PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Luiz Guilherme de Oliveira Pitta

Uma abordagem para o problema de conectividade em plataformas multilaterais de IoT

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Markus Endler

Rio de Janeiro

Março de 2018

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Luiz Guilherme de Oliveira Pitta

Uma abordagem para o problema de conectividade em plataformas multilaterais de IoT

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Markus Endler
Orientador
Departamento de Informática - PUC-Rio

Profª. Noemi de La Rocque Rodriguez
Departamento de Informática - PUC-Rio

Prof. Jose Viterbo Filho
UFF

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro
Técnico Científico - PUC-Rio

Rio de Janeiro, 28 de março de 2018

Luiz Guilherme de Oliveira Pitta

O autor graduou-se em Engenharia da Computação pela PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro) no ano de 2015. Ingressou no Mestrado em Informática na PUC-Rio no ano de 2016.

Ficha Catalográfica

Pitta, Luiz Guilherme de Oliveira

Uma abordagem para o problema de conectividade em plataformas multilaterais de IoT / Luiz Guilherme de Oliveira Pitta; orientador: Markus Endler. - 2018.

86 f. : il. (color); 30 cm

1. Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2018.

Incluí referências bibliográficas.

1. Informática - Teses. 2. Internet das Coisas. 3. Sistemas Distribuídos. 4. Sensores. 5. Atuadores. 6. Serviços Preditivos e Prescritivos. 7. Middleware. 8. Computação Móvel. I.Endler, Markus. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Agradeço a Deus por me guiar até aqui e me permitir a finalização de minha pesquisa de mestrado.

Agradeço profundamente aos meus pais Luiz Enio e Hebe por todo o apoio que me deram, pelo carinho que sempre esteve presente e, também, pelos puxões de orelha necessários. Sem todo o afeto deles, esse caminho teria sido bem mais difícil.

Agradeço ao meu tio Paulo pelo grande apoio e pelos importantes conselhos dados ao longo do mestrado e que muito me ajudaram.

Agradeço, em especial, ao meu orientador Professor Markus Endler por todo o apoio, disponibilidade, puxões de orelha, incentivos e importantes conversas que fizeram toda a diferença nesses dois anos de mestrado.

Agradeço aos professores Marco Molinaro, Noemi Rodriguez e Simone Diniz pelos significativos ensinamentos passados durante as matérias do mestrado ou em discussões à parte.

Agradeço aos professores tidos durante minha graduação na PUC-Rio dando base a minha formação acadêmica e permitindo meu continuar no mestrado.

Agradeço ao Instituto Gênesis do qual fiz parte, por seis meses, ao lado dos amigos Marcel, Marcelle e Tadeu recebendo orientações ao nosso projeto do aplicativo Tymo e, também, aos ensinamentos sobre negócios utilizados no desenvolvimento desse meu projeto do mestrado.

Agradeço, também, a todos os colegas do LAC pelo apoio, acolhimento e ajuda com as discussões e sugestões que ajudaram no desenvolvimento do meu trabalho. Em especial gostaria de agradecer ao Felipe Carvalho pelas dicas, assistência e apoio acadêmico dado ao longo do mestrado.

Agradeço ao grupo da secretaria do Departamento de Informática da PUC-Rio pelo apoio e suporte contínuo desde a época da minha graduação até a conclusão do meu mestrado.

Agradeço, em especial, ao meu grande amigo Tadeu, que desde a época da graduação sempre deu um grande apoio com suas valiosas dicas e a toda hora esteve disposto a ajudar no processo de construção do meu trabalho de mestrado.

Agradeço, também, aos amigos, Pedro, Fernando e Bruno que desde a graduação sempre me ajudaram e que durante o mestrado conseguimos manter a mesma parceria. Também agradeço a todos os colegas que estiveram junto comigo nas aulas e nos trabalhos em grupo; com certeza tudo isso somou para que eu pudesse crescer academicamente.

Agradeço, em especial, aos meus grandes amigos Diego, Jonas, Leão e Rachel pelos conselhos, paciência e apoio. Tenho toda certeza de que fizeram minha jornada mais fácil e tranquila.

Agradeço, também, a todos os meus amigos, que direta ou indiretamente, partilharam da minha trajetória e contribuíram com parceria e amizade nesse período do mestrado.

Agradeço, por fim, a PUC-Rio inclusive pela bolsa de isenção do mestrado e o INCT-CNPq pela bolsa de apoio financeiro.

Resumo

Pitta, Luiz Guilherme de Oliveira; Endler, Markus (Orientador). **Uma abordagem para o problema de conectividade em plataformas multilaterais de IoT.** Rio de Janeiro, 2018. 86p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A popularização da Internet das Coisas (IoT) abriu uma série de oportunidades para a geração de novas aplicações que não eram possíveis anteriormente. No cenário atual de IoT existem *marketplaces* que vendem soluções completas para os clientes com objetos inteligentes, *gateways* para a transmissão dos dados e provedores que analisam estes por uma taxa de assinatura. Partimos da visão de que no futuro deverá ocorrer uma "uberização" de IoT, onde cada pessoa poderá oferecer dados de sensores e acesso a atuadores para outra e que eles estarão categorizados com base no QoS dos objetos que os fornecem, similarmente como são classificadas *commodities* hoje. Além disso, haverá plataformas multilaterais onde essas informações poderão ser negociadas em combinação com provedores de conectividade, para transmitir os dados, e de análise. Uma plataforma que fornece esse serviço deve garantir que o fluxo de dados (e do estado) de objetos seja contínuo, sem expor para o cliente algum problema de conectividade entre eles e os provedores. Ou seja, ela deve ter mecanismos para detectá-los e rapidamente selecionar novos provedores, isso dentro de um cenário de intensa troca de dados. Este trabalho apresenta como contribuições um mecanismo de detecção contínuo de problemas de conectividade que utiliza o paradigma *Publish-Subscribe* para o envio de mensagens de identificação de problemas e uma solução arquitetural de uma plataforma baseada em conceitos de *marketplaces* para IoT, que inclui os serviços de "comoditização" dos provedores de serviço e o *matchmaking* para selecionar uma combinação destes para prestar serviços para o cliente. Um estudo de caso no domínio de *marketplaces* é conduzido, com a análise dos serviços da plataforma em vários cenários de testes e a avaliação do mecanismo de detecção de problemas de conectividade, com a simulação de diferentes falhas na conexão.

Palavras-chave

Internet das Coisas (IoT); Dados como Commodities; Sensores; Atuadores; Serviços Preditivos e Prescritivos; Middleware; Computação Móvel.

Abstract

Pitta, Luiz Guilherme de Oliveira; Endler, Markus (Advisor). **An approach to the connectivity problem in multilateral IoT platforms.** Rio de Janeiro, 2018. 86p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The popularization of the Internet of Things (IoT) opened up a series of opportunities for the generation of new applications that were not previously possible. In the current scenario of IoT there are marketplaces that sell complete solutions for users with smart objects, gateways for data transmission and providers that analyze these for a subscription fee. We start from the view that in the future an "uberization" of IoT should occur, where each person can offer sensor data and access to actuators to another and that they will be categorized based on the QoS of the objects that provide them, similarly as commodities are classified today. In addition, there will be multilateral platforms where this information can be negotiated in combination with connectivity providers, to transmit data, and analytics. A platform that provides this service must ensure that the data (and state) flow of objects is continuous, without exposing to the user any connectivity problems between them and the providers. That is, it must have mechanisms to detect problems and quickly select new providers, all this in a scenario of intense data exchange. This work presents as contributions a continuous connectivity problem detection mechanism that uses a Publish-Subscribe paradigm to send problem identification messages and an architectural solution of a platform based on marketplaces concepts for IoT, which includes the "commoditization" of service providers and a matchmaking service to select a combination of these to provide services to the customer. A case study in the domain of marketplaces is conducted, with the analysis of the services of the platform with several tests scenarios and the evaluation of the mechanism of detection of connectivity problems, with the simulation of different connection failures.

Keywords

Internet of Things (IoT); Data as Commodities; Sensors; Actuators; Predictive and Prescriptive Services; Middleware; Mobile Computing.

Sumário

Lista de figuras

Lista de tabelas

Have the courage to follow your heart and intuition. They somehow know what you truly want to become.

Steve Jobs

1
Introdução

O mercado de Internet das Coisas (IoT, *Internet of Things*) cresceu muito rapidamente nos últimos anos. *Startups* e grandes empreendimentos tem investido pesadamente nesse tipo de tecnologia [1] mas, apesar disso, as empresas atualmente não utilizam a maioria dos dados IoT que coletam [2], ou seja, grande parte dessas informações que poderiam ser analisadas e processadas para gerar algum valor e, por exemplo, serem utilizadas para melhorar os processos internos das empresas, são inutilizadas e descartadas.

Nesse contexto, IoT, uma das grandes revoluções e paradigmas que a computação está encarando hoje [3], aparece como uma visão de estender a conectividade a dispositivos cada vez menores e com baixo poder de processamento**,** também conhecidos como objetos/coisas móveis inteligentes (M-OBJs). Continuamente é gerado uma quantidade imensurável de informações pelas pessoas e/ou sensores e com IoT se tornando mais acessível, um número maior de pessoas e empresas terão a possibilidade de descobrir, através de sensores, dados úteis sobre o meio em que estão e que possa ser de interesse de terceiros. Por exemplo, alguns clientes podem estar interessados em dados relacionados a um determinado bairro da cidade, enquanto outros podem querer dados quantitativos coletados do número de pessoas em um local específico, como em um hospital ou em uma repartição pública. Os *marketplaces* digitais são sistemas que conectam provedores e consumidores de serviços que tratam de registros, conjuntos e fluxos de dados, garantindo alta qualidade, consistência e segurança num único lugar.

Alguns trabalhos sobre IoT já foram publicados, prevendo que nos próximos 10 anos a inovação será extremamente orientada a dados[1] e que eles serão a principal *commodity* do mercado, ou seja, serão precificados e comercializados como ouro, minério e petróleo são vendidos hoje em dia. Estima-se que no futuro o funcionamento da economia mundial será extremamente orientada aos dados gerados

[1] Joe McCann. Data Is The Most Valuable Commodity On Earth, March 2016. - http://sub-print.com/blog/data-is-the-most-valuable-commodity-on-earth

por IoT**.** Nessa linha, assumimos que os dados e informações sobre os lugares, as coisas e as pessoas serão o principal ativo da nossa futura sociedade e que os objetos físicos e os ambientes computacionais baseados na nuvem serão os recursos que permitirão a interação inteligente entre os meios físicos e digitais. Os dados sobre o estado de lugares, de pessoas e das coisas no mundo real são um bem valioso que serão negociáveis como produtos e serviços são negociados hoje. Apesar de algumas plataformas digitais com foco em comercialização de produtos e serviços de IoT terem sido desenvolvidas ao longo dos últimos anos, nenhuma delas possui flexibilidade e customizações destes produtos e serviços fornecidos. Por exemplo, caso o cliente deseje adquirir um pacote de monitoramento de jardim, composto de alguns sensores, com análise de dados, ele deve pagar pela solução completa e caso queira alguma modificação poderá ter que pagar uma taxa extra. Em outras palavras, apesar da crescente proliferação de aplicações usando objetos móveis inteligentes não existem padrões estabelecidos e tecnologias consolidadas para IoT em um nível global, sendo ainda um desafio extrair informações e deter o controle de atuadores num mundo com bilhões de coisas/objetos inteligentes, estacionários ou móveis (que chamaremos de Mobile Objects - M-OBJ).

O conceito de um *marketplace* para IoT tem como objetivo tornar públicos os dados e as informações obtidas através de provedores de dados IoT para trazer uma plataforma (para ofertas e procura) onde os proprietários de dispositivos inteligentes (M-OBJs) se conectem a clientes interessados em ter acesso aos dados gerados pelos seus dispositivos e/ou em adquirir controle de atuação sobre eles, similarmente a Google Play que que conecta usuários Android (compradores) com desenvolvedores de aplicativos móveis. Dentro desse conceito, podemos ilustrar alguns cenários em que os clientes do *marketplace* se beneficiariam da aquisição apenas de dados brutos de sensores, contratar um processamento ou análise estatística desses dados de sensores ou adquirir acesso (exclusivo) a um atuador:

- **Dados Brutos:** Hoje em dia, muitas pessoas escolhem por ter um estilo de vida mais saudável e praticam esportes regularmente. Por exemplo, um cliente do *marketplace*, que mora em uma grande metrópole, deseja saber qual caminho mais tranquilo para ele poder dar uma caminhada ou corrida sem ser atrapalhado pelo trânsito da rua e pela quantidade excessiva de pedestres. Para isso, ele pode comprar na plataforma os dados brutos de

densidade instantânea de pessoas na rua em vários pontos da cidade e saber em tempo real quais destes estão mais tranquilos e vazios para ele praticar a sua atividade física com mais tranquilidade.

- **Análise de Dados:** Uma família planeja viajar e para decidir o próximo destino precisam saber as condições da qualidade do ar devido ao histórico familiar de alergia à poeira e a sensibilidade às mudanças constantes de temperatura. Devido a isso vão precisar escolher um local onde a qualidade do ar seja melhor, com uma umidade relativa do ar média e temperaturas moderadamente estáveis. No *marketplace* poderão então existir provedores de dispositivos embarcados que disponibilizam para a compra os dados de sensores de temperatura, umidade e de estações de monitoramento de ar sem fio (*Wireless Air Monitoring Stations*[2], WAMS), com o monitoramento de CO, NO2, SO2, sensor de chumbo, etc.; portanto, essa família pode comprar os dados de várias cidades que, possivelmente, ela quer visitar e adquirir junto um serviço de análise de dados, que irá processar as informações dessas várias localidades e exibir um relatório completo de qual é o lugar mais adequado para eles irem na próxima viagem, se baseando nos parâmetros que foram definidos anteriormente.
- **Atuadores:** Considere o ano de 2025, onde veículos autônomos serão comuns nas ruas [4] e com o *marketplace* será possível alugar esse tipo de veículo com todos os comandos de atuação sobre ele disponíveis na tela do dispositivo móvel do cliente. Neste cenário considere um empresário que precisa se locomover para uma reunião externa de trabalho e deseja

Figura 1.1: Cenário de um cliente obtendo atuação sobre um carro autônomo

[2] Air Quality Monitoring - http://www.environnement-sa.com/produits-2/air-quality-monitoring-aqms-stations/

> alugar um carro que esteja nas redondezas. Para isso, ele abre o aplicativo da plataforma e "anuncia" que necessita de um carro próximo da localização que ele está. Então, ao achar um veículo disponível, o sistema dá ao cliente acesso exclusivo de atuação, pelo período que ele o estiver utilizando e cobrando, por esse serviço, conforme uma tarifa pré-definida, nos moldes de como é feito atualmente nos aplicativos Uber, Cabify, etc.

Os M-OBJs por definição são objetos móveis (*smart things*) e devido as características de aplicações IoT não são necessariamente estáticos. Considerando que os protocolos de conexão de IoT, em sua grande maioria, são baseados em tecnologias wireless de baixo alcance (e.g. *Bluetooth Low Energy*, NFC), basta o M-OBJ se afastar por mais de 30-50 metros do dispositivo em que está conectado que ele irá perder a conexão, sendo que este dispositivo (*smartphone*) está executando um software (M-Hub) para torná-lo um roteador de dados para a nuvem. Esses cenários de mobilidade irrestrita são comuns em IoT e, também, conhecidos como IoMT (*Internet of Mobile Things*, Internet das Coisas Móveis) [5]. Para contornar esse problema devem existir mecanismos que detectem possíveis problemas de conectividade e que restabeleçam a conexão, para que os clientes que adquiriram o serviço não se sintam prejudicados.

Outro aspecto que deve ser levado em consideração num *marketplace* de IoT é a substituição silenciosa (suave) de provedores de dados dos ambientes, pessoas ou coisas. Ou seja, ao tentar reestabelecer o serviço, após problemas de conectividade, o M-OBJ do qual o cliente, anteriormente, estava recebendo ou enviando dados pode não ter sido localizado e outro M-OBJ precise ser usado em seu lugar, mas prestando o mesmo tipo de serviço (e.g. enviar dados de temperatura) e com as mesmas características. Essa propriedade de manter o serviço com os mesmos critérios de qualidade é chamada em computação ubíqua de Qualidade de Serviço (*Quality of Service,* QoS) [12], que é caracterizada por um conjunto de parâmetros ou métricas (e.g. preço limite que o cliente se dispõe a pagar, precisão do sensor, frequência da leitura do sensor, latência máxima da transmissão para nuvem, nivel da bateria, tempo de duração da bateria, etc.) pré-definidos pelo cliente antes do começo da prestação do serviço.

Além disso, é necessário um mecanismo de comunicação flexível, dinâmico e assíncrono para que os clientes recebam os dados dos M-OBJ, tendo em vista a

natureza dinâmica das aplicações IoT. O paradigma *Publish-Subscribe* sustenta a implementação de mecanismos com essas características, pois nele existe o conceito de que as entidades publicadoras postam a informação e aqueles que se subscreveram a recebem através de um barramento de serviços. Em IoT, a comunicação geralmente é feita através de mensagens pequenas, porém frequentes, onde cada mensagem individual é sem importância, mas as propriedades estatísticas dos dados correspondentes possuem as informações relevantes [6].

Para ilustrar a necessidade de se ter um mecanismo que continuamente verifica se existem problemas de conectividade com os M-OBJs, considere o exemplo dos atuadores, descrito nessa seção, onde o cliente alugou um carro autônomo e detém o controle exclusivo de atuação sobre ele. Nesse contexto, existindo uma desconexão o cliente perderia o poder de atuação sobre o carro e, não havendo nenhum mecanismo para identificar esse problema rapidamente e reconectar, o carro estaria disponível para outras pessoas alugarem. Consequentemente, nossa pesquisa tem como objetivo descobrir se é possível criar mecanismos para identificar e corrigir esse problema rapidamente. Corroborando com esse problema, o Laboratory for Advanced Collaboration (LAC) da Pontificia Universidade Católica do Rio de Janeiro (PUC-Rio), vem desenvolvendo o ContextNet [7], um middleware móvel baseado na nuvem para aplicações IoT e o Mobile Hub (M-Hub) [5]. O ContextNet é composto da camada de comunicação SDDL [8, 9, 10,11] que fornece APIs e serviços baseados na nuvem para conectividade de *smartphones*. Já o M-Hub é responsável pela descoberta, conexão e comunicação com objetos inteligentes móveis usando o Bluetooth Low Energy (BLE). O SDDL e o M-Hub serão descritos em mais detalhes no Capítulo 2.

1.1. Definição do Problema

É amplamente reconhecido que IoT causará um grande impacto em quase todos os aspectos de nossos sistemas econômicos e em nossas vidas pessoais. Começamos a partir da premissa de que em alguns anos teremos objetos inteligentes (literalmente) em todos os lugares: em semáforos, placas de ruas, em casas, escritórios e fábricas, em veículos, no solo, em reservatórios de água, como tecnologia vestível

e até mesmo por baixo da nossa pele. Isso significa que haverá um substancial aumento na quantidade de provedores e na variedade dos tipos desses dispositivos, mas diversos desafios ainda são um problema para a adoção em massa desses M-OBJs [13], apesar da crescente popularização. Um desses desafios é, ao estar conectado e recebendo um fluxo de dados de um sensor ou controlando um atuador, acontecer um problema na conexão entre o M-OBJ e o nó móvel (e.g. *smartphones* e *tablets*) e a aplicação IoT não conseguir detectar e acabar interrompendo o serviço que estava sendo realizado. Mecanismos de detecção de problemas de conectividade devem ser executados continuamente e serem invisíveis para o cliente da aplicação, pois ele não deve perceber que aconteceram problemas ao longo da prestação do serviço.

Em aplicações de IoT nem sempre o usuário está conectado diretamente ao M-OBJ, ou seja, ele pode estar recebendo dados ou enviando comandos através de um *gateway*, que realiza o encaminhamento dessas informações para um nó móvel que, esse sim, está conectado ao M-OBJ. A detecção de problemas contínuos de conectividade não deve levar em consideração apenas o estado da conexão do M-OBJ, mas também considerar se o nó móvel, que encaminha as informações dele para o *gateway*, está conectado à internet no momento desse envio e, isto porque, mesmo se o M-OBJ estiver conectado corretamente ao nó móvel, o cliente do outro lado, que espera receber alguma confirmação, não irá recebê-la por causa desses outros problemas.

Tendo em vista os desafios que existem e ainda estão presentes em IoT, além das características importantes que um mecanismo de detecção contínua de problemas de conectividade em IoT, que leva em consideração o QoS [14] dos serviços prestados deve ter, este trabalho propõe uma extensão ao M-Hub em combinação com um serviço na nuvem. Esse serviço irá gerenciar as solicitações dos clientes criando uma plataforma que usa os conceitos de um *marketplace* para IoT e o QoS da aplicação para dar suporte à detecção de problemas de conectividade continuamente. Este aspecto consiste em ter mecanismos que ficam constantemente, tanto do lado do provedor em que o M-OBJ está conectado quanto do cliente que solicitou o serviço, verificando se aconteceu alguma perda de sinal de rede por parte dos nós móveis ou pela desconexão de algum M-OBJ para que seja notificada imediatamente a parte em que aconteceu o problema e iniciar o processo de restabelecimento do serviço. Assim, o problema técnico que estamos tratando nesta dissertação é a

substituição silenciosa e automática dos provedores de serviço, numa plataforma baseada em conceitos de um *marketplace* para IoT, que apresentam problemas de conectividade, desde a detecção do problema em sua conexão até a seleção e substituição deles por novos para que se tenha continuidade no serviço de fluxo de dados prestado.

O mecanismo de detecção contínua de qualidade de conectividade auxilia o desenvolvimento de aplicações que precisam manter um fluxo de dados constante, consistente e ininterrupto, como no exemplo da aplicação onde o cliente aluga um carro e detém o controle exclusivo sobre um atuador (as fechaduras e a ignição) do veículo descrita na Introdução do Capítulo 1. Definir esse mecanismo proporciona alguns desafios, já que não existem padrões definidos e estabelecidos em IoT [15] e, também, pelo fato de que a detecção precisará ser feita tanto no M-OBJ quanto nos nós móveis. O desafio será integrar tecnologias para criar uma estrutura de detecção contínua usando o paradigma *Publish-Subscribe* [16] que permite criar uma comunicação flexível, dinâmica e assíncrona entre dispositivos através de um barramento de serviço, os *Listners* do M-Hub que detectam quando um M-OBJ se desconectou e, por fim, o conceito de *Threads* para criar um serviço que verifica continuamente o sinal de internet de um nó móvel. Esta dissertação propõe investigar e responder as seguintes perguntas:

- É plausível criar um mecanismo para detecção contínua da qualidade de conectividade em ambientes e aplicações de IoT com mobilidade irrestrita que necessitam de um fluxo de dados constante, consistente e ininterrupto?
- É possível utilizar os conceitos de um *middleware* de comunicação *Publish-Subscribe* para enviar e receber mensagens que identificam, em tempo real, se houve perda de conexão por alguma das partes (M-OBJ ou nó móvel) no momento da execução de um serviço numa plataforma que utiliza os conceitos de um *marketplace* para IoT?
- É viável criar um algoritmo de *matchmaking* escalável para a seleção da melhor combinação de elementos possíveis (e.g. M-OBJs, provedor de conectividade e análise de dados) para a realização de monitoramento ou controle em alguma aplicação IoT rodando em tempo real?

1.2. Objetivos

O principal objetivo desta dissertação é propor um mecanismo para detecção contínua da qualidade de conectividade numa plataforma que utiliza os conceitos de um *marketplace* para IoT e, para isso, desenvolvemos um serviço de *marketplace* IoT e estendemos o M-Hub em duas frentes. Uma para detectar problemas de conexão entre os dispositivos conectados nele e, na outra, desenvolvemos uma API de comunicação com o servidor, para gerenciar a sua função (ou papel) como provedor de conectividade, no ecosistema de IoT, conforme descrito na Seção 1.1. Propomos aqui uma abordagem construída na plataforma Android para os nós móveis (*smartphones*) e em Node.js para o para o servidor responsável pelo serviço de *marketplace* que gerencia as solicitações dos clientes. Dessa forma é possível ter uma arquitetura modularizada que consegue detectar e informar ao cliente se o objeto inteligente, observado por ele, está com problemas de conexão e rapidamente ter uma solução para contorná-lo.

Este trabalho não teve como objetivo explorar os possíveis problemas de segurança da informação relativas a autenticação segura, ao acesso e compartilhamento de *smart things* e M-Hubs, bem como de sigilo e integridade dos dados dentro de uma plataforma baseada em conceitos de um *marketplace* para IoT. Além disso, não focamos no consumo de energia dos provedores pois este tópico já foi abordado em [5], devido eles serem baseados no M-Hub.

1.3. Contribuições

As principais contribuições desta dissertação são:

- A definição de uma arquitetura modularizada que utiliza conceitos de uma plataforma multilateral de IoT englobando os serviços de comoditização de dados, monitoramento de QoS, reputação e incentivos.
- Uma API para seleção da melhor combinação entre provedores de serviços para monitoramento ou controle em aplicações IoT com base em parâmetros de QoS e de geolocalização.
- A definição de métodos para a detecção de problemas de conectividade em objetos inteligentes e nós móveis num ambiente de alta mobilidade.

1.4. Metodologia

Um estudo inicial sobre plataformas multilaterais, modelos de negócios e detecção de problemas de conectividade em IoT foi realizado para pautar os trabalhos relacionados existentes e, também, o estado da arte na pesquisa. O estudo buscou compreender todos os aspectos que envolvem a construção e elaboração da arquitetura de uma plataforma de negócios digital, além de investigar as limitações e desafios da detecção de problemas de conectividade em dispositivos de IoT. Para a realização dos testes, dada a natureza desta dissertação, foi desenvolvida uma aplicação hipotética que simula as interações entre os clientes e os provedores de serviço em uma plataforma digital multilateral de IoT utilizando como base a API estendida do M-Hub e um servidor na nuvem como o gerenciador dos serviços dessa plataforma.

1.5. Organização da Tese

A sequência desta dissertação está organizada da seguinte maneira:

- **Capítulo 2 – Conceitos Fundamentais:** Neste capítulo são apresentados todos os conceitos e tecnologias que foram utilizados na fundamentação desta pesquisa
- **Capítulo 3 – IoTrade: Um Marketplace para IoT:** Neste capítulo são apresentados e discutidos a arquitetura, os principais conceitos de uma plataforma multilateral para IoT e uma implementação baseada neste conceito.
- **Capítulo 4 – Detecção de Problemas de Conectividade:** São apresentados os principais métodos para detectar a perda de conectividade de um dispositivo IoT no âmbito de uma plataforma que utiliza conceitos de um *marketplace* para IoT.
- **Capítulo 5 – Avaliação:** Este capítulo apresenta e discute os resultados da avaliação obtidos dos testes de performance, que englobam o algoritmo de *matchmaking*, a detecção de problemas de conectividade e o envio para múltiplos clientes a partir de um único M-Hub. Também, são apresentados alguns

cenários com base na implementação do *marketplace* proposto nesta dissertação.

• **Capítulo 6 –Trabalhos Relacionados:** Neste capítulo são apresentados e discutidos alguns trabalhos relacionados a detecção de problemas de conectividade e a plataformas digitais multilaterais para IoT.

• **Capítulo 7 – Conclusão e Trabalhos Futuros:** São apresentadas as conclusões sobre o trabalho e pesquisas futuras são sugeridas.

2
Conceitos Fundamentais

Este capítulo apresenta os conceitos fundamentais e tecnologias utilizadas nesta dissertação. Na Seção 2.1 será apresentado o que é uma plataforma multilateral e os seus conceitos, que foram a base para o *martketplace* proposto neste trabalho. Na Seção 2.2 será apresentado o modelo *Sensing as a Service* que influenciou a modelagem do esquema de precificação do IoTrade. Na Seção 2.3 será discutido um modelo de negócios para IoT e alguns exemplos. Na Seção 2.4 será apresentado o conceito do algoritmo de *matchmaking* e como ele é aplicado. Na Seção 2.5 será apresentado o conceito de comoditização de dados. Na Seção 2.6 será apresentado o paradigma *Publish-Subscribe* (Publicação-Subscrição). Na Seção 2.7 será apresentado o *middleware* SDDL, responsável pela camada de comunicação para o envio e recebimento de dados entre os dispositivos. Por último, na Seção 2.8 será apresentado o Mobile Hub (M-Hub), que foi estendido para suportar a detecção contínua de problemas de conectividade entre dispositivos móveis neste trabalho.

2.1.
Plataforma Multilateral

Plataforma Multilateral é um padrão de modelo de negócios que conecta membros de um grupo com outros grupos [18], como motoristas procurando por passageiros. Este modelo representa valor para um dos grupos participantes apenas se o outro estiver presente; além disso ele cria valor facilitando a interação entre diferentes lados e cresce na medida em que atrai mais clientes [19]. Esse tipo de paradigma existia bem antes do aparecimento de plataformas digitais multilaterais como Uber, Cabify, Amazon, Google e Facebook, apesar de que o crescimento da tecnologia e da informática fizeram com que elas se proliferassem em maior escala. Outros exemplos, que nasceram antes dessa era digital, são: Operadoras de cartão de crédito (comerciantes e donos de cartões), Microsoft Windows (clientes, fabricantes de hardware e de software) e fabricantes de vídeo game (clientes e produtores de jogos).

Figura 2.1: Exemplo do conceito de plataforma multilateral

Algumas características [20] bem conhecidas e que precisam existir para que esse tipo de plataforma funcione bem são espessura (isto é, deve ter número suficiente de participantes), confiabilidade (as regras devem ser bem definidas e confiáveis) e equilíbrio (ou seja, deve haver uma relação adequada entre provedores e consumidores). Para essas características poderem existir é preciso, antes de construir a plataforma, fazer um mapeamento para estudar o mercado e descobrir os "atritos" de interação que existem nele, identificando os possíveis lados e atividades de cada um para compor o modelo. Feito isso, a plataforma multilateral criada consegue realizar duas funções fundamentais que são a redução de custos de busca e reduzir os custos transacionais compartilhados entre os múltiplos lados [21], ou seja, esse modelo permite que um cliente consiga, num espaço único, achar o que ele procura mais facilmente reduzindo o seu tempo e custo de buscar por estar em um ambiente compartilhado por todos os provedores. Por exemplo, a empresa Opentable oferece uma plataforma online para os clientes fazerem reservas em restaurantes. Antes dela existir os clientes tinham que ligar para cada restaurante que eles tinham interesse e perguntar se tinham mesas disponíveis no dia e hora em que eles desejavam, ou seja, essa plataforma conseguiu diminuir os custos de busca dos clientes (custo da ligação e do tempo de busca do restaurante) e reduziu os custos dos restaurantes (manter um sistema próprio de reservas).

Além disso, são necessárias estratégias para poder se manter fazendo sucesso ou crescer ainda mais. Uma das estratégias mais usadas é não cobrar nada de um dos lados como forma de atraí-los para então poder fornecer esta base de clientes ao outro lado, estratégia usada, por exemplo, pela Opentable, citada acima, onde os

clientes que procuram restaurantes não pagam nada pela reserva. Segundo [22], as plataformas multilaterais que adotam uma estratégia combinada e baseada em inovação juntamente com forte penetração de mercado são aquelas que tem maior taxa de sucesso. Por exemplo, a Uber utiliza essa estratégia combinada e alavancou muito sua popularidade desde sua criação em 2009. A estratégia consiste em ir se instalando nas cidades e tomar conta do mercado antes de negociar com o governo local e pedir permissão para operar, ou seja, utiliza uma estratégia bem agressiva de penetração de mercado. A estratégia de inovação da Uber é sempre ir criando novos serviços, como o Uber Eats (serviço de entrega similar ao iFood), Uber Pets (viagens com animais de estimação), UberX (corridas mais baratas em carros mais simples), entre outros.

2.2. Sensing as a Service

O número de soluções IoT nos últimos anos tem crescido muito no mercado e cada vez mais as empresas têm investido em objetos inteligentes que se conectam a internet e conseguem realizar medições e analisar dados. A Xiaomi, por exemplo, tem se esforçado para criar um ecossistema de objetos inteligentes que interagem entre si e entre eles estão vários sensores para automatização de casas (sensores de qualidade do ar, presença e luminosidade) que se conectam a uma central na casa que envia os dados para o *smartphone* do dono e, com isso, ele pode tomar decisões e se manter informado à distância.

O modelo *Sensing as a Service* se baseia em uma visão e em um modelo de negócios que promove a negociação de dados entre os consumidores e produtores de dados [23]. Imagine que além dos sensores citados acima, existam outros espalhados pela casa, como RFIDs (*Radio-Frequency IDentification*) que compartilham, com a permissão do dono, os dados de tudo que a pessoa tem pela casa. Em troca empresas podem acessar esses dados e oferecer ofertas de produtos ou pagar pelo fluxo constante de dados que ele pode fornecer para elas.

Esse modelo fornece um estímulo para as pessoas adotarem soluções IoT e negociarem os dados gerados por elas, além de ajudar a recuperar o custo do investimento neste tipo de solução. Entretanto, esse modelo por si só não garante a adoção em massa pelas pessoas, pois as soluções IoT tem de ser boas o suficiente para

motivar, em primeiro lugar, as pessoas a comprá-las e, consequentemente, esse modelo forneceria uma motivação extra.

2.3. Modelo de Negócios

Uma plataforma multilateral precisa de um modelo de negócios eficiente para sobreviver no mercado. Ela sendo de IoT é necessário definir um esquema de precificação com uma cobrança inteligente para incentivar o *marketplace*. O *Sensing as a Service* (S²aaS) apresentado em [23, 28] propõe um modelo que encoraja as pessoas a adotarem soluções IoT e participarem em trocas de dados, já que ajuda os donos destes dados a recuperar o custo que eles tiveram ao adquirir e manter suas soluções IoT. Este tipo de modelo de negócios pode ser um motivador para o aumento do uso e do compartilhamento de dados nas soluções IoT que estão sendo oferecidas no mercado, pois usa incentivos para que as pessoas continuem voltando e usando a plataforma.

Sendo uma abordagem geral para armazenar de forma segura as informações de transações, negócios ou perfil de forma descentralizada e imutável, o *Blockchain* [29] é o encaixe natural para *marketplaces* de IoT, onde provedores e consumidores terão que negociar dados, controle de atuadores e serviços em troca de alguma moeda, que pode ser *Bitcoin* [30] ou qualquer outro meio de pagamento eletrônico. Mas atualmente, o *Blockchain* leva alguns minutos para validar uma transação, o que em relação a IoT é considerado muito tempo, pois todas as transações são realizadas em praticamente tempo real. Apesar do *Blockchain* não ser eficiente para ser usado na plataforma, dentro de sua parte financeira, ele pode ser usado para armazenar e validar dados de forma segura como é mostrado em [31]. Este mecanismo poderia ser usado no momento do registro dos provedores e M-OBJs no *marketplace* para a autenticação de seus serviços, garantindo ao cliente que ele está contratando um serviço de qualidade verificado por um meio seguro. Isto é, usando esse método podemos validar transações no *marketplace* se baseando no fato de que os dados trocados na plataforma já foram certificados por uma entidade externa.

2.4. Matchmaking

O processo de *matchmaking* [20] é usado em plataformas multilaterais para realizar uma combinação que melhor se adequa entre as partes interessadas. O serviço de *matchmaking* não acha literalmente a combinação perfeita, mas sim bons potenciais candidatos para fazerem negócio. Por exemplo, o Uber utiliza esse algoritmo para achar o melhor par motorista-passageiro.

Ao iniciar o desenvolvimento de uma plataforma que necessite de um serviço de *matching* é necessário definir bem o domínio, conhecer as características, necessidades e expectativas dos atores que vão participar do *matchmaking* e construir toda uma infraestrutura de suporte para este processo. O problema de não realizar um bom mapeamento, antes da construção do algoritmo, é ele não estar de acordo com as necessidades dos clientes e devolver combinações incorretas, frustrando os clientes. No campo de IoT é necessária uma análise mais acurada para o desenvolvimento do algoritmo em comparação com outros serviços (e.g. Uber, AirBnB, Amazon Marketplace) pois cada objeto inteligente tem uma especificação, fabricante e parâmetros diferentes e que afetam a precisão, latência e acurácia de cada medida sensoriada e, por consequência, os resultados retornados. Ou seja, o conceito de *matchmaking* é equivalente para todas as plataformas que o utilizam, entretanto, cada uma utiliza desenvolve seu próprio serviço de *matching* e customiza especialmente para o tipo de mercado em que está atuando pois cada um tem critérios e públicos diferentes.

2.5. Comoditização de Dados

O conceito de *commodity* se refere a produtos de qualidade e características uniformes que independem do produtor ou origem, sendo seu preço determinado pela oferta e procura internacional. Exemplos de *commodities* são petróleo, minério de ferro, soja, ouro e entre outras.

Os dados produzidos por IoT são uma fonte muito importante de informação para tentar entender melhor a sociedade em que vivemos e os padrões de eventos que ocorrem no mundo físico e no mundo digital. A venda desses dados gerados

através de infraestruturas e dispositivos IoT será algo comum no futuro[3] e eles serão classificados de acordo com a qualidade, confiabilidade, precisão, complexidade, disponibilidade, energia residual, etc., assim como as *commodities* de hoje são.

2.6. O Paradigma Publish-Subscribe

O paradigma baseado em eventos *Publish-Subscribe* traz vantagens por sustentar a implementação de um mecanismo de comunicação flexível, dinâmico e assíncrono, pois nele as entidades publicadoras publicam mensagens em um tópico, através de um barramento de serviços, e aqueles que se subscreveram nele a recebem por esses mesmo barramento [24]. A vantagem desse mecanismo é que quando vários clientes do IoTrade estão recebendo um fluxo de dados do mesmo provedor de conectividade, ele só precisa postar o dado uma única vez no barramento e todos os clientes recebem a mesma informação, ou seja, é enviada uma única mensagem com múltipla recepção. Outra vantagem desse paradigma é o desacoplamento referencial, isso é a impossibilidade (e não necessidade) das partes comunicantes se conhecerem mutuamente, o que vai de encontro com a ideia de um *marketplace* onde provedores e consumidores não precisam se conhecer, mas apenas serem informados sobre a qualidade (ou a categoria) do dado e da conectividade sendo contratada. Na figura 2.2 é possível observar a interação entre os elementos de um sistema *Publish-Subscribe*.

Ao escolher esse modelo, o objetivo é reduzir o número de mensagens enviadas (exigência decorrente de um paradigma *Publish-Subscribe*) enquanto mantendo o QoS, como a garantia de entrega. A intenção é usar estas duas vantagens

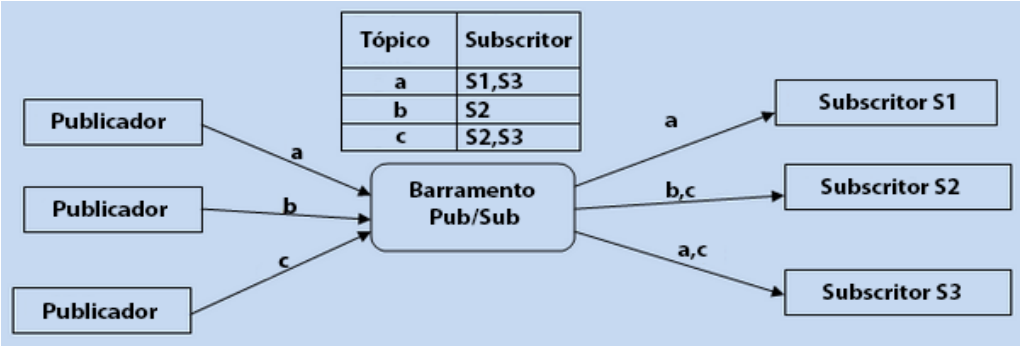


Figura 2.2: Um sistema *Publish-Subscribe* simples

[3] Erel Rosenberg. Data as a commodity: the challenge, June 2016. - http://www.dfrc.ch/data-as-a-commodity/

do mecanismo de comunicação tanto no envio do fluxo de dados constante entre os provedores de serviço e o cliente, quanto no envio de mensagens de detecção de problemas de conectividade. Devido a todas essas características favoráveis do *Publish-Subscribe*, escolhemos esse mecanismo de comunicação para o nosso trabalho.

2.7. Middleware SDDL

O *middleware* SDDL (*Scalable Data Distribution Layer*) é responsável por promover um mecanismo de trocas de dados escalável, em tempo real, confiável, de alto desempenho e interoperável [7, 8, 9] usando o padrão *Publish-Subscribe* baseado em tópicos, descrito na Seção 2.5. Esse *middleware* é uma extensão do *Object Management Group* DDS para nós móveis usando uma abordagem baseada em *gateways*, como ilustrado na Figura 2.3. O SDDL Core pode acomodar vários tipos de nós, incluindo *gateways*, nós que realizam processamento de dados ou monitorar nós atuadores controlados por humanos. Neste contexto, os *gateways* são intermediários entre a WLAN e os nós móveis e, assim, permitindo que eles interajam entre si. Cada *gateway* é um nó DDS e são utilizados os protocolos de comunicação DDSs e Mobile Realiable UDP (MR-UDP) [25] para gerenciar a comunicação dos nós móveis com o SDDL Core e lidar com as mensagens que eles enviam.

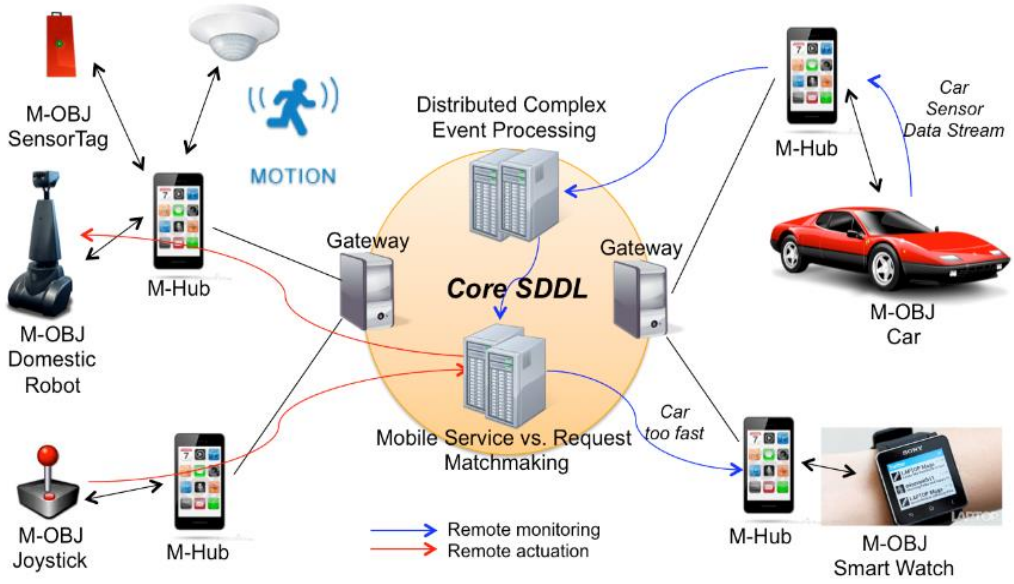


Figura 2.3: Visão geral da estrutura do M-Hub/SDDL

O SDDL também tem componentes que conseguem capturar mensagens que não foram recebidas e consegue enviá-las de novo para o nó destinatário.

Em nosso trabalho é utilizado esse *middleware* para realizar a comunicação de dados entre os M-Hubs, que foram usados como provedores de conectividade e de análise de dados, e os clientes. Todo recebimento de dados ou envio de comandos por parte dos nós móveis passa pelo SDDL Core que roteia a mensagem para o destinatário correto.

2.8. Mobile Hub

O Mobile Hub (M-Hub) é um componente de software que atua como um mecanismo de descoberta física de sensores e atuadores (M-OBJs) e promove a interação entre estes objetos. Além disso, no dispositivo (*smartphone*) que roda o M-Hub é possível realizar o processamento dos dados destes M-OBJs localmente. Nesse contexto o M-Hub implementa as funções de um *gateway* de IoT, dado que permite intermediar a comunicação entre objetos inteligentes e outras aplicações usando a sua conexão com a internet, através de redes como Wi-Fi ou redes celulares. Na figura 2.3 podemos ver um exemplo de uso do M-Hub interagindo com os M-OBJs e utilizando o *middleware* SDDL, apresentado na Seção 2.6, como intermediário para o envio e recebimento de dados.

A Figura 2.4 apresenta os componentes do M-Hub [5]. Nessa arquitetura existe o gerenciador de energia (*Energy Manager*) e quatro serviços principais: o S2PA (*Short-Range Sensor, Presence and Actuation*, Serviço de atuação e presença em sensores de curto alcance), *Location* (Localização), *Connection* (Conexão e o MEPA (*Mobile Event Processing Agent*, Agente de Processamento de Eventos Móveis).

A periodicidade e duração dos serviços do M-Hub são influenciadas pelo nível de bateria do *smartphone* (baixo, médio ou alto). Elas são ajustadas pelo *Energy Manager*, que periodicamente checa o nível de bateria e se o dispositivo está conectado em uma fonte de energia.

O serviço *Location* é responsável por obter a localização atual do M-Hub e pode ser obtida por provedores do *smartphone* como o GPS, triangulação da rede

(Wi-Fi ou rede celular) ou manualmente colocando uma localização, se ela for estática.

O serviço *Connection* é responsável por enviar e receber as mensagens pela nuvem, no formato JSON, através de uma conexão com a internet. As mensagens podem possuir diferentes níveis de relevância. As mensagens importantes são enviadas imediatamente, enquanto as que tem baixa importância são agrupadas para serem enviadas posteriormente. Esse mecanismo de relevância é gerenciado por esse serviço através de um parâmetro de importância nas mensagens (*HIGH, LOW*).

O MEPA é um serviço que realiza o processamento dos dados recebidos dos M-OBJs através de regras CEP. Estas regras precisam ser previamente criadas pelo usuário do M-Hub e podem ser modificadas ou excluídas a qualquer momento. O MEPA está subscrito a todo evento de leitura de dados de sensores que acontece e caso seja detectado um evento complexo, relacionado a estes dados lidos, é enviada uma mensagem alertando o usuário desse fato.

O S2PA é o serviço responsável pela descoberta, conexão, comunicação e monitoramento de M-OBJs próximos conectados através de alguma tecnologia WPAN (*Wireless Personal Area Network*) como *Bluetooth* ou BLE. Para lidar com a diversidades de tecnologias WPAN utilizadas pelos diversos M-OBJs existentes,

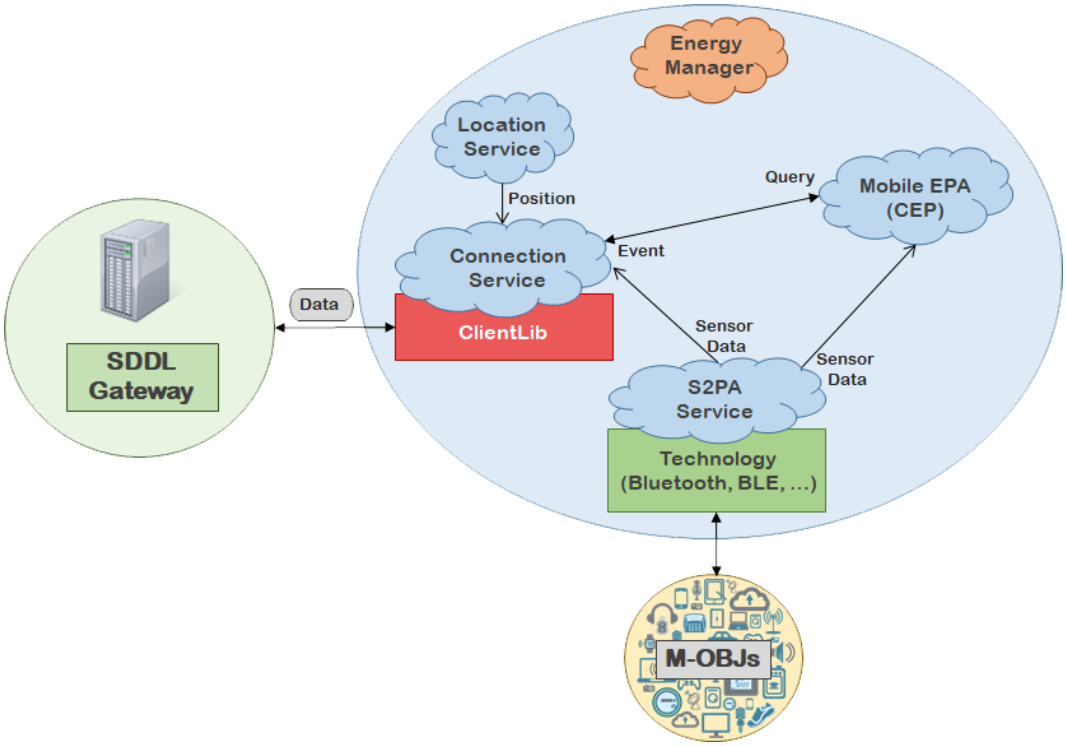


Figura 2.4: Arquitetura do M-Hub

o S2PA define uma API (*Application Programming Interface*) genérica chamada *Technology*, que fornece uma abstração para lidar com a comunicação com os M-OBJs independente do protocolo que eles implementam. No entanto, é necessário realizar a implementação de métodos para cada tipo de protocolo de conexão (e.g. busca de dispositivos, descoberta de serviços, ativação/desativação da interface de rádio, etc.).

Este trabalho estende o M-Hub para dar suporte a detecção contínua de problemas de conectividade. Fazendo uso da junção de um paradigma de *Publish-Subscribe*, *Listeners* que detectam quando um M-OBJ se desconecta, um serviço que monitora constantemente a conexão do M-Hub e um mecanismo de envio de mensagens de *acknowledge* que detecta por *timeout* se o dispositivo tem problemas de conectividade.

Aplicações IoT que consomem um fluxo constante e ininterrupto de dados em ambientes de alta mobilidade e com intensa troca de dados são as que mais se beneficiam desse tipo de mecanismo de detecção.

3
IoTrade: Um Marketplace para IoT

Neste capítulo, nós apresentamos as características e a implementação de uma plataforma que se baseia em conceitos de um *marketplace* para IoT. Nós partimos da premissa de que em alguns anos teremos dispositivos móveis em todos os lugares e que as pessoas terão, na maioria das vezes, um dispositivo inteligente pessoal (e.g. *smartphone* ou *tablet*) que estará sempre conectado à Internet através de redes 4G, 5G ou Wi-Fi. Essa interação cada vez maior entre os usuários IoT e as coisas inteligentes está sendo chamado de paradigma oportunístico de IoT [26] e o qual as considera uma coisa só [27]. Dessa maneira, a popularização de IoT e o aparecimento de novos tipos de dispositivos móveis vão contribuir bastante para o surgimento de diferentes aplicações voltadas para o mercado de IoT. Embora muitos desses dispositivos móveis sejam usados para aplicações locais, ou seja, aplicações onde os dados sensoriados são usados para controlar dispositivos próximos, devemos pensar sobre IoT em um cenário mais holístico, onde os dados de sensores são consumidos por clientes remotos, como pessoas físicas, empresas e repartições públicas ou grandes empresas privadas. Por exemplo, pode acontecer que a secretaria de meio ambiente de uma cidade possa estar interessada em coletar dados de sensores de temperatura e umidade instalados pelas pessoas em seus quintais e jardins. Ou então, uma empresa de construção civil pode estar interessada em dados sobre o fluxo atual de pedestres em diferentes partes de uma cidade para decidir onde irá construir seu próximo empreendimento, informações essas que poderiam ser fornecidas pela densidade de *smartphones* em determinadas áreas ou pelos dados coletados das câmeras de segurança direcionadas para as ruas. Assim, tais dados sobre estados e eventos de lugares, pessoas e coisas no mundo real são um bem valioso que será negociável como produtos e serviços são hoje. Além disso, nos próximos anos podemos, também, sugerir dois outros tipos de serviços de IoT negociáveis: o acesso a atuadores e a utilização de serviços de análise de dados sobre os dados coletados.

Da mesma forma que atualmente na negociação de *commodities*, onde os compradores não interagem diretamente com os produtores, tal comoditização também pode se aplicar aos serviços IoT (dados, atuação e análise). Em vez de se envolver em um contrato de serviço direto, o cliente apenas especificaria alguns atributos de qualidade exigidos sobre o serviço desejado e todos os provedores de serviços que satisfaçam a especificação requerida fariam ofertas de seus serviços ao cliente. Assim, os dados e serviços de IoT, em geral, estariam classificados de acordo com seu nível de precisão, novidade, escopo, confiabilidade, disponibilidade e outros atributos de qualidade. Além disso, os pedidos de serviço seriam segmentados e filtrados de acordo com a localização geográfica dos recursos. Por exemplo, alguns clientes podem estar interessados apenas em dados relacionados a um determinado bairro da cidade, enquanto outro cliente pode exigir dados de presença coletados em um local específico, como por exemplo num hospital.

Esta negociação anônima e global de dados e serviços de IoT como *commodities* é a nossa visão de como IoT pode vir a operar no futuro. Embora tenha alguma semelhança com os *marketplaces* digitais atuais, como AirBnB e Amazon, também terá sua própria maneira de funcionar em termos de estabelecimento de contratos, registro de transações e reputação. A Figura 3.1 representa as principais camadas de um *marketplace* para IoT e os elementos correspondentes (provedores e clientes) no ecossistema IoT. Abaixo, uma categorização resumida destes poderia ser:

Provedor de Objetos Inteligentes: uma empresa ou pessoa que possui alguns sensores / atuadores e que quer alugá-los para aplicações IoT.

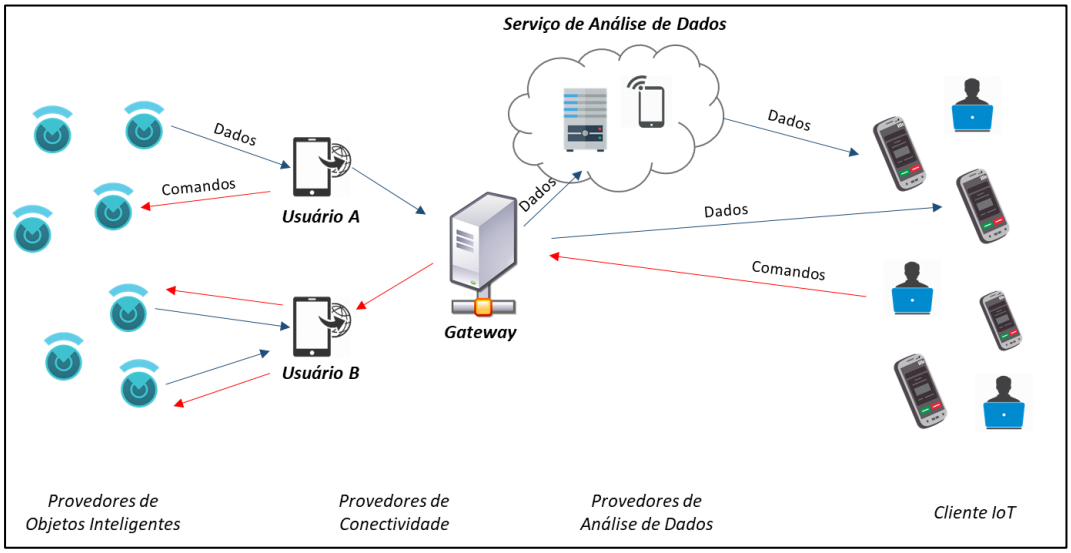


Figura 3.1: O ecossistema IoT com os atores principais

Provedor de Conectividade: uma empresa ou cidadão comum que vende a energia do seu *smartphone* e conexão WWAN para conectar os objetos inteligentes aos serviços da nuvem.

Provedor de Análise de Dados: uma empresa de TI que possui ferramentas de software para analisar (e registrar) o fluxo de dados de sensores, executar métodos de aprendizado de máquina para prever alguma situação, enviar alertas para operadores humanos das aplicações ou ativar ações automáticas pré-programadas para alguns atuadores. Exemplos podem ser o Watson Analytics[4] e o Google Cloud IoT[5].

Cliente IoT: uma empresa, administração pública ou cidadãos comuns que precisam de uma solução IoT completa, composta por (a) um conjunto de dispositivos inteligentes (numa determinada região geográfica - uma zona-alvo), (b) provedores de conectividade que estiverem próximos dos dispositivos inteligentes, e (c) provedores de análise de dados.

3.1. Requisitos e Market Design

Em [2] o autor mostra como um *marketplace* pode gerar valor para o mercado e as vantagens de se ter uma plataforma que centraliza ofertas de serviços e, como tal, pode ajudar a reduzir os custos operacionais, aumentar a segurança e garantir a interoperabilidade entre as soluções oferecidas. No entanto, um *marketplace* para o IoT possui características específicas que o diferenciam das atuais plataformas online. Primeiro, há o fato de se ter pelo menos três tipos de provedores que precisam ser selecionados e "contratados" simultaneamente para que o serviço se inicie. Isso significa que os *marketplaces* de IoT devem garantir um bom equilíbrio de todos esses tipos de provedores e com os clientes. Em segundo lugar, os contratos de serviço provavelmente serão muito mais dinâmicos e efêmeros do que os atuais contratos de plataformas como AirBnB, Alibaba, Amazon ou Uber. Por exemplo, se um cliente desejar adquirir controle sobre um atuador temporariamente isso significa que o "contrato" será válido apenas durante o prazo em que o cliente estiver com o controle, ou seja, durante um tempo limitado. Em terceiro lugar, ao contrário

[4] Watson Analytics - https://www.ibm.com/watson-analytics
5 Google Cloud IoT - https://cloud.google.com/solutions/iot/?hl=pt-br

do aluguel de carros ou de apartamentos ou casas de veraneio, as futuras aplicações IoT terão múltiplos propósitos, isto é, os clientes poderão fazer uma grande variedade de solicitações de acesso a sensores / atuadores e de análise de dados, e a multiplicidade de usos dentro de IoT será sempre crescente. Isso requer uma descrição clara e confiável do escopo, além da finalidade dessas aplicações de IoT e dos principais beneficiários econômicos e sociais gerados.

Como um *marketplace* para IoT possui vários componentes, sendo que alguns deles móveis, ele precisa ter capacidade de processamento, memória suficiente e protocolos de comunicação eficazes para a troca de informação para que não haja qualquer perda de informação entre os componentes móveis e os não-móveis. Além disso, ele também é responsável por desempenhar o papel de intermediário capaz de selecionar uma combinação entre os provedores de serviço (conectividade e análise de dados) e objetos inteligentes. Embora seja difícil prever como os *marketplaces* de IoT irão funcionar estimamos os seguintes requisitos:

Registro e atualização dos dados de provedores e objetos inteligentes em tempo real para manter o rastreamento de quais deles estão em uso e quais são os seus parâmetros de qualidade (e.g. quantidade de bateria e sinal de rede).

Enviar e Receber comandos/dados para os M-OBJs. Essa transmissão/recepção de informação inclui comandos e dados de atuadores e dados de sensores.

Converter dados de sensores que vem dos dispositivos para um formato que o cliente possa ler. Para isso é usado um driver específico de cada dispositivo que deverá ser implementado.

Solicitações de localização: os provedores receberão solicitações apenas para os locais que são de interesse do cliente, ou seja, enquanto o cliente não fizer uma solicitação de serviço os provedores não podem se anunciar para eles.

Incentivos para atrair clientes e provedores: para atrair e manter clientes e provedores na plataforma são necessários alguns incentivos. Para um *marketplace* IoT, além dos incentivos monetários e descontos fiscais, troca de informações coletadas por cupons de desconto e programas de fidelidade são algumas ferramentas atraentes para um público que não considera apenas compensação financeira.

Penalidades para provedores mal avaliados: para manter a qualidade dos serviços prestados, os prestadores mal avaliados serão penalizados e sua categoria de *commodity* será rebaixada temporariamente, mas se for um atuador e ocorrer uma desconexão do serviço haverá uma penalidade ainda mais severa (diminuição

de reputação e sua categoria de preço reduzida). A ideia de aplicar penalidades é precisamente obrigar os provedores a fornecer sempre o que eles têm de melhor e fazê-los prestar atenção ao nível de seus serviços.

Preços dinâmicos: para incentivar os provedores de conectividade, que fornecem sua conexão de dados, a manter um serviço melhor, sua categoria de *commodity* muda dinamicamente de acordo com alguns parâmetros, como seu sinal de internet (3G, 4G, Wi-Fi, etc.) e seu nível de bateria. Quanto maior esses parâmetros, o preço pago para eles pelo seu serviço será maior e com isso os provedores terão um incentivo para manter os parâmetros de rede e bateria os melhores possíveis e, como resultado, o cliente irá ter uma melhor experiência de uso.

Comoditização: Outro requisito importante do *marketplace* é que os serviços de conectividade, dados, atuação e de análise de dados estarão em constante verificação e classificação. Onde cada categoria terá seus parâmetros de qualidade e cada dado sensoriados ou provedor que estiver dentro desses parâmetros (e.g. precisão, latência, nível de novidade do dado, etc.).

Além desses requisitos orientados para negócios, temos alguns requisitos técnicos necessários para garantir o funcionamento desta plataforma multilateral:

Multiaplicação: capaz de executar para mais de um cliente por vez. O aplicativo cliente, o provedor de conectividade / análise de dados e o servidor devem se comunicar entre si.

Escalável: escalável para funcionar com bilhões de sensores, milhões de provedores de acesso e milhares de provedores de análise de dados.

Tempo real: a descoberta de novos fornecedores e a execução dos algoritmos devem ser quase instantâneos. Ele deve executar essas tarefas em menos de 500ms.

Tolerância a falhas: Capaz de se recuperar a partir de uma perda de conectividade durante um serviço e, rapidamente, executar o algoritmo de *matchmaking* (menos de 500ms) para restabelecer a conectividade com outros provedores que estão habilitados para continuar o serviço e tudo sem que o cliente perceba que aconteceu algum problema. Se não for possível encontrar uma correspondência, uma mensagem de fim de serviço aparece para o cliente.

Confiável e altamente disponível: esteja em operação 24 horas por dia e 7 dias por semana.

3.2. Arquitetura

Propomos aqui uma arquitetura nos moldes de um *marketplace* para IoT, que chamamos de IoTrade, com base nos requisitos apresentados na Seção 3.1. O IoTrade sendo baseado em conceitos de plataformas multilaterais tem o seu serviço principal centralizado. Na Figura 3.2 são mostradas as interações entre as partes interessadas (o cliente da aplicação IoT, os donos de M-OBJs, os provedores de conectividade e de análise de dados) e os principais serviços do IoTrade. O serviço principal do IoTrade é dividido nos componentes abaixo:

- **Comoditização** - Provedores e M-OBJs serão considerados *commodities* digitais, onde seu preço depende de seu QoS e da precisão das categorias de informações, que variará de acordo com esses parâmetros.
- **Monitoramento de QoS** - A plataforma monitorará constantemente os parâmetros QoS dos M-OBJs e provedores para garantir que eles estarão sempre fornecendo o nível de acurácia e QoS de acordo com a categoria a que pertencem.
- **Matchmaking** - É o componente que faz o trabalho que leva em conta todos os parâmetros de QoS e a reputação de provedores, e compara com as demandas dos clientes para encontrar a melhor alternativa (em custo e benefício) para as requisições de clientes IoT.

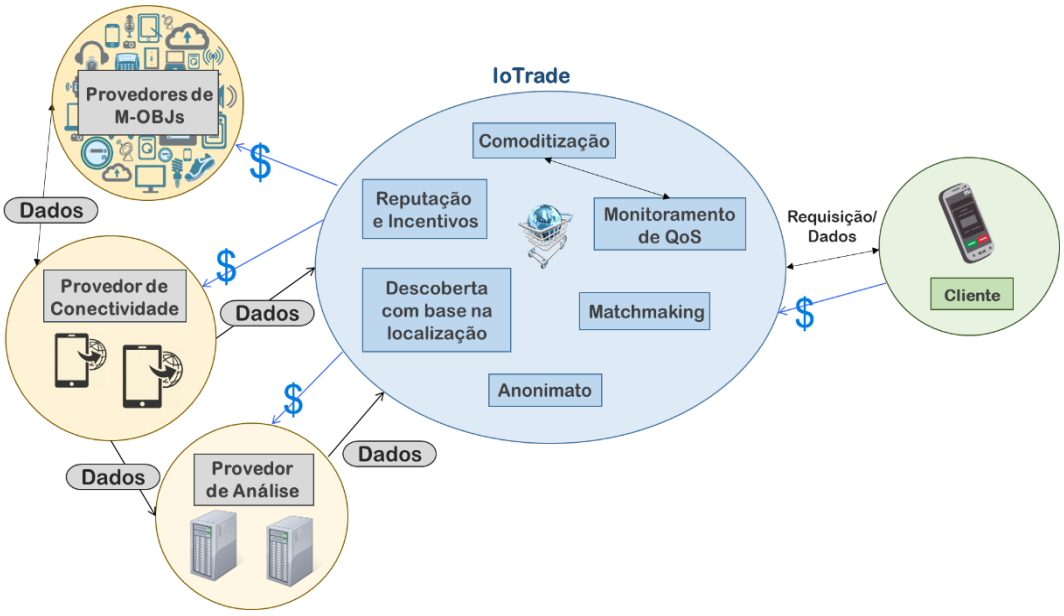


Figura 3.2: Arquitetura do IoTrade

- **Anonimato** - Todos os dados pessoais dos provedores e proprietários de M-OBJs são ocultos e não são usados no cálculo do algoritmo.
- **Descoberta com base na localização** - A plataforma leva em consideração a localização (input do cliente) e seus gostos para mostrar-lhe, por ordem de importância, os serviços e categorias de dados mais relevantes.
- **Reputação e Incentivos** - A aplicação tem mecanismos de incentivo e reputação para manter o interesse dos clientes atuais em continuar a usar a plataforma.

Os componentes propostos acima para a arquitetura do IoTrade funcionam em conjunto para criar o *marketplace*. Destes seis componentes principais iremos aprofundar a discussão sobre a necessidade, benefícios e limitações dos que mais se destacam, que são eles: Comoditização, Reputação de Incentivos e Monitoramento de QoS.

A comoditização de dados, ao contrário das *commodities* atuais (grãos, petróleo, minério de ferro, etc.), exigirá menos intervenção humana. Em vez disso, os agentes de software verificarão constantemente os sensores e atuadores para saber se eles estão funcionando, se eles estão fornecendo as informações precisas e se a análise de dados e os algoritmos são precisos, confiáveis e rápidos (isto pode ser verificado em relação aos dados reais coletados pelos sensores). Por exemplo, a qualidade de um serviço de análise de dados que estima a extensão de um engarrafamento pode ser verificada analisando se os carros "fluem bem" nas imagens das câmeras de trânsito. A limitação dessa abordagem é que serão muitos parâmetros de qualidade que estarão gerenciando essa classificação em *commodities* de dados, logo é preciso ter um controle bem estrito e softwares de apoio para não ter erros nas classificações das categorias dos dados.

Os mecanismos de reputação e incentivos são necessários em plataformas multilaterais [18] pois é preciso engajar o público e fazer com que ele continue voltando para comprar na plataforma, seja ela de IoT, varejo ou qualquer outro tipo. O benefício desse tipo de mecanismo é aumentar o fluxo de pessoas comprando na plataforma e uma forma de manter os donos de M-OBJs e provedores de conectividade e análise de dados motivados para oferecer um bom padrão de seus serviços. Por exemplo, mecanismos que penalizam um dono de sensor que está com a sua precisão abaixo do que ele especificou o motiva para melhorar ou consertar o seu dispositivo para voltar a ganhar mais dinheiro e, por consequência, lucrar mais.

Uma limitação, como no serviço de comoditização de dados, é a quantidade de parâmetros para serem gerenciados na reputação dos provedores que pode tornar confuso a transparência da avaliação para os provedores, por isso é preciso ter um controle bem estrito ao realizar a medição da reputação de cada um.

O monitoramento de QoS é necessário pois é neste módulo que é feita a verificação de todos os parâmetros de qualidade (e.g. nível de precisão do dado, relevância, latência, escopo, confiança e disponibilidade) para determinar se o provedor de serviço está cumprindo com a especificação. Além disso, este módulo está ligado diretamente com o módulo de comoditização de dados para fornecer a classificação de cada provedor e categorizá-lo de acordo com ela. O benefício deste serviço é a garantia de que os provedores estão classificados corretamente em sua categoria de dados e assegurar para o cliente que ele está pagando por um dado validado e certificado. A limitação é que devido a escala de bilhões de dispositivos móveis será necessário ter poder computacional para realizar a monitoração do QoS em tempo real sem que afete o desempenho da plataforma.

Abaixo, apresentamos as interações, mostradas na Figura 3.2, entre os provedores de serviço, clientes e o serviço principal do *marketplace* IoTrade:

M-OBJs: Enviam os dados sensoriados e recebem comandos de atuação dos provedores de conectividade.

Provedor de Conectividade: Recebem e enviam dados para os M-OBJs, enviam dados para o provedor de análise informações para serem processadas e encaminhadas ao cliente. Transmite para o serviço IoTrade seus dados de QoS e localização.

Provedor de Análise de Dados: Recebe dados do provedor de conectividade, os analisa e os envia para o cliente. Transmite para o serviço IoTrade seus dados de QoS e localização.

Cliente IoT: Requisições de compra de serviços para o serviço IoTrade.

3.3. Implementação

Levando em consideração a arquitetura apresentada na Seção 3.2, criamos e desenvolvemos um protótipo do IoTrade com parte dos requisitos que foram propostos na Seção 3.1 e que ao longo desta seção iremos discutir os aspectos que de fato foram implementados. Ele é responsável pela seleção automática, verificação

de qualidade e classificação de dados de sensores de objetos inteligentes, atuadores e serviços de análise de dados, bem como o *matchmaking* entre donos de M-OBJs e provedores de conectividade e de análise de dados, de um lado, e clientes do IoTrade, com suas demandas específicas, do outro. Essas demandas de IoT são codificadas em solicitações de clientes para dados e serviços IoT (*Requests for IoT Data and Services, RIDS*), que contêm um conjunto de atributos e "zonas alvo". Por exemplo, uma pessoa quer conhecer a temperatura em uma cidade para a qual ela deseja viajar, então ela marca esse lugar como seu alvo no IoTrade colocando que quer medir a temperatura com alta precisão; feito isso ela "anuncia" no sistema para que os provedores possam fazer ofertas. O lance vencedor será aquele que o *matchmaking* verificar que se encaixa melhor nos parâmetros que o cliente colocou. Esses atributos e a zona alvo determinam que uma *RIDS* é entregue apenas a provedores que tenham sensores / atuadores dentro da zona geográfica de destino que foi especificada e cuja precisão de dados do sensor, qualidade do atuador ou análise de dados corresponda aos níveis especificados nos atributos da *RIDS*.

O nosso protótipo do IoTrade atualmente está dividido nos seguintes núcleos: o servidor principal, o aplicativo cliente e os provedores de conectividade e de análise de dados. Para o desenvolvimento de cada parte foi escolhida a tecnologia que melhor se adaptada aos componentes, mas que podem ser estendidos para trabalhar com outras tecnologias. Por exemplo, no momento existe apenas a plataforma móvel para fazer os requisitos dos clientes ao servidor, mas poderíamos adicionar uma plataforma web com as mesmas características, já que estamos usando uma REST API para enviar as requisições ao servidor principal, ou seja, temos uma estrutura modularizada para adicionar novas plataformas ao sistema.

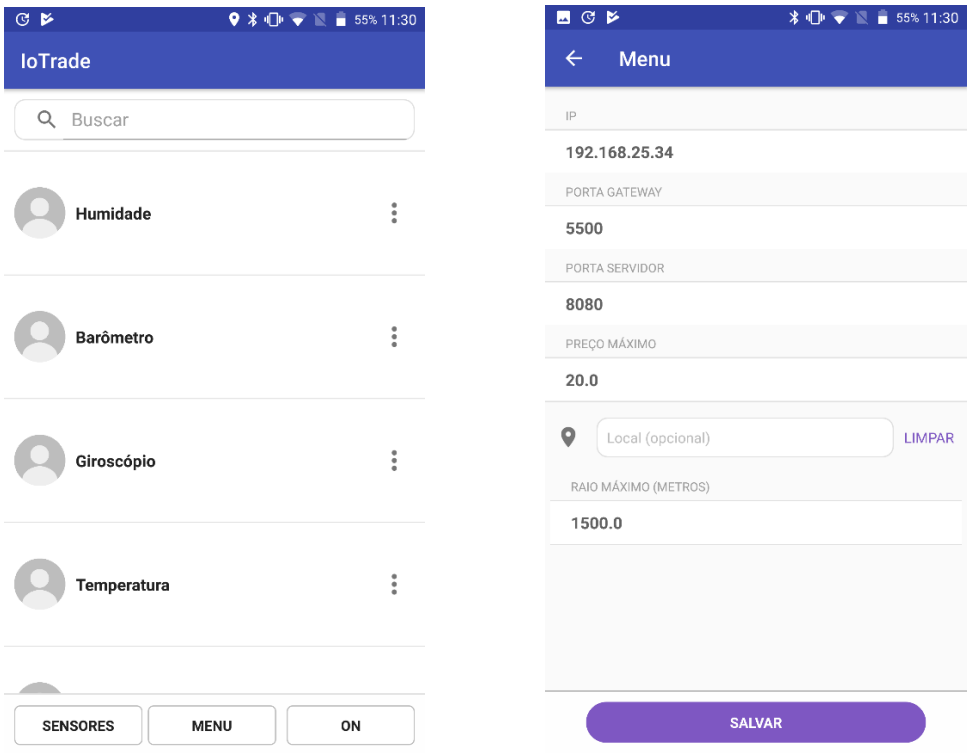
O servidor principal foi desenvolvido em Node.js[6] e é o responsável por receber todas as requisições dos clientes e redirecioná-las para o serviço correspondente, seja ele executar *matchmaking* ou atualizar a classificação dos provedores. Nesse protótipo nós não implementamos o monitoramento constante do QoS devido à complexidade de parâmetros que tem de ser considerados no QoS, apenas desenvolvemos uma rotina que suspende os provedores que tiverem uma nota abaixo de 3.0 na prestação dos serviços. Além disso mecanismos de incentivo, também, não foram implementados neste protótipo devido à complexidade e escala dos testes

[6] Node.js - https://nodejs.org/en/

que teriam de ser realizados para validar quais tipos de mecanismos seriam realmente eficientes num *marketplace* de IoT. Os requisitos “Preços dinâmicos” e “Comoditização”, mostrados na Seção 3.1, foram simplificados para que os testes no protótipo implementado pudessem ser feitos. Além disso, este protótipo do IoTrade não implementa qualquer transferência de valores monetários.

Mesmo assim, nosso protótipo já implementa cinco dos seis serviços (vide Seção 3.2) propostos para um *marketplace* de IoT. Ele é capaz de realizar a seleção de uma combinação de M-OBJs com provedores de serviços (*matchmaking*), realiza atuação, recebe dados e os mostra para o cliente, faz a descoberta dos dispositivos com base nos parâmetros passados pelo cliente, garante o anonimato dos provedores, faz a comoditização dos dados e tem mecanismos de reputação e incentivo.

A Figura 3.3 mostra duas telas do aplicativo cliente, desenvolvido em Android, e é onde os clientes selecionam os parâmetros de localização e preço que irão fazer o filtro das categorias de dados. Além disso, o aplicativo recebe os dados dos sensores e pode mandar comandos para atuadores. O provedor de conectividade foi



(a) Categorias de dados em alguma localização

(b) Menu Principal

Figura 3.3: Telas do IoTrade

implementado como uma extensão do M-Hub, eles se conectam aos objetos inteligentes por meio da tecnologia *Bluetooth Low Energy* (BLE) e transmitem os dados para SDDL Core, que faz o seu roteamento até chegar ao cliente no seu *smartphone*.

O provedor de análise de dados tem duas variantes implementadas (móvel e estação de trabalho fixa). A móvel, apesar de ter menos poder de processamento, já que utiliza o poder computacional do *smartphone,* tem a vantagem de ser portátil e consideravelmente mais barata que a variante fixa. Ela usa como base o MEPA Service do M-Hub, que utiliza o *Complex Event Processing* (CEP) para fazer o processamento dos dados e a transmissão dos dados processados ao cliente é análoga à maneira que o M-Hub faz. A variante fixa do provedor é ligeiramente mais simples por não implementar a parte que utiliza o CEP, mas nele existem funções que fazem algum tipo de operações matemáticas com os dados e que posteriormente são enviadas para o cliente, da mesma forma que é feita nos provedores móveis.

Um protocolo de *handshake* é utilizado quando um cliente seleciona uma categoria de dados para "compra". Ao selecionar a categoria desejada, o protocolo envia uma mensagem para os potenciais provedores selecionados e espera a resposta do *acknowledge*. Assim ele garante que os provedores escolhidos estão online no momento e funcionais, mas caso ocorra algum problema com eles *matchmaking* é executado novamente para selecionar uma nova combinação.

3.3.1. API REST do IoTrade

A API Retrofit do IoTrade foi desenvolvida para ser uma interface genérica, baseada no conceito de *Representational State Transfer* (REST)[7], para o envio/recebimento de dados entre o servidor e os atores da plataforma. A transmissão de dados/comandos entre M-OBJs não foi alterada e ela é contínua usando o *middleware* SDDL, pois essa estrutura já estava implementada no M-Hub e foi provada eficiente [5]. Já para realizar todas os serviços que o IoTrade propõe e que foram citados na Seção 3.2, foi necessário desenvolver essa API para que os provedores realizassem as solicitações ao servidor utilizando o paradigma *Request-Response*.

Essa API é portável para qualquer tipo de dispositivo**.** Hoje a temos implementada no M-Hub e no aplicativo IoTrade, desenvolvidos em Andoid, e a temos

[7] REST: http://www.restapitutorial.com/

no servidor, feito em Node.js. As principais funções e operações que podem ser feitas com ela estão mostradas na Figura 3.4, mas com o passar do tempo novos serviço podem ser implementados no sistema e ela pode ser estendida para acomodá-los. Algumas funções que precisam ser explicadas são as seguintes: **getSensorPrice** e **getConnectPrice**, pegam as informações de preço em tempo real para atualizar o contador de preço ao longo da prestação de um serviço, **getServices** é responsável por retornar as categorias de dados ou atuadores disponíveis na região que o cliente selecionou e que queria observar/atuar. Já as seguintes funções **getNewAnalytics**, **getSensorAlgorithm**, **getNewConnectionActuator** e **getSensorAlgorithmAnalytics** são responsáveis por chamarem o matchmaking e selecionar os **provedores**. Note que cada uma é responsável por um tipo de provedor, por exemplo, selecionar somente atuadores ou serviços com/sem a análise de dados. Por último, temos **setLocationMobileHub** e **setAnalyticsMobileHub** que atualizam em tempo real, no banco de dados, os estados dos provedores de conexão e o de análise de dados.


```
@POST("login_user/")login(User)
@POST("register_location")setLocationMobileHub(User)
@POST("register_analytics")setAnalyticsMobileHub(User)
@POST("remove_sensor_mobileHub")removeSensorMobileHub(Sensor)
@POST("convert_sensor_data")convertSensorData(Sensor)
@POST("get_sensor_registered")getSensorRegistered(Sensor)
@POST("get_sensor_matchmaking/")getSensorAlgorithm(ObjectServer)
@POST("get_actuator_matchmaking/")getConnectionActuator(ObjectServer)
@POST("get_matchmaking_analytics/")getAlgorithmAnalytics(ObjectServer)
@POST("get_new_analytics/")getNewAnalytics(ObjectServer)
@GET("get_sensor_price_information/")getSensorPrice()
@POST("get_connect_price_information/")getConnectPrice(ConnectPrice)
@GET("get_user/")getUser()
@POST("get_services_information/")getServices(ServiceIoT)
@POST("set_sensor_parameters")setSensorParameters(Sensor)
@POST("set_actuator_state/")setActuatorState(SensorPrice)
@POST("update_user_budget/")updateUserBudget(SensorPrice)
@POST("update_sensor_rating/")updateSensorRating(Response)
@POST("update_sensor_info/")updateSensorInformation(SensorPrice)
```

Figura 3.4: API REST do IoTrade

3.3.2. Algoritmo de Matchmaking

Esta seção descreve o desenvolvimento de um esqueleto do algoritmo de *matchmaking*. Esta base do algoritmo visa selecionar uma combinação de provedores (M-OBJs, conectividade e análise de dados) que melhor se adapte aos requisitos passados pelo cliente. Esses requisitos incluem o valor que o cliente está disposto a pagar, o QoS mínimo exigido e a localização inserida pelo cliente. O algoritmo está dividido em duas partes, a primeira é uma pesquisa de banco de dados, com o objetivo de filtrar os serviços e M-OBJs por categoria, e a segunda escolhe o melhor candidato a partir dos dados retornados pela primeira parte. O pseudocódigo do algoritmo, mostrado na Figura 3.5, apresenta os parâmetros de input do cliente e como as duas partes do algoritmo estão estruturadas. Para a primeira parte é levado em consideração se o cliente quiser incluir o serviço de análise de dados (opcional) ou se ele apenas deseja receber os dados brutos e, portanto, apenas escolhe um provedor de conectividade para enviar dados e M-OBJs para coletá-los.

```
1   Function matchmaking(latitude, longitude, category, radius, ha-
    sAnalytics, ID)

2   //Pre-Processing BEGIN
3   con <- getBestConnectionProvider(latitude, longitude, category, ra-
    dius, ID)

4   if hasAnalytics is TRUE then
5        analytics <- getbestAnalyticsProvider(con, category, ID)
6   end if

7   sens <- getbestSensorProvider(con, category, ID)
8   parcial <- getCombinationArray(con, analytics, sens, category)
9   //Pre-Processing END

10  if hasAnalytics is TRUE then
11       RETURN getRandomBestFitAnalytics(parcial) - {sensor, connection,
    analytics}
12  else
13       RETURN getRandomBestFit(parcial) - {sensor, connection}
14  end if
```

Figura 3.5: Pseudocódigo do Algoritmo de *Matchmaking*

Antes de executar o algoritmo um pré-processamento (entre as linhas 3 e 8 da Figura 3.5) é feito com o objetivo de selecionar os melhores candidatos entre provedores de serviços e M-OBJs para acelerar o processamento e a escolha, já que estamos em um cenário de tempo real que exige uma performance otimizada. Imagine o cenário onde existem bilhões de M- OBJs, milhões de provedores de conectividade e milhares de provedores de análise de dados, seria impraticável ter um

algoritmo escalável que funcionasse em tempo real para combiná-los, já que o número de combinações seria o produto entre eles todos. Com base nisso, o pré-processamento é feito para reduzir a escolha aos melhores provedores ranqueados em cada categoria de preço diferente, assim é possível executar o algoritmo em tempos muito inferiores aos sem o pré-processamento, como mostrado na Tabela 3.1.
Os requisitos básicos definidos ao implementar este algoritmo para a escolha de todos os provedores no pré-processamento devem ser: não estar suspenso dentro do sistema devido a queixas de clientes ou mau atendimento e ter uma boa reputação. No caso de um M-OBJ ser um atuador, ele não deve estar em uso para participar do algoritmo, pois o acesso a ele é sempre exclusivo de uma pessoa por vez. Para os provedores de conectividade os requisitos são: ter o sinal de rede maior que 3, em uma escala de até 10 (esta escala é definida a partir de uma função de conversão nativa da plataforma Android), tenha o sinal da bateria maior que 30, em uma escala de até 100, e estar dentro do raio e local estabelecidos pelo input do cliente. Para o provedor de análise de dados, os requisitos de sinal da bateria e da rede são semelhantes ao provedor de acesso.

No final desta fase todos os provedores de serviços escolhidos são agrupados e uma ordenação é estabelecida entre eles para ser escolhido, posteriormente, o melhor conjunto de provedores. O objetivo da ordenação é colocar em primeiro o me-

Iteração	**Com Pré-processamento (s)**	**Com Pré- processamento (s)**
1	0,2240	3,1830
2	0,2180	2,9840
3	0,2340	3,0560
4	0,2040	2,8530
5	0,2540	2,9720
6	0,1980	2,7730
7	0,2420	3,2470
8	0,2310	2,9910
9	0,1990	3,0600
10	0,2250	3,5530
Média	**0,2229**	**3,0672**

Tabela 3.1: Tempo de execução do Algoritmo de *Matchmaking*

lhor candidato para o conjunto de M-OBJs e provedores e, com isso, automaticamente o melhor conjunto será escolhido. A ordenação é feita em três blocos: primeiro o provedor de conectividade é ordenado, depois os sensores conectados a ele e, opcionalmente, por último o serviço de análise de dados. O provedor de conectividade é ordenado primeiro porque sem ele não é possível escolher sensores, uma vez que eles estão conectados via BLE a ele.

Os critérios de ordenação dos parâmetros dos M-OBJs e provedores no algoritmo foram os seguintes, desde o mais importante até o menos importante:

- Reputação, nota média dada ao provedor pelos clientes do seu serviço
- O sinal da rede do *smartphone*
- A quantidade de bateria
- Preço
- Para os M-OBJs e provedores de análise de dados, apenas a classificação e o preço foram levados em consideração.

Após o algoritmo devolver os melhores candidatos, a escolha do conjunto de provedores que serão selecionados é feita aleatoriamente para que exista uma distribuição justa entre todos os candidatos, ou seja, caso existam dados da mesma categoria de preço isto fará com que todos os fornecedores tenham chances iguais entre si. No caso de algum provedor mudar seus parâmetros de QoS isso é atualizado automaticamente no banco de dados e é levado em consideração em tempo real pelo algoritmo.

A linguagem *Cypher*[8], para pesquisa em bancos de dados de grafos Neo4j[9], possui uma semântica semelhante à linguagem SQL, mas nela é possível incluir alguns fragmentos de código e é possível executar um pequeno algoritmo dentro, mostrado no exemplo da Figura 3.6. A análise da complexidade do algoritmo na primeira parte, escrita em *Cypher*, é bem conhecida e documentada como é mostrado em [32]. Na segunda parte, escrita em *JavaScript*[10], o acesso é direto para a estrutura de dados onde os resultados estão armazenados, portanto a complexidade dessa segunda parte é O (1).

[8] Cypher - https://neo4j.com/developer/cypher-query-language/
[9] Neo4j - https://neo4j.com/
[10] Javascript - https://www.javascript.com/

```
CREATE (ber:City { lat: 52.5, lon: 13.4 }),(sm:City { lat: 37.5, lon: -122.3 })

RETURN 2 * 6371 * asin(sqrt(haversin(radians(sm.lat - ber.lat))+ cos(radians(sm.lat))*
cos(radians(ber.lat))* haversin(radians(sm.lon - ber.lon)))) AS dist
```

Figura 3.6: Exemplo da linguagem Cypher

3.3.3. Arquitetura de Comunicação

Na Figura 3.7 é mostrado como está organizada a arquitetura de Comunicação do IoTrade exibindo como cada componente está interagindo na troca de dados com o outro. O servidor do IoTrade e o SDDL *Gateway* são os dois componentes principais responsáveis pela comunicação entre os provedores e os clientes.

O servidor utiliza o padrão *Request-Response* [17] para uma comunicação segura e confiável através de uma API REST, implementada dentro dele, recebendo as solicitações de *matchmaking* dos clientes (entre outras citadas na Seção 3.2) e atualizando no banco de dados as informações recebidas em tempo real dos provedores de serviço. A comunicação entre o servidor e o banco de dados é feita através de uma API fornecida pelo *Neo4j*.

O *gateway* usa o padrão *Publish-Subscribe* que permite criar uma comunicação flexível, dinâmica e assíncrona através de um barramento de serviço permitindo gerar um fluxo de dados entre os M-OBJs, provedores e clientes e, também, para enviar e receber as informações de detecção de problemas de conectividade. Esse

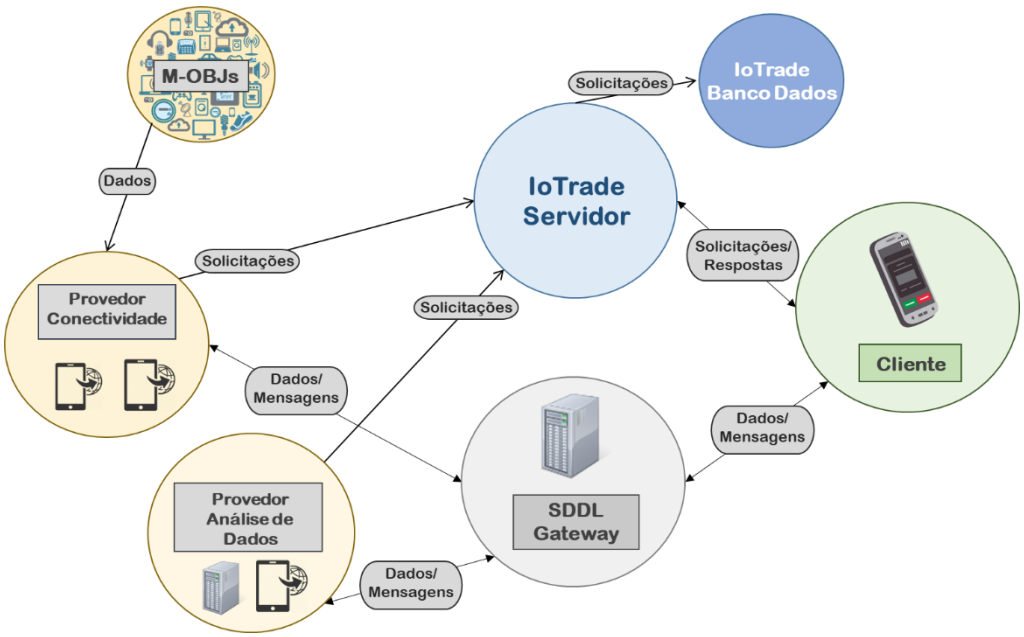


Figura 3.7: Arquitetura de Comunicação do IoTrade

paradigma é utilizado pelo *gateway* em substituição a um padrão síncrono (e.g. *Request-Response*) pois com a grande quantidade de fluxo de dados gerados em tempo real pelos M-OBJs parte das informações seriam perdidas.

Os provedores de conectividade se comunicam com os M-OBJs através da tecnologia BLE e, posteriormente, enviam os dados através do SDDL *Gateway*, podendo ser diretamente para o cliente ou, primeiro, para o provedor de análise de dados que irá processar as informações antes de enviá-las para o cliente através desse mesmo *gateway*.

3.3.4. Mapeamento dos Provedores no Banco de Dados

A padronização e a interoperabilidade são absolutas necessidades para que os objetos inteligentes e provedores possam comunicar entre si [33]. Hoje em dia os fabricantes em sua maioria não buscam uma padronização e focam em soluções proprietárias que eles mesmo desenvolvem, como por exemplo o Tizen[11] que é um sistema operacional exclusivo da Samsung para Smart TVs, geladeiras e *smartwatches*.

Neste este trabalho foi desenvolvida uma camada de interoperabilidade para que os M-OBJs conseguissem se comunicar com os provedores de conectividade. Como mostrado na Figura 3.8, cada sensor/atuador embutido em um M-OBJ é mapeado numa relação no banco de dados com as categorias de dados que ele está ligado, utilizando suas UUIDs. Note que no caso dos M-OBJs da figura, eles são conectados ao M-Hub utilizando a tecnologia BLE que usa UUIDs para identificar os serviços de cada objeto, mas se for utilizada outra tecnologia o mapeamento será de acordo com ela. A interface *Technology* dentro do M-Hub é que irá traduzir esse mapeamento de acordo com o protocolo de conexão que cada tecnologia WPAN usa e, com isso, garantir a interoperabilidade e padronização dos M-OBJs com o IoTrade.

[11] Tizen Os - http://developer.samsung.com/tv/develop/specifications/general-specifications

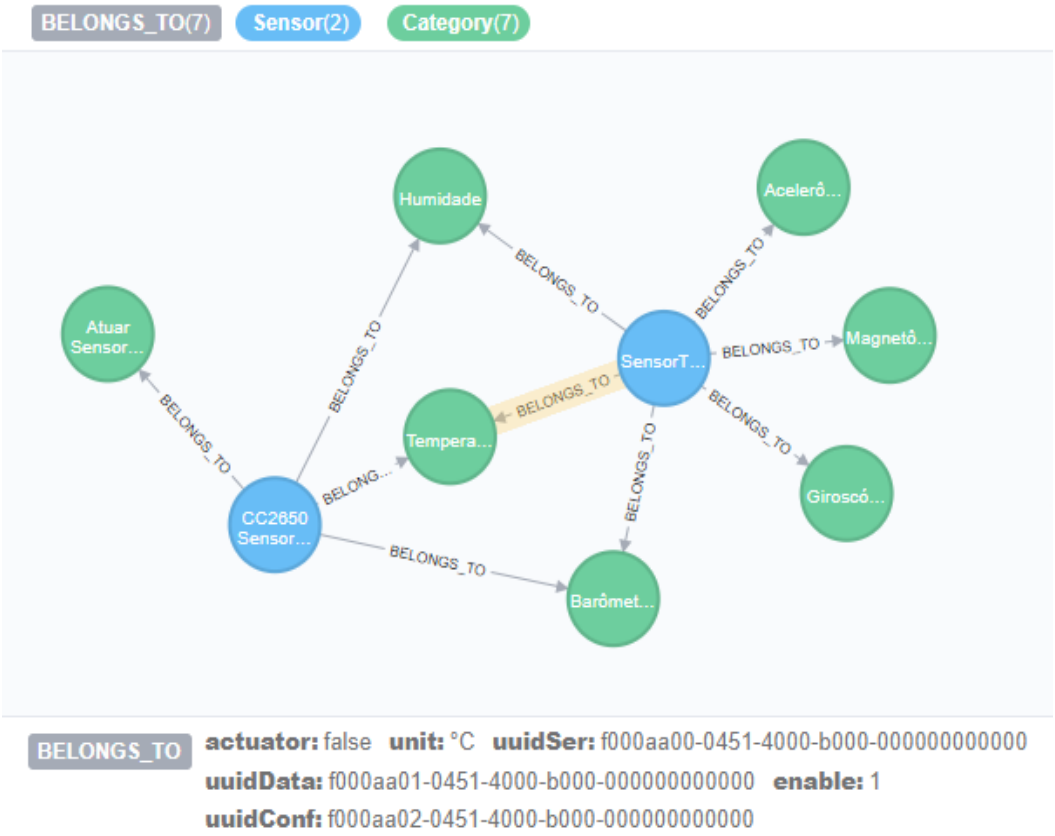


Figura 3.8: Exemplo de mapeamento dos provedores no BD Neo4j

Os provedores de análise de dados também utilizam um mecanismo semelhante, onde em cada relação ligada à categoria de dados ficam as informações de cada serviço de análise que ele presta (vide Seção 5.2.2). Além disso, mapeando os serviços dessa forma, toda vez que forem acrescentados novos, os clientes interessados poderão contratá-los em tempo real.

4
Detecção de Problemas de Conectividade

A detecção da qualidade de conexão presente no IoTrade foi projetada pensando na mobilidade irrestrita existente quando se trata de aplicações IoT, pois os dispositivos e provedores de serviços podem estar sempre em movimento e como não existem tecnologias que garantam a continuidade da conexão de dados por todo tempo em que está sendo prestado o serviço, foi necessário criar um mecanismo que ficasse monitorando constantemente a conectividade dos M-OBJs com o M-Hub e do M-Hub com a internet.

O protocolo, que será apresentado ao longo dessa seção, realiza o que chamamos de garantia de serviço contínuo, executando em conjunto da detecção de problemas de conectividade um *matchmaking* para encontrar novos provedores e, assim, certificando-se da continuidade do serviço que estava sendo prestado. Existem três casos que precisam de atenção ao acontecer uma desconexão. São eles: um dispositivo perder conexão com o M-Hub em que ele estava conectado, um M-Hub decidir parar de fornecer sua conexão de dados para a plataforma e quando o M-Hub perde seu sinal de rede (Wi-Fi ou rede celular).

No IoTrade os provedores de conectividade são móveis e implementados com base no M-Hub, portanto implementam obrigatoriamente este mecanismo de detecção. Além disso, existe uma classe de provedores de análise de dados que também são baseadas no M-Hub e por consequência são móveis. Assim, esta classe, da mesma forma que os provedores de conectividade, implementam este mesmo mecanismo.

Ao longo desse capítulo descrevemos em detalhes como são implementados cada um dos três casos e como está estruturado esse mecanismo de detecção contínua de problemas de conectividade no IoTrade. Na Figura 4.1 é mostrada a arquitetura do mecanismo dentro do M-Hub, apresentando em detalhes cada um destes casos e o caminho que é feito desde a detecção até a comunicação do problema ao IoTrade através do SDDL *Gateway*.

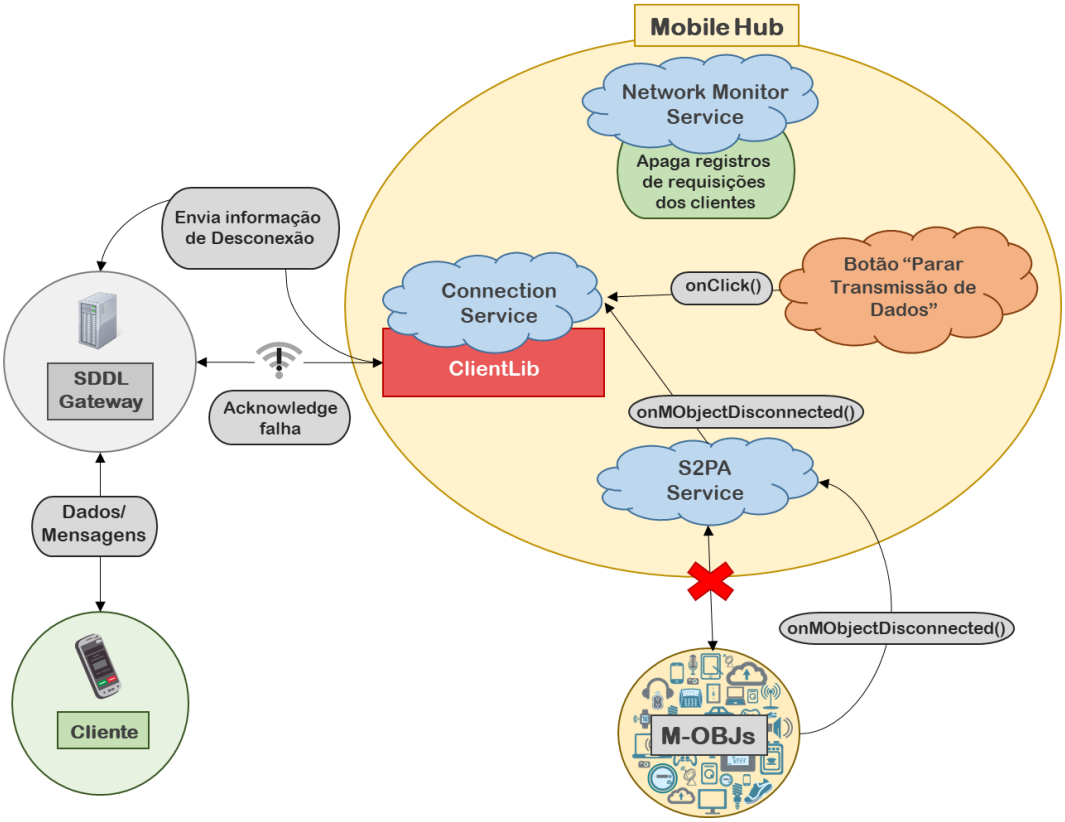


Figura 4.1: Tipos de problemas de conectividade que podem ocorrer

4.1. Objeto Inteligente se Desconecta

Quando um objeto inteligente se desconecta do dispositivo *master* em que ele está ligado existe um *Listner* no M-Hub que é chamado automaticamente e avisa ao servidor do IoTrade e ao aplicativo cliente que o objeto foi desconectado, mas não mostra imediatamente a informação na tela do *smartphone* do cliente pois o IoTrade ainda tenta achar um novo match, este com as mesmas características e parâmetros para continuar com a mesma qualidade o serviço que estava sendo prestado.

O *Listner* que é chamado pelo M-Hub é o *onMObjectDisconnected()* e ele está dentro da classe *TechnologyListener*, que é uma interface que deve ser implementada dentro do *service S2PA* do M-Hub. É nele que se recebe a informação da desconexão dos protocolos de comunicação (e.g. *Bluetooth* clássico, BLE ou Zigbee) que estão codificados dentro do M-Hub. Após o recebimento da desconexão, esse *Listner* envia pelo SDDL *Gateway*, através do *service Connection*, a informação de que existem problemas de conectividade com tal M-OBJ e, com isso, o aplicativo cliente do IoTrade começa a procurar novos provedores para substituir os antigos. Além disso, o servidor atualiza a informação de que tal objeto inteligente

não está mais ligado ao provedor de conectividade em questão. Por exemplo, um cliente compra os dados da categoria "Temperatura" e começa a recebê-los, mas o sensor que está fornecendo os dados se desconecta por algum problema e, com isso, o provedor de conectividade recebe a informação dessa desconexão e avisa tanto o servidor quanto o aplicativo cliente do IoTrade para realizar um novo match de provedores. Assim, caso o algoritmo de *matchmaking* não encontre uma combinação para continuar o serviço, o cliente recebe uma mensagem do término do fluxo de dados. Caso a resposta do algoritmo for positiva, o cliente não é informado de nada e o serviço continua como se não tivesse acontecido problema algum.

4.2. Provedor de Serviço Desliga sua Conexão de Dados

Este segundo caso guarda algumas semelhanças ao primeiro em relação a existir um *Listner* que avisa, pelo SDDL *Gateway*, o servidor e o aplicativo cliente que existem problemas de conectividade no M-Hub. O *Listner* em questão é o *onClick()* que é "disparado" quando o usuário realiza o *logout* no M-Hub (botão de stop) e, com isso, o provedor não oferece mais sua conexão de dados para fornecer um fluxo de dados aos clientes compradores.

O que difere do caso da Seção 4.1 é que, além de avisar o IoTrade para realizar um novo match, é preciso notificar o sistema que agora esse M-Hub está desligado e não pode participar da nova escolha de provedores. Neste caso podem existir provedores de conectividade e de análise de dados (implementado com base num M-Hub) que utilizam esse mecanismo, mas a maneira de executar a detecção e comunicação com o IoTrade em ambos é igual pois foram desenvolvidos a partir da mesma base. Por exemplo, um cliente contrata uma solução no IoTrade que envolva uma análise de dados de "Umidade". Para isso são selecionados provedores de conectividade conectados a sensores de "Umidade" e provedores que irão analisar tais dados, mas durante a prestação do serviço o provedor de análise de dados desiste de fornecer sua conexão e faz *logout* no sistema. Por isso, o M-Hub deste provedor informa o IoTrade que ele está se desconectando e o processo de seleção de um novo provedor se inicia, similarmente, ao do primeiro caso.

4.3. Provedor de Serviço sem Sinal de Internet

Para resolver o terceiro caso foi necessário desenvolver e acrescentar um *service* dentro do M-Hub, chamado de *NetworkMonitorService*. Esse *service* fica monitorando o sinal de rede do *smartphone* e ao detectar que se perdeu a conexão, ele apaga a informação sobre quem são os clientes que estão consumindo os seus dados de sua memória. Como o dispositivo está sem sinal de internet foi necessário, também, utilizar o protocolo de *handshake* descrito na Seção 3.3, onde o aplicativo cliente fica enviando constantemente, num intervalo curto de tempo, mensagens de *acknowledge* para verificar se o provedor de serviço está online. A Figura 4.2 mostra o diagrama de sequência com as mensagens que são trocadas no IoTrade para reestabelecer o fluxo de dados de um serviço que estava acontecendo.

Similarmente ao caso da Seção 4.2, podem existir provedores de conectividade e de análise de dados (implementado com base num M-Hub) e caso se perceba que um dos dispositivos contratados não respondeu em algum momento, o processo de encontrar novos provedores para continuar o serviço é iniciado e o provedor que não respondeu é desativado no IoTrade para que não seja selecionado novamente pelo algoritmo. Além disso, existe a possibilidade de um atuador estar conectado a um provedor de conectividade e como ele só responde a comandos enviados pelo cliente, não existiria uma forma de saber se ele está online ao contrário de um sensor que envia um fluxo de dados constantemente. Para detectar a conectividade deste objeto é usado novamente o protocolo de *handshake* e, com isso, é possível saber

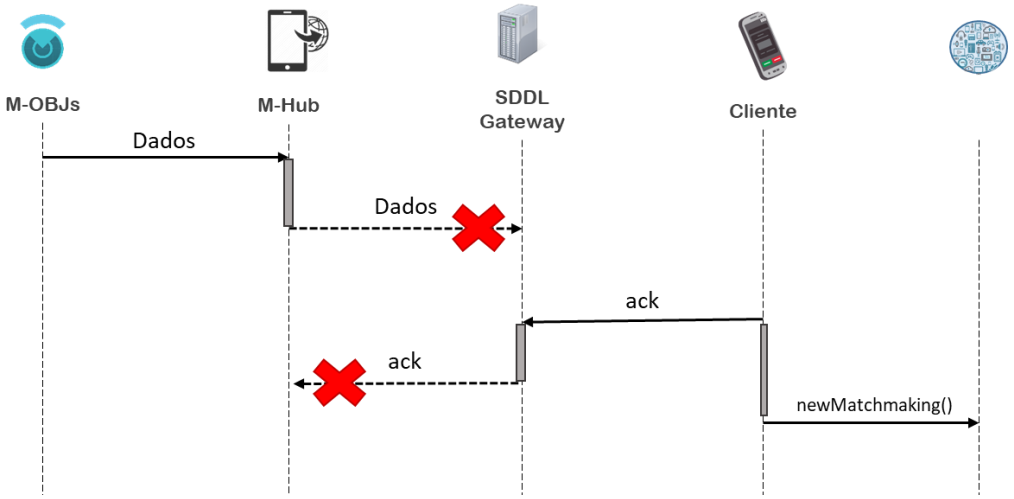


Figura 4.2: Diagrama de sequência das mensagens trocadas entre os agentes para reestabelecer o fluxo de dados

pela mensagem de *acknowledge* se o atuador está funcionando corretamente. Caso contrário é feito o mesmo processo de achar novos provedores de serviço. Por exemplo, o cliente do IoTrade adquiri acesso a um atuador e começa a enviar comandos de atuação para o provedor de conectividade ligado a ele, mas após algum tempo, as mensagens de *acknowledge* não são respondidas, portanto, o IoTrade entende que o provedor perdeu a conexão com a internet e começa o processo de seleção, similarmente aos outros casos descritos.

4.4. Discussão

Este capítulo descreveu o mecanismo de detecção contínua de problemas de conectividade implementado neste trabalho dentro do contexto de uma aplicação IoT que necessita de um fluxo de dados constante, consistente e ininterrupto. Ele constitui um aspecto importante para cenários com ambientes de alta mobilidade e com intensa troca de dados, onde os *gateways* (M-Hub e o SDDL, neste caso) e os objetos inteligentes (e.g. M-OBJs) podem ser móveis. Estes cenários de grande mobilidade se beneficiam de um mecanismo contínuo de detecção de problemas de conectividade pois conseguem antecipar o cliente e prover alternativas em caso de alguma falha.

Com a falta de um mecanismo de detecção de problemas de conectividade, uma aplicação não poderia saber que um objeto inteligente ou *gateway* móvel se desconectou ou perdeu o seu sinal de rede. Portanto, a aplicação deveria implementar mecanismos próprios para lidar com problemas na conectividade dos dispositivos.

Este mecanismo permite que o serviço não seja interrompido e termine inesperadamente pois, após a detecção de algum problema, ele avisa o IoTrade para iniciar o processo de *matchmaking* com intuito de achar novos provedores de serviço. O algoritmo seleciona os provedores com a mesma ou superior qualidade e precisão dos que estavam prestando o serviço, mas caso ele não ache provedores com estes parâmetros são escolhidos aqueles com reputação inferior, com o objetivo de não interromper o serviço. Além disso, o cliente do IoTrade não percebe que houve complicações, ou seja, tudo é gerenciado pelo sistema de forma invisível. O

único caso em que o cliente recebe um aviso é quando não se consegue achar provedores para a continuação do serviço.

Por fim, o mecanismo descrito neste capítulo apresentou uma maneira de detectar continuamente problemas de conectividade no âmbito de uma aplicação IoT e o que fazer em seguida e, com isso, estabelecendo um protocolo com uma convenção definida que controla e possibilita a realização de tal tarefa. No Capítulo 5 serão apresentados e discutidos com mais detalhes os testes de performance realizados a fim de checar este protocolo.

5
Avaliação

Este capítulo relata os experimentos e medições realizados com o objetivo de responder as questões referentes a essa pesquisa feitas na Seção 1.2. Também, são apresentados cenários utilizando a implementação do protótipo do IoTrade, mostrado na Seção 3.4, e que foi desenvolvido com base nos conceitos de um *marketplace* para IoT. A Seção 5.1 detalha as configurações que foram usadas em nossos experimentos, sendo seguida pela apresentação destes cenários. A Seção 5.3 descreve os testes de performance feitos sobre o protótipo e seus resultados, discutindo se o M-Hub, usado como provedor de conectividade, é escalável o suficiente para prover um fluxo de dados para muitos clientes ao mesmo tempo e se o algoritmo de *matchmaking* e o mecanismo de detecção contínua de problemas de conectividade rodam em tempos suficientemente aceitáveis em um ambiente no âmbito de IoT.

5.1.
Configuração do Experimento

Para todos os experimentos foi usado um desktop com processador Intel(R) Core(TM) i7-6700K 4.0 GHz com 16GB de RAM DDR4, rodando Windows 10. Ele foi usado para rodar o servidor que implementa os serviços do *marketplace* IoTrade, o banco de dados que armazena as informações da plataforma e o SDDL Core (*Gateway*, Gerenciador de Queries e Monitor Web), além de simular uma instância que implementa um provedor de análise de dados. Todos os testes foram executados usando uma rede local Wifi (IEEE 802.11bgn).

Os dispositivos usados como M-OBJs foram as SensorTags de modelo CC2541[12] e CC2650[13]. A SensorTag CC2541 possui seis sensores com diferentes tempos de update de dados: Umidade (> 100ms), Temperatura (> 250ms), Giroscó-

[12] Texas Instruments CC2541 Sensor Tag - http://www.ti.com/lit/ml/swru324b/swru324b.pdf
[13] Texas Instruments CC2650 Sensor Tag - http://www.ti.com/lit/ug/tidu862/tidu862.pdf

pio (> 0.125ms), Acelerômetro (> 20ms), Magnetômetro (> 12ms) e Pressão barométrica (> 2ms). A SensorTag CC2650 possui três atuadores (LED Vermelho, LED Verde e buzina) e cinco sensores com diferentes tempos de update de dados: Umidade (> 100ms), Temperatura (> 300ms), Pressão barométrica (> 100ms), Sensor óptico (> 100ms) e Movimento (> 100ms) englobando Giroscópio, Acelerômetro e Magnetômetro.

Para todos os testes de performance e cenários nós usamos o smartphone Xiaomi Mi Mix (modelo 2016), rodando o Android 7.0 Nougat, como o provedor de conectividade através de uma extensão do M-Hub, e o Motorola Moto Maxx (modelo 2014), rodando o Android 7.1 Nougat, com o aplicativo cliente do IoTrade simulando um cliente. Na Seção 5.2, em um dos cenários, foi utilizado um terceiro smartphone para simular um segundo cliente. Neste caso foi utilizado um Samsung Galaxy S7 Edge (modelo 2016) rodando Android 7.0 Nougat. Todas as especificações dos smartphones usados podem ser encontradas na Tabela 5.1. Para evitar interferências externas de outras aplicações dos *smartphones* (e.g. afetar o consumo de bateria ou tempo de processamento), quase todas elas foram desinstaladas com a exceção das que já vinham instaladas de fábrica neles e que puderam ser somente desativadas dentro do Android.

Smartphone	***Chipset***	***CPU***	***Memória (RAM)***	***Android (Versão)***	***Bateria***
Motorola Moto Maxx	Qualcomm Snapdragon 805	4-core (4x2.7 GHz)	3GB	7.1	3900 mAh
Samsung Galaxy S7 Edge	Samsung Exynos 8890	8-Core (4x2.3 GHz + 4x1.6 GHz)	4 GB	7.0	3600 mAh
Xiaomi Mi Mix	Qualcomm Snapdragon 821	4-core (2x2.35 GHz + 2x1.6 GHz)	6GB	7.0	4400 mAh

Tabela 5.1: Especificações dos Smartphones

A tecnologia WPAN usada para a comunicação entre o smartphone com o M-Hub e os M-OBJs foi a Bluetooth Low Energy (BLE). Como essa tecnologia necessita de um tempo para fazer a descoberta dos serviços que um M-OBJ possui e que pode ser um período relativamente longo dependendo da quantidade de sensores e atuadores existentes. Em nossos experimentos pulamos a primeira conexão e começamos a testar a partir das reconexões. Isso foi feito para evitar medidas muitos discrepantes. Outros testes que levam em consideração os tempos de conexão e desconexão, além da descoberta de serviços em M-OBJs conectados por BLE podem ser encontrados em [5]. Sendo assim, todos os experimentos foram iniciados com os M-OBJs já conectados ao M-Hub a priori e, também, todas as informações necessárias já cadastradas no banco de dados, conforme discutidas na Seção 3.4.4, para o funcionamento do protótipo do IoTrade foram realizadas antes do início dos experimentos. Por fim, essa tecnologia BLE tem uma característica chamada notificações que permite informar ao dispositivo *master*, conectado ao *slave* M-OBJ, os valores dos sensores assim que eles forem atualizados. Isso foi utilizado para identificar o momento exato de gerar um novo fluxo de dados e enviá-lo ao cliente no aplicativo cliente.

5.2. Cenários

Nesta seção descreveremos alguns cenários do protótipo do IoTrade, desenvolvido neste trabalho. Foram selecionadas as principais combinações de interação entre os provedores e os clientes para serem os casos de testes. Assim temos um cliente recebendo somente dados brutos de um sensor, outro que adquiriu a análise de dados para receber os dados processados e um atuando sobre um dispositivo.

Para os serviços estarem disponíveis e aparecerem para a compra do cliente, eles precisam estar previamente cadastrados no banco de dados da aplicação. Esse cadastro se faz necessário para o serviço IoTrade possuir maneiras de identificar os provedores de serviço e ter informações sobre eles para aplicar os algoritmos, como por exemplo o *matchmaking*, verificação de QoS e comoditização. Além disso, são atualizadas em tempo real as informações de quais M-OBJs estão ligados aos M-Hubs e a posição geográfica atual dos M-Hubs, ou seja, os serviços são mostrados em tempo real para os clientes. Com isso, caso um sensor ou atuador novo apareça

Figura 5.1: Cliente recebendo analisados (esquerda) e cliente recebendo dados brutos (direita)

na região selecionada e que é do interesse do cliente em monitorar ou controlar, ele poderá ter acesso instantaneamente. Os clientes preenchem os seguintes atributos para a busca de serviços:

- **Preço:** Valor máximo disposto a pagar
- **Localidade:** Área onde é buscado o serviço (padrão é usar a localidade atual, usando as coordenadas do GPS do smartphone)
- **Raio:** Raio máximo para a busca com base na localidade escolhida

Para estes cenários, ambos SensorTags (CC2541 e CC2650) foram usados e se comunicaram localmente através do protocolo BLE com uma instância do M-Hub instalada em um *smartphone*, agindo como provedor de conexão. Foram usados, também, dois smartphones simulando clientes compradores e uma aplicação desktop agindo como provedor de análise de dados, implementada com os mesmos protocolos de comunicação do M-Hub e que se comunica com o SDDL Core.

5.2.1. Dados Brutos e Análise de Dados

Neste cenário a compra de dados brutos e, também, de análise de dados é apresentada. O serviço de *matchmaking* vai selecionar a melhor combinação de provedores de conectividade e dispositivos inteligentes e começará a enviar os dados. No caso da análise de dados, a única diferença é que em vez do provedor de conectividade enviar os dados diretamente para o cliente, ele envia para o provedor de análise de dados e este realiza o processamento antes de enviar. Além disso, é mostrado o tempo que é gasto desde o começo do serviço e o seu custo, que, neste protótipo desenvolvido, aumenta similarmente a um taxímetro por cada fluxo de dados que ele recebe. Na Figura 5.1 são mostrados clientes recebendo diferentes tipos de informação de um mesmo provedor, dados brutos e dados processados em um gráfico.

Um provedor de análise de dados pode oferecer vários serviços, como é mostrado na Figura 5.2 cabendo ao cliente escolher qual deles é o mais adequado as suas necessidades. Todos esses serviços oferecidos são previamente cadastrados no

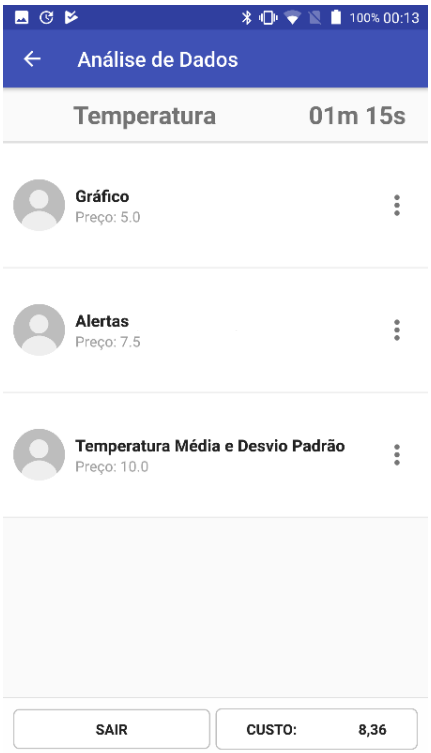


Figura 5.2: Lista com serviços disponíveis de análise de dados

banco de dados e caso o provedor adicione novos, eles irão aparecer automaticamente para os clientes assim que forem cadastrados.

O provedor de conectividade suporta o envio de dados para múltiplos clientes ao mesmo tempo. Existe uma estrutura de dados que mapeia cada sensor e adiciona o identificador de cada cliente nela para saber exatamente para quem enviar.

5.2.2. Atuação

Além de ter serviços que somente enviam dados para o cliente, é possível assumir o controle de um atuador e enviar comandos diretamente para o objeto. Ao adquirir o controle de um atuador, ele será exclusivo e não poderá ser escolhido por algum outro cliente; o sistema, ao detectar que o atuador está em uso, o exclui automaticamente de participar do algoritmo de *matchmaking*.

Funciona da seguinte maneira, no aplicativo do IoTrade, mostrado na Figura 5.3, aparecem todos os comandos de atuação disponíveis para o objeto inteligente

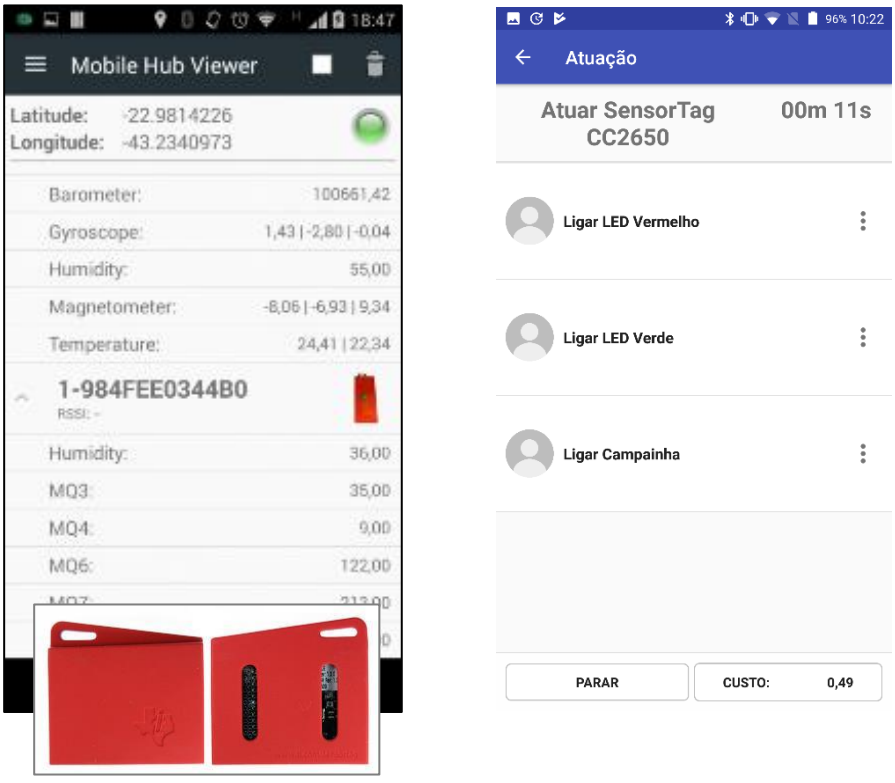


Figura 5.3: Provedor de Conectividade (M-Hub) conectado aos SensorTags (esquerda) e lista de comandos disponíveis para um atuador SensorTag CC2650 (direita)

selecionado, que da mesma forma da análise de dados estão previamente cadastrados no banco de dados. Ao clicar em algum comando, este é enviado para o SDDL Core que faz o seu roteamento até o M-Hub em que o atuador está conectado para que o comando seja executado, sendo que envio deste para o M-OBJ é feito pelo protocolo BLE.

5.3. Testes de Performance e Resultados

Nesta seção descrevemos os experimentos e medições realizados com o objetivo de testar a eficiência de componentes centrais do IoTrade. Nossa intenção é avaliar, através de testes de performance, se o M-Hub, como provedor de conectividade, é escalável o suficiente para fornecer um fluxo de dados para muitos clientes ao mesmo tempo e se o algoritmo de *matchmaking* e o mecanismo de detecção de problemas de conectividade tem funcionamento adequado para um ambiente onde se tem mobilidade irrestrita e um grande fluxo de informação. Foi utilizado uma metodologia de avaliação empírica onde conduzimos uma serie de experimentos quantitativos usando SensorTags, vários smartphones realizando requisições e simulações de M-OBJs para testar a escalabilidade do conjunto M-Hub/Servidor.

Para isso nós usamos o protótipo do *marketplace* IoTrade, que desenvolvemos, para simular a compra de vários serviços e, com isso, medir o tempo que leva para o algoritmo de *matchmaking* selecionar uma combinação de provedores de serviço. Também simulamos desconexões dos provedores para medir o tempo que leva para detectar os problemas de conectividade. Por fim, fizemos uma simulação onde um único M-Hub recebe solicitações de centenas de clientes e verificamos se ele consegue enviar num tempo razoável (< 1000 ms) para todos eles.

5.3.1. Detecção de Problemas de Conectividade pelos M-OBJs

Nesse experimento medimos o tempo, em segundos, dos três casos possíveis de perda de conectividade dentro do IoTrade, desde a desconexão até a nova seleção de provedores pelo serviço de *matchmaking*, e são eles: M-OBJ se desconecta do M-Hub, o provedor desabilita no aplicativo M-Hub o serviço de conectividade e o *smartphone* com o M-Hub fica sem sinal de internet.

Iteração	***M-OBJ Desconectou (s)***	***M-Hub parou de transmitir (s)***	***M-Hub sem internet (s)***
1	1,184	0,556	0,627
2	0,304	0,374	0,902
3	1,457	1,194	0,519
4	1,265	0,231	1,021
5	0,257	0,999	0,780
6	0,348	0,917	0,536
7	0,744	1,109	1,104
8	1,331	0,543	1,128
9	1,084	0,818	0,370
10	1,036	1,156	1,081
Média	**0,901**	**0,790**	**0,807**
Desvio Padrão	**0,455**	**0,343**	**0,279**

Tabela 5.2: Detecção de problemas de conectividade pelo IoTrade

O primeiro teste mede o tempo que leva para o M-Hub perceber que um M-OBJ, que estava sendo usado, perdeu a conexão e avisar o aplicativo IoTrade do cliente. A segunda medição é o caso em que o provedor decide que não quer mais oferecer sua conexão de dados para o público e para de transmitir os dados, ou seja, caso existam M-OBJs transmitindo dados, a arquitetura deve ser capaz de detectar e avisar o aplicativo do cliente, da mesma forma que na outra medição. O terceiro teste mede quanto tempo leva para a arquitetura detectar que o M-Hub ficou sem internet, o que o deixa impossibilitado de enviar dados para o cliente. A Tabela 5.2 apresenta os resultados.

Para os experimentos desta seção foram simuladas compras de dados de sensores, similar aos cenários, usando um *smartphone* com M-Hub como provedor de conectividade e um M-OBJ. Para simular os três casos possíveis de teste foram feitas as seguintes ações: o M-OBJ foi desligado enquanto ele estava fornecendo dados com a intenção de simular uma desconexão, para o M-Hub parar de transmitir o botão de parar os serviços foi pressionado (vide figura 5.4) e, por último, para emular uma perda de internet no M-Hub foram desligados o Wi-Fi e a rede celular (LTE) do *smartphone*.

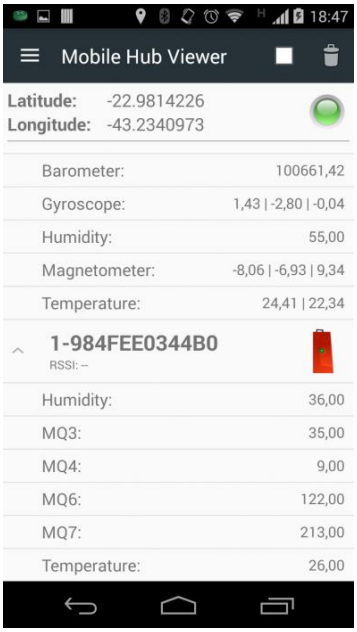


Figura 5.4: Tela do M-Hub com botão de para os *Services* no canto superior direito (*stop*)

Figura 5.5: Simulação de tempo de resposta do algoritmo de *matchmaking* para 500 provedores de conectividade e 50 provedores de análise de dados

Figura 5.4: Tela do M-Hub com botão de para os *Services* no canto superior direito (*stop*)

5.3.2. Simulação de Vários Clientes em um Mesmo M-Hub

Nesse teste simulamos que um M-Hub estava enviando dados para centenas de clientes diferentes, justamente para verificar se a arquitetura desenvolvida suportava escalar em um único provedor tantos clientes de uma única vez. Aqui, usamos um *smartphone* como o provedor de conectividade (M-Hub) e um M-OBJ. Para a realização do teste, simulamos que 500 clientes pediram dados de cinco dos seis sensores do M-OBJ, ou seja, para cada sensor existem 100 clientes recebendo dados deles. Estes 500 clientes foram inseridos, manualmente, direto na estrutura de dados, conforme mostra o código 5.1, que registra todos os clientes para quem ele deve mandar tal dado de um sensor. Os resultados estão na tabela 5.3.

5.3.3. Algoritmo de Matchmaking

Nós fizemos alguns experimentos preliminares com o algoritmo para determinar se era robusto e escalável o suficiente para trabalhar com muitos sensores e prestadores de serviços ao mesmo tempo. No IoTrade a informação de onde os pro-

Iteração	***M-Hub Enviou (s)***
1	0,103
2	0,061
3	0,763
4	1,083
5	1,092
6	0,312
Média	**0,569**
Desvio Padrão	**0,473**

Tabela 5.3: Simulação de envio para múltiplos clientes ao mesmo tempo

vedores e objetos inteligentes estão localizados e os seus atributos estão armazenados no banco de dados do sistema e atualizados em tempo real. Para estes testes em que estamos medindo os tempos de resposta de um algoritmo, nenhum dispositivo real foi usado devido à escala de milhares de dispositivos que precisaríamos**;** entretanto, simulamos informações de até 10.000 sensores, 1.000 provedores de conectividade e 50 provedores de análise de dados, inserindo-as manualmente no banco de dados. Assim, o serviço de *matchmaking* interpreta que as informações no banco de dados desses provedores de serviço são reais e as leva em consideração para a seleção. Os resultados estão mostrados nas figuras 5.6 e 5.7.

Para realizar os experimentos desta seção foram simuladas compras de dados de sensores, similarmente como foi feito na seção 5.3.1, usando um *smartphone* com M-Hub como provedor de conectividade, um M-OBJ e um outro *smartphone* executando o aplicativo do IoTrade fazendo simulações de solicitação de compra de fluxo de dados, que geram chamadas ao algoritmo de *matchmaking*. Os parâmetros de QoS não foram mudados durante este teste e eles foram equivalente para todos os provedores e M-OBJs, ou seja, a reputação com a máxima pontuação, o preço de *commodities* de cada um não foi alterado e provedores de conectividade com bateria e sinal de internet no máximo.

5.3.4. Discussão

Os experimentos para a detecção de problemas de conectividade, mostrados na seção 5.3.1, foram rodados 10 vezes e foram calculados a média e o desvio padrão das medições. Como se pode ver na Tabela 5.2, apesar de serem medições de casos de teste diferentes, os tempos médios variaram em menos de 0,095 segundos. Além disso, na maior parte do tempo a detecção ocorreu bem abaixo de 1 segundo, o que é um tempo bastante razoável para o IoTrade se recuperar e começar a selecionar novos provedores e M-OBJs, sem que o cliente perceba que a conexão foi perdida.

Os resultados de enviar dados para múltiplos clientes ao mesmo tempo foram bastante positivos, já que num cenário bastante estressante o provedor conseguiu enviar, sem problemas, uma grande quantidade de dados para essa centena de clientes. Isso só foi possível porque os dados não foram enviados individualmente para cada smartphone mas sim, de uma vez só, assim que a callback *onMObjectValueRead()*, disparada toda vez em que é lido um dado novo de algum sensor, é chamada. Funciona da seguinte maneira: é verificado se existem clientes no IoTrade

```
15  int tam = 500;
16  ArrayList<String> arrayList = new ArrayList<>();
17  for(int i =0 ; i<tam/5; i++){
18      arrayList.add("Teste"+i);
19  }
20  for(int i=0;i<5;i++){
21      if(!mActiveRequest.containsKey(macAddress)) {
22          ConcurrentHashMap<String, ArrayList<String>> map = new Con-
    currentHashMap<>();
23          map.put(uuidDatas.get(i%5), arrayList);
24          mActiveRequest.put(macAddress, map);
25      }else{
26          ConcurrentHashMap<String, ArrayList<String>> map = mAc-
    tiveRequest.get(macAddress);
27          map.put(uuidDatas.get(i%5), arrayList);
28          mActiveRequest.put(macAddress, map);
29      }
30  }
```

Código 5.1: Inserção manual de 500 compradores

que compraram dados através da estrutura de dados que guarda os identificadores dos clientes e, então, é enviada esta estrutura de dados junto com o valor do sensor lido para o SDDL Core que faz um broadcast para todos esses clientes. Como podemos ver na Tabela 5.3 o envio foi realizado corretamente para todos eles e com um tempo médio abaixo de 600 ms.

Para os testes de desempenho do algoritmo de *matchmaking*, foram criados vários cenários, mostrados nas figuras 5.5 e 5.6, com as mais variadas combinações de sensores, provedores de conectividade e análise de dados, a fim de "estressar" o servidor e testar quanto escalável o algoritmo poderia ser. As medidas de tempo foram feitas através de uma média aritmética executando 12 vezes cada cenário de teste. Os resultados foram positivos já que, nos cenários mais estressantes, o tempo não aumentou muito em relação aos outros. Isso mostra que o algoritmo funcionaria bem em condições de tempo real com troca de dados intensa.

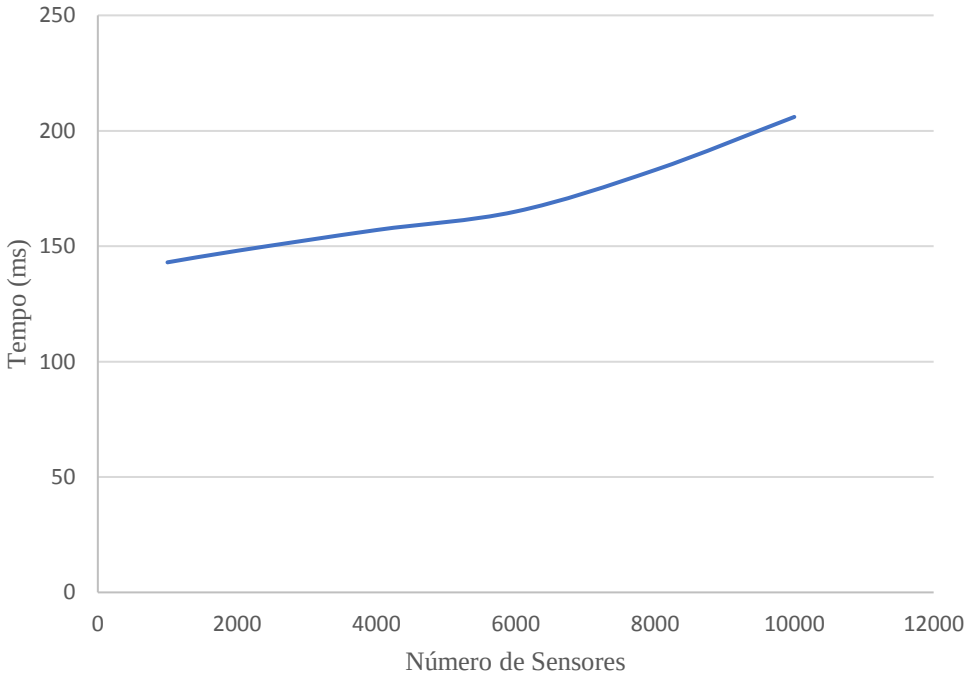


Figura 5.5: Simulação de tempo de resposta do algoritmo de *matchmaking* para 500 provedores de conectividade e 50 provedores de análise de dados

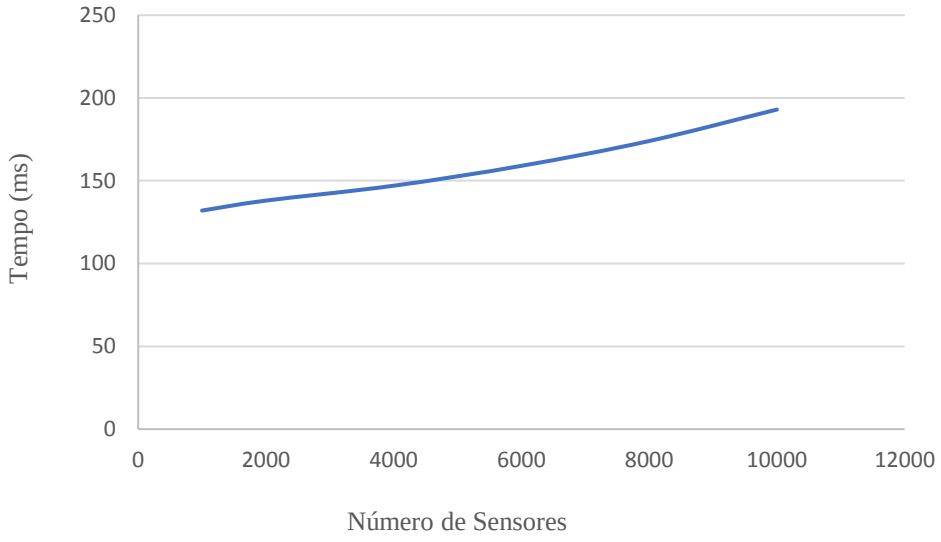


Figura 5.6: Simulação de tempo de resposta do algoritmo de *matchmaking* para 1000 provedores de conectividade e 50 provedores de análise de dados

6
Trabalhos Relacionados

Espera-se que até 2020 o número de dispositivos interligados atinja 100 bilhões [28] gerando uma receita de US $ 1,3 trilhão [3]. No entanto, existem muito poucos *marketplaces* de IoT em operação para tirar proveito dessa crescente tendência e a maioria deles usa soluções proprietárias com vendas de pacotes fechados, interoperabilidade inexistente com outras soluções no mercado e sem algum tipo de mecanismo que detecte problemas de conectividade entre dispositivos com reconexão automática. As plataformas multilaterais, os *marketplaces*, os serviços de *matchmaking* foram estudados recentemente [20] e muito sobre os padrões de design foram aprendidos de empresas como Amazon, Uber, AirBnB e Alibaba.

Este trabalho também apresenta um mecanismo assíncrono para detectar a perda de conectividade no âmbito desse conceito de *marketplace* para IoT, onde é necessária a manutenção de um fluxo constante e interrupto de dados enquanto um serviço está sendo prestado. Esse mecanismo utiliza o paradigma *Publish-Subscribe* para lidar com os aspectos da comunicação entre os produtores e consumidores de dados (e.g. cliente e provedores do serviço) através do uso de *brokers*. Não são encontrados na literatura trabalhos que propõem um conceito de *marketplace* para IoT com detecção contínua de problemas de conectividade enquanto existe um fluxo de dados constante entre dispositivos, entretanto, existem trabalhos com conceitos similares que já foram desenvolvidos para aproveitar esse crescente interesse pelas aplicações IoT sendo alguns apenas teóricos e outros já estão em operação no mercado.

Em [23] o autor apresenta um modelo de compra e venda de dados IoT chamado *Sensing-as-a-Service*. É proposto que os dados de sensores dos objetos inteligentes sejam vendidos para pessoas e empresas interessadas, ou seja, uma maneira de ganhar dinheiro com dados que antes seriam desperdiçados e que podem gerar informações valiosas (e.g. comportamento, hábitos das pessoas e padrões de vida). Além disso é uma maneira de estimular a adoção de soluções de IoT e recuperar o

investimento feito ao adquiri-las. O trabalho, entretanto, não propõe nenhuma infraestrutura de mercado com arquitetura e atores definidos, focando somente no modelo de negócios e cenários.

Em [2] são apresentados conceitos para a criação de um *marketplace* de sucesso voltado para IoT. Os autores mostram como um *marketplace* digital gera valor através da criação de um ecossistema, novas oportunidades de monetização e a criação de uma porta de entrada para a descoberta de dados. No trabalho não é mostrada uma implementação real dos conceitos, focando apenas na parte teórica de como desenhar uma plataforma de compartilhamento de dados IoT e como escolher um modelo de monetização.

Um exemplo de um *marketplace* que está operacional desde abril de 2017 é o Bridging the Interoperability Gap of the Internet of Things[14] (BIG IoT Marketplace) onde os clientes podem criar seus próprios serviços através de uma API disponível na plataforma e ofertá-los para o público geral ou podem, simplesmente, comprar serviços já disponíveis e que foram criados por outros clientes. O *marketplace* disponibiliza aos clientes suporte para publicidade e monetização das soluções desenvolvidas por eles.

O ThingWorx Marketplace[15] é uma plataforma tanto para compra de soluções de IoT prontas ou para a criação de aplicações, onde o cliente pode encontrar ferramentas, peças, dispositivos, software e parceiros para ajudar na implementação da solução. Também é disponibilizada a possibilidade de os clientes promoverem e oferecerem as soluções que eles próprios criaram na plataforma após uma aprovação feita previamente pelo *marketplace*.

O IoT Marketplace[16] oferece soluções de pacotes fechados de IoT para o cliente, onde eles podem escolher entre o sensor, a análise de dados e o serviço de conectividade. Esta solução é semelhante à anterior ThingWorx Marketplace, onde as ferramentas são oferecidas e o cliente cria a solução final. Os custos dessa solução são elevados, pois é necessário adquirir a solução completa (sensores e *gateway*) e pagar, separadamente, uma taxa mensal pela análise de dados.

[14] BIG IoT Marketplace, 2017. - http://big-iot.eu/project/
[15] ThingWorx, 2011. - https://www.thingworx.com/
[16] The IoT Marketplace, Libelium, 2016. - https://www.the-iot-marketplace.com/why-an-iot-marketplace

Alguns estudos recentes abordam protocolos para Negócios Eletrônicos (E-Business), modelos de negócios e esquemas de princípios focados no IoMT. Em [34] os autores propõem um modelo de negócios baseado no protocolo Bitcoin para lidar com E-Business com um modelo de transação P2P baseado em Blockchain. É apresentado o conceito das Decentralized Autonomous Corporations (corporações descentralizadas autônomas) que são "agências reguladoras" autônomas, que não estão ligadas a nenhum país, nas quais os usuários negociam diretamente as cripto-moedas, que são as moedas de troca nesse modelo. Também é mostrado o conceito de *commodities* de dados para esse tipo de mercado, englobando os dados pagos e propriedades inteligentes, que tem em comum a capacidade de serem transmitidas pela rede ou controladas diretamente por um dispositivo digital.

Em [35] os autores propõem um conceito de trocar a informação gerada através de um sensor por dinheiro eletrônico. Eles apresentam uma visão de uma infra-estrutura *Sensing-as-a-Service* desconstruída, onde o sensor envia automaticamente seus dados para um repositório e os oferece através de um *marketplace* para quem quiser adquirir. No trabalho é usado o *Bitcoin* como moeda de troca juntamente com o *Blockchain* para armazenar as transações feitas. No entanto, este trabalho não caracteriza os principais atores em um mercado voltado para IoT e não discute as características necessárias para a construção de um bom *marketplace*.

Em [38], um método é apresentado para classificar os serviços dos sensores com base na estimativa do custo de acesso aos seus serviços em redes de sensores sem fio. O uso do modelo *Sensing-as-a-Service* é defendido para permitir um meio de acesso e controle dos dispositivos sensores em aplicativos IoT. Os autores se concentraram na descoberta de serviços e propuseram um algoritmo que estima o custo de acesso aos serviços de sensores existentes na rede com base em parâmetros de energia e QoS. Porém, neste trabalho, não é mencionado um método para seleção de sensores com base nesse ranqueamento.

O trabalho [37] apresenta uma pesquisa de modelos econômicos e abordagens baseadas em princípios para gerenciar e selecionar sensores em redes de sensores sem fio para o IoT, voltadas para negociação de recursos entre as partes interessadas. E os autores se concentram especificamente nos recursos de detecção, na seleção da rede de acesso subjacente, no modo de endereçamento (ou seja, *anycast*, *unicast*, *multicast* e transmissão *broadcast*), nas capacidades do próprio sistema IoT e nas opções de segurança (controle de acesso, autenticação, etc.). Os recursos IoT

que podem ser negociados são, de acordo com o trabalho, os dados sensoriados, a energia para os dispositivos e serviços de servidor, o armazenamento na nuvem e os serviços de computação, o espectro de rede sem fio e a banda larga, os serviços de dados e informações (na nuvem ou nó) e os serviços baseados em localização. Entretanto nenhum *marketplace* específico ou algoritmo de *matchmaking* são propostos e implementados, apenas várias estratégias possíveis de geração de recursos são discutidas como: precificação baseada em custos e uso, precificação baseada no valor percebido pelo consumidor, oferta e demanda.

Em [36], os autores analisam os incentivos econômicos / de preços para detecção participativa e propõem um novo mecanismo de incentivo denominado RADP (*Reverse Auction Based Dynamic Price*). Nesse mecanismo de incentivo, os clientes colocam a venda os seus dados sensoriados por um valor, um provedor de serviços seleciona um número predefinido de clientes com os preços de lances mais baixos e, em seguida, os clientes selecionados recebem o valor do lance pelos seus dados sensoriados como recompensa. Assim o preço de venda, neste mecanismo, muda dinamicamente com base nos preços de oferta dos clientes. De acordo com o estudo, este mecanismo de incentivo "não só reduz o custo de incentivo para manter o mesmo número de participantes em mais de 60%, mas também melhora a equidade de distribuição de incentivo e bem-estar social".

Em [40] é feita uma comparação dos protocolos existentes hoje e que levam em consideração o QoS (*Quality of Service*) para IoT em redes WSN. Os autores analisam os protocolos em relação a escalabilidade, eficiência energética e em relação a dependência de topologia. O trabalho, então, propõe um protocolo para integrar dispositivos IoT em uma WSN usando QoS por meio de uma densa rede de pontos de acesso IEEE 802.15.4. Além disso, é demonstrada a viabilidade desse protocolo através de exemplos de aplicações no âmbito de IoT. Entretanto, não são informados os mecanismos utilizados para verificar e medir o QoS dos dispositivos.

Embora existam muitos protocolos para IoT, a grande maioria deles não considera objetos inteligentes móveis [5] e meios de detectar perdas de conectividade entre eles e os gateways. Em [15] o problema da perda de comunicação em dispositivos IoT é examinado usando um mecanismo de compartilhamento de conexão NFC. Os autores propõem usar a tecnologia NFC para o compartilhamento de conexão de dados. O dispositivo que não tem conexão ou que perdeu, temporaria-

mente, sua conectividade com a internet, compartilha seus dados através do protocolo NFC para um dispositivo que esteja com acesso a rede funcionado e, com isso, ele pode fazer o roteamento e enviar as informações pela internet, agindo como se fosse um *gateway*.

Em [39] é abordado o problema da conectividade de dispositivos de IoT, que são dependentes de *gateways* para se conectarem a internet. Os autores propõem uma arquitetura em que o *smartphone* seja o principal provedor de acesso à rede dos dispositivos de IoT. A conexão e envio de dados é feita através do protocolo Bluetooth Low Energy (BLE), que cria uma ligação ponto-a-ponto com o *smartphone*. Não é mencionado se o *smartphone* processa os dados recebidos, ou seja, se também age como um provedor de análise de dados.

6.1. Discussão

Os trabalhos apresentados nesse capítulo exibem algum grau de semelhança com a pesquisa desenvolvida nesta dissertação. Os *marketplaces* IoT Marketplace, ThingWorx e Bridging the Interoperability Gap são os que possuem o maior grau de semelhança, pois são plataformas multilaterais que lidam com dispositivos móveis, provedores de conectividade e análise de dados, mas ao contrário destes que possuem apenas interações consumidor-produtor (compra e venda), acreditamos que os *marketplaces* voltados para IoT são mais complexos do que isso, e os produtores de dados, provedores de serviços em nuvem e provedores de acesso à rede, por um lado, e os gerenciadores de aplicativos IoT, por outro lado, devem encontrar entre si, negociar e estabelecer contratos de vários níveis a fim de garantir um bom funcionamento das aplicações IoT. A Tabela 6.2 apresenta um resumo comparativo entre os *marketplaces* citados e os critérios de comparação entre eles estão na Tabela 6.1. Especificamente, o objetivo desta comparação é discutir todos os aspectos relevantes e comparar seu uso em cada um dos *marketplaces* apresentados.

Para que um mercado funcione bem, deve apresentar algumas características básicas e bem reconhecidas, como espessura, confiabilidade e equilíbrio. Utilizar mecanismos [23, 35, 36] que incentivem os clientes e os provedores de serviços a

participarem do *marketplace* é fundamental para o seu funcionamento. Por exemplo, oferecer cupons de desconto e programas de fidelidade são os mecanismos mais comuns usados por plataformas digitais para manter o cliente engajado e gastando

	Critério	**Descrição**
1	Centralizado (C) / Distribuído (D)	Os *marketplaces* que são plataformas centralizadas são chamados (C) e plataformas distribuídas chamadas (D)
2	Produtor / Consumidor ou Plataforma Multilateral	O Produtor/Consumidor (P/C) refere-se ao marketplace que produz e vende suas próprias soluções, enquanto a Plataforma Multilateral (MP) refere-se a *marketplaces* que conectam vários fornecedores em um local comum para a venda de serviços (neste caso, o sensor / atuador / acesso análise de dados e uso)
3	Compra ou Consumo de Serviços / Dados	Compra refere-se ao *marketplace* que vende a solução como um pacote fechado e o Consumo de Serviços / Dados refere-se ao *marketplace* em que o cliente só pode comprar um serviço ou adquirir dados brutos de objetos inteligentes sem comprar o pacote fechado (Objetos Inteligentes, Provedor de Conectividade e Análise de Dados)
4	Tolerância a Falha	O *marketplace* que oferece mecanismos de tolerância a falhas**,** que detectam automaticamente um problema e reinicia o serviço são referenciados por (A), enquanto que (M) se referem àqueles que, quando ocorre uma falha, avisam ao cliente que eles devem reiniciar o serviço ou ligar para a assistência técnica
5	Análise de Dados	Se o *marketplace* oferece serviços de análise de dados por um aluguel mensal é referenciado por (R), enquanto (D) se refere àqueles que oferecem o serviço sob demanda
6	Escalabilidade	Se o *marketplace* é escalável, onde oferece uma estrutura para suportar a crescente demanda de clientes, é classificado como (S), se não for escalável, é classificado como (NS)
7	QoS	Se o QoS for levado em consideração pelo *marketplace* e se houver um controle de qualidade**,** mesmo após a conclusão da venda do serviço**,** é referenciado por (X)

Tabela 6.1: Critérios de comparação utilizados na Tabela 6.2

dinheiro nelas. Neste trabalho, um dos mecanismos para incentivar os provedores de conectividade a manter o seu serviço com qualidade é premiar, com um retorno financeiro maior, aqueles que mantem seu sinal de rede maior, beneficiando também os clientes que terão uma maior qualidade no fluxo de dados que receberem.

Existem vários protocolos que levam em consideração o QoS para dispositivos IoT e eles são os responsáveis por medir e verificar a qualidade da rede. Os trabalhos [40, 41] propõem, respectivamente, um protocolo e uma arquitetura para fazer o controle do QoS de sensores em redes WSN, mas nenhum deles leva em consideração o QoS de um sistema que engloba, além de sensores, provedores de conectividade e análise de dados. Neste trabalho, propomos um serviço que é responsável por verificar periodicamente o QoS de todos esses provedores de serviço e, automaticamente, tomar providências caso seja detectado que o QoS de algum componente não passou na verificação.

Os trabalhos [37] e [38] abordam, respectivamente, a seleção de sensores em uma WSN e o ranqueamento de serviços de sensores em uma WSN. Ambos guardam similaridades com o algoritmo de *matchmaking* proposto neste trabalho, onde este tem a tarefa de selecionar uma combinação entre objetos inteligentes e provedores de conectividade e de análise de dados com base na geolocalização do cliente e no ranqueamento feito a partir de parâmetros derivados do QoS de cada um dos provedores de serviço. Apesar disso, os trabalhos citados focam apenas em sensores dentro de uma WSN e não no âmbito de um *marketplace* de IoT.

Embora existam trabalhos que utilizam métodos para lidar com o problema de conectividade em IoT [15, 39], nenhum deles propõe um mecanismo para detectar continuamente problemas de conectividade em um ambiente de fluxo constante e interrupto de dados. O mecanismo que nós propomos utiliza *Listeners* implementados dentro do M-Hub (provedor de conectividade) e que possibilita a detecção de desconexão de objetos próxima ao tempo real. Neste trabalho, defendemos o uso de mecanismos ativos no processo de detecção de problemas de conectividade em tempo real, dada a natureza de alta mobilidade existente em IoT, onde se tem uma grande probabilidade de problemas desse tipo acontecerem frequentemente e levando em consideração a escassez de recursos computacionais dos objetos inteligentes para lidar com isso.

O mercado de dados e serviços para IoT deve ter um crescimento importante nos próximos anos e parece que nenhuma das plataformas mencionadas acima está

Nome	**1**	**2**	**3**	**4**	**5**	**6**	**7**
BIG IoT Marketplace	(D)	(MP)	Consumo de Serviços / Dados	(M)		(NS)	
ThingWorx	(D)	(MP)	Compra	(M)	(R)	(S)	X
The IoT Marketplace	(C)	(P/C)	Compra	(M)	(R)	(S)	X
IoTrade	(C)	(MP)	Consumo de Serviços / Dados	(A)	(D)	(S)	X

Tabela 6.2: Avaliação de *Marketplaces*

verdadeiramente preparada para a escala gigantesca e a enorme diversidade de objetos inteligentes, tecnologias sem fio e serviços IoT. A grande heterogeneidade de objetos inteligentes, os vários tipos de provedores de conectividade que vão desde o LTE IoT (eMTC / NB-IoT) até soluções mais simples Wifi / WPAN e baseadas em smartphones, e um amplo escopo de serviços de análise de dados representam um desafio de como esses fornecedores serão comparados, selecionados e combinados. Para isso, acreditamos que os proprietários de objetos inteligentes e provedores de conectividade e análise de dados devem ser categorizados de acordo com a qualidade, confiabilidade, precisão e complexidade dos dados ou serviços, da mesma forma como as *commodities* de hoje são categorizadas com base em parâmetros de qualidade e onde o preço pago é diretamente proporcional ao QoS entregue, similarmente como é proposto em [34]. Isso torna esses dados IoT, acesso e análise numa mercadoria onde o valor do serviço é baseado na qualidade da categoria escolhida e o agrupamento de provedores tem o objetivo de reduzir custos transacionais [13].

7
Conclusão e Trabalhos Futuros

A popularização e a proliferação de dispositivos móveis (*smartphones* e *tablets*) juntamente com o aparecimento de objetos inteligentes nos últimos anos tem ajudado bastante a impulsionar o crescimento de IoT. É amplamente reconhecido que isso irá causar um grande impacto em todos os aspectos econômicos e sociais de nossa sociedade, o que proporciona muitas oportunidades e desafios de pesquisa.

Nosso trabalho apresenta uma abordagem para a detecção contínua de problemas de conectividade entre dispositivos e nós móveis, dentro de um ambiente de mobilidade irrestrita baseado em conceitos de um *marketplace* para IoT. Partindo do fato de que em um *marketplace* ao se adquirir um serviço, o cliente estabelece um contrato de compra com a parte que fez o anúncio deste. No caso específico dessa pesquisa, os donos de M-OBJs e os nós móveis vendem, respectivamente, seus dados, a sua conexão e o seu poder computacional para a análise de dados. Por estabelecerem um contrato de serviço, o fluxo de informação ou envio de comandos deve se manter constante, consistente e ininterrupto até o término deste. A respeito disso, este trabalho apresentou uma abordagem para usar um protocolo de mensagens assíncrono e uma extensão do M-Hub para criar mecanismos que continuamente realizam a detecção de problemas de conectividade e que consigam restaurá-la rapidamente, mantendo invisível para o cliente qualquer falha que tenha acontecido ao longo da vigência do contrato de serviço.

Nós apresentamos uma extensão do M-Hub juntamente com o serviço IoTrade, desenvolvidos, respectivamente, na plataforma Android e Node.js. A extensão foi desenvolvida para permitir a detecção de problemas de conectividade no M-Hub e nos M-OBJs ligados a ele utilizando um protocolo de comunicação baseado no paradigma *Publish-Subscribe*. O serviço IoTrade, implementado com base nos conceitos de um *marketplace* IoT, executa serviços que verificam constantemente o QoS dos nós móveis, realizam o *matchmaking* dos provedores de serviço e a comoditização dos dados e provedores (conectividade e análise de dados) com base

no QoS destes. É disponibilizada uma API REST para o IoTrade que auxilia desenvolvedores de aplicações a acessar e usar estes serviços desenvolvidos. Nós, também, apresentamos testes em dispositivos reais para mostrar o algoritmo de *matchmaking* e a detecção contínua de problemas de conectividade. Por último, para testar a escalabilidade da arquitetura, simulamos o envio e recebimento de um fluxo de dados num ambiente com milhares de M-OBJs e nós móveis.

Encarar a mobilidade irrestrita dos dispositivos móveis é um desafio em várias áreas dentro de sistemas distribuídos, especialmente, quando se precisa ter um fluxo de dados constante, já que, facilmente, acontecem problemas na conexão desses dispositivos com capacidades limitadas de IoT devido aos protocolos e tecnologias implementados neles. Foi desenvolvida uma aplicação hipotética no âmbito de um *marketplace* para IoT, a fim de demonstrar a proposta arquitetural de uma plataforma multilateral para IoT e apresentar um cenário onde o mecanismo de detecção contínua de problemas de conectividade pode ser aplicado. Construir essa aplicação não demanda muito esforço em termos de desenvolvimento, pois a API REST do IoTrade e a extensão do M-Hub suportam o processo de detecção, reconexão e uso de M-OBJs, bem como o gerenciamento dos nós móveis.

Finalmente, embora tenhamos encontrado alguns trabalhos relacionados que possuem algum grau de semelhança com a pesquisa apresentada nesta dissertação, nenhum deles se refere diretamente a uma arquitetura para um *marketplace* de IoT com mecanismos de detecção contínuo de problemas de conectividade num ambiente que necessite manter um fluxo constante e ininterrupto de dados, como o que foi mostrado e discutido neste trabalho. A abordagem aplicada aqui é inovadora dentro desse contexto.

7.1. Trabalhos Futuros

Levando em consideração que os resultados que foram apresentados mostraram que o mecanismo de detecção contínua de problemas de conectividade foi eficiente e que o tempo de resposta está dentro do esperado para uma aplicação IoT, estamos cientes de que melhorias, pesquisas, desenvolvimento de software e aplicações podem ser derivados desse trabalho. Portanto, dentro dessa seção pretendemos apresentar alguns trabalhos futuros que podem ser desenvolvidos ou estendidos

a partir dessa pesquisa. Em particular, nosso trabalho futuro inclui: estender o algoritmo de matchmaking para que ele possa ser customizado de acordo com as prioridades e critérios da aplicação que esteja sendo executada pelo cliente (e.g. parâmetros de precisão variáveis dependendo da exatidão que se quer o dado), propor uma maneira de integrar o monitoramento do QoS constante dos provedores e objetos inteligentes na arquitetura do protótipo do IoTrade, investigar os problemas e possíveis abordagens para que quando um M-Hub, que esteja prestando um serviço como um provedor de dados, mesmo perdendo a conexão com a Internet consiga fazer o *handover* para algum outro M-Hub que esteja nas redondezas, através de um protocolo de comunicação que não exija conexão com a rede. Isso seria útil para diminuir os tempos de detecção de problemas de conectividade e garantir o restabelecimento do serviço mais rapidamente.

Um aspecto válido de ser investigado é o consumo de energia pelo M-Hub executado em *smartphones*, visto que a autonomia da bateria destes dispositivos ainda é um limitador para o seu funcionamento. A otimização de seu funcionamento junto com um estudo sobre o seu consumo colaboraria para prever quanto tempo o dispositivo teria de carga antes de desligar e, com isso, poder tomar providências prévias para evitar a interrupção do serviço, como, por exemplo, preparar o *handover* para outro M-Hub com maior autonomia de bateria.

Outro aspecto a se desenvolver é aumentar o nível de categorização dos dados para que o cliente possa ter uma maior liberdade e flexibilidade na hora de escolher um sensor ou atuador, colocando filtros e opções mais específicas, por exemplo, ao querer alugar um carro o cliente vai poder escolher a cor, modelo, as características e as funcionalidades dele.

Por último, um outro aspecto importante a ser investigado é a escalabilidade do conjunto servidor, *middleware* e extensão do M-Hub num ambiente com múltiplas instâncias rodando do *middleware* e milhares de M-OBJ interagindo com os nós móveis. A escalabilidade é um parâmetro que é vital para qualquer sistema ou aplicação IoT, pois é um cenário de intensa troca de dados em tempo real e onde se tem perspectivas bastante realistas de que a quantidade de objetos inteligentes disponíveis no mercado irá aumentar num futuro próximo.

8
Referências bibliográficas

[1] C. Perera, C. H. Liu and S. Jayawardena, "**The Emerging Internet of Things Marketplace From an Industrial Perspective: A Survey,**" in *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 4, pp. 585-598, Dec. 2015.

[2] Johannes Deichmann, Kersten Heineke, Thomas Reinbacher, and Dominik Wee. **Creating a successful Internet of Things data marketplace** [Internet Article], October 2016. Retrieved from http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/creating-a-successful-internet-of-things-data-marketplace [Acessado 05-06-2017]

[3] GUBBI, J. et al. **Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions**. Future Generation Computer Systems, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 29, n. 7, p. 1645{1660, set. 2013. ISSN 0167-739X}.

[4] Bagloee, S.A., Tavana, M., Asadi, M. et al. **J. Mod. Transport.** (2016) 24: 284.

[5] L. Talavera Rios, M. Endler, I. Vasconcelos, R. Vasconcelos, M. Cunha, and F. Silva e Silva. **The mobile hub concept: Enabling applications for the internet of mobile things**. In *12th IEEE Workshop on Managing Ubiquitous Communications and Services (MUCS 2015)*, pages 123– 128, 2015.

[6] DACOSTA, F. **Rethinking the Internet of Things: A Scalable Approach to Connecting Everything**. 1st. ed. Berkely, CA, USA: Apress, 2013. ISBN 1430257407, 9781430257400.

[7] M. Endler, R. O. Vasconcelos, L. David, R. André, and L. Alves, "**A DDS-based middleware for scalable tracking and communication of wireless-connected mobile nodes in a WAN,**" Monografias em Ciência da Computação - MCC 06/2012, Dep. de Informática, PUC-Rio, ISSN 0103-9741. Monografias em Ciência da Computação - MCC 06/2012, Dep. de Informática, PUC-Rio, ISSN 0103-9741, Rio de Janeiro, 2012.

[8] L. D. Silva, R. Vasconcelos, R. A. Lucas Alves, G. Baptista, and M. Endler, "**A Communication Middleware for Scalable Real-time Mobile Collaboration,**" in *Proc. of the IEEE 21st International WETICE, Track on Adaptive and Reconfigurable Service-oriented and component-based Applications and Architectures (AROSA),* 2012, pp. 54–59.

[9] M. Endler, R. O. Vasconcelos, L. David, R. André, and L. Alves, "**A DDS-based middleware for scalable tracking and communication of wireless-connected mobile nodes in a WAN,**" Monografias em Ciência da Computação - MCC 06/2012, Dep. de Informática, PUC-Rio, ISSN 0103-9741. Monografias em Ciência da Computação - MCC 06/2012, Dep. de Informática, PUC-Rio, ISSN 0103-9741, Rio de Janeiro, 2012.

[10] L. D. Silva, R. Vasconcelos, R. A. Lucas Alves, G. Baptista, M. Endler, L. David, L. Alves, and R. André, "**A Large-scale Communication Middleware for Fleet Tracking and Management,**" in *Salão de Ferramentas, Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2012),* 2012, pp. 54–59.

[11] I. Vasconcelos, R. Vasconcelos, G. Baptista, C. Seguin, and M. Endler, "**Desenvolvendo Aplicações de Rastreamento e Comunicação Móvel usando o Middleware SDDL,** " in *Salão de Ferramentas, Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2013),* 2013.

[12] I. Awan, M. Younas and W. Naveed, "**Modelling QoS in IoT Applications,**" *2014 17th International Conference on Network-Based Information Systems,* Salerno, 2014, pp. 99-105.

[13] Luigi Atzori, Antonio Iera, Giacomo Morabito, **The Internet of Things: A survey**, Computer Networks, Volume 54, Issue 15, 2010, Pages 2787-2805.

[14] Karagiorgou, Sophia & Stamoulis, Georgios & Kikiras, Panagiotis. (2012). **Enabling QoS in the Internet of Things**. CTRQ 2012, The Fifth ….

[15] Turk I., Cosar A. (2015) **Internet Connection Sharing Through NFC for Connection Loss Problem in Internet-of-Things Devices**. In: Balandin S., Andreev S., Koucheryavy Y. (eds) Internet of Things, Smart Spaces, and Next Generation Networks and Systems. ruSMART 2015. Lecture Notes in Computer Science, vol 9247. Springer, Cham.

[16] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. **The many faces of publish/subscribe**. *ACM Comput. Surv.* 35, 2 (June 2003), 114-131.

[17] Rodríguez-Domínguez, Carlos et al. "**A Communication Model to Integrate the Request-Response and the Publish-Subscribe Paradigms into Ubiquitous Systems.**" Sensors (Basel, Switzerland) 12.6 (2012): 7648–7668. PMC. Web. 14 Jan. 2018.

[18] David S. Evans, Richard Schmalensee. **Matchmakers: The New Economics of Multisided Platforms**. Harvard Business Review, 2016.

[19] OSTERWALDER, A.;PIGNEUR, Y. **Business model generation**. Alta books, Rio de Janeiro. 2014. P. 77-81.

[20] Alvin Roth. **Who gets what, and why: The New Economics of Matchmaking and Market Design**. Eamon Dolan/Mariner Books, 2016.

[21] Hagiu, Andrei. (2009). **Multi-Sided Platforms: From Microfoundations to Design and Expansion Strategies**. SSRN Electronic Journal.

[22] Gorkem Ozer, Edward Anderson. **Innovation and breaching strategies in multi-sided platform markets: Insights from a simulation study**. In Traci Carte, Armin Heinzl, Cathy Urquhart, editors, Proceedings of the International Conference on Information Systems - Exploring the Information Frontier, ICIS 2015, Fort Worth, Texas, USA, December 13-16, 2015. Association for Information Systems, 2015.

[23] Charith Perera. **Sensing as a Service (S2aaS): Buying and Selling IoT Data**. *IEEE Internet of Things eNewsletters*, November Issue, 2016; IEEE Internet of Things eNewsletters, November Issue, 2016

[24] Huang, Y. & Garcia-Molina, H. **Wireless Networks** (2004) 10: 643.

[25] L. Silva, M. Endler, and M. Roriz. **MR-UDP: Yet another reliable user datagram protocol, now for mobile nodes**. Technical Report MCC 06/13, Departamento de Informatica, PUC-Rio, 2013.

[26] Guo, B., Zhang, D., Wang, Z., Yu, Z., Zhou, X. **Opportunistic IoT: exploring the harmonious interaction between human and the internet of things**. J. Netw. Comput. Appl. 36, 1531–1539, 2013.

[27] Vastardis, N., Yangi, K. **Mobile social networks: architectures, Social properties, and key research challenges**. *IEEE Commun. Surveys Tutorials*, 1355–1371, 2012.

[28] Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos. **Sensing as a service model for smart cities supported by Internet of Things**. Transactions on Emerging Telecommunications Technologies, Volume 25, Issue 1, Pages 81–93, January 2014.

[29] Iansiti, Marco, and Karim R. Lakhani. "**The Truth about Blockchain.**" Harvard Business Review 95, no. 1 (January–February 2017): 118–127

[30] Nakamoto S. **Bitcoin: A peer-to-peer electronic cash system**. Consulted, 2008, 1: 2012.

[31] G. Zyskind, O. Nathan and A. '. Pentland, "**Decentralizing Privacy: Using Blockchain to Protect Personal Data**," *2015 IEEE Security and Privacy Workshops*, San Jose, CA, 2015, pp. 180-184.

[32] Holzschuher, Florian & Peinl, Rene, **Performance of graph query languages: Comparison of cypher, gremlin and native access in Neo4j**. ACM International Conference Proceeding Series. 2013.

[33] Uckelmann D, Harrison M, Michahelles F (2011) **Architecting the Internet of Things**. Springer, Berlin

[34] Y. Zhang and J. Wen, "**An IoT electric business model based on the protocol of bitcoin,**" 2015 *18th International Conference on Intelligence in Next Generation Networks*, Paris, 2015, pp. 184-191.

[35] Dominic Worner, Thomas von Bomhard. **When Your Sensor Earns Money: Exchanging Data for Cash with Bitcoin**. *UbiComp '14 Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, Pages 295-298, 2014.

[36] J. S. Lee and B. Hoh. **Sell your experiences: a market mechanism based incentive for participatory sensing**. In *2010 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 60–68, March 2010.

[37] N. C. Luong, D. T. Hoang, P. Wang, D. Niyato, D. I. Kim and Z. Han, "**Data Collection and Wireless Communication in Internet of Things (IoT) Using Economic Analysis and Pricing Models: A Survey,**" in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2546-2590, Fourthquarter 2016.

[38] Wei Wang, Fang Yao, Suparna De, Klaus Moessner, Zhili Sun, **A ranking method for sensor services based on estimation of service access cost**, Information Sciences, Volume 319, 2015, Pages 1-17, ISSN 0020-0255

[39] Zachariah, Thomas & Klugman, Noah & Campbell, Bradford & Adkins, Joshua & Jackson, Neal & Dutta, Prabal. (2015). **The Internet of Things Has a Gateway Problem**. 27-32.

[40] Karagiorgou, Sophia & Stamoulis, Georgios & Kikiras, Panagiotis. (2012). **Enabling QoS in the Internet of Things**. CTRQ 2012, The Fifth.

[41] R. Duan, X. Chen and T. Xing, "**A QoS Architecture for IOT**," 2011 *International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, Dalian, 2011, pp. 717-720.