



UNIVERSIDADE FEDERAL DA BAHIA

TRABALHO DE GRADUAÇÃO

**Deteccção inteligente de efeitos colaterais indesejáveis na
Internet das coisas - um estudo de caso no Home Network
System**

Heron Sanches Gonçalves Pires Ferreira

Programa de Graduação em Ciência da Computação

Salvador
31 de outubro de 2016

HERON SANCHES GONÇALVES PIRES FERREIRA

**DETECÇÃO INTELIGENTE DE EFEITOS COLATERAIS
INDESEJÁVEIS NA INTERNET DAS COISAS - UM ESTUDO DE
CASO NO HOME NETWORK SYSTEM**

Este Trabalho de Graduação foi apresentado ao Programa de Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Daniela Barreiro Claro

Salvador
31 de outubro de 2016

Ficha catalográfica.

Ferreira, Heron Sanches Gonçalves Pires

Detecção inteligente de efeitos colaterais indesejáveis na Internet das coisas - um estudo de caso no Home Network System/ Heron Sanches Gonçalves Pires Ferreira– Salvador, 31 de outubro de 2016.

22p.: il.

Orientadora: Profa. Dra. Daniela Barreiro Claro.
Monografia (Graduação)– UNIVERSIDADE FEDERAL DA BAHIA, INSTITUTO DE MATEMÁTICA, 31 de outubro de 2016.

“1. Efeitos Colaterais. 2. Internet das Coisas. 3. Home Network System. 4. Serviço WEB. 5.Dispositivos. 6.REST. 7.Redes Neurais. 8.Multilayer Perceptron”.

I. Claro, Daniela Barreiro. II. UNIVERSIDADE FEDERAL DA BAHIA. INSTITUTO DE MATEMÁTICA. III Título.

CDD 005.13307

TERMO DE APROVAÇÃO

HERON SANCHES GONÇALVES PIRES FERREIRA

DETECÇÃO INTELIGENTE DE EFEITOS COLATERAIS INDESEJÁVEIS NA INTERNET DAS COISAS - UM ESTUDO DE CASO NO HOME NETWORK SYSTEM

Este Trabalho de Graduação foi julgado adequado à obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Programa de Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 31 de outubro de 2016

Profa. Dra. Daniela Barreiro Claro
Universidade Federal da Bahia

Prof. Dr. Cássio Vinicius Serafim Prazeres
Universidade Federal da Bahia

Prof. Dr.
Universidade Federal da Bahia

TODO

AGRADECIMENTOS

O que sabemos é uma gota, o que ignoramos é um oceano.
—ISAAC NEWTON (1687)

RESUMO

Palavras-chave: TODO

ABSTRACT

Keywords: TODO

SUMÁRIO

Capítulo 1—Introdução	1
Capítulo 2—Fundamentação Teórica	2
2.1 Internet das coisas	2
2.1.1 Serviços Web	4
2.1.2 SOAP	8
2.1.3 Serviços Web RESTFul	11
2.1.4 Dispositivos como serviços Web RESTFul	15
2.2 Home network system	17
2.3 Aprendizado supervisionado	17
2.3.1 Redes neurais	17
2.3.2 Métodos avaliativos	17
2.4 Interação de características	17
2.4.1 Efeitos colaterais desejáveis	18
2.4.2 Efeitos colaterais indesejáveis	18
Capítulo 3—Experimento	19
Capítulo 4—Conclusão	20

LISTA DE FIGURAS

2.1	Utilização de diferentes serviços Web para prover o serviço de pedido de um E-Commerce.(Papazoglou, 2008)	5
2.2	Modelo de serviço Web baseado em SOAP.(Belqasmi et al., 2011)	7
2.3	SOAP no modelo Modelo de serviço Web baseado em SOAP.(Papazoglou,)	8
2.4	Comunicação dos serviços Web e mensagem SOAP.(Papazoglou, 2008) . .	9
2.5	Estrutura da mensagem SOAP.(Papazoglou, 2008)	10
2.6	Estrutura da mensagem SOAP com declaração do atributo encodingStyle.(Papazoglou, 2008)	10
2.7	Estrutura da mensagem SOAP, Header.(Papazoglou, 2008)	11
2.8	a)Estilo SOAP RPC <Body>. b) Estilo SOAP RPC <Body>- resposta(Papazoglou, 2008)	12
2.9	Exemplo de um <Body>no estilo documento SOAP.(Papazoglou, 2008) .	13
2.10	Requisição SOAP no estilo RPC dentro do corpo da requisição HTTP.(Papazoglou, 2008)	14
2.11	Resposta SOAP no estilo RPC dentro do corpo da resposta HTTP.(Papazoglou, 2008)	14
2.12	Dispositivo sensor de obstaculo do elevador disponibilizado como serviço. Faixa de detecção compatível com a largura do elevador utilizado na maquete do experimento 3	15
2.13	a)Mensagem SOAP sobre HTTP. b)Mensagem utilizando REST sobre HTTP. Figura baseada em (Pautasso, 2014)	16
2.14	Resultados de perfomance entre duas aplicações com mesmas funcionalidades, mas um baseada em SOAP e outra em REST. Figura adaptada de (Belqasmi et al., 2012)	17

LISTA DE TABELAS

LISTA DE ALGORITMOS

Capítulo

1

Uma breve introdução sobre do que se trata esta monografia e a maneira como o texto está organizado.

INTRODUÇÃO

Este capítulo tem como objetivo fundamentar as bases necessárias dos campos de estudos utilizados nesta monografia.

FUNDAMENTAÇÃO TEÓRICA

2.1 INTERNET DAS COISAS

A Internet nesta última década tem contribuído de forma significativa na economia e sociedade, deixando como legado uma notável infraestrutura de rede de comunicação. O seu maior disseminador nesse período, vem sendo *World Wide Web* (WWW), o qual permite o compartilhamento de informação e mídia de forma global (Chandrakanth et al., 2014).

No âmbito da economia, por exemplo, o E-Commerce permitiu potencializar as vendas de produtos e serviços, com um faturamento estimado para o ano de 2016 de aproximadamente 56,8 bilhões de reais no Brasil, segundo *ecommercenews*¹. Além do benefício direto para sociedade provindo do E-Commerce, onde as pessoas podem realizar pesquisa de preços de serviços e produtos e, adquiri-los de forma confortável, sem precisar se deslocar até um ponto de venda, a Internet dispõe para a sociedade diversas outras oportunidades, como cursos a distância oferecidos por diversas universidades de todo o mundo, a exemplo dos cursos disponibilizados pela plataforma Coursera².

A Internet está se tornando cada vez mais persistente no cotidiano, devido, por exemplo, ao crescente número de usuários de dispositivos móveis, os quais possuem tecnologias de conexão com a Internet, as quais cada dia tornam-se mais acessíveis (presentes em locais que não tinham e, mais baratas) (Chandrakanth et al., 2014).

Em 2010 havia aproximadamente 1,5 bilhão de PCs conectados a Internet e mais que 1 bilhão de telefones móveis (Sundmaeker et al., 2010). Segundo Gartner³, 6,4 bilhões de coisas estarão conectadas até o final de 2016 e, em 2020 esse número atingirá cerca de 20,8 bilhões. A previsão de (Sundmaeker et al., 2010), a qual dizia que a denominada

¹<https://ecommercenews.com.br/noticias/pesquisas-noticias/e-commerce-brasileiro-deve-crescer-18-e-faturar-r-568-bilhoes-em-2016>

²<https://pt.coursera.org/>

³<http://www.gartner.com/newsroom/id/3165317>

Internet dos PCs seria movida para o que se chama de Internet das Coisas fica então mais evidente neste atual cenário.

A ideia básica da *Internet of things (IoT)*, traduzido para o português como Internet das Coisas é a presença pervasiva de uma variedade de "coisas ou objetos", tais como RFID tags, sensores, telefones móveis, dentre outros. Os quais, através de esquemas de endereçamento único são capazes de interagir com os outros e cooperar com seus vizinhos para alcançar um objetivo em comum (Atzori et al., 2010). Outros exemplos de "coisas ou objetos" podem ser pessoas, geladeiras, televisores, veículos, roupas, medicações, livros, passaportes, contanto que possam ser identificadas unicamente e possam se comunicar com as outras coisas e/ou possam ser acessados remotamente por humanos.

Dentre as diversas definições de Iot pode-se citar duas, a primeira define de maneira mais geral, tanto a atual realidade, quanto a prospecção futura. Já a segunda especifica melhor como deve ser o cenário ideal da Internet das Coisas, pois já embuti explicitamente os conceitos de capacidades de autoconfiguração, interoperabilidade e interfaces inteligentes.

1. Segundo (Nanosystems, 2008), *Internet of Things* significa rede mundial de objetos unicamente endereçáveis e interconectados, seguindo os protocolos dos padrões de comunicação.
2. IoT é parte integrante da futura Internet e pode ser definida como uma infraestrutura de rede global dinâmica com capacidades de autoconfiguração baseada nos padrões e interoperabilidade dos protocolos de comunicação onde coisas físicas e virtuais têm identidade, atributos físicos, personalidade virtual, usam interfaces inteligentes e, são integradas dentro da rede de informações (Sundmaeker et al., 2010).

Diante deste cenário da IoT, pode-se citar exemplos de aplicações em diversos domínios, tais como, logística e transporte; cuidados com a saúde; ambiente inteligente (casa (seção 2.2), escritório); (Atzori et al., 2010) verificação de procedência alimentícia; dentre outros. Abaixo se tem uma breve descrição de duas aplicações, uma na área de cuidados com a saúde e outra na verificação de procedência de alimentos, respectivamente.

- Dispositivos implantáveis com capacidade de comunicação sem fio podem ser utilizados para armazenar registros sobre a saúde de um paciente em situações de risco e podem ser decisivos para salvar a vida do paciente. A capacidade de ter acesso a essas informações nessas circunstâncias fazem com que hospitais possam saber de imediato como tratar um paciente que esta a caminho. Esta possibilidade é especialmente útil para pessoas com diabetes, cancer, problemas de coração na artéria coronária, doenças do pulmão, assim como a pessoas com implantes médicos complexos, tais como, marcapassos, tubos, transplantes de órgãos e aqueles que podem ficar inconscientes e incapazes de comunicar-se durante uma operação. (Weber e Weber, 2010)
- Rastreabilidade de produtos alimentícios ajudam os usuários a verificar a origem de um produto, assim como informações de composição química, dentre outros.

Mas também previne de doenças não desejadas. Por exemplo, avisos atuais sobre a mercadoria em questão podem ser disponibilizados à medida que o produto sai da origem e vai passando para os outros níveis de consumo, desta forma os consumidores podem evitar o contágio de doenças como gripe aviária e, encefalopatia espongiforme bovina (EEB), mais conhecida como doença da vaca louca.(Weber e Weber, 2010)

Apesar da grande potencialidade da visão da Internet das Coisas, a qual gerou em 2015 um faturamento em torno de 130,33 bilhões de dolares americanos e tem prospeção de chegar até 883.55 bilhões em 2020⁴, ainda existem muitos desafios a serem vencidos, tais como segurança da informação, armazenamento e processamento de grande quantidade de dados, dentre outros. Adentrando um pouco em um dos desafios da IoT, o da disponibilidade de uma interface de comunicação (acesso aos serviços e informações dos dispositivos) e programação comum aos objetos, pode-se dizer que a falta desta padronização faz com que se torne oneroso o desenvolvimento de aplicações para o objeto, pois cada coisa possui suas próprias interfaces, logo para cada dispositivo um desenvolvimento a parte. Mais difícil ainda é prover uma única funcionalidade ou serviço com a composição dos diversos objetos. Para diminuir a dificuldade deste cenário, pode-se disponibilizar os dispositivos como serviços WEB (seção 2.1.4), assim sendo pode-se utilizar os protocolos WEB como linguagem comum de integração dos dispositivos a Internet.(Franca et al., 2011)

2.1.1 Serviços Web

Segundo(Dustdar e Schreiner, 2005), um serviço Web é um sistema identificado por uma URL⁵, no qual suas interfaces públicas são definidas e descritas usando XML⁶ e, suas definições podem ser descobertas por outros sistemas. Sistemas então podem interagir com os serviços Web utilizando suas definições e descrições. Através destas utiliza-se mensagens XML que são transmitidas seguindo os protocolos padrão (definidos por W3C⁷) da Internet para que haja tal interação.

Serviços Web surgiram tendo como principal foco o reuso de aplicações existentes (dentre as quais incluíam código fonte legado) para que pudessem se integrar de forma leve com outras aplicações, geralmente essa integração tinha como referência o desejo de novas formas de compartilhamento dos serviços ao longo das diversas linhas do negócio ou entre parceiros. Para melhor entender o proposito dos serviços Web, considere, como exemplo, uma companhia de seguros que decidiu oferecer um serviço de cotação on-line. Então ao invés de desenvolver toda a aplicação do inicio, a empresa utiliza-se do serviço Web de outra empresa(especializada em cálculos de seguro). Este serviço pode, por exemplo,

⁴<http://www.marketsandmarkets.com/Market-Reports/iot-application-technology-market-258239167.html>

⁵Acrônimo para Uniform Resource Locator e é uma referência (um endereço) para um recurso na Internet.(Oracle,)

⁶Extensible Markup Language (XML) é uma simples e flexível linguagem de marcação utilizada para codificar documentos através de regras produzidas por humanos, as quais podem ser manipuladas (compreendidas) por máquinas (W3C, a)(W3C, b)

⁷<https://www.w3.org/>

oferecer um formulário de cotação e, após o cliente informar os dados necessários, o serviço Web apresentará uma cotação baseado nos dados informados. Além disso, caso o cliente escolher comprar tal seguro, pode-se ainda utilizar de outro serviço Web, oferecido por uma outra empresa, o qual irá processar o pagamento de acordo com os dados passados para ele e então retorna o resultado do processamento para o cliente e para a empresa original (que desejou construir o serviço de cotação on-line) (Papazoglou, 2008). O cenário da Figura 2.1 demonstra um outro exemplo de uso de diferentes serviços para prover um único serviço, o de um pedido de uma ou mais mercadorias, ambiente comum do E-Commerce.

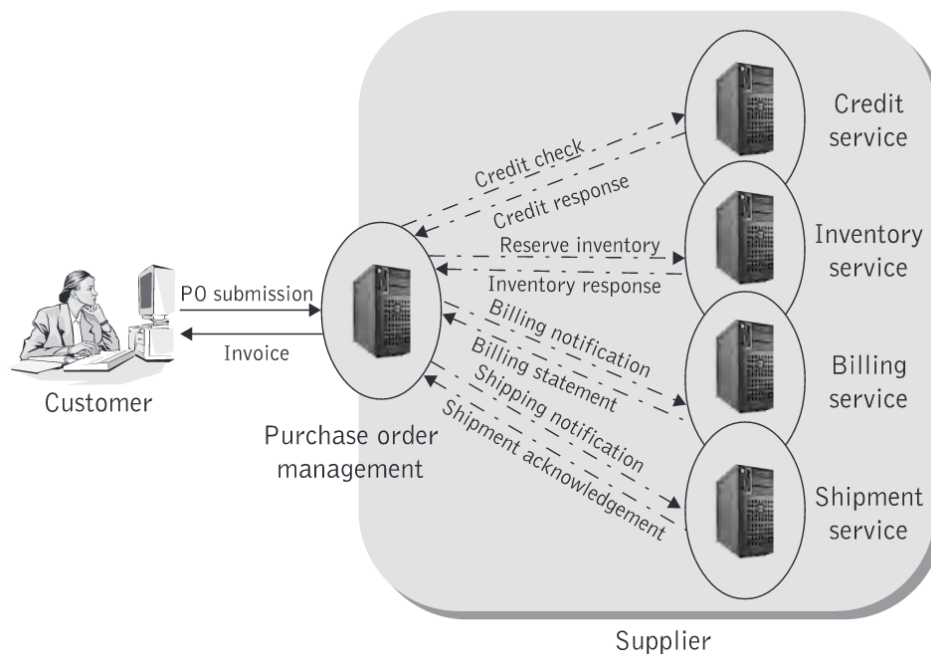


Figura 2.1: Utilização de diferentes serviços Web para prover o serviço de pedido de um E-Commerce. (Papazoglou, 2008)

O cenário do serviço (Figura 2.1) pode ser descrito da seguinte forma:

1. *Consumer* (Consumidor) escolhe seu(s) produto(s), informa dados do cartão de crédito e outras informações necessárias e, clica no botão comprar. Requisição é enviada ao *Purchase order management* (Gerenciador de pedidos de compra).
2. O gerenciador então verifica se o cartão do cliente tem crédito suficiente através da funcionalidade *Credit check* do serviço *Credit Service* e, o serviço dá uma resposta.
3. Gerenciador capta a resposta e, caso cliente tenha crédito suficiente, inicia o processo de reserva do pedido através da funcionalidade *Reserve inventory* do serviço *Inventory service*.
4. Gerenciador capta resposta *Inventory response*, caso o pedido tenha sido reservado com êxito, o gerenciador utiliza o *Billing service* para processar o pagamento.

5. *Billing service* retorna o estado do pagamento, se pagamento foi processado com êxito o gerenciador utiliza agora a funcionalidade *Shipping notification* do serviço *Shipment service* para prover o serviço de entrega.
6. Gerenciador recebe a resposta *Shipment acknowledgement* e informa ao cliente (*Invoice*) a data prevista para entrega do(s) produto(s) do cliente.

No cenário apresentado anteriormente há uma ressalva que deve ser feita, a qual dita uma das características de um serviço Web (serviço assíncrono ou síncrono, descrito logo a seguir), no item 5, por exemplo, o pagamento pode demorar mais de um dia para concluir seu processamento, então o cliente poderia receber um retorno do sistema do tipo, pagamento aguardando processamento, ou seja, a resposta do processamento do pagamento(realizado ou não realizado), poderia vir através de um envio de email ou SMS, somente depois de muito tempo.

Serviços Web possuem algumas características que se fazem importantes saber:

1. Podem ser ditos *stateless* (sem estado) ou *stateful* (com estado). Se um serviço pode ser invocado repetidamente sem ter que manter um contexto ou estado este é denominado *stateless*. Como exemplo de serviço *stateless* pode-se citar a recuperação de informação de um sensor de temperatura, pois não precisa de nenhum tipo de memória para manter o que aconteceu durante as requisições ao serviço. Já os serviços que precisam manter seus contextos preservados de uma invocação para a próxima requisição, estes são *stateful*. Por exemplo, um usuário quando chama um serviço (Levar compras) de um *textitHome Network Service* (ver seção 2.2), este então começa sua execução: um veículo(um serviço) leva a carga até uma determinada área próxima ao elevador(um serviço) e, uma garra mecânica (um serviço) pega a carga e coloca dentro do elevador. Observe que o serviço Levar compras é a composição de diferentes serviços e não pode ser chamado e executado diversas vezes continuamente sem a necessidade de manter seu contexto entre os diferentes serviços envolvidos, pois o ambiente não tem mais de um carro, mais de uma garra e elevador. Então a menos que este já tenha terminado toda a sua execução, poderia ser chamado e executado novamente. Por tal motivo este serviço mantém um estado (em execução ou sem trabalho) o qual depende dos estados dos serviços envolvidos.(Papazoglou, 2008)
2. Podem ser ditos com baixo nível de acoplamento (*loose coupling*) quando os serviços interagem uns com os outros e utilizam-se as tecnologias padrões da Internet, permitindo dessa forma construir pontes entre sistemas que de outra forma exigiriam grandes esforços de desenvolvimento de software. O termo acoplamento indica o nível de dependência entre serviços.(Papazoglou, 2008)
3. Podem ser ditos síncronos e assíncronos. Quando síncrono, clientes realizam suas requisições e ficam sempre aguardando uma resposta. O cliente é dependente de tal resposta para continuar sua computação. Se uma operação for incapaz de ser completada, todo o resto da computação é impedida de continuar. Enquanto que

nos assíncronos o cliente não necessita aguardar uma resposta para continuar a execução de sua aplicação. Quando um cliente invoca um serviço assíncrono, o cliente normalmente envia um documento inteiro, como uma ordem de pagamento ou, uma lista de itens de um carrinho de compra juntamente com o tipo de pagamento e endereço de entrega. O serviço aceita o documento inteiro e processa-o, podendo retornar ou não uma mensagem de resultado. A resposta do serviço, se existir alguma, pode acontecer horas ou talvez dias depois. (Papazoglou, 2008)

Serviços Web possuem baixo nível de acoplamento e utilizam padrões para oferecer suas funcionalidades, pois tais serviços tem a intenção de prover comunicação a diferentes tipos de aplicações, que possivelmente são executadas em diferentes plataformas. A *Web Service Description Language*(WSDL) usa o formato XML para descrever os métodos oferecidos por um serviço Web, incluindo parâmetros de entrada e saída, tipos de dados e, protocolo de transporte utilizado (normalmente o HTTP). Além de informações referente ao provedor do serviço, tais como, endereço e contato da empresa desenvolvedora do serviço. O *Simple Object Access Protocol*(SOAP, seção 2.1.2) é utilizado para as trocas de mensagens (formatadas em XML) entre as entidades envolvidas no modelo de serviço Web citado(Dustdar e Schreiner, 2005), como pode ser visto na Figura 2.2 e explicado logo abaixo. Esse tipo de serviço Web é chamado de serviço Web baseado em SOAP(Belqasmi et al., 2011).

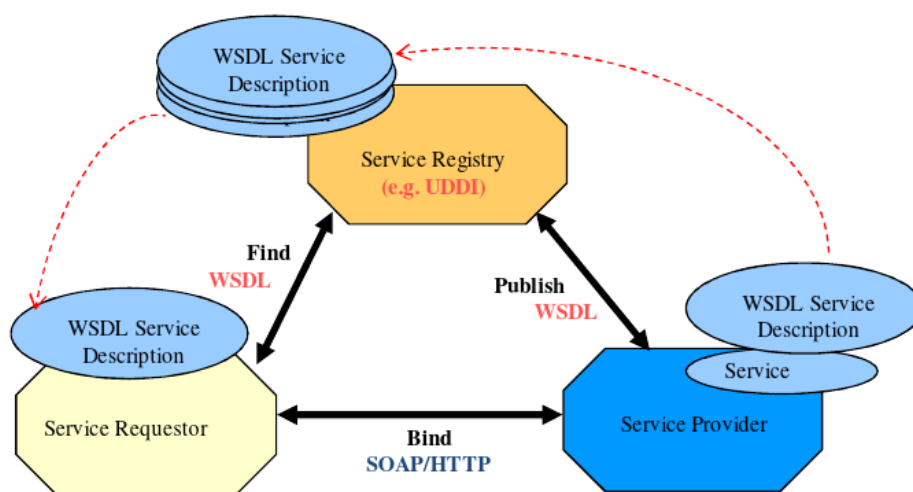


Figura 2.2: Modelo de serviço Web baseado em SOAP.(Belgasmi et al., 2011)

- Provedor (Service Provider) - cria e oferece o serviço Web, este precisa descrever o serviço em um formato padrão, neste caso WSDL e, publica-o (*Publish*) no Registro de serviço.
- Registro de serviço (Service Registry) - contém a descrição publicada pelo Provedor. O registro de serviço mais utilizado é o *Universal Description, Discovery and Integration* (UDDI), este em suas especificações define um conjunto de *Application*

Program Interfaces(APIs) tanto para publicação, quanto para descoberta(Belqasmi et al., 2011).

- Consumidor (Service Requestor) - obtém informações do registro, *Find*, e utiliza a descrição do serviço capturada para invocar (*Bind*) o serviço Web.

A descrição realizada da Figura 2.2 foi na verdade a descrição das 3 regras fundamentais do Service Oriented Architecture (SOA), que é um caminho lógico para projetar sistemas de software, os quais podem prover serviços tanto para o usuário final de aplicações quanto para outros serviços distribuídos na rede, através de interfaces que possam ser publicadas e descobertas.(Papazoglou, 2008)

Existe também outro tipo de abordagem de serviços Web, baseada nos princípios arquiteturais REST (ver seção 2.1.3).

2.1.2 SOAP

SOAP é um protocolo leve que foi desenvolvido com a intenção de prover troca de mensagens estruturadas em um ambiente descentralizado e distribuído. Este usa a tecnologia XML para definir um framework extensível de mensagens, o qual permite a construção de mensagens que podem ser trocadas através de uma variedade de protocolos subjacentes. O framework foi desenvolvido para ser independente de modelo de linguagem de programação ou qualquer outra implementação semântica(Gudgin et al.,). Diz-se que é um protocolo leve pelo fato de somente receber e enviar pacotes de protocolos de transportes (por exemplo, HTTP) e, processar as mensagens XML, quando construídos com os protocolos ORPC**(Papazoglou,). A figura 2.3 seguinte mostra como o SOAP atua no modelo Modelo de serviço Web baseado em SOAP (ver Figura 2.2). Nesta, o *SOAP-based middleware* converte as chamadas de procedimento de/para mensagens XML que são enviadas através do HTTP ou outro protocolo.

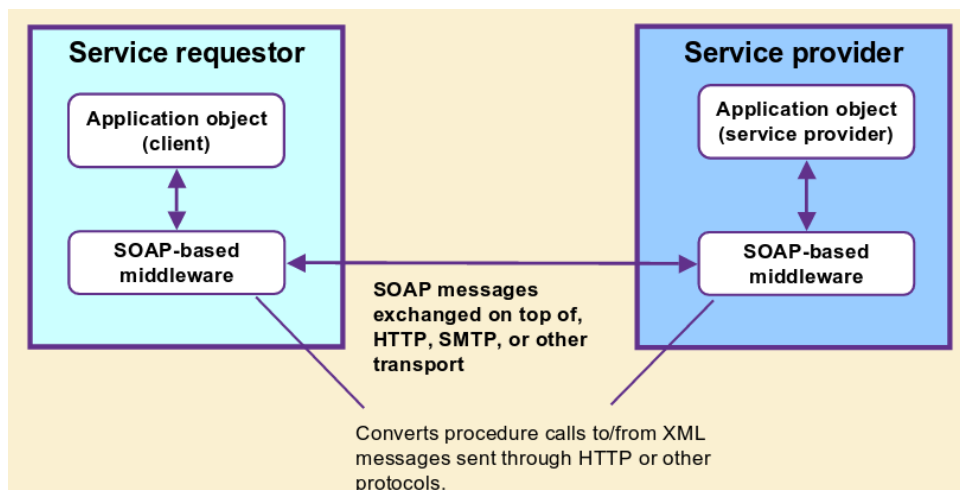


Figura 2.3: SOAP no modelo Modelo de serviço Web baseado em SOAP.(Papazoglou,)

Em outras palavras, SOAP é um protocolo de rede de aplicação utilizado para transferir mensagens entre instâncias de serviços descritas por interfaces WSDL (ver Figura 2.4).

Um documento XML SOAP é chamado de SOAP mensagem (ou SOAP envelope) e é usualmente transportado sobre outros protocolos de aplicação, mais frequentemente o HTTP. Mas existem outros protocolos de aplicação que podem ser utilizados: SMTP**, FTP**, RMI**. A mensagem SOAP, representada na Figura 2.4 é colocada no corpo da mensagem HTTP e então é encaminhada para seu destino. Na próxima camada, da hierarquia ilustrada na figura e no modelo OSI** (camada de transporte), a mensagem HTTP torna-se segmentos de bytes a serem enviados através do stream do TCP. Do outro lado (destinatário) um *HTTP listener* (fica a espera de novas mensagens) passa o corpo da mensagem HTTP para um processador de mensagens SOAP, o qual entende a sintaxe das mensagens e então é capaz de processá-las. (Papazoglou, 2008)

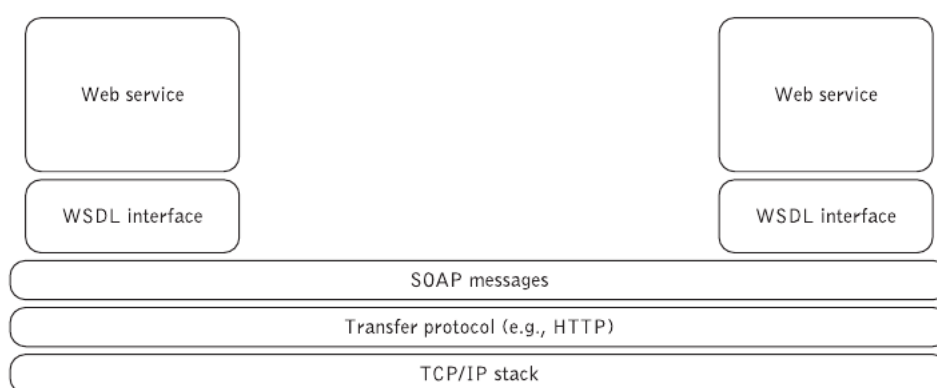


Figura 2.4: Comunicação dos serviços Web e mensagem SOAP. (Papazoglou, 2008)

Uma mensagem SOAP (ver Figura 2.5) é composta de um elemento <Envelope> (elemento raiz), o qual contém um <Header> (opcional) e um obrigatório <Body>. Caso um <Header> seja utilizado, este deve ser um filho imediato de <Envelope>, e preceder o <Body>. <Body> contém os dados principais da aplicação (dados arbitrários em um XML ou parâmetros para uma chamada de método). Uma mensagem SOAP pode ter uma declaração XML, <?xml version="1.0" encoding="UTF-8"?>, a qual indica a versão do XML usado e o formato da codificação. Para que a tag <env:Header ou Body ou Envelope> seja identificada como pertencente ao SOAP *namespace*, esta deve ser referenciada ao SOAP *namespace* URI, xmlns:env="http://www.w3.org/2003/05/soap-envelope". Se uma aplicação SOAP recebe uma mensagem que é baseada em outro *namespace*, isto gerará um erro. (Papazoglou, 2008)

O envelope SOAP pode especificar um conjunto de regras que serializam os dados XML da aplicação. Assim, tanto o Provedor quanto o Consumidor devem concordar com as regras de codificação (normalmente baixando o mesmo esquema XML que define eles). Para tal finalidade ambos podem utilizar-se do atributo global (pode ser declarado em qualquer nível do documento XML) *encodingStyle*, este define qual estilo de codificação deve ser utilizado no elemento no qual foi definido e em seus filhos (ver Figura 2.6). Com o intuito de favorecer a flexibilidade SOAP permite que as aplicações definam seu próprio estilo de codificação, mas na maioria dos casos as regras de codificação padrão são indicadas. (Papazoglou, 2008)

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope">

  <env:Header> <!-- optional -->
    <!-- header blocks go here . . . -->
  </env:Header>

  <env:Body>
    <!-- payload or Fault element goes here . . . -->
  </env:Body>
</env:Envelope>
```

Figura 2.5: Estrutura da mensagem SOAP.(Papazoglou, 2008)

```
<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/
    encoding/">
  ...
</env:Envelope>
```

Figura 2.6: Estrutura da mensagem SOAP com declaração do atributo encodingStyle.(Papazoglou, 2008)

O envelope SOAP pode conter no máximo um elemento <Header>, o qual contém todas as dicas que são relevantes para seus destinatários finais ou pontos de transporte intermediários. Também pode conter informação de onde o documento deve ser encaminhado, origem, e pode conter assinaturas digitais. O propósito do <Header> é encapsular extensões ao formato da mensagem sem combiná-las aos dados principais de carga da aplicação ou modificar a estrutura fundamental SOAP. Isto permite adição de especificações de características e funcionalidades, tais como segurança, transporte, referências a objetos, atributos QoS**, dentre outros, sem quebra das especificações da mensagem SOAP. Cada bloco dentro do <Header> deve ter seu próprio *namespace*, assim como já explanado anteriormente, o *namespace* permite que as aplicações SOAP identifiquem o bloco e os processe de acordo com as regras definidas no *namespace* em questão (ver Figura 2.7).(Papazoglou, 2008)

O <Body> pode ter um número arbitrário de filhos (chamados de entradas do <Body>), como também pode conter nenhum. Todas as entradas do <Body> que são filhos imediatos devem ter o *namespace*. Por padrão, o conteúdo do <Body> pode ser um XML qualquer sem especificação de regras. <Body> contém os dados principais da aplicação ou uma mensagem de erro, mas não os dois. Ou ainda pode ser vazio.(Papazoglou, 2008)

SOAP suporta dois tipos de estilo de comunicação, RPC (Remote Procedure Call) e documento (ou mensagem). Numa comunicação no estilo RPC/Literal, o cliente expressa sua requisição como uma chamada de método passando um conjunto de parâmetros. O


```

<env:Envelope
  xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  ...
  <env:Header>
    <tx:transaction-id
      xmlns:tx="http://www.transaction.com/transaction"
      env:mustUnderstand="true">
      512
    </tx:transaction-id>
    <notary:token xmlns:notary="http://www.notarization-
      services.com/token"
      env:mustUnderstand="true">
      GRAAL-5YF3
    </notary:token>
  </env:Header>
  ...
</env:Envelope>

```

Figura 2.7: Estrutura da mensagem SOAP, Header.(Papazoglou, 2008)

método então retorna uma resposta. Algumas regras devem ser seguidas e são exemplificadas na Figura 2.8:

- Uma URI deve identificar o endereço de destino onde o método sera chamado, que
- Deve ser passado o nome do método, \textless m:GetProductPriceResponse\textgreater
- Uma mensagem de resposta RPC deve conter um valor de retorno e quaisquer param

Em uma comunicação no estilo documento, SOAP é utilizado para trocar documentos contendo qualquer tipo de dados XML no <Body>, com o XML não codificado. No estilo de mensagem *Document/Literal*, o <Body> contém um elemento XML bem formado com dados da aplicação de qualquer tipo. O ambiente de execução SOAP aceita o elemento <Body> da maneira com que chegou e trata-o sobre a aplicação a qual foi destinado. Pode ou não ter uma resposta associado com a mensagem. A Figura 2.9 é um exemplo de mensagem neste estilo, a qual exemplifica os dados de um pedido de compra. Observa-se que este tipo de envio de dados não é muito bem conduzido para o estilo RPC, como o demonstrado na Figura 2.8. Em vez disso, a aplicação almeja transferir os dados da aplicação de uma única vez a um serviço para processamento futuro.(Papazoglou, 2008)

As requisições SOAP normalmente utilizam o protocolo de transporte HTTP. Quando utilizado as requisições SOAP são transportadas dentro do corpo do método POST do HTTP e suas respostas são retornadas nas respostas do HTTP (ver Figuras 2.10 e 2.11).

2.1.3 Serviços Web RESTFul

Segundo (Heffelfinger, 2014), Representational State Transfer (REST) é um estilo de arquitetura no qual serviços Web são vistos como recursos e podem ser identificados por

a)

```

<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
    <tx:Transaction-id
      xmlns:t="http://www.transaction.com/transactions"
      env:mustUnderstand='1'>
      512
    </tx:Transaction-id>
  </env:Header>
  <env:Body>
    <m:GetProductPrice>
      <product-id> 450R60P </product-id >
    </m:GetProductPrice >
  </env:Body>
</env:Envelope>

```

b)

```

<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
    <!-- Optional context information -->
  </env:Header>
  <env:Body>
    <m:GetProductPriceResponse>
      <product-price> 134.32 </product-price>
    </m:GetProductPriceResponse>
  </env:Body>
</env:Envelope>

```

Figura 2.8: a) Estilo SOAP RPC <Body>. b) Estilo SOAP RPC <Body>- resposta (Papazoglou, 2008)

Uniform Resources Identifiers (URIs). Serviços Web que são desenvolvidos usando REST são conhecidos como RESTful web services.

Os principais princípios do estilo arquitetural REST são: endereçamento global através de identificação dos recursos, interface uniforme compartilhada por todos os recursos, interações *stateless* entre os serviços, mensagens auto-descritivas e *hypermedia* como um mecanismo para descoberta descentralizada de recursos por referência (Pautasso, 2014). Na lista abaixo têm-se uma explanação à respeito dos princípios citados.

1. Identificação dos recursos - todos os recursos que são publicados por um serviço Web deveriam ser disponibilizados por um único e estável identificador global (Pautasso, 2014). A exemplo das URIs (Franca et al., 2011).
2. Interface uniforme - todos os recursos interagem através de uma interface uniforme, a qual prover um conjunto de métodos pequeno, genérico e funcionalmente suficiente para suportar todas as possíveis interações entre os serviços. Cada método tem um semântica bem definida em relação aos efeitos que causará no estado do recurso. No contexto da Web e do protocolo HTTP que é utilizado, “Interface uniforme” pode ser alcançado com os seus métodos (GET, PUT, DELETE, POST, HEAD, OPTIONS, dentre outros) os quais podem ser aplicados para todos os identificadores dos recursos Web (URIs). (Pautasso, 2014)
3. Interações *stateless* - os serviços não podem estabelecer sessões permanentes entre

```

<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope">

  <env:Header>
    <tx:Transaction-id
      xmlns:t="http://www.transaction.com/transactions"
      env:mustUnderstand='1'>
      512
    </tx:Transaction-id>
  </env:Header>
  <env:Body>
    <po:PurchaseOrder orderDate="2004-12-02"
      xmlns:m="http://www.plastics_supply.com/POs">
      <po:from>
        <po:accountName> RightPlastics </po:accountName>
        <po:accountNumber> PSC-0343-02 </po:accountNumber>
      </po:from>
      <po:to>
        <po:supplierName> Plastic Supplies Inc.
          </po:supplierName>
        <po:supplierAddress> Yara Valley Melbourne
          </po:supplierAddress>
      </po:to>
      <po:product>
        <po:product-name> injection molder </po:product-name>
        <po:product-model> G-100T </po:product-model>
        <po:quantity> 2 </po:quantity>
      </po:product>
    </ po:PurchaseOrder >
  </env:Body>
</env:Envelope>

```

Figura 2.9: Exemplo de um <Body>no estilo documento SOAP.(Papazoglou, 2008)

eles. Isto assegura que as requisições para um recurso sejam independentes entre si. No final de cada interação, não há estados compartilhados entre clientes e servidores. Requisições podem resultar em uma mudança de estado do recurso, mas o novo estado é imediatamente visível para todos clientes.(Pautasso, 2014)

4. Mensagens auto-descritivas - Serviços interagem através de requisição e mensagem de resposta, que contem tanto os dados (representações dos recursos) e correspondente *meta-data*. As representações podem variar de acordo com o contexto do cliente, interesses e habilidades. Por exemplo, um cliente *mobile* pode obter uma representação do recurso que exige pouco consumo de banda de dados. Da mesma forma, um navegador pode requisitar a representação de uma página Web em uma linguagem específica, de acordo com suas preferências. Esta característica aumenta de maneira significativa o grau de interoperabilidade, pois um cliente pode dinamicamente negociar o mais apropriado formato de representação com o recurso (também conhecido como *media type*), ao invés de ser forçado a sempre trabalhar com uma determinada representação do recurso. As requisições e mensagens de respostas também devem conter explicitamente *meta-data* sobre sua representação, desta maneira os serviços não precisam assumir algum tipo de acordo de sobre-

```

POST /Purchase Order HTTP/1.1
Host: http://www.plastics_supply.com  <! - Service provider -- >
Content-Type:application/soap+xml;
charset = "utf-8"
Content-Length: nnnn

<?xml version="1.0" ?>
<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
    <tx:Transaction-id
      xmlns:t="http://www.transaction.com/transactions"
      env:mustUnderstand='1'>
      512
    </tx:Transaction-id>
  </env:Header>
  <env:Body>
    <m:GetProductPrice>
      <product-id> 450R60P </product-id >
    </m:GetProductPrice >
  </env:Body>
</env:Envelope>

```

Figura 2.10: Requisição SOAP no estilo RPC dentro do corpo da requisição HTTP.(Papazoglou, 2008)

```

HTTP/1.1 200 OK
Content-Type:application/soap+xml;
charset = "utf-8"
Content-Length: nnnn

<?xml version="1.0" ?>
<env:Envelope
  xmlns:SOAP="http://www.w3.org/2003/05/soap-envelope"
  xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
    <!--! - Optional context information -->
  </env:Header>
  <env:Body>
    <m:GetProductPriceResponse>
      <product-price> 134.32 </product-price>
    </m:GetProductPriceResponse>
  </env:Body>
</env:Envelope>

```

Figura 2.11: Resposta SOAP no estilo RPC dentro do corpo da resposta HTTP.(Papazoglou, 2008)

carga no sentido de como tal representação seria analisada, processada e entendida.(Pautasso, 2014)

5. *Hypermedia* - Recursos podem ser relacionados entre si. *Hypermedia* embute referências

a outros recursos relacionados ou dentro de suas representações ou em sua correspondente *metadata*. Clientes então podem descobrir os *hyper-links* dos recursos relacionados ao processar suas representações e escolher seguir o link. Como exemplificado em (Franca et al., 2011), um sistema de uma instituição que possui um recurso “*lista_cursos*” (*listatodososcursosdainstituição*)

2.1.4 Dispositivos como serviços Web RESTFul

Como abordado em 2.1 um dos desafios referente a visão da IoT é sua interoperabilidade e, como possível solução se pode disponibilizar os dispositivos como serviços Web RESTFul. Desta maneira os dispositivos podem interagir entre si e com outros sistemas na Web.

Adotando esse padrão, os dispositivos podem ter suas propriedades disponíveis através de qualquer navegador, sem a necessidade de instalação de nenhum programa ou driver adicional, como pode ser exemplificado na Figura 2.12. Além disto, mashups físicos⁸ podem ser construídos com muito menos esforço do que as existentes abordagens, quebrando drasticamente a barreira para o desenvolvimento de aplicações com dispositivos (Guinard e Trifa, 2009). Pelo fato das coisas serem disponibilizadas como serviços Web, ainda pode-se utilizar-se dos outros recursos disponíveis na Web, a exemplo de sistemas de cache, balanceamento de carga, indexação e, pesquisa (Franca et al., 2011). Assim então promovendo a visão da Internet das Coisas.



Figura 2.12: Dispositivo sensor de obstaculo do elevador disponibilizado como serviço. Faixa de detecção compatível com a largura do elevador utilizado na maquete do experimento 3

No cenário da IoT os princípios REST têm sido amplamente aplicados na integração dos dispositivos inteligentes a Web, pois esses princípios parecem ser mais adequados para dispositivos com poucos recursos de hardware (Franca et al., 2011). Para deixar claro os motivos da adoção do REST para disponibilização de dispositivos como serviços Web ao invés dos serviços Web baseados em SOAP, será realizada uma comparação a seguir.

Uma das diferenças entre o REST e WS-* SOAP está na forma como o protocolo HTTP é empregado. Com REST, o HTTP é utilizado para prover interfaces uniformes onde cada método (GET, PUT, DELETE, POST, HEAD, OPTIONS, dentre outros) tem

⁸Aplicativos criados a partir da composição de dados e serviços de dispositivos físicos com outros recursos Web.

um semântica bem definida em relação aos efeitos que causará no estado do recurso(Franca et al., 2011), enquanto que nos WS-* SOAP cada serviço tem seu próprio conjunto de operações explicitamente declarados dentro de um documento WSDL(Pautasso, 2014). O SOAP utiliza o HTTP como um protocolo de transporte (HTTP é um protocolo da camada de aplicação de rede). As mensagens SOAP são adicionadas ao corpo do HTTP. Nos WS-* SOAP o método POST do HTTP é utilizado na troca de mensagens entre cliente e servidor e, a informação sobre qual funcionalidade deve ser executada está presente na mensagem SOAP e não na requisição HTTP. Os serviços Web baseados em SOAP utilizam a Web como meio de troca de mensagens, enquanto os serviços Web RESTful são parte da Web, ou seja, é visto como um terreno comum para as aplicações(Franca et al., 2011).

O formato da mensagem SOAP ocasiona um maior consumo de banda devido ao tamanho da mensagem SOAP (ver Figura 2.13), um dos principais motivos para utilizar dispositivos como serviços Web RESTful, pois muitos dispositivos na IoT possuem baixa capacidade de processamento e de banda de rede. Além disso, este formato pré-definido de mensagem força o cliente a tratar sempre este tipo de mensagem, enquanto utilizando REST é possível oferecer diferentes tipos de representações (JSON, XML, dentre outros).(Franca et al., 2011)

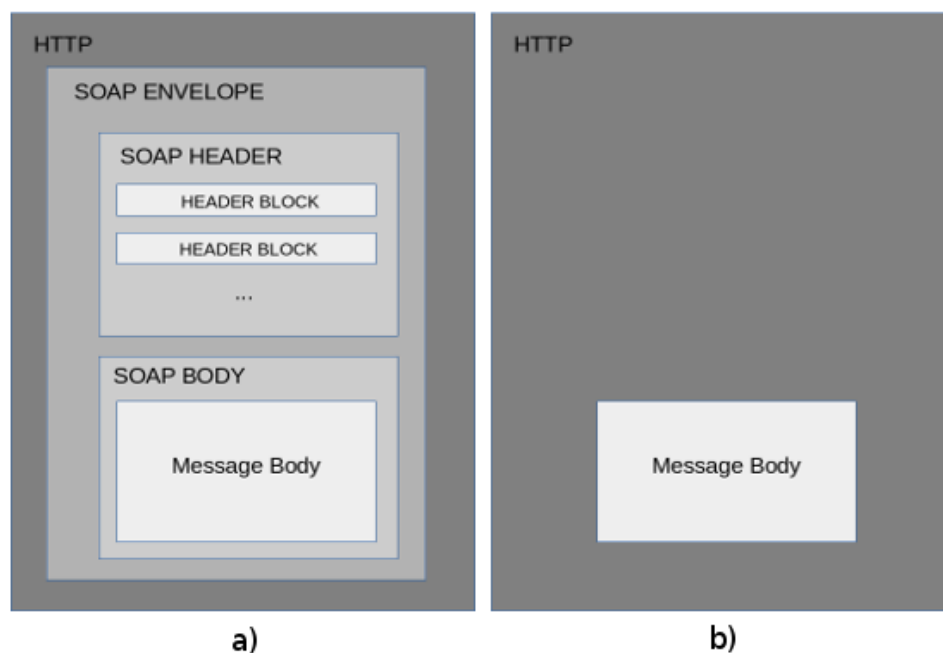


Figura 2.13: a) Mensagem SOAP sobre HTTP. b) Mensagem utilizando REST sobre HTTP. Figura baseada em (Pautasso, 2014)

Tratando o termo “fraco acoplamento” somente como a capacidade de fazer modificações no provedor de serviço sem afetar o cliente, os serviços Web RESTful são menos acoplados, pois as operações sobre os recursos não mudam, já que tais operações são sobre

os métodos do HTTP. Uma ressalva é que se as alterações forem feitas nos parâmetros passados nas mensagens, tanto o SOAP quanto o REST estarão no mesmo nível de acoplamento. (Franca et al., 2011)

Por fim, na Figura 2.14 é apresentado uma análise comparativa entre duas aplicações que oferecem as mesmas funcionalidades, sendo que uma foi construída utilizando serviços Web SOAP e a outra utilizando serviços Web baseado nos princípios arquiteturais REST. A aplicação em questão é de conferência multimídia e pode ser usada para criar uma nova conferência, adicionar ou remover um participante de uma conferência que esteja em progresso, atualizar o tipo de mídia (mudar de audio para vídeo) e deletar uma conferência. Ambas as aplicações foram desenvolvidas em um mesmo ambiente e foram publicadas em um mesmo servidor de aplicação. Vários cenários foram executados e performances foram medidas. Todas as medidas são de tempo (milissegundos) e cada resultado é a média de 10 experimentos. Observe que tanto na mesma máquina quanto em um ambiente distribuído a performance do REST é sempre melhor, sendo superior de 3 até 5 vezes em relação ao SOAP quando em ambiente distribuído.

	Baseado em SOAP			Baseado em REST		
Conferência Multimídia API	Demora no ambiente distribuído	Demora na mesma máquina	Demora de carga na rede	Demora no ambiente distribuído	Demora na mesma máquina	Demora de carga na rede
Criar conferência	848.4	381.7	767	171.4	102.7	273
Obter informação de conferência	818.6	335.3	546	172.3	98.6	177
Adicionar participante	1325.3	334.2	578	368.8	103.3	200
Remover participante	1322.3	357	588	382.9	107.2	195
Obter participantes	787.1	342.7	615	167.8	104.8	195
Obter informação de um participante	766.2	346.7	619	169.8	105	204
Finalizar conferência	1508.4	341.4	500	556.6	105.3	204

Figura 2.14: Resultados de performance entre duas aplicações com mesmas funcionalidades, mas um baseada em SOAP e outra em REST. Figura adaptada de (Belqasmi et al., 2012)

2.2 HOME NETWORK SYSTEM

TODO....

2.3 APRENDIZADO SUPERVISIONADO

2.3.1 Redes neurais

2.3.2 Métodos avaliativos

(Witten e Frank, 2005) pag 183 - livro data mining - parte escrita -

2.4 INTERAÇÃO DE CARACTERÍSTICAS

TODO....origem...causas...etc...

2.4.1 Efeitos colaterais desejáveis

TODO....

2.4.2 Efeitos colaterais indesejáveis

Segundo (Weiss et al., 2007) o problema de efeitos colaterais indesejáveis vem sendo tratado como *the feature interaction problem*.

Em (Weiss et al.,)???talvez seja bom mudar esta referencia??? *feature interaction* é definido como a interação entre independentes características^{***}, as quais podem ser intencionais ou não intencionais, podendo levar a efeitos colaterais indesejáveis. Segundo (??) tais efeitos colaterais indesejáveis podem ser um estado inconsistente do sistema ou inconsistência de dados, tais como, um comportamento não esperado (não desejado), uma quebra de requisito de segurança, dentre outros.

//TODO

EXPERIMENTO

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. pag 215 manual_eka(radomizacao)

MOTIVO: escolher os melhores parametros para o modelo em questão, neste caso, rede neural.

machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for three reasons:

simplification of models to make them easier to interpret by researchers/users,[1] shorter training times, enhanced generalization by reducing overfitting[2](formally, reduction of variance[1])

Procedimentos utilizados - experimento: -gera uma plotMatriz dos dados do dataset -verifica quais dados poderiam ser melhor separados por uma curva(par a par)

-neste caso, visualmente são esse abaixo: t_{num_width} $t_{num_diameter}$ (quem não tem width, tem diameter)

-testa com alguma combinação distinta dos parâmetros acima t_{num_width} , t_{num_mass} , $t_{num_diameter}$

-Verifica a generalização utilizando técnicas estatísticas, tais como, stratified-k-crossvalidation e/ou randomized-holdout

-escolhe o melhor modelo baseado no que você quer do modelo, neste caso o que tiver melhor recall, seguido de accuracy. POIS É MELHOR ERRAR DIZENDO QUE É, DO QUE ERRAR DIZENDO QUE NÃO É.

-gera um classificador com todo o dataset verificado na etapa anterior. -testa esse classificador com um dataset distinto do anterior. -DEMONSTRA OS RESULTADOS E COMPARA COM OS TESTES ANTERIORES.

Capítulo

4

Síntese da investigação e dos experimentos realizados nesta monografia

CONCLUSÃO

REFERÊNCIAS BIBLIOGRÁFICAS

- ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, p. 2787–2805, October 2010.
- BELQASMI, F.; FU, C.; GLITHO, R. Restful web services for service provisioning in next generation networks: A survey. *IEEE COMMUNICATIONS MAGAZINE, NETWORK and SERVICE MANAGEMENT*, 2011.
- BELQASMI, F. et al. Soap-based web services vs. restful web services for multimedia conferencing applications: A case study. *IEEE INTERNET COMPUTING*, 2012.
- CHANDRAKANTH, S. et al. Internet of things. *International Journal of Innovations and Advancement in Computer Science IJIACS*, v. 3, October 2014.
- DUSTDAR, S.; SCHREINER, W. A survey on web services composition. *Int. J. Web and Grid Services*, v. 1, p. 1–30, 2005.
- FRANCA, T. et al. Web das coisas: Conectando dispositivos físicos ao mundo digital. In: . Porto Alegre, RS: SBC: [s.n.], 2011. v. 1, p. 103–146.
- GUDGIN, M. et al. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. Disponível em: <https://www.w3.org/TR/soap12>.
- GUINARD, D.; TRIFA, V. Towards the web of things: Web mashups for embedded devices. In: *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web, International World Wide Web Conferences*. Madrid, Spain: [s.n.], 2009.
- HEFFELFINGER, D. *Java EE 7 with GlassFish 4 Application Server*. Third. [S.l.]: Packt Publishing Ltd, 2014.
- NANOSYSTEMS, i. C.-o. w. t. W. G. R. o. t. E. E. I. D. N. E. . R. I. G. M. . *Internet of Things in 2020: Roadmap for the Future*. 2008.
- ORACLE. *What Is a URL?* Disponível em: <https://docs.oracle.com/javase/tutorial/networking/urls/definition.html>.
- PAPAZOGLU, M. *Slides from Web Services: Principles and Technology*. Disponível em: https://www.cs.colorado.edu/~kena/classes/7818/f08/lectures/lecture_3_soap.pdf.
- PAPAZOGLU, M. *Web Services: Principles and Technology*. [S.l.]: Pearson, 2008. ISBN 978-0-321-15555-9.

PAUTASSO, C. Restful web services: Principles, patterns, emerging technologies. In: *Springer Science+Business Media*. New York: [s.n.], 2014.

SUNDMAEKER, H. et al. (Ed.). *Vision and Challenges for Realising the Internet of Things*. [S.l.]: European Union, 2010. ISBN 978-92-79-15088-3.

W3C. *Extensible Markup Language (XML)*. Disponível em: <https://www.w3.org/XML/>, <https://www.w3.org/XML/Schema>.

W3C. *XML Schema*. Disponível em: <https://www.w3.org/XML/Schema>.

WEBER, R. H.; WEBER, R. *Internet of Things - Legal Perspectives*. [S.l.]: Springer, 2010.

WEISS, M.; ESFANDIARI, B.; LUO, Y. Towards a classification of web service feature interactions. *Computer Networks*, v. 51, p. 359–381, 2007.

WEISS, M.; ORESHKIN, A.; ESFANDIARI, B. *Invocation Order Matters: Functional Feature Interactions of Web Services*.

WITTEN, I.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques*. second. [S.l.]: ELSEVIER, 2005. ISBN 0-12-088407-0.