacionados com a eficiência do processo comunicacional. Este "polling" de dados não parece estar diretamente relacionado com nenhuma das funcionalidades definidas já que ocorrem antes de qualquer subscrição ser feita aos dados do utilizador em questão. Estas mensagens apresentam um "payload" de 69 bytes.

7.2.2 Descoberta de sensor / serviço

Na secção anterior verificou-se a ausência de funcionalidades de anúncio de sensores / serviços. O mesmo acontece no caso da descoberta de sensores / serviços. Estas funcionalidades encontram-se *hardcoded* na plataforma.

7.2.3 Publicação de dados

Para a publicação de dados, esta plataforma segue o modelo *request-response*. Na Figura 7.8 apresenta-se a captura de mensagens durante aproximadamente 3 segundos. Cada pedido por parte do cliente representa 551 bytes de informação transferida e a resposta por parte do servidor XMPP representa 252 bytes perfazendo um total de 803 bytes para a realização desta funcionalidade.

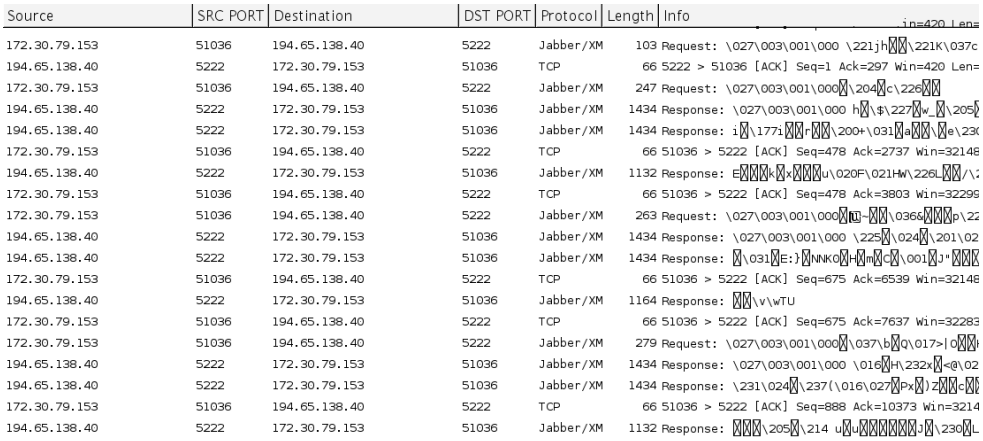


48.486335	172.30.50.159	50970	194.65.138.40	5222	Jabber/XM	551	Request: \027\003\001\001 \024\004 \022 215\jj\
48.581192	194.65.138.40	5222	172.30.50.159	50970	TCP	66	5222 > 50970 [ACK] Seq=1 Ack=560 Win=208 Len=0 TSval=34
48.581937	194.65.138.40	5222	172.30.50.159	50970	Jabber/XM	252	Response: \027\003\001\000 E3bbx \232\223 X, W3SR\03
48.617609	172.30.50.159	50970	194.65.138.40	5222	TCP	66	50970 > 5222 [ACK] Seq=560 Ack=187 Win=3412 Len=0 TSval
51.864894	172.30.50.159	50970	194.65.138.40	5222	Jabber/XM	551	Request: \027\003\001\001 H\201 u ,\236
51.918197	194.65.138.40	5222	172.30.50.159	50970	Jabber/XM	252	Response: \027\003\001\000 1Cb\026 \0244a \030 o
51.918449	172.30.50.159	50970	194.65.138.40	5222	TCP	66	50970 > 5222 [ACK] Seq=1045 Ack=373 Win=3412 Len=0 TSva

Figura 7.8: Publicação de dados enviados pelo servidor XMPP para a aplicação cliente

7.2.4 Subscrição de sensor / serviço e obtenção de dados

Da utilização desta plataforma verificou-se que o escolher de um utilizador para monitorização não provoca um aumento da frequência ou do tamanho dos pedidos. Quando se escolhe um sensor para monitorização, esta ação desencadeia uma fase em que se deteta um aumento da frequência de mensagens *request-response* com tamanhos variados, seguida de uma fase de "*polling*"com pedidos de 30 em 30 segundos com o "payload" de 69 bytes. A obtenção de dados foi capturada e verificou-se a chegada de pacotes de resposta com tamanhos variados entre 1000 bytes e 1500 bytes conforme apresentado na Figura 7.9.



Source	SRC PORT	Destination	DST PORT	Protocol	Length	Info
172.30.79.153	51036	194.65.138.40	5222	Jabber/XM	103	Request: \027\003\001\000 \221jh \221K\037c
194.65.138.40	5222	172.30.79.153	51036	TCP	66	5222 > 51036 [ACK] Seq=1 Ack=297 Win=420 Len=
172.30.79.153	51036	194.65.138.40	5222	Jabber/XM	247	Request: \027\003\001\000 \204 c\226
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1434	Response: \027\003\001\000 h \$\227 w_ \205
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1434	Response: i \177i r \200+\031 a \ e\23
172.30.79.153	51036	194.65.138.40	5222	TCP	66	51036 > 5222 [ACK] Seq=478 Ack=2737 Win=3214
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1132	Response: E k x u\020F\021HW\226L /\
172.30.79.153	51036	194.65.138.40	5222	TCP	66	51036 > 5222 [ACK] Seq=478 Ack=3803 Win=3229
172.30.79.153	51036	194.65.138.40	5222	Jabber/XM	263	Request: \027\003\001\000 ~ \0366 p\22
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1434	Response: \027\003\001\000 \225 \024 \201\02
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1434	Response: \031 E:} NNKO H m c \001 J"
172.30.79.153	51036	194.65.138.40	5222	TCP	66	51036 > 5222 [ACK] Seq=675 Ack=6539 Win=3214
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1164	Response: \v\wTU
172.30.79.153	51036	194.65.138.40	5222	TCP	66	51036 > 5222 [ACK] Seq=675 Ack=7637 Win=3228
172.30.79.153	51036	194.65.138.40	5222	Jabber/XM	279	Request: \027\003\001\000 \037\b Q\017>| O
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1434	Response: \027\003\001\000 \016 H\232x <@\02
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1434	Response: \231\024 \237(\016\027 Px)z c
172.30.79.153	51036	194.65.138.40	5222	TCP	66	51036 > 5222 [ACK] Seq=888 Ack=10373 Win=3214
194.65.138.40	5222	172.30.79.153	51036	Jabber/XM	1132	Response: \205 \214 u u J \230

Figura 7.9: Obtenção de dados enviados pelo servidor XMPP para a aplicação cliente

7.3 Análise de resultados

Os gráficos apresentados nesta secção foram obtidos através da equação 7.1 que se traduz no seguinte: O tamanho de uma transação é igual às trocas associadas ao estabelecimento de uma conexão somado ao número de subscriçoes vezes o respetivo tamanho e ainda o número de receções de dados multiplicado pelo tamanho dos pacote de dados recebido para cada estabelecimento de conexão. As variáveis relativas às trocas de mensagens para o estabelecimento da conexão perfazem 1131 bytes. As trocas relativas ao estabelecimento de canais e subscrição totalizam 893 bytes, enquanto que para a obtenção de dados são necessários 166 bytes de informação excetuando os dados a serem transferidos propriamente ditos.

$$L_{transacao} = n_{conexoes} * (L_{conexao} + n_{subscricao} * L_{subscricao} + n_{rececoes} * L_{rececao}) \quad (7.1)$$

Nesta análise interessa perceber em que casos interessa utilizar um sistema orientado a notificações em detrimento da utilização de apenas métodos HTTP que operem sobre recursos ReST.

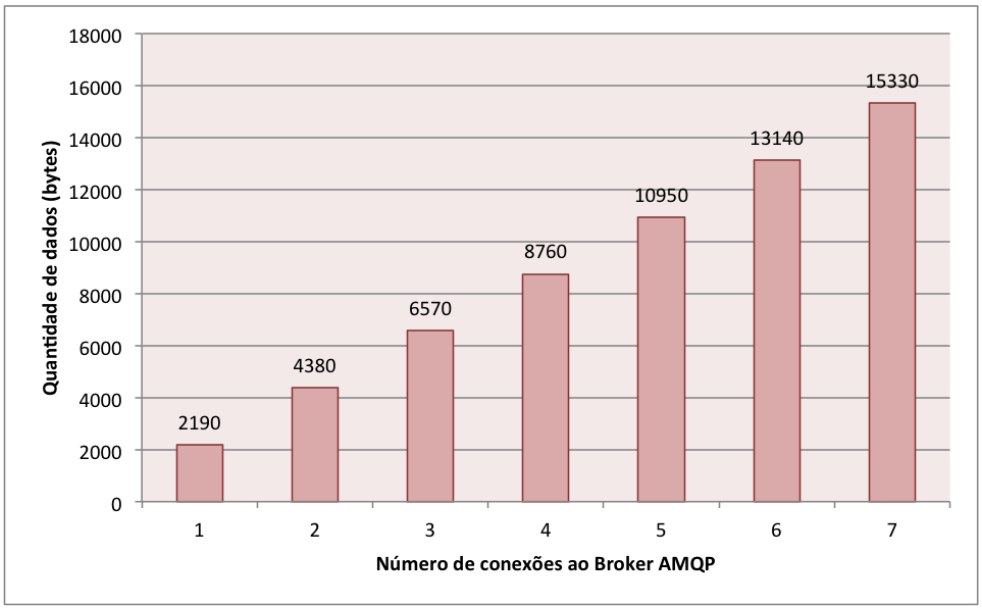


Figura 7.10: Efeito que o número de conexões ao *broker* AMQP possui na quantidade global de pacotes trocados.

A Figura 7.10 apresenta a variação da quantidade de dados necessários em função do número de conexões realizadas ao *broker* AMQP. Estes dados são referentes a apenas uma subscrição e uma única receção de dados enviados pelo servidor AMQP para a aplicação cliente. Cada vez que se perca a conetividade com a rede de comunicação a fase de estabelecimento de conexão é o principal responsável pelo aumento de dados trocados num cenário em que apenas se subscreva um utilizador e se receba dados relativos a esse utilizador apenas uma vez.

A Figura 7.11 apresenta a variação de dados necessários em função do número de subscrições efetuadas, tendo em conta o estabelecimento de apenas uma conexão ao *broker* AMQP e uma receção de dados. Cenários assim incluem a monitorização de vários sensores mas apenas um sensor publicou informação. Pressupõe-se que a conexão seja sempre a mesma não existindo a necessidade de estabelecimento de conexão mais do que uma vez.

A Figura 7.12 apresenta a variação de dados necessários em função do número de receções de dados por parte do *broker* AMQP, utilizando apenas uma conexão ao *broker* AMQP e realizando apenas uma subscrição junto deste. Cenários assim incluem o contexto de *healthcare* no qual um utilizador subscreve dados de um outro utilizador e recebe vários dados dele. Note-se que a variação de dados é baixa quando varia o número de receções.

Por último, a Figura 7.13 conclui os gráficos anteriores e compara com o estilo arquitetónico

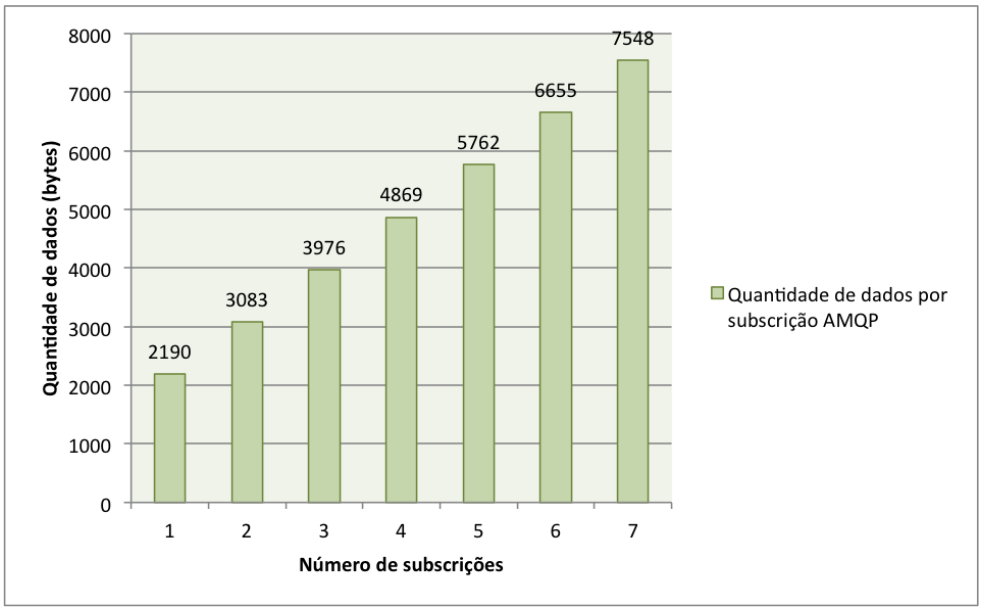


Figura 7.11: Efeito que o número de subscrições possui na quantidade global de pacotes trocados.

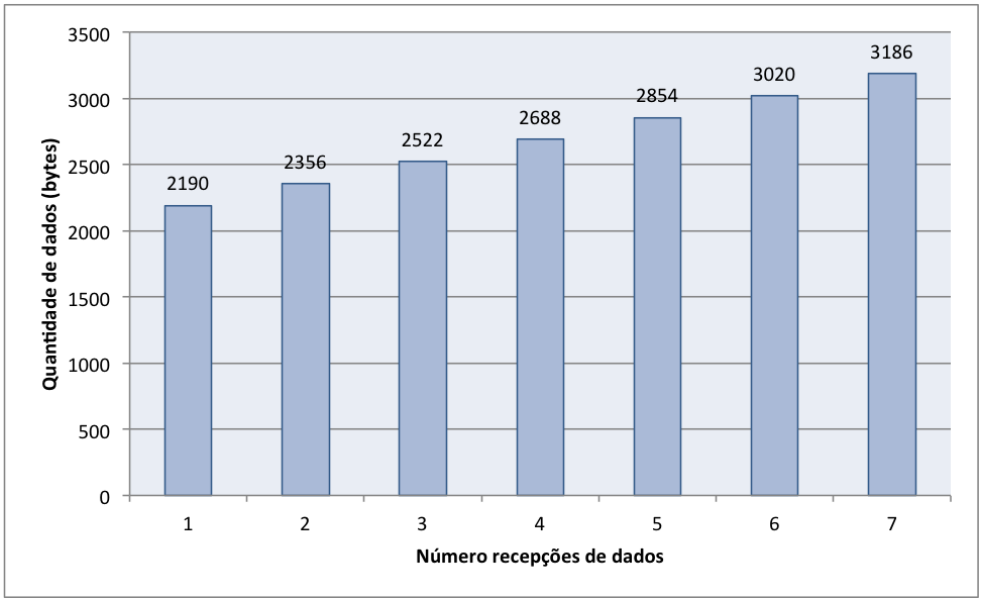


Figura 7.12: Efeito que o número de receções de dados possui na quantidade global de pacotes trocados.

ReST. Note-se qu o eixo das abcissas corresponde à variação do número de conexões ao analisar-se as barras azuis. Se por outro lado analisar-se as barras vermelhas, o eixo referido apresenta a variação do número de conexões estabelecidas ou ainda a variação de subscrições caso se analise as barras verdes. Por último, ao analisar-se o número de iterações ReST, o eixo das abcissas corresponde à variância do número de iterações. Através da sua análise pode-se concluir o seguinte:

- A fase de estabelecimento de conexão com o servidor AMQP necessita de várias trocas de informação; em sistemas de monitorização nos quais a mobilidade representa um fator chave é necessário ter cuidado com a utilização destes mecanismos, porque sempre que seja necessário reconetar ao *broker* são necessárias quantidades elevadas de informação trocada.
- Apenas a fase de subscrição de um dado recurso requer trocas de informação na ordem dos 900 bytes: este valor é aproximadamente o valor necessário para realizar pedidos HTTP GET para a obtenção de dados.
- O número de receções de dados utilizando a mesma conexão e apenas uma subscrição necessita de 166 bytes mais o número de bytes relativos à informação propriamente dita que se subscreveu. A partir da 6ª receção de dados compensa utilizar o mecanismo de subscrições AMQP uma vez que cada pedido HTTP GET necessita de mais informação para a receção de dados, sempre que apenas se necessite de realizar uma vez o estabelecimento da conexão e se possua apenas uma subscrição efetuada.

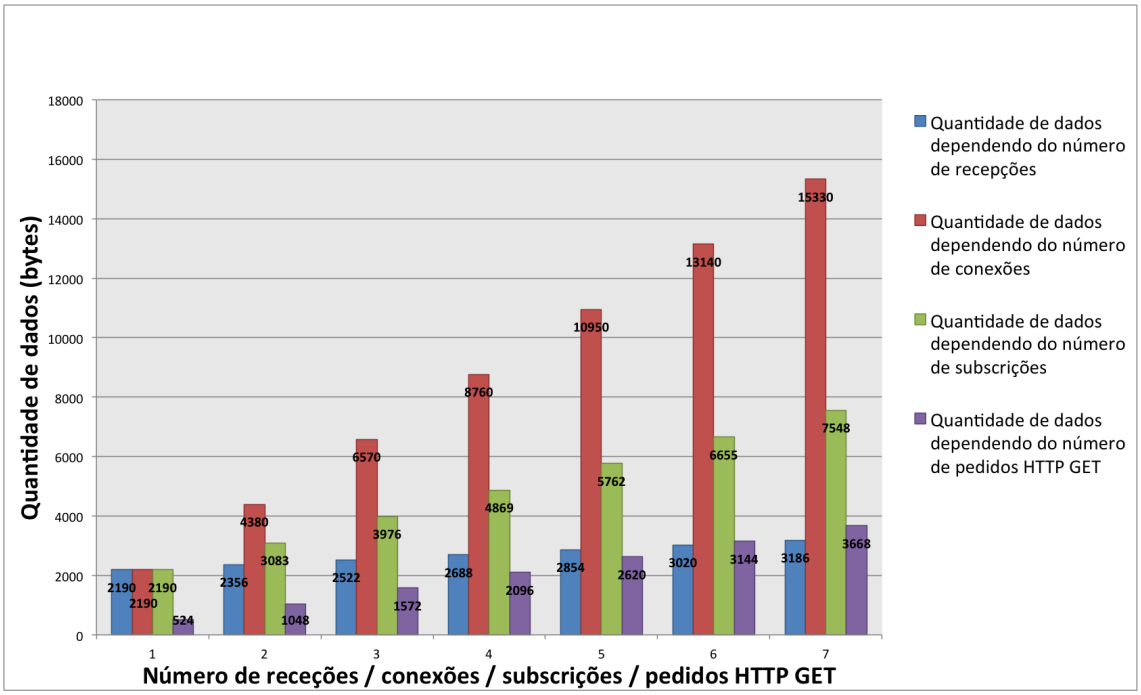


Figura 7.13: Efeito da variação de número de conexões, subscrições e receções de dados na quantidade de dados total utilizando AMQP.

Durante esta dissertação utilizou-se também uma plataforma que utiliza o protocolo XMPP como protocolo base de comunicação. A utilização desta plataforma aliada à inspeção dos pacotes trocados nas plataformas objetos de estudo na realização das funcionalidades apresentadas, pode-se aferir o seguinte:

- A mobilidade no cenário de *healthcare* representa um fator importante especialmente no caso do utilizador que está a ser monitorizado. Ao utilizarem-se protocolos que necessitem

de estabelecimento de conexões ao nível da camada de aplicação, sempre que se perca a conexão com a rede, é necessário restabelecer essa conexão. Para a publicação de dados e tendo em conta a mobilidade do utilizador monitorizado, a utilização de modelos *stateless* representa uma mais valia e melhor performance.

- O protocolo AMQP, apesar de possuir um *overhead* protocular superior ao XMPP, não realiza *polling* de dados. Após as fases de estabelecimento de conexão e subscrição de dados, apenas se verificam trocas de mensagens quando o servidor AMQP necessita de realizar *push* de informação junto das aplicações clientes.

- O *polling* de informação verificado na aplicação cujo protocolo base é o XMPP não parece estar relacionado com a execução de alguma funcionalidade. Um servidor ReST utilizaria essas trocas de informação para requisitar novos dados do serviço.

- Através da inspeção das mensagens capturadas na rede na realização das funcionalidades propostas nesta dissertação pelas duas plataformas, conclui-se que a aplicação desenvolvida em paralelo com esta dissertação necessita de mais trocas de mensagens e apresenta um custo transacional mais elevado que a desenvolvida utilizando ReST e AMQP.

- A partir da análise da arquitetura desenvolvida no âmbito desta dissertação, esta é mais simples, flexível e de fácil compreensão que a desenvolvida com base no protocolo XMPP (para mais detalhes consultar [Pru12]).

7.4 Esforço de desenvolvimento

Em relação ao esforço necessário para o desenvolvimento da plataforma no âmbito desta dissertação e graças às plataformas que se utilizaram para a criação dos servidores ReST, AMQP e aplicações cliente Android, considera-se que a plataforma desenvolvida requer um baixo nível de dificuldade e esforço associado. Mais concretamente e uma vez que as componentes desenvolvidas se encontram distribuídas na Internet de forma *decoupled*, o servidor ReST representa o sistema de informação onde os recursos estão disponíveis para interação, bastando para isso a geração de operações CRUD associadas aos recursos através de *scaffolding*. Graças à sua enorme flexibilidade e possibilidade de produtores e consumidores de mensagens definirem o tipo de roteamento de mensagens, a instalação do *broker* AMQP bem como a sua utilização não representa um grande esforço de desenvolvimento. É possível definir-se no servidor ReST os campos necessários para a subscrição de um qualquer mecanismo de notificações. Quanto ao desenvolvimento de aplicações cliente, a utilização de métodos HTTP para interação com o servidor ReST realiza-se de forma bastante transparente e a subscrição e receção de informações com origem no servidor AMQP também não representa uma grande obstáculo graças às bibliotecas de funções disponibilizadas em várias linguagens de programação entre as quais o JAVA. Também a utilização de formatos *lightweight* como é o caso do formato JSON permite uma rápida extração das informações contidas nas mensagens trocadas.

Já em relação ao esforço necessário para desenvolver novas funcionalidades para a plataforma que utiliza o protocolo XMPP, existe algum *overhead* ao utilizar a plataforma para transmitir um tipo de dados. É necessário criar um serviço chamado "Source Provider" para cada tipo de dados que se pretende enviar. Este serviço tem que estar a correr em algum servidor remoto e a sua única funcionalidade é a de publicar no *context broker* a informação, para esta ser reencaminhada para os potenciais subscritores dessa informação. Além disso, o tipo de informação a ser transmitida não pode ser definida em tempo de execução, sendo necessário a criação prévia de um *XML Schema*. As aplicações que pretendem publicar ou consumir este tipo de dados têm que incluir este *Schema* antes de serem compiladas. Apesar de algum *overhead* inicial, esta parece ser uma plataforma flexível até porque todos os componentes estão extremamente *decoupled*, sendo que cada serviço pode estar a correr num local remoto. Relativamente aos *consumers* e *sources* Android, é fácil criar um novo tipo sendo que todo o protocolo XMPP é implementado pela aplicação *Context Agent* que foi desenvolvida pela PT Inovação, não representando um grande esforço de desenvolvimento para o programador da aplicação.

Capítulo 8

Conclusões

Para realmente ser possível atingir os objetivos enunciados pela IoT e realização de comunicações M2M é necessário que as aplicações utilizem tecnologias e protocolos de comunicação eficientes, abertos e interoperáveis. Para isso é também necessário que se desenvolvam standards que suportem os conceitos subjacentes à IoT. Só assim será possível criar um sistema adequado às necessidades do futuro.

Além do desenvolvimento de uma plataforma híbrida composta pelo servidor ReST que publica mensagens no servidor AMQP, foi também realizada uma comparação desta plataforma com outra já existente que utiliza o protocolo XMPP introduzido também no âmbito desta dissertação. Esta comparação desenvolveu-se a dois níveis distintos: inicialmente realizou-se uma comparação entre as tecnologias utilizadas (ReST, AMQP e XMPP) num nível teórico das trocas envolvidas na execução de um conjunto de funcionalidades. Posteriormente e fruto da utilização das duas plataformas em questão, realizou-se uma comparação com dados extraídos das duas soluções por uma ferramenta de captura de dados na rede.

A multiplicidade de cenários em comunicações M2M representam um problema para a definição de uma arquitetura standard. Neste trabalho optou-se pela utilização de tecnologias abertas e interoperáveis, uma vez que apenas estas é que possibilitam a criação de novos standards e amplas utilizações a nível mundial em vários mercados verticais. O estilo arquitetónico ReST representa a base da plataforma desenvolvida uma vez que funciona como sistema de informação a partir do qual é possível registar e consultar informações acerca de vários recursos. A proposta nesta dissertação assenta num servidor ReST que contém todas as informações necessárias para a subscrição de notificações utilizando um qualquer protocolo de comunicação (neste trabalho recorreu-se a um *broker* AMQP). Esta decisão representa um ponto forte da plataforma desenvolvida uma vez que todos os clientes que desejem interagir com recursos, podem fazê-lo através de um protocolo amplamente utilizado (HTTP) além de poderem subscrever um dado recurso conforme a sua necessidade. É esta liberdade de escolha aliada à utilização de protocolos abertos que confere à plataforma desenvolvida a flexibilidade necessária para implementar sistemas de comunicações M2M.

Foi possível concluir que a utilização de um mecanismo de notificações nem sempre é a melhor escolha uma vez que estas tecnologias estão sujeitas ao estabelecimento de conexões ao nível da camada de aplicação. O número de vezes que uma conexão é interrompida assim como a periodicidade com que recebe informações e o número de subscrições que efetua, influenciam diretamente a necessidade de utilizar um dado mecanismo de notificações. Se a informação relacionada com a periodicidade com que um dispositivo publica informações estiver disponível e/ou essa periodicidade for elevada, ReST por si só apresenta-se como uma solução adequada neste tipo de cenários uma vez que é *stateless* e não necessita de estabelecimento de conexões. A presente dissertação contempla informação útil para que decisores possam tomar opções mais informadas e fundamentadas quando se trata de especificar uma arquitetura para a realização de comunicações M2M.

A mobilidade de um utilizador monitorizado num cenário de cuidados de saúde, representa um fator importante a ter em conta. A utilização de tecnologias que não necessitem de estabelecimento de conexões ao nível da camada de aplicação representam uma mais valia uma vez que a sua eficiência não depende da disponibilidade da rede. Tecnologias que recorram a modelos de transação *stateless* representam uma vantagem em relação a outros sistemas.

Como trabalho futuro poderia-se realizar um estudo acerca da aplicação de outros protocolos a comunicações M2M como por exemplo, o protocolo *Constrained Application Protocol* (CoAP) que opera ao nível da camada de aplicação e que faz uso do estilo ReST [FSHB]. Outros exemplos incluem os protocolos RestMS[1] e uma extensão ao protocolo XMPP denominada *Binary XMPP* [MSAF08].

[1]Página principal do protocolo disponível a partir da página de Internet http://www.restms.org/

Referências

[3GP07] 3GPP. 3GPP TR 22.868: Technical Specification Group Services and System Aspects; Study on Facilitating Machine to Machine Communication in 3GPP Systems;. Technical report, 2007.

[3GP11] 3GPP. Technical Specification Group Services and System Aspects; Service requirements for Machine-Type Communications. 0(Release 10):1–17, 2011.

[BHM+04] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris e Orchard David. Web Services Architecture. 2004.

[DJ 03] Piers Harding DJ Adams. XEP-0024: Publish/Subscribe, 2003.

[DVFE] Dominique Guinard, Trifa Vlad, Mattern Friedemann e Wilde Erik. 5 From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices 1.

[DZ83] John D Day e Hubert Zimmermann. The OSI reference model. *Proceedings of the IEEE*, 71(12):1334–1340, 1983.

[ETS] ETSI. Machine-to-Machine communications (M2M); Functional architecture.

[FSHB] Brian Frank, Zach Shelby, Klaus Hartke e Carsten Bormann. Constrained Application Protocol (CoAP).

[GSB] Sergio Gusmeroli, Harald Sundmaeker e Alessandro Bassi. Internet of Things Strategic Research Roadmap. *Energy*, pages 9–52.

[Hin] Pieter Hintjens. ØMQ - The Guide.

[HMESA08] Joe Hildebrand, Peter Millard, Ryan Eatmon e Peter Saint-Andre. XEP-0030: Service Discovery, 2008.

[I. 11] I. Fette. The WebSocket Protocol, 2011.

[LLLS11] Rongxing Lu, Xu Li, Xiaohui Liang e Xuemin Sherman Shen. COMMUNICATIONS GRS : The Green , Reliability , and Security of Emerging Machine to Machine Communications. *IEEE Communications Magazine*, 49(April):28–35, 2011.

[MSAF08] Pedro Melo, Peter Saint-Andre e Fabio Forno. Binary XMPP, April 2008.

[MSAM12] Peter Millard, Peter Saint-Andre e Ralph Meijer. XEP-0060: Publish-Subscribe, 2012.

[Nyg94] David F. Nygaard. World population projections, 2020. 2020 vision briefs 5, International Food Policy Research Institute (IFPRI), 1994.

[Pet07] Saint-Andre Peter. Jabber Software Foundation Renamed to XMPP Standards Foundation, 2007.

[Poi] Tutorials Point. What are webservices.

[Pru12] João Carlos Figueiredo Rodrigues Prudêncio. Application for Continuous Health Monitoring using Machine to Machine Communications, 2012.

[Rab] RabbitMQ. AMQP 0-9-1 Model Explained.

[RIJ99] R. Fielding, UC Irvine e J. Gettys. Hypertext Transfer Protocol – HTTP/1.1, 1999.

[Roy00] Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures, 2000.

[SA11] Peter Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core, 2011.

[SGFW10] Harald Sundmaeker, Patrick Guillemin, Peter Friess e Sylvie Woelfflé. Vision and Challenges for Realising the Internet of Things, 2010.

[Sun10] Harald Sundmaeker. *Vision and Challenges for Realising the Internet of Things*, 2010.

[Teh] Adrian Teh. XMPP Illustrated: Getting to Know XMPP.

[Uni05] International Telecommunication Union. ITU Internet Reports 2005: The Internet of Things - Executive Summary, 2005.

[WTJ+11] Geng Wu, Shilpa Talwar, Kerstin Johnsson, Nageen Himayat e Kevin D. Johnson. M2M: From Mobile to Embedded Internet, 2011.

[YB11] Alison Young e Lana Brindley. Messaging User Guide, 2011.