LIVRO de MINICURSOS

SBrT'18

Campina Grande - PB

XXXVI Simpósio Brasileiro de Telecomunicações e Processamento de Sinais

editoraIFPB

LIVRO DE MINICURSOS

Organização

Paulo Ribeiro Lins Júnior (IFPB)
Edmar Candeia Gurjão (UFCG)
Ruan Delgado Gomes (IFPB)
Jerônimo Silva Rocha (IFPB)

Realização

Sociedade Brasileira de Telecomunicações – SBrT
Universidade Federal de Campina Grande – UFCG
Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB

João Pessoa, 2019.

Dados Internacionais de Catalogação na Publicação - CIP
Biblioteca Nilo Peçanha - IFPB, *campus* João Pessoa

L788 Livro de minicursos SBRT 2018 / organizadores, Paulo Ribeiro Lins Júnior ... [et al.]. – João Pessoa : IFPB, 2019.
234 p. : il.

PDF
ISBN: 978-85-5449-021-8

Textos originalmente apresentados no Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, 2018.
1. Telecomunicações. 2. Processamento de sinais. I. Lins Júnior, Paulo Ribeiro. II. Título.

CDU 621.391

Lucrecia Camilo de Lima
Bibliotecária
CRB 15/132

Organização do SBrT 2018

Coordenação Geral
Edmar Candeia Gurjão (UFCG)

Editoria de Publicações e Minicursos
Paulo Ribeiro Lins Júnior (IFPB)
José Ewerton Pombo de Farias (UFCG, Iecom)

Coordenação de Divulgação
Bartolomeu Uchoa (UFSC)

Coordenação Local do Evento
Jerônimo Silva Rocha (IFPB)
Luciana Veloso (UFCG)

Coordenação de Informática e Inscrições
Raissa Rocha (UFS, Iecom)
Gustavo Nóbrega Martins (IFPE, Iecom)
Ruan Delgado Gomes (IFPB)

Coordenação de Captação de Recursos
Leocarlos Bezerra da Silva Lima (UFCG)
Bruno Barbosa Albert (UFCG)

Coordenação Financeira
Marcelo Sampaio de Alencar (Iecom, UFCG, UFBA)
Orleans Martins (UFPB, Iecom)

Coordenação Técnica
Marcelo Sampaio de Alencar (Iecom, UFCG, UFBA)
Wamberto José Lira de Queiroz (Iecom, UFCG)
Waslon Terllizzie A. Lopes (Iecom, UFPB)

Prefácio

No XXXVI Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2018), realizado de 16 a 19 de setembro de 2018, na cidade de Campina Grande, Paraíba, tivemos oito propostas de minicursos aceitas para o evento. Como critérios para a avaliação, consideramos a qualidade técnica das propostas, a relevância do tema e a forma de abordagem do conteúdo.

Este livro apresenta, em seus capítulos, material elaborado pelos autores de cada um dos minicursos apresentados (exceto um), abordando com profundidade o assunto tratado no minicurso. Ressaltamos que os textos aqui publicados, bem como as informações neles contidas, são de completa responsabilidade de seus autores.

Nosso objetivo com a publicação desse livro é disponibilizar material de alta qualidade técnica, com temas relevantes para a área, que possam servir de referência bibliográfica para pesquisadores e educadores da área de telecomunicações, processamento de sinais e correlatas.

A qualidade desta obra se deve, fundamentalmente, ao esforço de cada um dos autores dos minicursos. Somos imensamente gratos a eles pelas horas de trabalho despendidas na elaboração da proposta e do material de instrução.

Agradecemos também ao Comitê Organizador do SBrT 2018, em especial à Coordenação Geral do evento, na figura do Prof. Edmar Candeia, por ter topado a ideia da construção desse livro e pelo apoio e orientação ao longo de todo o processo.

Por fim, desejamos que tanto os minicursos, quanto essa publicação, possam servir como sementes fomentadoras do interesse para novas pesquisas, novos trabalhos e da evolução de nosso campo de trabalho.

Paulo Ribeiro Lins Júnior
Editoria de Publicações e Minicursos

Sumário

CAPÍTULO

1 Teoria e Prática na Virtualização de Funções de Redes em Ambiente de Nuvem

Epaminondas A. de Sousa Junior (UFES), João H. G. M. Corrêa (UFES), Isabella de A. Ceravolo (UFES), Rodrigo L. Guimarães (UFES), Magnos Martinello (UFES), Moisés R. N. Ribeiro (UFES)

Introdução

O uso de plataformas de computação em nuvem tem o objetivo de tornar simples e flexível a implantação e gestão de serviços baseados na Internet, minimizando o *overhead* com a hospedagem de tais serviços. A virtualização de funções de rede (NFV - *Network Functions Virtualization*) surgiu com o intuito de possibilitar a substituição de equipamentos dedicados de funções de rede, como roteadores, *firewall*, etc, por funções desenvolvidas em software executando sobre hardware de propósito geral. Essa mudança de paradigma é habilitada pela computação em nuvem, com destaque para a plataforma Openstack.

O OpenStack é uma das mais proeminentes plataformas de computação em nuvem de código aberto existentes atualmente, capaz de gerenciar e prover recursos de processamento, rede e armazenamento [1]. Um de seus módulos, o Tacker [2], pode ser usado como habilitador de NFV, disponibilizando funcionalidades de gestão e orquestração de funções de rede virtualizadas. Dessa forma, é possível criar novos serviços de rede, cujo ciclo de vida (implantação, modificação, deleção) pode ser gerido de modo automatizado.

Este minicurso propõe introduzir os conceitos relacionados ao paradigma NFV no ambiente de computação em nuvem sob um ponto de vista teórico e prático. Na primeira parte do minicurso são discutidas as bases conceituais de funções de rede virtualizadas (VNFs) utilizando a plataforma OpenStack como gerenciador de recursos da nuvem. A segunda parte é constituída de exercícios práticos de implementação de VNFs em diferentes cenários. Mais especificamente, serão testados protótipos das funcionalidades de gerência,

escalabilidade em um cenário de alta disponibilidade e encadeamento de VNFs, motivando assim o desenvolvimento de novos serviços de rede no ambiente de computação em nuvem.

NFV

A virtualização de funções de rede (NFV) surgiu com o objetivo de traçar um novo caminho para o provisionamento de funções de redes por empresas de telecomunicações, consistindo primordialmente em substituir equipamentos de rede específicos (*network appliances*), tais como, roteadores, *firewall*, *switches*, etc, com forte acoplamento de recursos proprietários, por *software* e automação em *hardware* de propósito geral [3],[4].

NFV tem como finalidade tornar as redes mais simples e flexíveis minimizando a dependência de restrições de *hardware* com a virtualização de funções de redes específicas. Assim, como benefícios esperados com a adoção da tecnologia NFV podemos citar a redução de CAPEX e OPEX, a diminuição do tempo para um novo produto chegar ao mercado, a interoperabilidade, melhor escalabilidade e flexibilidade para a instanciação de novos serviços, e por fim, possibilitar um incentivo à inovação e o fortalecimento de plataformas abertas [5].

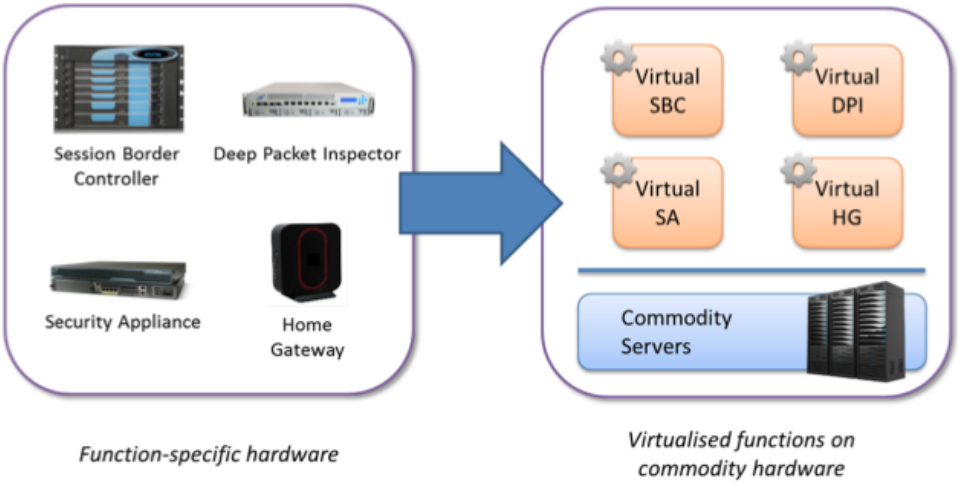


Figura 1.1 – Ideia do conceito de NFV [6]

A *European Telecommunications Standards Institute* (ETSI) definiu uma arquitetura base para guiar as empresas ao criarem produtos e protocolos baseados em NFV [7]. Esta arquitetura em questão é composta por vários módulos e camadas que descreverão como as funções de redes virtualizadas (VNFs) estarão dispostas levando em consideração, como serão instanciadas, suas conectividades, dados atribuídos, dependências, controle, etc [8].

A Figura 1.2 mostra a arquitetura padronizada NFV ETSI e destaca três blocos principais sendo eles: as funções de rede virtualizadas (VNFs), o gerenciador e orquestrador NFV (NFV *Management and Orchestration* – MANO) e a infraestrutura NFV (NFV *Infrastructure* – NFVI).

O bloco de funções de rede virtualizadas (VNFs) representa as implementações via software de funções de rede. Como exemplo de funções de rede, podemos citar: *firewall*,

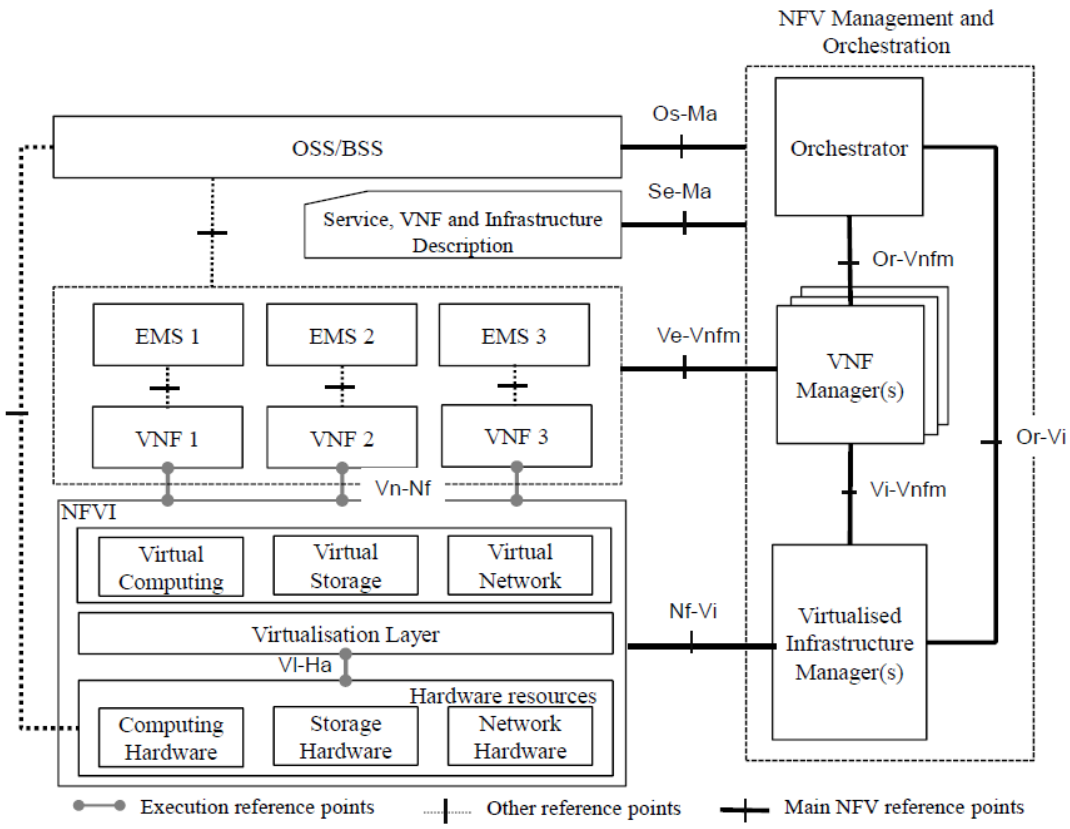


Figura 1.2 – Arquitetura de referência NFV ETSI

balanceador de carga, IDS, IPS, serviços de NAT, etc. Uma função de rede pode ser decomposta em uma ou mais máquinas virtuais. Cada uma dessas VMs associadas a uma VNF recebe o nome de *Virtualization Deployment Unit* (VDU) no modelo ETSI.

O bloco Infraestrutura NFV (NFVI) é constituído de todos os recursos computacionais disponíveis que servirão como ambiente para a instanciação, gerenciamento e execução das VNFs. A infraestrutura implementada pelo NFVI pode ainda estar espalhada em diferentes localizações geográficas. Possui três domínios, sendo eles o de computação, *hypervisor* e o de infraestrutura.

De forma geral, o domínio de computação representa todos os recursos de computação em que VNFs possam ser executadas, o domínio de *hypervisor* implementa uma camada virtual que tem como objetivo abstrair o *hardware* e implementar serviços para gerencias o ciclo de vida de VMs e, por fim, o domínio de infraestrutura de rede tem por objetivo prover a conectividade entre as VMs, utilizando várias técnicas para virtualização de recursos de rede.

O bloco de gerenciamento e orquestração de VNFs, conhecido como MANO, assume um conjunto de responsabilidade tais como, a alocação apropriada dos recursos disponibilizados pela infraestrutura NFV (NFVI), até a inteligência de gerencia e orquestração das VNFs, como por exemplo, a definição da forma como elas serão posicionadas e encadeadas na rede e a gerência de seus ciclos de vida. É composto por dois gerenciadores e um orquestrador sendo eles: *Virtualized Infrastructure Manager* (VIM),

VNF *Manager* (VNFM) e o NFV *Orchestrator* (NFVO).

O blodo VIM está encarregado de entregar alguns serviços no modelo NFV, dos quais podemos citar, principalmente, o gerenciamento do ciclo de vida de recursos virtuais em um domínio NFVI, isso significa criar, manter, modificar e remover VMs dentro de um NFVI; manter um inventário de máquinas virtuais associadas a recursos físicos; gerenciar o desempenho, bem como as falhas de recursos virtuais alocados na infraestrutura; manter APIs para expor recursos (físicos ou virtuais) a outros sistemas de gerenciamento etc.

A plataforma habilitadora de computação em nuvem OpenStack é considerada um exemplo de VIM, assumindo os papéis equivalentes dentro do contexto de nuvem. De modo geral, o conceito de IaaS para modelos de nuvem é bem similar ao conceito de VIM para o modelo de arquitetura NFV: uma plataforma capaz de gerenciar recursos em uma infraestrutura para prover soluções na forma de serviços virtualizados.

O gerenciador de VNFs é responsável pela gerencia do ciclo de vida de VNFs, como criação, atualização, remoção e monitoramento; gerência de falhas, configurações, desempenho e segurança; escalar o número de VNFs e também o número de VDUs de uma VNF; realizar funções como *health monitoring*, etc.

O bloco Orquestrador NFV (NFVO) possui a responsabilidade de orquestrar e gerenciar de forma ampla os serviços de rede fim-a-fim em NFV, possuindo uma visão global desses serviços na infraestrutura. O NFVO é dividido em duas grandes funções: orquestração de recursos e orquestração de serviços.

Em relação a orquestração de recursos, o NFVO atua coordenando, autorizando e liberando recursos de NFVI. No que diz respeito a orquestração de serviços, o NFVO atua gerenciando e orquestrando serviços de rede fim-a-fim utilizando VNFs disponíveis, podendo inclusive atuar gerenciando VNFs de diferentes VNFM, como por exemplo, a criação de um serviço de rede por meio de VNFs de diferentes fornecedoras. Além disso, também atua no gerenciamento de diferentes topologias de organização de VNFs, trabalhando com o conceito de VNF *Forwarding Graph*, em que esses serviços serão organizados em um grafo e coordenados pelo NFVO.

OpenStack

OpenStack é uma plataforma de computação em nuvem pública e privada, de código aberto, que permite controlar grandes conjuntos de recursos de computação, armazenamento e redes, pertencentes a um *datacenter*, tudo gerenciado através de uma interface dedicada que permite aos administradores controlar e provisionar recursos para múltiplos usuários. A plataforma surgiu inicialmente de uma parceria entre a Rackspace e a NASA, mas atualmente mais de duzentas companhias de *hardware*, *software* e serviços apoiam o seu desenvolvimento, que é mantido por uma comunidade global de desenvolvedores. Essa comunidade é chamada de Fundação OpenStack (*OpenStack Fundation)* [1].

Um dos principais objetivos da plataforma é construir um serviço de computação em nuvem que pudesse ser instalado em *hardware* padrão ou "customizado". Sobre essa camada de *hardware* atuam serviços compartilhados do OpenStack e os componentes responsáveis por controlar os grupos de computadores, armazenamento e recursos de rede. No topo desta estrutura está a camada de aplicações e administração de acesso, que é representada por uma interface *web*. Dessa forma, de acordo com a arquitetura padronizada na seção anterior, o OpenStack se enquadra no bloco de gerenciamento da infraestrutura virtualizada (VIM).

Uma das razões para que o OpenStack seja uma plataforma difundida e utilizada é que foi construída de forma modularizada, ou seja, cada função específica dentro do projeto é divida em um módulo separado. Dessa forma, o desenvolvimento de componentes podem variar, podendo ter grau de maturidade diferente para diversos módulos.

Dentre os componentes do OpenStack, alguns são considerados chaves para a implementação de um ambiente de nuvem com NFV, por serem responsáveis pela habilitação da tecnologia NFV e orquestração de recursos de processamento, armazenamento e rede. Os módulos chaves são: Nova (provisionamento de instancias de computação), Neutron (conectividade entre as redes virtuais), Keystone (gerência de acesso à nuvem), Glance (gerência de imagens), Horizon (painel de controle da nuvem via interface web), Ceilometer (coleta de dados para monitoramento), Tacker (orquestração e gerenciamento a tecnologia de redes virtualizadas). A Figura 1.3 traz alguns dos principais componentes que integram a plataforma OpenStack.

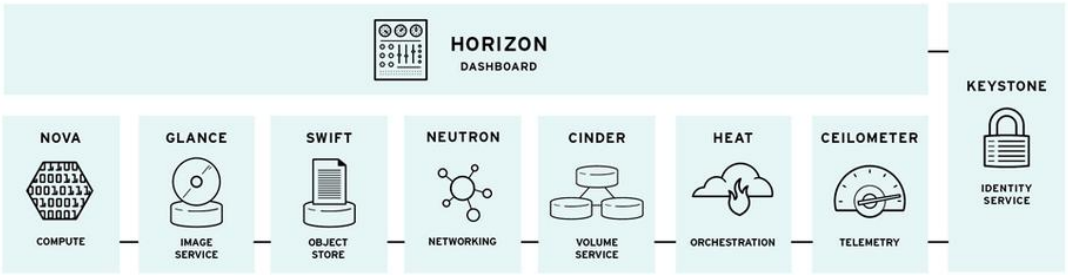


Figura 1.3 – Representação de alguns componentes da plataforma OpenStack [9]

O OpenStack oferece inúmeros cenários para sua implantação, que vão desde um cenário "tudo em um" (*All-in-one*), em que todos os componentes do OpenStack estão dispostos em um único servidor, até cenários de múltiplos nós (*Multi-node*) em que seus componentes são separados em vários nós na rede. No caso deste minicurso, a configuração utilizada é o DevStack, que é o OpenStack no cenário "tudo em um", pois todos os módulos necessários são instalados em um único servidor, na versão Ocata.

Módulo Tacker

O módulo Tacker é um projeto oficial da Fundação OpenStack que tem como principal objetivo implementar um gerenciador genérico de VNFs (VNFM) e um orquestrador NFV (NFVO), para instalar e operar serviços de rede e VNFs em uma plataforma habilitadora

de NFV, como o OpenStack [2]. É baseado no ETSI MANO *Architectural Framework*, como apresentado na Figura 1.2, e provê uma pilha funcional para orquestrar serviços de rede fim-a-fim usando VNFs [10].

O Tacker, como gerenciador de VNFs, é responsável pela instanciação, atualização e remoção de VNFs além de prover serviços de monitoramento. Como orquestrador NFV, tem a responsabilidade de orquestrar e gerenciar de forma ampla os serviços de rede fim-a-fim, tendo uma visão global desses serviços da infraestrutura.

O VNFM, implementado pelo Tacker, trabalha com o conceito de catálogos de VNFs conforme mostrado na Figura 1.4. O catálogo funciona como um repositório de diferentes tipos de VNFs que estarão prontas para serem instanciadas pelo gerenciador e que são armazenadas no banco de dados do Tacker. O catálogo é formado por descritores de funções de rede virtualizadas (VNFD), que são descritas em linguagem padronizada TOSCA.

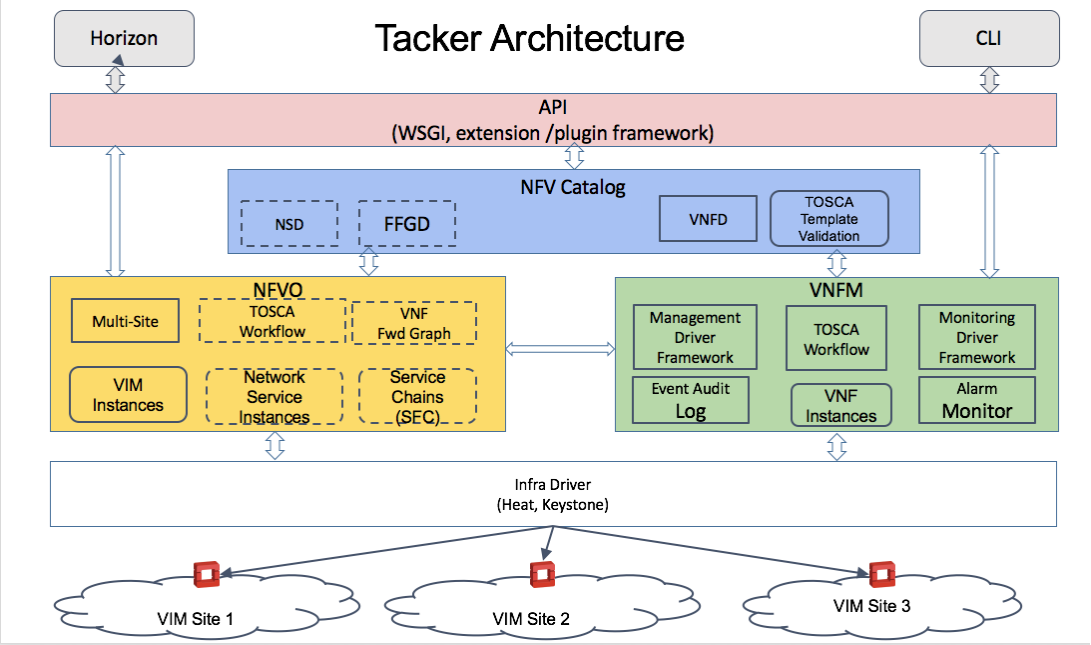


Figura 1.4 – Arquitetura do Tacker OpenStack [11]

O Tacker oferece a possibilidade de gerenciamento do ciclo de vida de VNFs, como criação, modificação e deleção, de dois modos: por linha de comandos, através da tacker-api, ou por uma interface amigável devidamente integrada à *dashboard* do OpenStack.

Cada VNF instanciada pode estar associada a uma ou mais VMs na infraestrutura virtual gerenciada pelo VIM. No modelo NFV ETSI essas VMs são chamadas de *Virtual Deployment Units* (VDUs). Portanto, caso o utilizador deseje instanciar uma VNF, deverá descrever suas VDUs e como essas VDUs estarão interconectadas em pontos de conexões conhecidos como *Connection Points* (CP). Caso o utilizador queira remover uma VNF, todas as VDUs associadas a VNF que será removida deverão ser também desalocadas de seus recursos. Além do mais, o utilizador poderá ter a opção de atribuir uma VNF a uma ou mais redes virtuais disponíveis na NFVI, bem como passar parâmetros para as VNFs no ato de sua criação, como *scripts* de inicialização ou pacote de recursos (*flavors*) que ela consumirá.

Para a descrição de VNFs, o Tacker utiliza uma linguagem padronizada pelo *Organization for the Advancement of Structured Information Standards* (OASIS) conhecida como TOSCA [12]. A linguagem TOSCA é conhecida por ser uma linguagem de descrição de uso de recursos para o contexto de computação em nuvem, mas também vem sendo utilizada no modelo NFV [13]. Dessa forma, do ponto de vista de alocação de recursos para VNFs, as descrições feitas em TOSCA precisam ser traduzidas para a forma como o OpenStack trata seus descritores de recursos, e neste caso, o serviço conhecido como *Heat* é o principal responsável por prover a automatização na alocação de recursos por meio de *scripts* conhecidos como *Heat Orchestration Templates* (HOT).

Para resolver essa diferença de padrões, o OpenStack ja introduz desde 2015 o projeto Tosca-Parser para traduzir templates em TOSCA para templates reconhecidos pelo *Heat* [14] . O processo pode ser observado na Figura 1.5, que conta com um elemento intermediário, o *heat-translator*, para atuar na tradução de TOSCA para HOT.

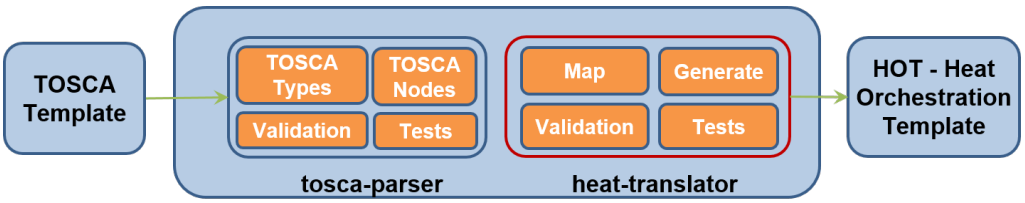


Figura 1.5 – Processo de tradução de TOSCA para HOT [15]

Parte Prática

Nesta seção serão apresentados como um usuário pode interagir com a plataforma OpenStack; como identificar os campos necessários de um *script* TOSCA; como modificar para atender as necessidades individuais. Posteriormente, serão apresentados as funcionalidades básicas de ciclo de vida e monitoramento, inclusive com exemplos para serem executados na prática. Após essas funcionalidades básicas, será introduzido como utilizar a característica de escalabilidade que o Tacker oferece, compondo assim um serviço escalável dentro do OpenStack, inclusive com a parte prática. Por fim, serão compostos serviços por meio de encadeamento de funções de rede, discutindo e inserido os *scripts* que realizam este encadeamento.

Introdução prática ao OpenStack

Há duas maneiras de se interagir com o OpenStak: por meio da linha de comando (CLI) ou por meio da interface *web*, disponibilizada por um módulo específico dentro da plataforma, o Horizon. Normalmente utiliza-se a interação por meio do Horizon, pois facilita a administração e abstrai a integração com os outros módulos e funcionalidades dentro do OpenStack. Na Figura 1.6 tem um exemplo de login no Horizon, requisitando

um usuário e senha para realizar a administração do projeto. Para acessar o Horizon basta digitar o endereço IP da máquina que o OpenStack está instalado.

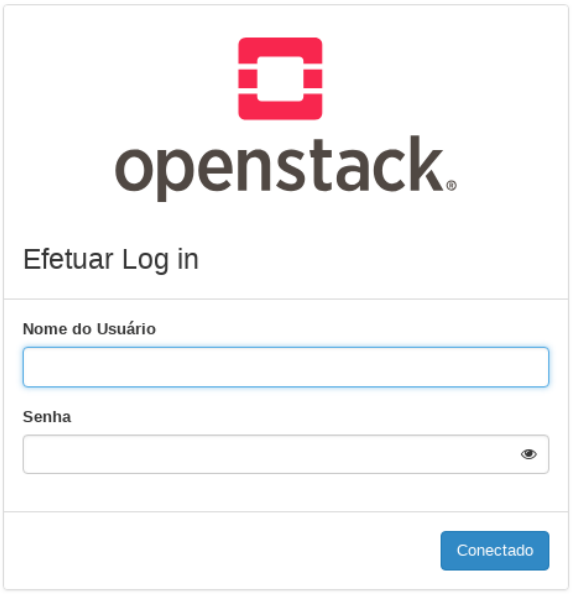


Figura 1.6 – Exemplo de tela de login no OpenStack

A título de testes e de apresentação das funcionalidades, todos os testes foram realizados e as imagens geradas com o usuário 'admin'. Após realizar o login na interface *web*, é apresentado um menu contendo as opções de projetos que aquele determinado usuário tem acesso. No lado esquerdo, como observado na Figura 1.7, tem acesso a vários contextos e abas dentro do OpenStack: Projeto, Admin, Identidade e NFV.

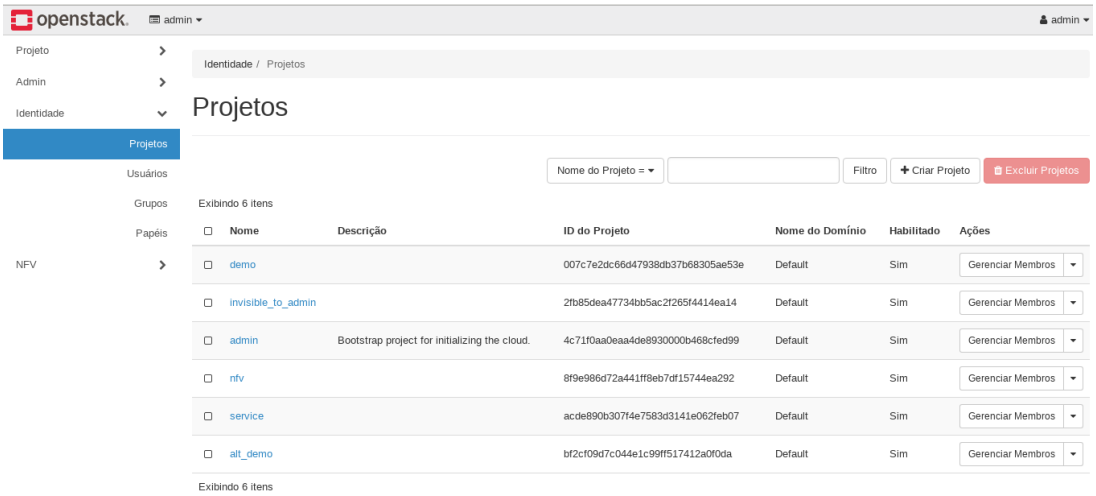


Figura 1.7 – Exemplo de tela do OpenStack

A aba 'Projeto' é responsável por toda a administração do projeto que o usuário está inserido: a Computação (responsável pelas máquinas virtuais, as imagens que poderão ser instanciadas, entre outras configurações mais avançadas); pode verificar também sobre a Rede no contexto do projeto (modificar a topologia de rede, inserindo novas redes, adicionando ou modificando roteadores, incluindo opções avançadas, como a definição de IPs Flutuantes e regras de segurança).

A aba 'Admin' está disponível apenas por que no exemplo foi utilizado o usuário 'admin', que é o administrador da nuvem. Nessa aba é possível fazer as configurações em relação a nuvem como um todo, inclusive o gerenciamento de hipervisores (utilização dos recursos de *hardware* físico). Nessa aba é possível fazer configurações que interfiram diretamente todos os projetos existentes na nuvem.

A aba 'Identidade', para um usuário comum, vai ser listados os projetos que esse usuário está inserido. Se o usuário for administrador, é possível realizar toda a administração dos projetos existentes na nuvem, em relação as permissões e os usuários participantes de cada projeto.

Por fim, na aba 'NFV' é que será realizada toda a configuração para a utilização de funções de redes virtualizadas. Essa aba só é inserida no contexto do Horizon se o módulo Tacker for instalado no OpenStack. Da mesma forma para outros módulos que exercem alguma função específica, o Horizon só habilita a aba se esse módulo for instalado.

Na aba 'NFV', há duas outras: 'VNF Management' responsável pelo gerenciamento de VNFs; e 'NFV Orchestration' responsável pela orquestração NFV. Perceba que o Tacker habilita os dois blocos padronizado pela ETSI e comentado na Seção 1, completando o bloco MANO da padronização. Assim, a junção entre o VIM (*Virtualized Infrastructure Manager*), habilitado pelo próprio OpenStack, e os VNFM (VNF *Manager*) e NFVO (NFV *Orchestrator*), habilitados pelo módulo Tacker, possibilitam a utilização na prática do paradigma NFV.

Dentro da aba 'VNF Catalog' estão listados todos os *scripts* na linguagem TOSCA, que foram inseridos na plataforma OpenStack. Esses *scripts* estão no catálogo, mas não necessariamente estão em execução, para verificar os que estão em execução é só verificar a aba 'VNF Manager'. O que deve conter esses *scripts* e como inseri-los na plataforma OpenStack são os temas das próximas seções.

Por fim, a última aba, chamada de 'NFV Orchestration' é responsável pela orquestração. Nessa aba destaca-se o 'VNFFG Catalog' e 'VNFFG Manager', são responsáveis pela exibição do catálogo e a execução, respectivamente, de *scripts* relacionados ao encadeamento de funções de redes, o SFC.

Descritores de VNF (VNFD)

O comportamento e a implantação de VNFs no Tacker são definidas em um template definido como Descritor de VNF (VNFD). O *template* é escrito em TOSCA e serão adicionados no catálogo de VNFs. Cada modelo de VNFD contém os campos conforme o Código 1.1.

Código 1.1 – Descrição do Template de uma VNF em TOSCA. Adaptado de [16]

```
tosca_definitions_version :
    # Define a versão do TOSCA que o template se baseia.
    # A versão atual é tosca_simple_profile_for_nfv_1_0_0.
```

```
tosca_default_namespace:
    #Campo opcional. Um namespace para incluir schema, types, version etc.

description:
    #Uma descrição curta sobre o template

metadata:
    template_name: #nome do template

topology_template:
    #Descreve a topologia da VNF no campo node_template
    node_template:
        #Descreve os tipos de nós da VNF
        VDU:
            #Descreve as propriedades e capacidades das VDUs
        CP:
            #Descreve as propriedades e capacidades dos pontos de conexão
        VL:
            #Descreve as propriedades e capacidades dos links  virtuais
```

Tipos de nós

Uma VNF é formada por VDUs, *Connect Points* (CPs) e *Virtual Links* (VLs), sendo que uma VNF necessariamente deve ter esses três componentes. Cada componente é definido como um nó, podendo possuir tipo, capacidade, propriedades, atributos e requisitos distintos.

Uma VDU é uma VM que hospeda uma função de rede (ou parte dela). Uma VNF é dividida em uma ou mais VDUs, ou seja, uma função de rede virtualizada pode ser dividida em uma ou mais VMs. O tipo da VDU é *tosca.nodes.nfv.VDU.Tacker* e define algumas propriedades das VDUs tais como: a imagem do sistema operacional a ser utilizada, as propriedades físicas (número de vCPUs, memória, disco), políticas de monitoramento etc. Um exemplo simples de um *script* de uma VDU com 1 vCPU, 512 MB e 1GB de HD, com a imagem *cirros* é ilustrado nos trechos de Códigos 1.2 e 1.3.

Código 1.2 – Exemplo de VDU. Adaptado de [16]

```
...
topology_template:
    node_template:
        VDU1:
            type: tosca.nodes.nfv.VDU.Tacker
```

```
            properties:
                image: cirros-0.3.4-x86_64-uec
                availability_zone: nova
            capabilities:
                nfv_compute:
                    disk_size: 1 GB
                    mem_size: 512 MB
                    num_cpus: 1
...
```

É possível definir a capacidade de uma VDU descrevendo separadamente a configuração da capacidade da VM ou então usando um *flavor*, ou seja, usando um catálogo de configurações de capacidade da VM, já definidas no OpenStack. O administrador do projeto pode criar seu próprio *flavor*, na aba específica do projeto no Horizon. O Código 1.3 é o mesmo Código 1.2 mas utilizando *flavor*, sendo que o *flavor m1.tiny* significa que a VM terá 1 vCPU, 512 MB de RAM e 1GB de HD.

Código 1.3 – Exemplo de VDU utilizando *flavor*. Adaptado de [16]

```
...
topology_template:
    node_template:
        VDU1:
            type: tosca.nodes.nfv.VDU.Tacker
            properties:
                image: cirros-0.3.4-x86_64-uec
                availability_zone: nova
                flavor: m1.tiny
...
```

Além das inúmeras propriedades, é possível fazer o monitoramento das VDUs via *monitoring_policy*, ou seja, através da adição de eventos de políticas de monitoramento. O Código1.4 mostra um serviço de monitoramento via *ping*. Caso haja três ou mais *pings* com duração maior que dois segundos em um período de 20 segundos, sendo que há um intervalo de dois segundos entre os testes de monitoramento, a VDU será re-instanciada.

Código 1.4 – Exemplo de monitoramento. Adaptado de [16]

```
...
        VDU1:
            type: tosca.nodes.nfv.VDU.Tacker
            properties:
                monitoring_policy:
```

```
                name: ping
                parameters:
                    monitoring_delay: 20
                    count: 3
                    interval: 2
                    timeout: 2
                actions:
                    failure: respawn
...
```

Os *Connect Points* (CPs) são usados para conectar o *Virtual Links* (VL) interno ou externo às VDUs. Um CP requer necessariamente um *Virtual Link* e um *Virtual Binding* associado a ele, sendo opcionais algumas propriedades. O código 1.5 apresenta a configuração de uma VDU com dois CPs, CP1 e CP2, conectadas a dois links virtuais, VL1 e VL2, respectivamente. Além disso as CPs estão sem proteção anti-falsificação e acessível ao usuário.

Código 1.5 – Exemplo de CP. Adaptado de [16]

```
...
        CP1:
            type: tosca.nodes.nfv.CP.Tacker
            properties:
                mac_address: fa:40:08:a0:de:0a
                ip_address: 10.10.1.12
                type: vnic
                anti_spoofing_protection: false
                management: true
                order: 0
                security_groups:
                    - secgroup1
                    - secgroup2
                requirements:
                    - virtualLink:
                        node:VL1
                    - virtualBinding:
                        node: VDU1
        CP2:
            type: tosca.nodes.nfv.CP.Tacker
            properties:
                type: vnic
                anti_spoofing_protection: false
```

```
            management: true
            order: 1
        requirements:
            - virtualLink:
                node: VL2
            - virtualBinding:
                node: VDU1
...
```

O tipo do CP é *nodes.nodes.nfv.CP.Tacker* e os requerimentos são: *virtualLink* e *virtualBinding* que indicam, respectivamente, em qual rede irá conectar e em qual VDU estará associada. Dentre algumas propriedades opcionais podemos citar *anti_spoofing_protection*, *management*, *order*, *mac_address*, *ip_address* e o tipo que especificam, respectivamente, se a proteção anti-falsificação está habilitada na VDU, se o CP é acessível ao usuário, a ordem numerada dos CPs dentro de uma VDU, o endereço MAC, o endereço IP e o tipo da interface do CP podendo ser *vnic* ou *sriov*.

O *Virtual Link* (VL) representa a entidade de ligação logica e fornece conectividade entre as VDUs. O Código 1.6 mostra um *script* de configuração de uma VL cujo fornecedor é o Tacker e está conectado à rede net01.

Código 1.6 – Exemplo de VL. Adaptado de [16]

```
...
    VL1:
        type: tosca.nodes.nfv.VL
        properties:
            vendor: Tacker
            network_name: net01
...
```

Gerenciamento do ciclo de vida de uma VNF

O objetivo desta tarefa é descrever como funciona o catálogo de VNFs, desde uma explicação breve de cada campo e como são inseridos os VNFDs no OpenStack. Além disso será feita a instanciação de uma VNF para demonstrar o ciclo de vida, desde a sua instanciação, utilização e até a deleção.

Para o teste de gerência do ciclo de vida, foi utilizado o *script* para criação de VNFs simples[1] em TOSCA, construído com base nos *scripts* cedidos pela equipe do Tacker [17] e nos documentos de referência da linguagem TOSCA. Este *script* desceve uma VNF

[1]Script para Gerenciamento do Ciclo de Vida de VNFs: `https://github.com/openstack/tacker/blob/stable/ocata/samples/tosca-templates/vnfd/tosca-vnfd-multi-vdu.yaml`

composta por 3 VDUs, cada uma com três pontos de conexão em redes pré-existentes no OpenStack: *net0* (10.10.0.0/24), *net1* (10.10.1.0/24) e *net_mgmt* (192.168.120.0/24). Para a validação da gerência do ciclo de vida básico, foram realizadas operações de inserção, modificação e deleção de VNFs. A Figura 1.8 ilustra o cenário descrito no *script*.

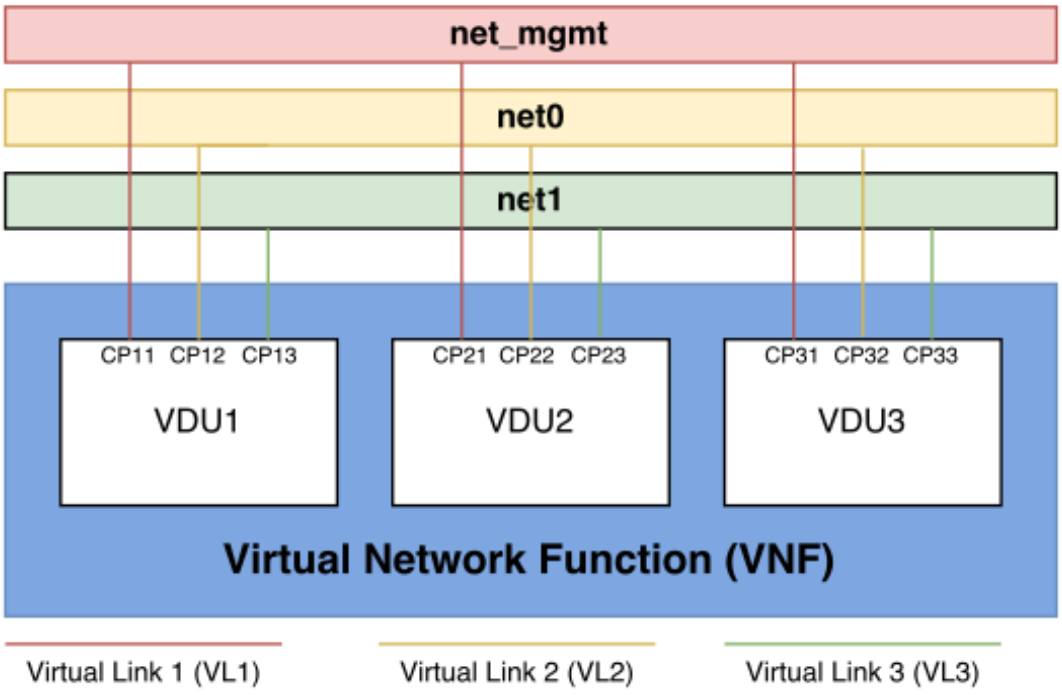


Figura 1.8 – Representação de uma VNF com múltiplos VDUs

O *script* descreve a VNF com três VDUs iguais contendo a seguinte configuração: 1 vCPU, 512 MB de RAM, 1 GB de HD. A imagem escolhida foi a *cirros-0.3.5-x86_64-disk* que já existe no repositório de imagens mantido pelo Glance no OpenStack.

Para acessar a *dashboard* do DevStack, primeiramente insere o endereço IP da máquina que o DevStack está instalado em seu navegador de preferência. É aberta a página de login do OpenStack, conforme a Figura 1.6. Digita-se o login e a senha e aparecerá uma nova página conforme a Figura 1.7. Inicialmente, deve-se alterar o projeto para nfv (que será usado para todos os experimentos) localizado no canto superior a esquerda. Para isso, temos que dar permissão ao usuário utilizado ao projeto nfv. Isso deve ser feito na aba 'Identidade' conforme descrito anteriormente.

Passo 1: Instanciar o descritor de VNF no catálogo

Ao acessar a *dashboard* do OpenStack, utilizando o menu localizado no canto esquerdo do *Horizon*, clica-se na aba **NFV -> VNF *Management* -> VNF *Catalog***. A Figura 1.9 mostra o catálogo de descritores de VNFs que inicialmente encontra-se vazio.

Ao clicar no botão ***Onboard* VNF** (para instanciar um VNFD), será aberta uma nova janela conforme a Figura 1.10.

Há duas maneiras de passar o *script* que descreve a VNF via *dashboard*: a primeira é passando um arquivo, realizando o *upload* do próprio computador utilizado; a segunda é

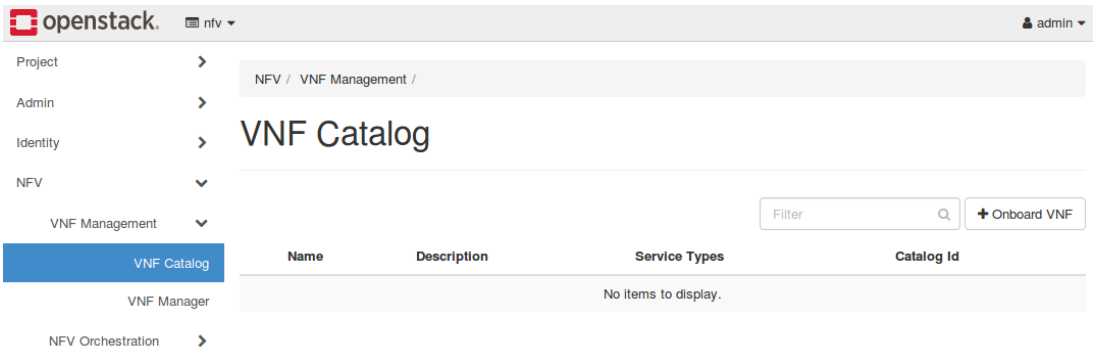


Figura 1.9 – Página de catálogos de VNFs no Tacker

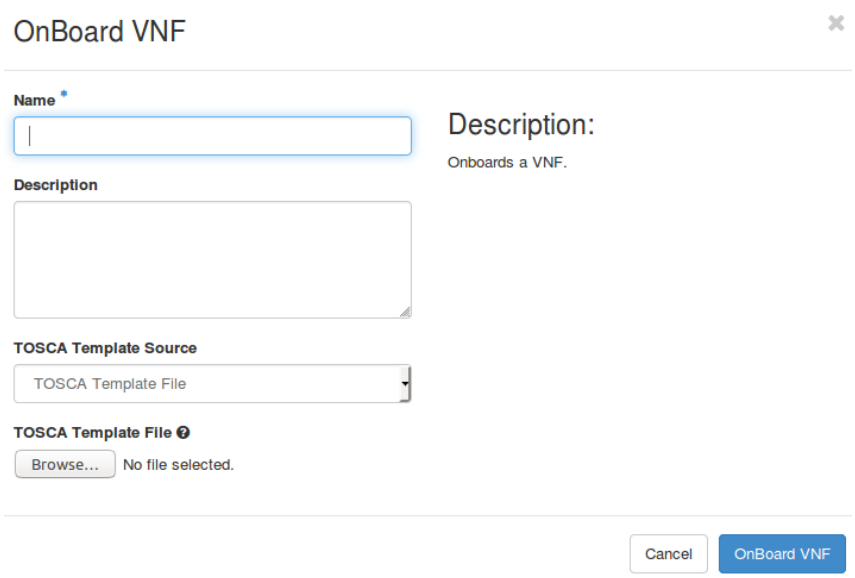


Figura 1.10 – Instanciando um descritor de VNF (VNFD)

trocando a opção em TOSCA *Template Source* para *Direct Input* que abrirá um campo para que seja inserido diretamente o *script* do VNFD.

Após colocar o nome do descritor de VNF, uma breve descrição e o *script* em TOSCA e clicando no botão ***OnBoard* VNF** que o descritor de VNF estará disponível no catálogo de VNF, conforme a Figura 1.11.

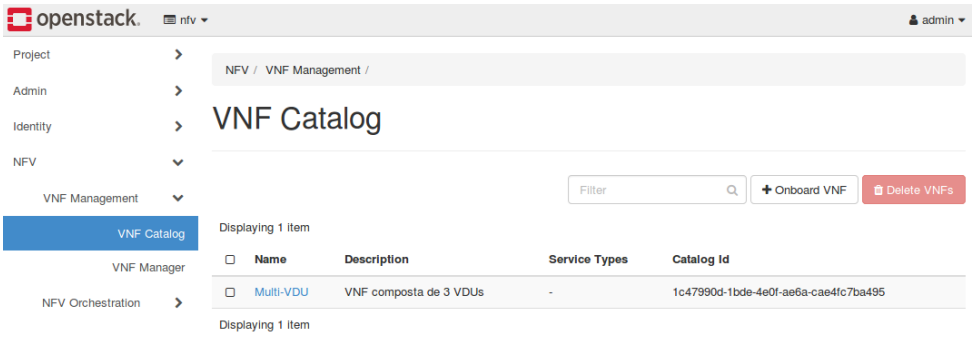


Figura 1.11 – Descritor de VNF ativo no catálogo

Passo 2: Instanciar uma VNF

Com o descritor criado, pode-se instanciar, de fato, a VNF nas aba **VNF *Manager*** no menu NFV. Neste ponto, em VNF *Manager*, ainda não tem a presença de nenhuma VNF, conforme a Figura 1.12.

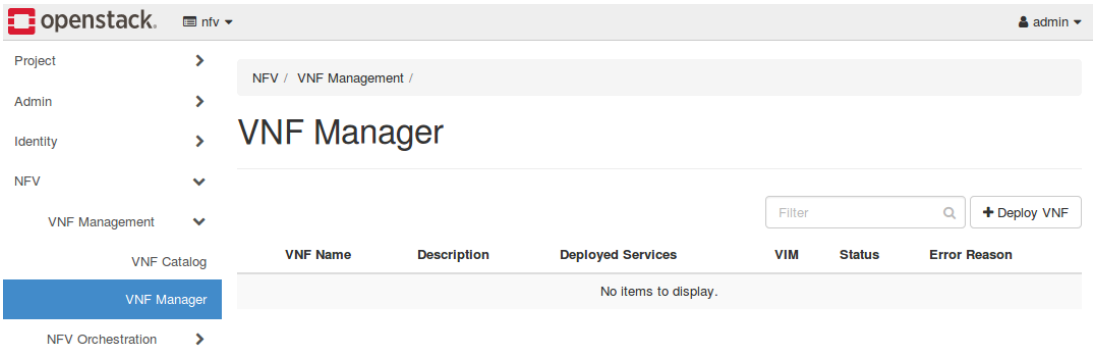


Figura 1.12 – Página de VNF Manager no Tacker

Ao clicar no botão ***Deploy* VNF**, aparecerá uma janela similar a da Figura 1.10. Contudo, aqui será instanciado uma VNF dizendo qual é o descritor no catálogo de VNFs que será utilizado para tal. Neste caso será o descritor inserido no passo anterior. Ao colocar o nome da VNF, uma breve descrição, selecionar a VIM e o VNFD; e apertando o botão o botão ***Deploy* VNF** será instanciado a VNF.

No momento da instanciação da VNF também existe a opção de adicionar um *script* sem que este esteja no catálogo. Porém quando o *script* é adicionado no catálogo, este é verificado para ver se o código está escrito corretamente, além de prover uma melhor organização. Dessa forma, recomenda-se inserir no VNFD primeiramente no catálogo e depois realizar a instanciação.

Ao confirmar a instanciação da VNF, ela entrará em estado *PENDING_CREATE* até se tornar efetivamente ativa com o estado *ACTIVE*. A Figura 1.14 mostra a VNF instanciada com o estado ativo.

Pode-se certificar que a VNF foi devidamente criada por meio da interface de visualização das topologias de rede no OpenStack, na aba ***Project -> Network Topology***. Nela, são visualizados três VMs (ou VDUs no contexto de NFV), cada uma ligada às três redes (*net0, net1 e net_mgmt*) descritas no *script* para criação de VNF simples como mostra a Figura 1.14.

Pode-se também observar as instancias criadas na aba ***Project -> Compute -> Instances***. Nela, observa-se as VMs com seus respectivos IPs em cada interface, a imagem utilizada, o status da VDU (se está ativa ou não), conforme ilustrada na Figura 1.16.

Passo 3: Interagindo com uma VNF

Há duas formas de se interagir com a VNF: via *dashboard* ou CLI. Podemos visualizar as VDUs associadas à VNF, via *dashboard*, conforme é ilustrado na Figura 1.16. Selecionando

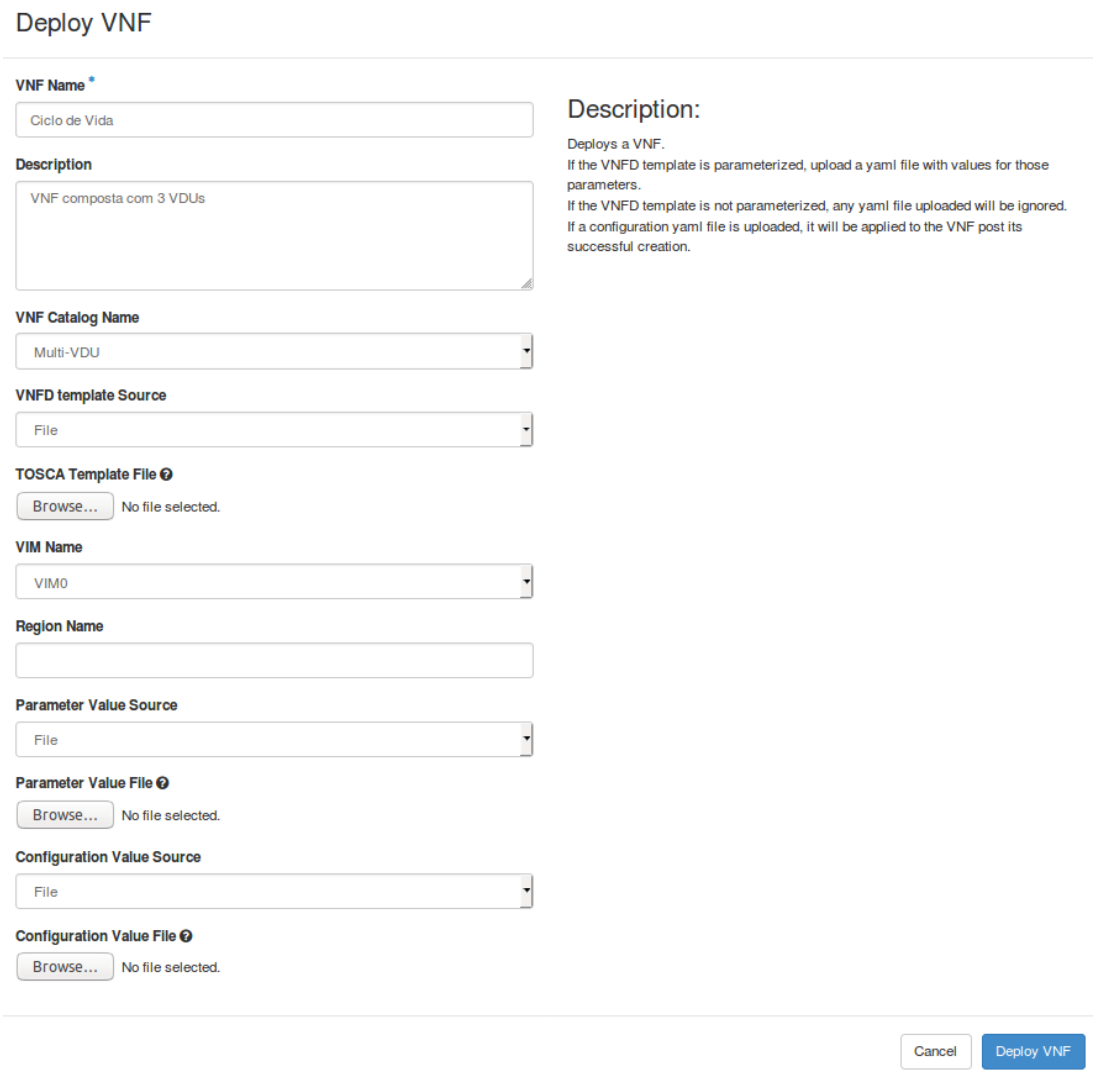


Figura 1.13 – Instanciando uma VNF em VNF *Manager*

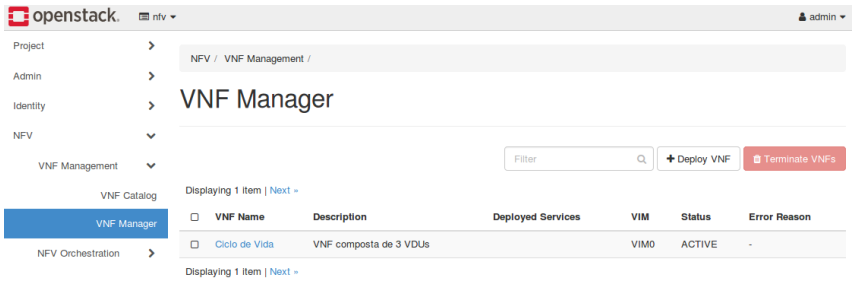


Figura 1.14 – VNF ativa em VNF *Manager*

a aba ***Console*** dentro de ***Instances***, a VDU é acessada via terminal conforme mostrado na Figura 1.17.

Neste cenário foi utilizada a imagem cirros, que é um uma pequena imagem do Linux, usada para testar implantação do OpenStack. Para utilizar outras imagens, contendo um determinado sistema operacional e/ou serviço específico, basta apenas inserir as imagens na aba ***Project -> Images***. O OpenStack aceita diversos tipos de formato de imagem[2].

[2]Formatos e como gerar as imagens: `https://docs.openstack.org/image-guide/index.html`

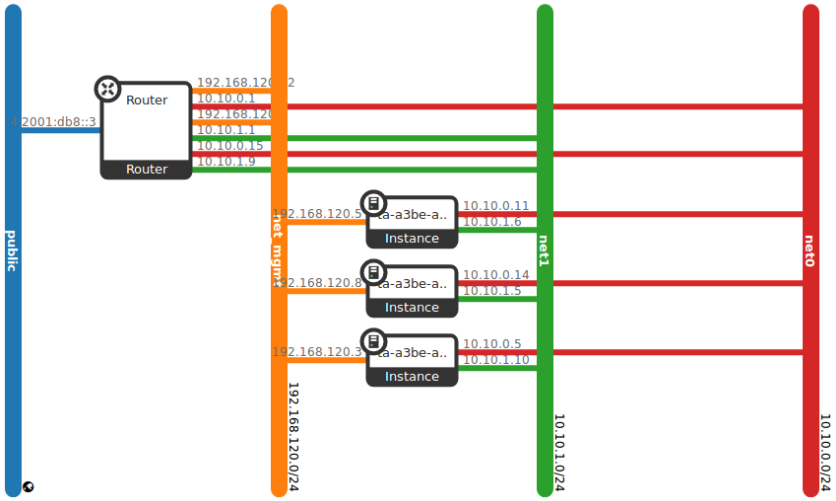


Figura 1.15 – Certificação da VNF ativa na topologia de rede do OpenStack

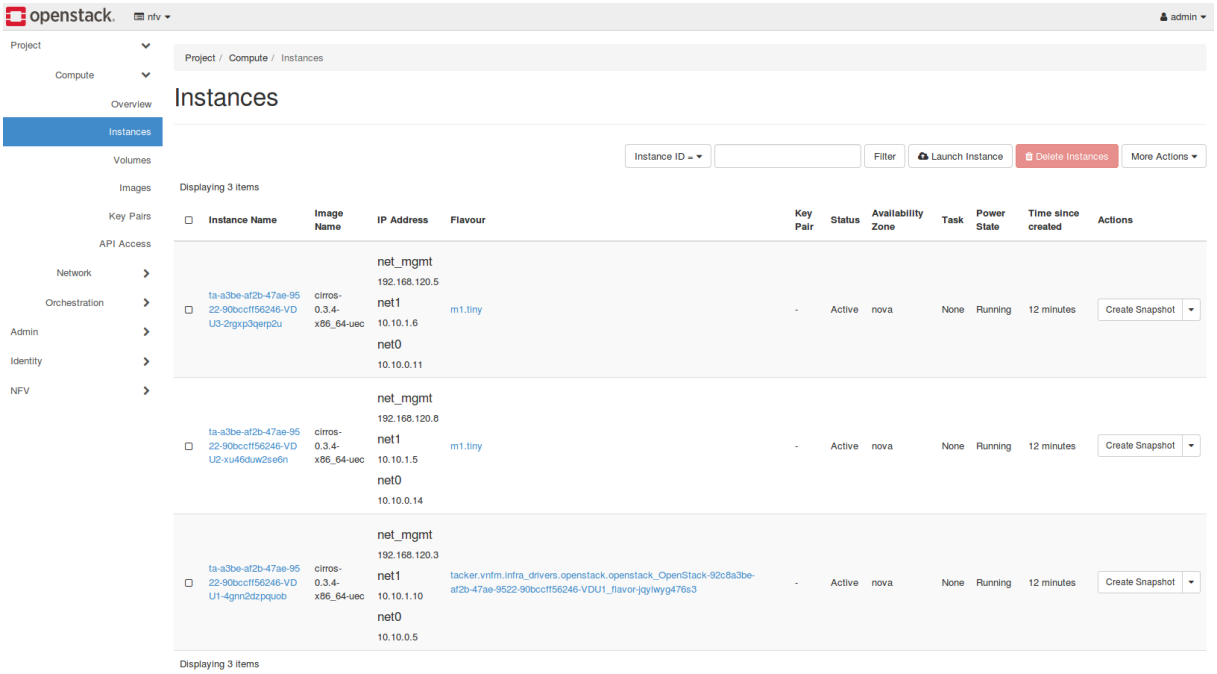


Figura 1.16 – Instancias da VNF

Pode-se inserir imagens dentro do OpenStack para que possa ser usada também.

Passo 4: Deletando uma VNF

Para a deleção, pode-se realizar selecionando a VNF desejada, localizado na aba **NFV -> VNF *Manager*** conforme mostrado na Figura 1.14, e clicando no botão ***Terminate* VNFs**. A VNF entrará em modo ***PENDING_DELETE*** e então será deletada da plataforma. Neste momento, todas as VDUs associadas a VNF serão desalocadas dos recursos da infraestrutura.

Monitorando uma VNF

O objetivo desta tarefa é demonstrar algumas funcionalidades básicas de monitoramento de VNFs tais como, o *health monitoring*.

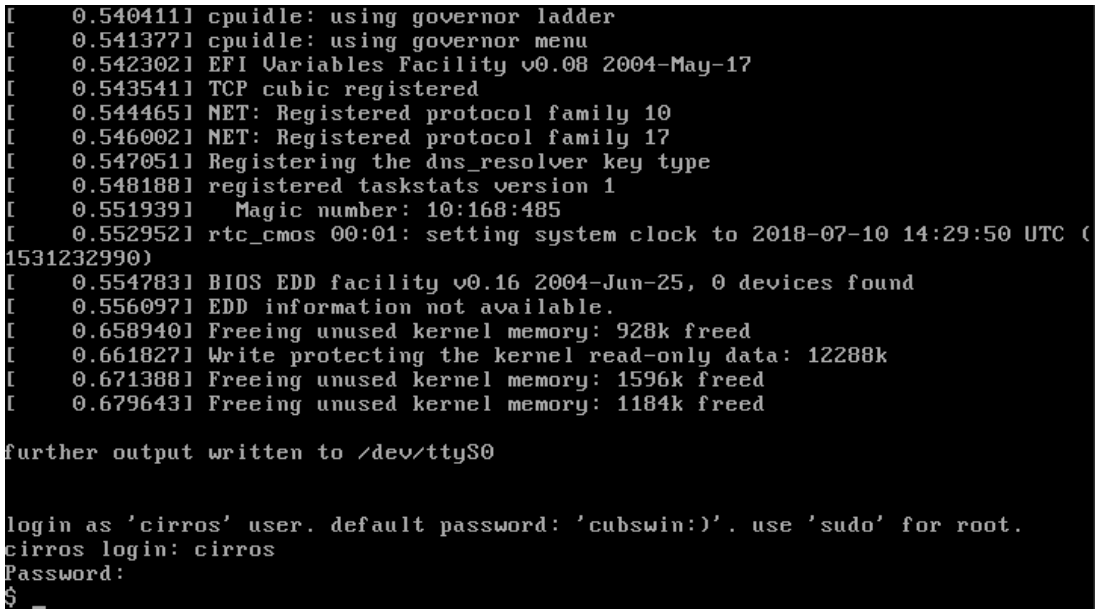


Figura 1.17 – Acessando a VDU

O gerenciador de VNFs precisa monitorar as condições de *status* das entidades VNF que implementa e administra. O *Tacker Monitoring Framework* do NFVM da arquitetura Tacker, conforme mostrado na Figura 1.4, realiza esse monitoramento utilizando apenas um método simples, o *ping*, e não suporta o monitoramento do uso de CPU/memória de elementos da VNF. Foi projetado um *driver* de monitoramento baseado em alarmes no Tacker para coletar alarmes/eventos desencadeados pelos projetos Ceilometer e AODH. Esse drive de monitoramento pode monitorar completamente os recursos que o Ceilometer pode suportar.

O código 1.4 descreve o *template* para fazer o monitoramento usando *Tacker Monitoring Framework*. Nele define-se qual monitoramento vai ser utilizado, por exemplo, *ping* ou *http-ping*. Pode-se descrever também quais parâmetros serão usados, tais como, *port*, *url*, *interval*, *timeout*, *monitoring_delay*, etc. Quando os parâmetros atingem seu valor limiar, são ativados eventos, como por exemplo, *failure*, que podem ter associados uma ou mais ações. *Respawn*, *autoscale*, *log* são exemplos de possíveis ações.

Para o teste simples de monitoramento foi utilizado um *script* de monitoramento de VNF[3] em TOSCA, construído com base no documento de descrição da linguagem TOSCA e nos *scripts* cedidos pelo projeto Tacker. O *script* tem a função de criar uma VNF com apenas uma VDU associada à um *Connect Point* que será ligada por meio de um *Virtual Link*, a rede *net_mgmt*, pré-existente no OpenStack.

O teste de monitoramento faz a verificação de inatividade de uma VNF através do ICMP. Caso o monitoramento tenha a contabilização de três estados de inatividade, ou seja, existiram pelo menos três estados em que a VDU teve que esperar a resposta de um *ping* maior que dois segundos em um intervalo de tempo de 45 segundos. No *script* a ação adotada foi a de *respawn*, que em caso de falha faça com que a VDU seja re-instanciada

[3]Script para Monitoramento simples de VNFs: `https://github.com/openstack/tacker/blob/stable/ocata/samples/tosca-templates/vnfd/tosca-vnfd-monitor.yaml`

conforme a leitura do monitoramento. A VDU re-instanciada terá novamente um monitor associado, com as mesmas políticas descritas no *script*.

Passo1: Instanciando uma VNF com *monitoring*

A instanciação da VNF com funcionalidade de monitoramento será semelhante a seção 1, utilizando o *script* de monitoramento de VNF.

Passo 2: Testando a funcionalidade de monitoramento

Após a VNF ser instanciada, podemos ver os detalhes da VNF tais como, os eventos desde a sua criação e política adotada clicando na VNF em questão na aba **VNF *Manager***, conforme ilustrado na Figura 1.18.

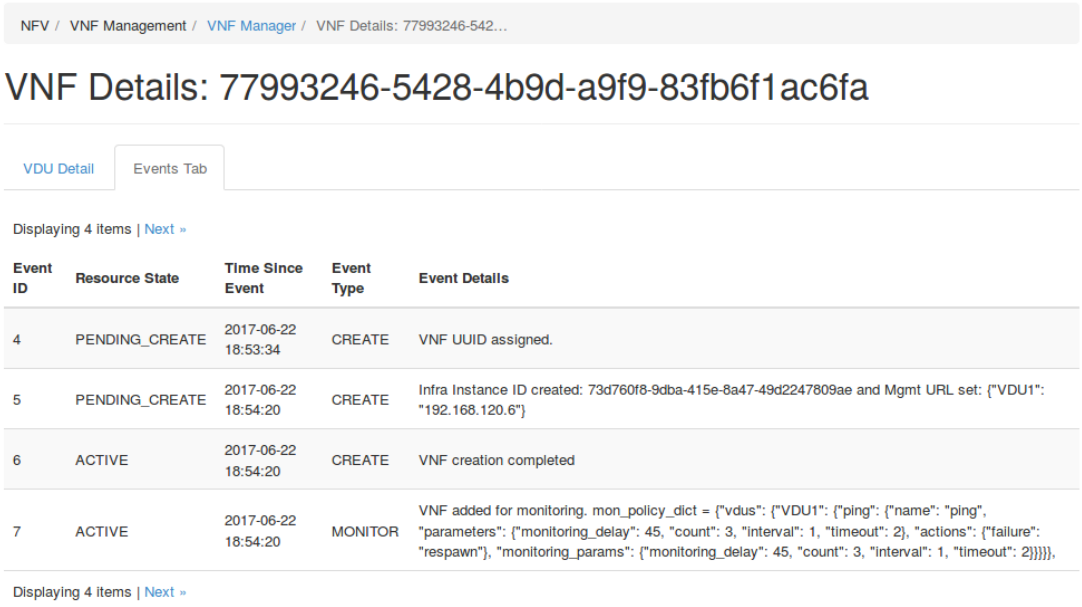


Figura 1.18 – Confirmação de associação do monitor à VNF.

Verifica-se que a VDU está ativa e associada a rede *net_mgmt*, conforme ilustrado na Figura 1.18, de acordo com os logs apresentado no OpenStack.

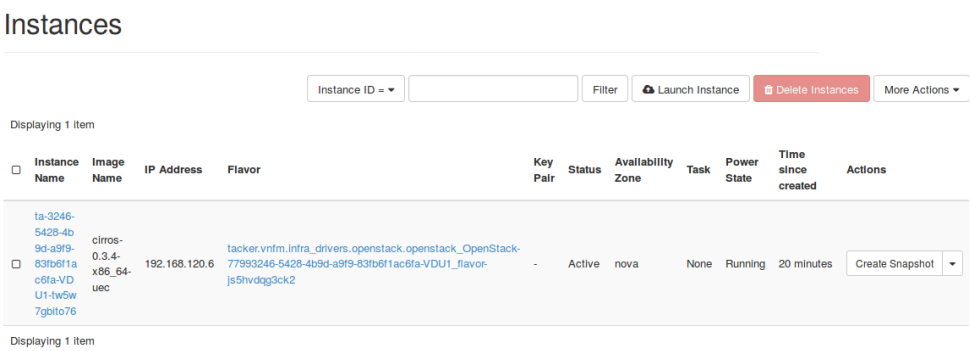


Figura 1.19 – VDU ativa e associada à rede net_mgmt

Para a confirmação do funcionamento do monitoramento, será forçada a ação de *respawn*, acessando a VDU associada à VNF e desativando sua interface de rede para que o monitor possa assumir um estado de inatividade, conforme ilustrado na Figura 1.20.

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr FA:16:3E:BD:9D:91
          inet addr:192.168.120.6  Bcast:192.168.120.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:febd:9d91/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:643 errors:0 dropped:0 overruns:0 frame:0
          TX packets:719 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:59702 (58.3 KiB)  TX bytes:65025 (63.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

$ ifconfig eth0 down_
```

Figura 1.20 – Desativando a interface de rede da VNF

Após executar o comando *sudo ifconfig eth0 down* para desativar a interface de rede, ocorrerá o estado de inatividade, após o intervalo de monitoramento, fazendo com que seja seja re-instancidada a VDU. A Figura 1.21 mostra a VDU re-instanciada com um novo endereço IP.

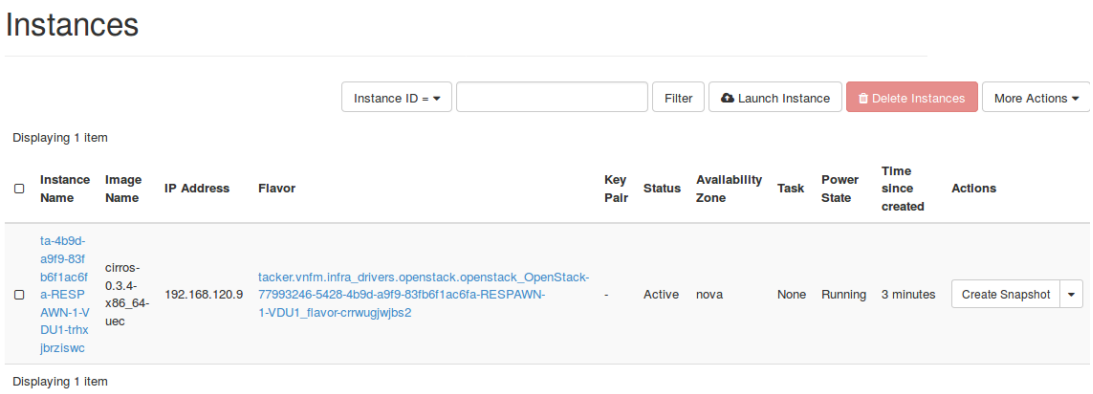


Figura 1.21 – VDU re-instanciada pela política de monitoramento

Podemos acompanhar a continuação dos eventos da VNF, descritos anteriormente na Figura 1.18, com a ação de *respawn* conforme apresentado na Figura 1.22.

Construindo serviços escaláveis no OpenStack

O objetivo desta tarefa é demonstrar a funcionalidade de escalabilidade horizontal de VNFs, tais como, *scale in* e *scale out*.

A escalabilidade de VNFs, função do bloco VNFM, consiste em adicionar às VNFs a capacidade de se escalar para dentro (*scale in*) ou para fora (*scale out*) por meio de um disparo (*trigger*) feito pelo usuário ou por alguma política baseada em condições de

VNF Details: 77993246-5428-4b9d-a9f9-83fb6f1ac6fa

VDU Detail | Events Tab

Displaying 9 items | Next »

Event ID	Resource State	Time Since Event	Event Type	Event Details
4	PENDING_CREATE	2017-06-22 18:53:34	CREATE	VNF UUID assigned.
5	PENDING_CREATE	2017-06-22 18:54:20	CREATE	Infra Instance ID created: 73d760f8-9dba-415e-8a47-49d2247809ae and Mgmt URL set: {"VDU1": "192.168.120.6"}
6	ACTIVE	2017-06-22 18:54:20	CREATE	VNF creation completed
7	ACTIVE	2017-06-22 18:54:20	MONITOR	VNF added for monitoring. mon_policy_dict = {"vdus": {"VDU1": {"ping": {"name": "ping", "parameters": {"monitoring_delay": 45, "count": 3, "interval": 1, "timeout": 2}, "actions": {"failure": "respawn"}, "monitoring_params": {"monitoring_delay": 45, "count": 3, "interval": 1, "timeout": 2}}}}},
8	DEAD	2017-06-22 19:45:59	MONITOR	
9	ACTIVE	2017-06-22 19:46:01	MONITOR	ActionRespawnHeat invoked
10	ACTIVE	2017-06-22 19:46:26	CREATE	Infra Instance ID created: 82997e9b-75ff-4396-9d52-d5db3ce1fca9 and Mgmt URL set: {"VDU1": "192.168.120.9"}
11	ACTIVE	2017-06-22 19:46:26	CREATE	VNF creation completed
12	ACTIVE	2017-06-22 19:46:26	MONITOR	VNF added for monitoring. mon_policy_dict = {"vdus": {"VDU1": {"ping": {"name": "ping", "parameters": {"monitoring_delay": 45, "count": 3, "interval": 1, "timeout": 2}, "actions": {"failure": "respawn"}, "monitoring_params": {"monitoring_delay": 45, "count": 3, "interval": 1, "timeout": 2}}}}},

Displaying 9 items | Next »

Figura 1.22 – Log de eventos da ação de *respawn* da VNF

estado, sendo a última utilizada em conjunto com os mecanismos de módulo Ceilometer do OpenStack, módulo que provê serviços de telemetria, para monitoramento e alarme.

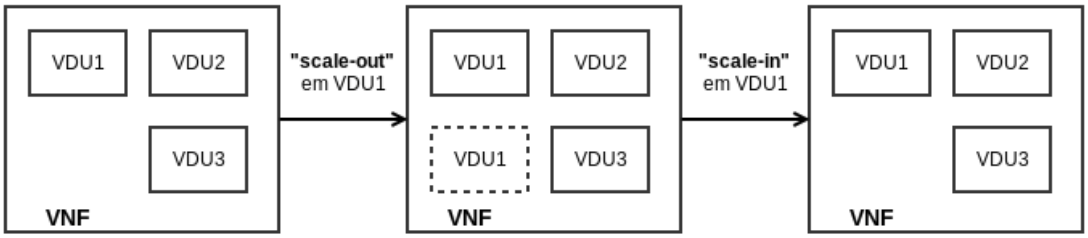


Figura 1.23 – Scaling horizontal de um componente(VDU1) da VNF

Para o teste da funcionalidade de scaling horizontal no Tacker, foi utilizado o *script* de VNF com *scale* manual[4] em TOSCA. Esse *script* levanta uma VNF contendo inicialmente uma VDU com a seguinte configuração: 1 vCPU, 512 MB e 1 GB de HD, com a imagem *cirros-0.3.4-x86_64-uec* e utilizando um CP conectado à rede *net0*. O Código 1.7 destaca parte do *script* de *scaling* horizontal.

Código 1.7 – Trecho do *script* de *scaling*

[4]Script VNF com funcionalidade de *Scale* manual: `https://github.com/openstack/tacker/blob/stable/ocata/samples/tosca-templates/vnfd/tosca-vnfd-scale.yaml`

```
policies :
  — SP1:
      type: tosca. policies .tacker.Scaling
      properties:
        increment: 1
        cooldown: 30
        min_instances: 1
        max_instances: 3
        default_instances: 1
        targets: [VDU1]
```

A política de *scaling* utilizada no *script* descreve como será feita a escalabilidade da VNF. Podemos definir a propriedade *increment* que determina o número em que o componente alvo será incrementado (em caso de *scale out*) ou decrementado (em caso de *scale in*). *Cooldown* corresponde ao tempo mínimo de espera para uma nova operação de *scaling*. Os parâmetros *min_instances* e *max_instances* indicam o número mínimo e máximo de instâncias que a VNF pode ter, respectivamente. *Default_instances* define o número de instâncias de cada alvo no ato da criação da VNF e , por fim, o parâmetro *target* determina quais componentes (VDUs) da VNF serão atingidas no processo de *scaling*, neste caso VDU1.

Passo 1: Permissão para acessar o Tacker via CLI

Baixe o arquivo "OpenStack RC File v3". Após o download do arquivo, copie-o para a pasta onde o OpenStack está instalado. Ao acessar a VM do OpenStack, basta executar o comando *source* seguido pelo nome do arquivo via terminal conforme é ilustrado na Figura 1.24.

```
stack@GT-NosFVeraTO2:~$ ls
devstack  local.conf  nfv-openrc.sh
stack@GT-NosFVeraTO2:~$ source nfv-openrc.sh
Please enter your OpenStack Password for project nfv as user admin:
stack@GT-NosFVeraTO2:~$
stack@GT-NosFVeraTO2:~$
```

Figura 1.24 – Executando arquivo para utilização do Tacker via CLI

Passo 2: Instanciar a VNF com funcionaliade de *Scale*

A instanciação da VNF com funcionalidade de *Scale* será semelhante a seção 1, utilizando o *script* de VNF com *scale* manual.

Passo 3: Interagindo com a VNF

Após a instanciação da VNF é possível verificar os eventos associados a ela na aba Eventos da VNF, conforme é ilustrado na Figura 1.25.

Pode-se notar que, nesse estado inicial, a VNF possui apenas uma instância de VDU (com IP 10.10.0.8). Conforme especificado no *script*, o alvo de *scale* será a VDU1; é esperado que seu número de instâncias altere conforme as operação de *scaling*.

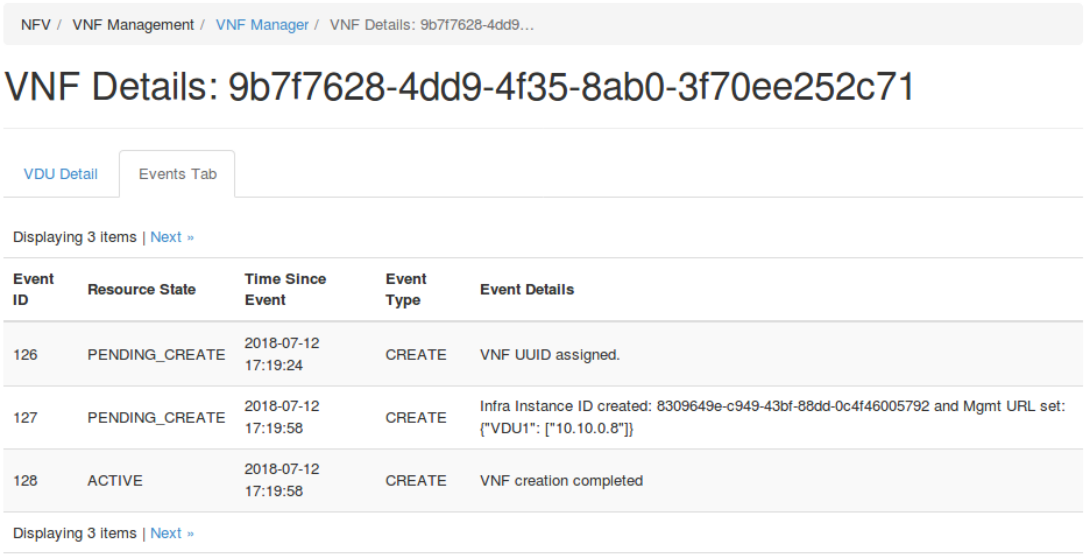


Figura 1.25 – Eventos associados à VNF antes do *Scaling*

O *scaling* é feito de forma manual via *tacker-api*. O *scale out* e *scale in* da VNF por meio da *tacker-api* pode ser feito através do seguinte comando:

Código 1.8 – Comando para *scaling* manual

tacker vnf—scale ——vnf—name <Nome da VNF> ——scaling—policy—name <Política de scaling> ——scaling—type <IN|OUT>

Passo 4: *Scale out*

A Figura 1.26 apresenta o uso da *tacker-api* para a realização de *scale out* em uma VNF. A política utilizada de *scaling* está definida no Código 1.8. O nome da VNF é "*Scale*" que pode observada utilizando o comando *tacker vnf-list*. Esse comando lista todas as VNFs mostrando algumas informações como *name*, *mgmt_url*, status entre outros. O parâmetro *mgmt_url* mostra todas as instâncias e os seus respectivos IPs.

```
stack@GT-NosFVeraTO2:~$
stack@GT-NosFVeraTO2:~$ tacker vnf-list
+--------------------+-------+--------------------+--------+--------------------+--------------------+
| id                 | name  | mgmt_url           | status | vim_id             | vnfd_id            |
+--------------------+-------+--------------------+--------+--------------------+--------------------+
| 9b7f7628-4dd9-4f3  | Scale | {"VDU1":           | ACTIVE | 8f387ee5-882b-436  | 81125ca7-66ae-4429 |
| 5-8ab0-3f70ee252c  |       | ["10.10.0.8"]}     |        | 9-9b94-26ff1bdcf5  | -ab2a-05bbc582a183 |
| 71                 |       |                    |        | c8                 |                    |
+--------------------+-------+--------------------+--------+--------------------+--------------------+
stack@GT-NosFVeraTO2:~$ tacker vnf-scale --vnf-name Scale --scaling-policy-name SP1 --scaling-type out
```

Figura 1.26 – Operação de *scale out* na VNF

Verifica-se, através da aba de eventos da VNF, que ocorrerá um estado de *PENDING_SCALE_OUT* para escalar uma nova VDU, conforme é mostrado na Figura 1.27. Já que o único alvo das operações de *scale* é VDU1, após a operação, se verificar na aba instâncias no Horizon, pode-se ver duas instâncias de VDU1.

O *Scale in* é feito de forma semelhante ao *Scale out*, substituindo o tipo de *scaling*, no comando executado de scale de OUT para IN.

VDU Detail | Events Tab

Displaying 5 items | Next »

Event ID	Resource State	Time Since Event	Event Type	Event Details
126	PENDING_CREATE	2018-07-12 17:19:24	CREATE	VNF UUID assigned.
127	PENDING_CREATE	2018-07-12 17:19:58	CREATE	Infra Instance ID created: 8309649e-c949-43bf-88dd-0c4f46005792 and Mgmt URL set: {"VDU1": ["10.10.0.8"]}
128	ACTIVE	2018-07-12 17:19:58	CREATE	VNF creation completed
129	PENDING_SCALE_OUT	2018-07-12 18:06:04	SCALE	
130	ACTIVE	2018-07-12 18:06:37	SCALE	

Displaying 5 items | Next »

Figura 1.27 – Eventos associados à VNF após uma operação de *scale out*

Composição de serviços através do encadeamento de funções de rede

Esta prática tem como objetivo explorar as possibilidades de habilitação de SFC no OpenStack.

O encadeamento de VNFs ou *Service Function Chaining* (SFC), função do bloco de orquestração NFVO, define como as VNFs serão encadeadas em um fluxo de rede. O IETF [18] define o conceito de SFC como sendo uma forma de distribuir, de forma ordenada e sequencial, os serviços de rede ao longo de uma infraestrutura para que o tráfego flua entre estas segundo alguma classificação.

Em um exemplo simples, poderíamos forçar que o tráfego fluísse de uma máquina A para uma máquina B, passando necessariamente por um *firewall* colocado entre elas, mesmo que o *firewall* não esteja de fato no ponto de vista de suas tabelas de roteamento. A Figura 1.28 ilustra um cenário desse tipo, em que o tráfego que sai de uma máquina cliente é encadeado por duas funções de rede, um *firewall* e um IDS (*Intrusion Detection System*), para então alcançar um servidor *web* do domínio SFC.

Existem algumas soluções que permitem realizar o SFC na plataforma OpenStack. Várias delas passam pela utilização de um projeto que permite habilitar SFC no módulo Neutron, denominado *networking-sfc*[5]. O Tacker traz a funcionalidade da implementação de VNFFG (VNF *Forwarding Graph*) utilizando o *networking-sfc* internamente.

VNF *Forwarding Graph*

O VNF *Forwarding Graph* no Tacker é usado para orquestrar e gerenciar o tráfego através de VNFs. As definições genéricas de VNFFG são descritas em TOSCA são processadas em classificadores e o mapeamento de VNFs. O mapeamento de VNFs compõe uma lista

[5]Networking-SFC Documentation. 2016: `http://docs.openstack.org/developer/networking-sfc/`

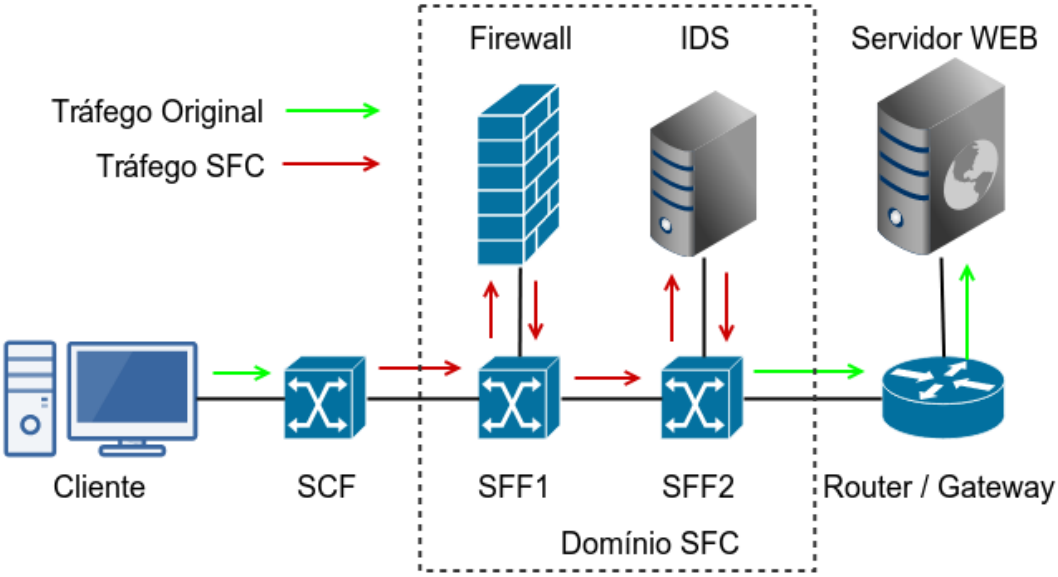


Figura 1.28 – Cenário de SFC

ordenada de funções de rede para que o tráfego percorra, enquanto o classificador decide qual tráfego deve passar por elas.

Para a criação de um VNFFG são necessários uma origem e destino; e uma ou mais funções de redes que será encadeada. O classificador será um *script* em TOSCA que definirá uma porta de origem de onde iniciará o fluxo, uma porta destino, além dos protocolos que poderão ser utilizados, etc. As VNFs no VNFFG são funções de rede que ficam entre a origem e destino que representam os grupos de pares de portas no *networking-sfc*.

O comando abaixo ilustra como é instanciada uma VNFFG via CLI. Nesse comando, o *vnffgd-template* define um descritor de VNFFG que está no OpenStack para instanciar o VNFFG e o *vnf-mapping* lista um mapeamento de instancias VNFD para VNF. Também é possível habilitar a simetria, ou seja, o caminho de ida dos pacotes que serão encaminhados da origem para o destino seja o mesmo de volta, criando um classificador extra para garantir que o tráfego flua através de um caminho inverso.

Código 1.9 – Instanciamento de uma VNFFG via CLI

```
tacker vnffg–create ––vnffgd–template <template do vnffgd> \
––vnf–mapping <mapeamento das VNFs> \
––symetrical <TRUE|FALSE> <nome do VNFFG>
```

Para o teste da funcionalidade do SFC no OpenStack utilizando o VNFFG foi montado um cenário de conforme ilustrado na Figura 1.29.

Este cenário de SFC utiliza duas VMs, fonte e destino; *scripts* para um VNFFG[6] e duas

[6]Script para o VNFFG: `https://github.com/openstack/tacker/blob/stable/ocata/samples/tosca-templates/vnffgd/tosca-vnffgd-sample.yaml`

VNFs, VNF1[7] e VNF2[8], representando as funções de rede pelas quais o pacote que sai da fonte em direção ao destino tem que, obrigatoriamente, passar; construído com base nos *scripts* cedidos pela equipe do Tacker [17] e nos documentos de referência da linguagem TOSCA [12]. Lembrando que tantos as VMs quanto as VNFs estão na mesma rede, podendo também estar em redes distintas. As setas na cor preta indicam o caminho de ida e a azul indica o caminho de retorno. O VNFFG pode ser realizado tanto via CLI quanto via *dashboard*. Parte do experimento foi realizado via CLI, por ser mais fácil de visualizar os dados, como endereço IP, MAC; etc. Tanto as VMs e VNFs terão a seguinte configuração: 1 vCPU, 512 MB de RAM e 1GB de Disco contendo a imagem *cirros*. Ambas as VNFs terão a funcionalidade de encaminhador de pacotes L3.

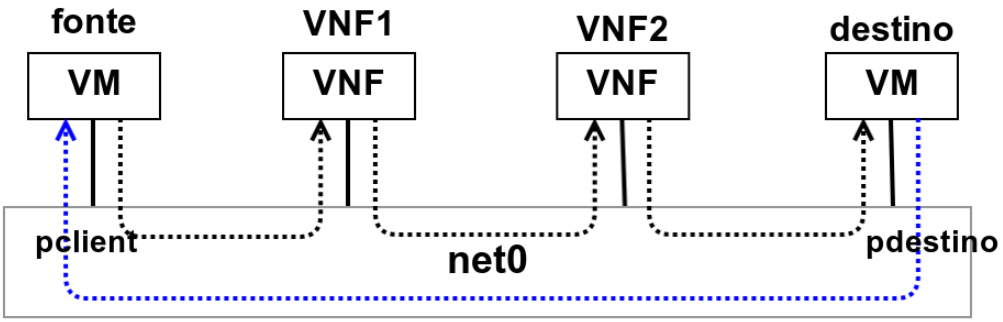


Figura 1.29 – Cenário de teste de SFC usando VNFFG

Passo 1: Criação dos *Connect Points* (CPs) das VMs

Inicialmente é preciso criar os *connect points* das VMs, fonte e destino, pois se torna fácil de observar seus atributos como, endereço IP e MAC, id da portas do Neutron; e também desabilitar a proteção ML2 *Port Security* para habilitar que uma instância em específico possa repassar tráfegos com MAC/IP de origem diferentes de seu próprio. O código abaixo descreve como são criadas as portas, via CLI, ligadas a rede net0 para o cenário de teste.

Código 1.10 – *Script* para criação de portas, via CLI

```
openstack port create pfonte ––network net0 ––disable–port–security
openstack port create pdestino ––network net0 ––disable–port–security
```

A Figura 1.30 ilustra a mensagem de retorno quando um CP é criado, na qual é possível observar algumas informações importantes, como por exemplo endereço IP, MAC; id da porta etc.

[7] Script para a VNF1: `https://github.com/openstack/tacker/blob/stable/ocata/samples/tosca-templates/vnffgd/tosca-vnffg-vnfd1.yaml`

[8] Script para a VNF2: `https://github.com/openstack/tacker/blob/stable/ocata/samples/tosca-templates/vnffgd/tosca-vnffg-vnfd2.yaml`

```
stack@ubuntu:~$ openstack port create pfonte --network net0 --disable-port-security
+-----------------------+--------------------------------------------------------------+
| Field                 | Value                                                        |
+-----------------------+--------------------------------------------------------------+
| admin_state_up        | UP                                                           |
| allowed_address_pairs |                                                              |
| binding_host_id       |                                                              |
| binding_profile       |                                                              |
| binding_vif_details   |                                                              |
| binding_vif_type      | unbound                                                      |
| binding_vnic_type     | normal                                                       |
| created_at            | 2017-10-30T16:55:50Z                                         |
| description           |                                                              |
| device_id             |                                                              |
| device_owner          |                                                              |
| dns_assignment        | None                                                         |
| dns_name              | None                                                         |
| extra_dhcp_opts       |                                                              |
| fixed_ips             | ip_address='10.10.0.6', subnet_id='cdbd2094-91d5-4e60        |
|                       | -ad4f-916a4fef0c46'                                          |
| id                    | 7f4eeb6b-0598-41a9-aef3-266ec609c636                         |
| ip_address            | None                                                         |
| mac_address           | fa:16:3e:ac:d6:06                                            |
| name                  | pfonte                                                       |
| network_id            | 7314d2e5-cab6-4bcd-8ada-6207346b79f9                         |
| option_name           | None                                                         |
| option_value          | None                                                         |
| port_security_enabled | False                                                        |
| project_id            | d505bc8566ed42a6969779c86c000c6a                             |
| qos_policy_id         | None                                                         |
| revision_number       | 4                                                            |
| security_groups       |                                                              |
| status                | DOWN                                                         |
| subnet_id             | None                                                         |
| updated_at            | 2017-10-30T16:55:50Z                                         |
+-----------------------+--------------------------------------------------------------+
```

Figura 1.30 – Criação de um *Connect Point* (CP)

Passo 2: Instanciação das VMs fonte e destino

Nesta etapa, as VMs fonte e destino serão instanciadas. Os comandos abaixo mostram como instanciar VMs via CLI no OpenStack

Código 1.11 – *Script* para instanciamento de VMs, via CLI

```
openstack server create fonte --nic-port-id=pfonte --flavor m1.tiny \\
--image cirros-0.3.4-x86_64-uec
openstack server create destino --nic-port-id=pfonte --flavor m1.tiny \\
--image cirros-0.3.4-x86_64-uec
```

Passo 3: Criação do descritor de VNFFG (VNFFGD)

Para a criação do VNFFGD, utilizamos o script para a descrição do VNFFG. O trecho de Código 1.12 descreve o classificador de fluxo e mostra algumas informações que merecem destaque.

Código 1.12 – Trecho do script de VNFFGD

```
...
Forwarding_path1:
    type: tosca.nodes.nfv.FP.Tacker
    description: creates path (CP12->CP22)
    properties:
```

```
    id: 51
    policy:
      type: ACL
       criteria :
        - network_src_port_id: 7f4eeb6b-0598-41a9-aef3-266ec609c636
        - ip_dst_prefix: 10.10.0.13/24
    path:
      - forwarder: VNF1
        capability: CP12
      - forwarder: VNF2
        capability: CP22
...
```

O *Forwarding_Path* (FP) possui alguns parâmetros importantes, tais como: *network_src_port_id* e *ip_dst_prefix*, que representam respectivamente, a porta da rede na qual a VM fonte está conectada e o endereço IP da VM destino. Além disso, o FP decide quais VNFD participação do SFC e quais *Connect Points* serão usados.

O Descritor de VNFFG pode ser instanciado no Catálogo de VNFFG na aba **NFV -> NFV *Orchestration* -> VNFFG *Catalog*** ou via CLI. Para instanciar o VNFFGD no catálogo de VNFFG via CLI, foi usado o comando mostrado na Figura 1.31.

```
stack@ubuntu:~$ tacker vnffgd-create VNFFGD --vnffgd-file vnffgd.yaml
Created a new vnffgd:
+-------------+--------------------------------------------------------------------------------------------------------+
| Field       | Value                                                                                                  |
+-------------+--------------------------------------------------------------------------------------------------------+
| description |                                                                                                        |
| id          | 56e91ed2-0c32-449d-8ee8-55523574b15f                                                                   |
| name        | VNFFGD                                                                                                 |
| template    | {"vnffgd": {"imports": ["/opt/stack/tacker/tacker/vnfm/tosca/lib/tacker_defs.yaml",                    |
|             | "/opt/stack/tacker/tacker/vnfm/tosca/lib/tacker_nfv_defs.yaml"], "description": "Sample VNFFG          |
|             | template", "topology_template": {"node_templates": {"Forwarding_path1": {"type":                       |
|             | "tosca.nodes.nfv.FP.Tacker", "description": "creates path (CP12->CP22)", "properties":                 |
|             | {"policy": {"type": "ACL", "criteria": [{"network_src_port_id":                                        |
|             | "7f4eeb6b-0598-41a9-aef3-266ec609c636"}, {"ip_dst_prefix": "10.10.0.13/24"}]}, "path":                 |
|             | [{"capability": "CP12", "forwarder": "VNFD1"}, {"capability": "CP22", "forwarder": "VNFD2"}],          |
|             | "id": 51}}}, "description": "Sample VNFFG template", "groups": {"VNFFG1": {"type":                     |
|             | "tosca.groups.nfv.VNFFG", "description": "HTTP to Corporate Net", "members":                           |
|             | ["Forwarding_path1"], "properties": {"vendor": "tacker", "connection_point": ["CP12", "CP22"],         |
|             | "version": 1.0, "constituent_vnfs": ["VNFD1", "VNFD2"], "number_of_endpoints": 2,                      |
|             | "dependent_virtual_link": ["VL12", "VL22"]}}}}, "tosca_definitions_version":                           |
|             | "tosca_simple_profile_for_nfv_1_0_0"}}                                                                 |
| tenant_id   | d505bc8566ed42a6969779c86c000c6a                                                                       |
+-------------+--------------------------------------------------------------------------------------------------------+
```

Figura 1.31 – Criação de um VNFFGD ao catálogo de VNFFG via CLI

Passo 4: Criação das VNFs

Para a criação das VNFs foram usados dois *scripts* em TOSCA, VNFD1 e VNFD2. As VNFs representam as funções de rede que estarão entre a fonte e o destino. Cada VNF com funcionalidade diferente terá seu próprio descritor.

As VNFs neste cenário tem a funcionalidade de encaminhador de pacotes L3, possuindo apenas um *Connect Point* cada, com o *flavor m1.tiny*. Semelhante ao VNFFGD, as VNFDs podem ser instancias no catálogo de VNFs via CLI ou dashboard. Na Figura 1.32 é mostrado como um descritor de VNF é instanciado no catálogo.

```
stack@ubuntu:~$ tacker vnfd-create VNFD1 --vnfd-file vnfd1.yaml
Created a new vnfd:
+-----------------+--------------------------------------+
| Field           | Value                                |
+-----------------+--------------------------------------+
| created_at      | 2017-10-30 17:13:05.134048           |
| description     | Demo example                         |
| id              | 46f8c496-21bd-4ca8-b91f-09bf92faf890 |
| name            | VNFD1                                |
| service_types   | vnfd                                 |
| template_source | onboarded                            |
| tenant_id       | d505bc8566ed42a6969779c86c000c6a     |
| updated_at      |                                      |
+-----------------+--------------------------------------+
```

Figura 1.32 – Instanciação de um descritor de VNF no catálogo

A partir dos descritores de VNFs no catálogo, podemos instanciar as VNFs. A Figura 1.33 mostra como é feita a instanciação da VNF1 a partir do descritor VNFD1 que está armazenada no catálogo via CLI.

```
stack@ubuntu:~$ tacker vnf-create VNF1 --vnfd-name VNFD1
Created a new vnf:
+----------------+--------------------------------------+
| Field          | Value                                |
+----------------+--------------------------------------+
| created_at     | 2017-10-30 17:14:50.697825           |
| description    | Demo example                         |
| error_reason   |                                      |
| id             | ca7e66c0-3d6d-4a26-921b-c149a8ba4833 |
| instance_id    | 73007be5-5164-47b4-93b6-23a9430bca90 |
| mgmt_url       |                                      |
| name           | VNF1                                 |
| placement_attr | {"vim_name": "VIM0"}                 |
| status         | PENDING_CREATE                       |
| tenant_id      | d505bc8566ed42a6969779c86c000c6a     |
| updated_at     |                                      |
| vim_id         | 44a8a83d-6727-49fd-bccc-d5f482d80d7a |
| vnfd_id        | 46f8c496-21bd-4ca8-b91f-09bf92faf890 |
+----------------+--------------------------------------+
```

Figura 1.33 – Instanciação da VNF1 via CLI

Passo 5: Criação da VNFFG

Para a criação da VNFFG é necessário que as VNFs descritas dentro do VNFFGD estejam instanciadas. É possível instanciar a VNFFG via CLI, conforme descrito na Seção 1, ou via *dashboard*, conforme a Figura 1.34.

Para instanciar uma VNFFG são necessários o descritor de VNFFG, um mapeamento das VNFs e a habilitação ou não da simetria. O mapeamento é usado para declarar a ordem das VNFs que serão usadas no SFC e a simetria é usada par aindicar se o caminho de volta

Figura 1.34 – Instanciação de uma VNFFG

será o inverso do caminho de ida. A Figura 1.35 mostra o VNFFG instanciado com o nome SFC e ativo no OpenStack.

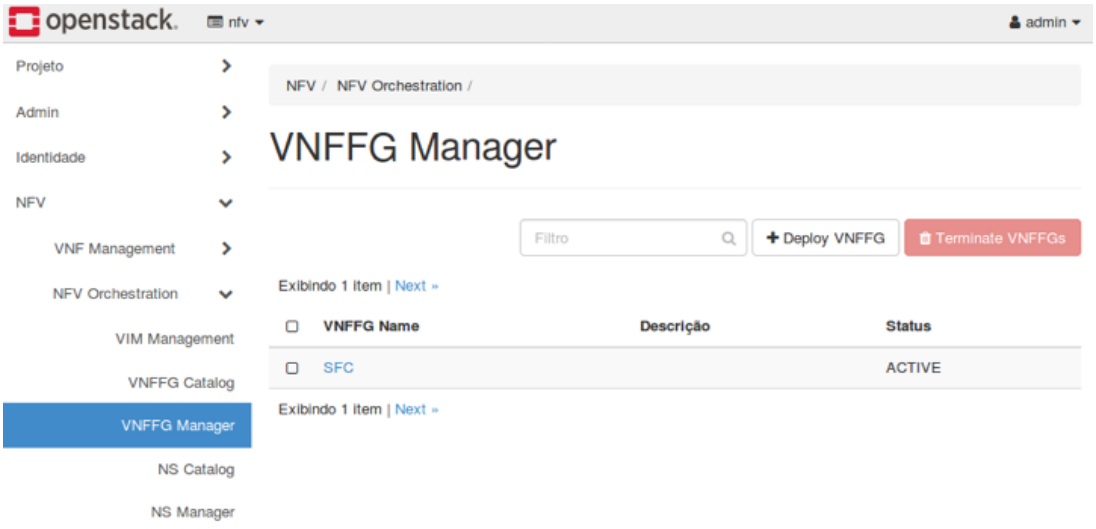


Figura 1.35 – VNFFG instanciada no VNFFG *Manager*

Passo 6: Interagindo com o SFC

O experimento está montado, necessitando apenas de adicionar as funcionalidades de encaminhador de pacotes nas VNFs. O comandos abaixo foi usado no *kernel* em ambas VNFs para habilitar a funcionalidade de encaminhador de pacotes L3.

Código 1.13 – Habilitação do encaminhador de pacotes L3

```
sudo —i
echo 1 > /proc/sys/net/ipv4/ip_forwarding
```

Para fazer a verificação do experimento, foram abertas 4 janelas no kernel representando cada interface de rede usada no experimento. Pode-se observar o nome de cada interdace utilizada utilizando o comando *ifconfig | grep "tap"* acessando o Devstack via CLI. As interfaces *tap* são portas virtuais da *Linux Bridge* usadas no OpenStack. Foi utilizado o comando *tcpdump* nas interfaces das VMs e das VNFs para observar o tráfego de pacotes que passam por elas.

Ao Executar o comando *ping*, é possível observar que os pacotes saem da VM origem e vai para a VM destino, obrigatoriamente, passando por VNF1 e VNF2. É possível observar duas mensagens *echo request* em VNF1 e VNF2, na qual representam que o pacote entrou na VNF, foi processado e saiu pela mesma interface de rede. Após o pacote chegar na VM destino, a mensagem de retorno *echo reply* é encaminhada diretamente para a VM fonte passando apenas pelo roteador, conforme descrito no esboço do experimento.

Conclusão

Este minicurso apresentou os conceitos relacionados ao paradigma NFV, a utilização de funções de redes virtualizadas, a partir da plataforma de nuvem OpenStack, juntamente com o modulo Tacker, responsável por habilitar o NFV. Após a fundamentação prática, o minicurso partiu para a apresentação da plataforma OpenStack, principalmente no conhecimento da *dashboard*, o Horizon.

Posteriormente, tratando efetivamente sobre NFV, foi explicado os campos necessários de um *script* TOSCA, para inserir uma função de rede virtualizada no OpenStack. Por fim, foram apresentados as funcionalidades básicas, como a gerencia (inserir e deletar uma VNF, realizar um monitoramento, etc.), a escalabilidade, provendo um cenário de alta disponibilidade, e também a funcionalidade de encadeamento de funções de rede, o SFC.

Partindo das funcionalidades apresentadas durante o minicurso, abre-se uma gama de possibilidades para criar outros cenários (mais complexos) utilizando o OpenStack e suas funcionalidades de suporte à orquestração de VNFs. Assim, motiva-se o desenvolvimento de novos serviços no âmbito da computação em nuvem.

Agradecimentos

Este trabalho recebeu financiamento do projeto GT-NosFVeraTO, por meio da RNP sob o contrato de no. RNP/FEST 002917, e do projeto FUTEBOL sob o contrato de no. MCTIC/RNP/FEST 688941. Além disso, os autores gostariam de agradecer o financiamento parcial proveniente de recursos de bolsas CNPq, CAPES e FAPES.

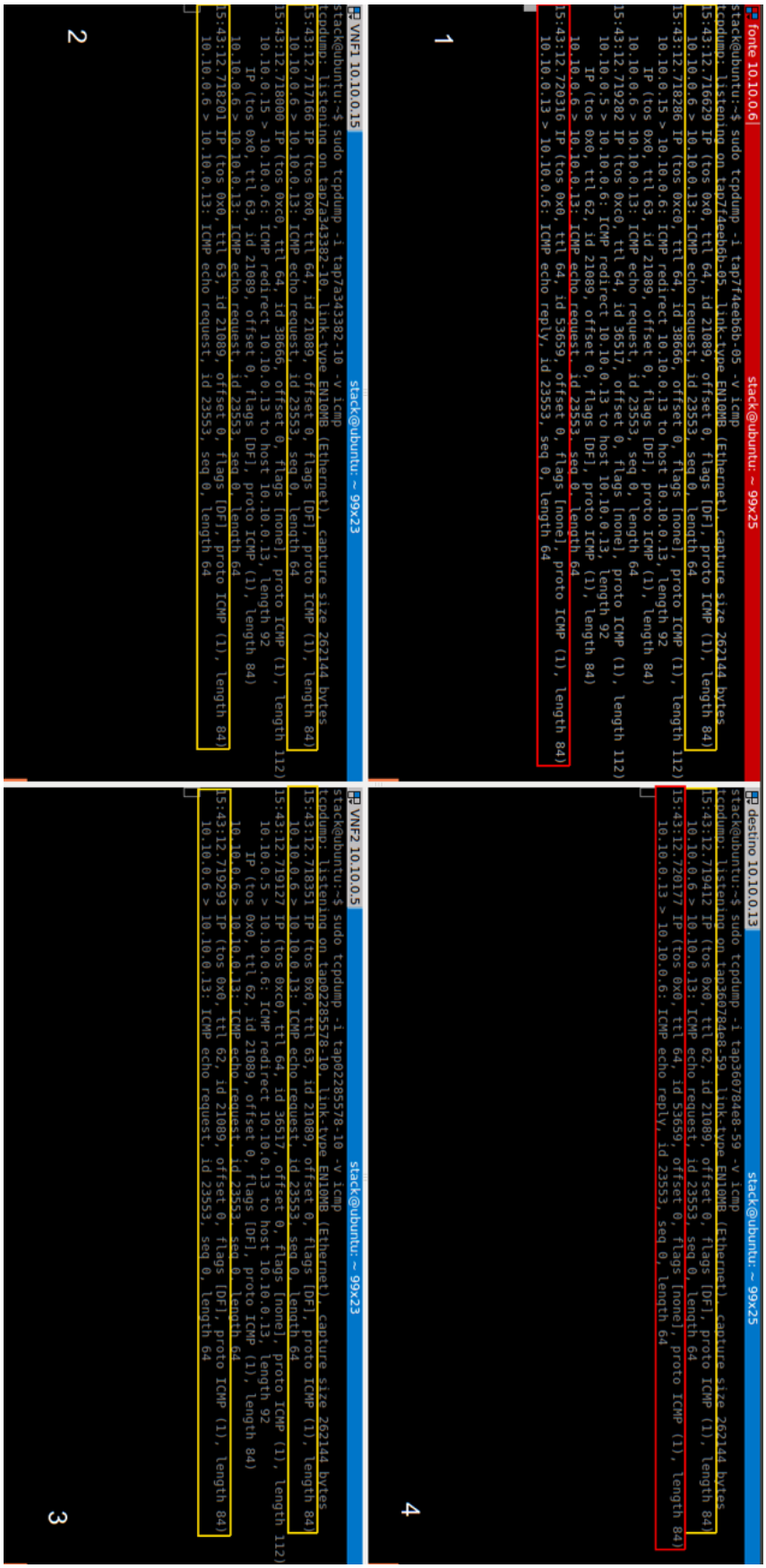


Figura 1.36 – Teste do SFC

Referências Bibliográficas

[1] Project Openstack. Openstack documentation. *http://docs.openstack.org/*, 2018-20-07.

[2] Wiki Tacker. Tacker - openstack nfv orchestration. *https://wiki.openstack.org/wiki/Tacker*, 2018-20-07.

[3] C Cui et al. Network functions virtualisation: Network operator perspectives on industry progress. white paper no. 3, issue 1. In *SDN and OpenFlow World Congress, Dusseldorf-Germany*, 2014.

[4] Raj Jain and Subharthi Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31, 2013.

[5] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in nfv: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

[6] Margaret Chiosi, Don Clarke, Peter Willis, Andy Reid, James Feger, Michael Bugenhagen, Waqar Khan, Michael Fargano, Chunfeng Cui, Hui Deng, et al. Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action. In *SDN and OpenFlow World Congress*, volume 48. sn, 2012.

[7] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[8] Mehmet Ersue. Etsi nfv management and orchestration-an overview. In *Proc. of 88th IETF meeting*, 2013.

[9] John Meadows. Delivering public cloud functionality in openstack, 2014.

[10] Jinlin Chen, Yiren Chen, Shi-Chun Tsai, and Yi-Bing Lin. Implementing nfv system with openstack. In *Dependable and Secure Computing, 2017 IEEE Conference on*, pages 188–194. IEEE, 2017.

[11] OPENSTACK. Tacker - openstack nfv orchestration, 2016.

[12] OASIS. Tosca simple profile for network functions virtualization (nfv), 2016.

[13] Jokin Garay, Jon Matias, Juanjo Unzilla, and Eduardo Jacob. Service description in the nfv revolution: Trends, challenges and a way forward. *IEEE Communications Magazine*, 54(3):68–74, 2016.

[14] W TOSCA-PARSER. Parser for tosca simple profile in yaml, 2015.

[15] B. F CLOUDS. Swimming in brahmaputra with opnfv parser, 2016.

[16] OPENSTACK. Vnf descriptor template guide, 2018.

[17] Project Tacker. Tacker templates, 2017.

[18] NFVISG ETSI. Gs nfv-man 001 v1. 1.1 network function virtualisation (nfv); management and orchestration, 2014.

CAPÍTULO

2 Processamento de Sinais sobre Grafos: Fundamentos e Aplicações

Juliano B. Lima (UFPE), Wallace A. Martins (UFRJ), Guilherme B. Ribeiro (UFPE), Vitor Rosa M. Elias (UFRJ)

Introdução

Dados de variáveis multidimensionais definidos sobre estruturas de rede são a todo instante gerados, armazenados e processados em sistemas relacionados às diversas áreas da engenharia e da tecnologia. Medições em um conjunto de sensores e dispositivos de Internet das Coisas [1, 2, 3, 4], número de citações em uma rede de colaboração científica ou relações em mídias sociais (*collaboration graph*, ou *social graph*) [5], interações entre indivíduos em um ecossistema (*ecological networks*) [6], são alguns exemplos de aplicações em que os dados obtidos estão intimamente ligados à topologia da rede sobre a qual estão definidos.

Estes sistemas, que geram dados multivariados, têm crescido exponencialmente em número, com a miniaturização e barateamento de diversos tipos de sensores e a consolidação de conceitos como *cloud storage/computing* e *Big Data* [7]. Aliada à abundância dos dados, está a importância da informação que deles pode ser extraída; o processamento adequado destes dados massivos será um requisito fundamental, por exemplo, para o bom desempenho de empresas na economia e para a continuidade da pesquisa em *smart cities*, que busca aproveitar o imenso fluxo de dados gerados nas cidades para prover soluções para problemas urbanos [8].

Todos esses exemplos trazem sistemas cuja estrutura pode ser modelada por um grafo a cujos vértices estejam associadas variáveis de interesse, como na Fig. 2.1 [9]. Com essa ótica, surgiu e tem se desenvolvido o *processamento de sinais sobre grafos* (GSP, do inglês *graph signal processing*), um ferramental teórico criado para estender métodos clássicos de processamento de sinais para casos em que o domínio é irregular, representado por um

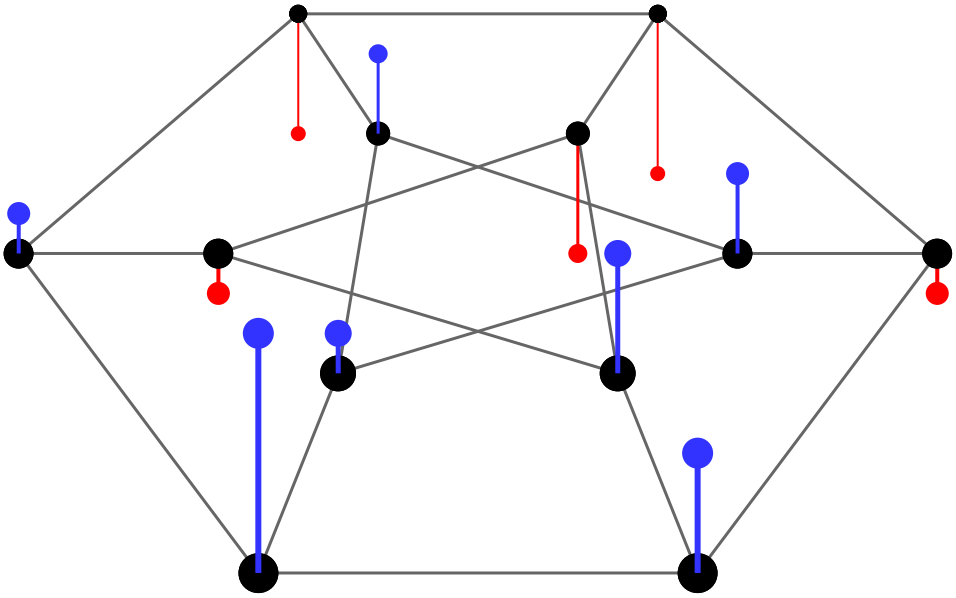


Figura 2.1 – Exemplo de sinal definido sobre um grafo. As arestas do grafo representam relações de dependência, ou semelhança, entre as amostras do sinal.

grafo arbitrário. Neste cenário, duas grandes abordagens ganharam corpo ao longo dos anos. Uma delas baseia-se em processamento algébrico de sinais, utilizando a matriz de adjacência ponderada do grafo como operador de deslocamento unitário, o bloco elementar da álgebra. Esta teoria considera sinais definidos tanto em grafos direcionados como não-direcionados, e com pesos de valores reais ou complexos [10]. A teoria espectral de grafos serviu de base para fundamentar a segunda abordagem, que utiliza a matriz Laplaciana do grafo para construir uma base para o espaço de sinais [11]. Este ramo de GSP analisa sinais definidos sobre grafos não-direcionados com pesos reais não-negativos.

A definição do domínio de um sinal como sendo um grafo leva a dificuldades em conceitos fundamentais de processamento de sinais. A translação unitária de um sinal $x[n]$ para a direita, por exemplo, é feita em DSP[1] por uma simples substituição de variáveis $x[n-1]$. Entretanto, o sentido de direita e esquerda é inexistente para grafos em geral. Uma possibilidade ingênua seria rotular os N vértices do grafo, de v_0 até v_{N-1}, definindo a amostra $x[n]$ sobre o vértice v_n; o sinal deslocado de uma unidade seria definido como o resultado da atribuição de cada amostra $x[n]$ do sinal original ao vértice $v_{(n-1) \bmod N}$. Tal opção, no entanto, tem repetibilidade dependente da rotulação dos vértices [11]. Isso ilustra como conceitos tão simples em DSP podem merecer estudo e atenção particulares em GSP.

Este capítulo pretende ser uma introdução útil e clara para o leitor à área de processamento de sinais sobre grafos. As duas seções seguintes apresentam a terminologia de teoria dos grafos e conceitos iniciais envolvendo a definição de sinais sobre grafos. Na sequência, a quarta e quinta sessões conduzem o leitor através da construção dos dois principais ferramentais teóricos de GSP, que são postos em prática nos exemplos e aplicações apresentados na seção 2. A sexta seção discute os conceitos de amostragem e reconstrução

[1] Acrônimo de *digital signal processing*, utilizado para se referir à teoria clássica de processamento digital de sinais.

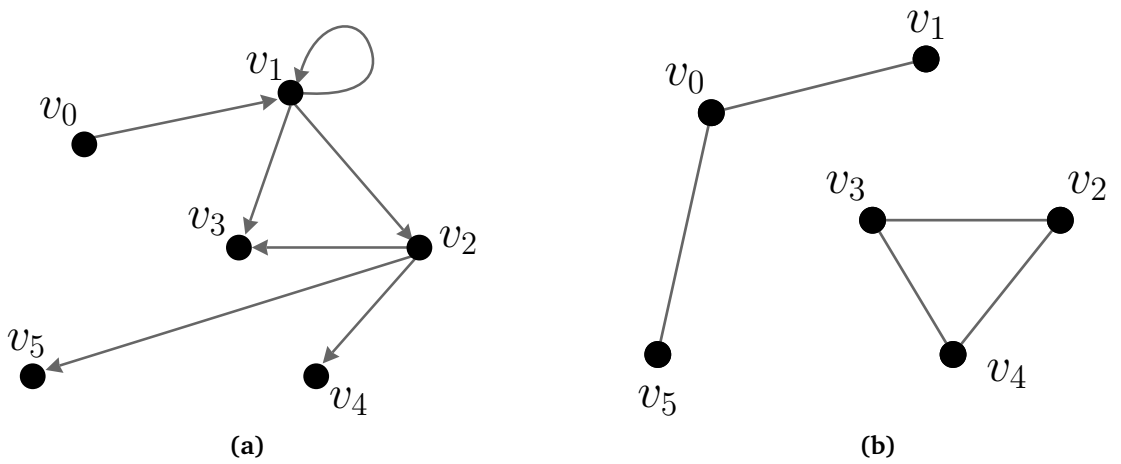


Figura 2.2 – Exemplos de grafos (a) direcionado e (b) não-direcionado, definidos sobre o mesmo conjunto de vértices.

de sinais sobre grafos, com especial atenção para métodos de estimação adaptativa. A seguir, são apresentados comentários sobre implementação das ferramentas de GSP em *software*, guiando o leitor pelos principais pacotes criados para MATLAB e orientando como utilizar funções e bibliotecas em Python para este fim. Na conclusão deste capítulo há comentários acerca de oportunidades de estudo na área.

Teoria dos Grafos: Apresentando a Terminologia

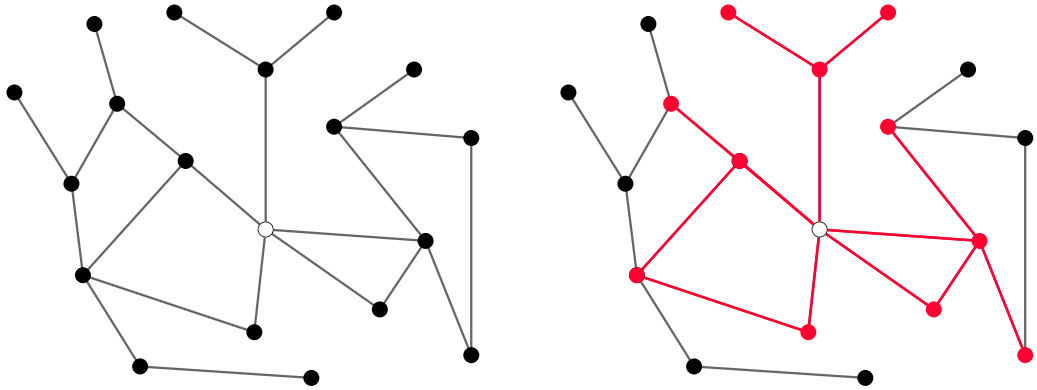
Um *grafo* é definido em sua forma mais geral como sendo o par $(\mathcal{V}, \mathcal{E})$, onde $\mathcal{V}$ é um conjunto cujos elementos chamam-se *vértices* e $\mathcal{E}$ (conjunto das *arestas*) é um subconjunto de $\mathcal{V}^2$ [12]. Para os propósitos de GSP, convém definir um grafo como sendo a estrutura $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ cujas relações entre os $|\mathcal{V}| = N$ vértices são capturadas pelas entradas da matriz de adjacência ponderada $\mathbf{A}$: se $A_{i,j} \neq 0$, então o vértice $v_j \in \mathcal{V}$ influencia o vértice (*adjacente*) $v_i \in \mathcal{V}$ e diz-se que há uma aresta de v_j para v_i com peso $A_{i,j}$. Por sua própria definição, $\mathbf{A}$ é uma matriz quadrada de dimensões $N \times N$. A soma dos pesos das arestas incidentes em um vértice v_i é o *grau de entrada* (*indegree*) de v_i, ou d_i^-; a soma dos pesos das arestas que partem de v_i é o *grau de saída* (*outdegree*) d_i^+.

O grafo é *direcionado* (ou dirigido, orientado, ou simplesmente um digrafo) se e somente se a matriz $\mathbf{A}$ não é simétrica. Caso contrário, o grafo é dito *não-direcionado*, e o *grau* do vértice v_i é definido como $d_i = d_i^+ = d_i^-$. A Fig. 2.2 mostra exemplos de grafos direcionado e não-direcionado.

A *matriz Laplaciana* de um grafo é dada por $\mathbf{L} = \mathbf{D} - \mathbf{A}$, com $\mathbf{D}$ sendo a *matriz de grau* do grafo, uma matriz diagonal com a i-ésima entrada igual ao grau do vértice v_i.

Chama-se *laço* uma aresta que *parte de* e *chega a* um mesmo vértice. Fala-se em *múltiplas arestas* sempre que a um único par de vértices estão associadas duas ou mais arestas. Um grafo não-direcionado é dito *simples* se ele não possui laços nem múltiplas arestas.

Um grafo *completo* de N vértices é aquele em que quaisquer dois vértices tomados



(a) (b)

Figura 2.3 – (a) Um grafo; (b) em vermelho, o conjunto $\mathcal{N}(i,2)$ em que v_i é o vértice branco.

em $\mathcal{V}$ são adjacentes. Um grafo é *conexo* quando, para quaisquer dois vértices $v_i, v_j \in \mathcal{V}$, existe um *caminho* (conjunto de arestas distintas com vértices em comum) ligando v_i a v_j. Equivalentemente, diz-se que um grafo é conexo se e somente se, para todo par $(v_i, v_j) \in \mathcal{V}^2$, existe uma sequência de inteiros menores ou iguais a N, como $(i,k,\ell,m,\ldots,n,j)$, tal que as entradas $A_{i,k}, A_{k,\ell}, A_{\ell,m}, \ldots, A_{n,j}$ de $\mathbf{A}$ são todas não nulas.

Grafos *ponderados* são aqueles em que a cada aresta associa-se um valor numérico (um *peso*); em grafos *não-ponderados*, as entradas não nulas da matriz de adjacência são todas unitárias. Um grafo é dito cíclico se ele possuir algum caminho fechado e acíclico caso contrário.

Um *subgrafo* de $\mathcal{G}$ é um grafo $\mathcal{G}'$ cujos vértices e arestas formam subconjuntos dos conjuntos de vértices e arestas de $\mathcal{G}$. Uma *componente conexa* de um grafo $\mathcal{G}$ é um subgrafo conexo de $\mathcal{G}$ (vide, por exemplo, o grafo da Fig. 2.2, que possui 2 componentes conexas).

Pode-se definir a vizinhança de um vértice v_i como $\mathcal{N}_i$, o conjunto de vértices adjacentes a v_i. Denota-se por $\mathcal{N}(i,K)$ o conjunto de vértices conectados a v_i por um caminho com K ou menos arestas, como uma vizinhança dentro de um raio K (vide Fig. 2.3).

Para uma introdução mais completa à Teoria dos Grafos, os autores recomendam as referências [12, 13, 14, 15].

Definindo o Sinal e seu Domínio

Um sinal $\mathbf{s}$ sobre um grafo $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$, com $|\mathcal{V}| = N$, é definido como uma função discreta que mapeia $\mathcal{V}$ em um conjunto de grandezas escalares, usualmente os números complexos ou reais,

$$s: \mathcal{V} \to \mathbb{C} \mid s(v_i) = s_i, \tag{2.1}$$

de modo que um sinal definido sobre $\mathcal{G}$ é um vetor em $\mathbb{C}^N$ *indexado pelos vértices de* $\mathcal{G}$. Uma vez que se determina uma rotulação e ordem específica para os elementos de $\mathcal{V} = \{v_1, \ldots, v_N\}$, não há ambiguidade em representar o sinal como o vetor $\mathbf{s} = (s_0\ s_1\ \ldots\ s_{N-1})^T$,

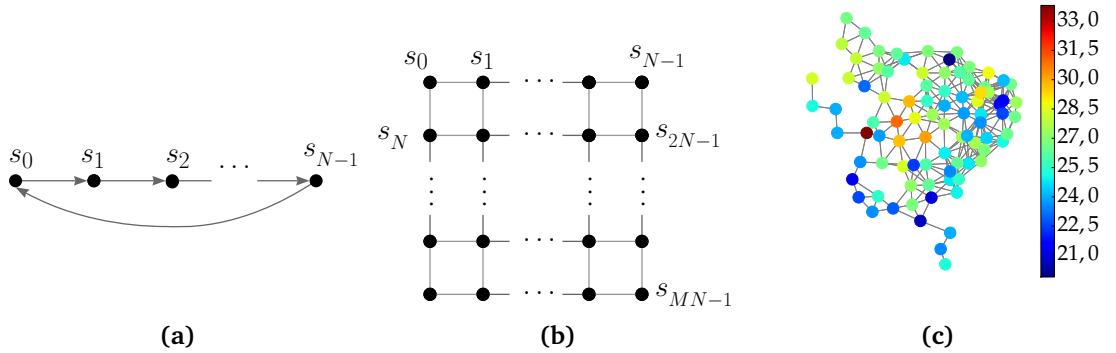


Figura 2.4 – Representações de sinais sobre (a) um grafo em anel direcionado, (b) um grafo em grade retangular uniforme e (c) um grafo formado por cidades do Nordeste brasileiro.

$s_i \in \mathbb{C}$, $0 \leq i \leq N-1$. Representa-se por $\mathscr{S}$ o espaço de sinais definidos sobre um certo grafo $\mathscr{G} = \{\mathscr{V}, \mathbf{A}\}$.

A Fig. 2.4 traz exemplos de representações de sinais sobre grafos, nos quais a rotulação dos vértices é implícita (associa-se a amostra s_i ao vértice v_i). Os valores das amostras dos sinais podem ser indicados *numericamente*, junto aos vértices (Figs. 2.4a e 2.4b), ou por meio de uma escala de cores (Fig. 2.4c). Ao longo deste capítulo, será priorizado o uso desta última representação.

Um sinal de comprimento finito e de tempo discreto é modelado pelo grafo em anel direcionado (considera-se os pesos unitários), como mostrado na Fig. 2.4a: a noção de evolução temporal é capturada pelas arestas direcionadas; a periodicidade imposta pelas condições de fronteira da análise de Fourier de tempo discreto é modelada pela aresta realimentando a última amostra à primeira. O grafo na Fig. 2.4b é um modelo para imagens digitais [16] chamado *nearest-neighbor*, em que a dependência entre pixels é aproximada para existir apenas entre vizinhos, e o grafo na Fig. 2.4c é um exemplo de grafo de rede de sensores[2], com pesos das arestas dados pelo inverso da distância euclidiana, sobre o qual definiu-se o sinal da temperatura à meia-noite de 01 de fevereiro de 2012 em cidades do Nordeste brasileiro[3].

As características espectrais de um sinal dependem fortemente do domínio sobre o qual ele é definido. Em geral, diz-se que um sinal contém majoritariamente baixas frequências se amostras *adjacentes* têm valores *próximos*, e altas frequências se têm valores *díspares*. Considerando sinais definidos sobre grafos, fica evidente que a noção de amostras adjacentes depende da topologia do grafo em questão, e portanto *um mesmo sinal pode apresentar espectros distintos se definido sobre grafos diferentes*. A Fig. 2.5 confirma essa intuição, apresentando o espectro de um sinal segundo a base de autovetores da matriz Laplaciana do grafo, para dois grafos distintos; na Fig. 2.5c, as amostras de maior valor são

[2]Quando não é especificado em contrário, chama-se *grafo de (rede de) sensores* um grafo conectado com vértices distribuídos uniformemente numa região do espaço euclidiano bidimensional.

[3]Fonte: Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP) do Instituto Nacional de Meteorologia. Acesso gratuito, disponível em: `http://www.inmet.gov.br/portal/index.php?r=bdmep/bdmep`

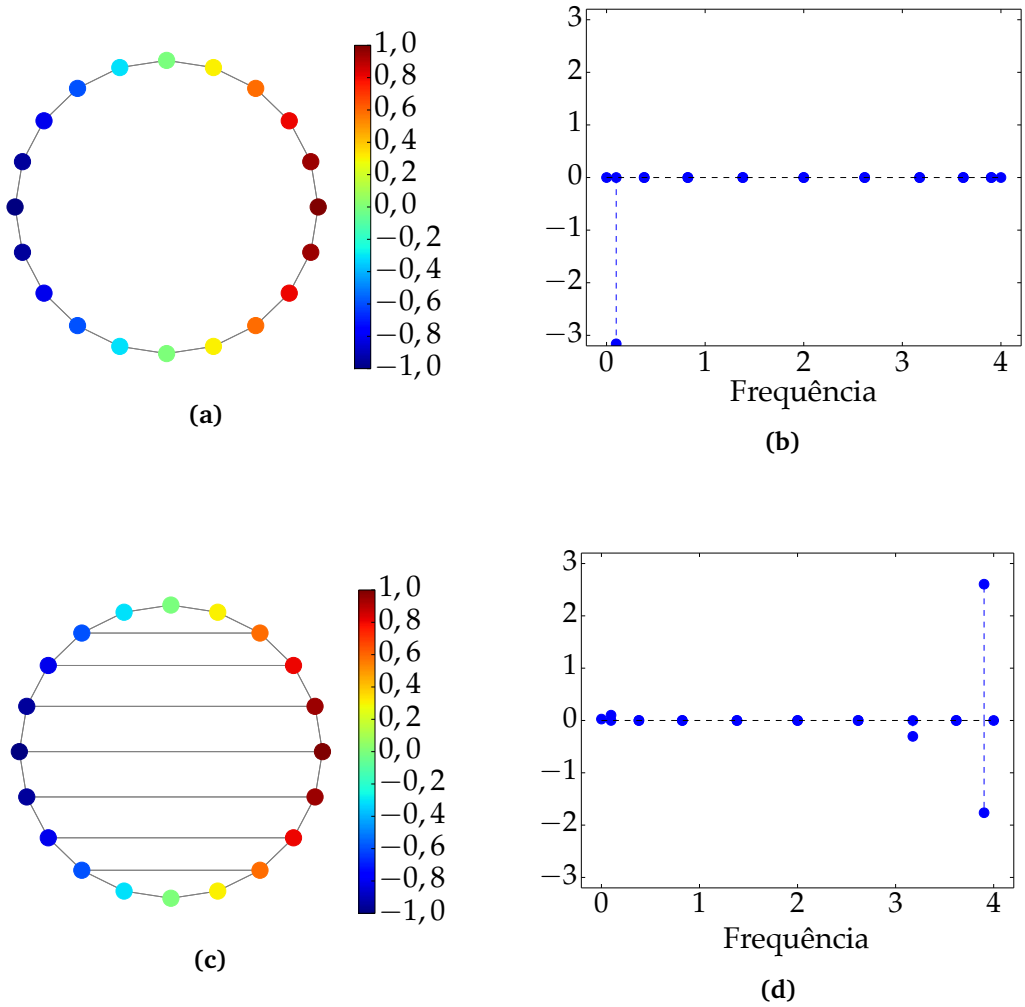


Figura 2.5 – Um mesmo sinal definido sobre (a) um grafo não-direcionado em anel e (c) um grafo derivado do primeiro, mas com topologia distinta. (b) Espectro de Fourier, segundo GSP_L, do sinal sobre o grafo em *(a)* e (d) em *(c)*.

adjacentes àquelas de menor valor, o que causa componentes de maior frequência do que se o grafo usado como domínio fosse um anel a pesos constantes (Fig. 2.5a).

Inferindo grafos

Em algumas situações em GSP, dispõe-se de um conjunto de dados, um ou mais sinais, mas não se tem o grafo subjacente. Informações sobre o contexto da aplicação e o fenômeno que gerou o sinal, no entanto, geralmente sugerem um ou outro método adequado para se estimar o grafo em questão.

Para grafos de sensores (não-direcionados), por exemplo, como na Fig. 2.4c, ponderar as arestas utilizando uma distribuição gaussiana sobre a distância euclidiana,

$$A_{i,j} = \exp\left(\frac{-\text{dist}^2(v_i, v_j)}{2\theta^2}\right), \tag{2.2}$$

é muitas vezes adequado. Um limiar T é escolhido tal que, se $\text{dist}(v_i, v_j) > T$, então faz-se $A_{i,j} = 0$, e os parâmetros T e θ são determinados segundo indicar a aplicação.

No entanto, o uso do critério em (2.2) pode levar a um compromisso entre manter o grafo conexo e obter uma matriz de adjacência esparsa. Se a distância entre um vértice e seu vizinho mais próximo for muito maior do que a distância média entre vértices e seus vizinhos mais próximos, fazer com que o grafo seja conexo implica impor um limiar T grande, o que produziria muitas arestas. Para contornar esse problema, obtendo um grafo que capture razoavelmente as inter-relações entre os vértices, uma alternativa é ligar um vértice aos K mais próximos, ponderando adequadamente as arestas.

Essas técnicas supõem que há uma métrica de distância adequada para avaliar a similaridade entre as amostras de um par de vértices. Mas, dada a diversidade de possibilidades de sinais e aplicações, estimar a topologia do grafo subjacente a certo conjunto de dados já constitui um desafio em si [9, 17].

GSP_A: Grafos e o Processamento Algébrico de Sinais

Em 2006, Püschel e Moura publicaram sua teoria de processamento algébrico de sinais (ASP, do inglês *algebraic signal processing*) [18]. A descrição de sinais e filtros do ponto de vista algébrico é intencionalmente abstrata e genérica: tanto os filtros como os sinais são simplesmente elementos de espaços vetoriais dotados de multiplicação e distributividade. Especificamente, o espaço de filtros é uma *álgebra*, sobre a qual toma-se um *módulo* para ser o espaço de sinais. Uma descrição detalhada sobre ASP foge ao escopo deste capítulo, mas o leitor interessado pode tirar bastante proveito dos trabalhos originais de Püschel e Moura [19, 20].

O aspecto de ASP que levou ao processamento sobre grafos, o que de fato nos interessa diretamente, é que o espaço de filtros é *gerado* pelo operador de atraso unitário de sinais. Um exemplo tornará a afirmação menos obscura: seja a álgebra polinomial $\mathbb{C}[x]/(x^N-1)$, ou $\mathscr{A} = \{\sum_{\ell=0}^{N-1} a_\ell x^\ell | a_\ell \in \mathbb{C}\}$, que consiste em todos os polinômios de coeficientes complexos com grau menor do que N, dotados da adição e multiplicação polinomiais usuais, módulo x^N-1. É comum representar sinais e filtros de tempo discreto e comprimento N como elementos de $\mathscr{A}$, caso em que a convolução cíclica torna-se a simples multiplicação polinomial modular. Por exemplo, o sinal $\mathbf{s}_1 = (0 \;\; 2 \;\; 1 \;\; 0)$ é representado por $s_1(x) = x^2 + 2x$, e sua versão deslocada de uma unidade, $\mathbf{s}_2 = (0 \;\; 0 \;\; 2 \;\; 1)$, é dada por $s_2(x) = x^3 + 2x^2$. Fica claro que, neste caso, o filtro de atraso unitário é $d(x) = x$, cujas potências *geram* a base $(1, x, x^2, \ldots, x^{N-1})$ para o espaço de filtros. Esta relação se mantém para qualquer álgebra em ASP; portanto, Moura certamente concluiu que, ao definir um operador de deslocamento unitário para sinais sobre grafos, isto daria início à construção da teoria de GSP.

Partindo do grafo em anel direcionado com pesos unitários, modelo para o domínio

de tempo discreto, observou-se que a matriz de adjacência do grafo,

$$\mathbf{C} = \begin{bmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{bmatrix}, \tag{2.3}$$

realiza precisamente o papel de atraso num sinal de tempo discreto. Ou seja, se um sinal $\mathbf{s} = (s_1\ s_2\ \ldots\ s_N)^T$ definido em um grafo em anel é multiplicado à esquerda por $\mathbf{C}$, tem-se

$$\mathbf{s}^{\langle 1 \rangle} = \mathbf{C}\mathbf{s}, \tag{2.4}$$

com $\mathbf{s}^{\langle 1 \rangle} = (s_N\ s_1\ \ldots\ s_{N-1})^T$. Generalizando a ideia, Sandryhaila e Moura definiram o *operador de deslocamento unitário como sendo a matriz de adjacência do grafo sobre o qual o sinal está definido* [21]. É por isso que, neste capítulo, a vertente de GSP derivada da teoria de ASP é identificada por GSP_A, indicando o papel fundamental exercido pela matriz de adjacência. Para um sinal $\mathbf{x}$ sobre um grafo qualquer $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$, $\mathbf{A}$ age como um *filtro* de atraso (ou deslocamento) sobre $\mathbf{x}$, e a versão deslocada deste sinal é representada por $\mathbf{x}^{\langle 1 \rangle} = \mathbf{A}\mathbf{x}$.

Filtros sobre grafos

Observar a matriz de adjacência como um filtro levou à definição de um filtro de sinais sobre grafos como sendo qualquer matriz $\mathbf{H} \in \mathbb{C}^{N \times N}$ [22], visto que o produto matriz-vetor sempre resulta num vetor (ou *filtro* × *sinal* = *sinal*). Isso implica que os filtros sobre grafos são sempre lineares, uma vez que a distributividade da multiplicação em relação à adição matricial garante que

$$\mathbf{H}(\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2) = \alpha_1\mathbf{H}\mathbf{x}_1 + \alpha_2\mathbf{H}\mathbf{x}_2. \tag{2.5}$$

A propriedade de invariância no tempo (ou ao deslocamento) requer que $\mathbf{A}\mathbf{H}\mathbf{x} = \mathbf{H}\mathbf{A}\mathbf{x}$, $\forall \mathbf{x}$. Nesse contexto, filtros lineares e invariantes ao deslocamento, ou LSI (do inglês *linear and shift-invariant*), têm a forma de um polinômio $h(\cdot)$ avaliado em $\mathbf{A}$,

$$h(\mathbf{A}) = \sum_{\ell=0}^{L-1} h_\ell \mathbf{A}^\ell, \tag{2.6}$$

com L menor ou igual ao grau do polinômio mínimo $m_\mathbf{A}$ de $\mathbf{A}$ [21, 10]. Assim, filtros LSI são uma série de potências finita no operador de deslocamento, exatamente como ocorre em processamento clássico de sinais de tempo discreto, em que os filtros LTI têm representação em termos de polinômios em z^{-1}, pela transformada Z. Além disso, os filtros LSI formam uma álgebra polinomial $\mathbb{C}[x]/m_\mathbf{A}(x)$.

A transformada de Fourier sobre grafos

Uma vez que a transformada de Fourier de um sinal é a sua projeção em uma base de funções invariantes à filtragem linear e invariante no tempo (LTI, do inglês *linear and time-invariant*) [23], em GSP_A define-se a transformada de Fourier sobre grafos (GFT, do inglês *graph Fourier transform*) como a decomposição de um sinal em termos de uma base de autovetores da filtragem LSI [24].

Seja $\mathbf{A}$ a matriz de adjacência de um grafo de N vértices. Se $\mathbf{A}$ for diagonalizável, tem-se[4]

$$\mathbf{A} = \mathbf{V}\tilde{\mathbf{V}}\mathbf{V}^{-1}, \tag{2.7}$$

em que $\mathbf{V}$ contém os N autovetores de $\mathbf{A}$ em suas colunas, isto é,

$$\mathbf{V} = (\mathbf{v}_0\ \mathbf{v}_1\ \ldots\ \mathbf{v}_{N-1}). \tag{2.8}$$

Como filtros LSI são polinômios em $\mathbf{A}$, as colunas de $\mathbf{V}$ formam uma base de vetores invariantes à filtragem LSI para o espaço de sinais $\mathscr{S}$ sobre o grafo com matriz de adjacência $\mathbf{A}$. Desta forma, um sinal $\mathbf{x} \in \mathscr{S}$ pode ser decomposto em suas componentes na base $\mathbf{V}$ como

$$\begin{aligned}\mathbf{x} &= \widehat{x}_0\mathbf{v}_0 + \cdots + \widehat{x}_{N-1}\mathbf{v}_{N-1} = \mathbf{V}(\widehat{x}_0\ \widehat{x}_1\ \ldots\ \widehat{x}_{N-1})^T \\ &= \mathbf{V}\widehat{\mathbf{x}},\end{aligned} \tag{2.9}$$

e esta é definida como a equação de síntese da *transformada de Fourier sobre grafos*. A equação de análise da GFT é, portanto,

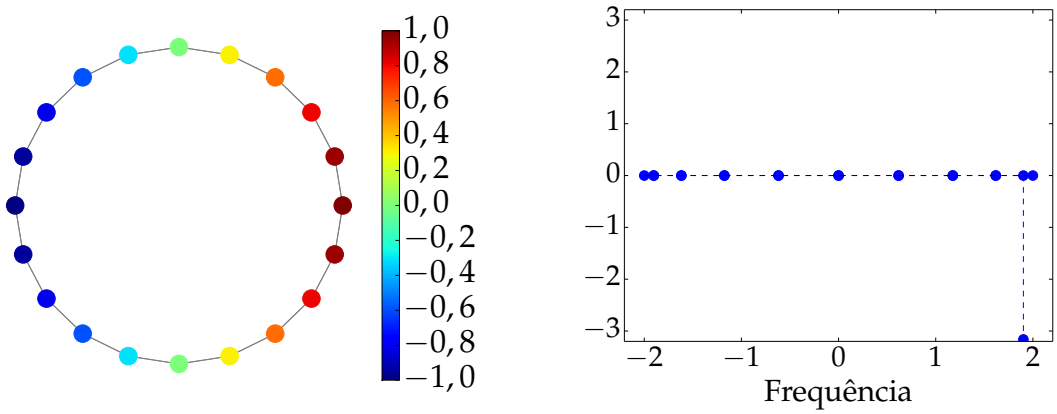
$$\widehat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}. \tag{2.10}$$

Para sinais de tempo discreto, foi assinalado que seu domínio é modelado como um grafo com matriz de adjacência $\mathbf{C}$ em (2.3). Como $\mathbf{C}$ é circulante, ela é diagonalizada pela matriz da transformada discreta de Fourier (DFT, do inglês *discrete Fourier transform*), dada por $\mathbf{F}$, $F_{n,k} = \exp\left(-\jmath\frac{2\pi}{N}nk\right)$. Assim, pode-se escrever

$$\mathbf{C} = \mathbf{F}^{-1}\tilde{}_\mathbf{C}\mathbf{F}, \tag{2.11}$$

em que $\tilde{}_\mathbf{C}$ é uma matriz diagonal com os autovalores de $\mathbf{C}$. Vê-se que a matriz da GFT, para grafos em anel, é $\mathbf{V}^{-1} = \mathbf{F}$, o que resulta na desejável propriedade de que a GFT de sinais de tempo discreto coincide com a DFT, demonstrando consistência com a teoria clássica.

[4] Se $\mathbf{A}$ não for diagonalizável, o raciocínio pode ser repetido utilizando-se a forma canônica de Jordan.



(a) (b)

Figura 2.6 – (a) Sinal sobre um grafo em anel não-direcionado e (b) seu espectro em GSP_A.

O domínio da frequência

A definição empregada para a GFT sugere interpretar os autovetores $\mathbf{v}_i$ da matriz de adjacência como as "componentes de frequência" associadas às *frequências de grafo* representadas pelos autovalores λ_i (como a componente de Fourier $e^{-j\Omega t}$, no domínio do tempo contínuo t, é associada à frequência Ω). A menos que os polinômios característico e mínimo de $\mathbf{A}$ sejam iguais, uma mesma frequência estará associada a duas ou mais componentes linearmente independentes, como ocorreu com o sinal na Fig. 2.6. Na mesma figura, nota-se que, embora o sinal seja visualmente suave, seu espectro possui componentes associadas a autovalores de grande magnitude, o que levanta a questão de definir um critério coerente para se falar em *altas* e *baixas* frequências de sinais sobre grafos.

Para fundamentar matematicamente a noção de frequência no contexto de GSP, partiu-se de uma métrica usual para sinais de tempo discreto: a *variação total*, que calcula a soma das diferenças entre amostras adjacentes de um sinal e, portanto, assume valores maiores para sinais de maior frequência. Sua expressão matemática, para certo sinal de comprimento finito $\mathbf{x} = (x_0 \;\; x_1 \;\; \ldots \;\; x_{N-1})$, é

$$TV(\mathbf{x}) = \sum_{n=0}^{N-1} |x_n - x_{n-1 \bmod N}|. \tag{2.12}$$

De (2.3) e (2.4), vê-se que (2.12) pode ser reescrita em termos da norma ℓ_1[5] como $TV(\mathbf{x}) = \|\mathbf{x} - \mathbf{C}\mathbf{x}\|_1$, utilizando a matriz de adjacência do grafo em anel para realizar o deslocamento cíclico do sinal. Assim, uma generalização da função $TV(\cdot)$ para sinais definidos sobre grafos quaisquer, i. e., a *variação total sobre grafos* para um sinal $\mathbf{s}$ sobre o grafo $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$, foi definida como

$$TV_G(\mathbf{s}) \triangleq \|\mathbf{s} - \mathbf{A}^{norm}\mathbf{s}\|_1, \tag{2.13}$$

[5] A norma ℓ_1 é um caso particular da norma ℓ_p de um vetor $\mathbf{x} \in \mathbb{C}^N$, definida como $\|\mathbf{x}\|_p \triangleq \left(\sum_{k=0}^{N-1} |x_k|^p\right)^{1/p}$. Quando não for explicitamente indicado, a notação sem subscrito $\|\cdot\|$ indica a norma ℓ_2.

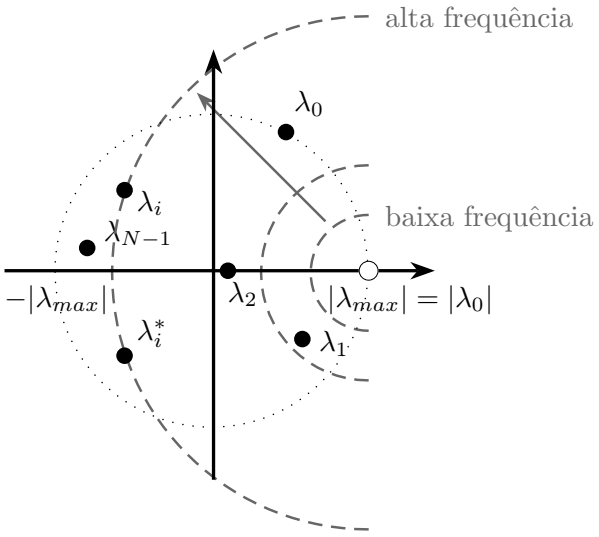


Figura 2.7 – Ordenamento das frequências de sinais sobre grafos, de baixa para alta, no plano complexo. Fonte: adaptado de [25].

com $\mathbf{A}^{norm} = |\lambda_{max}|^{-1}\mathbf{A}$ e λ_{max} o autovalor de $\mathbf{A}$ com maior módulo. A normalização de $\mathbf{A}$ visa evitar a magnificação excessiva das amostras do sinal deslocado [25].

Seja $\mathbf{A}$ diagonalizável e com autovalores (possivelmente complexos) ordenados tais que

$$|\lambda_0| \leq |\lambda_1| \leq \cdots \leq |\lambda_{N-1}| \triangleq |\lambda_{max}|, \tag{2.14}$$

associados aos autovetores $(\mathbf{v}_i)_{i=0,\ldots,N-1}$ escolhidos de modo que $\|\mathbf{v}_k\|_1 = 1$. Tomando a variação total de um autovetor $\mathbf{v}_k$ associado a λ_k, tem-se

$$TV_G(\mathbf{v}_k) = \|\mathbf{v}_k - \mathbf{A}\mathbf{v}_k\|_1 = \|\mathbf{v}_k - \frac{1}{|\lambda_{max}|}\lambda_k\mathbf{v}_k\|_1 = \left|1 - \frac{\lambda_k}{|\lambda_{max}|}\right| \|\mathbf{v}_k\|_1 = \left|\lambda_k - |\lambda_{max}|\right| \frac{\|\mathbf{v}_k\|_1}{|\lambda_{max}|}$$

de forma que, como foi feito $\|\mathbf{v}_k\|_1 = 1$, tem-se a equivalência

$$\left|\lambda_i - |\lambda_{max}|\right| \leq \left|\lambda_j - |\lambda_{max}|\right| \iff TV_G(\mathbf{v}_i) \leq TV_G(\mathbf{v}_j), \tag{2.15}$$

ou seja, componentes de frequência associadas a autovalores mais próximos do ponto $|\lambda_{max}|$ no plano complexo são mais suaves e são, portanto, ditas de *baixa frequência*. A Fig. 2.7 ilustra esse ordenamento para as frequências de grafos, o que esclarece a leitura do espectro do sinal na Fig. 2.6a, cuja matriz de adjacência tem autovalores reais.

Para verificar a consistência desta noção de frequência, tomemos o grafo da Fig. 2.8. O número de mudanças de sinal (número de arestas ligando vértices com amostras de sinal distinto) e a variação total TV_G de cada autovetor da matriz de adjacência do grafo são mostrados na Fig. 2.9. Ambas as métricas visam quantificar a taxa de variação de um sinal sobre o grafo, mas, uma vez que o número de mudanças de sinal ignora as variações que não cruzam o zero, a variação total sobre grafos representa de forma mais fiel a medida de frequência em um sinal. A Fig. 2.9b mostra como $TV_G(\mathbf{v}_k)$ cresce monotonicamente à medida que o respectivo autovalor de $\mathbf{v}_k$, λ_k, se afasta do ponto $|\lambda_{max}|$ no plano complexo.

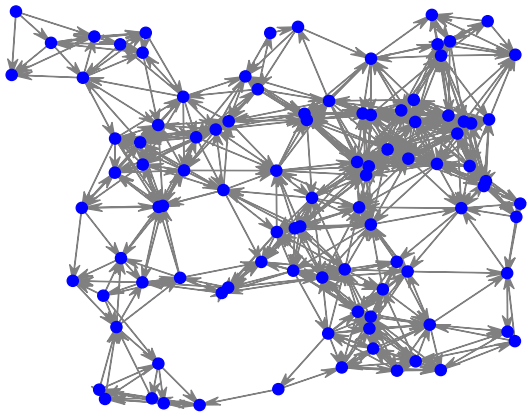
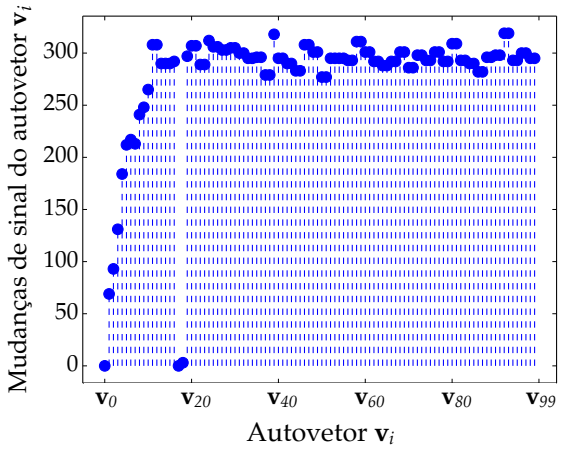
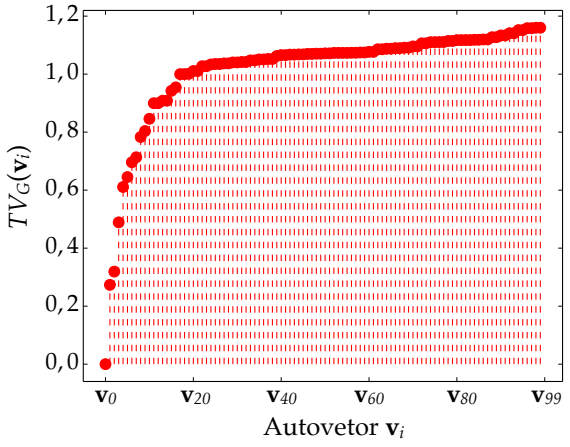


Figura 2.8 – Grafo de sensores direcionado, com 100 vértices, sem laços ou múltiplas arestas.



(a)



(b)

Figura 2.9 – (a) Número de mudanças de sinal e (b) variação total dos autovetores $(\mathbf{v}_i)_{i=0,\ldots,N-1}$ da matriz $\mathbf{A}$, ordenados de modo que os respectivos autovalores se disponham do mais próximo ao mais distante do ponto real $|\lambda_{max}|$ no plano complexo.

Para concluir a análise do domínio da frequência, convém apresentar a resposta em frequência de um filtro LSI sobre grafos. A definição dada na subseção 2 considera a ação de uma matriz sobre um sinal $\mathbf{x}$ no domínio dos vértices de um grafo $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$. Para entender como o filtro age no domínio da GFT, doravante chamado domínio da frequência, será utilizada (2.7) e a representação polinomial de filtros LSI. Seja o filtro $\mathbf{H} = \sum_{\ell=0}^{L} h_\ell \mathbf{A}^\ell$ e sua resposta ao sinal $\mathbf{x}$ dada por

$$\mathbf{H}\mathbf{x} = \sum_{\ell=0}^{L} h_\ell \mathbf{A}^\ell \mathbf{x} = \sum_{\ell=0}^{L} h_\ell \left(\mathbf{V}\tilde{}\mathbf{V}^{-1}\right)^\ell \mathbf{x} = \mathbf{V}\left(\sum_{\ell=0}^{L} h_\ell \tilde{}^{\ell}\right)\mathbf{V}^{-1}\mathbf{x}. \tag{2.16}$$

Tomando a GFT em ambos os lados da última equação, tem-se

$$\mathbf{V}^{-1}\mathbf{H}\mathbf{x} = h(\tilde{})\widehat{\mathbf{x}}, \tag{2.17}$$

de forma que a ação no domínio da frequência a que corresponde a filtragem por $\mathbf{H}$ é a multiplicação pela matriz $h(\tilde{})$, i. e. $h(\tilde{})$ representa a resposta em frequência do filtro $\mathbf{H}$.

GSP_L: a Teoria Espectral de Grafos em GSP

A teoria espectral dos grafos é um ramo de teoria dos grafos que estuda propriedades oriundas da autodecomposição das matrizes de adjacência e Laplaciana de grafos. Com forte base em álgebra linear, ela encontra aplicações em mecânica quântica, redes de comunicações, química etc. [15], e foi utilizada por Ortega, Shuman e outros para desenvolver um ferramental teórico de processamento de sinais sobre grafos distinto de GSP_A [11]; esse ferramental considera, em particular, o espectro da matriz Laplaciana, e por isso é referido aqui por GSP_L. Neste ramo de GSP, os autores costumam restringir seu estudo apenas a grafos não-direcionados e com arestas de peso real não-negativo, como observaram Sandryhaila e Moura [10]. O arcabouço teórico apresentado a seguir é baseado na matriz Laplaciana

$$\mathbf{L} = \mathbf{D} - \mathbf{A}; \tag{2.18}$$

oportunamente, pode-se considerar também a sua versão normalizada, a qual é largamente utilizada na área de teoria espectral de grafos e é expressa por

$$\mathscr{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}. \tag{2.19}$$

Uma importante propriedade da matriz $\mathbf{L}$ para grafos não-direcionados e ponderados pode ser derivada a partir de sua decomposição em termos da *matriz de incidência orientada*, aqui denotada por $\mathbf{B}$ (são considerados grafos sem laços ou múltiplas arestas).

Definição 1 (Orientação de um grafo)**.** *Seja o grafo não-direcionado $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$. Uma orientação de $\mathcal{G}$ é um grafo $\mathcal{G}' = \{\mathcal{V}, \mathbf{A}'\}$ onde a cada aresta de $\mathcal{G}$ foi imposta uma direção qualquer.*

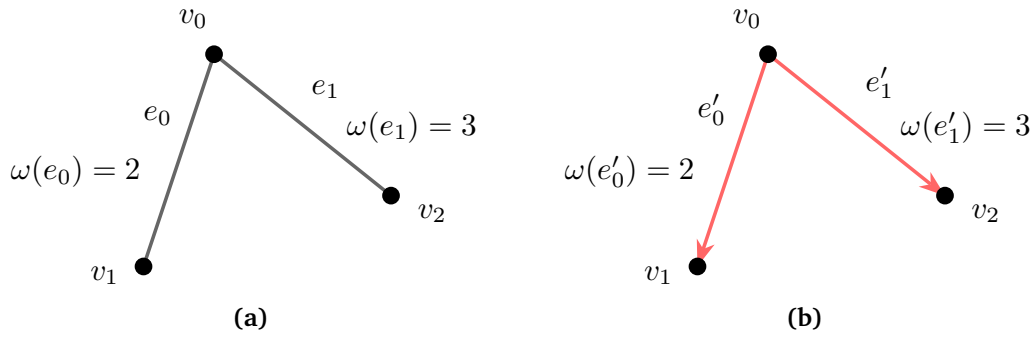


Figura 2.10 – (a) Um grafo não-direcionado, ponderado e conexo, e (b) uma possível orientação sua. A função $\omega : \mathcal{V}^2 \to \mathbb{R}^+$ mapeia a aresta e no valor real não-negativo $\omega(e)$.

Definição 2 (Matriz de incidência orientada)**.** *Seja o grafo não-direcionado $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$, com $|\mathcal{V}| = N$ vértices e $|\mathcal{E}| = E$ arestas ponderadas segundo a função $\omega : \mathcal{E} \to \mathbb{R}^+$. Seja, ainda, $\mathcal{G}' = \{\mathcal{V}, \mathbf{A}'\}$ uma orientação de $\mathcal{G}$. Uma matriz de incidência orientada $\mathbf{B} \in \mathbb{R}^{N \times E}$ de $\mathcal{G}$ tem entradas dadas por*

$$B_{i,j} \triangleq \begin{cases} \sqrt{\omega(e'_j)}, & \text{se } e'_j \text{ chega em } v_i \\ -\sqrt{\omega(e'_j)}, & \text{se } e'_j \text{ parte de } v_i \\ 0, & \text{c.c.} \end{cases} \tag{2.20}$$

É importante notar que, no contexto de GSP_L, no qual os grafos têm pesos reais não-negativos, a matriz de incidência orientada $\mathbf{B}$ tem entradas sempre reais. O Teorema 1 [26, Proposição 2.3] traz a decomposição da matriz Laplaciana em termos da matriz de incidência orientada.

Teorema 1. *Para um dado grafo não-direcionado, com matriz de incidência orientada $\mathbf{B}$, a matriz Laplaciana é dada por*

$$\mathbf{L} = \mathbf{B}\mathbf{B}^T. \tag{2.21}$$

Exemplo 1. *Para ilustrar a decomposição da matriz Laplaciana para grafos simples ponderados, em termos da matriz de incidência de uma orientação do grafo, usemos o grafo da Fig. 2.10a, do qual uma possível orientação é mostrada na Fig. 2.10b. A matriz de incidência para esta orientação é*

$$\mathbf{B} = \begin{matrix} \\ v_0 \\ v_1 \\ v_2 \end{matrix} \begin{matrix} \begin{matrix} e_0 & e_1 \end{matrix} \\ \begin{bmatrix} -\sqrt{2} & -\sqrt{3} \\ \sqrt{2} & 0 \\ 0 & \sqrt{3} \end{bmatrix} \end{matrix}, \tag{2.22}$$

de modo que

$$\mathbf{B}\mathbf{B}^T = \begin{bmatrix} -\sqrt{2} & -\sqrt{3} \\ \sqrt{2} & 0 \\ 0 & \sqrt{3} \end{bmatrix} \begin{bmatrix} -\sqrt{2} & \sqrt{2} & 0 \\ -\sqrt{3} & 0 & \sqrt{3} \end{bmatrix} = \begin{bmatrix} 5 & -2 & -3 \\ -2 & 2 & 0 \\ -3 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} 0 & 2 & 3 \\ 2 & 0 & 0 \\ 3 & 0 & 0 \end{bmatrix} = \mathbf{D} - \mathbf{A} = \mathbf{L}.$$

O Teorema 1 permite demonstrar que a matriz Laplaciana de um grafo não-direcionado é positiva semi-definida ($\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0, \forall \mathbf{x}$), pois para qualquer sinal real $\mathbf{x} = (x_0\ x_1\ \ldots\ x_{N-1})^T$,

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \mathbf{x}^T\mathbf{B}\mathbf{B}^T\mathbf{x} = (\mathbf{B}^T\mathbf{x})^T(\mathbf{B}^T\mathbf{x}) = \|\mathbf{B}^T\mathbf{x}\|^2 \geq 0. \tag{2.23}$$

Assim, todos os autovalores de $\mathbf{L}$ são reais não-negativos. Além disso, pode-se mostrar que zero é um autovalor com multiplicidade igual ao número de componentes conexas do grafo não-direcionado [11] e, portanto, grafos conexos têm apenas um autovalor de $\mathbf{L}$ nulo. Como conclusão, os autovalores de $\mathbf{L}$, denotados por γ, podem ser ordenados como $\gamma_0 = 0 < \gamma_1 \leq \cdots \leq \gamma_{N-1}$.

O operador diferença e a GFT

De (2.18), percebe-se que a matriz Laplaciana age como um operador diferença sobre um sinal $\mathbf{x}$ definido em um grafo $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$, atualizando cada amostra pela diferença entre o valor em certo vértice e a soma dos valores em vértices adjacentes. Ou seja,

$$\mathbf{L}\mathbf{x} = \mathbf{D}\mathbf{x} - \mathbf{A}\mathbf{x}, \tag{2.24}$$

$$\Rightarrow (\mathbf{L}\mathbf{x})_i = d_i x_i - \sum_{k \mid v_k \in \mathcal{N}_i} A_{ik}x_k = \sum_k A_{ik}x_i - \sum_{k \mid v_k \in \mathcal{N}_i} A_{ik}x_k = \sum_{k \mid v_k \in \mathcal{N}_i} A_{ik}[x_i - x_k]. \tag{2.25}$$

Na concepção de sua teoria e formulação de sua transformada de Fourier sobre grafos (GFT), Shuman *et al.* basearam-se no fato de que a transformada de Fourier de tempo contínuo é uma expansão de um sinal numa base de exponenciais complexas, que são autofunções do operador Laplaciano unidimensional (segunda derivada) [27, 11]:

$$\Delta e^{j\Omega t} = \frac{\partial^2}{\partial t^2} e^{j\Omega t} = -\Omega^2 e^{j\Omega t}. \tag{2.26}$$

Com isto, a GFT de um sinal $\mathbf{x}$, sobre o grafo não-direcionado $\mathcal{G} = \{\mathcal{V}, \mathbf{A}\}$ com matriz Laplaciana $\mathbf{L} = \mathbf{U}\grave{}\mathbf{U}^{-1}$, foi definida como a sua expansão sobre a base de autovetores de $\mathbf{L}$, ou seja,

$$\widetilde{\mathbf{x}} \stackrel{\Delta}{=} \mathbf{U}^{-1}\mathbf{x} \qquad \text{(análise)} \ ; \qquad \mathbf{x} = \mathbf{U}\widetilde{\mathbf{x}}, \qquad \text{(síntese)} \tag{2.27}$$

e como a matriz $\mathbf{L}$ é real e simétrica, é sempre diagonalizável. Mais do que isso, existe uma matriz de autovetores *real* e *ortonormal* $\mathbf{U}$, i. e. $\mathbf{U}^{-1} = \mathbf{U}^T$ [28, Teorema 4.4].

Tal definição traz uma ideia de frequência para sinais sobre grafos coerente com a interpretação clássica. Em (2.26), a informação de frequência está contida nos autovalores $-\Omega^2$ associados a cada autofunção $e^{j\Omega t}$, e autovalores próximos de zero estão associados a harmônicos mais suaves. O mesmo observa-se nos autovalores e autovetores da Laplaciana de um grafo: autovetores associados a menores autovalores são mais suaves que aqueles

associados a maiores autovalores. Como a Laplaciana age como um operador diferença, a suavidade de $\mathbf{x}$ pode ser medida pela norma de $\mathbf{Lx}$, uma métrica similar à variação total em GSP_A (cf. (2.12)), e, se $\mathbf{u}_i$ é o autovetor de $\mathbf{L}$ associado a γ_i,

$$\|\mathbf{L}\mathbf{u}_i\| = \|\gamma_i \mathbf{u}_i\| = \gamma_i \|\mathbf{u}_i\| \quad (\text{pois } \gamma_i \geq 0), \tag{2.28}$$

de forma que, normalizando os autovetores de $\mathbf{L}$, percebe-se que se $\gamma_i \leq \gamma_j$ então $\|\mathbf{L}\mathbf{u}_i\| \leq \|\mathbf{L}\mathbf{u}_j\|$ e $\mathbf{u}_i$ é mais suave que $\mathbf{u}_j$.

Além disso, observando o autovetor associado ao único autovalor nulo do Laplaciano para grafos não-direcionados conexos, tem-se que

$$\mathbf{L}\mathbf{u}_0 = \gamma_0 \mathbf{u}_0 = \mathbf{0}, \tag{2.29}$$

que se trata de um sistema de equações cujas soluções formam o espaço nulo de $\mathbf{L}$. Como a dimensão deste espaço é igual à multiplicidade geométrica de $\gamma_0 = 0$, que é igual a 1, basta encontrar uma solução não-trivial de (2.29) para obter uma base dos possíveis autovetores $\mathbf{u}_0$. De (2.25), vê-se que

$$(\mathbf{L}\mathbf{u}_0)_i = 0 \Rightarrow \sum_{k \mid v_k \in \mathcal{N}_i} A_{ik}[u_{0i} - u_{0k}] = 0, \tag{2.30}$$

o que mostra que qualquer vetor *constante* é solução para (2.29). Isto equivale a dizer que o autovetor associado ao autovalor (frequência) nulo é constante, como ocorre na teoria clássica de processamento de sinais. Se os autovetores forem normalizados, então cada entrada de $\mathbf{u}_0$ vale $1/\sqrt{N}$.

A Fig. 2.11 ilustra o conceito de frequência em GSP_L. Foi utilizado um grafo de sensores com 1000 vértices, com arestas ponderadas segundo (2.2). A componente de frequência nula é o autovetor $\mathbf{u}_0$, e a suavidade das componentes diminui à medida que se aumenta a frequência (autovalor). A Fig. 2.11f mostra o número de mudanças de sinal para cada autovetor em função do respectivo autovalor, calculado como o número de arestas ligando vértices cujo produto das amostras é negativo.

Filtragem

O conceito de filtragem de sinais sobre grafos, no presente ferramental teórico, é definido primeiro no domínio da frequência: a resposta em frequência de um filtro $\mathbf{H}$ é o vetor $\widetilde{\mathbf{h}} \in \mathbb{R}^N$ tal que, para certo sinal $\mathbf{x}$ com GFT $\widetilde{\mathbf{x}} = \mathbf{U}^{-1}\mathbf{x}$ sobre um certo grafo, sua saída nesse filtro é

$$\widetilde{\mathbf{y}} \triangleq \widetilde{\mathbf{h}} \odot \widetilde{\mathbf{x}} = \text{diag}\,(\widetilde{\mathbf{h}})\widetilde{\mathbf{x}}, \tag{2.31}$$

em que $\odot$ representa o produto de Hadamard e $\text{diag}\,(\cdot)$ é uma matriz diagonal com as entradas da diagonal dadas pelo argumento. Tomando a GFT inversa em ambos os membros

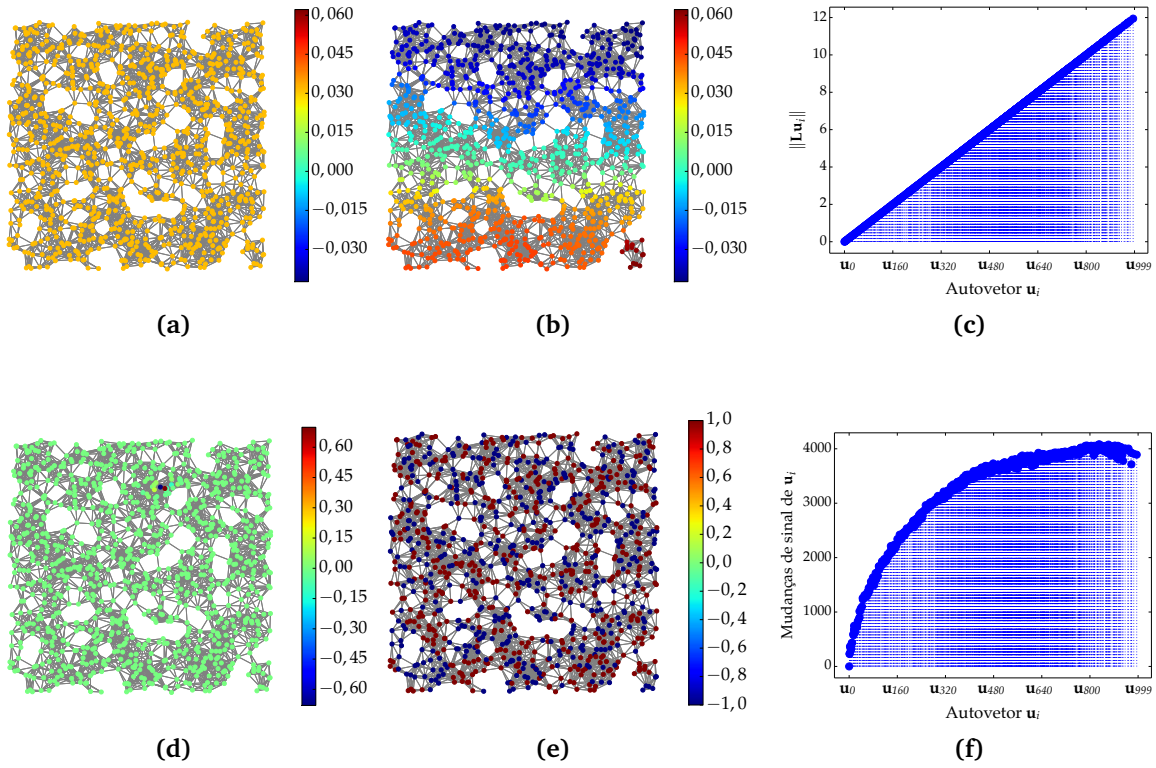


Figura 2.11 – Alguns autovetores da matriz Laplaciana de um grafo de sensores com 1000 vértices: (a) $\mathbf{u}_0$, (b) $\mathbf{u}_1$ e (d) $\mathbf{u}_{999}$. Como $\mathbf{u}_{999}$ tem um pico difícil de discernir das demais amostras do sinal, em (e) é mostrada uma versão binarizada $\bar{\mathbf{u}}_{999}$, valendo 1 em amostras positivas de $\mathbf{u}_{999}$ e -1 nas negativas. Em (c) e (f) são mostradas medidas de suavidade de um sinal: a norma da diferença $\|\mathbf{L}\mathbf{u}_i\|$ e o número de mudanças de sinal dos autovetores $\mathbf{u}_i$.

da equação,

$$\mathbf{U}\widetilde{\mathbf{y}} = \mathbf{y} = \mathbf{U}\Big[\operatorname{diag}(\widetilde{\mathbf{h}})\widetilde{\mathbf{x}}\Big] = \mathbf{U}\operatorname{diag}(\widetilde{\mathbf{h}})\mathbf{U}^{-1}\mathbf{x}, \tag{2.32}$$

ou seja, a filtragem no domínio dos vértices do grafo é dada por

$$\mathbf{y} = \mathbf{H}\mathbf{x}, \tag{2.33}$$

com

$$\mathbf{H} \triangleq \mathbf{U}\operatorname{diag}(\widetilde{\mathbf{h}})\mathbf{U}^{-1}. \tag{2.34}$$

Pode-se fazer as componentes de $\widetilde{\mathbf{h}}$ serem polinômios de grau menor ou igual a K nos autovalores de $\mathbf{L}$ [11], de modo que

$$\widetilde{h}_\ell = \sum_{k=0}^{K} a_k \gamma_\ell^k, \tag{2.35}$$

e que a resposta do filtro à entrada $\mathbf{x}$, obtida pela aplicação em (2.31) da GFT inversa, tenha entradas

$$y_i = \sum_{\ell=0}^{N-1} u_{\ell,i}\widetilde{h}_\ell \widetilde{x}_\ell, \tag{2.36}$$

em que $u_{\ell,i}$ é a i-ésima componente do autovetor $\mathbf{u}_\ell$. Substituindo (2.35) em (2.36),

$$y_i = \sum_{\ell=0}^{N-1} u_{\ell,i} \left(\sum_{k=0}^{K} a_k \gamma_\ell^k \right) \left(\sum_{j=0}^{N-1} u_{\ell,j} x_j \right) = \sum_j x_j \left[\sum_k a_k \left(\sum_\ell \gamma_\ell^k u_{\ell,i} u_{\ell,j} \right) \right].$$

Mas como $L_{i,j} = \left(\mathbf{U} \grave{} \mathbf{U}^T\right)_{i,j} = \sum_{\ell=0}^{N-1} \gamma_\ell u_{\ell,i} u_{\ell,j}$, então

$$\sum_{\ell=0}^{N-1} \gamma_\ell^k u_{\ell,i} u_{\ell,j} = \left(\mathbf{L}^k\right)_{i,j} \tag{2.37}$$

e, portanto,

$$y_i = \sum_{j=0}^{N-1} x_j \sum_{k=0}^{K} a_k \left(\mathbf{L}^k\right)_{i,j}. \tag{2.38}$$

Por fim, usa-se o fato de que, se o menor número de arestas ligando os vértices v_i a v_j é maior do que k, então $\left(\mathbf{L}^k\right)_{i,j} = 0$ ([29, Lema 5.2], como citado em [11]). Com isso,

$$y_i = \sum_{j | v_j \in \mathcal{N}(i,K)} H_{i,j} x_j, \tag{2.39}$$

em que $H_{i,j} = \sum_{k=0}^{K} a_k \left(\mathbf{L}^k\right)_{i,j}$. É indicado por (2.39) que a filtragem no domínio dos vértices é uma operação linear localizada, combinando linearmente amostras na vizinhança de um vértice dentro de um raio de até K arestas.

Amostragem e Reconstrução de Sinais sobre Grafos

Analogamente ao que ocorre em DSP, a amostragem[6] e a reconstrução de sinais definidos sobre grafos desempenham papéis importantes em GSP. Nesse contexto, o conceito de limitação de faixa precisa ser estendido convenientemente para o caso de sinais em grafos de forma a viabilizar suas reconstruções perfeitas, as quais se dão geralmente via um simples procedimento de interpolação por mínimos quadrados [30], desde que alguns requisitos técnicos sejam atendidos [31]. Quando os sinais amostrados estão sob influência de ruídos, abordagens adaptativas [32, 33, 34] são alternativas promissoras para estimar sinais sobre grafos. Esta estimação adaptativa baseia-se nos conhecidos algoritmos *least-mean-squares* (LMS) e *recursive least-squares* (RLS) [35] e viabiliza a reconstrução e o rastreamento em tempo real de sinais cujos modelos probabilísticos são não-estacionários. As principais idéias relacionadas a esses conceitos são descritas a seguir.

[6] Talvez uma definição mais precisa seja de *subamostragem*, em vez de *amostragem*, tendo em vista que o sinal em grafo é originalmente definido no domínio discreto dos vértices. Porém será mantido o uso mais comum do termo *amostragem*.

Limitação de faixa e amostragem

No contexto de GSP_L, a extensão do conceito de limitação de faixa dá-se assim:

Definição 3 (Limitação de faixa)**.** *Um sinal $\vec{x}$ é limitado em faixa, ou espectralmente esparso (eesparso), quando sua representação no domínio da frequência $\widetilde{\vec{x}}$, dada pela GFT em* (2.27)*, é esparsa. Mais especificamente, tomando $\mathscr{F}$ como um subconjunto de índices de $\{0,1,\ldots,N-1\}$, o sinal $\vec{x}$ é dito $\mathscr{F}$-eesparso se os elementos de $\widetilde{\vec{x}}$ com índices em $\overline{\mathscr{F}} \triangleq \{0,1,\ldots,N-1\} \setminus \mathscr{F}$ são nulos, isto é, $\widetilde{\vec{x}}_{\overline{\mathscr{F}}}$ é um vetor nulo de dimensão $N - |\mathscr{F}|$.*

Cabe aqui ressaltar que a definição acima também se aplica ao contexto de GSP_A, apenas mudando $\widetilde{\vec{x}}$ por $\widehat{\vec{x}}$ definido em (2.10). Optou-se por trabalhar com GSP_L apenas para fixar a notação.

Note que, se $\vec{x}$ é $\mathscr{F}$-eesparso, então a operação de síntese em (2.27) pode ser realizada por

$$\vec{x} = \vec{U}_{\mathscr{F}} \widetilde{\vec{x}}_{\mathscr{F}}\,, \tag{2.40}$$

onde $\vec{U}_{\mathscr{F}}$ é a matriz $N \times |\mathscr{F}|$ formada por autovetores da Laplaciana do grafo indexados por $\mathscr{F}$.

Em DSP, sinais temporais com faixa limitada podem ser amostrados e reconstruídos sem perda de informação, desde que o critério de Nyquist seja satisfeito; descreve-se agora um resultado semelhante para sinais sobre grafos. Inicialmente, considere que as operações de amostragem e reconstrução de um sinal $\vec{x}$ podem ser vistas como resultado da pré-multiplicação do sinal original por uma matriz de amostragem $\vec{D}$ e por uma matriz de interpolação $\vec{\Phi}$. Para a reconstrução perfeita do sinal original, deseja-se encontrar matrizes $\vec{D}$ e $\vec{\Phi}$ tais que $\vec{x} = \vec{\Phi}\vec{D}\vec{x}$ para qualquer $\vec{x}$ descrito por (2.40).

A etapa de amostragem consiste na observação dos valores de um sinal sobre grafo num *conjunto de amostragem* $\mathscr{D} \subseteq \mathscr{V}$. Nesse contexto, pode-se especificar a matriz de amostragem como $\vec{D}_{\mathscr{D}}$, uma matriz diagonal com entradas $d_i = 1$, se $v_i \in \mathscr{D}$, e $d_i = 0$, caso contrário. Assim, obtém-se o vetor

$$\vec{x}_{\mathscr{D}} \triangleq \vec{D}_{\mathscr{D}}\vec{x}\,. \tag{2.41}$$

Para reconstruir um sinal $\mathscr{F}$-eesparso $\vec{x}$ a partir de uma versão amostrada $\vec{x}_{\mathscr{D}}$, utiliza-se a expressão (2.40) e, quando a matriz $\vec{U}_{\mathscr{F}}^T \vec{D}_{\mathscr{D}} \vec{U}_{\mathscr{F}}$ tem posto completo, $\vec{\Phi}$ é escolhida como

$$\vec{\Phi} = \vec{U}_{\mathscr{F}} \big(\vec{U}_{\mathscr{F}}^T \vec{D}_{\mathscr{D}} \vec{U}_{\mathscr{F}}\big)^{-1} \vec{U}_{\mathscr{F}}^T\,, \tag{2.42}$$

de tal forma que o procedimento de amostragem e interpolação descrito por $\vec{\Phi}\vec{D}_{\mathscr{D}}\vec{x}$ resulta no sinal original $\vec{x} = \vec{U}_{\mathscr{F}} \widetilde{\vec{x}}_{\mathscr{F}}$. A condição de posto completo é satisfeita desde que o conjunto de amostragem escolhido $\mathscr{D}$ garanta a igualdade [31, 36]

$$\operatorname{rank}(\vec{D}_{\mathscr{D}} \vec{U}_{\mathscr{F}}) = |\mathscr{F}|\,. \tag{2.43}$$

Como $\text{rank}(\vec{D}_{\mathscr{D}}) = |\mathscr{D}|$, a partir de (2.43) conclui-se que uma condição necessária para a recuperação perfeita de um sinal sobre grafo amostrado é $|\mathscr{D}| \geq |\mathscr{F}|$, isto é, o número de amostras retidas deve ser, pelo menos, a quantidade de componentes não-nulas de $\widetilde{\vec{x}}$.

O Algoritmo 1 mostra como empregar a estratégia de interpolação para lidar com sinais sobre grafos com faixa limitada e variantes no tempo. Este algoritmo basicamente calcula uma estimativa instantânea de mínimos quadrados $\vec{x}_{\text{e}}[k]$ para o sinal original de faixa limitada $\vec{x}[k]$ baseado no sinal amostrado atual $\vec{D}_{\mathscr{D}}\vec{x}_w[k]$ no instante do tempo $k \in \mathbb{Z}$, onde $\vec{x}_w[k]$ é o vetor de sinal medido, possivelmente corrompido por ruído.

Algorithm 1 Interpolação linear instantânea

1: $k \leftarrow 0$
2: $\vec{\Phi} = \vec{U}_{\mathscr{F}}\left(\vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{U}_{\mathscr{F}}\right)^{-1}\vec{U}_{\mathscr{F}}^T$ **while** *(true)* **do**
3:
end
$\vec{x}_{\text{e}}[k] = \vec{\Phi}\vec{D}_{\mathscr{D}}\vec{x}_w[k]$
4: $k \leftarrow k+1$
5: **end**

Seleção do conjunto de amostragem

Até este ponto qualquer escolha de *conjunto de amostragem* $\mathscr{D}$ respeitando a condição (2.43) resulta numa recuperação perfeita do sinal original ao aplicar o Algoritmo 1. No entanto, em uma situação prática é de se esperar que as medições obtidas sejam corrompidas por ruído.

Com base nesta suposição, uma modelagem mais adequada para um sinal sobre um grafo seria

$$\boldsymbol{x}_w[k] = \vec{x}[k] + \boldsymbol{w}[k], \tag{2.44}$$

onde $\boldsymbol{x}_w[k]$ é o sinal aleatório ruidoso[7] disponível nos vértices do grafo, $\vec{x}[k]$ é o sinal $\mathscr{F}$-eesparso original, e $\boldsymbol{w}[k]$ é um sinal aleatório de ruído com média zero e matriz de covariância $\vec{C}_w[k]$.

Considerando o modelo em (2.44), pode-se calcular algumas figuras de mérito para um sinal estimado $\vec{x}_{\text{e}}[k]$, como o *mean-square deviation* (MSD) dado por

$$\begin{aligned}\text{MSD} &= \mathbb{E}\left\{\|\boldsymbol{x}_{\text{e}}[k] - \vec{x}[k]\|_2^2\right\} \\ &= \mathbb{E}\left\{\|\vec{U}_{\mathscr{F}}\left(\vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{U}_{\mathscr{F}}\right)^{-1}\vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\boldsymbol{w}[k]\|_2^2\right\},\end{aligned} \tag{2.45}$$

ou o *square deviation* (SD) dado por

$$\begin{aligned}\text{SD} &= \|\vec{x}_{\text{e}}[k] - \vec{x}[k]\|_2^2 \\ &= \|(\vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{U}_{\mathscr{F}})^{-1}\vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{w}[k]\|_2^2.\end{aligned} \tag{2.46}$$

[7] Aqui, $\boldsymbol{x}$ denota um vetor aleatório com realizações denotadas como $\mathbf{x}$.

As expressões em (2.45) e (2.46) apresentam uma dependência explícita de $\vec{D}_{\mathscr{D}}$, indicando que o conjunto de amostragem $\mathscr{D}$ pode ser apropriadamente escolhido para otimizar a figura de mérito desejada. No entanto, o problema de otimização resultante é de natureza combinatória.

A fim de obter um bom compromisso entre o erro de reconstrução obtido e o tempo necessário para o cálculo do conjunto de amostragem $\mathscr{D}$, uma abordagem comum é empregar um algoritmo guloso para a otimização. Um algoritmo guloso basicamente reduz a complexidade computacional geral, procurando por uma seleção ótima em cada estágio e esperando encontrar um valor final próximo do ótimo global. Informações detalhadas a respeito da capacidade de reconstrução de estratégias gulosas para GSP são apresentadas em [37].

Um exemplo de seleção do conjunto de amostragem é apresentado no Algoritmo 2, que emprega em cada iteração uma busca gulosa pelo índice $n \in \{0, 1, \ldots, N-1\}$ a ser adicionado ao conjunto atual $\mathscr{D}$ para maximizar o autovalor mínimo não negativo de $\vec{U}_{\mathscr{F}}^{T}\vec{D}_{\mathscr{D}\cup\{n\}}\vec{U}_{\mathscr{F}}$. O número de índices no conjunto final $\mathscr{D}$ é denotado como D. Este método foi inicialmente sugerido em [31] e tem a mesma forma que uma das estratégias de amostragem descritas em [32]. De fato, seguindo uma ideia similar a [31], pode ser mostrado que o Algoritmo 2 usa um esquema guloso para minimizar o SD em (2.46).

Algorithm 2 Seleção gulosa de $\mathscr{D}$.

1: $\mathscr{D} \leftarrow \emptyset$ **while** $|\mathscr{D}| < D$ **do**
2:
end
$d = \underset{n\in\{0,1,\ldots,N-1\}}{argmax} \ \lambda_{min}^{+}(\vec{U}_{\mathscr{F}}^{T}\vec{D}_{\mathscr{D}\cup\{n\}}\vec{U}_{\mathscr{F}})$
3: $\mathscr{D} \leftarrow \mathscr{D} + \{d\}$
4: **end**
5: **return** $\mathscr{D}$

Estimação adaptativa de sinais sobre grafos

Uma tentativa inicial de mesclar a tradicional área de filtragem adaptativa [35] com o novo campo de GSP é feita em [32], onde os autores sugerem estratégias baseadas em LMS para lidar com o problema de reconstrução de um sinal definido sobre grafo. Além disso, um algoritmo baseado em RLS é proposto em [33] para uma tarefa de estimação idêntica.

A razão para o emprego de uma estratégia adaptativa num contexto de estimação de sinais sobre grafos vem da possibilidade de uma reconstrução robusta e do rastreamento em tempo real de sinais aleatórios com estatísticas variantes no tempo. A robustez da abordagem adaptativa é útil para trabalhar em cenários ruidosos, como (2.44), quando espera-se que uma estratégia adaptativa forneça um MSD menor em comparação a uma interpolação instantânea do sinal, como no Algoritmo 1.

Estimação LMS

O algoritmo de filtragem adaptativa mais utilizado é o LMS [38, 39]. Entre muitas razões, como a convergência não polarizada para a solução de Wiener e o comportamento estável quando implementado com precisão finita, a popularidade do algoritmo LMS se deve principalmente à sua simplicidade e baixa complexidade computacional [35].

Num contexto de reconstrução de sinais sobre grafos [32, 40], o algoritmo LMS é projetado de forma a minimizar a média do erro quadrático sobre as amostras dos vértices em $\mathscr{D}$,

$$\min_{\widetilde{\vec{x}}_{\mathscr{F}}[k]} \mathbb{E}\left\{ ||\vec{D}_{\mathscr{D}}(\boldsymbol{x}_w[k] - \vec{U}_{\mathscr{F}}\widetilde{\vec{x}}_{\mathscr{F}}[k])||_2^2 \right\} . \tag{2.47}$$

Usando a abordagem de minimização baseada em gradiente estocástico para o problema (2.47), encontra-se uma expressão de atualização para $\widetilde{\vec{x}}_{\mathscr{F}}[k+1]$, que por sua vez gera uma estimativa $\vec{x}_{\mathrm{e}}[k]$ para o sinal de faixa limitada $\vec{x}[k]$ em (2.40), que corresponde à equação de atualização do LMS [40]

$$\vec{x}_{\mathrm{e}}[k+1] = \vec{x}_{\mathrm{e}}[k] + \mu\vec{U}_{\mathscr{F}}\vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}(\vec{x}_w[k] - \vec{x}_{\mathrm{e}}[k]), \tag{2.48}$$

em que $\mu \in \mathbb{R}_+$ é o chamado fator de convergência [35], parâmetro que controla o comportamento do algoritmo para melhorar sua velocidade de convergência ou reduzir o erro em estado estacionário.

Dessa forma, o procedimento para encontrar uma estimativa em tempo real de $\vec{x}[k]$ com base na equação de atualização do LMS (2.48) é apresentado no Algoritmo 3. O valor inicial $\vec{x}_{\mathrm{e}}[0]$ deve ser um sinal limitado em faixa (como $\vec{x}_{\mathrm{e}}[0] = \vec{0}$) e o tamanho de passo μ deve ser suficientemente pequeno para que o algoritmo seja estável [32].

Algorithm 3 Estimação LMS de sinais sobre grafos

1: $k \leftarrow 0$ **while** *(true)* **do**
   end
2: $\vec{x}_{\mathrm{e}}[k+1] = \vec{x}_{\mathrm{e}}[k] + \mu\,\vec{U}_{\mathscr{F}}\vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}(\vec{x}_w[k] - \vec{x}_{\mathrm{e}}[k])$
3: $k \leftarrow k+1$
4: **end**

Estimação RLS

Outra técnica bem conhecida na área de filtragem adaptativa consiste num procedimento iterativo que resolve o problema de mínimos quadrados para cada sinal de entrada de forma recursiva, sendo este chamado de algoritmo RLS [35]. A estratégia LMS leva muito tempo até atingir seu estado estacionário, exigindo alternativas, como o método RLS, uma abordagem muito mais rápida. No entanto, o compromisso encontrado ao utilizar um algoritmo de menor tempo de convergência, como o RLS, é o aumento considerável na complexidade computacional, o que pode ser um fator limitante, dependendo da aplicação.

De forma similar à ideia do LMS, o algoritmo RLS para estimação de sinais sobre grafos [34] pretende minimizar uma função do erro quadrático sobre as amostras nos vértices em $\mathscr{D}$, da forma

$$\min_{\widetilde{\vec{x}}_{\mathscr{F}}} \sum_{l=1}^{k} \beta^{k-l} \|\vec{D}_{\mathscr{D}}(\vec{x}_w[l] - \vec{U}_{\mathscr{F}}\widetilde{\vec{x}}_{\mathscr{F}})\|^2_{\vec{C}_w^{-1}} + \beta^k \|\widetilde{\vec{x}}_{\mathscr{F}}\|^2_{\vec{\Pi}}, \tag{2.49}$$

em que $0 \ll \beta \leq 1$ é o fator de esquecimento, e $\vec{\Pi}$ é uma matriz de regularização, geralmente escolhida como $\vec{\Pi} = \delta\vec{I}$, para um pequeno $\delta > 0$.

Ao resolver o problema convexo em (2.49), verifica-se que a estimativa para o sinal limitado em faixa $\vec{x}[k]$ é dada por

$$\vec{x}_{\mathrm{e}}[k] = \vec{U}_{\mathscr{F}}\vec{\Psi}^{-1}[k]\boldsymbol{\psi}[k], \tag{2.50}$$

onde $\vec{\Psi}[k]$ e $\boldsymbol{\psi}[k]$ são calculados por

$$\begin{aligned} \vec{\Psi}[k] &= \beta\vec{\Psi}[k-1] + \vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{C}_w^{-1}[k]\,\vec{U}_{\mathscr{F}}\,, \\ \boldsymbol{\psi}[k] &= \beta\boldsymbol{\psi}[k-1] + \vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{C}_w^{-1}[k]\,\vec{x}_w[k]\,. \end{aligned} \tag{2.51}$$

Assim, com base nas equações de atualização (2.50) e (2.51), é apresentada a técnica RLS para estimação de sinais definidos sobre grafos [34] no Algoritmo 4. Os valores iniciais deste algoritmo podem ser tomados como $\vec{\Psi}[0] = \vec{\Pi}$ e um vetor $\boldsymbol{\psi}[0]$ aleatório, como $\boldsymbol{\psi}[0] = \vec{0}$.

Algorithm 4 Estimação RLS de sinais sobre grafos

1: $k \leftarrow 0$ **while** *(true)* **do**
   end
2: $\vec{\Psi}[k] = \beta\vec{\Psi}[k-1] + \vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{C}_w^{-1}[k]\,\vec{U}_{\mathscr{F}}$
3: $\boldsymbol{\psi}[k] = \beta\boldsymbol{\psi}[k-1] + \vec{U}_{\mathscr{F}}^T\vec{D}_{\mathscr{D}}\vec{C}_w^{-1}[k]\,\vec{x}_w[k]$
4: $\vec{x}_{\mathrm{e}}[k] = \vec{U}_{\mathscr{F}}\vec{\Psi}^{-1}[k]\boldsymbol{\psi}[k]$
5: $k \leftarrow k+1$
6: **end**

Recuperação prescindindo do suporte do sinal na frequência

Até então, foi assumido o conhecimento do suporte (faixa) do sinal no domínio da frequência para sua reconstrução/estimação. De fato, todos os algoritmos até então apresentados utilizam a matriz $\vec{U}_{\mathscr{F}}$, que corresponde a uma seleção adequada dos autovetores da matriz Laplaciana de acordo com o pré-conhecimento do suporte do sinal no domínio da frequência. Uma abordagem alternativa seria, dado o conhecimento do conjunto de amostragem $\mathscr{D}$, estimar diretamente o valor do sinal nos demais vértices do grafo, de modo que o sinal estimado seja suave. No contexto de $\mathrm{GSP}_{\mathrm{A}}$, Sandryhaila e Moura em [25] quiseram estimar um sinal de valores discretos, de modo a aplicar o método a problemas de classificação, e para tanto aplicaram um algoritmo de otimização para obter

o sinal que minimizava a norma $\|\mathbf{s} - \mathbf{A}^{norm}\mathbf{s}\|_2$, sujeito à condição de que o sinal estimado divergisse pouco do sinal original nos vértices com valores conhecidos *a priori*.

Uma abordagem semelhante para GSP_L seria utilizar a matriz Laplaciana aplicada ao sinal para estimar sua suavidade. Isto se traduz no problema de otimização

$$\mathbf{x}_\text{e} = \underset{\mathbf{s}\in\mathbb{R}^N}{\arg\min} \|\mathbf{L}\mathbf{s}\|_2, \quad \text{sujeito a} \quad \mathbf{D}_{\mathscr{D}}\mathbf{s} = \mathbf{D}_{\mathscr{D}}\mathbf{x}, \tag{2.52}$$

em que $\mathbf{x}$ é o sinal original sobre o grafo, não necessariamente limitado em faixa. Algoritmos adaptativos poderiam ser deduzidos a partir dessa abordagem também.

Exemplos e Aplicações

Nesta seção, pretende-se ilustrar o uso dos conceitos de GSP a cenários com dados estruturados em rede, a fim de ressaltar como a consideração acerca do domínio subjacente ao sinal provê informação crucial para o melhor tratamento dos dados. Os exemplos envolverão a remoção de ruído em uma rede de sensores, o problema de estimação de dados climáticos sobre um grafo de cidades brasileiras (uma boa oportunidade para comentar as técnicas de reconstrução de sinal), e uma aplicação de GSP a compressão de imagens de *light-field*. Concluindo a seção, serão mencionados diversos outros contextos de aplicação encontrados na literatura.

Remoção de ruído em rede de sensores

Consideremos o sinal suave definido sobre o grafo de Minnesota, como na Fig. 2.12a, com componentes dadas por

$$s_i = \cos\big(\text{coord}_x(v_i)\big) + \text{sen}\big(\text{coord}_y(v_i)\big), \tag{2.53}$$

em que coord_x e coord_y indicam as coordenadas geográficas dos vértices sobre o estado de Minnesota.

Ao sinal $\mathbf{s}$ foi adicionado uma realização $\mathbf{w}$ de um ruído gaussiano de média nula e desvio padrão igual a 20% da amplitude de $\mathbf{s}$, resultando no sinal mostrado na Fig. 2.12b. A análise espectral do sinal original e do ruído gaussiano mostrou que, embora o ruído seja aproximadamente branco, o sinal — por ser suave — concentra sua energia em componentes de baixa frequência, em autovalores de módulo aproximadamente $0 \leq \gamma_i \leq 0{,}5$ (vide Fig. 2.12c). Por isso, criou-se um filtro passa-baixa ideal com espectro $\widetilde{\mathbf{h}}_{LPF}$, de ganho unitário e frequência de corte $\gamma_\text{corte} = 0{,}5$, a fim de reduzir a influência do ruído sobre o sinal.

O sinal obtido após a filtragem passa-baixa ideal é mostrado na Fig. 2.12d. A raiz quadrada do erro médio quadrático (RMSE, *root-mean-square error*) em relação ao sinal original caiu de

$$\|\mathbf{s} - (\mathbf{s} + \mathbf{w})\|_2 = \|\mathbf{w}\|_2 \approx 44{,}6\% \tag{2.54}$$

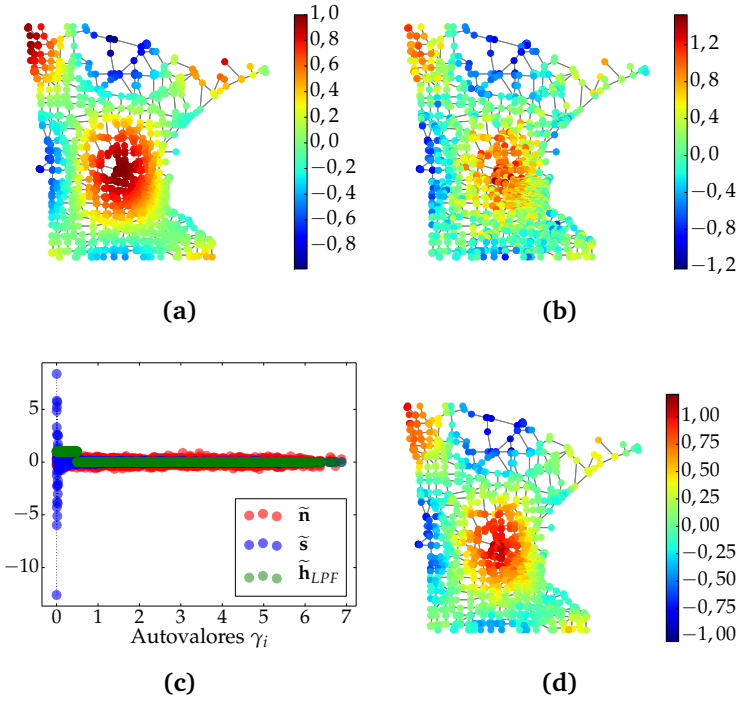


Figura 2.12 – (a) Sinal suave sobre o grafo de Minnesota. (b) Resultado da adição de um ruído gaussiano com desvio padrão de 20% da amplitude do sinal original. (c) Espectro do sinal original, do ruído gaussiano e do filtro passa-baixa ideal. (d) Sinal resultante da filtragem passa-baixa.

para

$$\|\mathbf{s} - \mathbf{U}\big(\widetilde{\mathbf{h}}_{LPF} \odot (\widetilde{\mathbf{s}} + \widetilde{\mathbf{w}})\big)\|_2 \approx 18{,}9\%, \tag{2.55}$$

uma redução pela metade. É importante observar que, por mais simples que seja o exemplo, ele ressalta a importância de se considerar a estrutura subjacente a sinais definidos em domínios irregulares: a alta compactação de energia do sinal $\mathbf{s}$, por exemplo, só foi possível graças à decomposição na base de autovetores do Laplaciano do grafo que lhe serve de domínio. Com estas considerações, técnicas simples podem obter resultados úteis e relevantes em contextos como este.

Estimação de sinais climáticos

Este exemplo se concentra na aplicação dos conceitos de amostragem e recuperação de sinais em grafos para a estimação de sinais climáticos. Para tanto, serão utilizados sinais de temperatura e índices pluviométricos de estações meteorológicas brasileiras. Inspirado pela aplicação de compressão de sinais definidos sobre grafos em [21], verifica-se que o sinal extraído de uma distribuição espacial de estações meteorológicas medindo sinais climáticos locais é geralmente suave, o que permite sua aproximação por um sinal sobre grafo de faixa limitada. A vantagem de representar um sinal sobre grafo de sinais climáticos por um número reduzido de suas amostras é notória, pois possibilita estimar índices climáticos em certas regiões sem ter um elemento sensor naquele local, reduzindo a quantidade de dispositivos sensores e de dados coletados/armazenados/transmitidos.

Buscando comparar o desempenho das estratégias adaptativas apresentadas na seção

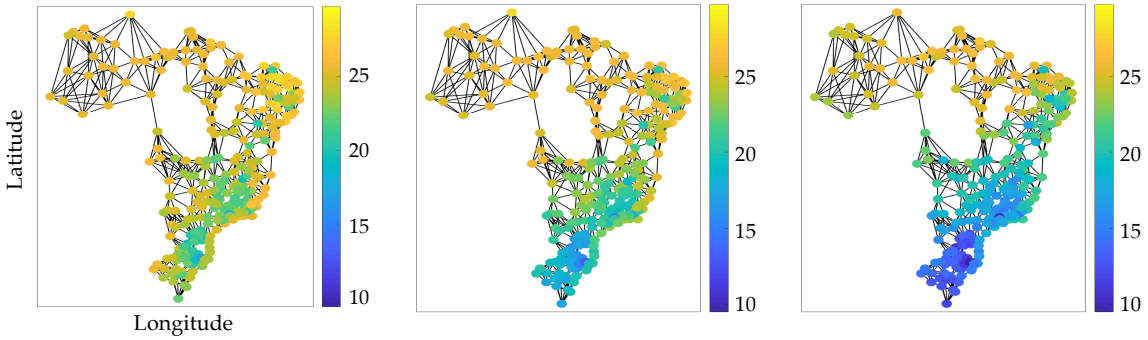


Figura 2.13 – Sinais em grafos ilustrando as temperaturas médias mensais das estações meteorológicas brasileiras durante o período de 1961-1990. Meses: (a) Janeiro, (b) Abril e (c) Julho.

2 com o método tradicional de reconstrução de sinais sobre grafos segundo o Algoritmo 1 daquela seção, foram organizadas duas bases de dados, na forma de tabelas, do site do Instituto Nacional de Meteorologia (INMET):[8] a primeira tabela contém as coordenadas de latitude e longitude de estações meteorológicas ativas, enquanto a segunda apresenta uma temperatura média mensal registrada em algumas dessas estações, durante o período de 1961-1990. Destes dados obtém-se um total de 299 nós para o grafo $\mathcal{G} = (\mathcal{V}, \mathbf{A})$ a ser construído, em que cada um desses vértices representa uma estação meteorológica. Assim, um nó v_n do grafo $\mathcal{G}$ e seu valor de sinal $x_n[k]$ correspondem, respectivamente, às coordenadas geográficas e à temperatura média da estação meteorológica associada a determinado mês.

Como não há conexões explícitas entre estações meteorológicas, há alguma liberdade no projeto do conjunto de arestas $\mathcal{E}$ e escolha dos pesos $A_{i,j}$. Para simplificar, as arestas do grafo foram construídas conectando um vértice v_n a seus $K = 8$ nós vizinhos mais próximos [11], considerando que a distância entre dois vértices do grafo é dada pela fórmula de Haversine, que avalia a distância entre pontos usando suas coordenadas de latitude e longitude, de forma que

$$A_{i,j} = \begin{cases} \exp\left(-\frac{d_{\mathrm{H}}(i,j)^2}{2\theta^2}\right), & \text{se } (v_i, v_j) \in \mathcal{E} \\ 0, & \text{caso contrário} \end{cases} \tag{2.56}$$

em que $d_{\mathrm{H}}(i,j)$ é a distância de Haversine entre os vértices v_i e v_j, θ é escolhido como $2 \cdot 10^3$ e a condição $(v_i, v_j) \in \mathcal{E}$ verifica se a aresta ligando os nós v_i e v_j faz parte do conjunto de arestas $\mathcal{E}$ do grafo. Assim, com base neste procedimento, foram obtidos os grafos ilustrados na Fig. 2.13.

Para decidir quantos componentes de frequência são necessários para representar o sinal com um erro de reconstrução razoável, avalia-se o *average reconstruction error* (ARE) $\|\vec{x} - \vec{x}^{(P)}\|_2 / \|\vec{x}\|_2$ para diferentes estimativas $\vec{x}^{(P)}$ usando apenas P componentes frequenciais do sinal original $\vec{x}$, correspondente aos dados do mês julho representados na Fig. 2.13(c).

[8] Fonte: Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP) do INMET. Acesso gratuito, disponível em: `http://www.inmet.gov.br/portal/index.php?r=bdmep/bdmep`

O sinal reconstruído $\vec{x}^{(P)}$ é obtido ordenando os valores absolutos de $\widetilde{\vec{x}}$ obtidos de (2.27) e selecionando os índices p dos P maiores componentes para formar o conjunto auxiliar $\mathscr{F}_P$ que definirá o suporte de tamanho P. Com base nestes índices $p \in \mathscr{F}_P$, escolhe-se o p-ésimo autovetor de $\vec{U}$ e a p-ésima componente frequencial de $\widetilde{\vec{x}}$ para definir $\vec{U}_{\mathscr{F}_P}$ e $\widetilde{\vec{x}}_{\mathscr{F}_P}$. Então, a estimativa $\vec{x}^{(P)}$ usando P componentes é dada por $\vec{U}_{\mathscr{F}_P}\widetilde{\vec{x}}_{\mathscr{F}_P}$. A Fig. 2.14 exibe o percentual de ARE para diferentes valores P no intervalo $[50, 250]$. Considerando que um erro de reconstrução de 2.5% é aceitável na aplicação atual, aproxima-se o sinal original pelos seus $P = 200$ maiores componentes frequenciais. A partir desta suposição, pode-se definir o conjunto de faixa limitada aproximada $\mathscr{F} = \mathscr{F}_P$, onde $|\mathscr{F}| = P = 200$.

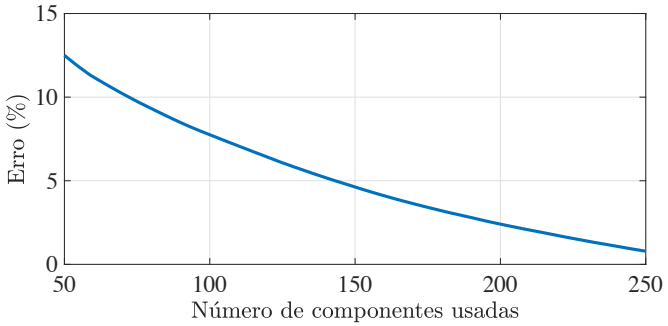


Figura 2.14 – Erro de reconstrução percentual quando o sinal original é comprimido usando P componentes frequenciais.

Com base neste sinal $\mathscr{F}$-eesparso $\vec{x}^{(P)}$, é necessário escolher o valor $|\mathscr{D}|$ bem como quais vértices $v_n \in \mathscr{V}$ do sinal sobre grafo devem ser amostrados. Como mencionado em [37], aumentar o número de amostras em $\mathscr{D}$ sempre diminui o MSD em (2.45). No entanto, como também deseja-se reduzir a quantidade de nós a serem medidos, foi tomado $|\mathscr{D}| = 210$ por ser um compromisso razoável. O conjunto de amostragem $\mathscr{D}$ foi então obtido por meio do Algoritmo 2, com $M = |\mathscr{D}| = 210$. Então, neste ponto, obtém-se o sinal sobre grafo amostrado e de faixa limitada que será usado para testar as estratégias de interpolação linear instantânea e de reconstrução adaptativa.

A Fig. 2.15 ilustra os resultados da simulação de reconstrução, em que as leituras dos nós do grafo $\vec{x}_w[k]$ são representadas pelo cenário ruidoso em (2.44), com um sinal sobre grafo constante $\vec{x}[k] = \vec{x}^{(P)}$ até o instante de tempo $k = 2500$, quando o sinal de referência é escalado por um fator de 1,1. Os parâmetros dos algoritmos foram escolhidos como: estimação inicial $\vec{x}_e[0] = \vec{0} \in \mathbb{R}^{299}$, matriz de covariância de ruído $\vec{C}_w = \sigma_w^2\vec{I}$, com $\sigma_w^2 = 0{,}01$, fator de convergência $\mu = 0{,}25$ e fator de esquecimento $\beta = 0{,}95$. Como esperado, o Algoritmo 1 (interpolação *instantânea*) ignora os valores de entrada anteriores e, após um curto período de tempo, as estratégias adaptativas obtêm uma estimativa mais precisa do sinal original $\vec{x}[k]$. Entre as técnicas adaptativas, fica claro que o algoritmo RLS apresenta uma convergência bem mais rápida que o LMS.

Para ilustrar o caso em que não se conhece o suporte $\mathscr{F}$, mas ainda se tem a expectativa de que o sinal sobre o grafo seja suave, considere o seguinte problema proposto: dados os valores de pluviometria média do mês de janeiro de 609 cidades brasileiras, retirados do

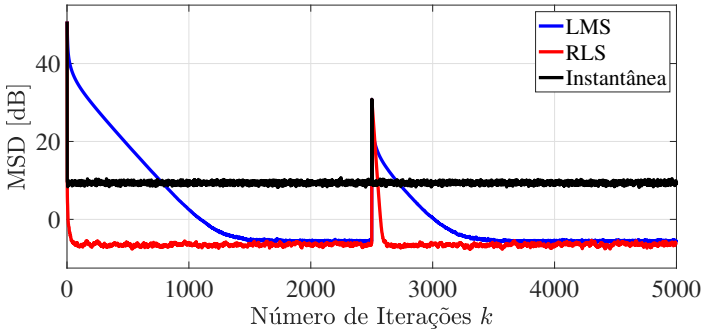


Figura 2.15 – MSD para diferentes algoritmos de reconstrução usando um sinal constante $\vec{x}[k]$ com uma transição em degrau em $k = 2500$.

banco de dados da Embrapa[9], como se poderia estimar o valor deste índice em outras cidades do Brasil? A ideia é obter um sinal $\vec{x}_e$ com os índices de precipitação média em janeiro que seja suave, ao mesmo tempo em que preserve os dados nas 609 cidades conhecidos *a priori*. Isto se traduz no problema de otimização da função objetivo descrita em (2.52), em que o conjunto de amostragem $\mathscr{D}$ indica as cidades cujos dados estão no banco de dados.

Na Fig. 2.16a, as 5570 cidades brasileiras são representadas por vértices de um grafo sem arestas[10], das quais às 609 da base de dados da Embrapa são atribuídos os valores de pluviometria média. Para executar o algoritmo de otimização com mais rapidez, optou-se por reduzir o número de vértices para $N = 1000$, obtendo o sinal mostrado na Fig. 2.16b. O grafo foi criado ligando um vértice aos 5 vizinhos mais próximos, de modo a ter um grafo conexo ao mesmo tempo em que se evita uma matriz de adjacência demasiado cheia. Ponderou-se as arestas segundo (2.56) e, uma vez que se fez o grafo conexo e não foi utilizado um limite para julgar quais arestas seriam criadas, não há parâmetro que indique qual o melhor valor da variância θ a ser usado, razão pela qual se fez $2\theta^2 = 1$.

O sinal estimado deveria, idealmente, aproximar o mapa na Fig. 2.16d, com as isoietas (curvas de mesmo índice pluviométrico) médias do mês de janeiro, relativas ao período de 1977 a 2006[11]. É importante dirigir a atenção para a escala de cores do mapa, que é inversa ao padrão adotado ao longo deste capítulo: quanto mais quente a cor, menor o valor representado.

A restrição em (2.52) foi imposta dentro da declaração da função objetivo, a fim de reduzir seu número de variáveis, de modo que havia apenas $1000 - 609 = 391$ variáveis independentes. Dessa forma, trata-se de uma otimização sem restrições (*unconstrained optimization* [41]) com 391 variáveis, e para resolvê-la foi utilizado o algoritmo BFGS e a função `optimize.minimize()` do pacote Scipy, da linguagem Python. O resultado

[9] *Banco de Dados Climáticos do Brasil*, produzido pela Embrapa e pela ESALQ/USP, disponível em: `https://www.cnpm.embrapa.br/projetos/bdclima/index.html`

[10] As informações de latitude e longitude dos 5570 municípios brasileiros, utilizada para posicionar os vértices do grafo, foram retiradas da seguinte base: `https://github.com/kelvins/Municipios-Brasileiros`.

[11] Dados do Serviço Geológico do Brasil, também conhecido como CPRM, por conta de sua razão social Companhia de Pesquisa de Recursos Minerais, empresa pública vinculada ao Ministério de Minas e Energia. Este e outros atlas pluviométricos do Brasil estão disponíveis em: `http://www.cprm.gov.br/publique/Hidrologia/Mapas-e-Publicacoes/Atlas-Pluviometrico-do-Brasil-1351.html`

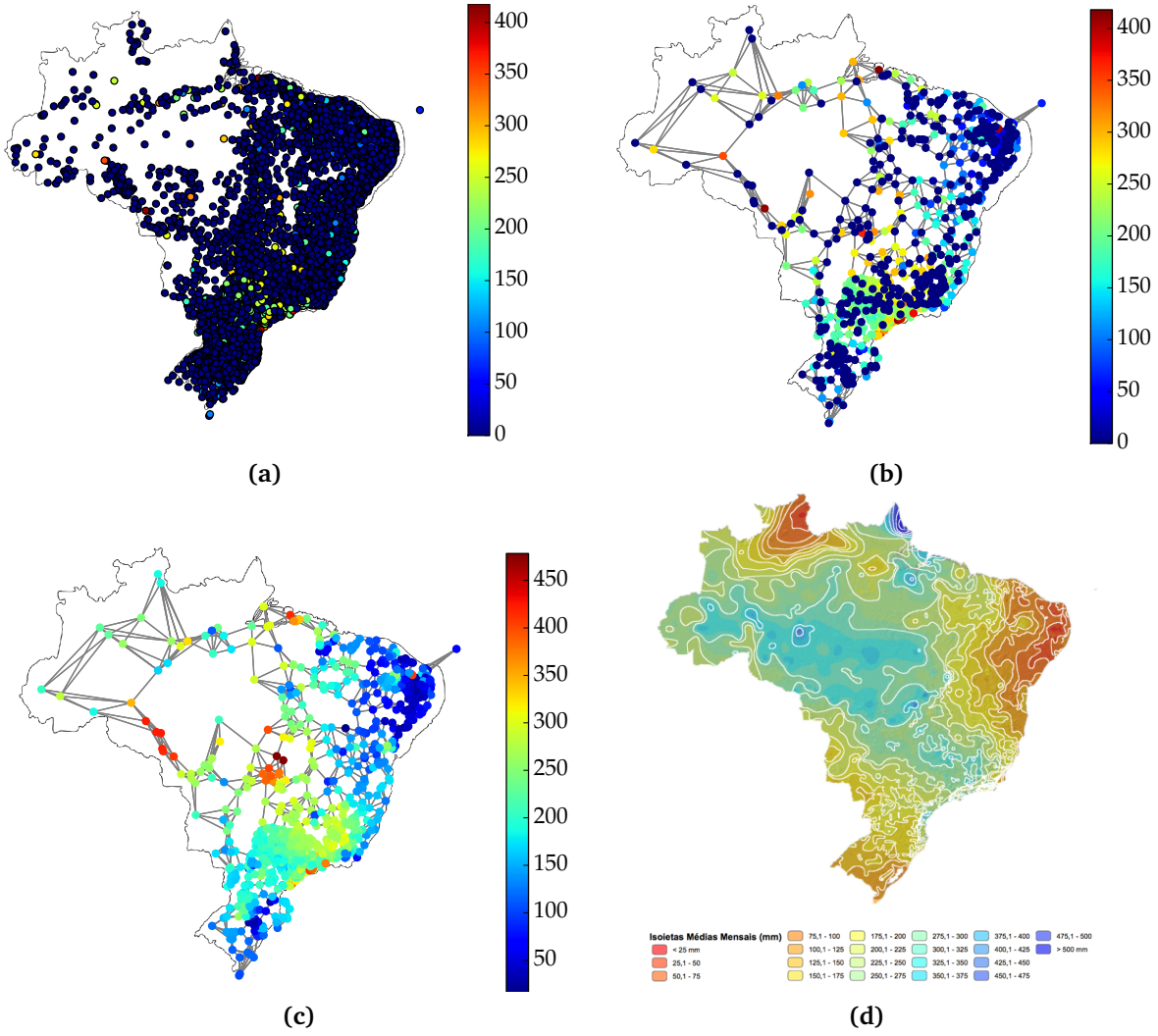


Figura 2.16 – (a) Sinal sobre o grafo (sem arestas) dos 5570 municípios brasileiros, com amostras nulas exceto nos 609 vértices correspondendo ao banco de dados da Embrapa. (b) Sinal $\mathbf{x}_{\mathscr{G}}$ com 1000 amostras, 609 das quais provêm de base de dados pluviométricos e as demais são nulas. (c) Sinal $\mathbf{x}_{\mathrm{e}}$, estimando as amostras anteriormente nulas de modo a obter um sinal suave. (d) Isoieta média do mês de janeiro, correspondente ao período de 1977 a 2006. O contorno do mapa do Brasil sobre os grafos é inexato, estando aqui apenas para guiar o leitor, sem a pretensão de alinhá-lo com as coordenadas corretas de cada cidade. Foi desenhado por Felipe Micaroni Lalli e está disponível sob licença CC By-SA no endereço `https://commons.wikimedia.org/wiki/File:Contorno_do_mapa_do_Brasil.svg`.

obtido após 2000 iterações do algoritmo de minimização é exibido na Fig. 2.16c e mostra que o algoritmo conseguiu aproximar de forma razoável o mapa de isoietas na Fig. 2.16d.

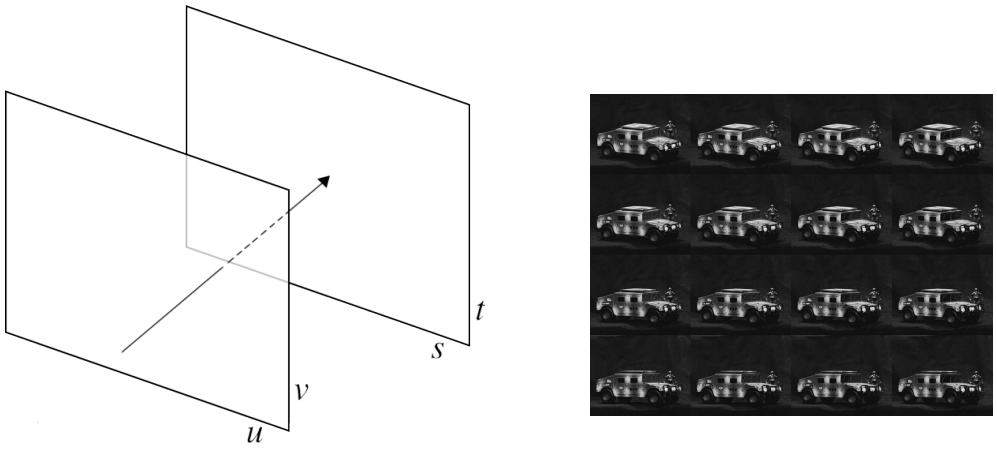
Compressão de *light fields*

Outra possível aplicação de GSP explora a propriedade de compactação de energia exercida pela GFT para a compressão de dados. Esta compactação permite maior eficiência da quantização de coeficientes de um sinal no domínio da frequência quando comparada com a quantização do sinal em seu domínio original. Neste exemplo, considera-se a compressão de imagens de *light fields* [42]. O imageamento de *light field* é uma tecnologia de captura de dados 4-D de uma cena, o que a torna uma tecnologia promissora para indústrias de entretenimento, como fotografia e cinema. Idealmente, esta tecnologia capturaria toda a informação sobre o campo de luz associado a uma cena sob a forma 5-D da função plenóptica $L(x, y, z, \theta, \phi)$, que indica a radiança (quantidade de luz) que passa por todos os pontos (x, y, z) do espaço em todas as direções (θ, ϕ). Esta informação permitiria manipulação de elementos de uma imagem após sua captura, como por exemplo ajuste do plano focal ou mudança da posição da vista da cena.

Na prática, a captura da função plenóptica não é realizável e uma parametrização é adotada para capturar informações da cena em 4-D. Esta parametrização é composta por um plano *uv* (plano da câmera) e um plano *st* (plano focal), como ilustrado na Fig. 2.17a. Implementações comuns para esta parametrização são realizadas das seguintes formas:

- Um arranjo de câmeras, localizado no plano *uv* com todas as cameras focadas na cena, localizada no plano *st*, criando uma versão discreta do plano *uv*;
- Uma única câmera se movendo por uma grade de posições no plano *uv*, capturando a cena do plano *st* a partir de cada posição na grade;
- Um arranjo de microlentes posicionado dentro de uma câmera digital convencional, de modo que cada microlente captura luz de uma direção, gerando diferentes vistas da cena.

Em todos os casos, o resultado de uma única captura de imagem em *light field* é um conjunto de vistas como ilustrado na Fig. 2.17b. Muitas vezes, a quantidade de imagens capturadas de uma única cena está na ordem de centenas ou milhares, o que gera uma enorme quantidade de dados. Algumas abordagens consideram o uso do *High-effiency video coding* (HEVC), que usa a transformada do cosseno discreto (DCT, do inglês *discrete-cosine transform*), para comprimir dados de *light field*. O HEVC processa *resíduos* de blocos de pixels que compõem uma imagem inteira. O resíduo é resultado do processo de *predição* e pode ser entendido como a diferença entre o bloco de pixels no *frame* que está sendo codificado e blocos de pixels em *frames* anteriores. Como tipicamente blocos em *frames* consecutivos são similares, os resíduos possuem menor entropia do que os blocos originais, facilitando a quantização. No caso de *light field*, diferentes vistas da cena fazem o papel



(a) **(b)**

Figura 2.17 – (a) Parametrização 4-D e (b) exemplo de resultados de captura de *light field*.

dos *frames*. O papel da transformada é mapear dados de resíduos para um domínio onde a quantização se torna mais eficiente. Como demonstrado em [43, 44], a GFT é capaz de concentrar energia em uma quantidade menor de coeficientes quando comparada com a DCT. Assim, se a mesma quantidade de coeficientes da transformada for mantida para as duas transformadas e a transformada inversa for aplicada, a GFT alcança *distorção* menor para o bloco reconstruído do que aquela alcançada pela DCT.

Como já demonstrado, a GFT está diretamente ligada à matriz de adjacências. Em alguns casos, a inferência de **A** depende dos dados e, portanto, não existe uma matriz de GFT fixa. Com isso, um sistema de compressão de dados baseado na GFT precisa conhecer **A** ou **V** (ou **U** no caso de GSP_L) no codificador e no decodificador, o que faz com que a quantidade de dados a ser transmitida cresça consideravelmente além dos elementos comprimidos. Nesta aplicação, a inferência e aplicação de **A** são exploradas de forma a minimizar seu impacto.

Considera-se GSP_A e grafos do tipo apresentado na Fig. 2.4b para representar blocos de resíduos de 32×32 pixels, seguindo o modelo *nearest-neighbor* (NN) [16]. Com esta estrutura, cada pixel está associado a um vértice e se liga a, no máximo, quatro vizinhos mais próximos (três para pixels na borda e dois para pixels no canto), oferecendo uma matriz de adjacência esparsa. Além disso, considera-se um único peso para arestas horizontais ligando todos pares de pixels dadas duas colunas do bloco. O mesmo é considerado para arestas verticais ligando pares de pixels dadas duas linhas. Por exemplo, todas as arestas ligando pixels da coluna de s_0 com a coluna de s_1 na Fig. 2.4b possuem o mesmo peso. Outro peso é dado a todas as arestas ligando as colunas de s_1 e s_2 e outro às arestas ligando as linhas dos pixels s_0 e s_N. Para um bloco de $M_1 \times M_2$ pixels, estas considerações resultam em **A** com estrutura fixa e $(M_1-1)+(M_2-1)$ coeficientes não nulos. É importante notar que, ao se transmitir a matriz **A** ao invés da matriz de transformada, adiciona-se complexidade ao decodificador, que precisa calcular seus autovetores.

Considerando que blocos da mesma posição t_0 (Fig. 2.18) em resíduos diferentes são similares, é possível usar uma única matriz **A** para representar blocos em diferentes resíduos, mitigando ainda mais seu impacto. A partir do modelo NN, os valores das arestas são

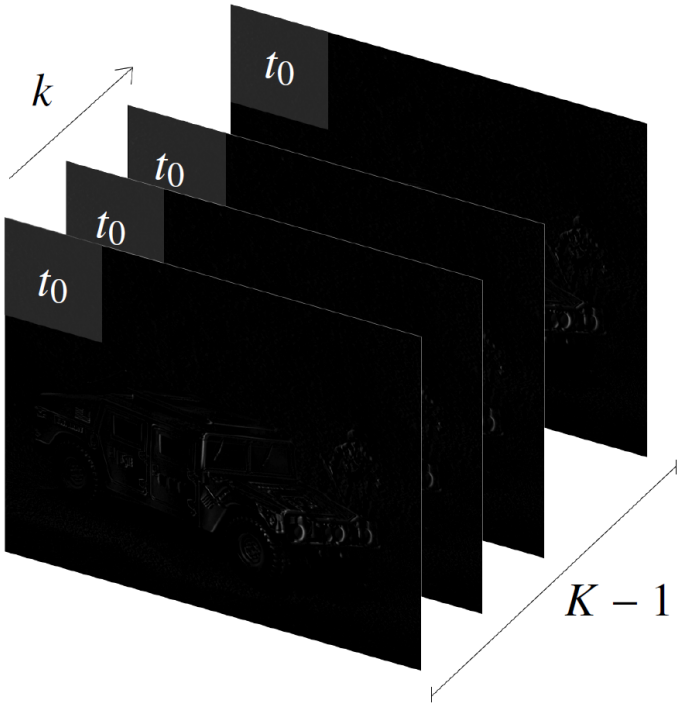


Figura 2.18 – Representação de blocos em uma dada posição t_0 em um grupo de $K-1$ resíduos.

calculados, para um único bloco, de modo a reduzir a distorção ℓ_2 introduzida pelo operador de atraso, isto é, $\|\mathbf{As}-\mathbf{s}\|_2$, onde $\mathbf{s}$ é o vetor contendo os valores dos pixels do bloco. Este raciocínio pode ser estendido para calcular uma matriz representativa de múltiplos blocos ao se minimizar $\sum_{k=k_1}^{k_2}\|\mathbf{As}_k-\mathbf{s}_k\|_2$, considerando blocos k_1 até k_2 em um grupo de $K-1$ blocos (por exemplo, um bloco para cada um dos $K-1$ resíduos de uma linha do *light field* com $K\times K$ vistas). Esta abordagem faz com que a eficiência da transformada seja mais uniforme quando a mesma matriz é aplicada para vários blocos. Se um único bloco for utilizado no cálculo de $\mathbf{A}$, a transformada será mais eficiente neste bloco [45].

Esta metodologia foi aplicada a uma base de dados composta por três *light fields* reais, capturados por uma câmera móvel, e a quatro *light fields* gerados em computador. Considerando que apenas uma matriz $\mathbf{A}$ é calculada para blocos numa mesma posição em cada linha do *light field*, foi analisada a eficiência da remoção de coeficientes e a distorção em erro quadrático médio (MSE, do inglês *mean squared error*), comparando com a DCT. A simulação considera 100 coeficientes mantidos da DCT (de um total de $32\times 32=1024$) e remove coeficientes da GFT enquanto o MSE resultante é menor que o alcançado pela DCT. Para cada posição de bloco, a matriz $\mathbf{A}$ foi calculada de duas formas: em função apenas do bloco pertencente ao resíduo central da linha, minimizando $\|\mathbf{As}-\mathbf{s}\|_2$; e em função de todos os resíduos da linha do *light field*, minimizando $\sum_{k=k_1}^{k_2}\|\mathbf{As}_k-\mathbf{s}_k\|_2$.

Gerando cada matriz $\mathbf{A}$ (associada a cada posição de bloco) a partir de blocos pertencentes apenas ao resíduo central, observa-se que a GFT é capaz de fornecer o mesmo MSE com 6% menos coeficientes que a DCT, na média dos 7 *light fields*, considerando também todos os coeficientes necessários para a transmissão da matriz $\mathbf{A}$. Este valor chega a 21,22% dependendo do *light field*. Como apenas blocos de um resíduo foram utilizados, a eficiência em outros resíduos (não considerados no cálculo de $\mathbf{A}$) é menor,

resultando em um desvio padrão médio de 5,3 coeficientes necessários por resíduo. Quando todos os resíduos são considerados na inferência da matriz de adjacências, a redução em número de coeficientes melhora para 8,4% na média dos *light fields* e o desvio padrão médio na quantidade de coeficientes por resíduo cai para 1,4. Embora o sistema de compressão adotado seja uma versão simplificada (com menos etapas e realizando eliminação de coeficientes ao invés de quantização) daqueles empregados no estado da arte, estes resultados mostram que a GFT é capaz de oferecer desempenho competitivo para compressão em relação a transformadas tradicionais.

Outras áreas de aplicação

Os exemplos anteriores ilustram o uso de GSP em processamento de redes de sensores, um contexto que naturalmente surge da definição do domínio do sinal como sendo um grafo. No entanto, outros cenários menos evidentes permitem a aplicação da teoria.

Primeiramente, o leitor deve lembrar do comentário de que uma imagem pode ser modelada como um grafo não-direcionado em forma de grade (Fig. 2.4b) [16], segundo o modelo NN, considerado na seção anterior. Com ajuste apropriado dos pesos das arestas, Sandryhaila e Moura mostraram que a GFT leva a uma compressão de imagens bastante eficiente, com erros de aproximação por truncamento do espectro menores do que aqueles utilizando-se a DFT, DCT ou DWT [24]. De fato, a análise espectral de imagens via DCT, conhecida por sua propriedade de redução de dimensionalidade, está diretamente relacionada à GFT segundo GSP_L, pois o Laplaciano do grafo da Fig. 2.4b, com pesos unitários, é diagonalizado pela matriz da DCT [46]. No contexto de imagens de *light field*, além do uso de GSP em compressão, a teoria também tem fornecido ferramentas para codificação [47] e esquemas de super-resolução [48]. Para uma revisão extensa sobre aplicação de GSP em compressão, restauração, filtragem e segmentação de imagens, os autores recomendam [49].

Outra importante área de estudo para GSP envolve redes biológicas (*biological networks*), i. e. o tratamento de sistemas em rede – humanos ou da natureza – para inferência ou processamento de dados. Por exemplo, com respeito às redes genéticas regulatórias, o uso de métodos baseados em grafos levou à melhoria dos três esquemas do estado-da-arte em inferência de redes [50, 51]. Nguyen *et al.* representaram o corpo humano como uma grade dinâmica tridimensional e aplicaram bancos de filtros de *wavelets* sobre grafos para comprimir os dados de posição e cor, superando os métodos usuais [52]. O processamento de sinais cerebrais, no entanto, parece ser a mais intrigante e prolífica dentre estas aplicações, surgindo pela atribuição de sinais como os de imagem por ressonância magnética funcional (fMRI, do inglês *functional magnetic resonance imaging*) a grafos definidos por redes cerebrais funcionais [53, 54]. Tais trabalhos têm demonstrado, por exemplo, correlação entre as componentes de baixa e alta frequências destes sinais e o processo de aprendizado de uma tarefa motora [55].

Como mencionado na subseção 2, a regularização de um sinal sobre grafos com

amostras discretas foi utilizada como método de classificação de dados em [25]. Este é um exemplo dos muitos usos de GSP em aprendizado de máquina, dos quais se pode também citar o aprendizado semi-supervisionado pelo uso de filtros adaptativos sobre grafos [56], sistemas de recomendação consistindo em filtragem colaborativa e baseada em conteúdo, utilizando regularização com variação total em grafos [57], e detecção de comunidades em redes via projeto de *wavelets* em grafos, o que permite mineração multiescala de comunidades [58].

Considerações sobre Implementação em *Software*

O rápido desenvolvimento da teoria de processamento de sinais sobre grafos, na última década, trouxe consigo o interesse pela criação de ferramentas computacionais que pudessem facilmente pô-la em prática. Embora os cálculos em GSP envolvam basicamente álgebra matricial, o que já é satisfatoriamente atendido por diversos *softwares* de computação científica, a implementação de ferramentas específicas justifica-se pela conveniência de se ter tipos de dados e operadores definidos diretamente para GSP, como classes particulares de grafos, transformadas e filtros, por exemplo.

As ferramentas típicas da comunidade

Certamente a primeira ferramenta com que o leitor terá contato, ao buscar seus primeiros códigos em GSP, será a GSPBOX[12]. Trata-se de uma *toolbox* em MATLAB criada pelo *Signal Processing Laboratory LTS2* da *Ecole Polytechnique Fédérale de Lausanne*, dentre cujos autores figuram nomes importantes para este capítulo, como David Shuman e Pierre Vandergheynst.

O GSPBOX foi construído sob a perspectiva de GSP_L e, portanto, representa os sinais sobre grafos segundo o autoespaço da matriz Laplaciana. O pacote traz uma grande quantidade de operadores, classes específicas de grafos e funções, todos com descrições disponíveis em sua documentação. Aqueles que não dispõem do MATLAB, podem executar os comandos básicos do GSPBOX no equivalente Octave, de código aberto. No entanto, algumas funções, como por exemplo a `gsp_nn_graph`, que cria um grafo conectado a partir de uma *pointcloud*, requerem outras *toolboxes* específicas, como a Statistics e a Machine Learning, o que pode ser um incômodo para o programador de Octave. A Fig. 2.19a traz um grafo toroidal e um dos autovetores de sua matriz Laplaciana, gerada via GSPBOX.

O segundo pacote para MATLAB que convém mencionar é o GraSP[13] [60], criado por Benjamin Girault para implementar algumas ferramentas da sua teoria de sinais

[12] Sua documentação, *links* para *download* e demais informações estão disponíveis no endereço: `https://epfl-lts2.github.io/gspbox-html/` [59]. O código é distribuído gratuitamente, sob licença GNU GPLv3.

[13] O pacote GraSP, com seus arquivos e documentação, está disponível em `https://gforge.inria.fr/projects/grasp/`.

estacionários sobre grafos (que não cobrimos neste capítulo), como a sua versão da operação de translação no domínio dos vértices. A motivação para a sua definição é criar um deslocamento que preserve a norma ℓ_2 do sinal, como ocorre na teoria clássica com sinais de tempo discreto. O resultado é um deslocamento que não preserva a norma ℓ_1 do sinal, e tem interpretação menos direta do que a translação pela matriz de adjacência de GSP_A. Veja-se, por exemplo, a Fig. 2.19b, em que a translação de Girault foi aplicada três vezes ao impulso unitário localizado no vértice v_{10}, e o sinal resultante não espalhou-se tanto a partir do vértice de origem, como ocorreria com o deslocamento usual pela matriz **A**.

Aspectos da implementação em Python

Aqueles que não possuem o MATLAB/Octave, ou já têm familiaridade com programação em Python, podem animar-se com a notícia de que Nathanaël Perraudin, membro da equipe do GSPBOX, uniu-se a outros pesquisadores para implementar as funcionalidades desta *toolbox* como uma biblioteca em Python, a PyGSP [61]. No entanto, tentamos instalar este pacote e não obtivemos sucesso, com inúmeras *exceptions* sendo continuamente acusadas. Como resultado, buscamos implementar em Python, do zero, algumas operações em GSP, tomando por base os pacotes usuais de computação científica da linguagem: Numpy, Scipy e Matplotlib. Esta pequena biblioteca de funções, referida doravante como GSPy[14], por comodidade, foi utilizada para gerar a maioria dos exemplos e gráficos de GSP neste capítulo. É importante ressaltar que o insucesso no contato com PyGSP, até onde sabemos, foi exceção, e não regra, e o leitor é encorajado a instalar e aproveitar as funcionalidades desta biblioteca. No entanto, compartilhamos a seguir alguns aspectos básicos considerados na escrita do GSPy, para aqueles que desejarem (ou tenham a necessidade de) produzir seu próprio código em Python para manusear sinais e ferramentas de GSP.

Assim como no GSPBOX, a exibição de um grafo requer apenas sua matriz de adjacência ponderada **A**, de dimensões $N \times N$, e um *array* $N \times 2$ de números reais, para armazenar as coordenadas dos vértices do grafo. A criação de uma imagem contendo o grafo requer dois comandos principais: as funções do Matplotlib `scatter()`, para representar os vértices, e `arrow()`, para arestas direcionadas (aquelas não-direcionadas consistem simplesmente de um gráfico de linha, `plot()`). Para exibir um *sinal* sobre um grafo, basta adicionar na chamada da função `scatter` o parâmetro de cor `c=signal`, em que `signal` é o vetor contendo o sinal em questão. A barra lateral contendo a escala pseudocolorida é exibida com o comando `colorbar()` do Matplotlib. Um sinal sobre um grafo em anel direcionado, gerado desta forma, é mostrado na Fig. 2.19c.

Uma operação fundamental que surge em aplicações de GSP é a GFT, que é rapidamente implementada utilizando as funções de álgebra linear do Numpy. O código 2.1 mostra a aplicação de um filtro passa-baixa a um sinal `s`, definido sobre um grafo de matriz

[14] O leitor é convidado a conferir o código, disponível no endereço `https://github.com/gboaviagem/GSPy`.

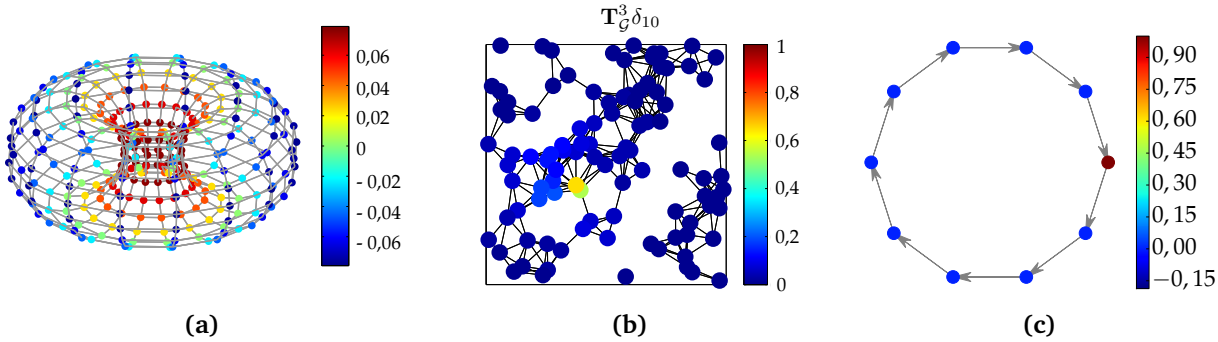


Figura 2.19 – (a) Autovetor sobre um grafo toroidal, gerado com o GSPBOX. (b) Operador de translação de Girault aplicado três vezes sobre um impulso num grafo de sensores, originalmente sobre o vértice amarelo, utilizando o GraSP. (c) Impulso unitário sobre o vértice v_0 de um grafo direcionado em anel, gerado pelo GSPy.

de adjacência `A`. A função `stem` e as constantes `fsize` (tamanho de fonte) e `msize` (tamanho do marcador) foram definidas em outro trecho do *script*.

```
import numpy as np
[eigvals,V] = np.linalg.eig(A) # eigendecomposition of the adjacency matrix
Vinv = np.linalg.inv(V)
ss = np.dot(Vinv,s) # GFT according to GSP_A

hh_LPF = 1.0 -  1.0/(1 + np.exp(2*(eigvals - np.mean(eigvals))/np.mean(np.abs(
    eigvals)))) # filter spectrum

plt.figure()
stem(eigvals,ss,fsize,msize,'b',':',r'\widehat{\mathbf{s}}$') # input signal
    spectrum
stem(eigvals,ss*hh_LPF,fsize,msize-2,'r',':',r'$h(\Lambda)*\widehat{\mathbf{s}}
    $') # output signal spectrum
```

Código 2.1 – Exemplo de filtragem passa-baixa num certo sinal `s`.

Algumas Oportunidades de Estudo

Como comentado, a abordagem de GSP_L foi desenvolvida considerando grafos não-direcionados, com pesos reais não-negativos. Nesses casos, $\mathbf{L}$ é diagonalizável e positiva semi-definida, o que gera propriedades convenientes para a aplicação dos conceitos teóricos, como ter frequências sempre reais e uma componente DC associada à frequência nula. O arcabouço de GSP_A apresenta a vantagem de considerar grafos gerais, desde que sem laços e múltiplas arestas, mas também traz desvantagens. Como comentado em [62], o uso da matriz de adjacência como bloco elementar e a aplicação a grafos direcionados carrega uma série de empecilhos quando a matriz $\mathbf{A} = \mathbf{V}_G\mathbf{J}\mathbf{V}_G^{-1}$ é não-diagonalizável. Por exemplo, há os problemas de instabilidade numérica, que ocorrem na decomposição de Jordan para matrizes de ordem prática, e o fato de que $\mathbf{V}_G$ é uma base de autovetores (generalizados) que não são em geral ortogonais, e por isso a GFT não preserva o produto escalar.

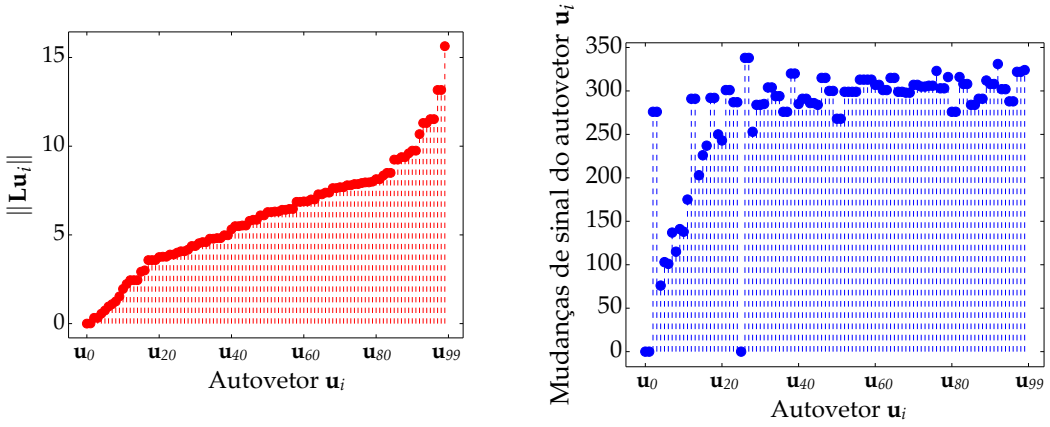


Figura 2.20 – (a) Norma $\|\mathbf{Lu}\|$ e (a) número de mudanças de sinal dos autovetores do grafo direcionado da Fig. 2.8, dispostos segundo a ordem crescente dos módulos dos respectivos autovalores.

Além disso, [63] mostra que ambas as abordagens ainda apresentam o problema da não-unicidade da definição da GFT, sempre que há frequências repetidas (λ_i ou γ_i). Pode-se ilustrar o problema com um exemplo numérico, utilizando um grafo completo de 3 vértices (um triângulo) não-ponderado (pesos unitários). Grafos completos de n vértices têm matriz Laplaciana *normalizada* com autovalores 0, com multiplicidade 1, e $n/(n-1)$, com multiplicidade $n-1$ [15, Exemplo 1.1]. Portanto, sua matriz Laplaciana não-normalizada pode ser expressa como

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} = \mathbf{U}_1 \grave{} \mathbf{U}_1^T = \begin{bmatrix} 1/\sqrt{3} & -1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{3} & 0 & 1/\sqrt{2} \\ 1/\sqrt{3} & 1/\sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ -1/\sqrt{2} & 0 & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \end{bmatrix},$$

mas também uma outra matriz de autovetores ortonormal é possível, substituindo os autovetores associados a $\gamma = 3$ pela soma e diferença (normalizadas) dos vetores $(-1/\sqrt{2}, 0, 1/\sqrt{2})^T$ e $(-1/\sqrt{2}, 1/\sqrt{2}, 0)^T$,

$$\mathbf{U}_2 = \begin{bmatrix} 1/\sqrt{3} & -2/\sqrt{6} & 0 \\ 1/\sqrt{3} & 1/\sqrt{6} & -1/\sqrt{2} \\ 1/\sqrt{3} & 1/\sqrt{6} & 1/\sqrt{2} \end{bmatrix}. \tag{2.57}$$

Isso leva a duas definições distintas da GFT, pois um mesmo sinal passa a ter um ou outro espectro, a depender se a matriz $\mathbf{U}_1$ ou $\mathbf{U}_2$ (ou qualquer uma das outras infinitas possibilidades) é considerada. Por exemplo, o sinal $\mathbf{x} = (-4 \quad 0 \quad 4)^T$ teria, neste caso, espectros dados por $\mathbf{U}_1^T\mathbf{x} \approx (0 \quad 5{,}66 \quad 0)^T$ e $\mathbf{U}_2^T\mathbf{x} \approx (0 \quad 4{,}9 \quad 2{,}83)^T$. Deri e Moura abordam este problema sugerindo o uso de projetores espectrais oblíquos (*oblique spectral projectors*) para obter uma representação única da GFT [63].

Embora GSP_L considere apenas os grafos não-direcionados, a teoria não impede *a priori* a aplicação a grafos direcionados, uma vez que a matriz Laplaciana continua sendo definida por $\mathbf{L} = \mathbf{D} - \mathbf{A}$, em que $\mathbf{D}$ é escolhida como a matriz de graus de entrada ou de saída.

Poderíamos, por exemplo, manter a definição de frequência como sendo os autovalores de **L**, pois teríamos que a norma $\|\mathbf{Lu}_i\|$ varia linearmente com *o módulo* de cada autovalor,

$$\|\mathbf{Lu}_i\| = \|\gamma_i \mathbf{u_i}\| = |\gamma_i|. \tag{2.58}$$

A Fig. 2.20 traz a norma $\|\mathbf{Lu}_i\|$ e o número de mudanças de sinal dos autovetores de **L** para o grafo *direcionado* da Fig. 2.8 (pág. 54), indicando que há relação entre o módulo dos autovalores e a suavidade do autovetor associado. No entanto, a matriz Laplaciana para grafos direcionados não é mais simétrica, por isso não é garantida ser positiva semi-definida ou diagonalizável.

Muitos outros problemas em GSP ainda se mantêm abertos à investigação, seja para desenvolver formas eficientes para se aplicar a teoria no contexto de redes e dados estruturados massivos (como o *método ágil inexato* de J. Deri [63, 64], que reduz o tempo de cálculo — de semanas para minutos — da GFT baseada em projetores espectrais), seja para melhor compreender e aplicar as técnicas utilizadas em cenários como os da segunda seção. Por exemplo, os Teoremas da Amostragem (vide seção 2) e da Incerteza para grafos já foram extensivamente estudados [37, 65, 66], mas ainda há oportunidade para análise, como acerca da robustez de amostragem de sinais não-perfeitamente limitados em faixa [67]. Ortega *et al.* também ressaltam que muitas das ferramentas de GSP foram criadas para qualquer tipo de grafo, muito embora a investigação de grafos específicos possa gerar novas possibilidades em GSP, para certas aplicações [68]. Trata-se, enfim, de um campo de estudo recente, abrangente e transversal, com diversas opções de direções para tomar. Se é verdade o que disse David Hilbert, que "somente enquanto um ramo da ciência oferecer uma abundância de problemas ele estará vivo"[15], então GSP será uma área de pesquisa ativa ainda por muitos anos.

Referências Bibliográficas

[1] K. M. Alam, M. Saini, and A. El Saddik. Toward social Internet of vehicles: Concept, architecture, and applications. *IEEE Access*, 3:343–357, 2015.

[2] L. Guo, Z. Ning, Q. Song, L. Zhang, and A. Jamalipour. A QoS-oriented high-efficiency resource allocation scheme in wireless multimedia sensor networks. *IEEE Sensors J.*, 2016.

[3] F. Ma, B. Yao, and M. Yao. Non-planar unclustered peterson graphs as scale-free models of the Internet of things. In *Information Technology, Networking, Electronic and Automation Control Conference, IEEE*, pages 1040–1043. IEEE, 2016.

[15] "*As long as a branch of science offers an abundance of problems, so long is it alive; a lack of problems foreshadows extinction or the cessation of independent development*". Discurso *Mathematical Problems*, Segundo Congresso Internacional de Matemáticos, Paris, 1900.

[4] L. Yu, L. Feng, C. Chen, T. Qiu, L. Li, and J. Wu. A novel multi-feature representation of images for heterogeneous IoTs. *IEEE Access*, 4:6204–6215, 2016.

[5] F. Chung. Graph theory in the information age. *Notices of the AMS*, 57(6):726–732, 2010.

[6] A. J. Golubski, E. E. Westlund, J. Vandermeer, and M. Pascual. Ecological networks over the edge: Hypergraph trait-mediated indirect interaction (TMII) structure. *Trends in Ecology & Evolution*, 31(5):344–354, may 2016.

[7] G. I. McKinsey et al. Big data: the next frontier for innovation, competition and productivity. *McKinsey Global Institute*, 2011.

[8] R. K. Jain, J. M. F. Moura, and C. E. Kontokosta. Big data+ big cities: Graph signals of urban air pollution [exploratory sp]. *IEEE Signal Process. Mag.*, 31(5):130–136, 2014.

[9] J. Mei and J. M. F. Moura. Signal processing on graphs: Causal modeling of unstructured data. *IEEE Trans. Signal Process.*, 2016.

[10] A. Sandryhaila and J. M. F. Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *IEEE Signal Process. Mag.*, 31(5):80–90, 2014.

[11] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83–98, 2013.

[12] Paulo Feofiloff, Yoshiharu Kohayakawa, and Yoshiko Wakabayashi. Uma Introdução Sucinta à Teoria dos Grafos. 2011. `https://www.ime.usp.br/~pf/teoriadosgrafos/`.

[13] Cláudio Leonardo Lucchesi. *Introdução à teoria dos grafos*. IMPA, 1979.

[14] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. Graph theory, volume 244 of graduate texts in mathematics, 2008.

[15] Fan R. K. Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[16] Aliaksei Sandryhaila and José MF Moura. Nearest-neighbor image model. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 2521–2524. IEEE, 2012.

[17] Stefania Sardellitti, Sergio Barbarossa, and Paolo Di Lorenzo. Graph topology inference based on transform learning. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, dec 2016.

[18] Markus Püschel and José M. F. Moura. Algebraic Signal Processing Theory. *CoRR*, abs/cs/0612077, 2006.

[19] Markus Püschel and José M. F. Moura. Algebraic signal processing theory: Foundation and 1-D time. *IEEE Trans. Signal Process.*, 56(8):3572–3585, 2008.

[20] Markus Püschel and José M. F. Moura. Algebraic signal processing theory: 1-D space. *IEEE Trans. Signal Process.*, 56(8):3586–3599, 2008.

[21] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs. *IEEE Trans. Signal Process.*, 61(7):1644–1656, 2013.

[22] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs: Graph filters. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Institute of Electrical and Electronics Engineers (IEEE), may 2013.

[23] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. *Signals and Systems*. Prentice-Hall signal processing series. Prentice Hall, 1997.

[24] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs: Graph Fourier transform. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Institute of Electrical and Electronics Engineers (IEEE), may 2013.

[25] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Trans. Signal Process.*, 62(12):3042–3054, jun 2014.

[26] Jean Gallier. Spectral theory of unsigned and signed graphs – applications to graph clustering: a survey, December 2017.

[27] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. A windowed graph Fourier transform. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 133–136. Ieee, 2012.

[28] Serge Lang. Álgebra linear, 2003.

[29] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[30] S. K. Narang, A. Gadde, and A. Ortega. Signal processing techniques for interpolation in graph structured data. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5445–5449, May 2013.

[31] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević. Discrete signal processing on graphs: Sampling theory. *IEEE Trans. Signal Process.*, 63(24):6510–6523, dec 2015.

[32] P. Di Lorenzo, S. Barbarossa, P. Banelli, and S. Sardellitti. Adaptive least mean squares estimation of graph signals. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):555–568, Dec 2016.

[33] P. Di Lorenzo, E. Isufi, P. Banelli, S. Barbarossa, and G. Leus. Distributed recursive least squares strategies for adaptive reconstruction of graph signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 2289–2293, Aug 2017.

[34] Paolo Di Lorenzo, Paolo Banelli, Elvin Isufi, Sergio Barbarossa, and Geert Leus. Adaptive graph signal processing: Algorithms and optimal sampling strategies. *CoRR*, abs/1709.03726, 2017.

[35] P. S. R. Diniz. *Adaptive Filtering: Algorithms and Practical Implementation*. Springer, 2013.

[36] S. Chen, A. Sandryhaila, and J. Kovačević. Sampling theory for graph signals. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3392–3396, April 2015.

[37] Luiz F. O. Chamon and Alejandro Ribeiro. Greedy sampling of graph signals. *IEEE Trans. Signal Process.*, 66(1):34–47, jan 2018.

[38] B. Widrow and M. E. Hoff. Adaptive switching circuits. *WESCOM Conv. Rec.*, (pt 4):96–140, Aug 1960.

[39] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson. Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings of the IEEE*, 64(8):1151–1162, Aug 1976.

[40] P. Di Lorenzo, P. Banelli, and S. Barbarossa. Optimal sampling strategies for adaptive learning of graph signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1684–1688, Aug 2017.

[41] Jorge Nocedal and Stephen J Wright. Numerical optimization 2nd, 2006.

[42] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 31–42, New York, NY, USA, 1996. ACM.

[43] Wei Hu, Gene Cheung, Antonio Ortega, and Oscar C. Au. Multiresolution graph fourier transform for compression of piecewise smooth images. *IEEE Transactions on Image Processing*, 24(1):419–433, jan 2015.

[44] G. Shen, W.-S. Kim, S. K. Narang, A. Ortega, Jaejoon Lee, and HoCheon Wey. Edge-adaptive transforms for efficient depth map coding. In *28th Picture Coding Symposium*. IEEE, dec 2010.

[45] Vitor Elias and Wallace Martins. On the use of graph fourier transform for light-field compression. *Journal of Communication and Information Systems*, 33(1):92–103, 2018.

[46] Giulia Fracastoro, Sophie M Fosson, and Enrico Magli. Steerable discrete cosine transform. *IEEE Trans. Image Process.*, 26(1):303–314, 2017.

[47] Xin Su, Mira Rizkallah, Thomas Maugey, and Christine Guillemot. Graph-based light fields representation and coding using geometry information. In *IEEE International Conference on Image Processing (ICIP)*, 2017.

[48] Mattia Rossi and Pascal Frossard. Graph-based light field super-resolution. In *Multimedia Signal Processing (MMSP), 2017 IEEE 19th International Workshop on*, pages 1–6. IEEE, 2017.

[49] Gene Cheung, Enrico Magli, Yuichi Tanaka, and Michael K. Ng. Graph spectral image processing. *Proc. IEEE*, pages 1–24, 2018.

[50] Aurélie Pirayre, Camille Couprie, Frédérique Bidard, Laurent Duval, and Jean-Christophe Pesquet. Brane cut: biologically-related a priori network enhancement with graph cuts for gene regulatory network inference. *BMC Bioinformatics*, 16(1):368, 2015.

[51] Aurélie Pirayre, Camille Couprie, Laurent Duval, and Jean-Christophe Pesquet. Brane clust: Cluster-assisted gene regulatory network inference refinement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2017.

[52] Ha Q Nguyen, Philip A Chou, and Yinpeng Chen. Compression of human body sequences using graph wavelet filter banks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6152–6156. IEEE, 2014.

[53] Leah Goldsberry, Weiyu Huang, Nicholas F Wymbs, Scott T Grafton, Danielle S Bassett, and Alejandro Ribeiro. Brain signal analytics from graph signal processing perspective. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 851–855. IEEE, 2017.

[54] Nora Leonardi and Dimitri Van De Ville. Tight wavelet frames on multislice graphs. *IEEE Trans. Signal Process.*, 61(13):3357–3367, 2013.

[55] Weiyu Huang, Leah Goldsberry, Nicholas F Wymbs, Scott T Grafton, Danielle S Bassett, and Alejandro Ribeiro. Graph frequency analysis of brain signals. *IEEE J. Sel. Topics Signal Process.*, 10(7):1189–1203, 2016.

[56] Siheng Chen, Fernando Cerda, Piervincenzo Rizzo, Jacobo Bielak, James H Garrett, and Jelena Kovačević. Semi-supervised multiresolution classification using adaptive graph filtering with application to indirect bridge structural health monitoring. *IEEE Trans. Signal Process.*, 62(11):2879–2893, 2014.

[57] Kirell Benzi, Vassiiis Kalofolias, Xavier Bresson, and Pierre Vandergheynst. Song recommendation with non-negative matrix factorization and graph total variation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2439–2443. IEEE, 2016.

[58] Nicolas Tremblay and Pierre Borgnat. Graph wavelets for multiscale community mining. *IEEE Trans. Signal Process.*, 62(20):5227–5239, 2014.

[59] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, August 2014.

[60] Benjamin Girault, Shrikanth S. Narayanan, Antonio Ortega, Paulo Gonçalves, and Eric Fleury. Grasp: A matlab toolbox for graph signal processing. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2017.

[61] Michaël Defferrard, Lionel Martin, Rodrigo Pena, and Nathanaël Perraudin. PyGSP: Graph signal processing in python, 2017.

[62] Stefania Sardellitti, Sergio Barbarossa, and Paolo Di Lorenzo. On the graph Fourier transform for directed graphs. *IEEE J. Sel. Topics Signal Process.*, 2017.

[63] Joya A Deri and José MF Moura. Spectral projector-based graph Fourier transforms. *IEEE J. Sel. Topics Signal Process.*, 11(6):785–795, 2017.

[64] Joya A. Deri. *Graph Signal Processing: Structure and Scalability to Massive Data Sets*. PhD thesis, Carnegie Mellon University, 2016.

[65] Siheng Chen, Rohan Varma, Aarti Singh, and Jelena Kovačević. Signal recovery on graphs: Fundamental limits of sampling strategies. *IEEE Trans. Signal Inf. Process. Netw.*, 2(4):539–554, 2016.

[66] Xiaohan Wang, Jiaxuan Chen, and Yuantao Gu. Generalized graph signal sampling and reconstruction. In *Signal and Information Processing (GlobalSIP), 2015 IEEE Global Conference on*, pages 567–571. IEEE, 2015.

[67] Mikhail Tsitsvero, Sergio Barbarossa, and Paolo Di Lorenzo. Signals on graphs: Uncertainty principle and sampling. *IEEE Trans. Signal Process.*, 64(18):4845–4860, 2016.

[68] Antonio Ortega, Pascal Frossard, Jelena Kovacevic, Jose M. F. Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proc. IEEE*, 106(5):808–828, may 2018.

CAPÍTULO

3 Design de Antenas Planares utilizando o *High Frequency Structure Simulator* (HFSS) da ANSYS®

Alison C. da Silva (UFCG), Bruno L. Nobre (UFCG), Camila Caroline R. de Albuquerque (UFCG), Carolina C. P. e Silva (UFCG), Marina Lua Ferreira (UFCG), Samuel M. A. Morais (UFCG), Alexandre Jean R. Serres (UFCG)

Introdução

O que é uma antena?

Por definição, uma antena é um dispositivo criado para **transmitir** ou **receber** energia eletromagnética. Muitas vezes também são chamadas de sistemas irradiantes.

A informação original é alterada através de algum tipo de modulação e tratamento, por exemplo, e é transmitida ou guiada por um cabo até chegar a **antena transmissora**. Esta antena então irradia essa informação pelo meio (ar), até que ela chega a outra antena (**antena receptora**), que nesse caso fará a recepção do sinal; o sinal continua o caminho por um cabo até o dispositivo que fará a demodulação (e outros tratamentos), recuperando a informação original.

A informação é preservada porque a antena atua como um **transdutor** (ou seja, um dispositivo que converte um tipo de energia em outro) casando os condutores que geram esses campos. Por exemplo na transmissão, o campo eletromagnético gerado corresponde a determinada tensão e corrente alternada. Já na recepção, a mesma referência de tensão e corrente alternada será induzida.

Principais tipos de antenas

A **antena isotrópica** é uma antena ideal que irradia de forma uniforme em todas as direções, ou seja, sua irradiação tem formato esférica. Apesar de ser impossível de ser construída na prática, a antena isotrópica serve como uma referência para os parâmetros das antenas reais.

As antenas a seguir são os tipos mais comumente vistos em nosso dia a dia.

- **Dipolo:** A antena dipolo, normalmente, consiste de uma antena que tem metade do tamanho do comprimento de onda irradiado. Dentre suas características principais está incluso seu diagrama de radiação do tipo *broadside*. Não emitindo energia no sentido de posicionamento do seu elemento, mas sim em sentido perpendicular. Seu formato clássico se assemelha a uma maçã.

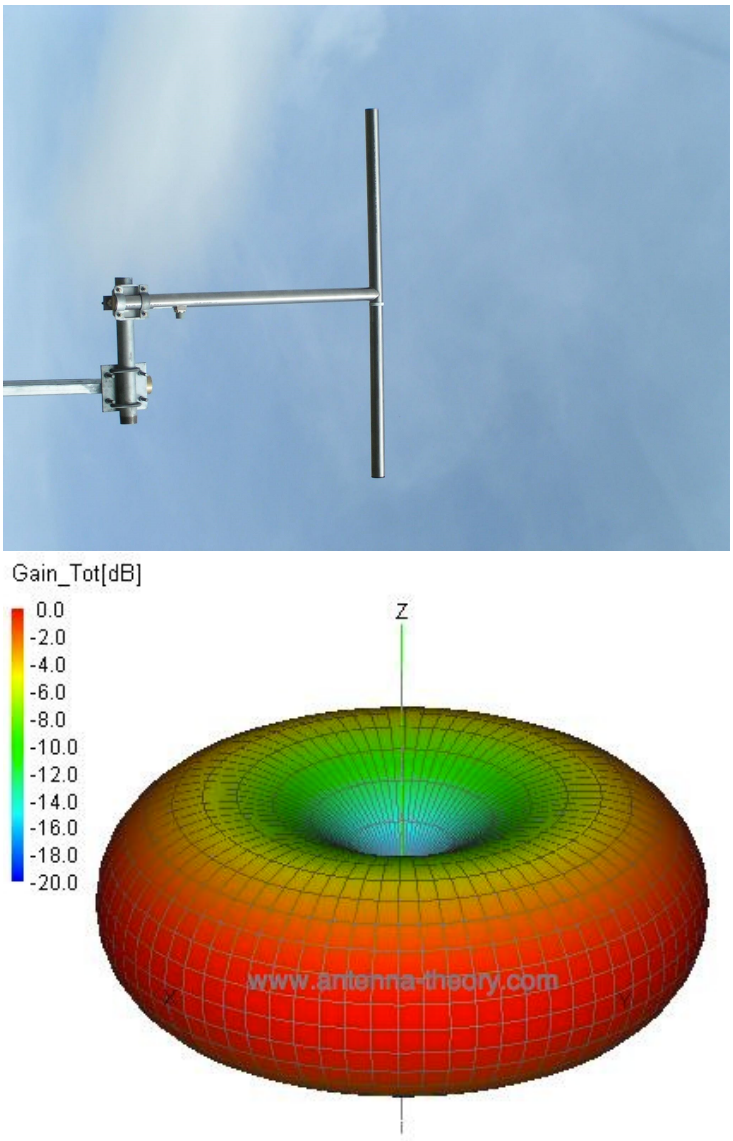


Figura 3.1 – Antena Dipolo e o seu Diagrama de Irradiação, respectivamente.

- **Monopolo:** A antena monopolo terá, basicamente, as **características da dipolo divididas por dois**. Ela terá o tamanho de um quarto do comprimento de onda irradiado e seu diagrama de radiação é basicamente o da Figura 3.2 cortado no meio. Ela é uma figura muito presente em paisagens urbanas.

Figura 3.2 – Antena Monopolo

- **Corneta**: Para entender o funcionamento dessa antena podemos traçar um paralelo com instrumentos musicais como o trompete, saxofone ou um clarinete. Todos eles possuem aquela abertura circular na extremidade, com um raio crescente em relação ao resto do instrumento (Campânula). Essa parte do instrumento tem como objetivo amplificar o som gerado pelo músico. O mesmo acontece com a antena corneta, que irá pegar o sinal gerado e aumenta-lo gradualmente até que ele saia da sua extremidade, transformando uma onda guiada em onda irradiada. Essa antena costuma trabalhar em UHF (frequências acima de 300 MHz) e é bem diretiva.

- **Antenas Planares:** As antenas planares, diferentemente do que vimos até agora, são como gravuras impressas em um livro. Onde ao invés da tinta utilizamos algum material condutor (como o cobre) e ao invés de papel teremos algum substrato (como o PET). Essas antenas dispõem de uma gama de diferentes aplicações. Podem

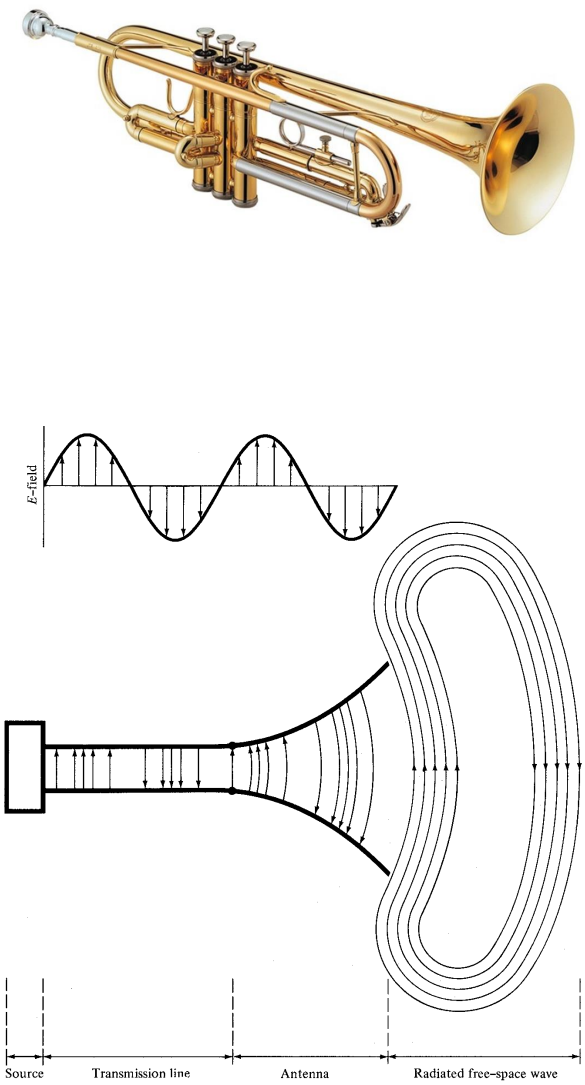


Figura 3.3 – Um exemplo de instrumento com campânula e o processo de irradiação de uma antena corneta, respectivamente.

servir como sensores, sistemas de segurança, controle de estoque, entre várias outras aplicações. Podem ser rígidas ou flexíveis, ativas ou passivas, grandes ou pequenas. Várias tecnologias de ponta envolvem a utilização de antenas planares em algum nível.

Introdução às Principais Ferramentas do *Software*

Antes de começarmos a aprender a simular nossas antenas, precisamos conhecer as ferramentas disponíveis no HFSS, as principais ferramentas podem ser vistas na tabela a seguir.

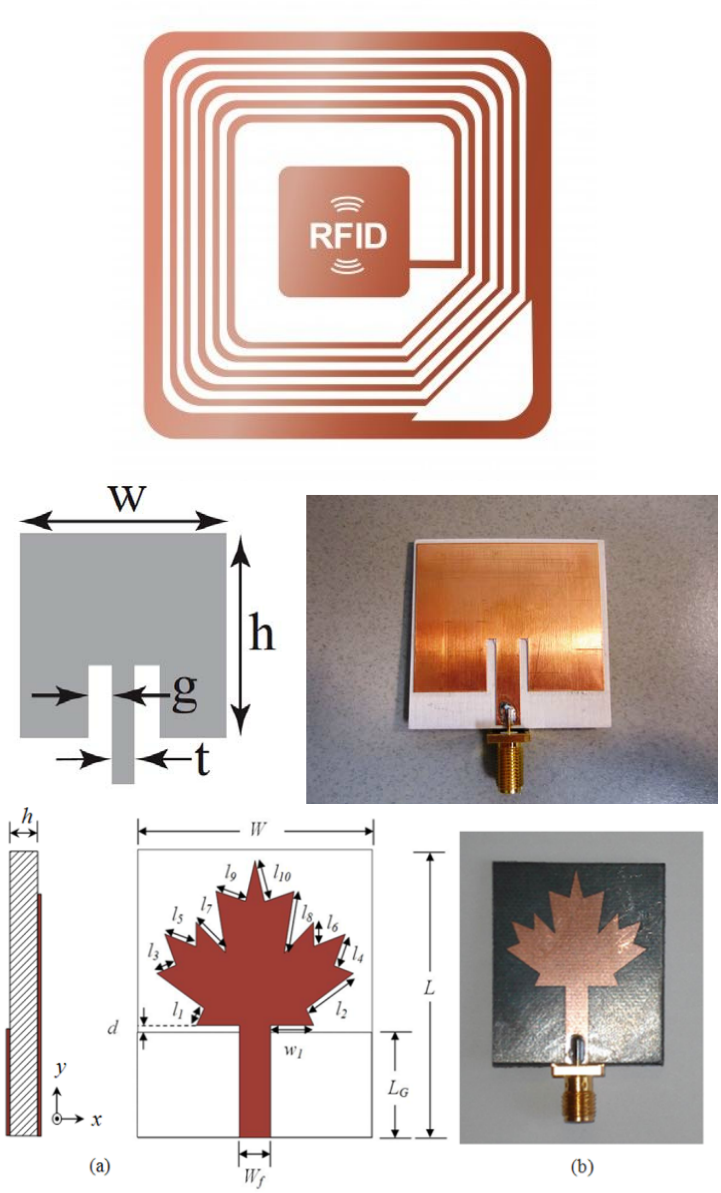


Figura 3.4 – Tag de identificação por radiofrequência, uma antena Patch e uma antena bioinspirada, respectivamente.

Tabela 3.1 – Ferramentas do HFSS.

Ferramenta	Descrição
	Cria um novo projeto
	Insere um novo projeto no HFSS
	Usado para abrir projetos já existentes
	Usado para salvar projetos
	Cola componentes ou partes de um projeto
	Copia componentes ou partes de um projeto

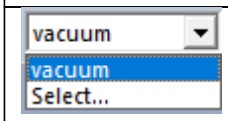
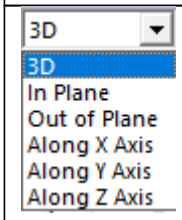
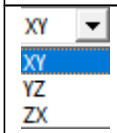
Continua na próxima página.

Tabela 3.1 – continuação da página anterior

Ferramenta	Descrição
	Desfaz o último comando
	Refaz o último comando
	Verifica se estão corretas as excitações, contornos, *setups* e outras etapas do projeto
	Cria um *setup* de análise do projeto a partir da frequência desejada
	Permite deslocar o espaço de visualização arrastando o mouse com o botão esquerdo pressionado
	Rotaciona em torno do centro do modelo
	Rotaciona em torno do centro da tela
	Rotaciona em torno do eixo atual
	Rotaciona em torno da posição do cursor
	Amplia/reduz ao arrastar o mouse com o botão esquerdo pressionado
	Usa o botão esquerdo do mouse para criar um retângulo definindo a área de ampliação
	Usa o botão esquerdo do mouse para criar um retângulo definindo a área de redução
	Ajusta todo o projeto na tela para melhor visualização
	Ajusta os desenhos selecionados na tela para melhor visualização
	Desfaz a última modificação de visualização
	Refaz a última modificação de visualização
	Escolhe os objetos que deseja ocultar/mostrar na vista ativa
	Oculta os objetos selecionados na vista ativa
	Oculta os objetos selecionados em todas as vistas
	Mostra os objetos selecionados na vista ativa
	Mostra os objetos selecionados em todas as vistas
	Mostra apenas os objetos selecionados na vista ativa
	Mostra apenas os objetos selecionados em todas as vistas
	Desenha um retângulo
	Desenha um círculo
	Desenha um polígono regular (a partir do número de arestas)

Continua na próxima página.

Tabela 3.1 – continuação da página anterior

Ferramenta	Descrição
	Desenha uma elipse
	O cursor adere a um vértice do objeto
	O cursor adere a um ponto da grade
	Fornece a lista de opções de materiais a serem atribuídos a um objeto
	Atribui um material a um objeto
	Duplica um objeto espelhando-o em relação a um plano
	Duplica um objeto em torno de um eixo
	Duplica um objeto em relação a uma linha reta
	Espelha o objeto em relação a um plano
	Rotaciona o objeto em relação a um eixo (X, Y ou Z)
	Move um objeto a partir de um ponto
	Imprime a geometria de um objeto sobre outro
	Divide objetos que estão nos planos XY, YZ ou XZ
	Cria um novo objeto a partir da intersecção de dois ou mais objetos já existentes
	Retira objetos de um objeto já existente
	Une dois ou mais objetos em um único objeto
	Move o cursor para um ponto no espaço 3D relativo ao ponto de referência
	Seleciona o plano a ser desenhado no pelo comando "*Draw Plane*"a partir de uma lista
	Desenha um plano
	Desenha um ponto
	Desenha uma região que engloba os objetos no projeto atual

Continua na próxima página.

Tabela 3.1 – continuação da página anterior

Ferramenta	Descrição
	Desenha um espiral (2D ou 3D)
	Desenha uma hélice (3D)
	Desenha um toroide
	Desenha uma esfera (3D)
	Desenha um cone (3D)
	Desenha um poliedro regular (3D)
	Desenha um cilindro
	Desenha uma caixa
	Fornece ajuda online para o *script* a partir da barra de menu
	Fornece ajuda online para os conteúdos do software a partir da barra de menu
	Desenha uma superfície a partir de uma equação
	O cursor adere ao ponto central de uma aresta. O ponto pode ser de um objeto em 1D, 2D ou 3D.
	O cursor adere ao ponto central de uma face do objeto.
	O cursor adere ao ponto entre o inicio e o centro da aresta.
	O cursor adere ao ponto no centro de um arco.
	Habilita a grade no plano de fundo.
	Habilita a régua na parte inferior da área de trabalho
	Alterna entre o objeto com as faces pintadas ou só as arestas.
	Abre uma janela onde é possível escolher partes do objeto para esconder/mostrar no desenho.
	Cria um novo ponto de origem à uma certa distância do ponto de origem inicial.
	Cria um novo ponto de origem com os eixos rotacionados.
	Cria um novo ponto de origem á uma distância do original e com os eixos rotacionados.
	Insere um layout em 3D do HFSS
	Insere um projeto do HFSS-IE

Continua na próxima página.

Tabela 3.1 – continuação da página anterior

Ferramenta	Descrição
	Insere um projeto do Q3D extractor
	Insere um projeto do Q2D extractor
	Insere um projeto de circuito
	Insere um projeto de *netlist* do circuito
	Insere um projeto do Maxwell 3D
	Insere um projeto do Maxwell 2D
	Insere um projeto do RMxprt
	Insere um projeto do circuito de Maxwell
	Insere um projeto do *Simplorer*
	Insere um item do *Filter Design*
	Seleciona a vista superior
	Seleciona a vista inferior
	Seleciona a vista lateral direita
	Seleciona a vista lateral esquerda
	Seleciona a vista frontal
	Seleciona a vista posterior
	Seleciona a vista e perspectiva trimétrica
	Seleciona a vista e perspectiva dimétrica
	Seleciona a vista e perspectiva isométrica

Construção de Estruturas

Ferramentas de construção, materiais e camadas

Uma antena planar, em sua forma mais simples, consiste em 3 camadas principais: **elemento irradiante**, **substrato** e **plano de terra**.

O elemento irradiante é uma camada de material metálico e, como o próprio nome expressa, é responsável pelo processo de irradiação das ondas eletromagnéticas na antena. Seu projeto (comprimento, largura e tipo de alimentação) influencia diretamente no comportamento da mesma. O plano de terra (ou plano refletor) também é uma camada de material metálico, cuja influência está relacionada à distribuição espacial da potência

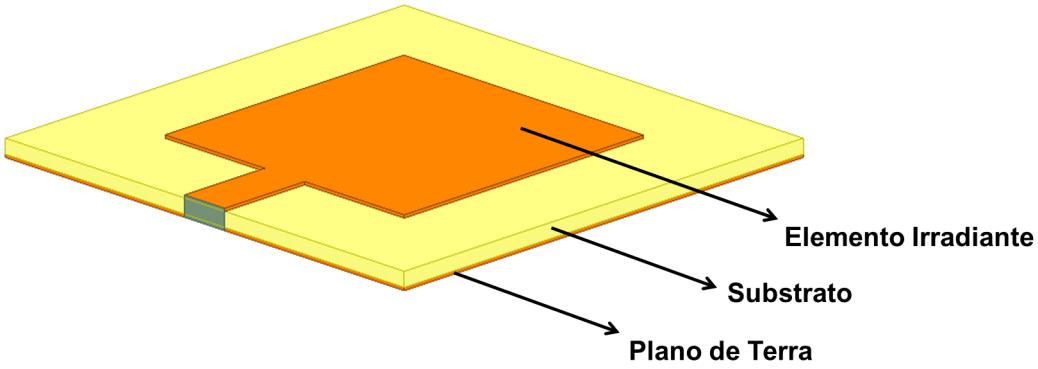
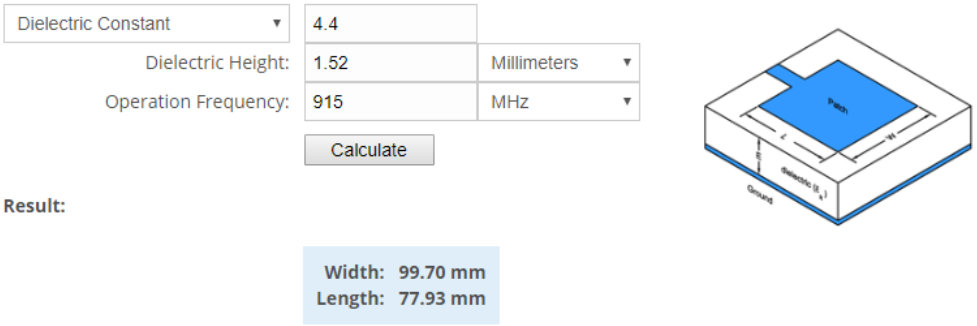


Figura 3.5 – Camadas principais de uma antena planar.

irradiada na antena. O substrato, por sua vez, é uma camada de material dielétrico, cujos principais parâmetros são constante dielétrica, tangente de perdas e altura.

Vamos começar a projetar uma antena *patch*. O primeiro passo é se utilizar dos cálculos para aproximar as dimensões do material condutor. Utilizando uma ferramenta online, temos nossas dimensões aproximadas para uma antena que funcione a 915Mhz utilizando FR4 como substrato e cobre como condutor.



$$Width = \frac{c}{2f_o\sqrt{\frac{\varepsilon_R+1}{2}}}; \quad \varepsilon_{eff} = \frac{\varepsilon_R+1}{2} + \frac{\varepsilon_R-1}{2}\left[\frac{1}{\sqrt{1+12\left(\frac{h}{W}\right)}}\right]$$

$$Length = \frac{c}{2f_o\sqrt{\varepsilon_{eff}}} - 0.824h\left(\frac{(\varepsilon_{eff}+0.3)\left(\frac{W}{h}+0.264\right)}{(\varepsilon_{eff}-0.258)\left(\frac{W}{h}+0.8\right)}\right)$$

Figura 3.6 – Cálculo para dimensões aproximadas da antena *patch*. *(Fonte: https://www.pasternack.com/t-calculator-microstrip-ant.aspx)*

Assim, nossa meta é chegar a uma frequência de ressonância de 915 MHz como o maior ganho possível. Como essas equações da Figura 3.6 são apenas aproximações, ajustes serão feitos mais a frente.

Vamos começar construindo a camada do substrato. Para isso, utilizamos o botão *Draw Box*  e construímos uma caixa com dimensões 149.7mm × 127.93mm × 1.52mm, porque o substrato deve ser um pouco maior que a parte do cobre, então estimamos um aumento de 50mm para todos os lados como ponto de partida. Na barra lateral direita (Figura 3.7), clicamos duas vezes sobre *CreateBox* e, na janela que será aberta (Figura 3.8),

podemos editar as dimensões do objeto criado de acordo com os parâmetros do substrato que será utilizado para a construção da antena.

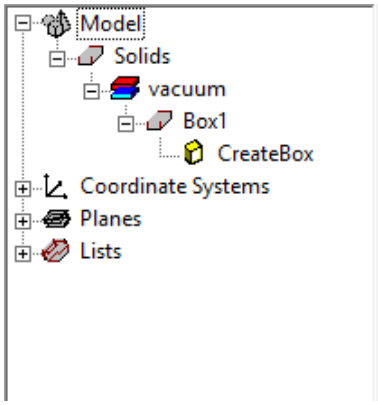


Figura 3.7 – Barra lateral direita.

- Em *Position*, determinamos as coordenadas do ponto de início da construção da caixa [-(99.7+50)/2 ,-(77.93+50)/2 ,-(1.52/2)];
- Em *XSize*, *YSize* e *ZSize* determinamos as dimensões da caixa, usando valores positivos ou negativos dependendo da direção a ser seguida em cada eixo cartesiano a partir do ponto de início (Ex.: $XSize = 104.7$mm, $YSize = 127.93$mm e $ZSize = 1.52$mm).

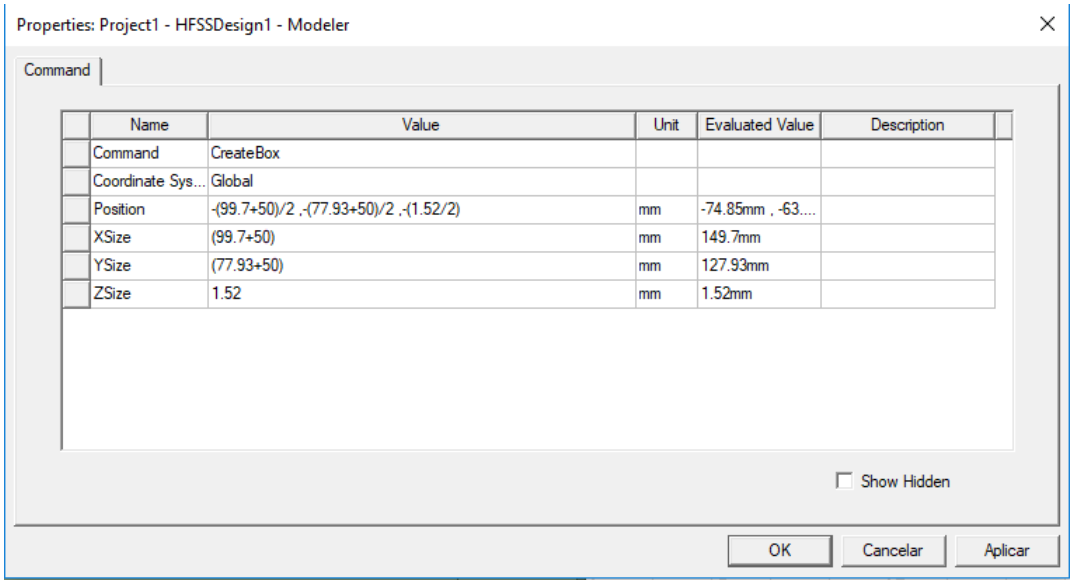


Figura 3.8 – Editar as dimensões do objeto criado.

Com os valores de exemplo, construímos um substrato de 104.7mm × 127.93mm, com 1.52mm de espessura e centralizado na origem dos eixos catersianos, como mostrado na Figura 3.9.

Na mesma barra lateral, se clicarmos duas vezes sobre *Box1* podemos modificar algumas características da caixa criada na janela que será aberta (Figura 3.10):

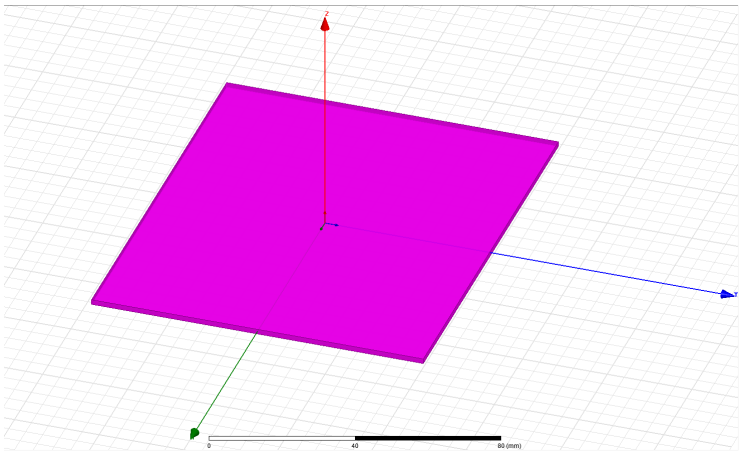


Figura 3.9 – Base para o substrato.

- Em *Name*, podemos renomear o objeto (recomenda-se que todos os objetos criados no projeto sejam renomeados com o nome do que representam);
- Podemos também alterar a sua cor (*Color*) e a transparência (*Transparent*);
- Nesta janela, um dos itens mais importantes é o *Material*, onde selecionaremos o material que constitui aquela camada específica. O caminho *Material* → *Edit* abre uma nova janela (Figura 3.11) onde é possível pesquisar os materiais pelo nome ou propriedade (permissividade, permeabilidade, tangente de perdas, etc.), como também adicionar novos materiais. Para este projeto, selecionaremos o substrato de FR4 ($\varepsilon_r = 4.4$ e $tg\Theta$=0.02). A dimensão de 1.52mm utilizada anteriormente é referente ao valor de espessura mais comum deste substrato.

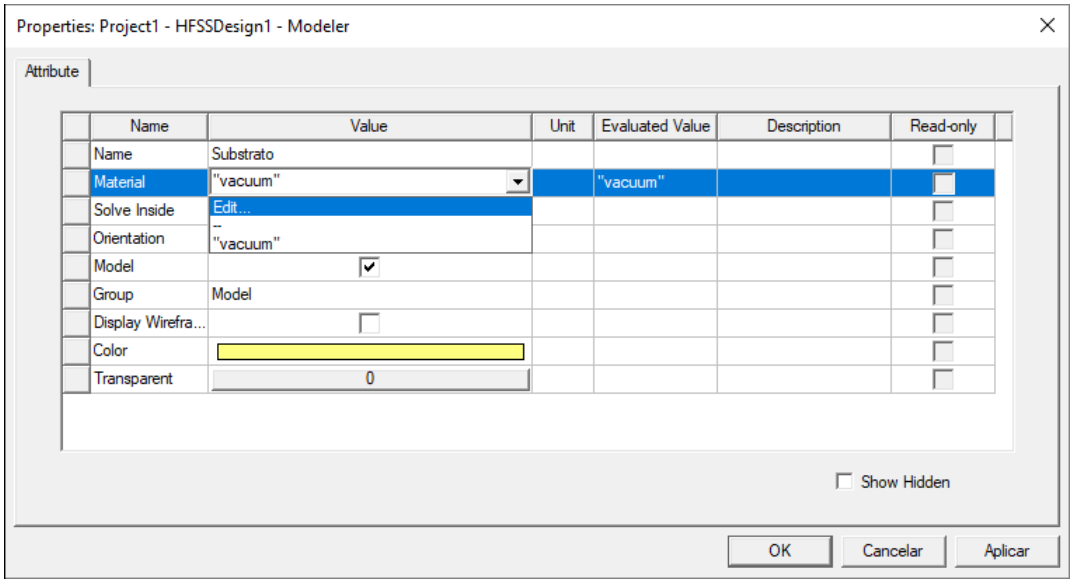


Figura 3.10 – Editar as características da camada do subtrato.

Da mesma forma, construímos a camada do plano de terra. Em *CreateBox* (Figura 3.12):

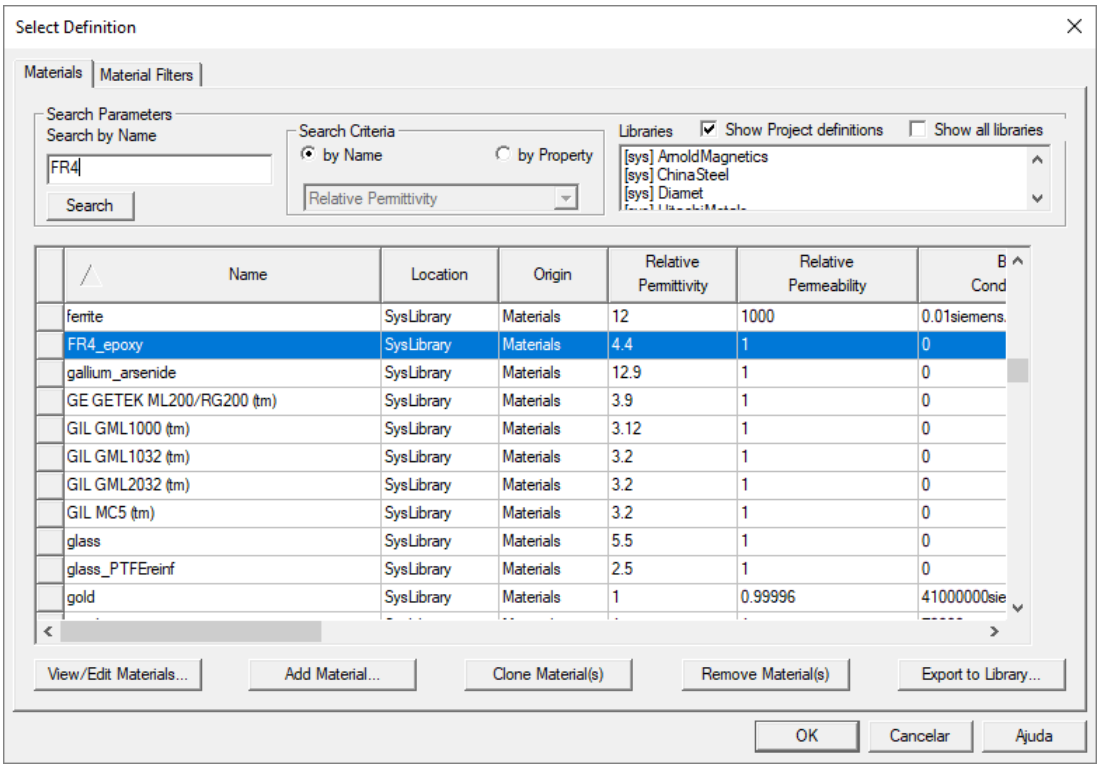


Figura 3.11 – Selecionar o material que compõe a camada do substrato.

- *Position*: -52.35mm, -63.965mm, -0.795m;
- *XSize* = 104.7mm, *YSize* = 127.93mm e *ZSize* = -0.035mm

O valor negativo do *ZSize* significa que a partir da posição (*Position*) $z = 0$mm, esta camada terá 0.032mm de dimensão no sentido negativo do eixo z.

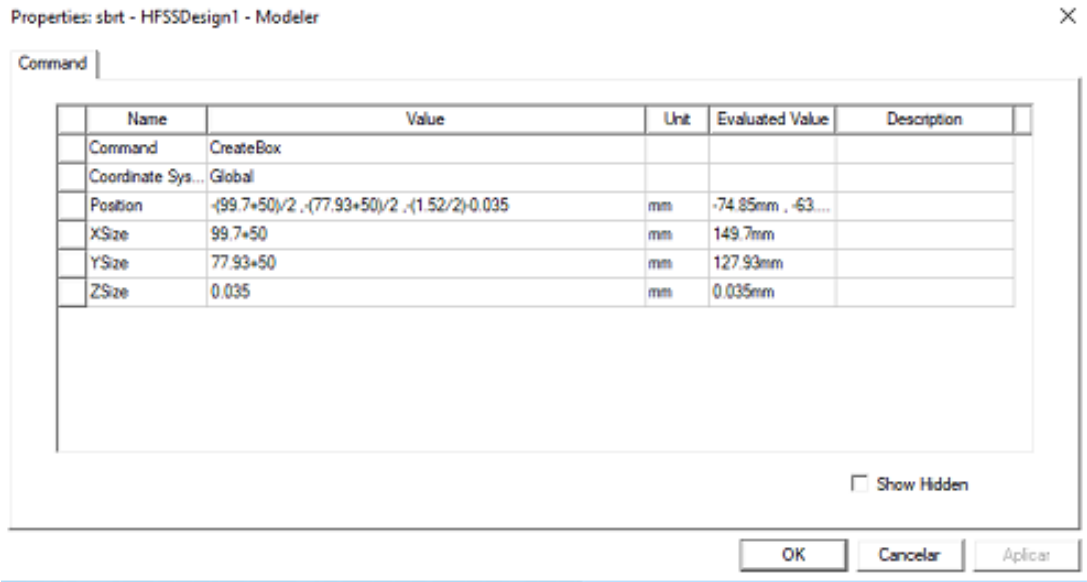


Figura 3.12 – Editar as características da camada do plano de terra.

Com dois cliques sobre *Box1* (Figura 3.13), renomeamos a camada e definimos o material como cobre (*copper*).

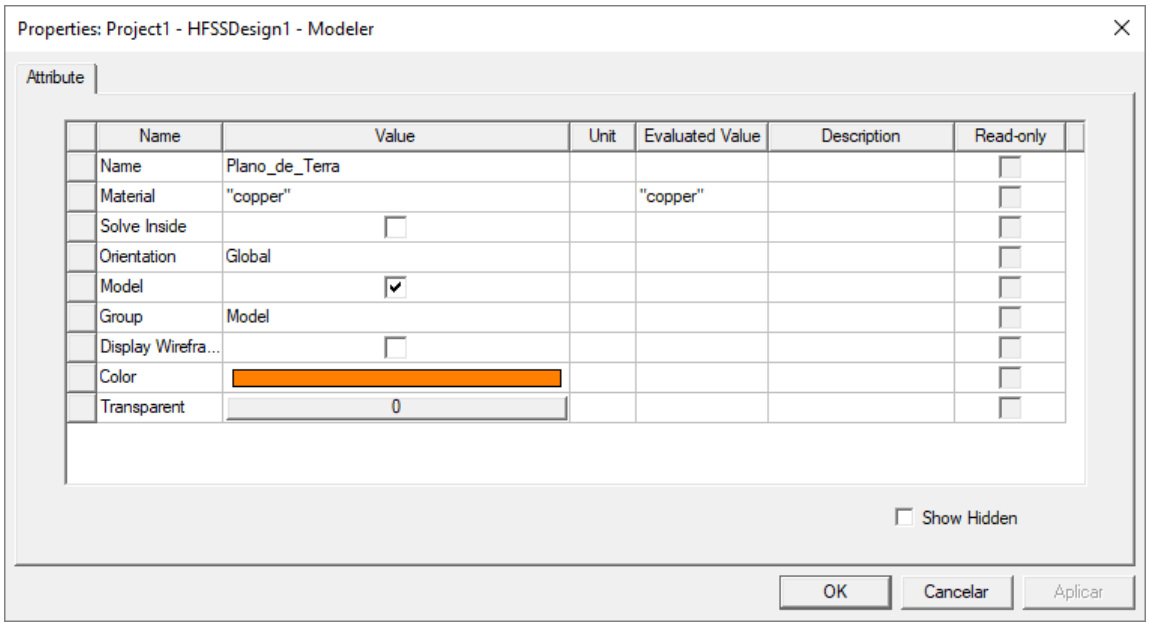


Figura 3.13 – Selecionar o material que compõe a camada do plano de terra.

Começamos a construção da camada superior pela linha de alimentação (*feed line*). A linha de alimentação, como mostra a Figura 3.5, começa a partir de uma das arestas do substrato e segue até o *patch* irradiante. Construímos, portanto, para a linha de alimentação uma caixa com as seguintes dimensões:

- *Position*: -2mm,38.965mm,0.76mm;
- $XSize = 4$mm, $YSize = 25$mm e $ZSize = 0.035$mm.

E, em seguida, construímos o *patch* irradiante:

- *Position*: 49.85mm, 38.965mm, -0.76mm;
- $XSize = 99.7$mm, $YSize = 77.93$mm e $ZSize = 0.035$mm.

Lembre-se de renomear os objetos de acordo com sua respectiva função e alterar o material de ambos para cobre. Depois, selecionamos esses objetos simultaneamente e clicando uma vez com o botão direito, seguimos o caminho *Edit* → *Boolean* → *Unite*, para unir as duas estruturas em uma única (Figura 3.14).

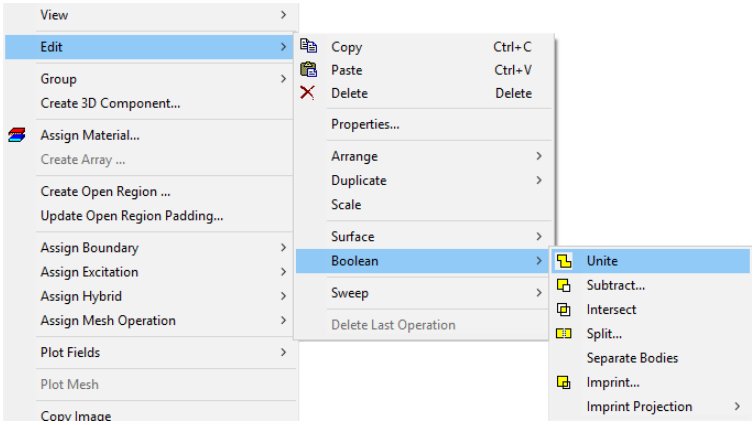


Figura 3.14 – Comando para unir objetos.

Obtendo a Figura 3.15.

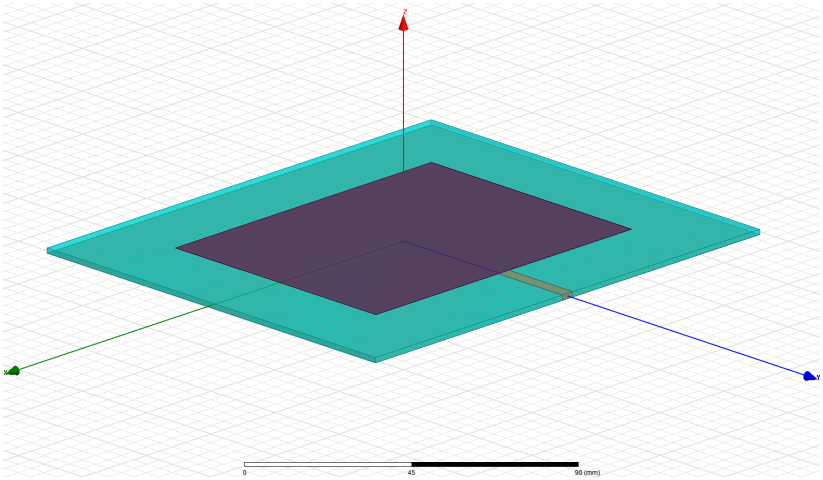


Figura 3.15 – Antena Construída

Alimentação

Utilizamos o botão *Draw Rectangle* e criamos uma superfície qualquer. Na barra lateral direita (Figura 3.16) clicamos duas vezes sobre *CreateRectangle* e, na janela que será aberta (Figura 3.17), podemos editar as dimensões da superfície criada.

- Em *Position*, determinamos as coordenadas do ponto de início da construção da superfície (Ex.: 25mm, -2.5mm, 1.55mm);
- Em *Axis*, alteramos o eixo normal à superfície, que neste caso será o eixo x;
- Em *YSize* e *ZSize* determinamos as dimensões da superfície, usando valores positivos ou negativos dependendo da direção a ser seguida em cada eixo cartesiano a partir do ponto de início (Ex.: *YSize* = 5mm e *ZSize* = -1.58mm).

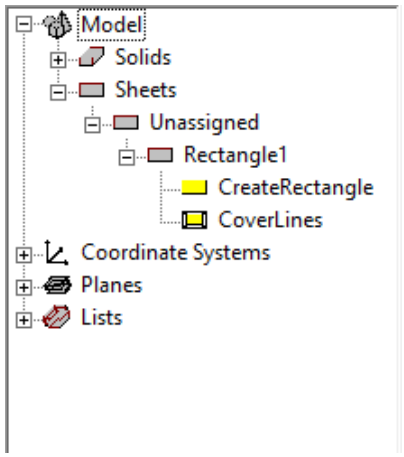


Figura 3.16 – Barra lateral direita.

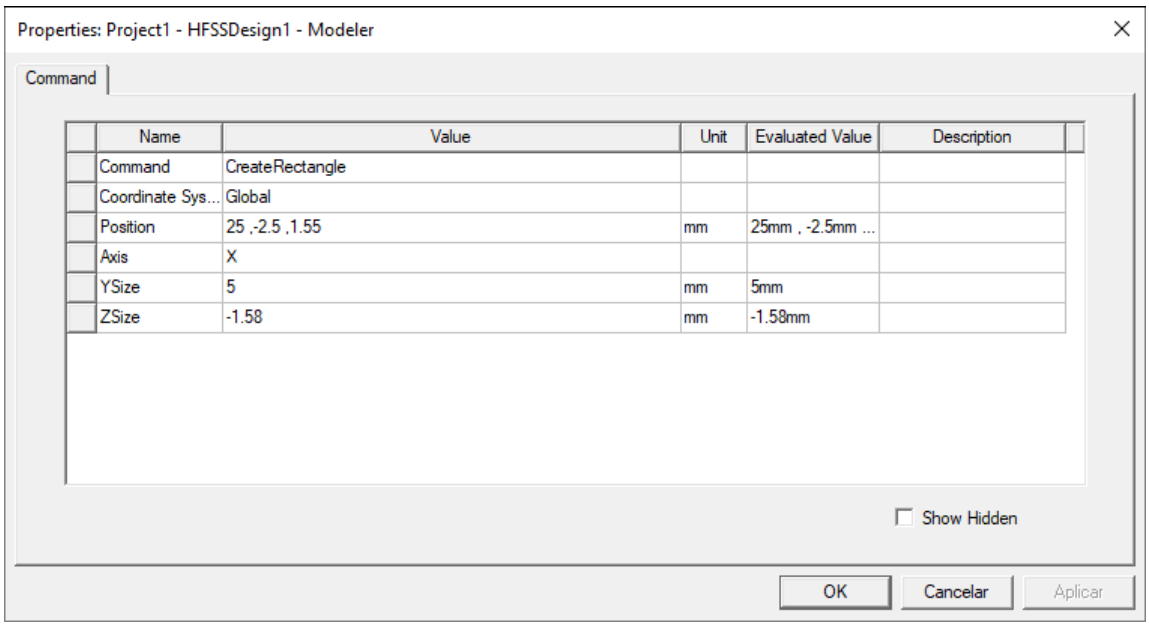


Figura 3.17 – Editar o eixo e as dimensões da superfície criada.

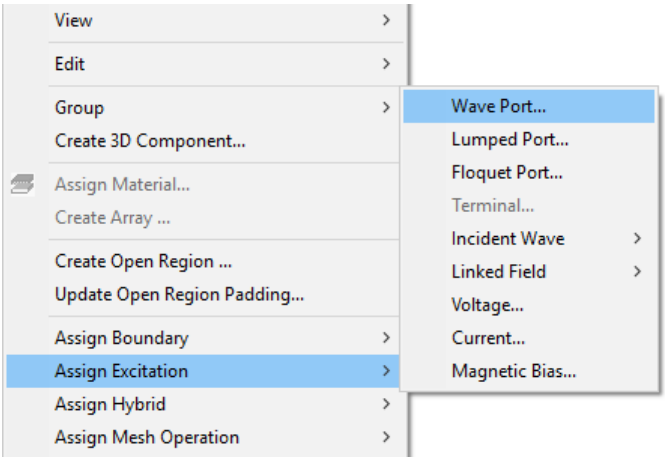


Figura 3.18 – Editar o eixo e as dimensões da superfície criada.

Feito isso, selecionamos o retângulo criado e, clicando com o botão direito, definimos ele como uma excitação. Para isso, seguimos o caminho *Assign Excitation* → *Wave Port*.

Na nova janela que será aberta (Figura 3.19), selecionamos na coluna *Integration Line* a opção *New Line*, e construímos a linha da porta da forma representada na Figura 3.20.

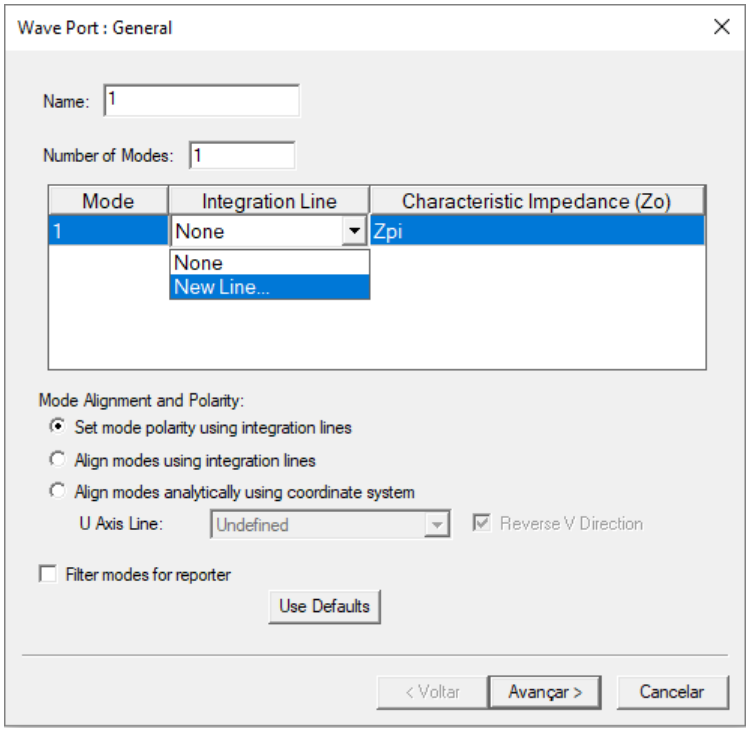


Figura 3.19 – Editar as propriedades da excitação.

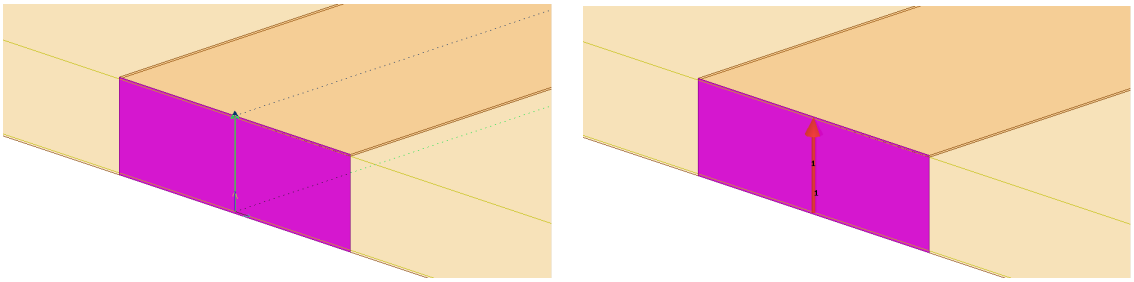


Figura 3.20 – Criação da linha de alimentação.

Criada a linha, clique em Avançar e, na janela seguinte, clique em Concluir para finalizar.

Projeto de variáveis

Uma ferramenta bastante útil no design e otimização de projetos no *HFSS* é a criação de variáveis. Para isto, criamos um elemento do projeto seguindo normalmente o procedimento explicado em 2.2.1, contudo, ao inserir as dimensões de *XSize*, *YSize* e *ZSize* ao clicar em *CreateBox*, antes com valores numéricos, agora devem ser inseridas variáveis - de preferência com nomes intuitivos para facilitar o manuseio das mesmas posteriormente.

Assim, primeiramente, criaremos uma variável para denominar a dimensão x do substrato, *substratex*, inserindo-a no campo *XSize*. Ao fazê-lo, uma nova janela irá abrir, como mostra a Figura 3.21. Nela, aparecerá o nome da nova variável em *Name*, o tipo de variável (*Unit Type*), que escolheremos *Length* por se tratar de um comprimento, a unidade (*Unit*) - que escolheremos mm - e o valor (*Value*), que a princípio deixaremos em 149.7mm, como calculado previamente.

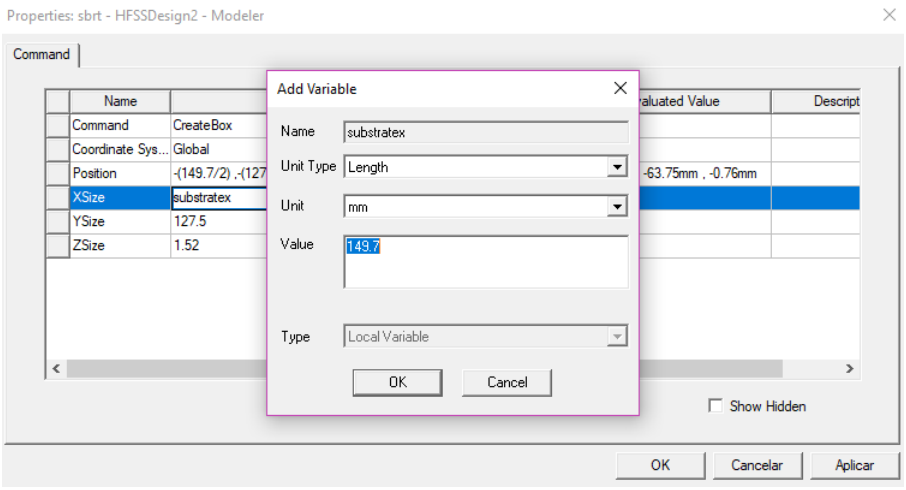


Figura 3.21 – Criação de uma nova variável

A Figura 3.22 mostra o uso de variáveis na criação do substrato, nas quais *substratex* e *substratey* foram utilizadas para denominar as dimensões x e y do substrato, respectivamente, e a variável *h* foi utilizada para denominar a dimensão z do substrato, ou sua altura (*height* em inglês, por isso o uso da letra h).

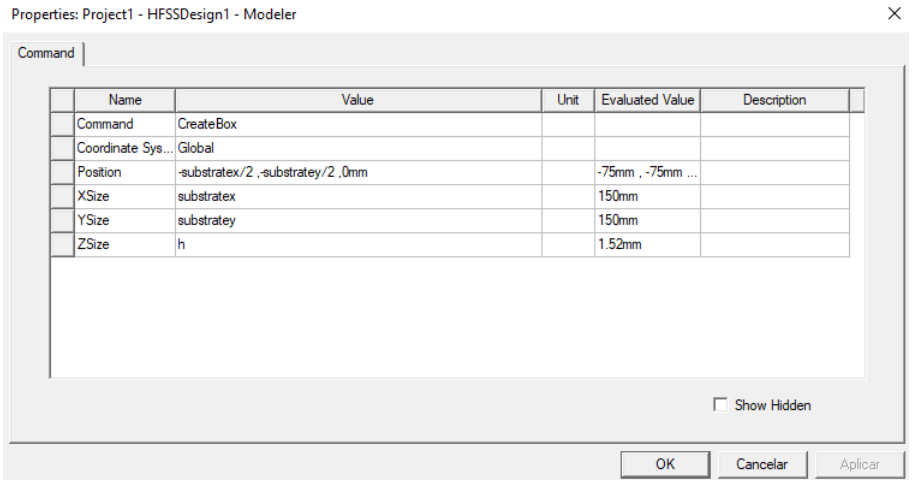


Figura 3.22 – Utilização de variáveis

Na Figura 3.22 é possível observar, também, que a *Position* foi alterada, isto foi feito para que o substrato fique centralizado na origem dos eixos X e Y e para que o plano de terra da antena fique no valor 0 do eixo Z, crescendo positivamente. Esta escolha se deu como uma forma de padronizar uma referência intuitiva para o projeto e facilitar a adição

de novos elementos no desenho. Futuramente, este passo será de extrema importância na otimização do *design*, uma vez que ao mudar a dimensão de uma variável é interessante que o desenho acompanhe tal mudança.

Continuaremos, então, inserindo variáveis para parametrizar o restante do projeto da antena planar.

Com o substrato criado, será parametrizada agora a linha de alimentação da antena. Iremos editar as dimensões da mesma forma que foi feito na seção 2.2.2. A Figura 3.23 mostra os valores inseridos, calculados previamente. Note que em *Position* uma análise deve ser feita para que o novo elemento seja bem posicionado. Os valores inseridos garantem que a linha de alimentação inicie na lateral do substrato e cresça em direção ao seu interior (de forma negativa) e que fique centralizado em relação à sua espessura (*Ly*).

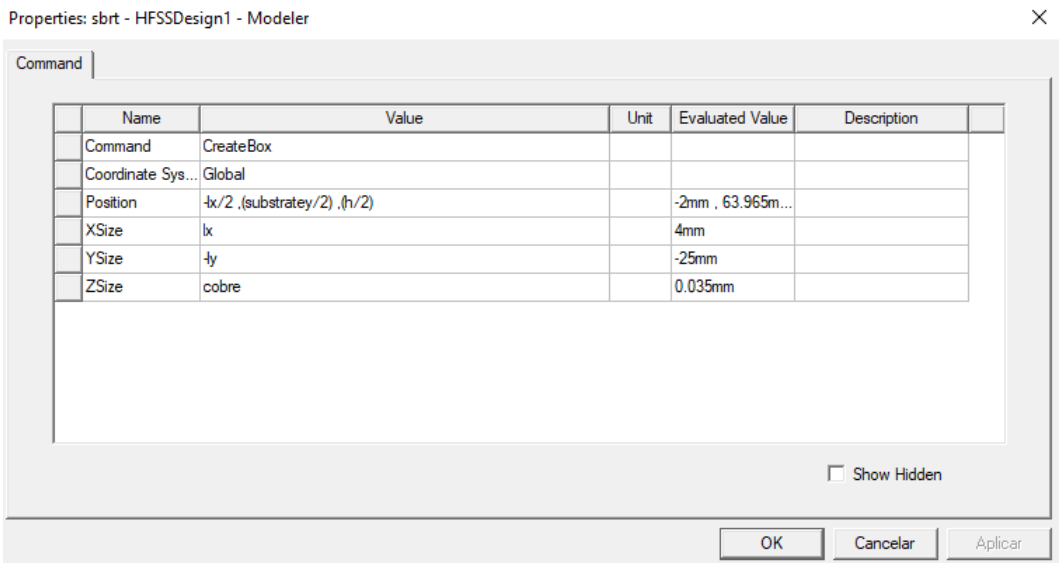


Figura 3.23 – Parametrização da linha de alimentação

Agora iremos modificar a antena *patch* em si, para isto, iremos também criar variáveis específicas para o elemento irradiante e utilizaremos os valores calculados na Seção 2.2.1. A Figura 3.24 mostra os valores inseridos e as novas variáveis criadas para as dimensões x, y e z da antena (*patchx*, *patchy*, *cobre* respectivamente), as relações inseridas em *Position* localizaram a antena no final da linha de transmissão e centralizada em relação ao eixo y.

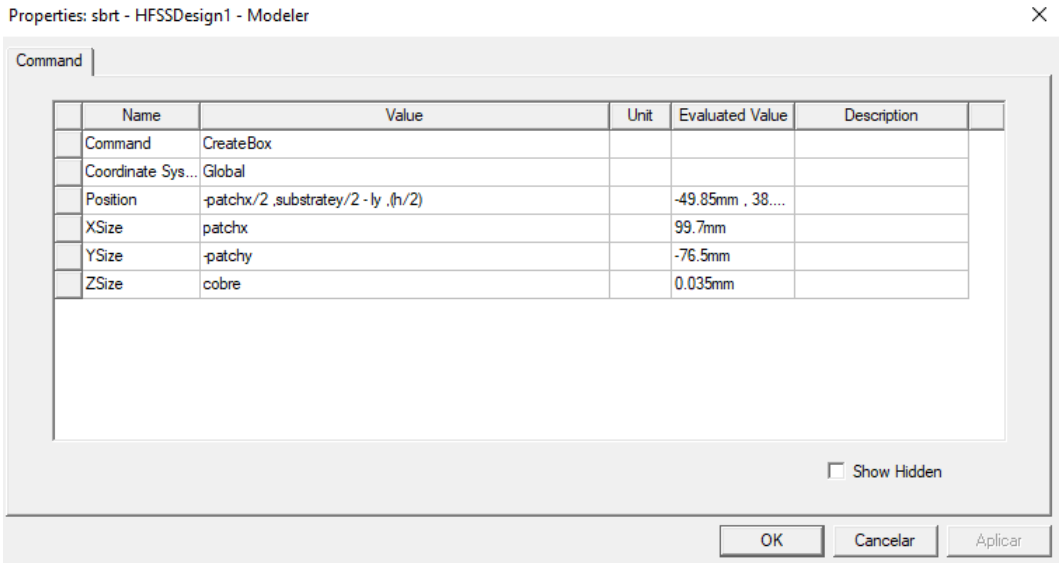


Figura 3.24 – Parametrização da antena

Modificaremos, ainda, a alimentação da antena como explicado na seção 2.2.2. Para isto, não precisaremos criar novas variáveis uma vez que suas dimensões são comuns a variáveis já existentes, como a largura da linha de alimentação (Ly) e a altura do substrato (h). As novas dimensões da alimentação são mostradas na Figura 3.25.

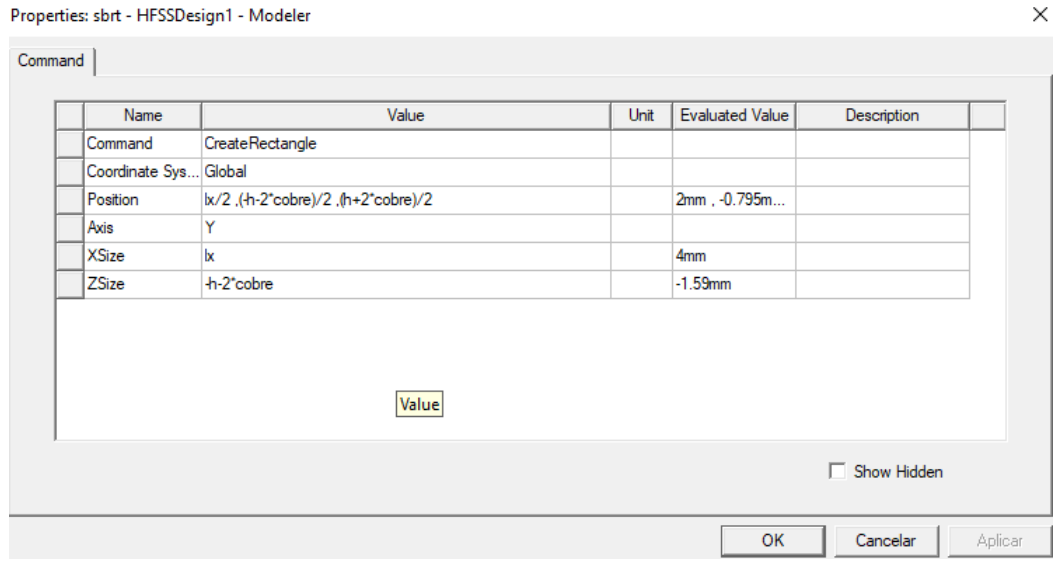


Figura 3.25 – Parametrização da alimentação

Para o plano de terra da antena também não será necessário criar novas variáveis, uma vez que suas dimensões são as mesmas do substrato (fora a espessura, que tem o valor do cobre - 0.035mm) e inserimos as dimensões explicitadas na Figura 3.26.

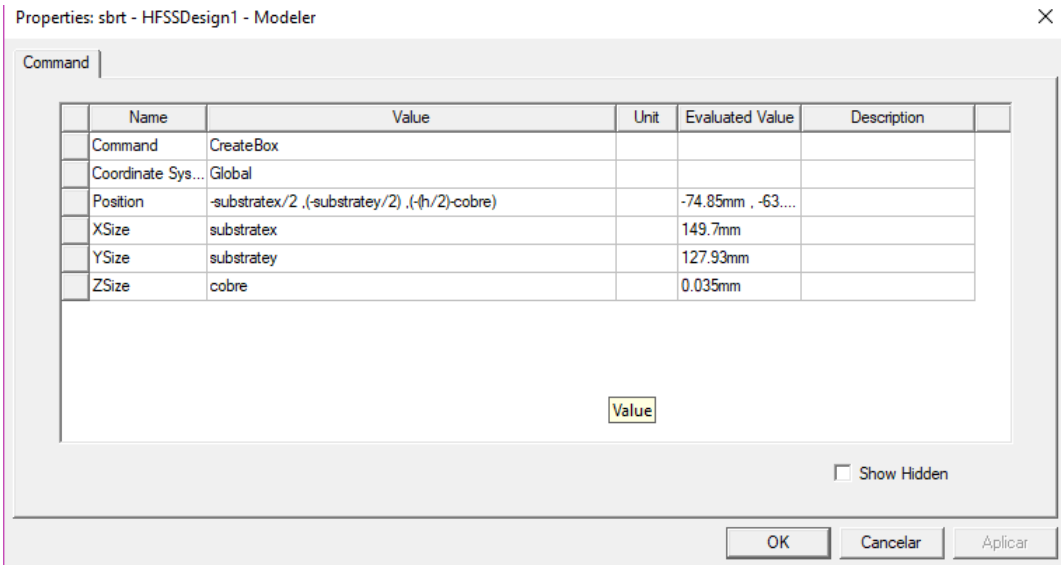


Figura 3.26 – Parametrização do plano de terra

Condições de Fronteira

Para que o HFSS realize suas simulações é necessário acrescentar uma região de radiação para simular as condições de propagação das ondas eletromagnéticas. Para isto, utilizaremos a ferramenta *Create Region*, mostrada na Figura 3.27.

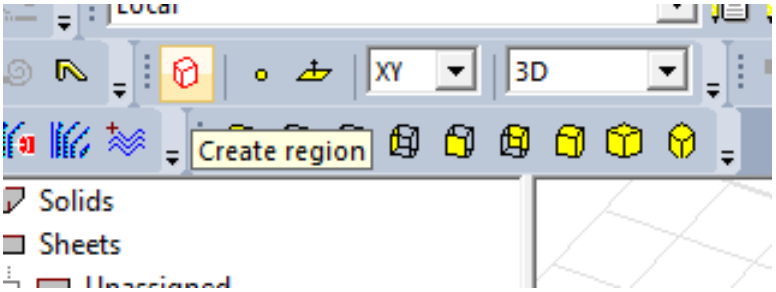


Figura 3.27 – Criação da região de radiação

Ao criar a nova região, uma nova janela irá se abrir (Figura 3.28). Selecionaremos a opção *Pad all directions similarly*e, e seguida escolheremos o *Padding type* como *Absolute Offset* para que tenhamos um valor fixo em todas as direções. Este valor é recomendado que seja, no mínimo, um quarto do comprimento de onda, isto é, $\lambda/4$. No nosso caso, para uma frequência de 915MHz, $\lambda/4$ é aproximadamente 81.97mm. Como este é o valor mínimo para validar o funcionamento da antena, utilizamos o valor 100mm na aba *Value*.

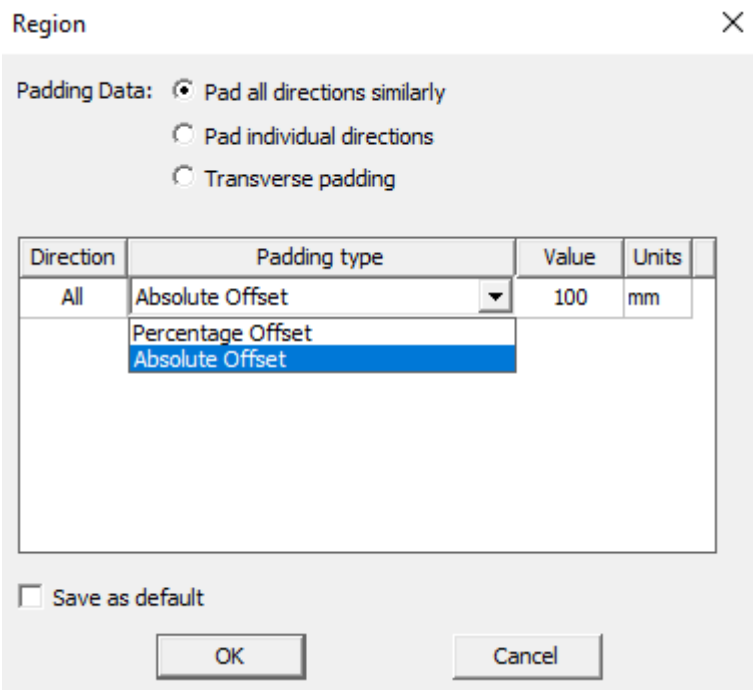


Figura 3.28 – Parâmetros da região de radiação

Além disto, como criamos retângulos para representar as regiões metálicas da antena e não sólidos, como apresentado na seção 2.2.1, é necessário indicar para o HFSS que se tratam de superfícies condutoras. Para isto, após selecionar os elementos correspondentes à antena e à linha de alimentação e uni-los (seguindo o caminho *Edit* → *Boolean* → *Unite* - Figura 3.14), clicamos com o botão direito em cima da nova estrutura criada e clicamos em *Assign Boundary* → *Perfect E...* para indicar que trata-se de um condutor perfeito, em seguida, uma nova janela irá se abrir confirmando o nome da nova fronteira criada e devemos clicar em Ok para confirmar.

Este mesmo procedimento deve ser seguido para o plano de terra e qualquer outra superfície metálica criada no projeto.

Para a caixa de ar que foi criada em *Create Region*, deve ser feito um procedimento similar, mas seguindo o caminho *Assign Boundary* → *Radiation...* para determinar que trata-se de uma região de radiação.

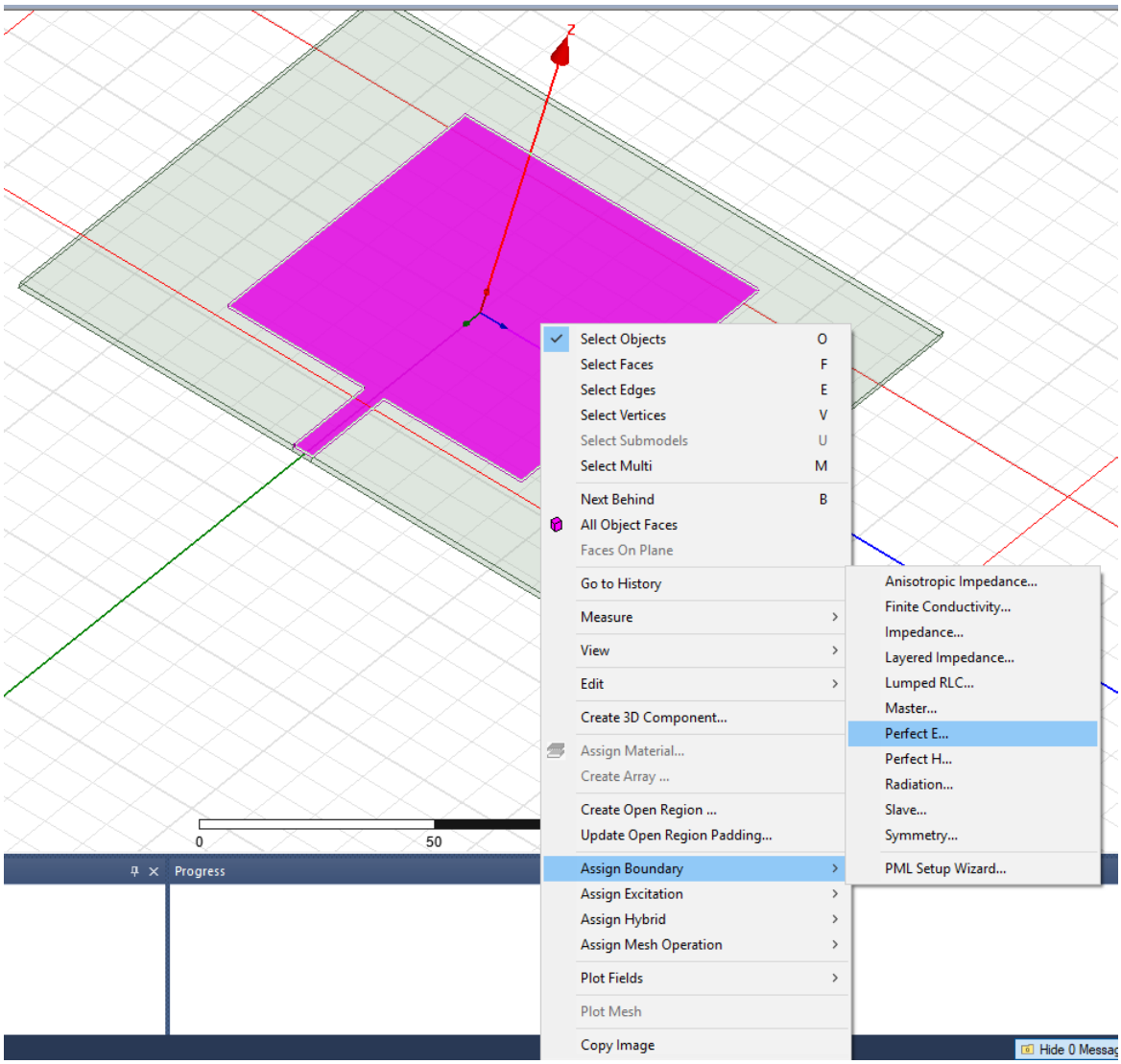


Figura 3.29 – Criação de *boundary*

Por fim, a alimentação deve ser criada como explicado na seção 2.2.3 e sua nova antena está pronta para ser simulada.

Setup de Simulação

Para realizar a simulação da antena é necessário criar um *Setup de Simulação*, para isto, no painel lateral esquerdo, clicamos com o botão direito do mouse em *Analysis* e em seguida "*Add Solution Setup...*" (Figura 3.30).

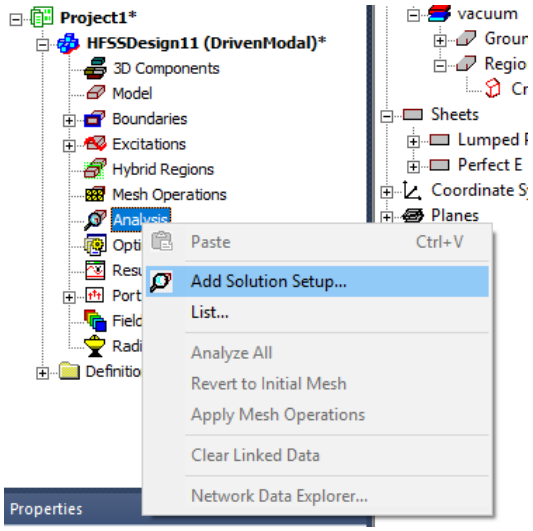


Figura 3.30 – Criação de *setup* de simulação

Ao criar o novo *setup* de simulação, uma nova janela irá abrir (Figura 3.31) para ser especificada a frequência de análise do *design*. Como aqui calculamos as dimensões para uma antena a 915MHz, colocamos este valor no termo *Solution Frequency*. Além disso, alteramos o *Maximum Number of Passes* para 20 para garantir a convergência no resultado simulado.

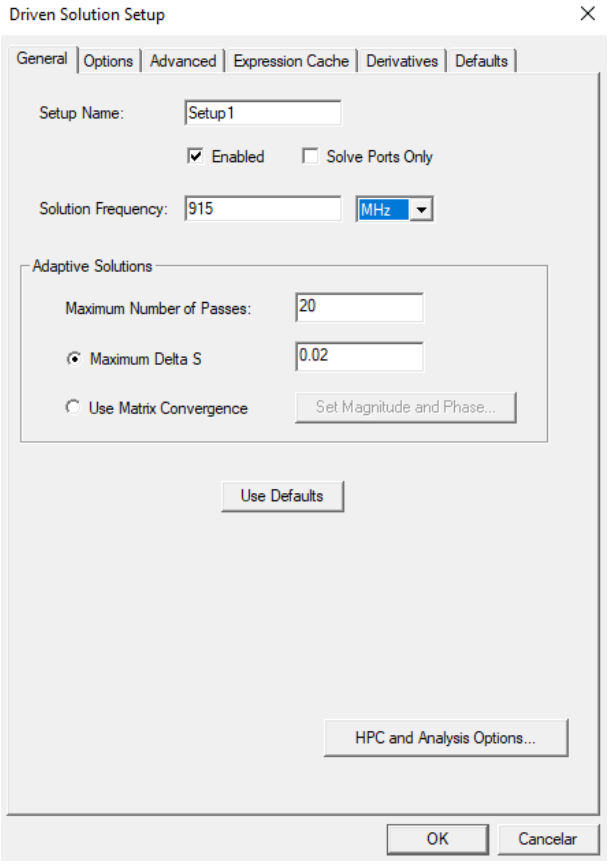


Figura 3.31 – Determinação da frequência de análise

Com a simulação criado, devemos agora indicar o espectro de frequência que será

analisado na simulação e, para isto, clicamos com o botão direito no novo *Setup* criado e selecionamos *Add Frequency Sweep* (Figura 3.32).

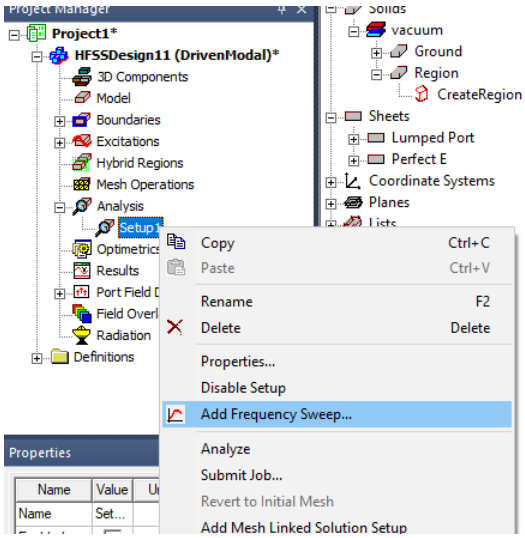


Figura 3.32 – Criação de *sweep*

Ao criar a varredura de frequência (*Frequency Sweep*), uma janela será aberta para que os parâmetros deste intervalo sejam determinados. Deve-se determinar o valor inicial, final e o passo de análise do projeto. No caso, como esperamos que a frequência de ressonância da antena seja em 915MHz, colocamos o intervalo entre 500MHz e 1500MHz com um passo de 1MHz, o que resultou em 1001 pontos, ou frequências, a serem analisados.

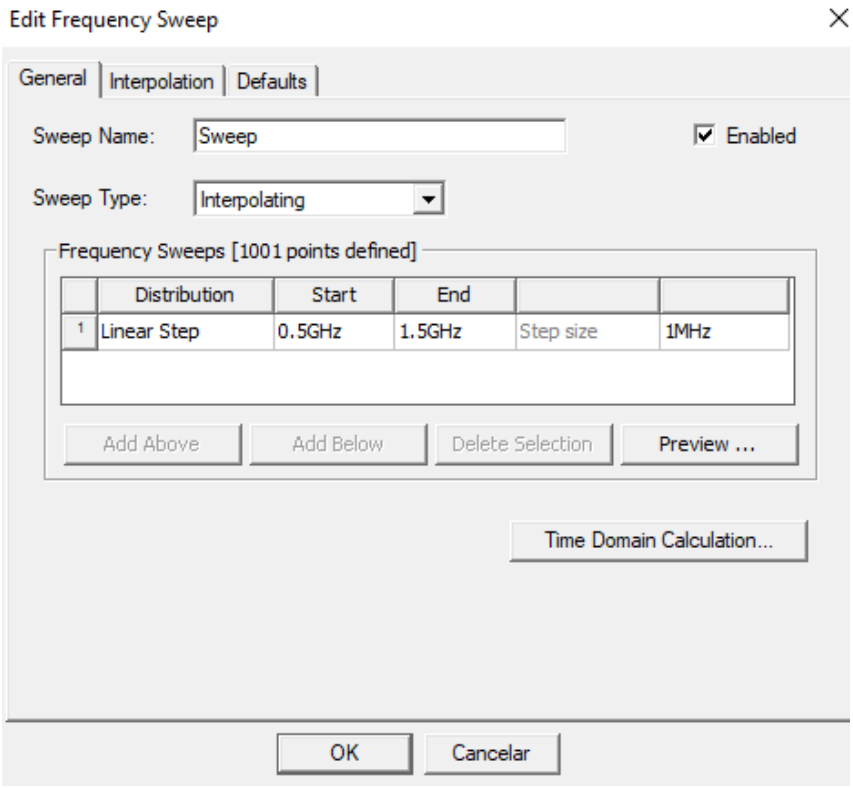


Figura 3.33 – Parâmetros do *sweep*

Finalmente, clicamos com o botão direito no novo *Sweep* criado e selecionamos *Analyze* para realizar a simulação.

Resultados

Neste tópico serão discutidos como visualizar os resultados obtidos após a realização das simulações e algumas interpretações que podem ser feitas a partir deles.

O *HFSS* consegue oferecer uma grande variedade de gráficos, entretanto, os mais utilizados no projeto de antenas são: **Diagrama de Radiação** e **Perda de Retorno (S11)**.

Para conseguirmos plotar o Diagrama de Radiação em 3D devemos primeiramente criar uma Radiação de Campos Distantes, para isso deve-se clicar com o botão direito em *Radiation* → *Insert Far Field Setup* → *Infinite Sphere...*, como na Figura 3.34.

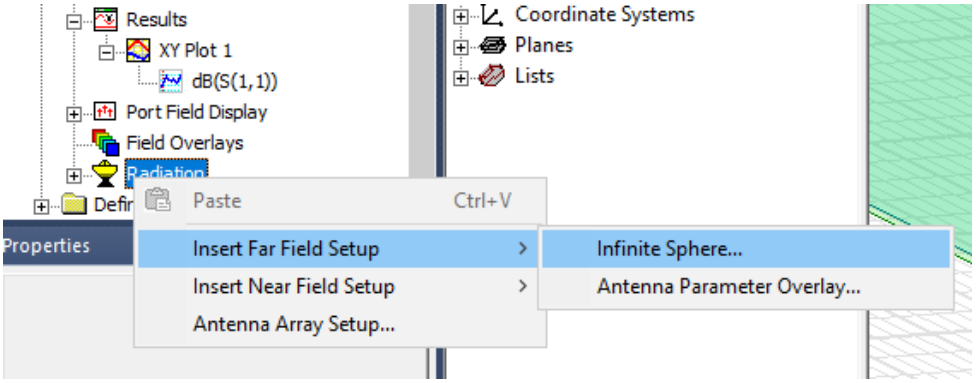


Figura 3.34 – Criação de um *Setup* de Radiação para Campos Distantes

Uma caixa de diálogo aparecerá, certifique-se que os parâmetros estão referidos na Figura 3.35 e aperte OK.

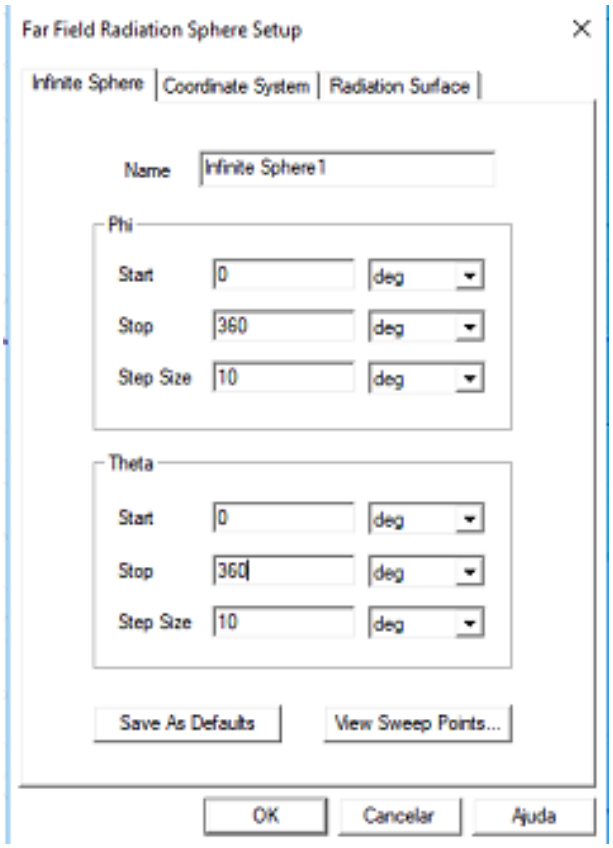


Figura 3.35 – Caixa de diálogo para determinação dos parâmetros da Radiação para campos distantes.

Em seguida, aperte com o botão direito em *Results* → *Create Far Fields Report* → *3D Polar Plot*, como mostrado da Figura 3.36.

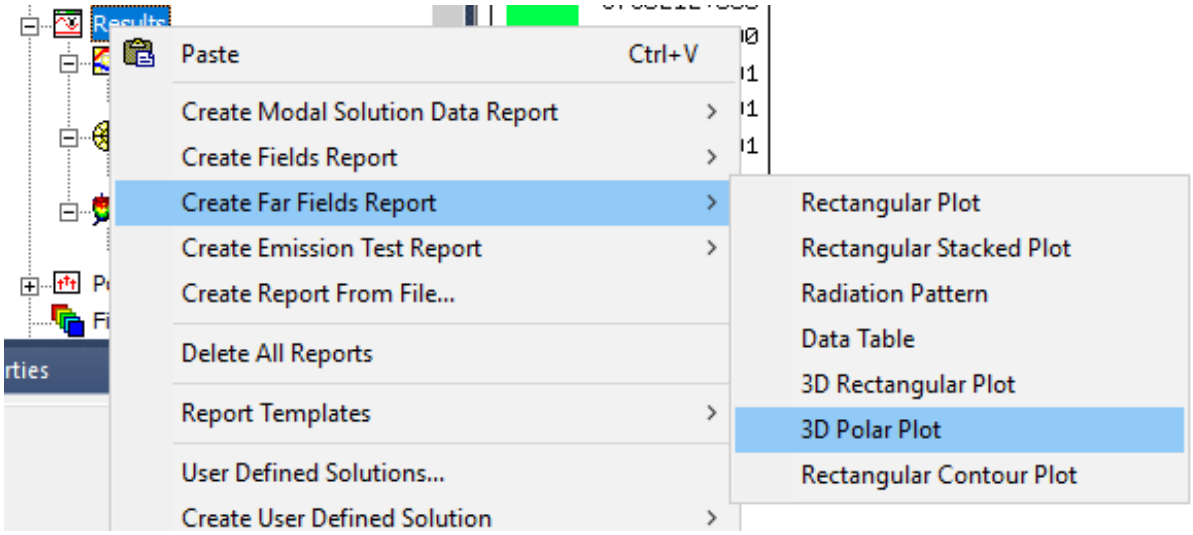


Figura 3.36 – Criação do Diagrama de Irradiação 3D.

Vamos alterar os parâmetros na caixa de diálogo que será aberta (Figura 3.37). Na aba **Category** devemos selecionar a opção **Gain**, na coluna **Functions** selecionamos a opção **dB**. Após isso, selecionamos a opção **New Report**. O diagrama de irradiação na Figura 3.38.

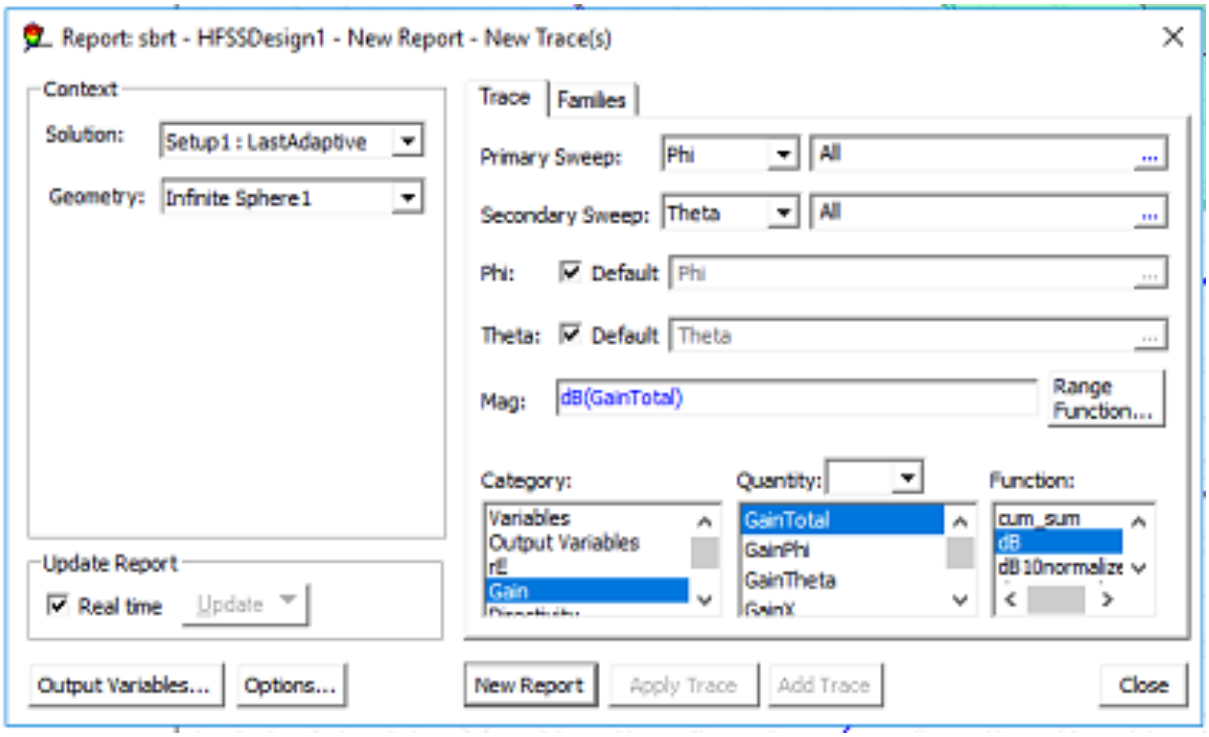


Figura 3.37 – Caixa de Diálogo com os parâmetros corretos para construção do Diagrama de Irradiação 3D.

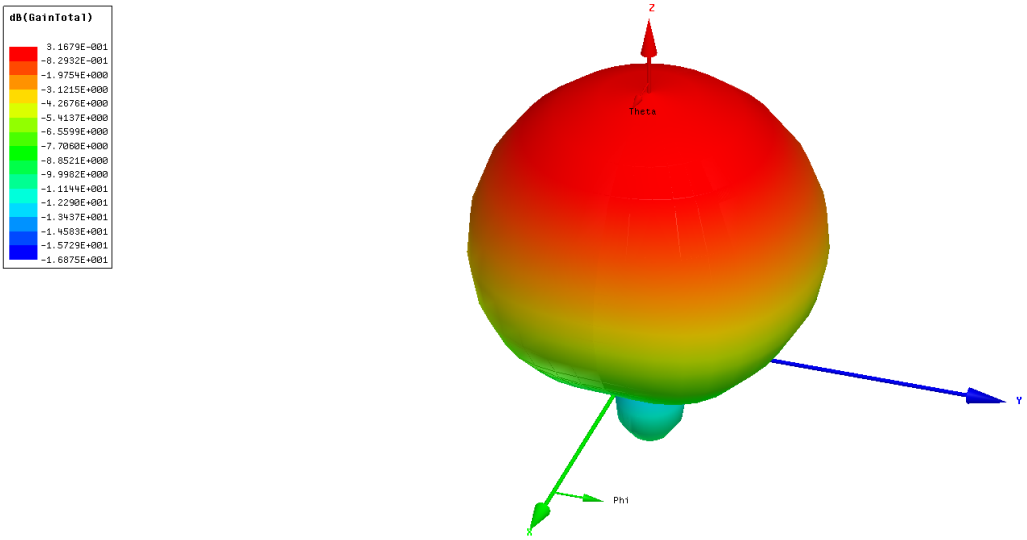


Figura 3.38 – Diagrama de irradiação da antena projetada.

Já para a perda de retorno (S_{11}), parâmetro bastante importante na análise dos resultados de uma antena, devemos apertar com o botão direito em *Results* → *Create Modal Solution Data Report* → *Rectangular Plot*.

Ao fazer isto, selecionamos *Category: S Parameter*, *Quantity: S(1,1)*, *Function: dB*. Para criar o gráfico, selecione *New Report*. A Figura 3.39 mostra o resultado obtido com os valores calculados teoricamente.

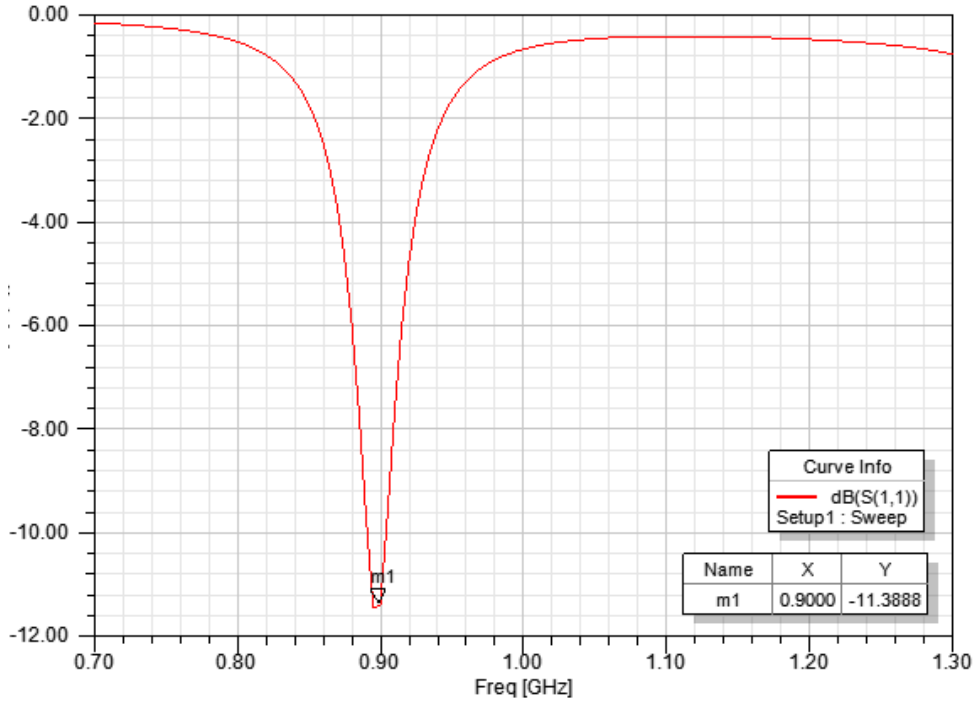


Figura 3.39 – Visualização do resultado de Perda de Retorno (S_{11}).

Interpretação de resultados

Como sabemos, o S_{11} é um parâmetro que mede a reflexão da onda incidente na antena na frequência desejada. Desta forma, é desejável que seu valor seja o menor possível (-∞) na frequência de trabalho, o que significa a potência de entrada totalmente (ou majoritariamente) transmitida (e não refletida). Como pode ser observado na Figura 3.39, com os valores calculados a partir das expressões teóricas a antena apresenta frequência de ressonância em 900MHz, diferente dos 915MHz esperado. O *design* deve ser, portanto, otimizado de como que a frequência desejada seja obtida.

Na Figura 3.38 temos o valor do ganho da antena que, como pode ser observado, é bastante baixo. Isto se deve ao fato de que, como a antena ainda não apresenta bom desempenho na frequência de trabalho, o ganho esperado é baixo, visto que este é calculado na frequência específica de trabalho. Posteriormente será mostrado que ao melhorar a Perda de Retorno (S_{11}) o ganho é, consequentemente, melhorado também.

Parametrização

A utilização de variáveis nas dimensões da antena criada nos permite utilizar uma ferramenta extremamente útil na otimização desta. O *Opmetrics* é uma ferramenta do HFSS que permite definir um intervalo de varredura para as variáveis criadas, no qual o *software* deve analisar a estrutura no objetivo de determinar os melhores valores para estas variáveis que otimizem o desempenho da antena.

Para adicionar uma nova parametrização, clicamos com o botão direito sobre *Opmetrics* → *Add* → *Parametric* como ilustra a Figura 3.40.

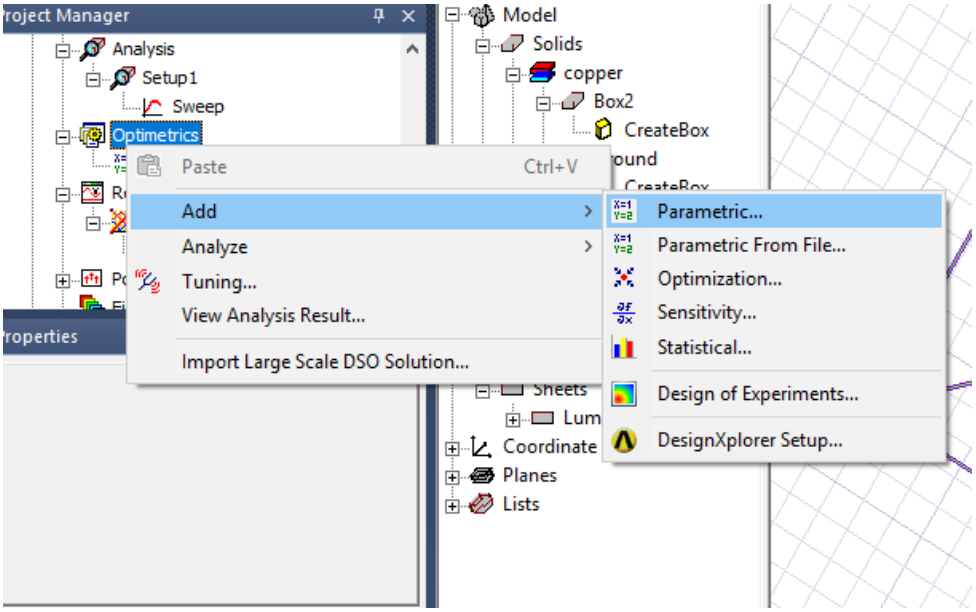


Figura 3.40 – Adicionar uma nova parametrização.

Na janela que será aberta, na aba *Sweep Definitions*, clicamos em *Add* para adicionar variáveis à parametrização (Figura 3.41).

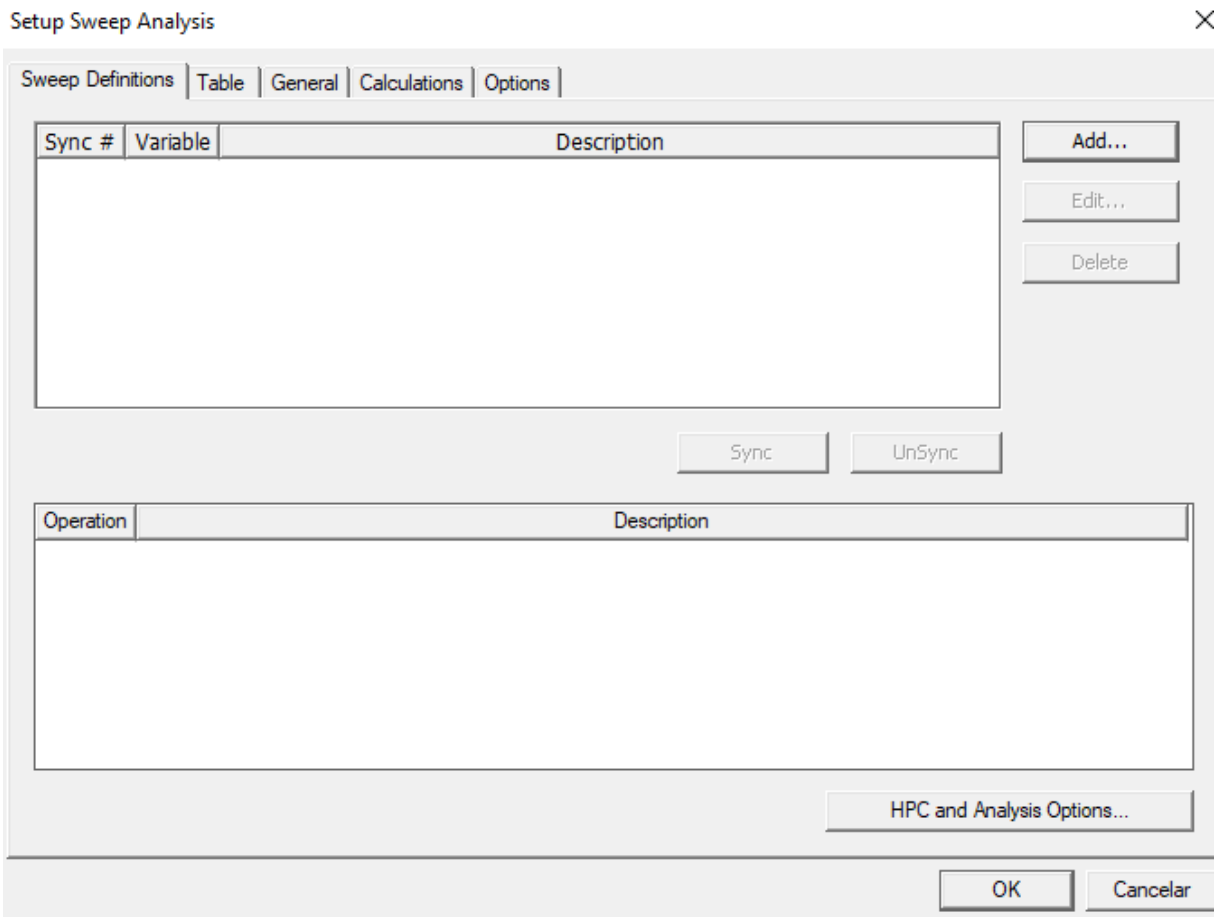


Figura 3.41 – Adicionar variáveis à parametrização.

Uma nova janela será aberta (Figura 3.42), onde em *Variable* podemos selecionar cada uma das variáveis que queremos adicionar à parametrização, bem como adicionar o valor de início (*Start*) e fim (*Stop*) da varredura e o *Step* desejado. Selecionamos, neste caso, apenas a variável *patchy* para a parametrização com um *Linear Step* de 1mm entre 75mm e 79mm.

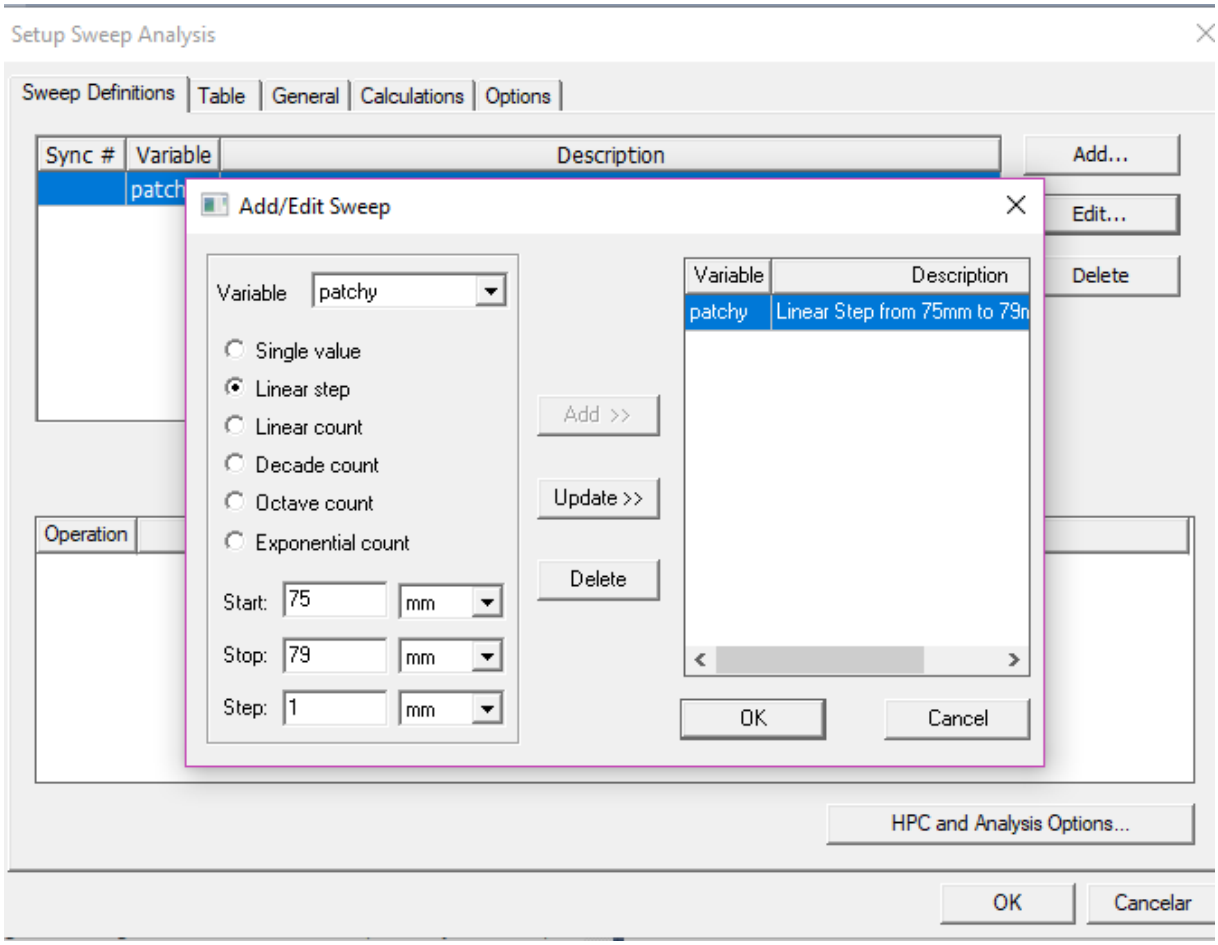


Figura 3.42 – Definir o intervalo de varredura da parametrização.

Definidas todas as variáveis, clicamos mais uma vez com o botão direito sobre *Opmetrics* → *Analyze* → *All*. Depois de concluída a simulação, plotamos todos os resultados seguindo o mesmo procedimento mostrado anteriormente, com a diferença de que na aba *Families* determinamos que sejam plotados todos os valores (*Edit* → *Use all values*) para cada uma das variáveis, como ilustra a Figura 3.43.

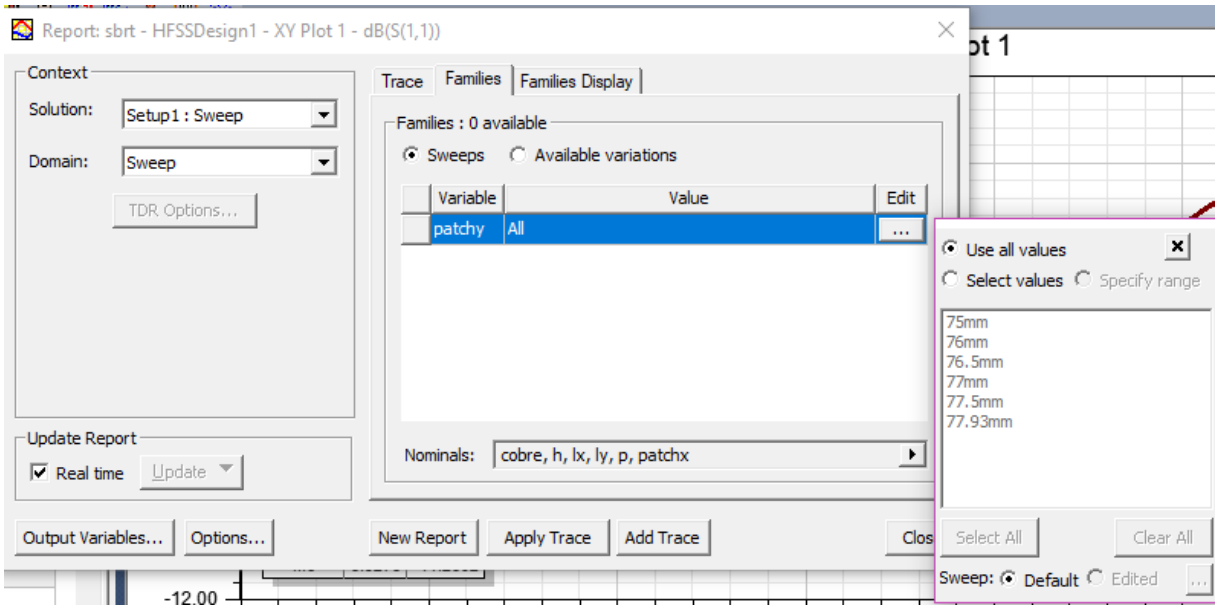


Figura 3.43 – Plotagem dos resultados da parametrização.

Com os resultados parametrizados para a Perda de Retorno S_{11} da antena, ilustrados na Figura 3.44 verificamos que o melhor resultado, considerando a frequência de ressonância desejada de 915 MHz, ocorre para um valor de $patchy = 76.5$ mm. Com esse valor, obtemos um valor de -11.90 dB na frequência de 912 MHz.

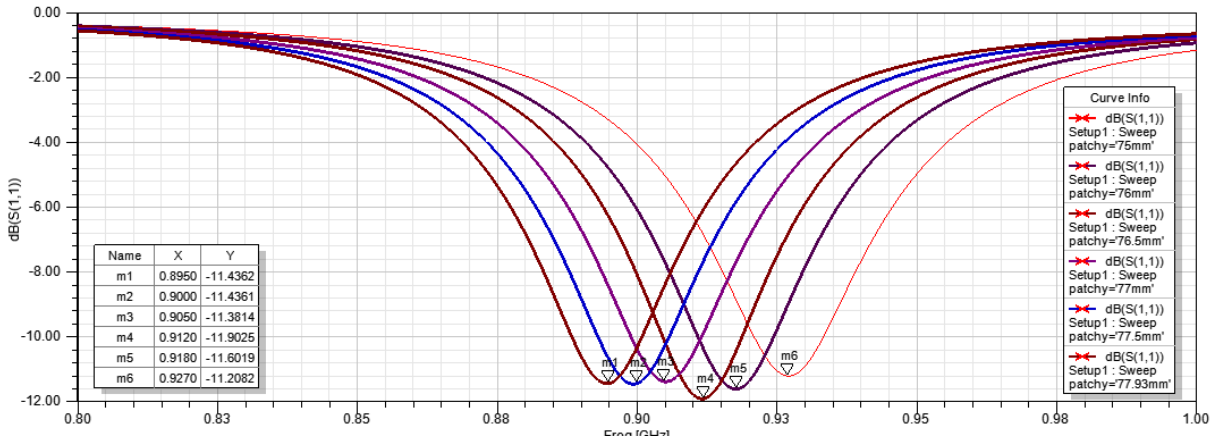


Figura 3.44 – Visualização dos resultados parametrizados da Perda de Retorno (S_{11}).

Plotamos novamente o diagrama de irradiação da antena, considerando o valor encontrado para a variável *patchy* e, pela Figura 3.45, é possível observar uma considerável melhora no valor do ganho da antena, que agora encontra-se em 2.50 dB.

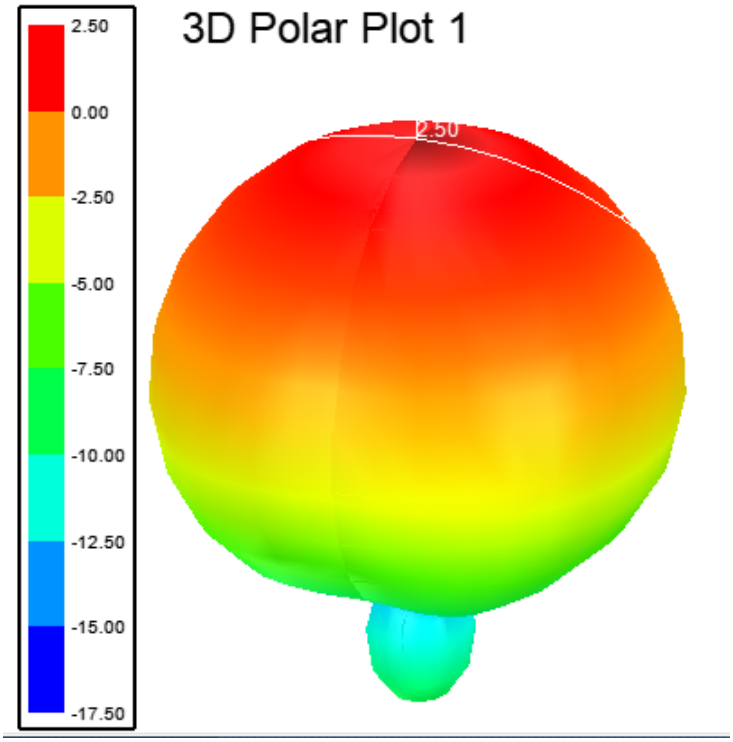


Figura 3.45 – Diagrama de irradiação da antena após a otimização.

CAPÍTULO

4 Monitoramento Lógico e Físico do Tráfego em Redes de Internet das Coisas

Syllas Rangel C. Magalhães(UFC), Victória Tomé Oliveira(UFC), Francisco Evangelista N. Filho(UFC), Magdiel Campelo Alves de Sousa(UFC), Jermana Lopes de Moraes(UFC), Wendley S. Silva

Introdução

Nos últimos anos, houve um crescimento no número de dispositivos sem fio utilizados pela população, e as previsões para o futuro é que essa quantidade aumente significativamente; entre os anos de 2016 e 2021, as projeções indicam que ocorrerá um crescimento de 7,3 vezes na quantidade de dispositivos *inteligentes* conectados [1].

Conceituar Internet das Coisas (IoT, do inglês *Internet of Things*) não é uma tarefa fácil. O termo foi utilizado pela primeira vez pelo britânico Kevin Ashton e seus colegas de laboratório, em um trabalho desenvolvido na Procter & Gamble [2]. Contudo, IoT nada mais é que a extensão da internet atual, que pode ser definida como uma infraestrutura de rede que liga objetos físicos e virtuais através da captura de dados e comunicação a uma plataforma que possibilita a execução de uma aplicação [2]. Contudo, IoT também pode ser compreendida como uma infraestrutura de rede que liga objetos físicos e virtuais através da captura de dados e comunicação a uma plataforma que possibilita a execução de uma aplicação [3]. Essa infraestrutura de comunicação inclui a Internet, outras redes existentes e aquelas ainda em desenvolvimento.

Segundo o IEEE [4] (Instituto de Engenheiros Eletricistas e Eletrônicos, do inglês *Institute of Electrical and Electronics Engineers*) descreve a expressão *Internet of Things* como "Uma rede de itens, cada um embarcado com sensores, que estão conectados à internet". Umas das mais completas definições é a da Cisco [5], que utilizou outro conceito, a Internet de Tudo (*Internet of Everything* - IoE), para explicar melhor a Internet das Coisas. A Internet de Tudo é a conexão em rede de pessoas, dados, processos e coisas. A IoE é composta de

muitas transições de tecnologia, incluindo a Internet das Coisas. E a Internet das Coisas é definida como a conexão em rede de objetos físicos. IoT é uma das muitas transições tecnológicas que permitem a IoE.

A Internet das Coisas (IoT) refere-se à interconexão desses objetos através da rede mundial de computadores, a internet. A IoT pode ser vista como a combinação de diversas tecnologias, as quais são complementares no sentido de viabilizar a integração dos objetos no ambiente físico ao mundo virtual. IoT pode conectar dispositivos onipresentes e instalações com várias redes para fornecer serviços eficientes e seguros para todas as aplicações a qualquer hora e em qualquer lugar, a interoperabilidade entre essas redes IoT é de fundamental importância para a entrega de informações, e é necessário ter em mente também que não há apenas objetos conectados, e sim, também, informações, comportamentos humanos etc [6].

Neste trabalho, serão apresentadas as definições e motivações para as pesquisas de *Internet of Things*. Também serão discutidos os elementos que compõem a IoT, suas principais plataformas e os seus principais protocolos de comunicação. Todo esse embasamento é necessário pois serão apresentadas aplicações voltadas para a área de IoT, a parte prática do minicurso ficará na Seção *Análise de Tráfego*, onde serão apresentadas as técnicas de monitoramento de tráfego em redes IoT, bem como será detalhada a utilização de *softwares* para o monitoramento.

O minicurso aqui descrito consiste em uma extensão do minicurso apresentado no ENUCOMP 2017 [7]. As principais diferenças são que este minicurso: (i) trata do monitoramento físico e lógico das redes de IoT; (ii) apresenta novos cenários (mais próximos das redes de IoT executáveis na prática); e (iii) incorpora a análise de novos protocolos de comunicação.

Elementos da IoT

Um ambiente de IoT pode ser fragmentado em algumas tecnologias complementares entre si, ou seja, viabilizam a integração dos objetos do ambiente físico com o virtual. Essas tecnologias se dividem em seis blocos ou elementos (*identification*, *sensing*, *communication*, *computation*, *services* e *semantics* [8]), como pode ser visto na Figura 4.1. Compreender cada elemento é uma tarefa importante, pois contribui para a definição e entendimento funcional da *Internet of Things* – sendo útil, até mesmo, para identificação de falhas.

Figura 4.1 – Elementos de *Internet of Things* [8].

Importante ressaltar que alguns autores preferem condensar os seis blocos em três: textithardware, *middleware* e *presentation* [3]. Todos os ambientes ou redes de IoT são caracterizados por esses elementos, que serão detalhados como segue.

Identification

Identification ou no português, Identificação, é o bloco responsável por atribuir, de acordo com a demanda, identificadores aos objetos e serviços ofertados pelo ambiente IoT – simplificando o gerenciamento das diversas conexões existentes e contribuindo para uma identificação exclusiva (utilização única) dos objetos. É a partir deste processo que é possível identificar quais solicitações ocasionaram certos serviços. Cabe ressaltar que o processo *Identification* permite diferenciar um IP ("nome" de um objeto) de seu endereço ("nome" ou valor atribuído dentro do ambiente ou rede IoT), facilitando a identificação e solução de problemas, como sobrecarga e perda de solicitações (erros comuns quando são utilizadas, sobretudo, redes públicas [8]). São protocolos comumente utilizados nesta etapa: IPv4, IPv6 e 6LoWPAN (um dos mais difundidos, pois foi especialmente projetado para redes sem fio de baixa energia) [9].

Sensing

Sensing ou, em português, Detecção, é o elemento que faz uso de sensores e atuadores para coletar dados dos objetos presentes em um ambiente IoT. Em seguida, os dados são enviados para um *data warehouse* (sistema de armazenamento de dados digitais), banco de dados ou nuvem (*cloud services*) para que sejam armazenados. Em seguida, ocorre o processamento ou análise dos dados para que as ações sejam executadas. Com os avanços em pesquisas ligadas à *Internet of Things*, muitas empresas estão investindo em tecnologias capazes de realizar as tarefas do bloco *Sensing*, como a WeMo e ZigBee.

Os *Single Board Computers* (SBCs) são utilizados nesta etapa para realizar o gerenciamento dos sensores e atuadores. Normalmente, os diversos SBCs que compõem um ambiente IoT se conectam com um *hub*, que irá realizar a coleta dos dados [8].

Communication

Sabe-se que a *Internet of Things* faz uso de diversos componentes. Logo, gerenciar a comunicação entre eles não deve ser uma tarefa simples, sobretudo pelos ambientes IoT operarem, preferencialmente, em baixas frequências ou ambientes com muitos ruídos e perdas [8].

Na etapa *Communication* ou, em português, Comunicação, são realizados estudos de protocolos e tecnologias capazes de fornecer serviços inteligentes eficientes e com perdas minimizadas. Entre as tecnologias populares, podem ser citadas: *WiFi*, *Bluetooh*, IEEE 802.15.4, *Z-wave*, NFC (*Near Field Communication*), UWB (*Ultra-wide bandwish*), RFID (*Radio-frequency identification*) e *LTE-advanced*. Algumas dessas, e outras, serão detalhadas na Seção 4.

Computation

O bloco *Computation* ou Computação, em português, é definido como o "cérebro físico"de um ambiente IoT e é responsável por realizar o processo descrito na Seção 4. A criação dos Sistemas Operacionais em Tempo Real (RTOS) e a evolução dos serviços em nuvem permitiram um processamento mais refinado (em tempo real) de um volume considerável de dados com um custo de memória lógica menor, cenário perfeito para a *Internet of Things* [8]. Esses dados permitem que os usuários aperfeiçoem suas redes IoT.

Com os avanços tecnológicos, surgem anualmente diversos *hardwares* e *softwares* cada vez mais robustos. Durante o projeto de uma rede IoT, esses são escolhidos de forma a respeitar as necessidades da aplicação (Indústria, *Smart Homes*, *Healthcare*, entre outras). Atualmente, entre os *hardwares* mais difundidos em ambientes IoT encontram-se Arduino, *Raspberry Pi®*, *BeagleBone* e Galileo Gen 2; em se tratando de *software* priorizam-se aplicações desenvolvidas em linguagens orientadas à objetos (Java, Python, Ruby, C#, C++) ou de baixo nível (C).

Services

Services ou, em português, Serviços, define e discrimina, literalmente, os serviços e funcionalidades aprovisionados por um ambiente ou uma rede IoT. No geral, eles se dividem em quatro classes: *identity-related services*, *information aggregation services*, *collaborative-aware services* e *ubiquitous services* [8]. De forma mais objetiva, o bloco *Services* é utilizado para esmiuçar, por exemplo, *Smart Grids*, *Smart Cities* e *Healthcare* em termos de suas funções. Explorar as classes previamente citadas foge do escopo deste trabalho.

Semantics

De forma análoga à definição utilizada na seção anterior, o bloco *Semantics* ou Semântica, no português, pode ser definido como as sinapses – contato entre os neurônios para que ocorra a transmissão de impulsos elétricos/informações – do “cérebro físico”. Nesta etapa são extraídas informações/conhecimento de um ambiente IoT para prover os serviços necessários. Esse processo é realizado em quatro etapas, que podem ser definidas intuitivamente: descoberta, utilização, análise e modelagem dos recursos fornecidos [7].

Em outras palavras, *Semantics* funciona como uma espécie de Inteligência Artificial (AI) utilizada na automatização de um processo.

Ademais, uma síntese das tecnologias citadas anteriormente pode ser vista na Tabela 4.1.

Plataformas de IoT

As principais plataformas de aprendizagem para a Internet das coisas são o Raspberry Pi, o Arduino e o NodeMCU. Cada um possui suas particularidades e características para

Elementos de *Internet of Things*		Exemplos
Identification	Nomear	EPC, *uCode*, DOI, etc.
	Endereçar	IPv4, IPv6, 6LoWPAN, etc.
Sensing		Sensores, atuadores, dispositivos vestíveis, embarcados, etc.
Communication		*WiFi*, *Bluetooh*, IEEE 802.15.4, *Z-wave*, *LTE-advanced*, UWD, RFID, etc.
Computation	*Hardware*	Arduino, *RaspBerry Pi*®, *BeagleBone*, Galileo Gen 2, *Smartphones*, *Smarthings*, etc.
	Software	Contiki RTOS, TinyOS, LiteOS, RiotOS, OAA, Nimbits, etc.
Services		*Smart grids*, *Smart homes*, *Smart city*, etc.
Semantics		RDF, EXI, OWL, etc.

Tabela 4.1 – Síntese dos elementos de IoT [8].

a utilização nos cenários IoT. Os módulos de *hardware* para implementar IoT já estão disponíveis a um custo factível para aplicação. Um microcontrolador, em particular, NodeMCU, virou uma opção extremamente interessante devido a seu custo muito reduzido e aos seus recursos, suficientes para diversas aplicações de IoT, sendo o Arduino uma alternativa viável para substituir o NodeMCU.

Os experimentos práticos foram desenvolvidos com o auxílio do *software* Wireshark e do uso de plataformas de baixo custo, como Raspberry Pi, Arduino e NodeMCU, bastante utilizadas no contexto de IoT. A seguir, cada uma dessas plataformas é analisada mais detalhadamente.

Wireshark

O Wireshark[1] é um *software* livre, multiplataforma, que permite a captura de pacotes (*sniffers*) e análise detalhada do tráfego de Redes de Computadores, incluindo os protocolos. Através dessa aplicação é possível controlar o tráfego de uma rede e monitora a entrada e saída de dados do computador, em diferentes protocolos, ou da rede à qual o computador está ligado.

Raspberry Pi

O Raspeberry Pi[2] é um microcomputador *single-board* criado originalmente para fins educativos. Entretanto, devido ao pequeno tamanho, preço acessível e grande poder de processamento desses dispositivos, sua utilização em aplicações mais complexas foi rapidamente adotada. Hoje, existem diversos modelos desse microcomputador, que vão desde os mais simples, voltados para aplicações que exigem menos poder computacional, aos mais completos, como o Raspberry Pi 3, que possui memória RAM e velocidade de *clock* na casa dos gigas.

[1] https://www.wireshark.org
[2] https://www.raspberrypi.org/

Arduino

O Arduino[3] é uma plataforma de prototipagem eletrônica de baixo custo com *hardware* livre que, assim como o Raspberry Pi, foi pensada inicialmente para fins educacionais e, posteriormente, passou a ser utilizada em projetos mais robustos. Existem também vários modelos de Arduino, sendo um dos mais comuns o Arduino Uno, com 14 pinos de entrada e saída digitais e 6 analógicas. Devido ao seu tamanho e baixíssimo custo, essa plataforma tem grande potencial para compor as redes de IoT.

NodeMCU

O termo NodeMCU[4] se refere originalmente a uma *firmware open source* que, posteriormente, se expandiu para incluir uma plataforma *open hardware* de baixo custo, ambas projetadas para a Internet das Coisas. O *hardware* é baseado no microcontrolador ESP-8266 que, além dos pinos de entrada e saída analógicos e digitais, assim como no Arduino, oferece conectividade WiFI. Hoje, o NodeMCU já inclui diversos módulos e permite fácil programação por meio do Arduino IDE.

Protocolos de IoT

Nos últimos anos, houve um crescimento no número de dispositivos sem fio utilizados pela população, e as previsões para o futuro é que essa quantidade aumente significativamente [1]. Esse fenômeno ocorre, sobretudo pela heterogeneidade dos dispositivos utilizados em ambientes IoT. Devido a isso, torna-se essencial o estudo e desenvolvimento de protocolos que possam, por exemplo, suprir limitações impostas pelos dispositivos, como ocorre em muitos casos.

Muitos padrões de IoT são propostos para facilitar e simplificar trabalhos de programadores de aplicativos e provedores de serviços. Diferentes grupos foram criados para fornecer protocolos de apoio ao IoT incluindo esforços liderados pelo *World Wide Web Consortium* (W3C), Força-Tarefa de Engenharia da Internet (IETF), EPCglobal, Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) e Instituto Europeu de Normas de Telecomunicações (ETSI) [8].

Como podemos observar na Tabela 4.2, neste minicurso, classificamos os protocolos IoT em três amplas categorias, a saber: protocolos de aplicação, protocolos de descoberta de serviço e protocolos de infraestrutura. Nas subseções a seguir, será apresentada uma visão geral de alguns destes protocolos, citando suas principais características e funcionalidades, de acordo com a classificação proposta. Dar-se-á uma atenção especial aos protocolos CoAP e MQTT, ambos da camada de aplicação.

[3] https://www.arduino.cc/
[4] http://www.nodemcu.com/

Protocolos de aplicação		AMQP, CoAP, DDS, HTTP, REST, MQTT, XMPP
Protocolos de descoberta de serviços		mDNS, DNS-SD
Protocolos de infraestrutura	Roteamento	RPL
	Camada de Rede	6LoWPAN, IPv4/IPv6
	Camada de Enlace	IEEE 802.15.4
	Camada Física/Dispositivo	LTE-A, EPCglobal, IEEE 802.15.4, Z-Wave

Tabela 4.2 – Protocolos relacionados à IoT [8].

Protocolos de aplicação

O protocolo HTTP é utilizado na Internet para prover o acesso à informação desde 1990, constituindo a base para a comunicação de dados na *World Wide Web* (WWW). O HTTP foi projetado para redes com computadores pessoais, com maior poder de processamento se comparado aos dispositivos utilizados em redes IoT. As restrições impostas pelos equipamentos utilizados em ambientes IoT limitam o uso do protocolo HTTP. Sendo assim, foram desenvolvidos outros protocolos na camada de aplicação como alternativa ao HTTP, como, por exemplo, o CoAP e o MQTT, ambos projetados para a troca de informação entre dispositivos com baixo poder computacional. Além destes, outros protocolos vêm sendo utilizados em redes IoT, como é caso do XMPP, AMQP e DDS. A seguir, alguns desses protocolos serão detalhados.

Constrained Application Protocol (CoAP)

O grupo de trabalho IETF *Constrained RESTful Environments (CoRE)* criou o CoAP, que é um protocolo da camada de aplicação para soluções IoT [10], com o objetivo de otimizar o uso da arquitetura REST (*REpresentational State Transfer*) em nós - por exemplo, microcontroladores de 8 *bits* com RAM e ROM limitadas - e redes (e.g. *6LoWPAN*) com poder computacional restrito [11, 12]. A forma de interação do CoAP é semelhante ao modelo cliente/servidor no protocolo HTTP, entretanto, ele também oferece recursos para M2M (*Machine-to-Machine*), como descoberta interna, suporte *multicast* e trocas de mensagens assíncronas. Semelhante ao HTTP, uma solicitação CoAP é enviada por um cliente para solicitar uma ação usando *Uniform Resource Identifiers* (URIs), que em seguida é respondido pelo servidor com um código de resposta [13]. Em contraposição ao REST (que utiliza HTTP sobre TCP), o CoAP utiliza o UDP por padrão, o que o torna mais adequado para as aplicações IoT. Além disto, o CoAP altera algumas funcionalidades HTTP para atender aos requisitos de IoT, como operação na presença de links com perdas e ruídos e baixo consumo de energia. Entretanto, como o CoAP foi projetado com base em REST existe uma fácil interoperabilidade entre os protocolos HTTP e CoAP.

O CoAP pode representar sua arquitetura em duas camadas: a primeira sendo responsável pela implementação dos mecanismos de requisição/resposta e a segunda sendo responsável pela comunicação e, opcionalmente, confiabilidade sobre UDP. Ele possui,

0 1	2 3	4 7	8 15	16 31
Ver		TKL	Código	ID da Mensagem
Token (caso exista, com TKL bytes) ...				
Opções (caso existam) ...				
1 1 1 1 1 1 1 1			*Payload* (caso exista) ...	

Tabela 4.3 – Formato da mensagem CoAP [12].

ainda, quatro tipos de mensagens: *confirmable*, *non-confirmable*, *reset* e *acknowledgement*. Uma mensagem do tipo *confirmable* (CON) requer uma resposta do receptor com uma confirmação de recebimento. Essa é contrária à mensagem *non-confirmable* (NON), que não garante o recebimento no receptor, uma vez que este tipo de mensagem, como o próprio nome sugere, não exige confirmação. A mensagem de confirmação propriamente dita é chamada de *acknowledgement* (ACK). Ela é transmitida em resposta a uma mensagem *confirmable* recebida de forma correta. Por fim, a tipo *reset* (RST) é enviada basicamente em três situações: quando ocorre erro na mensagem; quando a mensagem não é compreensível; quando o receptor não está interessado na comunicação com o remetente.

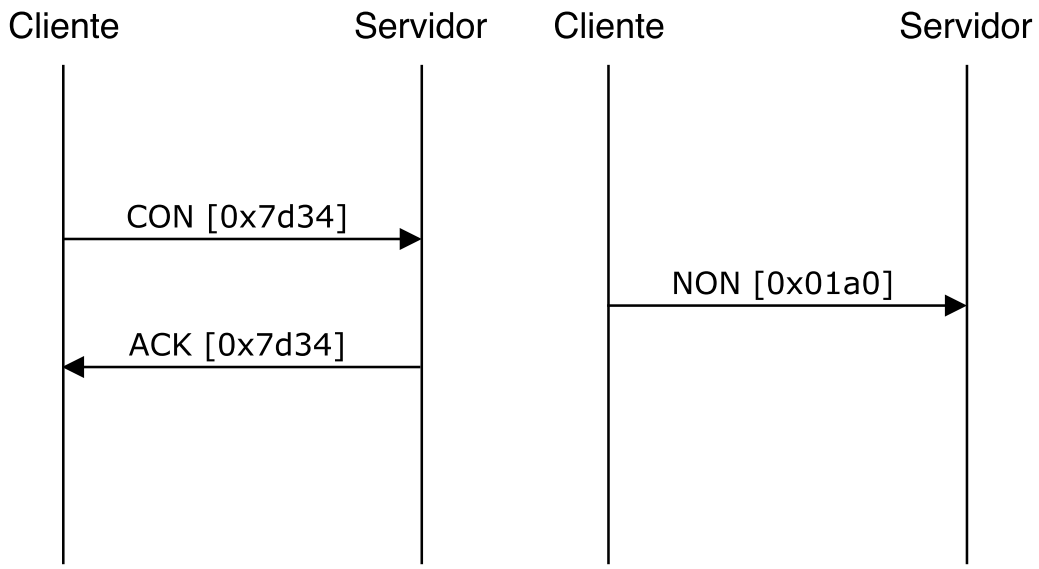
As mensagens CoAP são codificadas em um formato binário simples, como visto na Tabela 4.3. A primeira parte da mensagem é um cabeçalho fixo de 4 bytes, que pode ser seguido por um *token*, algumas opções e um *payload*, como ilustrado na Tabela 4.3. O cabeçalho fixo é dividido em cinco partes: Ver (versão do protocolo), T (tipo de mensagem), TKL (tamanho do campo *Token*), Código e ID da mensagem. Os três bits mais significativos do campo código carregam a informação de qual classe a mensagem pertence, podendo indicar uma requisição (0), uma resposta bem-sucedida (2), uma resposta de erro do cliente (4) ou uma resposta de erro do servidor (5). Já os cinco bits menos significativos desse campo, representam uma subclasse de mensagem. Por exemplo, para uma mensagem de requisição que utiliza o método GET, o campo código seria representado por “0.01”, em que o primeiro dígito indica a classe (nesse caso requisição) e os dois dígitos depois do ponto representam a subclasse (nesse caso GET) [12]. Na tabela 4.4 estão representados todos os códigos de resposta CoAP.

Na Figura 4.2 são exibidas duas situações de uma comunicação CoAP. Na esquerda (Figura 4.2a) é mostrado um exemplo de uma transmissão de mensagem com confirmação (ACK). Neste caso, o transmissor continua retransmitindo a mensagem de acordo com um *timeout* padrão até que receba uma mensagem ACK com o mesmo ID da mensagem enviada (neste caso 0x7d34). Já na Figura 4.2b é mostrado um exemplo simples de uma transmissão não confiável (sem confirmação), em que o transmissor simplesmente envia uma mensagem e não requer nenhum tipo de confirmação. Em ambas as situações, quando o destinatário não é capaz de processar a mensagem, ou seja, nem mesmo sendo capaz de fornecer uma resposta de erro adequada, ele responde com uma mensagem *reset* (RST).

Existem quatro métodos de requisição CoAP que foram definidos em [10], sendo eles *GET* (código 0.01), *POST* (código 0.02), *PUT* (código 0.03) e *DELETE* (código 0.04). As requisições podem ser realizadas em mensagens do tipo *confirmable*

Código	Nome	Descrição
2.01	Created	Resposta a uma requisição PUT ou POST (criado)
2.02	Deleted	Resposta a uma requisição que torna um recurso indisponível
2.03	Valid	Indica que a resposta identificada pela *entity-tag* é válida
2.04	Changed	Resposta a uma requisição PUT ou POST (modificado)
2.05	Content	Indica a presença do conteúdo requisitado em um GET
4.00	Bad Request	Indica que o servidor não “entendeu” a requisição
4.01	Unauthorized	O cliente não está autorizado a executar a ação solicitada
4.02	Bad Option	O servidor não reconheceu uma ou mais opção crítica
4.03	Forbidden	Acesso ao recurso é proibido
4.04	Not Found	Não encontrado
4.05	Method Not Allowed	Método não permitido
4.06	Not Acceptable	O recurso de destino não possui uma representação aceitável
4.12	Precondition Failed	Precondição falhou
4.13	Request Entity Too Large	Entidade de requisição muito grande
4.15	Unsupported Content-Format	*Context-format* não suportado
5.00	Internal Server Error	Erro interno no servido
5.01	Not Implemented	Não implementado
5.02	Bad Gateway	Erro quando o servidor atuava como *gateway* ou *proxy*
5.03	Service Unavailable	Serviço indisponível
5.04	Gateway Timeout	*Timeout* quando o servidor atuava como *gateway* ou *proxy*
5.05	Proxying Not Supported	*Proxying* não suportado

Tabela 4.4 – Códigos de resposta CoAP [12].

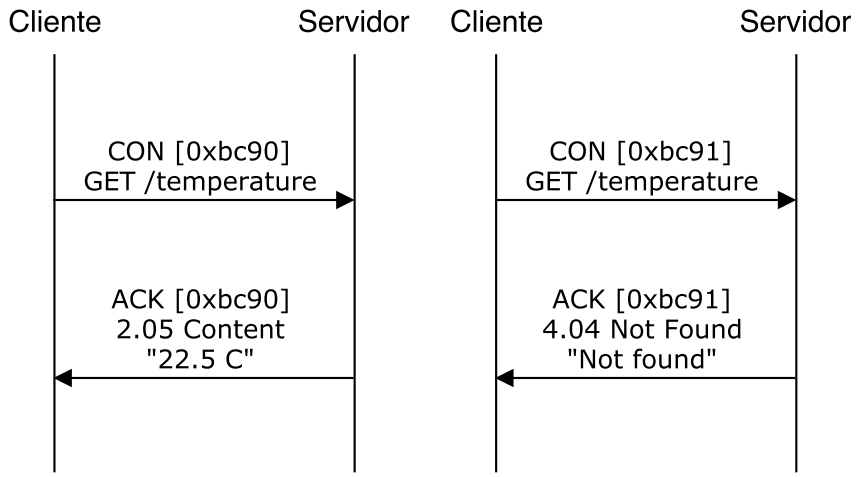


(a) Com confirmação **(b)** Sem confirmação

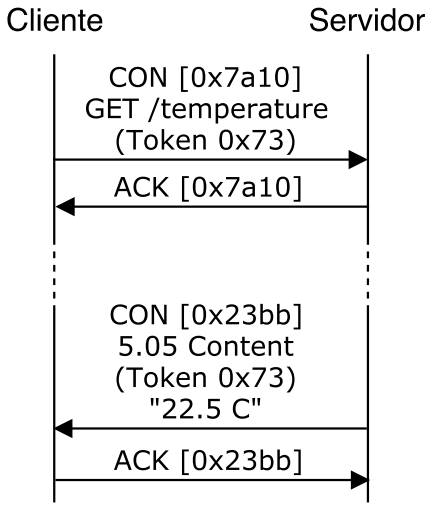
Figura 4.2 – Comunicação CoAP com e sem confirmação de recebimento [12].

(CON) ou *non-confirmable* (NON). No caso da resposta a uma requisição CoAP do tipo *confirmable* estar disponível imediatamente, esta é enviada junto à mensagem de confirmação (ACK), ocasião em que é chamada de resposta *piggy-backed*. Na Figura 4.3a são ilustradas duas situações onde ocorre uma resposta *piggy-backed*. Na primeira, a requisição obteve uma resposta satisfatória e no segundo caso uma resposta de 4.04 (*Not found*). Por outro lado, a Figura 4.3b mostra uma requisição GET do tipo *confirmable* com respostas separadas. Neste caso, o servidor não estava disponível para responder imediatamente a

requisição. Dessa forma, para que o cliente fique ciente de que a requisição foi recebida corretamente e pare de retransmitir a mensagem, é enviada uma confirmação (ACK) vazia. Após algum tempo, quando a resposta estiver disponível, o servidor a envia com solicitação de confirmação. Cabe ressaltar que, neste caso, a distinção entre as respostas esperadas pelo cliente é feita a partir do *Token* (não confundir com ID da mensagem) que, apesar de ter sido abstraído nos exemplos anteriores, está em todas as mensagens CoAP. No caso de uma requisição sem confirmação (NON), a comunicação ocorrerá de forma semelhante, porém, todas as respostas também serão do tipo *non-confirmable* (não existirão mensagens de ACK na comunicação).



(a) Com resposta *piggy-backed*



(b) Com respostas separadas

Figura 4.3 – Requisições CoAP [12].

Message Queue Telemetry Transport (MQTT)

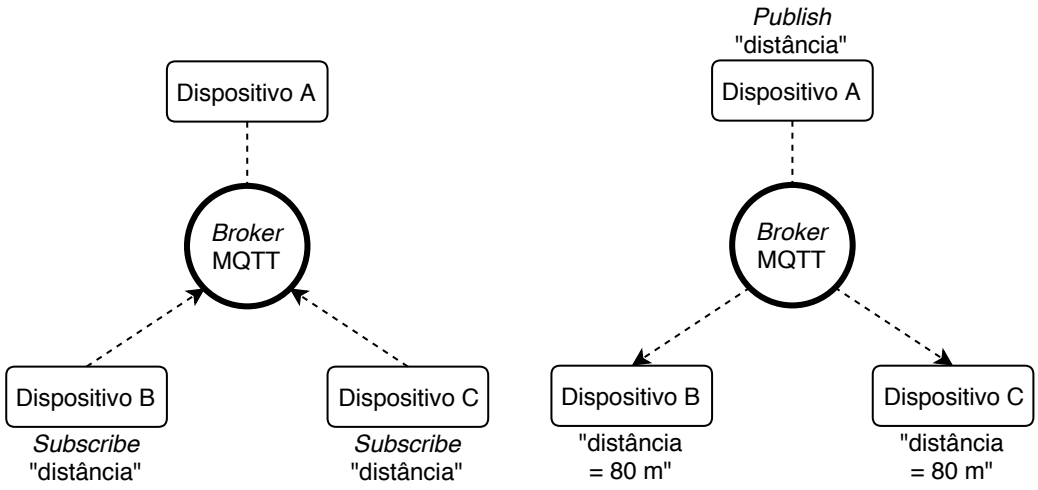
O protocolo *Message Queue Telemetry Transport* (MQTT) foi criado no ano de 1999 por dois engenheiros: Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech). Em março de

2013 foi padronizado pela OASIS[1], e desde então é muito utilizado em diversas aplicações relacionadas à *Internet of Things*. O MQTT, atualmente em sua versão 3.1.1, é otimizado para redes com processamento limitado, pouca memória e alta latência. Por esse motivo, ele é leve, eficiente, possui largura de banda agnóstica aos dados, ou seja depende do caso de uso e de como o *payload* é estruturado, sendo capaz de suportar vários níveis de Qualidade de Serviço (QoS) [14].

Ao ser realizada uma comparação entre o MQTT e outros protocolos comumente utilizados, como o HTTP, é possível observar suas vantagens, como tempo de resposta mais rápido, baixo uso de largura de banda, baixo consumo de bateria e confiabilidade na transmissão dos pacotes [15].

Este protocolo faz uso da arquitetura cliente ⟷ servidor e utiliza o paradigma *publish/subscribe* (pub/sub). No MQTT todos os dispositivos ou clientes se conectam em um *broker*, que funciona como o servidor (é um *hub*). A função do *broker* é receber, enfileirar e enviar as mensagens recebidas dos dispositivos que enviam (*publishers*) para os que recebem (*subscribers*) [16]. Para realizar comunicação com o *broker* é necessário uma conexão TCP/IP, TSL ou *WebSocket* [17].

Toda a troca de mensagem realizada no MQTT se baseia na definição de tópicos (do inglês, *topic*), que é um endereço. Após estabelecida a conexão com o *broker*, os dispositivos que desejarem enviar mensagens (denominados *publishers*) e os que desejarem receber devem inscrever-se em um ou mais *topics* (*subscribe*) para obterem as informações enviadas nesse ou nesses *topics*. Segue um exemplo para esclarecer essas definições.



(a) Conexão ao *broker* **(b)** Mensagem enviada (*publish*).

Figura 4.4 – Exemplo de comunicação MQTT [Elaborada pelos autores].

Imagine a seguinte situação: três dispositivos, *A*, *B* e *C*. *A* utiliza um sensor que mede constantemente uma distância, que deve ser repassada periodicamente aos outros dois dispositivos (*B* e *C*). Fazendo uso do MQTT, inicialmente os três dispositivos estabelecem uma conexão (suponhamos que seja TCP/IP) com o *broker*. Em seguida, como pode ser visto na Figura 4.4a, os dispositivos que desejam obter a distância (*B* e *C*) se inscrevem (*subscribe*)

[1]https://www.oasis-open.org/org

no *topic* "distância". Ou seja, toda vez que *A* enviar uma mensagem para o *broker*, *B* e *C* (após realizada a inscrição) estarão aptos a recebê-la. Na Figura 4.4b é possível observar que o dispositivo *A* enviou (*publish*) para o *topic* "distância" o valor "80 m". Logo, *B* e *C* recebem essa mensagem ("distância" = 80 m).

O protocolo MQTT funciona através de sucessivas trocas de pacotes de controle [17]. Um pacote de controle contém três partes (algumas opcionais), que sempre estarão na ordem a seguir: *fixed header* ou cabeçalho fixo no português, *variable header* ou cabeçalho variável no português e o *payload*. O único segmento obrigatório é o *fixed header*, que possui tamanho variável entre 2 e 5 *bytes*. O *byte* 1 contém o tipo de pacote de controle MQTT (bits 7 - 4) e algumas *flags* ou sinalizadores específicos para cada pacote de controle MQTT no português (bits 3 - 0). Essas *flags* carregam diversas informações, sendo de grande relevância entre essas o nível de QoS da transmissão (*bits* 1 e 2), que pode ser: QoS 0, onde a mensagem é entregue, no máximo, uma vez; QoS 1, sendo a mensagem entregue, pelo menos, uma vez; QoS 2, cuja mensagem é entregue exatamente uma vez. O *byte 2*, denominado comprimento restante, codifica os *bytes* restantes no pacote (exclui o tamanho do *fixed header*). Seu tamanho varia de 1 a 4 *bytes*, sendo os sete *Low Significant Bits* (LSB) de cada *byte* uma codificação da informação e os sete *Most Significant Bits* (MSB), também de cada *byte*, um indicativo de continuação – "1" simboliza a existência de, pelo menos, mais um *byte* de comprimento restante e "0" o oposto. A estrutura de um pacote de controle MQTT pode ser vista na Tabela 4.5, onde os *bytes* 3 e 4 mostram informações opcionais (depende do tipo de mensagem para existir).

7 6 5 4	3 2 1 0
Tipo de pacote MQTT	Sinalizadores (*flags*)
Comprimento restante (1∼4 bytes)	
Cabeçalho variável (opcional)	
Payload (opcional)	

Tabela 4.5 – Estrutura de um pacote de controle MQTT [8].

Exemplos de pacotes de controle MQTT podem ser vistos na Tabela 4.6. Cabe ressaltar que a comunicação deve ocorrer de forma serial, ou seja, dois pacotes não podem ser enviados simultaneamente, pois isso ocasiona desconexão do cliente. Alguns comandos, sobretudo *PUBLISH* e *SUBSCRIBE* serão mais detalhados na Seção 4. Em [18] são detalhadas todas as *flags* e informações sobre os pacotes de controle MQTT.

Protocolos de descoberta de serviços

A alta escalabilidade da Internet das coisas solicita um mecanismo de gerenciamento de recursos que seja apto a se registrar e descobrir recursos e serviços de forma autoconfigurada, eficiente e dinâmica. Os principais protocolos nesta área são o DNS de *multicast* (mDNS) e DNS *Service Discovery* (DNS-SD), que podem constatar recursos e serviços oferecidos pelos dispositivos IoT.

Nome	Código	Direção de Fluxo	Descrição
Reservado	0	Proibido	Reservado
CONNECT	1	Cliente → Servidor	Requisição de conexão
CONNACK	2	Cliente ← Servidor	ACK de conexão
PUBLISH	3	Cliente ⇄ Servidor	Publicação de mensagem
PUBACK	4	Cliente ⇄ Servidor	ACK de publicação
PUBREC	5	Cliente ⇄ Servidor	Publicação recebida (QoS 2, parte 1)
PUBREL	6	Cliente ⇄ Servidor	Publicação liberada (QoS 2, parte 2)
PUBCOMP	7	Cliente ⇄ Servidor	Publicação completa (QoS 2, parte 3)
SUBSCRIBE	8	Cliente → Servidor	Requisição de *subscribe*
SUBACK	9	Cliente ← Servidor	ACK de *subscribe*
UNSUBSCRIBE	10	Cliente → Servidor	Requisição de cancelamento de *subscribe*
UNSUBACK	11	Cliente ← Servidor	ACK de cancelamento de *subscribe*
PINGREQ	12	Cliente → Servidor	Requisição PING
PINGRESP	13	Cliente ← Servidor	Resposta PING
DISCONNECT	14	Cliente → Servidor	Solicitação de desconexão
Reservado	15	Proibido	Reservado

Tabela 4.6 – Exemplos de pacotes de controle do MQTT versão 3.1.1 [17].

Apesar do mDNS e o DNS-SD terem sido projetados originalmente para dispositivos cheios de recursos, existem estudos de pesquisa que adaptam versões leves deles para ambientes IoT.

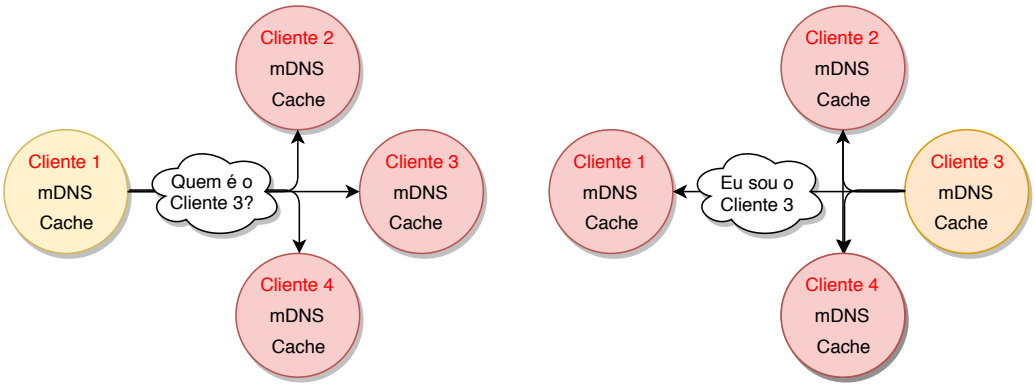
A IoT precisa de algum tipo de arquitetura sem depender de um mecanismo de configuração. Em tal arquitetura, dispositivos inteligentes podem entrar na plataforma ou deixá-la sem afetar o comportamento de todo o sistema. O mDNS e DNS-SD podem suavizar este caminho de desenvolvimento. No entanto, a principal desvantagem desses dois protocolos é a necessidade de armazenamento em cache de entradas DNS, especialmente quando se trata de dispositivos com recursos limitados. Contudo, cronometrando o cache para um intervalo específico e esgotando-o pode resolver esse assunto [8].

DNS de *multicast* (mDNS)

O mDNS é um protocolo de descoberta de serviços desenvolvido pelo *Zeroconf IETF group* e sua função é determinar nomes de domínios sem a necessidade de um servidor DNS convencional [19]. O mDNS é a escolha apropriada para dispositivos embarcados baseados na Internet, pois não há necessidade de reconfiguração manual ou administração extra para gerenciar dispositivos, é capaz de executar sem infraestrutura, podendo continuar seu funcionamento mesmo se houver falha de infraestrutura.

A descoberta de serviços com o mDNS funciona da seguinte forma: cada nó que ingressa na rede transmite uma mensagem contendo sua descrição de serviço para os outros nós na rede, como mostrado na Figura 4.5. Todos os nós clientes que recebem esse pacote registram a descrição do serviço e seu provedor correspondente em uma tabela de consulta de serviço local. Quando um serviço é desejado, o cliente procura a descrição do serviço em sua tabela de consulta. Quando um cliente encontra o endereço de um provedor de serviços

em sua tabela de consulta, ele envia uma mensagem unicast para o provedor de serviços solicitando o serviço desejado [20], como mostrado na Figura 4.5.



(a) Request **(b)** Response

Figura 4.5 – Request/Response do Protocolo mDNS [8].

DNS *Service Discovery* (DNS-SD)

O DNS-SD adiciona o suporte necessário para descobrir os serviços de rede através do DNS. Tais serviços incluem impressão, transferência de arquivos, compartilhamento de músicas, servidores para compartilhamento de fotos, documentos e outros arquivos, além de serviços fornecidos por outros dispositivos locais. Para os usuários, a descoberta de serviço de rede facilita a computação, permitindo que os usuários procurem serviços na rede, em vez de precisarem encontrar o serviço manualmente. Os padrões existentes e o trabalho feito por outras empresas e grupos asseguram que o suporte a várias plataformas esteja disponível.

A função de emparelhamento dos serviços requeridos pelos clientes usando mDNS é chamada de descoberta de serviço baseada em DNS (DNS-SD). Usando esse protocolo, os clientes podem descobrir um conjunto de serviços desejados em uma rede específica, empregando mensagens DNS padrão. A Figura 4.6 fornece uma ilustração de como esse protocolo funciona. O DNS-SD, como o mDNS, pode prover uma operação em *Zeroconf*, que é basicamente um conjunto de técnicas para criar de forma automática uma rede IP sem necessitar de configuração ou servidores, para conectar máquinas sem administração ou configuração externa [21].

Essencialmente, o DNS-SD utiliza o mDNS para enviar pacotes DNS para endereços multicast específicos através do UDP. Há duas etapas principais para processar a descoberta de serviço: a primeira é encontrar nomes de host dos serviços necessários, como impressoras, e combinar endereços IP com seus nomes de host usando o mDNS. Encontrar nomes de host é importante porque os endereços IP podem mudar, enquanto os nomes, não. A segunda é utilizando a função *pairing*, onde ela envia uma mensagem *multicast* com IP e número da porta do *host* utilizado para descobrir um conjunto de serviços de uma rede específica. Usando o DNS-SD, os nomes das instâncias na rede podem ser mantidos constantes pelo maior tempo possível para aumentar a confiabilidade [8].

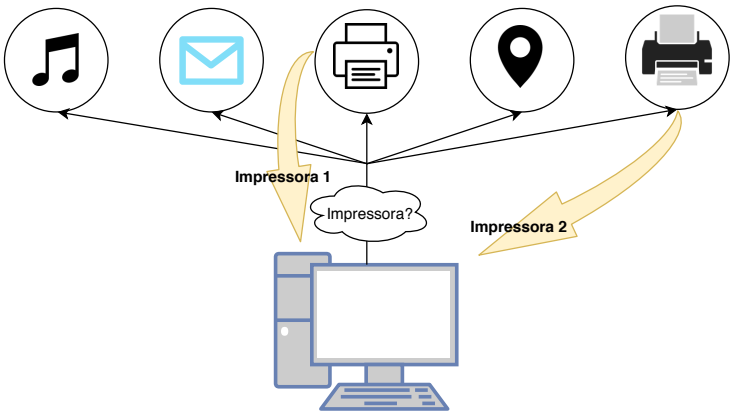


Figura 4.6 – Serviço de Impressão por DNS-SD [8].

Protocolos de infraestrutura

Os protocolos de infraestrutura são necessários para estabelecer a comunicação subjacente necessária para as aplicações IoT. O roteamento é um elemento-chave da infraestrutura IoT e muitos outros parâmetros desses sistemas, como confiabilidade, escalabilidade e desempenho, que dependem fortemente dessa tecnologia. Portanto, há uma necessidade de mais investigação sobre melhorias e otimizações de protocolos de roteamento para atender aos requisitos de IoT [7].

Os principais protocolos de infraestrutura de uma rede de IoT são: RPL, 6LowPAN, IEEE 802.15.4, BLE, EPCglobal, LTE-A, Z-Wave. Cada protocolo contém suas propriedades, suas caracteristicas, vantagens e desvantagens. Neste trabalho, dar-se-á uma breve descrição dos protocolos 6LowPAN e IEEE 802.15.4

6LoWPAN

As redes de Área Pessoal Sem Fio de Baixa Potência (WPANs) nas quais muitas comunicações IoT acontecem, possuem algumas características diferentes das tecnologias anteriores da camada de enlace, como tamanho limitado de pacote, vários comprimentos de endereço e baixa largura de banda [22]. Portanto, houve a necessidade de criar uma camada de adaptação que se ajustasse aos pacotes IPv6 às especificações IEEE 802.15.4.

O acrônimo 6LoWPAN significa IPv6 *Over Low Power Wireless Personal Network*. O 6LoWPAN foi inventado com a finalidade de conservar as especificações que nos permitem usar o IPv6 em redes IEEE 802.15.4, pois o IPv6 tolera uma alta quantidade de endereçamentos de dispositivos, porém seu tamanho é de 128 *bits*, o que acarreta um grande problema, pois se o dispositivo tiver uma baixa capacidade de memória e baixa potência ele não o suportará [7].

O 6LoWPAN é a especificação dos serviços de mapeamento exigidos pelo IPv6 sobre WPANs de baixa potência para manter uma rede IPv6 [23]. Este protocolo se baseia na geração de que a Internet é formada por IP, ou seja, cada dispositivo deverá possuir um IP fazendo, assim, parte da *Internet of Things* (IoT). Cada rede de 6LoWPAN compartilha o mesmo prefixo de endereço. A capacidade total de cada rede é de 64.000 dispositivos,

devido aos limites impostos pelo endereçamento utilizado pelo IEEE 802.15.4, que usa 16 *bits* de endereços para cada dispositivo na rede, obtendo uma identificação IPv6 única. O 6LoWPAN, sem dúvida, apresenta-se como a melhor alternativa para a integração das WPANs à Internet e às redes IP.

O padrão fornece compactação de cabeçalho para reduzir a sobrecarga de transmissão, a fragmentação para atender ao requisito de IPv6 Maximum Transmission Unit (MTU) e o encaminhamento para a camada de link para suportar a entrega de múltiplos saltos [24].

Os datagramas envolvidos pelo 6LoWPAN são seguidos por uma combinação de alguns cabeçalhos. Esses cabeçalhos são de quatro tipos, que são identificados por dois bits [23]: (00) NO 6LoWPAN Header, (01) Dispatch Header, (10) Mesh Addressing e (11) Fragmentation. No 6LoWPAN Header, os pacotes que não estiverem de acordo com a especificação 6LoWPAN serão descartados. A compactação de cabeçalhos IPv6 ou multicast é executada especificando o cabeçalho Dispatch. O cabeçalho Mesh Addressing identifica os pacotes IEEE 802.15.4 que devem ser encaminhados para o linklayer. Para datagramas cujos comprimentos excedam um único quadro IEEE 802.15.4, o cabeçalho Fragmentation deve ser usado. O 6LoWPAN remove muitas sobrecargas do IPv6 de tal forma que um pequeno datagrama IPv6 pode ser enviado através de um único salto IEEE 802.15.4 no melhor dos casos. Também pode comprimir cabeçalhos IPv6 em dois bytes [24].

IEEE 802.15.4

O protocolo IEEE 802.15.4 foi desenvolvido para especificar uma sub-camada para o Medium Access Control (MAC) e uma camada física (PHY) para redes de área privada sem fio de baixa taxa de transmissão (LR-WPAN). Devido às suas especificações, como baixo consumo de energia, baixa taxa de transmissão de dados, baixo custo e alta mensagem *throughput*, ele também é utilizado pela IoT, M2M e WSNs [25].

O protocolo IEEE 802.15.4 fornece uma comunicação confiável, operabilidade em diferentes plataformas e pode lidar com um grande número de nós (cerca de 65k), também fornece um alto nível de segurança, criptografia e serviços de autenticação. No entanto, não fornece garantias de QoS [8].

O padrão IEEE 802.15.4 foi fragmentado em duas camadas: uma desenvolvida pela IEEE (camada inferior) e outra pela *ZigBee Alliance* (camada superior), como pode ser observado na Tabela 4.7.

Usuário	Aplicação
ZigBee Alliance	Suporte a Aplicação
	Rede (NWK)/Segurança (SSP)
IEEE 802.15.4	MAC
	PHY

Tabela 4.7 – Camadas da tecnologia Zigbee [26].

Este protocolo é a base para o protocolo ZigBee como eles se concentram na oferta de serviços de baixa taxa de dados em dispositivos com restrição de energia e constroem

uma pilha de protocolo de rede completa para WSNs. O protocolo IEEE 802.15.4, mais conhecido como *ZigBee*, foi criado pelo IEEE em parceria com a *ZigBee Alliance* [26].

Em redes de comunicações que utilizam o protocolo *ZigBee* o dispositivo fica por longos períodos sem se comunicar com o outro. Seu tempo de acesso conectado é cerca de 30 ms. Por conta dessa característica o protocolo IEEE 802.15.4 é bastante econômico em relação ao consumo de energia.

As comunicações por *ZigBee* são realizadas na faixa das frequências ISM (*Industrial Scientific and Medical*), sendo elas: 2.4 Ghz (mundialmente), 915 Mhz (na América) e 868 Mhz (na Europa). Nesse contexto, a taxa de transferência dos dados é de até 250kbps na frequência de 2.4 Ghz operando com 16 canais; 40 kbps na frequência de 915 Mhz operando com 10 canais; 20 kbps na frequência de 868 Mhz operando com 1 canal [27].

Métricas Físicas e Técnicas de Suavização

Uma das formas de determinar a possibilidade de existência ou não de transmissão observando diretamente o enlace, é utilizando métricas físicas. As principais métricas de monitoramento do enlace baseadas em potência do sinal são aquelas obtidas diretamente dos circuitos integrados dos equipamentos. Dentre as mais utilizadas, podem-se citar o Indicador de Potência do Sinal - RSSI (do inglês, *Received Signal Strength Indicator*, Relação Sinal-Ruído - SNR (do inglês, *Signal-to-Noise Ratio* e Indicador de Qualidade de Enlace - LQI (do inglês, *Link Quality Indicator*).

As métricas que são obtidas diretamente dos circuitos por meio da observação do sinal eletromagnético são chamadas de baseadas em potência do sinal. Sua principal característica é que dispensam computação adicional para obtenção do valor final. Existem também as métricas baseadas em estatísticas de pacotes, como o *Packet Reception Ratio* - PRR (taxa de pacotes recebidos), no entanto não são objetos de estudo deste trabalho.

As métricas físicas utilizadas para o desenvolvimento deste trabalho são listadas e aprofundadas a seguir.

***Received Signal Strength Indicator* - RSSI** é uma medida da potência de radiofrequência de um canal, que pode ser devido a outras transmissões de dados na mesma frequência, radiações de fundo, ou ruídos térmicos. A métrica RSSI também é utilizada para o *Clear Channel Assessment* - CCA que pode ser entendida como uma técnica utilizada para identificar uma possível transmissão em um tempo instantâneo por meio da detecção de níveis de energia acima dos limites estabelecidos para uma transmissão naquele canal.

***Link Quality Indicator* - LQI** foi proposto no padrão IEEE 802.15 [28] e é transmitido como um campo na estrutura de cada pacote, e sua implementação é específica de acordo com o fabricante. A única exigência da referida norma é que ele deve ser calculado utilizando-se uma janela de no mínimo oito observações. Em redes multissaltos seus valores podem ser modificados a cada etapa à medida que a mensagem se propaga através dos equipamentos.

***Signal-to-Noise Ratio* - SNR** compara o nível de um sinal desejado para o nível de ruído de fundo e pode ser expresso como $SNR = \frac{P_{sinal}}{P_{ruído}}$, em que P significa a potência média. Esta métrica é frequentemente utilizada para estimar a taxa de erro de *bits* - BER (do inglês, *Bit Error Rate*), que pode ser extrapolada para a taxa de erros de pacotes recebidos.

Tais métricas, por razões de micro variações na energia elétrica recebida pelo equipamento, ou por interferências ocorridas no mesmo espectro de frequência observável, ou mesmo por ruído térmico, geram valores que podem se alterar abruptamente em um pequeno intervalo de tempo. Mesmo que não ocorressem tais variações, a quantidade de valores fornecidos pelas métricas, a depender do equipamento, pode chegar a mais de 10 mil amostras por segundo. Em virtude desse grande volume de dados recebidos, para trabalhar de forma mais adequada com tais métricas, é importante realizar uma suavização dos seus valores recebidos.

Os métodos de suavização variam de acordo com o problema e os dados. As principais técnicas de suavização são a modelagem estatística, o sistema de modelagem inteligente ou uma modelagem híbrida usando ambas as técnicas. As técnicas clássicas de suavização tratadas neste trabalho são Simples Média Móvel - SMA (do inglês, *Moving Average*) e Suavização Exponencial Simples - SES (do inglês, *Single Exponential Smoothing*). Suavização e previsão, no estudo de séries temporais, possuem alta similaridade, visto que uma suavização gera um valor que pode ser utilizado para predizer um resultado em um passo adiante. Dessa forma, nas subseções seguintes utilizaremos o termo previsão para ficar mais adequado com as referências utilizadas como fonte.

Simple Moving Average - SMA

A técnica de previsão Média Móvel Simples utiliza como previsão para um determinado ponto futuro a média de observações passadas. Esta técnica fornece uma previsão de curto prazo, o que a torna uma escolha interessante quando os dados não apresentam tendência ou sazonalidade. As médias móveis são apresentadas como simples, centradas ou ponderadas. Modelos de médias simples adotados neste trabalho podem ser expressos como:

$$y_t = \hat{X}_t = \frac{x_{t-1} + x_{t-2} + ... + x_{t-n}}{n}, \tag{4.1}$$

onde n significa o tamanho da janela do tempo necessário para determinar a média e x_t o valor previsto.

O termo média móvel é usado porque, à medida que a próxima observação se torna disponível, a média das observações é recalculada, incluindo essa observação no conjunto de observações e negligenciando a observação mais antiga [29]. Este método só é recomendado para prever séries estacionárias, caso contrário a precisão da previsão será prejudicada, devido aos pesos atribuídos às observações de n serem todos iguais e nenhum peso ser dado às observações anteriores nesse período.

Single Exponential Smoothing - SES

A técnica mais conhecida como suavização exponencial simples - SES é semelhante à Média Móvel por extrair das observações das séries temporais o comportamento aleatório por suavização de dados históricos. No entanto, a inovação introduzida pela SES vem do fato desta técnica atribuir pesos diferentes para cada observação na série. Enquanto na SMA as observações são utilizadas para encontrar o valor futuro e contribuem em igual proporção para o cálculo desta previsão, na SES as informações mais recentes são destacadas pela aplicação de um fator que determina sua importância [30].

A SES é útil como uma técnica para suavizar (ou filtrar) variações aleatórias e tem as seguintes propriedades: (i) a redução do peso é atualizada em um dado antigo; (ii) é extremamente fácil de calcular; e (iii) uma quantidade mínima de dados é necessária para computar [31]. O argumento para o tratamento diferenciado de algumas observações da série temporal baseia-se no pressuposto de que as últimas observações contêm mais informações sobre o futuro e, portanto, são mais relevantes para a previsão. No trabalho de [32], as formulações e propriedades são descritas, e apresenta os principais estudos sobre a técnica SES. No trabalho de [33] é mostrado a utilização da técnica SES em dados de telecomunicações. Segundo [34], técnica SES pode ser descrita pela seguinte equação:

$$\hat{Z}_t = aZ_t + (1-a)\hat{Z}_{t-1}, Z_0 = Z_1, t = 1, ..., N \tag{4.2}$$

onde a é a constante de suavização e Z_to valor exponencialmente suavizado. Se a equação for resolvida recursivamente, pode-se observar que os valores antigos de Z_t exercem menos influência na predição $\hat{Z}_t$.

Análise de Tráfego

As seções anteriores trouxeram uma visão geral de alguns conceitos básicos no contexto de IoT, em especial foram apresentados os protocolos MQTT, CoAP, mDNS e DNS-SD, além de algumas métricas de monitoramento de enlace baseadas na potência do sinal. Esta seção, por outro lado, tem como objetivo mostrar na prática alguns dos conceitos vistos anteriormente, com foco no monitoramento lógico (análise do fluxo de informações dos protocolos apresentados) e monitoramento físico do tráfego em uma rede de Internet das coisas, mais especificamente em um ambiente de automação residencial, como por exemplo, em *Smart Home*.

Para realização dos experimentos é utilizado o microcomputador *single-board* Raspberry Pi 3 com sistema operacional Raspbian[5], em sua versão com *desktop*. Entretanto, os experimentos podem ser facilmente adaptados a outros sistemas operacionais, assim como a outros microcomputadores.

Para simular um ambiente com automação residencial é utilizada a plataforma *open*

[5]https://www.raspberrypi.org/downloads/raspbian/

hardware, o NodeMCU. Este é utilizado para comandar os sensores e atuadores envolvidos, como também, com pequenas adaptações, um Arduino equipado com uma *shield* Wi-Fi também pode ser utilizado. O cenário adotado para este experimento é mostrado na Figura 4.7. No decorrer dos experimentos são analisados fluxos de dados específicos dentro do ambiente considerado, com auxílio da ferramenta de análise de rede, o Wireshark.

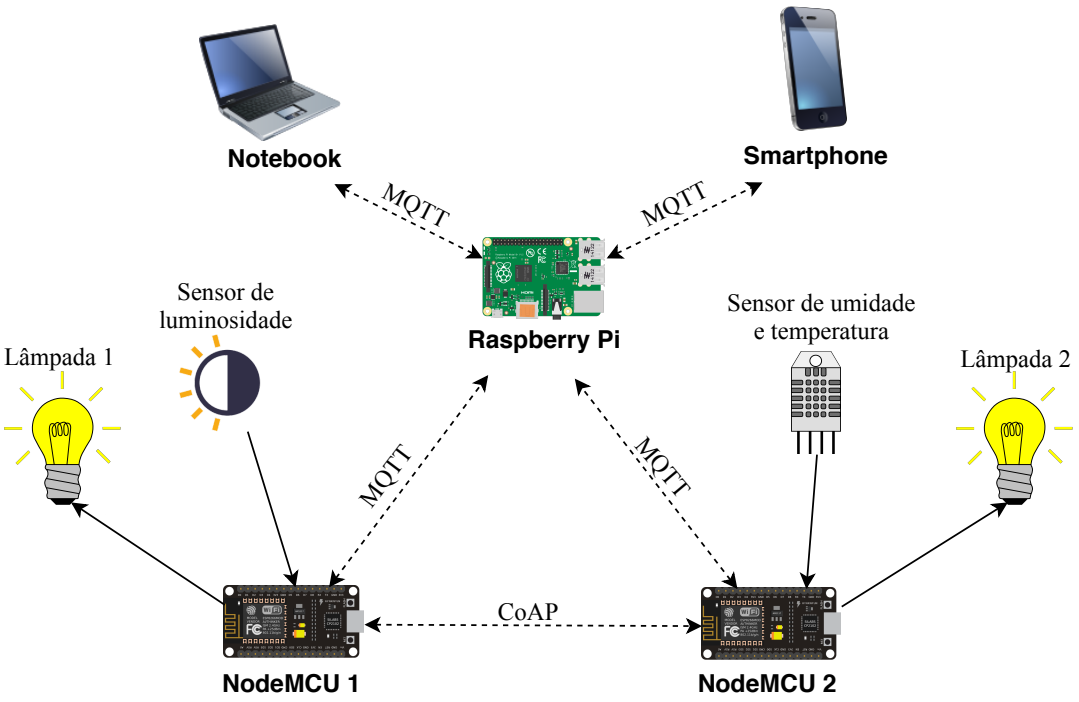


Figura 4.7 – Cenário adotado [Elaborada pelos autores].

Preparando o ambiente

Os requisitos para realizar os experimentos são descritos abaixo:

- 1 Rapberry Pi 2 ou 3 com SO Raspbian;
- 1 Adaptador Wi-Fi com suporte ao modo monitor;
- 2 NodeMCUs;
- 1 Roteador Wi-Fi;
- Arduino IDE;
- Wireshark versão 2.2.6 ou superior;
- Conexão com a internet.

Para os experimentos, o primeiro passo é a instalação do analisador de protocolos Wireshark. No Raspbian, abra o terminal e digite o seguinte comando:

```
$ sudo apt-get install wireshark
```

Além disso, ainda no Raspbian, será necessária a instalação de um *broker* MQTT, que poderá ser acessado apenas localmente. Entretanto, um *broker* online pode ser utilizado sem nenhum prejuízo aos procedimento aqui apresentados. Existem diversas implementações do MQTT para diferentes plataformas. Neste trabalho será utilizada uma das implementações mais utilizada e estável do protocolo, o Mosquitto. Este suporta as versões 3.1 e 3.1.1 do MQTT e pode ser facilmente instalado no Raspberry Pi. No entanto, antes da instalação é importante certificar-se que será instalada a versão mais recente do *broker*, para isso basta abrir o terminal no Raspbian e digitar os seguintes comandos:

```
$ sudo apt-get update
$ sudo apt-get install mosquitto
```

Após isso, o *broker* MQTT já estará rodando automaticamente na porta TCP padrão: 1883. O próximo passo é a preparação dos clientes. Existem diversos clientes MQTT para diferentes plataformas, não sendo, portanto, obrigatória a utilização dos mesmos clientes aqui descritos para a obtenção dos mesmos resultados. Para o envio de comandos da plataforma Android para os NodeMCUs é utilizado o cliente Android MQTT Dash (disponível na Google Play Store[6]) em um *smartphone* e o cliente Arduino PubSubClient (disponível para *download* no GitHub[7]) nos NodeMCUs.

No Arduino IDE, após a instalação da biblioteca PubSubClient, é possível observar o funcionamento do cliente MQTT por meio do exemplo "mqtt_esp8266" que pode ser acessado no menu "Arquivo->Exemplos->PubSubClient". Para que a comunicação MQTT ocorra conforme ilustrado na Figura 4.7, o NodeMCU 1 é inscrito no tópico "lampada1" (tópico que receberá comandos para ligar/desligar a lâmpada 1) e é publicador do tópico "sensor1" (onde serão publicadas atualizações periódicas do *status* do sensor de luminosidade). De forma equivalente, o NodeMCU 2 é inscrito no tópico "lampada2" (tópico que receberá comandos para ligar/desligar a lâmpada 2) e é publicador dos tópicos "sensor2" e "sensor3" (tópicos que receberão as atualizações de umidade e temperatura do ambiente, respectivamente).

Para garantir a consistência das informações entre os NodeMCUs é utilizada a comunicação CoAP. Assim, sempre que houver alguma mudança de *status* em um dos recursos de algum dos dispositivos, esta comunicação poderá ser transmitida a todos os nós da rede. Para essa tarefa a comunicação MQTT também poderia ser utilizada, simplesmente inscrevendo os NodeMCUs em todos os tópicos, porém como o presente trabalho possui caráter didático, a comunicação CoAP é introduzida para que seja possível analisá-la em uma situação real.

Para tornar a comunicação CoAP possível entre os NodeMCUs é utilizada a biblioteca ESP-CoAP, disponível no GitHub[8]. O NodeMCU 1 é utilizado como servidor

[6]https://play.google.com/store/apps/details?id=net.routix.mqttdash&hl=pt_BR
[7]https://github.com/knolleary/pubsubclient
[8]https://github.com/automote/ESP-CoAP

CoAP, fornecendo serviços nos *endpoints* “coap://IP_do_NodeMCU1:5683/lampada1” e “coap://IP_do_NodeMCU1:5683/sensor1”, em que IP_do_NodeMCU1 é o endereço IP do NodeMCU 1 e 5683 é a porta padrão do CoAP. Já o NodeMCU 2 é utilizado como cliente CoAP e é observador dos recursos fornecidos pelo NodeMCU 1, i.e., é notificado toda vez que um recurso é alterado no primeiro NodeMCU 1 (por exemplo, a troca do estado da lâmpada 1). Para a atualização do estado dos atuadores/sensores dos demais nós na rede, poderiam ser criados recursos no servidor CoAP para cada atuador e sensor, de forma que os clientes possam enviar mensagens de atualização de seus sensores/atuadores e observar os demais recursos. Entretanto, para os fins deste trabalho apenas o cenário inicialmente descrito é suficiente.

Além disso, para facilitar o acesso aos serviços dentro da rede e a resolução de nomes localmente, sem a necessidade de infraestrutura de DNS tradicional, é utilizado o mDNS em conjunto com o DNS-SD. Para isso, a biblioteca ESP8266mDNS é utilizada em ambos os NodeMCUs. Com o uso dessa biblioteca, para associar um nome aos NodeMCUs basta utilizar o comando `MDNS.begin(nomenode)`, em que “nomenode” será o nome associado ao NodeMCU. Associando os nomes “nodemcu1” e “nodemcu2”, respectivamente, ao NodeMCU 1 e ao NodeMCU 2, o serviço CoAP, por exemplo, rodando no NodeMCU 1 na porta padrão pode ser acessado simplesmente por “coap://nomenode.local:5683/recurso”. Além disso, para associar um serviço a uma das placas, basta utilizar o comando `MDNS.addService("coap", "udp", 5683)`, em que “coap” é o nome dado ao serviço, “udp” é o protocolo em que o serviço está rodando e “5683” é a porta utilizada. A descoberta desse serviço utilizando o mDNS (DNS-SD) pode ser feita por meio do comando `MDNS.queryService("coap","udp")`. No NodeMCU 1 é adicionado o serviço “coap” e em toda inicialização do NodeMCU 2 é realizada uma busca para descobrir o IP associado ao NodeMCU que possui o serviço “coap”. Assim, embora o endereço IP do NodeMCU 1 mude, a conexão ainda poderá ser feita por meio da descoberta de serviço. O código completo e comentado utilizado em ambos os NodeMCUs pode ser encontrado no GitHub[9].

Por fim, no dispositivo Android, já com o MQTT Dash instalado, o primeiro passo é criar a conexão com o servidor. Para isso, abra o *app* e clique no botão (+), no canto superior direito, em seguida preencha as informações necessárias, sendo essas informações, o nome da conexão (identificada pelo campo “Name”) e endereço do *broker* (identificado pelo campo “Address”). Os outro dados necessários, como porta e *Client ID* já são preenchido automaticamente com valores padrão. Existe ainda a possibilidade de informar o nome de usuário e senha, entretanto, como estes não foram configurados no servidor deverão ficar em branco. Neste momento a conexão com o *broker* já pode ser estabelecida. Para que o dispositivo Android possa controlar as lampadas, é criado um botão no MQTT Dash, que a cada vez que é tocado faz uma publicação no tópico “lampada1”. Para criar o botão basta acessar a conexão criada e clicar novamente no botão (+), em seguida algumas opções

[9]https://github.com/syllasrangel/Minicurso_SBrT2018

serão exibidas, selecione a opção “Switch/button” e dê um nome ao botão, e em seguida, nos campos “Topic (sub)” e “Topic (pub)” digite os tópicos em que o cliente Arduino deve ser inscrito e publicador, respectivamente, neste caso “lampada1” em ambos. Feito isso, clique no botão salvar (canto superior direto). Procedimento semelhante pode ser feito para a lâmpada 2 e para os sensores, com a diferença que para os sensores deve ser selecionada a opção “Text” e apenas o campo “Topic (sub)” deve ser preenchido.

Captura de tráfego em redes sem fio IEEE 802.11

O padrão IEEE 802.11 define as especificações para o controle de acesso ao meio - MAC (do inglês, *Media Access Control*) e para a camada física - PHY (do inglês, *Physical layer*) em WLANs (do inglês, *Wireless Local Area Networks*). Apesar da tendência em se buscar alternativas com eficiência energética cada vez maior nas redes de Internet das coisas, é inquestionável que o Wi-Fi (tecnologia baseada no padrão IEEE 802.11) apresenta um papel fundamental nas comunicações sem fio nestas redes, especialmente em ambientes comerciais e residencias, em que conexões de alta taxa de dados como a Internet são requeridas. Assim, neste trabalho, será considerada a captura do tráfego de dados em uma rede IEEE 802.11.

Existem várias informações que podem ser capturadas em uma rede IEEE 802.11, como pacotes de controle e de gerenciamento, ou ainda informações da camada de rádio, como potência do sinal e taxa de dados. Entretanto, neste momento o foco será o tráfego regular de dados, i.e, a troca de informações entre os dispositivos na rede. Existem dois possíveis cenários para a realização dessa captura: (i) o *software sniffer* que captura na interface de rede através da qual os pacotes serão transmitidos e recebidos; ou (ii) a captura é realizada em uma interface de rede externa à comunicação. No primeiro caso, nenhuma configuração especial é necessária. No entanto, caso seja necessária a captura de dados em uma comunicação externa ao dispositivo ou de informações relacionadas a métricas físicas, será necessária a captura em um modo especial, chamado de modo monitor.

O suporte ao modo monitor é altamente dependente da plataforma, adaptador de rede, *driver* e da biblioteca de captura adotada. Assim, é muito importante verificar a compatibilidade do adaptador de rede utilizado. Para o Raspberry Pi 3, especificamente, existe um projeto chamado nexmon[10] que adiciona suporte ao modo monitor para a sua interface de rede *on-board* no sistema Raspbian. Entretanto, para evitar maiores problemas, é recomendado a utilização de um adaptador de rede externo que possua suporte nativo na plataforma adotada. Em `https://elinux.org/RPi_USB_Wi-Fi_Adapters` é mostrada uma lista de adaptadores que foram testados com o Raspberry Pi em diferentes modos de operação.

Para ativar o modo monitor de um adaptador de rede no Raspbian é necessário a instalação da ferramenta de análise de rede Aircrack-ng ou equivalente. Para a instalação do Aircrack-ng no terminal do Raspbian, basta digitar o seguinte comando:

[10]https://github.com/seemoo-lab/nexmon

```
$ sudo apt-get install aircrack-ng
```

Após a instalação, para ativar o modo monitor basta utilizar o seguinte comando:

```
$ sudo airmon-ng <start|stop> <interface> [canal]
```

em que o campo "<interface>" deve ser substituído pelo nome da interface que possui suporte ao modo monitor, geralmente nomeada como "wlanX", em que X é um número e no campo "[canal]" deve ser informado o canal que a rede Wi-Fi está utilizando. No Brasil, os canais são numerados de 1 a 11 e constituem um dos filtro utilizados nas redes Wi-Fi, além do canal, existe um filtro por SSID (do inglês, *Service Set Identifier*) e um filtro por endereço MAC. Com a captura em modo monitor todos os pacotes IEEE 802.11 transmitidos em determinado canal podem ser capturados, independente dos filtros. Na wiki[11] do Wireshark é feita uma explicação detalhada sobre os filtros e modos de captura. Para iniciar uma captura em modo monitor no Raspbian, após ativar o modo monitor na interface de rede desejada, basta abrir o Wireshark com o comando `$ sudo wireshark`, em seguido no menu "Capture->Options" procurar a interface de rede "wlanXmon" criada pelo comando "airmon-ng" feito anteriormente e marcar a caixa "monitor mode". Feito isso, bata iniciar a captura na interface. A captura feita em modo monitor é apresentada na Figura 4.8. É importante notar que todos os pacotes capturados são pacotes do protocolo "802.11", em que é possível visualizar apenas informações como: o canal utilizado, a largura de banda do canal, taxa de transmissão, entre outras informações relativas ao meio físico de transmissão. Nesses pacotes, os dados são criptografados, assim para que seja possível ver o tráfego regular de dados será necessário descriptografá-los. Para que o Wireshark consigo descriptografar os dados é necessário que duas condições sejam atendidas: (i) a senha da rede que se pretente analisar o tráfego é conhecida e (ii) 4 pacotes de *handshake* EAPOL da rede de interesse são capturados. Os pacotes EAPOL são enviados quando um dispositivo solicita conexão na rede e podem ser identificados pelo filtro EAPOL no Wireshark. Dado que as condições são satisfeitas, basta acessar o menu "Edit->Preferences->Protocols->IEEE 802.11" e clicar em "Edit" e adicionar uma *key* de acordo com a criptografia adota na rede, por exemplo, para uma rede que utiliza criptografia WPA a *key* deve ser no formato "senha:[SSID]". Na wiki[12] do Wireshark é feita uma explicação detalhada do procedimento de descriptografia na plataforma.

Nas subseções seguintes é feita uma pequena análise do tráfego de dados gerado pelos protocolos mDNS, DNS-SD, MQTT e CoAP dentro do cenário adotado. Para gerar o tráfego é simulado a conexão dos dispositivos na rede e, em seguida, o envio de um comando do *smartphone* para ligar a lâmpada 1.

[11] https://wiki.wireshark.org/CaptureSetup/WLAN#A802.11_Filter_.28Modes.29
[12] https://wiki.wireshark.org/HowToDecrypt802.11

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000		LiteonTe_6e:96:67 (…	802.11	34	Clear-to-send, Flags=........
2	0.000016564		LiteonTe_6e:96:67 (…	802.11	34	Acknowledgement, Flags=........
3	0.000651928		Tp-LinkT_4b:68:63 (…	802.11	34	Acknowledgement, Flags=........
4	0.001754726		Raspberr_4a:c6:21 (…	802.11	34	Clear-to-send, Flags=........
5	0.001781447	Raspberr_4a:c6:21	LiteonTe_6e:96:67	802.11	117	QoS Data, SN=1630, FN=0, Flags=.p.....T
6	0.001797282	Tp-LinkT_4b:68:63 (…	Raspberr_4a:c6:21 (…	802.11	52	802.11 Block Ack, Flags=........
7	0.003225002		Tp-LinkT_4b:68:63 (…	802.11	34	Acknowledgement, Flags=........
8	0.008059667		LiteonTe_6e:96:67 (…	802.11	34	Clear-to-send, Flags=........
9	0.008093888		LiteonTe_6e:96:67 (…	802.11	34	Acknowledgement, Flags=........
10	0.009590937		Tp-LinkT_4b:68:63 (…	802.11	34	Acknowledgement, Flags=........
11	0.040145615		LiteonTe_6e:96:67 (…	802.11	34	Clear-to-send, Flags=........
12	0.040189733		LiteonTe_6e:96:67 (…	802.11	34	Acknowledgement, Flags=........
13	0.040839161		Tp-LinkT_4b:68:63 (…	802.11	34	Acknowledgement, Flags=........
14	0.042353711		Raspberr_4a:c6:21 (…	802.11	34	Clear-to-send, Flags=........
15	0.042414185	Raspberr_4a:c6:21	LiteonTe_6e:96:67	802.11	117	QoS Data, SN=1631, FN=0, Flags=.p.....T
16	0.042430176	Tp-LinkT_4b:68:63 (…	Raspberr_4a:c6:21 (…	802.11	52	802.11 Block Ack, Flags=........
17	0.057831170		LiteonTe_6e:96:67 (…	802.11	34	Acknowledgement, Flags=........

Figura 4.8 – Captura em modo monitor no Wireshark [Elaborada pelos autores].

Análise de tráfego da comunicação mDNS/DNS-SD

Toda consulta DNS para um nome que termine com ".local"é enviada para o endereço *multicast* de *link* local IPv4 do mDNS 224.0.0.251 (ou seu equivalente para o endereço IPv6 FF02::FB). Dessa forma, quando existe uma requisição de resolução de nome de *host* local (por meio de uma consulta mDNS), todos os nós da redes irão recebê-la e apenas o ou os nós autorizados irão respondê-la. Os dois primeiros pacotes mDNS capturados saõ apresentados na Figura 4.9. O NodeMCU 1 (IP 192.168.0.111) faz uma requisição de resolução do nome “raspberrypi.local” (nome padrão associado ao Raspeberry no Raspbian) para o endereço de *multicast* mDNS. Em seguida, o Raspberry (IP 192.168.0.160) responde diretamente ao NodeMCU 1 (em uma resposta *unicast*), informando seu endereço. Entretanto, como ilustrado pelos dois pacotes posteriores, nem sempre a resposta será em *unicast*. Os pacotes 2410 e 2442 ilustram a descoberta de serviços baseada em mDNS (DNS-SD). Nesse caso, o NodeMCU 2 (IP 192.168.0.112) faz uma consulta mDNS com interesse no recurso “coap” que roda sobre “udp”, neste caso apenas uma resposta foi recebida no endereço *multicast*, vinda do NodeMCU 1 (dono do recurso consultado). Vale ressaltar que nessa situação mais de uma resposta é aceitável, já que vários nós na rede podem possuir este recurso.

mdns

No.	Time	Source	Destination	Protoc	Length	Info
1276	7.560401646	192.168.0.111	224.0.0.251	MDNS	135	Standard query 0xf506 A raspberrypi.local
1297	7.566677002	192.168.0.160	192.168.0.111	MDNS	156	Standard query response 0xf506 A raspberrypi.local,
2410	23.734257641	192.168.0.112	224.0.0.251	MDNS	134	Standard query 0x0000 PTR _coap._udp.local
2442	23.811758412	192.168.0.111	224.0.0.251	MDNS	293	Standard query response 0x0000 PTR nodemcu1._coap._

Figura 4.9 – Captura de pacotes mDNS [Elaborada pelos autores].

Análise de tráfego da comunicação MQTT

Na mesma situação descrita anteriormente, quando analisado o tráfego de dados MQTT (por meio do filtro “mqtt” do Wireshark) é possível observar os pacotes mostrados nas Figuras 4.10 e 4.12. Na Figura 4.10 é mostrado o primeiro pacote MQTT capturado pelo Wireshark. Analisando melhor esta solicitação é possível perceber algumas informações relevantes para a comunicação, a primeira informação importante que este pacote fornece é

a versão do protocolo na qual a comunicação se estabelecerá. Na Figura 4.11 é apresentado em "Version: 4" que o cliente está solicitando comunicação com a versão 3.1.1 do protocolo MQTT, que conforme especificado em [18] é representado pelo número 4. Caso o *broker* não seja capaz de se comunicar com esta versão do MQTT, a solicitação de conexão é recusada. Entretanto, na Figura 4.10 pode ser visto que a conexão foi aceita (um pacote CONNACK foi recebido), indicando que o servidor é capaz de se comunicar com a versão especificada do protocolo. A solicitação de conexão deve conter, ainda, outras informações importantes, como usuário, senha, *Client ID*, *Keep Alive*, entre outras. No exemplo não foram utilizados usuário e senha, entretanto, em uma implementação prática a utilização destes campos é extremamente recomendada.

Ainda na Figura 4.11 é possível ver o campo *Client ID*, onde é informada uma identificação para o dispositivo solicitante da conexão (neste caso "nodemcu1"), esta deve ser única para que não ocorram conflitos. Já no campo *Keep Alive* é informado o tempo máximo, em segundos, entre o ponto em que o cliente acaba de transmitir um pacote de controle e o ponto em que este começa a enviar o próximo. Em caso de não existir nenhuma informação a ser transmitida, o cliente deve enviar periodicamente um pacote PINGREQ, de modo que o tempo não seja excedido. Caso contrário o servidor irá interpretar como uma falha de rede e desconectará o cliente.

Além disso, na Figura 4.10 é possível observar a solicitação de inscrição do cliente NodeMCU 1 ao tópico "lampada1" (pacote 1316), que é aceita logo em seguida pelo *broker* (pacote 1318).

mqtt

No.	Time	Source	Destination	Protoc	Length	Info
1309	7.573915758	192.168.0.111	192.168.0.160	MQTT	134	Connect Command
1314	7.575820744	192.168.0.160	192.168.0.111	MQTT	123	Connect Ack
1316	7.577595873	192.168.0.111	192.168.0.160	MQTT	127	Subscribe Request (id=2) [lampada1]
1318	7.578450974	192.168.0.160	192.168.0.111	MQTT	123	Subscribe Ack (id=2)

Figura 4.10 – Captura de pacotes MQTT na conexão do NodeMCU 1 [Elaborada pelos autores].

```
MQ Telemetry Transport Protocol, Connect Command
  > Header Flags: 0x10, Message Type: Connect Command
    Msg Len: 20
    Protocol Name Length: 4
    Protocol Name: MQTT
    Version: MQTT v3.1.1 (4)
  > Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget), Clean Session Flag
    Keep Alive: 15
    Client ID Length: 8
    Client ID: nodemcu1
```

Figura 4.11 – Pacote de solicitação de conexão do NodeMCU 1 [Elaborada pelos autores].

Na Figura 4.12 é possível ver inicialmente a solicitação de conexão do cliente Android (IP 192.168.0.109), seguida da solicitação de inscrição, de forma similar ao que ocorreu com o NodeMCU 1. Após isso, o cliente Android MQTT Dash envia uma mensagem do tipo PUBLISH para o tópico "lampada1" do *broker*, que responde com uma mensagem do tipo

PUBACK, isso ocorre pois diferente do NodeMCU 1, o cliente Android solicitou uma conexão com nível de qualidade de serviço 1 (QoS 1). Logo após, o *broker* retransmite a mensagem para todos os inscritos do tópico "lampada1", neste caso apenas para o NodeMCU 1. Perceba que na mensagem enviada ao NodeMCU 1 não houve confirmação de recebimento, isso ocorre porque o nível de qualidade de serviço utilizado na comunicação foi o QoS 0 (do inglês, *Fire and Forget*).

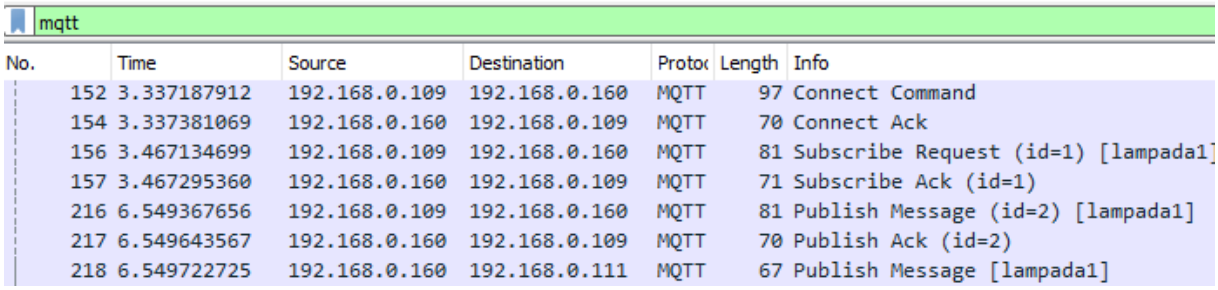


Figura 4.12 – Pacote de solicitação de conexão e de publicação do cliente Android [Elaborada pelos autores].

Análise de tráfego da comunicação CoAP

Ainda utilizando como base o exemplo anterior, após a solicitação de mudança no *status* da lâmpada 1 (feita pelo cliente Android) ser concluída com sucesso, o recurso "lampada1" do servidor CoAP é alterado. Nessa situação, como o cliente CoAP "nodemcu2" é observador do recurso, uma mensagem CoAP deve ser enviada ao NodeMCU 2. Na Figura 4.13 é mostrado toda a comunicação CoAP.

Inicialmente, no NodeMCU 2, é feita uma requisição GET que inclui uma opção "Observe" vazia (padrão definido para ativar a observação em determinado recurso) e uma opção "Uri-Path" com o recurso a ser observado, isso pode ser observado na Figura 4.14. Em seguida, o servidor CoAP (NodeMCU1) envia uma resposta com o *status* atual do recurso em uma mensagem no mesmo formato da requisição inicial e com o mesmo *token*, indicando que a solicitação de observação foi bem sucedida. Após alguns segundos, no momento em que houve uma mudança no *status* da lâmpada 1, uma notificação informando o novo *status* do recurso é enviada ao NodeMCU 2 (pacote 3767), novamente em uma mensagem com o mesmo formato e *token* da requisição inicial. Por fim, o pacote 4253 indica uma nova mudança no recurso observado.

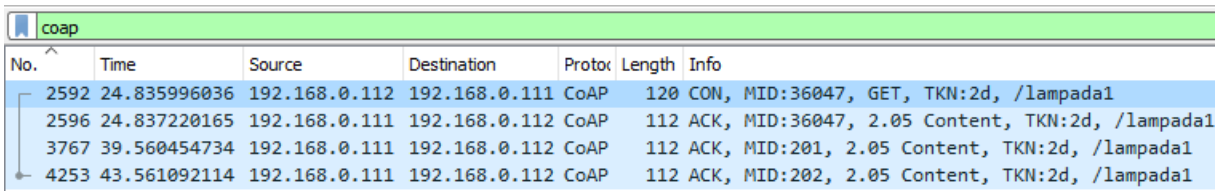


Figura 4.13 – Captura dos pacotes CoAP [Elaborada pelos autores].

```
Constrained Application Protocol, Confirmable, GET, MID:36047
   01.. .... = Version: 1
   ..00 .... = Type: Confirmable (0)
   .... 0001 = Token Length: 1
   Code: GET (1)
   Message ID: 36047
   Token: 2d
 > Opt Name: #1: Observe: 0
 > Opt Name: #2: Uri-Path: lampada1
   [Response In: 4253]
   [Uri-Path: /lampada1]
```

Figura 4.14 – Solicitação de observação de recurso CoAP [Elaborada pelos autores].

Experimento – Monitoramento utilizando métricas físicas

Para a realização do monitoramento utilizando métricas físicas, será utilizado o RSSI e como suavizador, a SMA, conforme discutido na Seção 4. A ideia é identificar de forma superficial se há uma probabilidade de transmissão na rede sem fio em estudo. O seguinte trecho de código é responsável pela conexão na rede WiFi utilizando o NodeMCU.

```
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(700);
  Serial.print(".");
}
```

Em seguida, é necessário suavizar os valores de RSSI recebidos, conforme sugerido no trecho de código exibido adiante. Para isso, foi utilizada uma biblioteca já existente para médias móveis simples, obtida em `https://github.com/ainehanta/arduino-sma`. Por padrão, o SMA adotado utiliza uma janela de tamanho 10, cujo valor pode ser alterado no arquivo `SimpleMovingAverage.h`.

```
#include <SimpleMovingAverage.h>
SimpleMovingAverage avg;

//depois do trecho de codigo do WIFI.status, adicionar:
  avg.begin();
  suave = avg.update(WiFi.RSSI());
```

Depois, deve-se aplicar uma lógica simples para calcular o indicador de uso da rede, conforme o código descrito no seguinte código:

```
int getIndicator() {
  if (WiFi.status() != WL_CONNECTED)
    return -1;
  int dBm = suave;
  if (dBm <= -100)
    return 0;
```

```
  if (dBm >= -50)
    return 100;
  return 2   (dBm + 100);
}
```

Por fim, dentro do método `loop`, faz-se a coleta e atualização dos valores de RSSI utilizando a seguinte linha de comando:

```
suave = avg.update(WiFi.RSSI());
```

O código completo para execução deste experimento pode ser encontrado no GitHub[13].

É importante salientar que neste trabalho foi apresentado um modo simples de inferir o uso de uma rede utilizando métricas físicas, mas a partir disso, podem ser adotadas estratégias mais robustas e inteligentes para a estimativa de uso de uma rede, como adotando-se algoritmos de aprendizagem de máquina para classificar os valores de RSSI que são basais e os valores que são de transmissão, por exemplo.

Conclusão

Neste capítulo foram apresentadas definições que contribuem para o entendimento do conceito de *Internet of Things* (IoT), bem como um breve panorama histórico que guia o leitor desde as primeiras pesquisas até o panorama futuro. Como o próprio nome do capítulo sugere, o trabalho teve como foco o monitoramento lógico e físico do tráfego de redes de IoT através do uso de um *software sniffer*. Os cenários que foram apresentados são executáveis na prática (reais) e utilizam-se de protocolos e dispositivos robustos, porém simples, comumente utilizados em ambientes IoT.

Para os trabalhos futuros, pretende-se implementar novos cenários (cada vez mais próximos da realidade), possivelmente mais desafiadores – com vários dispositivos conectados entre si, bem como estudo e execução de novas estratégias e protocolos para monitoramento lógico e físico do tráfego de redes de IoT. Além disso, durante a execução deste trabalho, algumas limitações foram encontradas, como a necessidade do *Node* estar conectado à rede sem fio para realizar o monitoramento físico e a limitação das implementações do servidor CoAP e do cliente MQTT utilizados. Esses pontos também serão aprimorados em trabalhos futuros.

Referências Bibliográficas

[1] Cisco. Global mobile data traffic forecast update, 2016-2021. `http://www.cisco.com/assets/sol/sp/vni/forecast_highlights_mobile/index.html`, 2017.

[13] https://github.com/syllasrangel/Minicurso_SBrT2018

[2] Kevin Ashton. In the real world, things matter more than ideas. *RFID Journal*, 2009.

[3] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[4] IEEE Internet Initiative. Towards a definition of the internet of things (iot). https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf, 2015.

[5] A. Dixit V. Gupta J. Bradley, C. Reberger. Internet of everything: A $4.6 trillion public-sector opportunity. https://www.cisco.com/c/dam/en_us/about/business-insights/docs/ioe-public-sector-vas-white-paper.pdf, 2013.

[6] Nan Zhang Xinyu Yang Hanlin Zhang Wei Zhao Jie Lin, Wei Yu. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Communications Surveys & Tutorials*, 4(5):1125 – 1142, 2017.

[7] Syllas Rangel C. Magalhães, Victória Tomás Oliveira, Francisco Evangelista N. Filho, and Wendley S. Silva. Monitoramento de tráfego em redes de internet das coisas. *Encontro Unificado de Computação*, 10(11):831–855, 2017.

[8] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[9] Sergio Ferreira. Ipv6 nas redes de sensores sem fio. https://comum.rcaap.pt/bitstream/10400.26/17524/2/SergioFerreira_Tese_Mestrado_2012-2013.pdf, 2013.

[10] Zach Shelby Carsten Bormann, Angelo P. Castellani. Coap: An application protocol for billions of tiny internet nodes. *IEEE Internet Computing*, 16(2):62–67, 2012.

[11] Koojana Kuladinithi, Olaf Bergmann, Thomas Pötsch, Markus Becker, and Carmelita Görg. Implementation of coap and its application in transport logistics. *Proc. IP+ SN, Chicago, IL, USA*, 2011.

[12] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). 2014.

[13] Manveer Joshi and Bikram Pal Kaur. Coap protocol for constrained networks. *International Journal of Wireless and Microwave Technologies*, 5(6):1–10, 2015.

[14] Janakiram MSV. Get to know mqtt: The messaging protocol for the internet of things. https://thenewstack.io/mqtt-protocol-iot/, 2016.

[15] Whei-Jen Chen, Rahul Gupta, Valerie Lampkin, Dale M Robertson, Nagesh Subrahmanyam, et al. *Responsive Mobile User Experience Using MQTT and IBM MessageSight*. IBM Redbooks, 2014.

[16] Pratik Desai. *Python Programming for Arduino*. Packt Publishing Ltd, 2015.

[17] OASIS Standard. Mqtt version 3.1.1. `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf`, 2014.

[18] OASIS Standard. Mqtt version 3.1.1. `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html`, 2014.

[19] M. Krochmal S. Cheshire. Multicast dns. *Apple Inc.*, 2013.

[20] Mehdi Mohammadi. Iot code recipes: Rpl, mdns and rest. `https://github.com/mehdimo/IoTCodeRecipes`, 2016.

[21] M. Krochmal S. Cheshire. Dns-based service discovery. *Apple Inc.*, 2013.

[22] Stephen Dawson-Haggerty David E. Culler Jonathan W. Hui Philip Levis Jeonggil Ko, Andreas Terzis. Connecting low-power and lossy networks to the internet. *IEEE Communications Magazine*, 49:96–101, 2011.

[23] Xavier Vilajosana Thomas Watteyne Luigi Alfredo Grieco Gennaro Boggia Mischa Dohler Maria Rita Palattella, Nicola Accettura. Standardized protocol stack for the internet of (important) things. *IEEE Communications Magazine*, 15:1389–1406, 2013.

[24] David E. Culler Jonathan W. Hui. Extending ip to low-power, wireless personal area networks. *IEEE Communications Magazine*, 12:37–45, 2008.

[25] IEEE 802 Working Group et al. Ieee standard for local and metropolitan area networks part 15.4: Low-rate wireless personal area networks (lr-wpans). *IEEE Std*, 802:4–2011, 2011.

[26] Manuela Ferreira de Lima Vitor Paranhos de Oliveira Carneval Bruna Luisa Ramos Prado Vasques, Igor Bichara de Azeredo Coutinho. Zigbee. `https://www.gta.ufrj.br/grad/10_1/zigbee/index.html`, 2010.

[27] DesmontaCia. Zigbee ou ieee 802.15.4 conheça a tecnologia a fundo. `https://desmontacia.wordpress.com/2011/02/22/zigbee-ou-ieee-802-15-4-conhea-a-tecnologia-a-fundo`, 2011.

[28] IEEE. IEEE 802.15.4 Standard. `https://standards.ieee.org/getieee802/download/802.15.4k-2013.pdf`, 2013. Viewed in Jun 2016.

[29] Pedro Alberto Morettin and Clélia Maria de Castro Toloi. *Modelos para previsão de séries temporais*, volume 1. Instituto de matemática pura e aplicada, 1981.

[30] Steven C Wheelwright, Spyros G Makridakis, et al. Forecasting methods for management. 1973.

[31] Spyros Makridakis, Steven C Wheelwright, and Rob J Hyndman. *Forecasting methods and applications*. John Wiley & Sons, 2008.

[32] Everette S Gardner. Exponential smoothing: The state of the art?part ii. *International journal of forecasting*, 22(4):637–666, 2006.

[33] Everette S Gardner and Joaquin Diaz-Saiz. Exponential smoothing in the telecommunications data. *International Journal of Forecasting*, 24(1):170–174, 2008.

[34] Pedro A Morettin and Clélia Toloi. *Análise de séries temporais*. Blucher, 2006.

CAPÍTULO

5 Predição de Campo e Planejamento de Redes em Sistemas de Comunicações Sem Fio

Joabson Nogueira de Carvalho (IFPB)

Introdução

A implantação de sistemas de comunicações sem fio, assim como outros projetos de engenharia, é precedida de uma etapa de simulação, em que se busca maximizar a cobertura e eficiência do sistema, além de minimizar os custos, evitando possíveis problemas de desempenho insuficiente da rede e até interferências.

Em geral, as questões que precisam ser respondidas no planejamento de sistemas sem fio são referentes à localização e posicionamento das antenas no ambiente, bem como os demais parâmetros de transmissão, como potência e frequência a ser utilizada para um determinado desempenho do sistema. Assim, a simulação desses sistemas tem como base a predição de campo, que é a etapa de cálculo da distribuição do campo produzido para as condições de projeto. Nessa etapa, é possível realizar numericamente ajustes no sistema de transmissão, como alterar a altura ou azimute da antena, potência e frequência do transmissor e até a localização do sistema de transmissão, de forma a otimizar o funcionamento do sistema antes de efetuar a instalação propriamente dita. Uma simulação adequada, aliada a um bom planejamento reduz custos de implantação e aumenta a eficiência do sistema. Em alguns casos, a partir dos valores de campo obtidos na simulação, é possível prever o funcionamento do sistema, obtendo parâmetros secundários como a taxa de transmissão, taxa de erro de bit (BER) e área de cobertura.

Na etapa de predição de campo são usados modelos consagrados na literatura capazes de prever o comportamento do campo em determinadas situações. Em geral, os modelos numéricos são utilizados em *softwares* específicos, que usam maquetes digitais do terreno e/ou cartas topográficas para o cálculo. Normalmente, a distribuição de campo é verificada

numa escala numérica, apresentando um valor calculado para o campo elétrico, ou em escala de cores distribuídos no mapa para melhor visualização.

Neste capítulo são apresentados e discutidos os principais modelos de predição de campo para a faixa de radiofrequência entre 300 MHz e 3 GHz, que atualmente é a porção do espectro radioelétrico com maior número de aplicações de sistemas sem fio. Para exemplificar a aplicação dos modelos de predição de campo serão desenvolvidas duas aplicações, uma para uma rede WLAN (*Wireless Local Area Network*), operando com protocolo *IEEE 802.11b*, em uma aplicação *indoor* e para um sistema do Serviço Móvel Pessoal – SMP, com arquitetura LTE-4G (Long Term Evolution), em ambiente externo e aberto. Nos dois casos, as aplicações foram desenvolvidas na suíte *WinProp*, produzida pela empresa *Altair HyperWorks*.

Modelos de Predição de Campo

O planejamento de sistemas de comunicações sem fio exige o cálculo da distribuição de campo na região em estudo. Os modelos de propagação mais simples são derivados do modelo de propagação no espaço livre, em que apenas a frequência e a distância entre transmissor e receptor são consideradas. Em geral, os modelos de predição de campo são divididos em três classes principais: modelos empíricos; modelos semi-empíricos e modelos determinísticos.

Os modelos empíricos são baseados em ajustes de curvas obtidas a partir de um conjunto de dados medidos, obtendo-se expressões analíticas que aproximam os valores de campo para uma gama de parâmetros do ambiente. Os valores medidos são usados para calcular a perda do sinal ao longo do percurso. Como os modelos dessa classe são baseados em valores medidos, esses carregam implicitamente todos os fatores de propagação do ambiente. Os modelos empíricos são mais simples de serem implantados e requerem menor esforço no pré-processamento [1]. No entanto, para a maioria dos casos, os parâmetros do ambiente e frequência diferem bastante dos valores adotados no levantamento das curvas iniciais, limitando a aplicação dos métodos ou exigindo compensações adicionais numéricas para aumentar a precisão. Dentre os métodos empíricos podemos citar o *Método de Okumura-Hata*, o *COST-231* e o *Método de Previsões Ponto-Área da Recomendação UIT-R P.1546-1*.

Os modelos semi-empíricos são baseados em dados empíricos, levantados a partir de medições e aspectos determinísticos, utilizando a teoria eletromagnética clássica, que considera os fenômenos de reflexão, refração e difração nas diversas estruturas do ambiente, o que conduz a resultados mais aproximados, embora um maior esforço de pré-processamento seja requerido. Dentre os métodos semi-empíricos destaca-se o *Modelos de Walfisch-Bertoni* e *Ikegami*.

Os modelos determinísticos são baseados na teoria eletromagnética clássica, aplicação da óptica geométrica e métodos numéricos para determinar a solução exata do campo. Para

estes modelos, um elevado esforço computacional e de pré-processamento é exigido, sendo o mesmo mais empregado em ambientes indoor ou em pequenos espaços externos, como a propagação em picocélulas em sistemas de comunicação celular.

A seguir, são apresentados os principais modelos de predição de campo utilizados no planejamento de sistemas de comunicações sem fio na faixa de radiofrequência considerada, entre 300 MHz e 3 GHz que envolve a maioria das aplicações de sistemas de comunicações sem fio na atualidade.

Modelos Empíricos

Os modelos empíricos são produzidos a partir de valores medidos, efetuados no local em estudo, por maio de regressão ou ajustes de curvas, obtendo-se fórmulas simples e de fácil implementação. Dentre os modelos empíricos, o *Modelo de Okumura Hata* é o mais utilizado em sistemas de comunicações móveis celular.

Modelo de Okumura Hata

O Modelo de Okumura é um modelo empírico que se baseia em uma família de curvas, obtidas a partir de extensivas campanhas de medições realizadas na cidade de Tóquio. Aplica-se esse modelo na gama de frequências entre 150 MHz e 1920 MHz, embora possa ser extrapolado para além desse valor, e distâncias de 1 a 100 km, para alturas de antenas variando de 30 a 1000 m.

O estudo de Okumura produziu um conjunto de curvas para ajustes do modelo em outras situações, como diferentes alturas de antena, adensamento de edificações e ajustes na frequência e distância, tornando a aplicação do método imprecisa. Hata apresentou uma formulação para o modelo de Okumura como uma fórmula padrão para a perda de propagação em ambientes urbanos, com equações de correção para aplicações em ambientes suburbanos e rurais. O *Modelo de Okumura-Hata*, como ficou conhecido, é válido para a faixa de 150 a 1500 MHz [2].

No modelo de Hata, a perda mediana no caminho de propagação ($L(dB)$) para áreas urbanas é determinada por meio da Expressão (5.1):

$$L_{urbano}(dB) = 69,55 + 26,16\log(f) - 13,82\log(h_t) - a(h_r) + [44,9 - 6,55\log(h_t)]\log(d), \tag{5.1}$$

em que:
$L =$ atenuação em dB;
$f =$ frequência em MHz ($150 \leq f \leq 1500$);
$d =$ distância em km ($1 \leq d \leq 20$);
$h_t =$ altura do transmissor em metros ($30 \leq h_t \leq 200$);
$a(h_r) =$ fator de correção em dB;

h_r = altura do receptor em metros ($1 \leq h_t \leq 10$).

Para cidades pequenas e médias, o fator de correção é dado pela Expressão (5.2)

$$a(h_r) = (1,1\log(f) - 0,7)h_r - (1,56\log(f) - 0,8). \tag{5.2}$$

Para cidades grandes, o fator de correção depende da frequência, o fator de correção é dado pela Expressão (5.3) para ($f \leq 300$) MHz e pela Expressão (5.4) para ($f > 300$) MHz

$$a(h_r) = 8,29(\log(1,54h_r)^2 - 1,1, (f \leq 300MHz), \tag{5.3}$$

$$a(h_r) = 3,2(\log(11,75h_r)^2 - 4,97, (f > 300MHz). \tag{5.4}$$

Para áreas com características suburbanas e rurais, a perda mediana no percurso pode ser obtida pelas Equações (5.5) e (5.6), respectivamente.

$$L_{suburbano}(dB) = L_{urbano}(dB) - 2[\log(\frac{f}{28})]^2 - 5,4 \tag{5.5}$$

$$L_{rural}(dB) = L_{urbano}(dB) - 4,78[\log(f)]^2 + 18,33\log(f) - 40,94 \tag{5.6}$$

O *Método de Okumura Hata* é aplicado em uma vasta gama de aplicações com boas aproximações. No entanto, as diversas constantes devem ser ajustadas localmente para melhores resultados.

Modelos Determinísticos

Os modelos baseados na teoria eletromagnética possuem maior precisão e confiabilidade dos parâmetros de saída, quando comparados aos modelos empíricos. Os modelos deterministicos se baseiam na teoria de propagação das ondas eletromagnéticas e seus mecanismos de propagação, como reflexão, difração e dispersão. Em geral, os modelos determinísticos podem ser empregados aos vários tipos de ambiente e aplicações.

Dentre os modelos que empregam a teoria de propagação de ondas eletromagnéticas, destacam-se o Traçado de Raios (*Ray-Tracing*), Óptica geométrica - GO, Método das Diferenças Finitas no Domínio do Tempo - FDTD (*Finite-Difference Time-Domain*), Método dos Elementos Finitos - FEM (*Finite Element Method*) e a Teoria Geométrica da Difração - UTD (*Geometrical Theory of Diffraction*) [3].

A aplicação dos modelos baseados na propagação de ondas demanda uma etapa de pré-processamento. Em aplicações urbanas (externas) ou em ambiente *indoor* é necessário informações sobre topografia, vegetações, contruções, móveis e outros elementos do cenário, capazes de interferir na distribuição do campo. Geralmente programas de predição de campo utilizam maquetes eletrônicas georeferenciadas, contendo informações do relevo, vegetação e construções do ambiente, como mostrado na Figura 5.1.

Para obtenção de resultados precisos, as aplicações que utilizam modelos

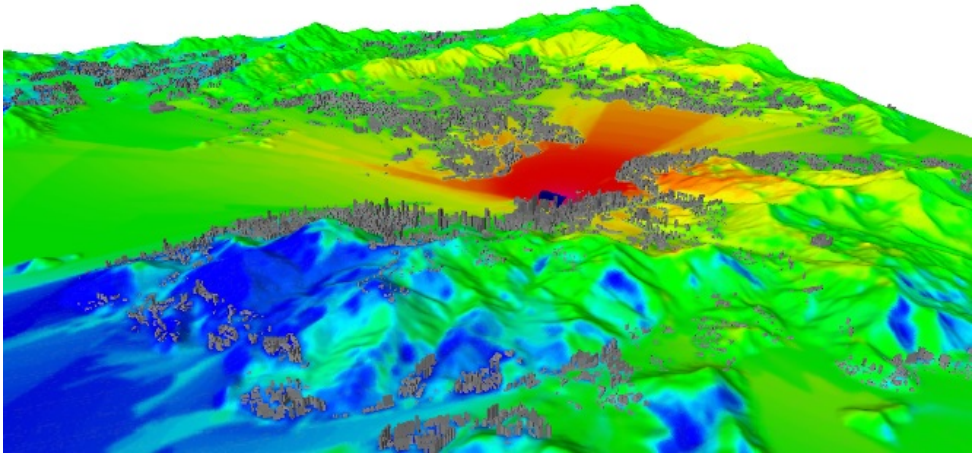


Figura 5.1 – Maquete digital do terreno para aplicações em áreas abertas. Fonte: [4].

determinísticos requer mais detalhes do cenário, tanto da sua geometria como das suas características eletromagnéticas, em que cada elemento do cenário deve ser descrito pela sua forma, dimensão e características dielétricas, incluindo a constante dielétrica, tangente de perdas e condutividade. Em geral, a escolha do modelo depende da relação entre o tamanho dos objetos e o comprimento de onda. Com tantas variáveis, algumas aplicações requerem um complexo modelamento numérico, sendo recomendado o truncamento do resultado, com precisão aceitável.

Para reduzir o esforço computacional, uma classe de modelos vem se destacando, a que considera o caminho de maior energia entre o transmissor e o receptor. Esse tipo de modelo é considerado semi-determinístico, com destaque para o *Modelo do Caminho Dominante*.

Modelo do Caminho Dominante

O *Modelo do Caminho Dominante* é derivado do método de traçado de raios, que utiliza a óptica geométrica e propriedades de reflexão, refração e difração para determinar os caminhos possíveis entre o transmissor e o receptor. Em ambientes urbanos, o número de caminhos possíveis aumenta consideravelmente, elevando o tempo de processamento ou imprecisões, quando se usa critérios de truncamento no modelo.

No *Modelo de Traçado de Raios* o percurso da onda eletromagnética é aproximado pela óptica geométrica, com a utilização de raios como na propagação da luz. Para estimar a perda de percurso, diversos raios são lançados a partir do transmissor. Cada raio é tratado individualmente no ambiente, respeitando os mecanismos de propagação, como reflexão, refração e difração. Os diversos raios que atingem o ponto de interesse (receptor) são somados, considerando a amplitude e fase de cada um, obtendo-se a intensidade de campo naquele ponto específico [3].

O *Modelo do Caminho Dominante* considera apenas o caminho mais relevante, em que

os raios possuem mais energia e podem contribuir de forma significativa na composição da soma final [5]. Esse procedimento reduz o tempo de computação mantendo a precisão de modelos como o traçado de raios. O modelo de traçado de raios não computa a resposta ao impulso do canal. Dentre as vantagens do modelo destacamos a de não exigir pré-processamento do cenário e reduzido tempo de computação para resultados com boa precisão numérica. Na Figura 5.2 é mostrada a diferença da aplicação dos modelos de traçado de raios (a), que considera diversos raios chegando ao receptor e caminho dominante (b), que considera apenas o caminho de maior energia.

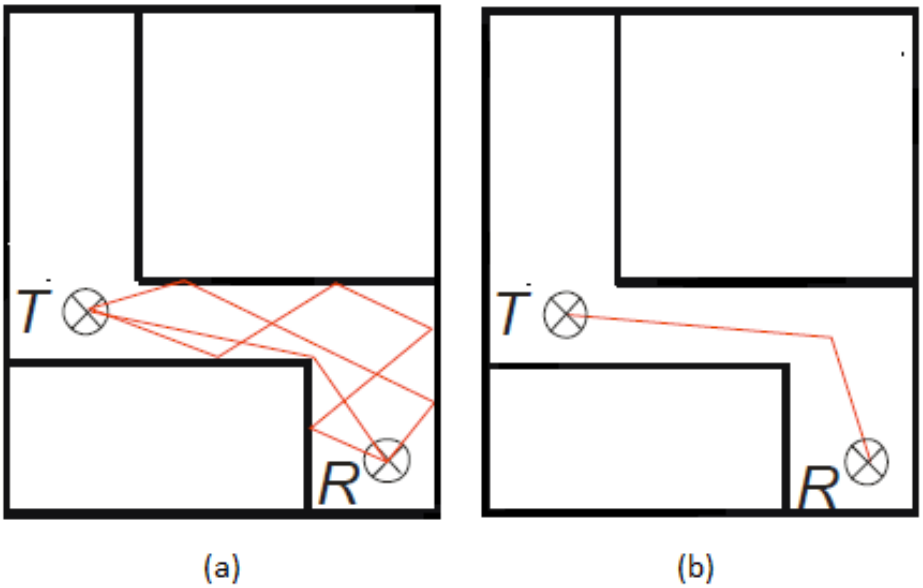


Figura 5.2 – Comparação entre o Modelo de Traçado de Raios (a) e Caminho Dominante (b). Fonte: [4].

O *Modelo do Caminho Dominante* foi adaptado para diversos ambientes, produzindo sub modelos que incluem:

- *Modelo do Caminho Dominante Indoor* – IDP;
- *Modelo do Caminho Dominante Urbano* – UDP;
- *Modelo do Caminho Dominante Rural* – RDP.

Para aplicação do modelo, a área de interesse deve ser dividida em reticulados uniformes, em geral quadrados de tamanho suficiente para que o valor do campo seja considerado uniforme no seu interior. O *Modelo do Caminho Dominante* determina o caminho com trajetórias de maior energia entre o transmissor e cada *pixel* (reticulado) da região, devendo ser considerados os efeitos de propagação para cada caminho.

Para computar a intensidade de campo em cada *pixel* no IDP, a Expressão (5.7) deve ser utilizada [6].

$$E = 104,77 - 10p\log(\frac{d}{m}) - \sum_{i=0}^{n} f(\varphi, i) - \sum_{j=0}^{m} t_j + \frac{1}{c}\sum_{k=0}^{c} w_k + g_t + p_t, \qquad (5.7)$$

em que:
d = distância entre o transmissor e o *pixel*;
p = expoente de perda de percurso, dependente da situação da propagação;
$f(\varphi, i)$ = função para cada trajetória de atenuação;
t_j = perda de transmissão para cada penetração j através de paredes;
w_k = ganho de guiamento para cada pixel ao longo do caminho de propagação;
g_t = ganho da antena na direção do percurso considerado;
p_t = potência do transmissor (em dBm).

O valor de p depende da situação de propagação. Em prédios com muito mobiliário $p = 2,4$, enquanto que nos prédios vazios $p = 2,0$ é sugerido. A função f produz a perda (em dB) que é causada por uma difração. Todas as perdas de difração são acumuladas ao longo de um caminho de propagação. As perdas de transmissão (penetração) t_j também são acumuladas ao longo de um caminho de propagação, bem como os efeitos de guia de onda. O efeito do guiamento de onda ocorrem se a onda se propaga através de um longo corredor e ocorrem reflexões nas paredes. Assim, um ganho de guia de onda adicional em dB é adicionado e acumulado ao longo de todos os *pixels* no caminho de propagação. O ganho direcional da antena g_t (na direção do caminho de propagação) e a potência do transmissor p_t são também considerados na expressão.

Em ambientes urbanos, a versão UDP deve ser empregada. Nesse caso, a Expressão (5.8) deve ser utilizada para calcular a intensidade do campo recebido em cada caminho de propagação.

$$E_{rec} = C + G_T - L_{dist}(d) - \sum_{i=1}^{N} L_{diff}(\phi_i) + G_W(d), \tag{5.8}$$

sendo:
C = constante que descreve a potência transmitida;
G_T = ganho da antena transmissora na direção do percurso considerado;
L_{dist} = perda em função da distância;
$L_{diff}(\phi)$ = função para cada trajetória de atenuação;
$L_(diff)$ = perda por difração;
G_W = ganho devido ao efeito de guiamento da onda.

A dependência com a distância é descrita pelo parâmetro L_{dist} e varia para as seguintes situações: L_{dist} depende da situação $LOS/NLOS$ e da distância entre o transmissor e receptor. Se essa distância for maior que a distância do ponto de interrupção (bloqueio do sinal), dois expoentes de perda de caminho (antes e depois do ponto de interrupção) são aplicados. A distância do ponto de interrupção pode ser determinada usando a Expressão (5.9).

$$d_{bp} = \frac{4\pi h_t h_r}{\lambda}. \tag{5.9}$$

Considerando h_r (altura da antena do receptor), h_t (altura da antena do transmissor) e o comprimento de onda λ. A dependência da distância é levada em conta de acordo com a Expressão (5.10).

$$L_{dist}(d) = \begin{cases} 10p_1 \log(d);\ d \leq d_{bp} \\ 10p_1 \log(\frac{d}{d_{bp}}) + 10p_2 \log(d);\ d > d_{bp} \end{cases} . \tag{5.10}$$

Os parâmetros p_1 e p_2 descrevem os diferentes expoentes da perda de percurso para a área antes e depois do ponto de interrupção. Além disso, as situações de LOS são tratadas de maneira diferente das situações de $NLOS$. Isso significa que há quatro expoentes de perda de trajetória considerados para a dependência de distância:

- LOS antes do ponto de interrupção;
- LOS após o ponto de interrupção;
- NLOS antes do ponto de interrupção;
- NLOS após o ponto de interrupção.

O parâmetro L_{diff} representa uma perda individual devida à difração no percurso de propagação. Se houver mais de uma difração, deve ser considerada a soma de todas as perdas de difração individuais. Para o UDP foi desenvolvido um novo modelo de difração que permite o ajuste simples do comportamento [4]. A perda de difração depende apenas do ângulo φ entre o incidente e o raio difratado em uma cunha de difração. Assim, o fator $L_{diff}(\varphi)$ é dado pela Expressão (5.11).

$$L_{diff}(\varphi) = L_{min} + \begin{cases} L(\varphi);\ \varphi < \varphi_{max} \\ L_{max};\ \varphi \geq \varphi_{max} \end{cases} . \tag{5.11}$$

A Suite *WinProp*

A Suite *WinProp* é um conjunto de *softwares* utilizada para análise de propagação e planejamento de redes de rádio, com aplicações que variam de satélite a terrestre, de rural (por meio de links de rádio urbanos) a *indoor*. O *software* é distribuido pela empresa *Altair | HyperWorks*, podendo ser aplicado nos seguintes cenários:

- Rural e Residencial;
- Urbano e Suburbano;

- *Indoor* e Campus;
- Túneo e metrô;
- Veicular e sistemas variantes no tempo;
- Geo e Satélite.

O *WinProp* suporta bases de dados topográficos (*pixels*) georeferenciados com ou sem informação de altura de edificações, vegetação e outros elementos. Também suporta bases de dados (em três dimensões ou planares) de objetos ou paredes. O *software* contém editores gráficos para construção dos elementos do cenário e edição das propriedades eletromagnéticas, bem como a conversão de diversos formatos de bases de dados.

A ferramenta possibilita a escolha de diversos modelos de predição de campo, incluindo modelos empíricos e semi-empíricos (calibração com medições são possíveis), modelos de traçado de raios em três dimensões, bem como o Modelo de Caminho Dominante (DPM). Além da predição da perda do caminho, o atraso e a propagação angular também podem ser computados, bem como $LOS/NLOS$, resposta ao impulso do canal, perfil angular e caminhos de propagação.

Dependendo da aplicação, o *WinProp* oferece simuladores de rede estáticos, Monte-Carlo e dinâmicos. O *software* permite o planejamento de cobertura e capacidade, bem como simulações de rede (desempenho de algoritmos, análise de atrasos, etc.). O usuário pode definir o tráfego (dependente de localização) para o circuito e para serviços de comutação de pacotes (incluindo as distribuições estatísticas, mobilidade, dentre outros).

No planejamento de redes é uma ferramenta poderosa, pois diferentes parâmetros de transmissão podem ser definidos (largura de banda, MCS, taxa de dados, SNIR (Razão Sinal-Ruído mais Interferência), limite de sinal, potência de transmissão, figura de ruído, dentre outras) e os mapas de cobertura para os parâmetros de saída (atribuição de célula, melhor servidor, qualidade de canal, potência recebida em *downlink* – DL e *uplink* –UL, SNIR) que são calculados individualmente para cada modo de transmissão.

O *WinProp* calcula a capacidade (taxa de transferência, taxas máximas de dados, atrasos de pacote, QoS) dos diferentes enlaces de rádio e células na rede com base na análise de cobertura e nas suposições de tráfego. Limitações de capacidade e células sobrecarregadas podem ser detectadas facilmente e as redes podem ser otimizadas para fornecer alta capacidade e rendimento.

Melhorias de capacidade devido ao uso de MIMO (*Multiple-Input Multiple-Output*) são modeladas com precisão devido aos sofisticados modelos de propagação determinísticos. Configurações de antenas arbitrárias (linear, circular) são possíveis e seu impacto no canal de rádio durante a análise de propagação é considerado no planejamento da rede.

A seguir, o *WinProp* é usado para análise da distribuição de campo de uma rede WLAN no interior de uma edificação e em seguida para predição de campo e análise de desempenho de um sistema de comunicações móveis LTE-4G em um ambiente urbano.

A suite *WinProp* disponibiliza um conjunto de ferramentas computacionais para aumentar a acurácia da simulação, com destaque para:

- *Proman* – *Software* de simulação;
- *Wallman* – Usado na construção de base de dados vetorial do cenário, com ferramentas de desenho e conversão;
- *Aman* – Permite a construção dos modelos de antenas a partir do seu diagrama de irradiação;
- *Tuman* – Utilizado para simulação em ambientes fechados, como túnel e outros locais com características semelhantes.

Criando Cenários com o *Wallman*

O *Wallman* é uma ferramenta computacional da suíte *WinProp* que permite a edição da base de dados do cenário a ser utilizado no *Proman*. O *software* disponibiliza um conjunto de ferramentas de desenho para o usuário, bem como a possibilidade de editar as propriedades eletromagnéticas dos materiais. Dessa forma, é possível incluir elementos do cenário como móveis e diversos tipos de paredes em uma simulação *indoor* ou edificações, vegetação e lagos em ambientes urbano.

Os bancos de dados que descrevem cenários urbanos contêm informações sobre cada um dos edifícios. Cada edificação é descrita como um cilindro poligonal com uma altura uniforme e definida acima do nível da rua. Os cantos da forma poligonal são, portanto, bidimensionais. Além disso, diferentes propriedades de material podem ser atribuídas a cada edifício. Em bancos de dados de ambientes *indoor*, a orientação de objetos é arbitrária. Essa é a diferença básica para os bancos de dados de ambientes urbanos. Para modelar um edifício simples com quatro paredes e um telhado plano, são necessários cinco objetos e cada objeto tem quatro cantos. Portanto, os bancos de dados de ambientes internos são limitados a áreas menores com menos objetos. Mas o ambiente pode ser o mais arbitrário possível – desde um pequeno campus até um único quarto. A Figura 5.3 mostra a diferença de um cenário *indoor* (a) e um cenário urbano (b), ambas produzidas com o *Wallman*.

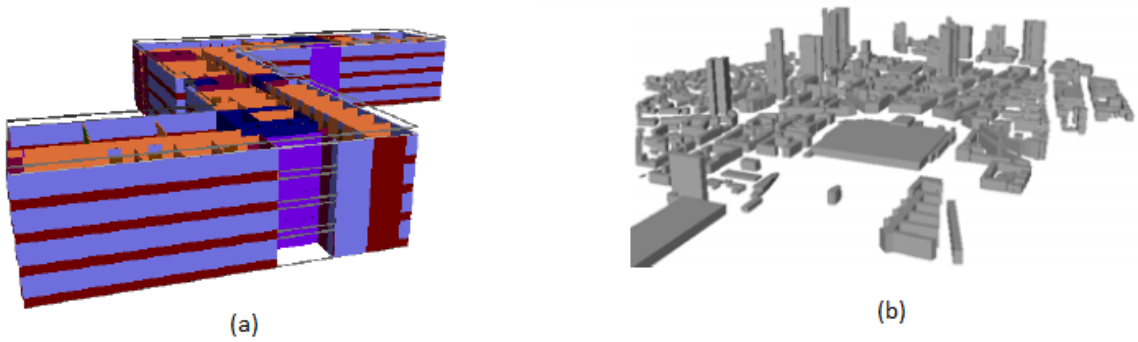


Figura 5.3 – Caixa de diálogo - Base de Dados Vetorial (a) cenário *indoor*, (b) cenário urbano. Fonte: [7].

A maneira mais simples de elaborar uma planta do local de interesse com o uso do *Wallman* é a partir de um desenho digitalizado (ou a partir da planta baixa do local), podendo ser uma imagem. Neste caso, o *Wallman* permite que o usuário coloque o desenho numa camada e desenhe numa outra camada superposta, cobrindo o desenho com o uso das ferramentas gráficas do *software*.

Para criar uma nova base de dados, um novo banco de dados interno deve ser aberto no menu Arquivo *Arquivo - Nova base de dados*. Depois disso, uma caixa de diálogo é mostrada, em que o Tipo de novo banco de dados, ou seja, *Banco de Dados Interno* deve ser selecionado. O modo de operação deve ser escolhido para *Desenhar com bitmap no plano de fundo*. A tela de configuração é mostrada na Figura 5.4.

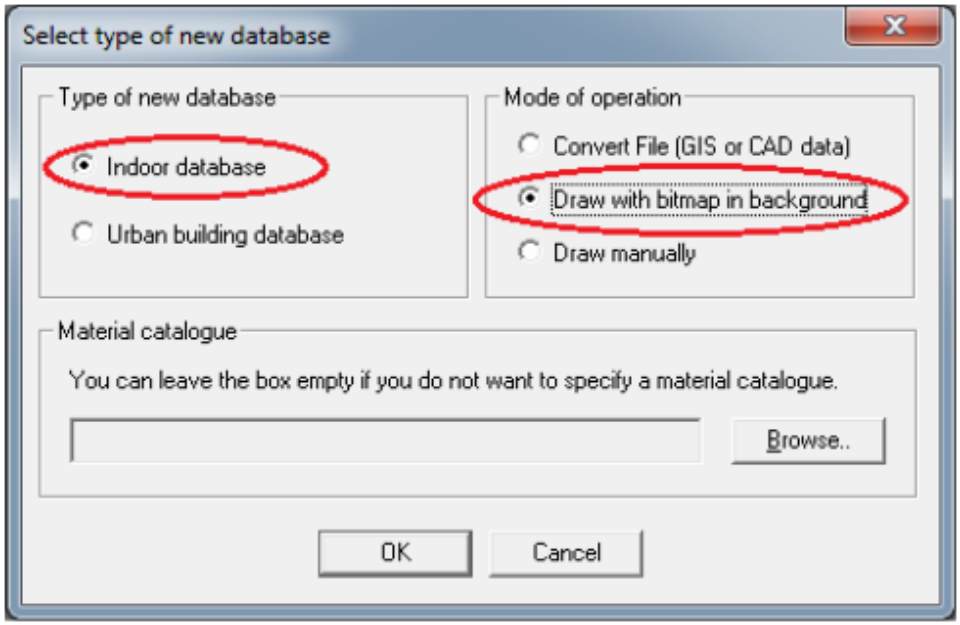


Figura 5.4 – Caixa de diálogo - Nova base de dados. Fonte: [7].

Em seguida, alguns valores do banco de dados atual podem ser especificados, como o modo de desenho ortogonal, que é mais adequado para criar um banco de dados com base em uma imagem de fundo da vista superior, a altura padrão das paredes (pé direito), dentre outros, como mostrado na Figura 5.5. Também é possível atribuir as propriedades dielétricas do material nesta caixa de diálogo.

Na configuração de imagens, o arquivo da planta digitalizada do ambiente deve ser adicionado. Em imagens importadas de cenários urbanos, deve-se utilizar um Raster para georeferenciamento da imagem. Esse passo é importante pois na simulação o ambiente é configurado em máscaras superpostas, sendo uma delas contendo as informações do relevo e outra com o cenário construído no *Wallman*. Portanto os diversos elementos do cenário devem ser desenhados exatamente na sua posição, de forma a se obter uma perfeita superposição da máscara contendo a topografia e a máscara com os detalhes do cenário. Depois desse passo, a tela do *Wallman* apresenta a figura digitalizada e o usuário deve utilizar as ferramentas de desenho para incluir cada elemento do cenário. Na Figura 5.6 é

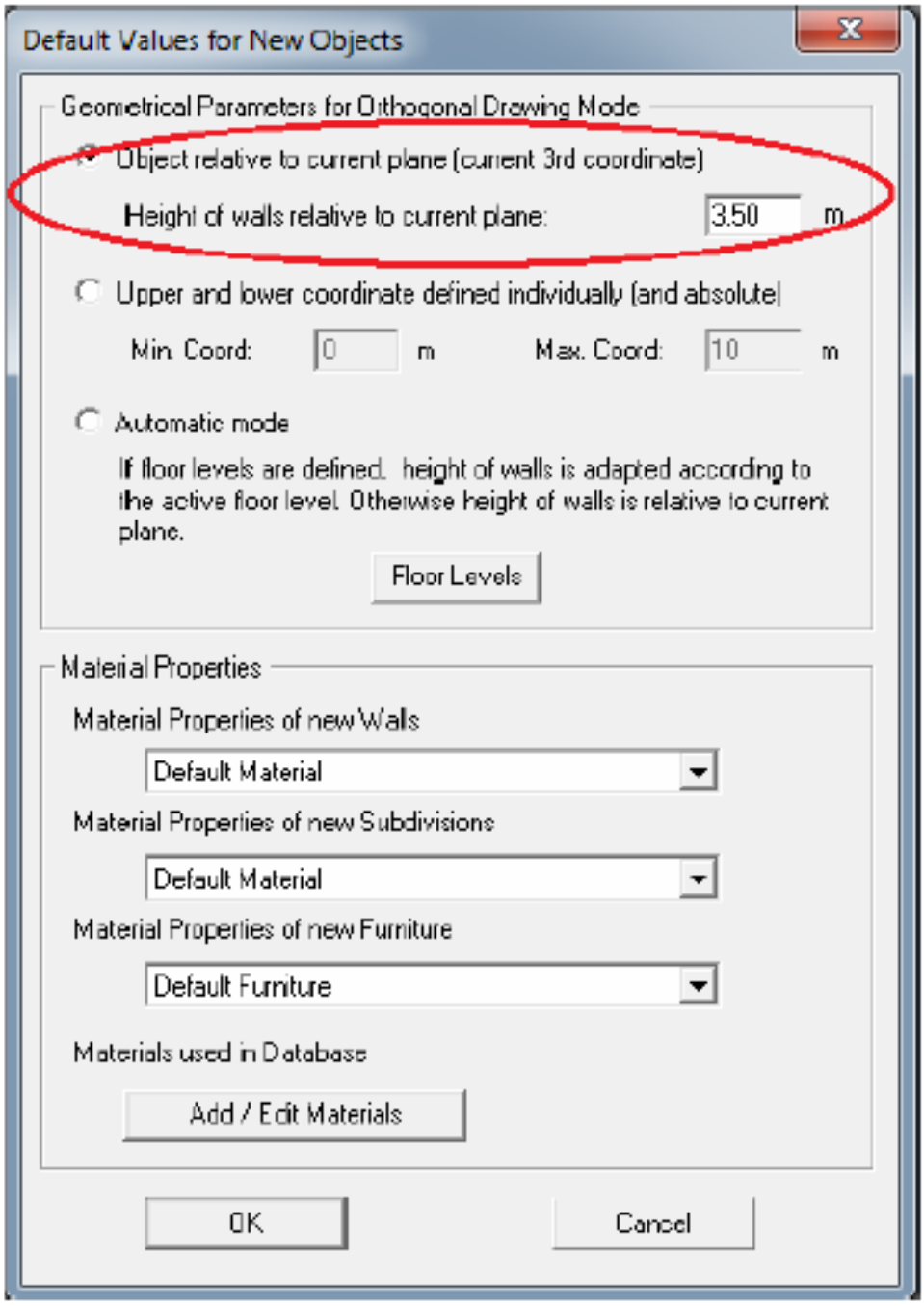


Figura 5.5 – Caixa de diálogo - Definição dos valores padrão. Fonte: [7].

mostrada a tela do *Wallman*, em que é possível a construção da base de dados sobre uma figura digitalizada do ambiente.

Propagação em Cenários *Indoor*

Prever as características da propagação entre duas antenas dentro de um edifício é essencial para o projeto sistemas de comunicação sem fio, como rede de telefonia sem fio, redes locais sem fio (WLAN), dentre outras. Além disso, muitas vezes é necessário prever a distribuição de campo no interior de uma edificação proveniente de serviços externos, como telefonia móvel. A instalação de sistemas celulares com estações radiobase internas envolve

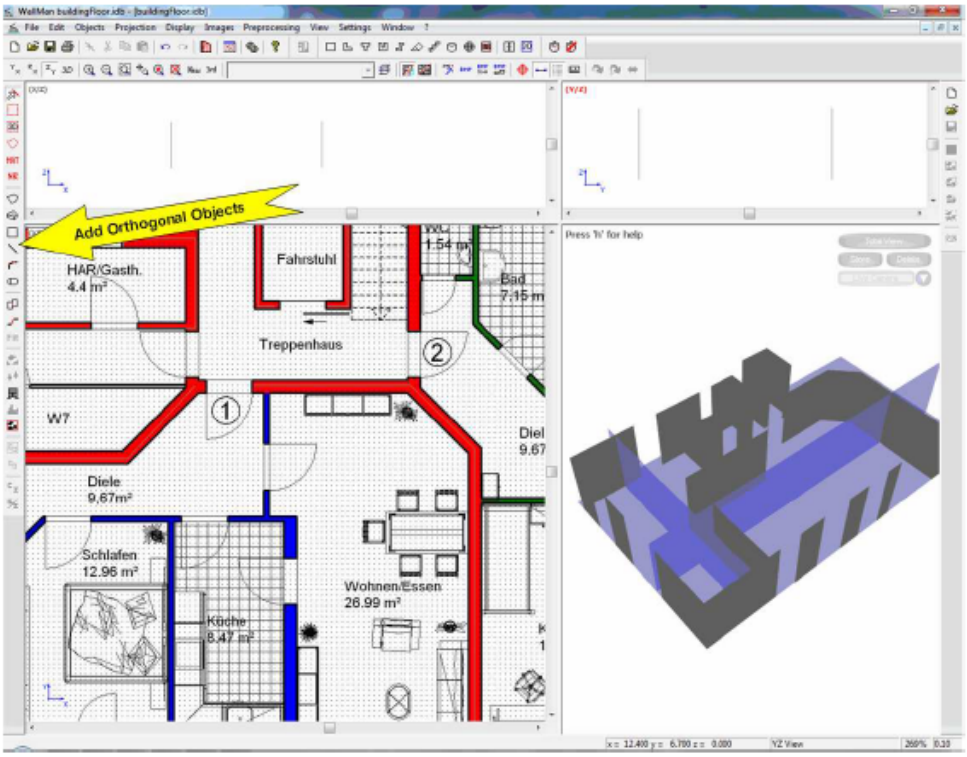


Figura 5.6 – Edição da base de dados vetorial - *Wallman*. Fonte: [7].

o uso de modelos de propagação *indoor*.

O canal de propagação interior difere consideravelmente do exterior. Em situações de propagação *indoor* a distância entre o transmissor e o receptor é menor que no exterior, mas a potência de transmissão também é menor e bastante afetada pela atenuação causada por paredes internas e pela mobília, por isso a potência do sinal recebido pode ser menor que em abientes externos. A distância curta implica em menor atraso dos ecos e, consequentemente, menor atraso de propagação. As variações temporais do canal devido ao deslocamento são mais lentas comparadas às antenas móveis instaladas em veículos. No caso de sistemas externos, existem vários parâmetros de propagação importantes a serem previstos, como a perda de percurso e as características estatísticas do envelope de sinal recebido, importantes para aplicações de planejamento de cobertura. As características de espalhamento no tempo e na frequência são essenciais para avaliação do desempenho do sistema.

Para utilizar a suíte *WinProp* na análise de propagação interior, os projetos internos são baseados em bancos de dados vetoriais do interior dos edifícios, que precisam ser modelados com o *Wallman*. Em geral, vários tipos de banco de dados pode ser usado com diversos modelos de predição. Para os modelos empíricos e semi-empíricos (*Multi-Wall*, *Motley-Keenan*, Espaço Livre Modificado e Caminho Dominante *Indoor*), o banco de dados no formato (*.idb) deve ser usado.

O *Modelo de Traçado de Raios Indoor* – IRT executa um pré-processamento sofisticado antes que o campo seja calculado. Nesse pré-processamento, todas as paredes são divididas em reticulados quadrados, com todas as arestas divididas em segmentos. Com isso, todas as relações de visibilidade entre esses elementos são calculadas. Tal informação é armazenada

em uma estrutura de árvore. Isso significa uma discretização da busca do caminho dos raios. Para a predição do campo, os caminhos individuais não precisam ser determinados. Em vez disso, a determinação do caminho é reduzida para uma busca em árvore, o que torna o cálculo muito mais rápido.

Construção de Cenários *Indoor*

Para exemplificar a utilização da suite *WinProp* na predição de campo em ambientes internos, vamos considerar um escritório, de dimensões 25 x 15 m, com paredes de tijolo e divisórias de madeira. A planta baixa e perspectiva do local é mostrada na Figura 5.7. Como referência, o *Wallman* considera o canto superior esquerdo como a coordenada $(x, y) = (0, 0)$.

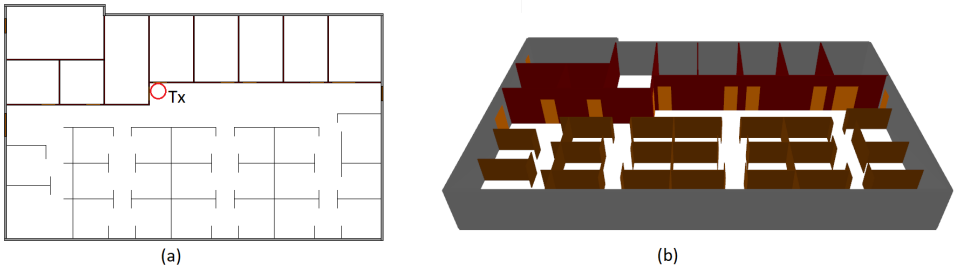


Figura 5.7 – Base de Dados para aplicação *indoor* (a) Planta Baixa, (b) Perspectiva. Fonte: [7].

Para edifícios de vários andares, é possível definir várias alturas para a predição de cobertura (e também para planejamento de redes). No caso de um elevado número de andares, não é necessário prever a cobertura de todos os transmissores em cada andar. Por exemplo, um transmissor no piso térreo não causa interferência no último andar, nem sofre interferência do piso mais alto. A fim de acelerar a simulação no caso de edifícios de múltiplos andares, é possível definir uma faixa de altura (por exemplo, 3m), de modo que a cobertura de um transmissor seja prevista apenas para alturas de predição dentro do intervalo definido (ou seja, se a diferença entre as coordenadas do transmissor e altura de predição estão dentro desse intervalo).

A base de dados foi construída com o uso do *Wallman*, a partir da importação do arquivo em formato de figura (JPG). Dessa forma, cada parede do ambiente foi modelada escolhendo os parâmetros eletromagnéticos dos materiais correspondentes à faixa de frequências definida. O modelo foi utilizado no software *Proman*, como um novo projeto, com a utilização os parâmetros relacionados na Tabela 5.1.

Air Interface
Análise de propagação (sem considerar o planejamento de redes)
Cenary
Cenário *indoor*

Tabela 5.1 – Configuração inicial do *Proman* para aplicações *indoor*.

Após o carregamento da base de dados deve ser incluido um novo *site*, em que é configurado o transmissor. É possível configurar o tipo de transmissão em função da antena a ser utilizada, com a escolha entre um *site* omnidirecional ou setorizado. Os parâmetros da transmissão podem ser inseridos no menu de configuração. Na Figura 5.8 é mostrado o menu de configuração da predição de campo. Os diversos parâmetros podem ser ajustados na aba correspondente. A configuração da célula, incluindo a localização do transmissor, altura da antena, frequência, potência e tipo da antena podem ser inseridos no menu "Célula", como mostrado na Figura 5.9.

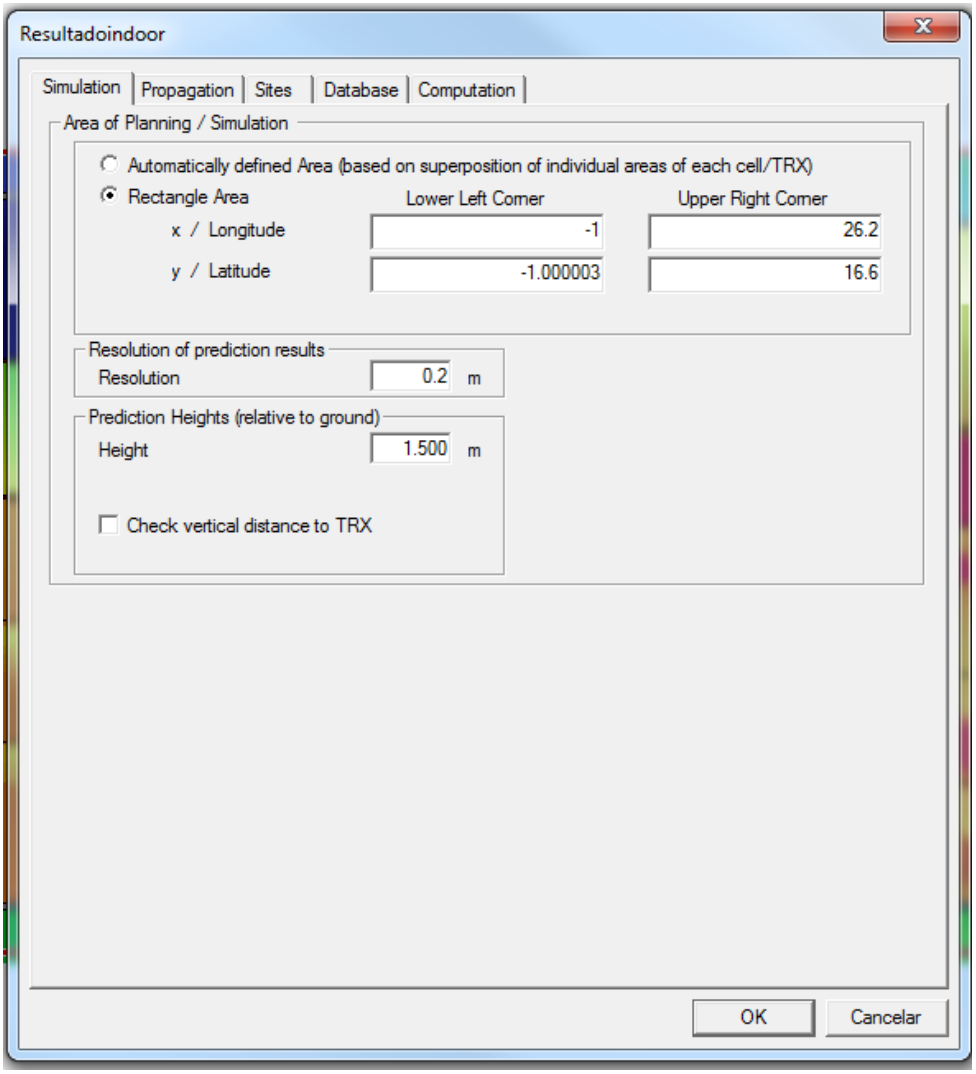


Figura 5.8 – Menu de configuração da simulação. Fonte: Elaborado pelo Autor.

Para o exemplo, o transmissor foi posicionado no local indicado na Figura 5.7 (a). Os parâmetros de configuração são mostrados na Tabela 5.2.

Para o exemplo, o modelo escolhido foi o *Caminho Dominante indoor*. A simulação foi configurada para uma resolução de 0,2 m. Isso significa que a área em estudo é dividida em um reticulado quadrado, com 20 cm de lado (tamanho do *pixel*). O modelo calcula o caminho de maior energia entre a antena transmissora e cada *pixel*, considerando a geometria e os materiais constantes no modelo. O resultado obtido para a predição de campo nesse caso é mostrado na Figura 5.10.

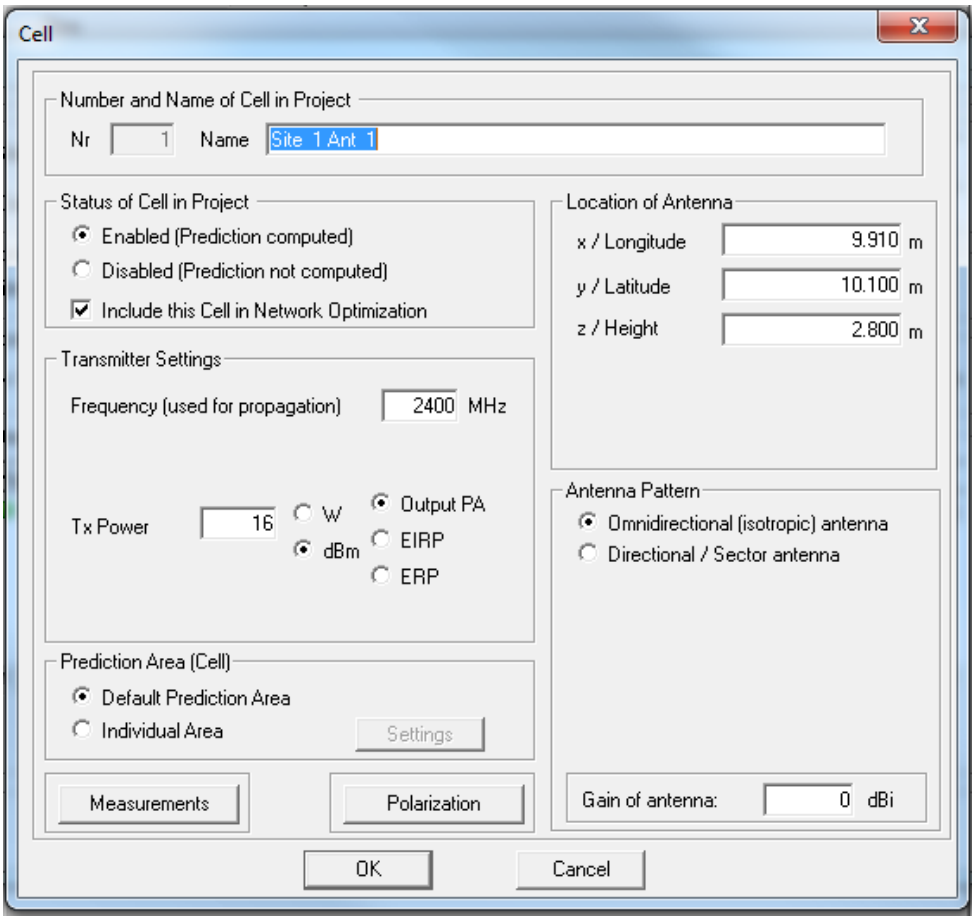


Figura 5.9 – Menu de configuração do site. Fonte: Elaborado pelo Autor.

Parâmetro	Configuração
x / longitude	9,91 m
y / latitude	10,1 m
z / altura	2,8 m
Frequência	2400 MHz
Potência	16 dBm (PA)
Antena	Onidirecional (antena isotrópica)
Ganho da antena	0 dBd

Tabela 5.2 – Configuração dos parâmetros do site.

O resultado é apresentado numa escala de cores que pode ser configurável. Também é possível algumas análises de percurso, informando as coordenadas do ponto de início e fim da trajetória ou até mesmo traçando um caminho com o auxílio do *mouse*. Para o cenário considerado, pode-se verificar se o local de instalação do transmissor fornece uma boa distribuição de campo em uma ampla área do cenário. No entanto, praticamente não tem cobertura do serviço no interior a sala situada no canto superior esquerdo da planta. O sistema permite fácil relocação do transmissor e/ou sua antena ou potência.

O software *Proman* oferece módulos de planejamento de rede para redes 2G/2,5G; 3G/HSDPA/LTE; WLAN; WiMAX e TETRA. Também é possível considerar planejamentos de rede estáticos, bem como simuladores dinâmicos de rede. Além desses recursos de

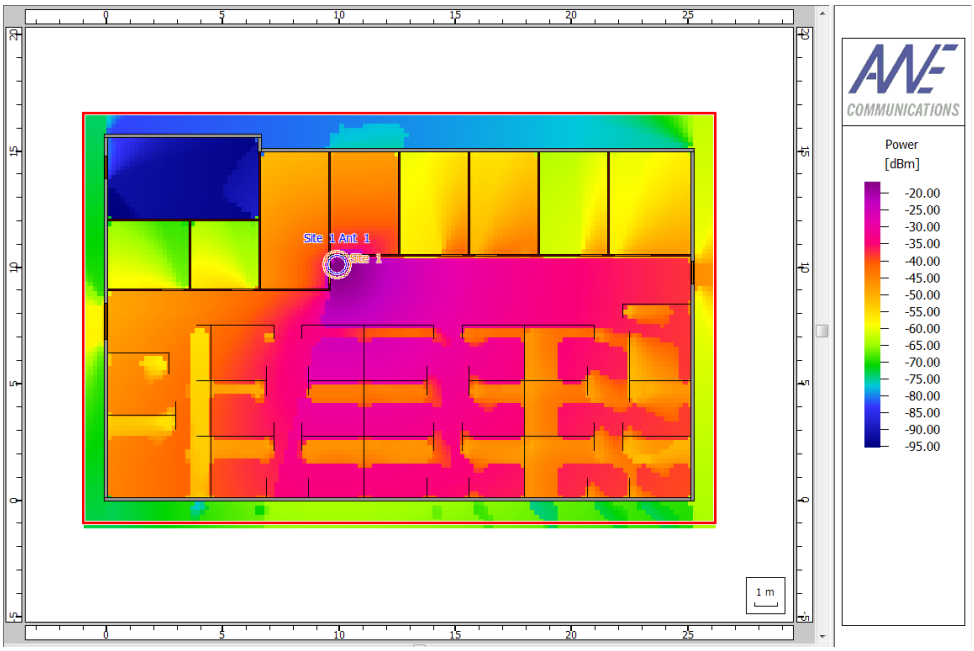


Figura 5.10 – Distribuição de campo para o cenário considerado. Fonte: Elaborado pelo Autor.

planejamento de rede, o *Proman* também suporta o planejamento de redes de transmissão (terrestre e satélite). Simulações de rede são baseadas nos resultados de propagação de onda dos transmissores dentro da rede e na definição da interface da rede sem fio. Portanto, as previsões de propagação de onda para todos os transmissores do projeto de rede são obrigatórias antes que os resultados do planejamento de rede possam ser calculados.

Como exemplo, apresenta-se a análise de desempenho de uma rede WLAN, com protocolo 802.11b. Os parâmetros da interface da aplicação são configurados na tela correspondente (*Air Interface)*. A interface foi comissionada com 13 canais, com acesso CDMA sem utilização de MIMO. Os modos de transmissão foram configurados com múltiplas modulações, sendo a máxima taxa (11 M*bit*/s) obtida para a modulação 256-QAM.

O cenário considerado é uma edificação térrea, com corredores e salas diversas. No ambiente foram usados cinco roteadores, instalados no teto da edificação, e posicionados conforme mostrado na Figura 5.11. Os roteadores foram configurados para operar em 3 canais diferentes, sendo que os canais 1 e 2 são repetidos duas vezes no ambiente. O desejável é obter a máxima taxa de tramsmissão em todo o ambiente.

Para determinar os diversos parâmetros do sistema é necessário calcular o campo proveniente de cada transmissor em cada um dos pixels do cenário. Nesse caso, foi considerada uma resolução de 1 m (tamanho do *pixel*). O *Proman* efetua os cálculos para cada transmissor separadamente. Na Figura 5.12 mostra-se a distribuição de campo proveniente do *site* 3. Também é possivel verificar a potência do campo ao longo de uma linha. Na Figura 5.13 é mostrada a distribuição da potência ao longo do corredor onde está instalado o *site* 3. Verifica-se que a máxima intensidade ocorre no ponto em que o equipamento está instalado.

O *Proman* possibilita a visualização de vários parâmetros da rede, todos obtidos a partir do calculo da intensidade de campo em cada pixel da base de dados considerada. Os

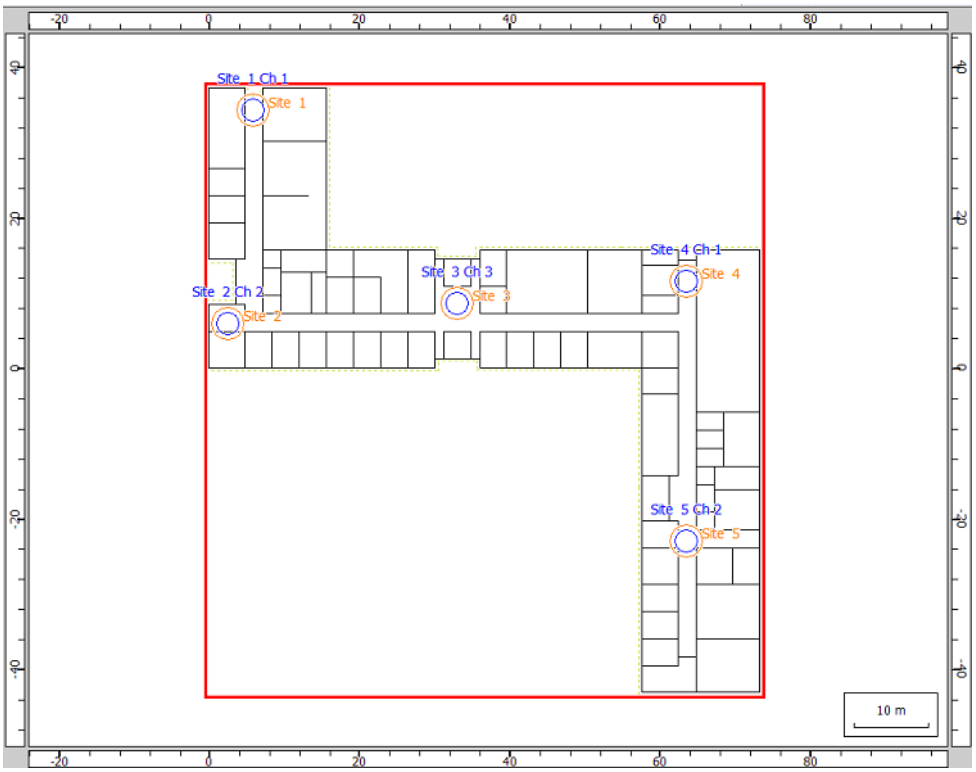


Figura 5.11 – Base de dados para o planejamento de redes. Fonte: Elaborado pelo Autor.

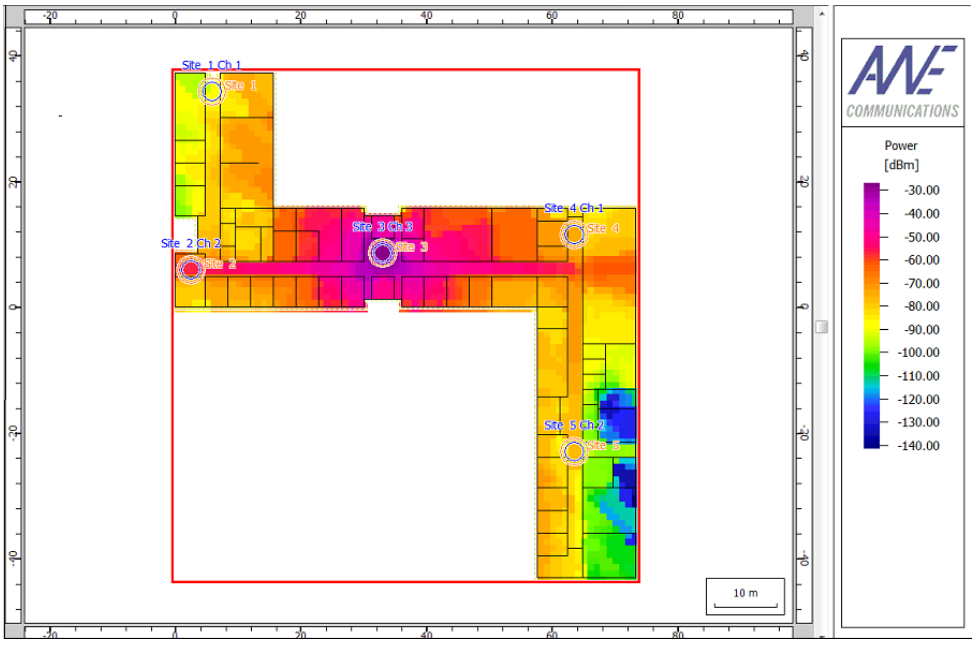


Figura 5.12 – Distribuição de campo considerando o *site* 3. Fonte: Elaborado pelo autor.

resultados possíveis incluem:

- Área da célula/*site*: indica a célula/*site* de melhor conexão para cada posição.
- LOS: Situação de desobstrução entre Tx e Rx (linha de visada).
- Mapas de multi serviços: Análise dos diversos modos de transmissão considerados para o serviço (como da taxa de transmissão)

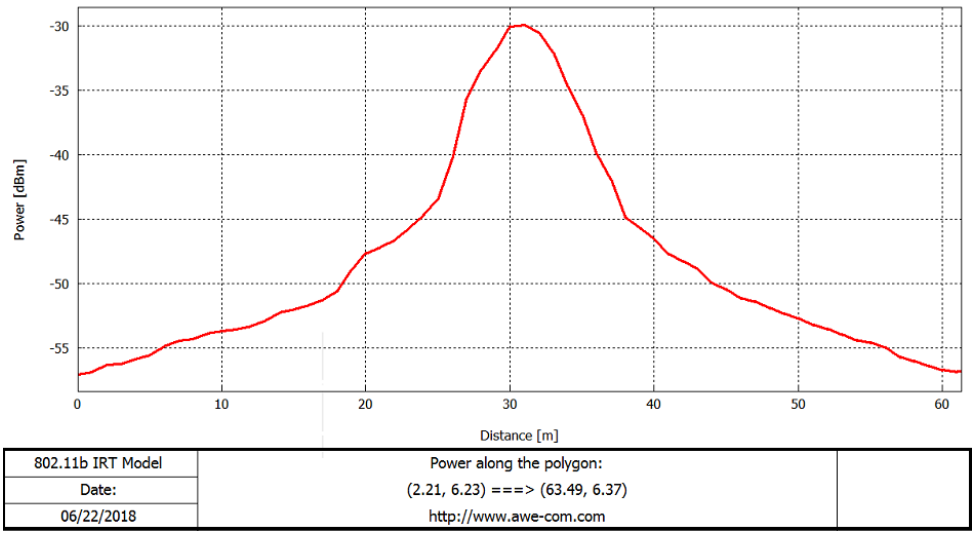


Figura 5.13 – Distribuição de campo ao longo do corredor central. Fonte: Elaborado pelo autor.

- Número de portadoras recebidas: Indica o número de portadoras recebidas em cada pixel para sistemas multi portadora.
- Caminho Dominante: Dados do calculo do caminho dominante, considerando o percurso entre o site e cada pixel.
- Potência/campo recebido: Intensidade de campo (potência) do sinal recebido.
- SNIR: Relação sinal interferência em cada pixel para redes celulares.

Os resultados sempre são mostrados como "mancha" de cobertura sobre o próprio cenário construído (base de dados), permitindo uma rápida visualização do parâmetro. Dessa forma, é possível alterar a configuração da instalação de forma a otimizar a cobertura desejada. Para a rede WLAN considerada, os *sites* foram posicionados de maneira a maximizar a taxa de transmissão em todo o cenário. Para este caso, a área de melhor *site* é mostrada na Figura 5.14. Para o exemplo foi possível maximizar a cobertura. Na Figura 5.15 é mostrado que em todo o cenário a taxa máxima de transmissão de $11Mbps$ foi obtida com a utilização de cinco *sites*, localizados nos pontos mostrados na Figura 5.11.

Propagação em Cenários Urbanos

Os modelos de predição aplicados em regiões abertas geralmente consideram a propagação direta, incluindo difração múltipla em terrenos e edifícios. A dispersão e reflexão em colinas, montanhas e edifícios geralmente são negligenciados, de forma a simplificar o modelo. As previsões baseiam-se no conhecimento da topografia do terreno, que são modeladas em bancos de dados georeferenciados.

Como regra, no caso de propagação em áreas abertas, o número de caminhos possíveis entre o transmissor e o receptor aumenta com a distância entre eles, o que afeta diretamente no desempenho de sistemas com modulação digital. Assim, para maior precisão na

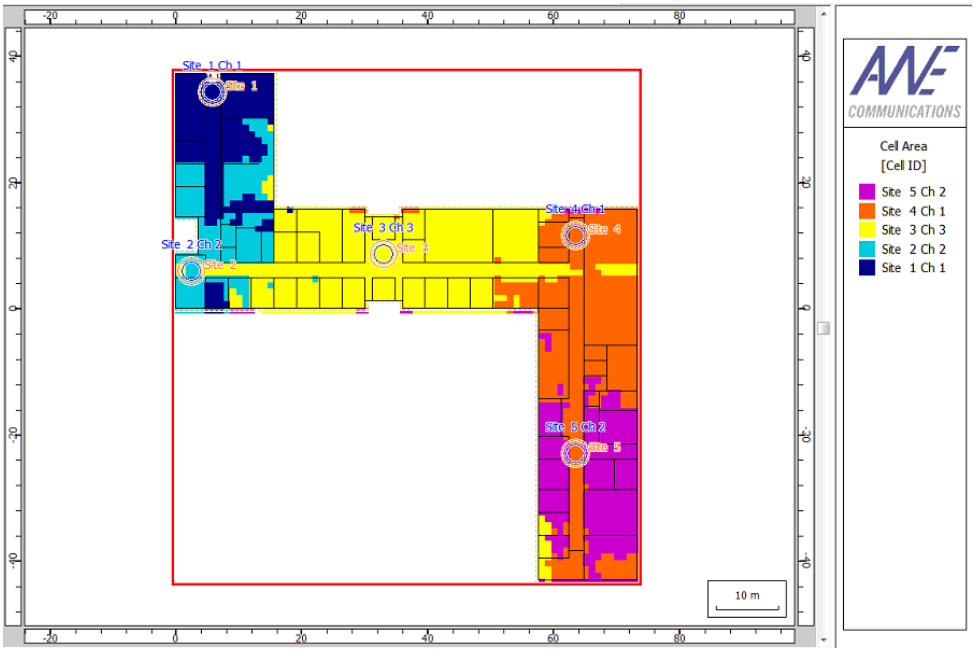


Figura 5.14 – Área de melhor *site* (*site* 3). Fonte: Elaborado pelo autor.

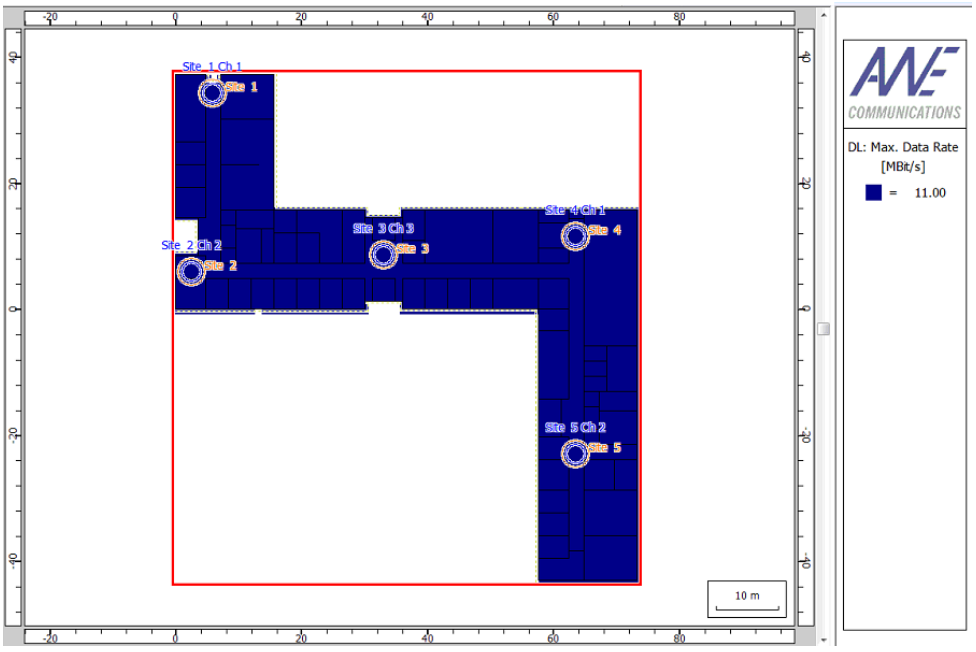


Figura 5.15 – Máxima taxa de transmissão. Fonte: Elaborado pelo autor.

.

predição de campo, abordagens determinísticas mais sofisticadas, com inclusão de múltiplas interações no espaço 3D devem ser empregadas. Modelos de propagação de caminhos múltiplos, como o *Modelo de Traçado de Raios Rural*, considerando fenômenos como reflexões múltiplas e efeitos de orientação de onda em desfiladeiros, são necessários para obter previsões precisas do nível de sinal e da resposta de impulso do canal. No entanto, esses modelos exigem bancos de dados vetoriais para a especificação do ambiente de simulação.

A propagação de ondas em cenários rurais e suburbanos é muitas vezes caracterizada pela propagação de caminhos múltiplos devido a interações da onda eletromagnética

(reflexões, difrações e espalhamento) em vários obstáculos, como colinas, edifícios e torres. A utilização dos Modelos empíricos baseados em medições para esses cenários ignoram a topografia entre transmissor e receptor (*Modelo Hata-Okumura*, *Modelo Empírico de Dois Raios*) ou consideram apenas o sombreamento devido a obstáculos no plano vertical (*Modelo Determinístico de Dois Raios*).

Abordagens mais sofisticadas incluem múltiplas difrações no plano vertical (Modelo de Difração Gume de Faca) ou calculam a onda que guia os obstáculos em 3D (*Modelo do Caminho Dominante*). Todos esses modelos de predição listados acima estão se concentrando "apenas" em um único caminho de propagação. Mas, na realidade, muitas vezes existe outros caminhos de propagação entre o transmissor e o receptor. Portanto, a superposição de todos esses caminhos leva a uma predição mais precisa. Assim, um modelo (determinístico) de propagação de caminhos múltiplos, como o *Modelo de Traçado de Raios Rural*, considerando fenômenos como reflexões múltiplas e efeitos de orientação de onda em desfiladeiros é necessário para obter predições precisas do nível de sinal e da resposta de impulso do canal.

A propagação de ondas eletromagnéticas em áreas com baixa densidade de edifícios depende principalmente da topografia. Os dados vetoriais dos edifícios não devem ser considerados em tais cenários. Para predições baseadas em bancos de dados de pixels, deve-se selecionar um banco de dados topográfico. As bases de dados topográficas da *WinProp* descrevem a elevação topográfica de todos os pixels dentro da área de interesse. Eles são matrizes de *pixels* binárias. As extensões dos bancos de dados são (*.tdb – se banco de dados único) ou (*.tdi – se bancos de dados indexados).

Construção da Base de Dados

A construção do cenário a ser utilizado no cálculo da predição de campo no *WinProp* deve incluir as informações da topografia do terreno e de todos os elementos que compõem o cenário. As informações topográficas estão disponíveis na Internet em diversos serviços cartográficos, em geral, no formato Geotiff, referenciados no sistema de coordenadas UTM. Trata-se de um arquivo do tipo tabela, com informações de cada *pixel*, sendo x, y (posição) e z (altura). O software *Wallman* proporciona a conversão para o formato (*.tdb), compatível com o *Proman*.

O *Wallman* também é empregado para a construção dos diversos elementos do cenário. Para facilitar a construção, pode-se utilizar uma imagem georeferenciada como referência. Nesse caso, um cuidado especial deve ser tomado na localização de cada elemento (edifício, vegetação, etc.), pois a planta com os elementos deve ser superposta à planta contendo as informações de relevo. Como exemplo, neste Capítulo é utilizada uma área da cidade de João Pessoa, cujas informações do relevo estão disponível no *Global Data Explorer*, do *U.S. Department of the Interior*. Para construção dos elementos da paisagem, foi utilizada uma imagem de alta resolução obtida no *Google earth* e georeferenciada no software *QuantumGis*. O cenário montado para esse caso é mostrado na Figura 5.16.

Figura 5.16 – Cenário para predição do campo em ambiente urbano. Fonte: Elaborado pelo autor.

Predição de Campo em Ambiente Urbano

O *Proman* é utilizado para calcular a distribuição do campo partir da base de dados contruída, com as informações de relevo e demais elementos do cenário. Como exemplo, uma estação do sistema de comunicações móveis LTE 4G foi inserida no cenário. Para a simulação, foi escolhido o *Método do Percurso Dominante Urbano*. O resultado da distribuição de campo é mostrado na Figura 5.17.

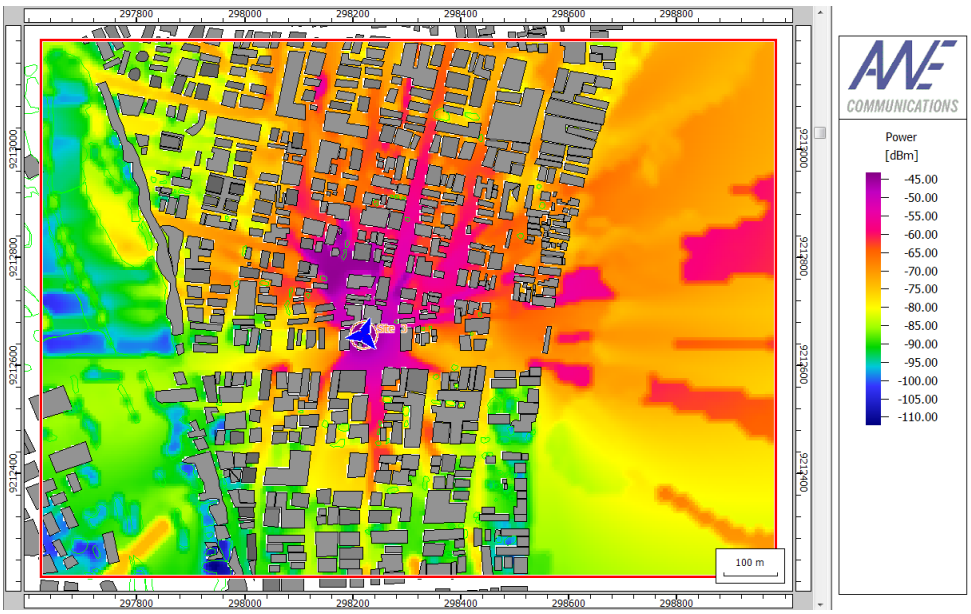


Figura 5.17 – Distribuição do campo para o cenário considerado. Fonte: Elaborado pelo autor.

A distribuição da potência eletromagnética é mostrada em escala de cores na figura. Pode-se verificar algumas regiões de baixa desidade de potência em regiões próximas ao

site, principalmente devido ao sombreamento causado pelos edifícios existentes. Isso pode ser corrigido com intervenções no sistema irradiante, como ajustes no azimute, *down-tilt* ou até mesmo na sua altura. Também é possível verificar que o nível de potência fica mais elevado nas proximidades do *site*, como esperado.

Quando se considera uma rede composta por diversos *sites*, como em sistemas de comunicação móvel celular, a distribuição de campo para cada *site* é calculada separadamente. Nesse caso, a configuração dos diversos parâmetros da interface para a rede são inseridos no *Proman*, como frequência dos canais, potência do transmissor, tipo de acesso considerado (CDMA, FDMA, OFDM...), além dos parâmetros de transmissão, que incluem o tipo de modulação, taxa de codificação, número de blocos e SNIR mínima para o modo de transmissão. Assim, para o cenário considerado, os diversos parâmetros de saída são calculados para cada *pixel*, a partir dos valores da intensidade de campo e/ou potência eletromagnética.

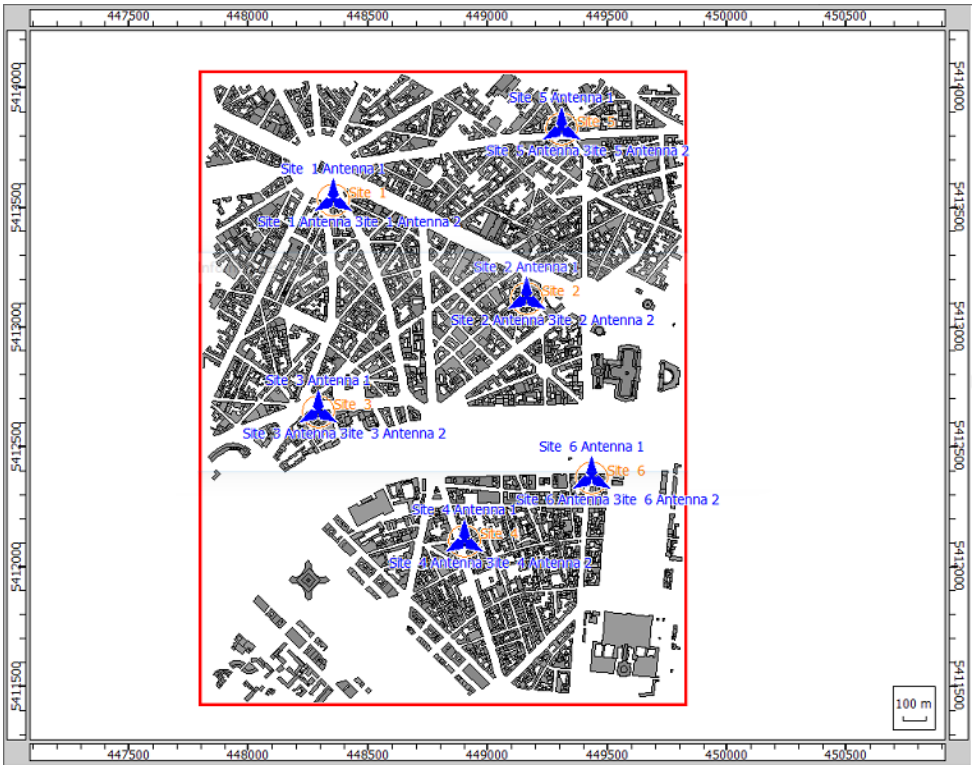


Figura 5.18 – Cenário considerado para a rede LTE-4G. Fonte [4].

Para considerar a interface aérea LTE no projeto de planejamento de rede de rádio, o arquivo (*.wst) *WinProp* LTE correspondente (wst significa “padrão sem fio”) deve ser selecionado na definição do projeto. Como exemplo, conside-se uma rede de telefonia móvel celular, operando no padrão LTE-4G na região central da cidade de Paris. Foram considerados seis *sites* setorizados, cada um com três setores. Para a simulação foi considerado que todas as células são equipadas com antenas diretivas, com 120º de abertura e ganho de 11,59 dBi, modelo 741984, fabricadas pela Kathrein, cuja configuração está disponível no *software* utilizado. O cenário considerado é mostrado na Figura 5.18. Os dados de acesso múltiplo, modos de transmissão e parâmetros do receptor são apresentados

nas Tabelas 5.3, 5.4, 5.5 e 5.6.

Acesso Múltiplo: OFDM/SOFDMA				
Potência de retorno da subportadora de controle				0 dB
Potência de retorno da subportadora de referencia				0 dB
Taxa de amostragem				384/250 LTE
Carregamento da célula				Subportadoras usadas
Reuso de frequência				Não usado
Número máximo de portadoras				1024
Guarda de subportadoras				423
Símbolos				Pilot/ref
Número de subportadoras por grupo				12
	Subportadoras	Referência	Dados	Símbolos
Total	600	-	-	14
Controle	500	-	-	1
Referência	100	-	-	3
Dados	-	-	-	10

Tabela 5.3 – Configuração do acesso múltiplo. Fonte: Elaborado pelo autor.

Separação entre uplink e downlink		120 MHz
Largura de faixa		10 MHz
Separação entre portadoras		10 MHz
Frequência da Portadora		
ID (nº canal)	Frequência DL	Frequência UL
3100	2665 MHz	2535 MHz

Tabela 5.4 – Separação Duplex e Frequências. Fonte: Elaborado pelo autor.

Potência máxima (PA)	23 dBm
Figura de ruído	6 dB
Margem de desvanecimento rápido	0 dB
Ganho da antena	1 dBi

Tabela 5.5 – Características de transmissão. Fonte: Elaborado pelo autor.

1 - 64 QAM - Bidirecional	
Downlink / Uplink	
Modulação	64-QAM
Code Rate	4/5
Recursos/Bloco	1
Overhead	11,5 %
SNIR min	17,2 dB
Potência de Retorno	3 dB
2 - 16 QAM - Bidirecional	
Downlink / Uplink	
Modulação	16-QAM
Code Rate	4/5
Recursos/Bloco	1
Overhead	11,5 %
SNIR min	11,8 dB
Potência de Retorno	3 dB
3 - QPSK - Bidirecional	
Downlink / Uplink	
Modulação	QPSK
Code Rate	4/5
Recursos/Bloco	1
Overhead	11,5 %
SNIR min	5,6 dB
Potência de Retorno	0 dB

Tabela 5.6 – Modos de transmissão. Fonte: Elaborado pelo autor.

As propriedades para *downlink* e *uplink* são definidas individualmente, porém, por padrão, de forma simétrica (com relação aos parâmetros de transmissão, *back-off* de energia, SNIR e potência de transmissão). Além disso, é possível alternar do modo bidirecional para a análise individual somente do *downlink* ou somente do *uplink*. O primeiro bloco define os parâmetros para a transmissão de dados com modulação, taxa de código, número de blocos de recursos e razão de *overhead*. Esses parâmetros resultam em uma taxa de dados viável para esse modo de transmissão. O número de blocos de recursos é definido de acordo com a largura de banda (25 para 5 MHz, 50 para 10 MHz e 100 para 20 MHz). O back-off de potência Tx deve ser definido em relação à modulação utilizada. No caso de modulação QAM, é recomendado um back-off de 3 dB. O retorno de energia transmitida reduz a SNIR para o modo de transmissão correspondente, o que pode levar à situação em que esse modo de transmissão não está mais disponível.

O desempenho da rede LTE (em termos de taxa de transferência possível) é derivado do mapa de SNIR calculado. Além da potência do sinal disponível, o impacto mais significativo é dado devido à interferência proveniente das células vizinhas, pois dentro de uma célula os diferentes usuários são separados no domínio da freqüência e/ou do tempo.

O nível de interferência pode ser influenciado pela definição de um fator de carga correspondente na página de simulação. Este valor representa a energia Tx média assumida

no *downlink* para as células vizinhas e é definida em relação ao valor máximo. Um valor de 100% significa que todas as células vizinhas transmitem a potência total (ou seja, 100% de carga), o que representa claramente o pior caso para o rendimento disponível. Ao contrário, um valor de 0% de desempenho.

Os resultados para a simulação de redes LTE no *Proman* podem ser enquadrados em três categorias:

- Considerando a atribuição da célula e todos os modos de transmissão
  - Área da célula
  - Área do *site*
  - Melhor servidor
  - Maxima Taxa de Dados (DL e UL)
  - Portadora de serviço: Interferência e Potência de Ruído (DL)
  - Portadora de serviço: Sinal e Interferência e Potência de Ruído (DL)
- Considerando a atribuição da célula
  - Potência recebida
  - *Site* recebido
  - Célula recebida
  - Portadora recebida
  - SNIR Máxima (DL)
- Para cada modo de transmissão
  - DL: Minima potência de transmissão da estação de base
  - DL: Máxima potência de recepção da estação móvel
  - DL: Máxima SNIR
  - DL: Probabilidade de recepção
  - UL: Minima potência de transmissão da estação móvel
  - UL: Máxima potência de recepção da estação de base
  - UL: Probabilidade de recepção.

Assim, os resultados consideram duas configurações diferentes no lado de transmissão. A primeira quando se utiliza a mínima energia necessária no transmissor para atingir o nível de recepção (ou seja, considerando o controle de potência), para os mapas “Min Tx Power BS” e “Min Tx Power MS”, a segunda considera a transmissão de potência total (ou seja, sem controle de potência) para todos os outros mapas.

Na simulação realizada não foi considerado o controle de potência em nenhuma das células incluidas. Os elementos do cenário foram construídos no *Wallman* e uma planta vetorizada adicional foi considerada. Todos os arquivos contendo o cenário foram disponibilizados pelo fabricante. O primeiro resultado mostra a distribuição do campo em todo o ambiente, considerando apenas o setor 2 do *site* 1, Figura 5.19.

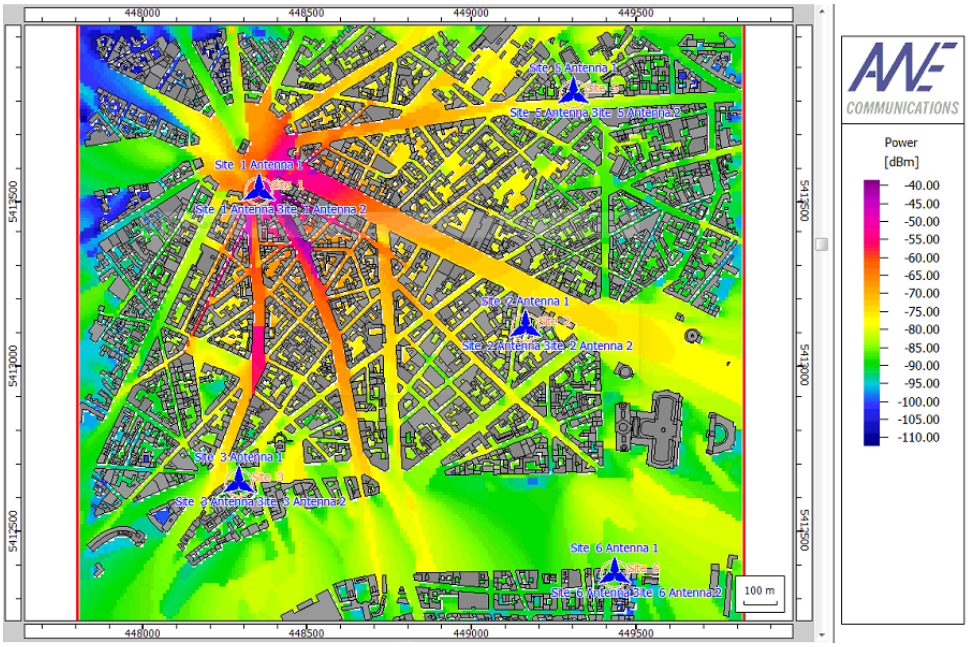


Figura 5.19 – Distribuição de campo irradiado pela antena 2 do *site* 1. Fonte: [4].

Como são utilizadas antenas diretivas, pode ser verificado uma maior intensidade de campo na região de abertura da antena. Um outro detalhe pode ser verificado ao se ampliar a figura nas proximidades da antena. O modelo do Caminho Dominante permite identificar o sombreamento produzido por alguns edifícios, como mostrado na Figura 5.20, que revela o detalhe do sombreamento produzido por uma das edificações no sinal produzido pela antena do setor 2 do *site* 1.

Após calcular o campo de cada antena em todo o ambiente, os valores da SNIR são calculados em cada elemento do reticulado (de lado l). A partir dessas informações e, de acordo com os modos de transmissão considerados e a configuração do receptor, é possível obter todos os resultados possíveis, relacionados anteriormente. Na Figura 5.21 é mostrada a área de melhor site para o exemplo, considerando os parâmetros de instalação citados anteriormente.

O resultado mostra a necessidade de alterações nos parâmetros do sistema simulado, que pode ser altura ou posição das antenas, ajuste no azimute de cada setor ou outras configurações nos modos de transmissão. Um resultado esperado é que a taxa de transmissão varie com a distância de cada antena. O resultado para a taxa de transmissão para o *downlink* é mostrado na Figura 5.22.

Outro resultado possível se refere à avaliação da cobertura na região de interesse a partir da Função da Distribuição Cumulativa (CDF), expressa por (5.12).

Figura 5.20 – Sombreamento causado por edificação. Fonte: Elaborado pelo Autor.

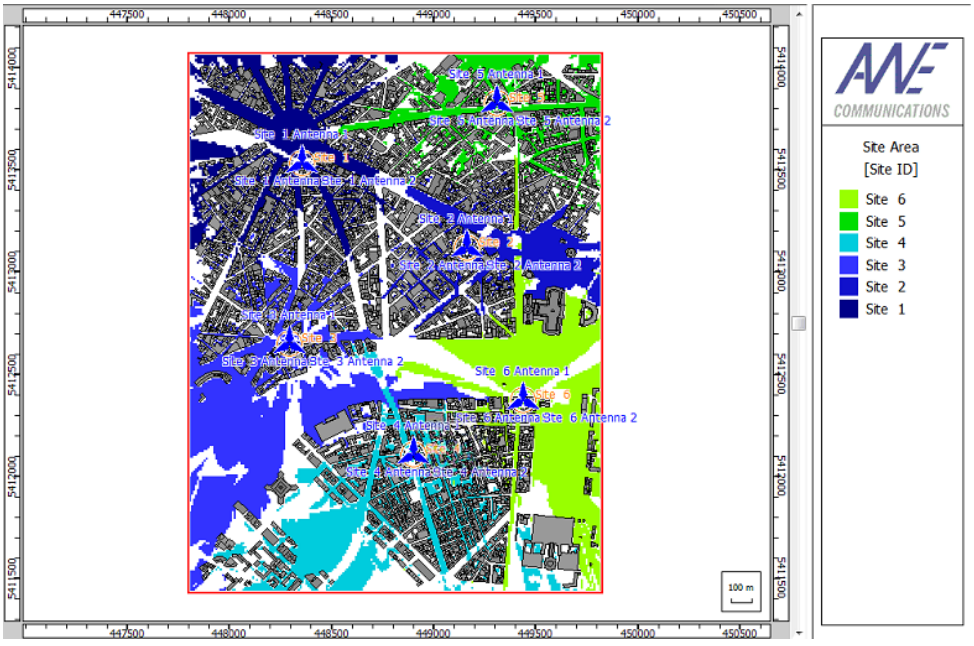


Figura 5.21 – Área de melhor *site*. Fonte: Elaborado pelo Autor.

$$\int_{-\infty}^{x} p_x(x)\,dx. \tag{5.12}$$

Dessa forma, a CDF é definida como a probabilidade de que um certo nível de sinal não seja recebido. A porcentagem de interrupção da transmissão é dada pela Expressão (5.13).

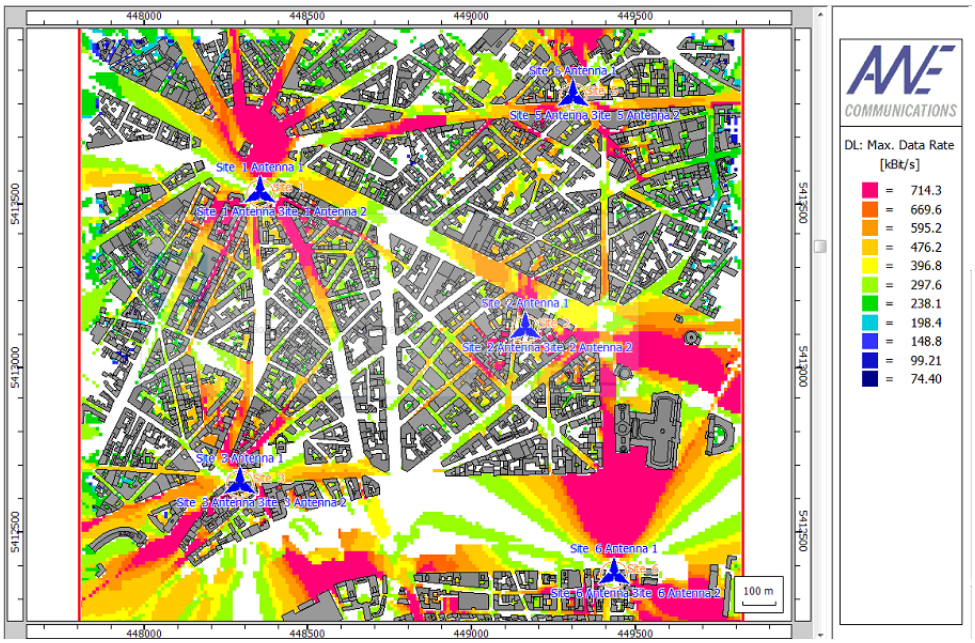


Figura 5.22 – Taxa de transmissão DL para todos os modos de transmissão. Fonte: Elaborado pelo Autor.

$$P_{bloqueio} = P_X(x_{min}). \tag{5.13}$$

Para a região em estudo, a CDF define a probabilidade do sinal recebido pela estação estar abaixo de um determinado nível. Portanto, pode-se comparar o valor obtido para a probabilidade de bloqueio e comparar com o grau de serviço (GoP) especificado para a rede. No caso em tela, o limiar de recepção da estação móvel é de -90 dBm. Na Figura 5.23 é mostrada a probabilidade acumulada da potência recebida pelo móvel. Podemos afirmar que não existe nenhum local na região estudada onde o nível do sinal recebido pelo móvel esteja abaixo da potência mínima ($P_X(x_{-90dBm} = 0)$).

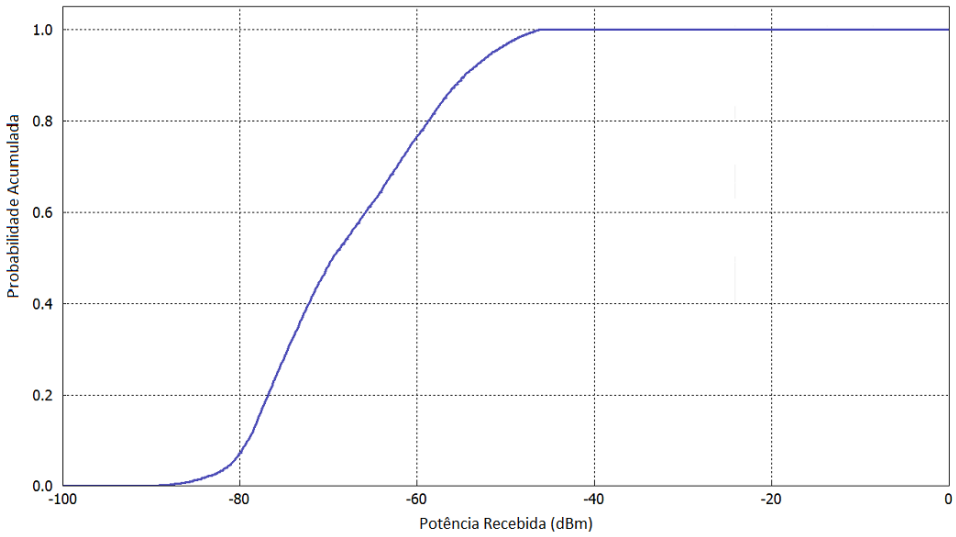


Figura 5.23 – CDF para a potência recebida pela EM. Fonte: Elaborado pelo Autor.

Considerações Finais

O dimensionamento de um sistema de comunicações sem fio envolve muitas variáveis, como o tipo de serviço, tecnologia e topologia do sistema, densidade de usuário, topografia e/ou obstrução do ambiente, dentre outras. O principal parâmetro a ser dimensionado se refere à quantidade, localização e posicionamento das antenas para prover uma cobertura adequada. Nesse sentido, o uso de ferramentas de predição de campo é fundamental para o dimensionamento adequado do sistema, em que os ajustes podem ser planejados de forma adequada com economia de recursos e tempo.

Neste Capítulo, foi apresentada a Suíte *WinProp*, que é um conjunto de ferramentas computacionais utilizadas para o dimensionamento de sistemas de comunicação sem fio. O *WinProp* disponibiliza softwares para construção de plantas digitais do local, com a possibilidade de edição e importação de outros formatos de arquivos gráficos, construção dos modelos de antenas e o ambiente de simulação. A escolha do método de predição do campo deve ser adequada aos parâmetros de transmissão e a possibilidade do planejamento de rede aumenta a potencialidade de uso da ferramenta.

Foram apresentados os principais modelos de predição de campo utilizados no planejamento de sistemas de comunicação sem fio, assim como o ambiente de edição dos cenários (*Wallman*) e de simulação (*Proman*). Como exemplo, foi apresentado o uso do *WinProp* no calculo da predição de campo em um ambiente interno e também em uma área aberta. Por fim, aspectos sobre o planejamento de redes foram apresentados para uma rede de comunicação celular no padrão LTE-4G.

A utilização de ferramentas computacionais para predição de campo e projeto de redes de comunicações sem fio é fundamental para o planejamento de novas redes e/ou ajustes de redes já existentes. Dessa forma, é essencial para a formação de Engenheiros de Telecomunicações, bem como para os profissionais da área de RF que atuam nas empresas da área. Deve ser incentivado também o desenvolvimento de modelos e simuladores baseados em cenários locais, aumentando a precisão dos resultados e tornando o planejamento mais eficiênte.

Referências Bibliográficas

[1] Carvalho Andrei P Carvalho Larissa P and Carvalho Joabson N. Estudo da cobertura de tv digital na cidade de joão pessoa. In *17 Congresso Brasileiro de Micro-ondas e Optoeletronica - MOMAG*, 2016.

[2] Theodore S Rappaport et al. *Wireless communications: principles and practice*, volume 2. 1996.

[3] André Mendes Cavalcante. Estratégias computacionais aplicadas em técnicas de traçado

de raios 3d para o aumento da eficiência na caracterização de canais de propagação de redes sem fio. *Belém, Agosto de*, 2007.

[4] René Wahl, Gerd Wölfle, Philipp Wertz, Pascal Wildbolz, and Friedlich Landstorfer. Dominant path prediction model for urban scenarios. *14th IST Mobile and Wireless Communications Summit, Dresden (Germany)*, 2005.

[5] G Wolfle and Friedrich M Landstorfer. Dominant paths for the field strength prediction. In *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE*, volume 1, pages 552–556. IEEE, 1998.

[6] R Wahl. An introduction to indoor dominant path prediction model. *AWE Communications GmbH*, 2005.

[7] Petar S Međedović and Darko S Šuka. Softwares for urban electromagnetic wave propagation modeling. *Infoteh-Jahorina*, 11:422–427, 2012.

CAPÍTULO

6 Do HEVC para o VVC: Principais Melhorias e Oportunidades de Pesquisa da Próxima Geração de Padrões de Codificação de Vídeo

Jean Felipe Fonseca de Oliveira (Samsung) e José Raimundo Barbosa (IFPB)

Introdução

Segundo o relatório *Visual Networking Index* sobre tráfego de dados na Internet, realizado pela CISCO [1], cerca de 80% do trafego em 2020 será destinando para transmissão de vídeos. Entre os principais fatores contribuintes estão o aumento de acesso e uso de serviços de *streaming*, como Youtube e Netflix, e o aumento da demanda de largura de banda provocada pelo aumento da qualidade e experiências proporcionadas pelas novas tecnologias dos vídeos digitais, tais como vídeos em *Ultra Higth Definition* (UHD) e 360° [1] [2].

As experiências providas pelas novas tecnologias de vídeos demandam uma quantidade desafiadora de recursos de processamento, armazenamento e transmissão de dados. Por exemplo, a quantidade de *pixels* nos vídeos UHD leva a uma melhoria significativa na percepção dos detalhes da imagem, permitindo novas experiências para os espectadores, mas também representam desafios para a maioria dos dispositivos do mercado atual [2]. No caso da definição UHD 4K, um vídeo possui resolução de 4096x2160 *pixels*, enquanto o *Full HD* possui resolução 1920x1080. Considerando uma sequência de vídeo UHD 4K, sem compressão, com 24 bits por pixel e 24 quadros por segundo, seria necessária uma taxa de bits de 5 Gbit/s para transmissão do vídeo em tempo real [3]. A demanda de largura de banda pode aumentar consideravelmente quando levadas em conta outras tecnologias como vídeos UHD 8K, vídeo 360° omnidirecional, Vídeos 3D, vídeos com grandes taxas de quadro etc.

Para atender à demanda de recursos e prover suporte para as novas tecnologias de vídeos

digitais, é necessário o desenvolvimento de codificadores cada vez mais eficientes e robustos. Estes codificadores são desenvolvidos a partir de uma padrão, ou seja, uma documentação que descreve a estrutura e comportamento de um codificador. Essas descrições possibilitam que várias iniciativas possam trabalhar em conjunto no desenvolvimento de módulos independentes, que resultam no conjunto de ferramentas necessárias para o funcionamento adequado do codificador [4].

Atualmente há duas principais linhas de padrões de codificadores no mercado: O padrão H.26X é um dos mais importantes no segmento, criado e mantido pela *International Standards Organization* (ISO) e *International Telecommunications Union* (ITU); E o padrão VPX, desenvolvido pela Google e atualmente mantido pela a *Alliance for Open Media*, responsável pela versão AV1, sucessora da versão VP9 e principal competidor do H.265. A Figura 6.1 ilustra a linha de tempo dos principais padrões de codificadores estabelecidos no mercado.

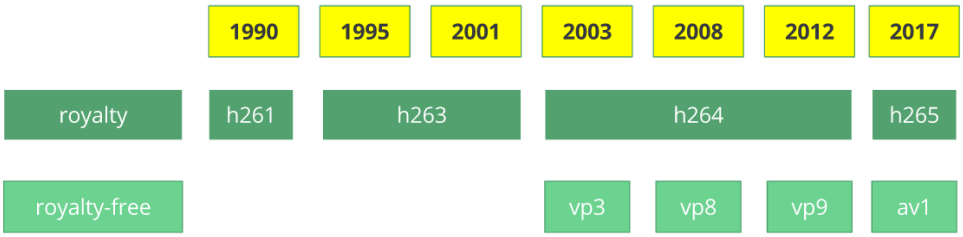


Figura 6.1 – Linha cronológica dos principais codificadores do mercado [5].

Neste capítulo são analisadas as principais características do codificador *High Efficiency Video Coding* (HEVC), um dos principais codificadores consolidados no mercado atual. Também são apresentadas as características e perspectivas propostas para a nova geração de codificadores. Para isso, este trabalho será organizado da seguinte maneira: na primeira sessão são abordadas as principais tenologias de vídeos digitais atuais, o que permite identificar os desafios a serem superados pelas próximas gerações de codificadores. Na sessão seguinte, são apresentas algumas das métricas objetivas para avaliação de qualidade de vídeo mais presentes na literatura. As novas tecnologias de vídeos digitais também representam desafios para os mecanismos de avaliação de qualidade de imagem e eficiência de compressão, sendo necessário constante aprimoramento ou desenvolvimento de novas abordagens. Em seguida são apresentados o padrão de codificador HEVC e a proposta para a nova geração de codificadores denominada *Versatile Video Coding* (VVC), apresentada numa breve comparação entre os dois padrões.

Vídeos Digitais e Novos Desafios

Nas ultimas décadas, com o avanço da tecnologia, os vídeos digitais apresentaram mudanças significativas em relação à perspectiva convencional de uma simples sequência de imagens. O aumento da qualidade dos vídeos digitais e a variedade de aplicações

atualmente possíveis, possibilitam novas experiências para os usuários, das quais muitas ainda representam grandes desafios, como discutido nas seções a seguir.

Vídeos *Ultra High Definition* (UHD)

Os vídeos com resolução UHD possuem uma matriz de *pixeis* consideravelmente maior que os vídeos convencionais *Full HD*, como é ilustrado na Figura 6.2, o que proporciona uma alta riqueza de detalhes da imagem. Com este padrão de qualidade, é possível obter novas experiências visuais e possibilidades para aplicações como, por exemplo, cinema digital e a tele-medicina [6].

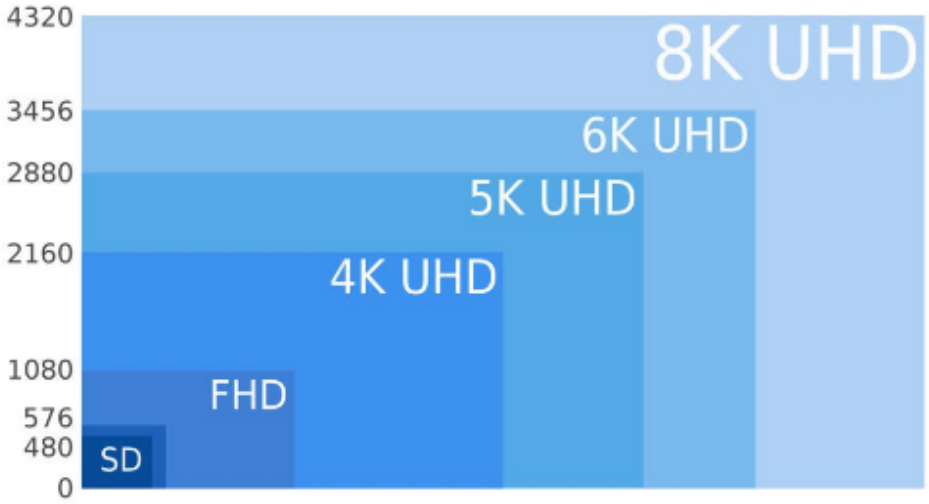


Figura 6.2 – Variações de definições de vídeo

Além do aumento da matriz de *pixeis*, outras alterações em relação às características destes vídeos também estão se tornando cada vez mais comuns, como o aumento da taxa de quadros, em alguns casos podendo chegar até 100 *frames per second* (fps), e também o aumento da quantidade de *bits* por componentes, podendo chegar em até 16 *bits* por componente.

Para trabalhar com este tipo de vídeo também existem desafios devido à alta demanda de processamento, armazenamento e largura de banda para transmissão [7]. Porém, a produção de conteúdo UHD, especialmente 4K está crescendo rapidamente. Isto é atribuído ao aumento do número de dispositivos com suporte para captura e reprodução deste tipo de vídeo [8].

Vídeo *High Dynamic Range* (HDR)

Os vídeos com HDR possuem características mais extensas do que os vídeos convencionais, o que viabiliza um melhoramento nos efeitos de luminosidade das imagens, como ilustrado na Figura 6.3.

Figura 6.3 – Comparação de vídeos sem e com HDR [9].

Este resultado é obtido a partir do intervalo da taxa de profundidade de cor da imagem, de 8-bit e 16-bit, chamado de alcance dinâmico. À medida que o valor aumenta é possível agrupar mais informações sobre a coloração da imagem. Vídeos HDR utilizam essa característica para determinar com mais fidelidade a representação da cor do mundo real no vídeo digital [10].

Vídeo 360º Omnidirecional

Vídeo 360° é uma tecnologia na qual é apresentada para o usuário uma visualização de 360° do ambiente em que o vídeo foi gravado, proporcionando a sensação de imersão, uma vez que tal proposta se diferencia do convencional em que o vídeo é apresentado em um quadrado fixo, como uma TV ou tela de projeção.

Este tipo de vídeo é obtido por meio de computação gráfica, principalmente no desenvolvimento de jogos, ou por meio de câmeras especiais que captam vários ângulos simultaneamente, como é ilustrado na Figura 6.4.

Este tipo de vídeo apresenta diversos desafios em relação à compressão do *stream*. Com o aumento de ângulos capturados para proporcionar o efeito de 360°, ocorre um aumento significativo na demanda de recursos necessários para processar, armazenar e transmitir o *stream* de vídeo, sendo necessárias técnicas de compressão mais eficientes e apropriadas para manipular tais vídeos. Atualmente os principais codificadores e métricas presentes no mercado manipulam os vídeos 360° por meio de uma representação 2D, como é ilustrado na Figura 6.5. Esse processo pode resultar em perdas de informações da imagem, uma vez que é necessário remontar ou particionar a imagem em um plano diferente do original [12].

Figura 6.4 – Câmera Samsung Gear 360 Spherical, para captura de vídeos 360º [11].

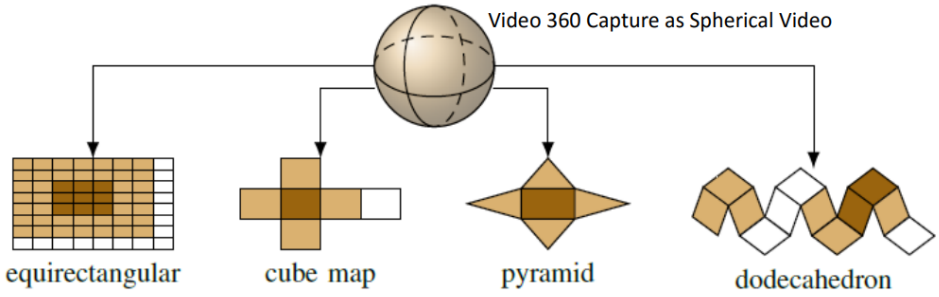


Figura 6.5 – Representação 2D de um stream 360 [13]

Métricas Objetivas para Avaliação de Qualidade de Vídeos Digitais

A avaliação de qualidade dos vídeos pode ser realizada de forma subjetiva, isto é, pela avaliação humana [14]. No entanto, esse tipo de avaliação pode ser ineficaz na detecção de pequenos erros de segmentações da imagem, que geralmente são difíceis de serem percebidos pelo olho humano e podem ser exaustivos em aplicações longas ou no caso

de avaliação em tempo real [15][16]. Dessa forma, as técnicas objetivas de avaliação de vídeo funcionam como um bom *trade-off*, porque são mais rápidas e mais baratas, e podem indicar com eficiência a existência de pequenas degradações [16][17]. Nas seções a seguir são descritas algumas métricas objetivas comumente utilizadas para realizar a avaliação de qualidade de vídeos digitais.

Métrica PSNR

A Relação de sinal/ruído de pico (PSNR) é uma das métricas mais utilizadas devido à facilidade do cálculo e o baixo custo computacional [18]. Este método expressa a relação máxima de um sinal e o ruído desse sinal (*Mean Squared Error* - MSE) e é constantemente usado para medir a qualidade estrutural das imagens. Para isso, é necessário formular o erro quadrático médio para a representação digital de imagens monocromáticas I e K de tamanho mxn. O MSE é definido por

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2. \tag{6.1}$$

Desta forma, o valor PSNR é representado por

$$PSNR = 10 \cdot log_{10}(\frac{MAX_I^2}{MSE}) = 20 \cdot log_{10}(\frac{MAX_I}{\sqrt{MSE}}), \tag{6.2}$$

em que MAX indica o valor de pixel máximo possível em uma imagem.

A Equação 6.2 resulta em um valor entre 0 e ∞. Na literatura, um intervalo entre 0 e 50 é normalmente utilizado para limitar a representação da faixa de valores da PSNR, sendo 0 o pior caso e 50 o melhor.

Métrica SSIM

O índice de semelhança estrutural (SSIM) é um modelo consolidado e constantemente usado para comparação de qualidade de vídeos. Baseia-se no pressuposto de que o sistema visual humano é altamente adaptado para extrair informações estruturais de imagens [19]. Portanto, cada pixel tem uma forte dependência dos outros e essa dependência aumenta com a proximidade. Esta dependência apresenta uma informação importante sobre a estrutura dos objetos na imagem e quantifica a mudança estrutural de uma imagem como uma boa aproximação à qualidade percebida [20].

Para identificar as informações estruturais, a métrica SSIM usa uma abordagem estatística baseada na luminância média e no contraste n de bloco da imagem. A média (μ), o desvio padrão (σ^2) e a covariância (σ_{fg}) são calculados para cada bloco. A média e o desvio padrão são estimativas aproximadas da luminância e contraste da imagem, respectivamente. A covariância é a medida de quanto um sinal é diferente do outro. O SSIM usa a medida da distorção estrutural em vez do próprio erro. Se $x = \{xi|i = 1, 2, ..., N\}$ representa o sinal original e $y = \{yi|i = 1, 2, ..., N\}$ representa o sinal distorcido, em que i é

o valor do índice de pixels, o índice de similaridade estrutural pode ser calculado de acordo com [21][20]

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{6.3}$$

Na Equação 6.3, μ_x representa a média de x e μ_y representa a média de y. A variância de x é representada por σ_x^2 e a variância de y por σ_y^2. O $c_1 = (k_1 L)^2$ e o $c_2 = (k_2 L)^2$ são as variáveis para estabilizar a divisão, $L = 2^n - 1$ é o intervalo dinâmico de bits e os valores $k_1 = 0,01$ e $k_2 = 0,03$ são definidos por padrão.

Métrica PW-SSIM

O índice de semelhança estrutural de ponderação perceptual (PW-SSIM) usa uma abordagem baseada em ponderação para avaliar a qualidade da imagem, dando mais importância às regiões visualmente mais importantes[22]. Para isso, a magnitude dos vetores de gradiente do vídeo original é calculada usando máscaras de Sobel [23], então é gerada uma moldura em que os valores do pixel são as magnitudes dos gradientes. Então, esse quadro é dividido em blocos de 8×8 *pixels* e para cada bloco, a Informação Perceptual Espacial (SI) é calculada [24]. O SI é expresso por

$$SI = \left(\frac{1}{N-1} \sum_{i=1}^{1} (\mu_S - S)^2 \right)^{\frac{1}{2}}, \tag{6.4}$$

em que N é o número de *pixels* por bloco e μ_S representa a magnitude média do gradiente em um bloco. Finalmente, o PWSSIM é calculado usando

$$PWSSIM(f,h) = \frac{\sum_{d=1}^{D} SSIM_d(f,h) \cdot SI_d}{\sum_{d=1}^{D} SI_d}, \tag{6.5}$$

em que D é o número de blocos, f é a representação de um vídeo 2D e h representa um vídeo 2D degradado.

Métrica Bjontegaard

A métrica de Bjontegaard é considerada o estado da arte no que se refere à avaliação de mecanismos de codificação de vídeo e é uma das métricas mais difundidas na literatura atual [25][26]. Esta métrica permite obter os valores da taxa de bits global (Taxa BD) em porcentagem para dois algoritmos de codificação diferentes, considerando o mesmo PSNR ou a diferença geral de qualidade de vídeo (BD-PSNR) em decibéis (dB), entre dois codificadores, considerando a mesma taxa de bit.

Em [26], a Taxa BD é exemplificada por meio da Figura 6.6, onde as curvas são extraídas por interpolação linear. Para os cálculos da Taxa BD, o eixo da taxa é colocado em escala logarítmica e as curvas de distorção de taxa são interpoladas. As porções sobrepostas das

curvas de distorção de taxa são usadas para o cálculo dos valores de BD. Dependendo do alcance de sobreposição das duas curvas, as medidas de BD resultantes são mais ou menos significativas. Portanto, é necessária uma interpretação cuidadosa dos números resultantes.

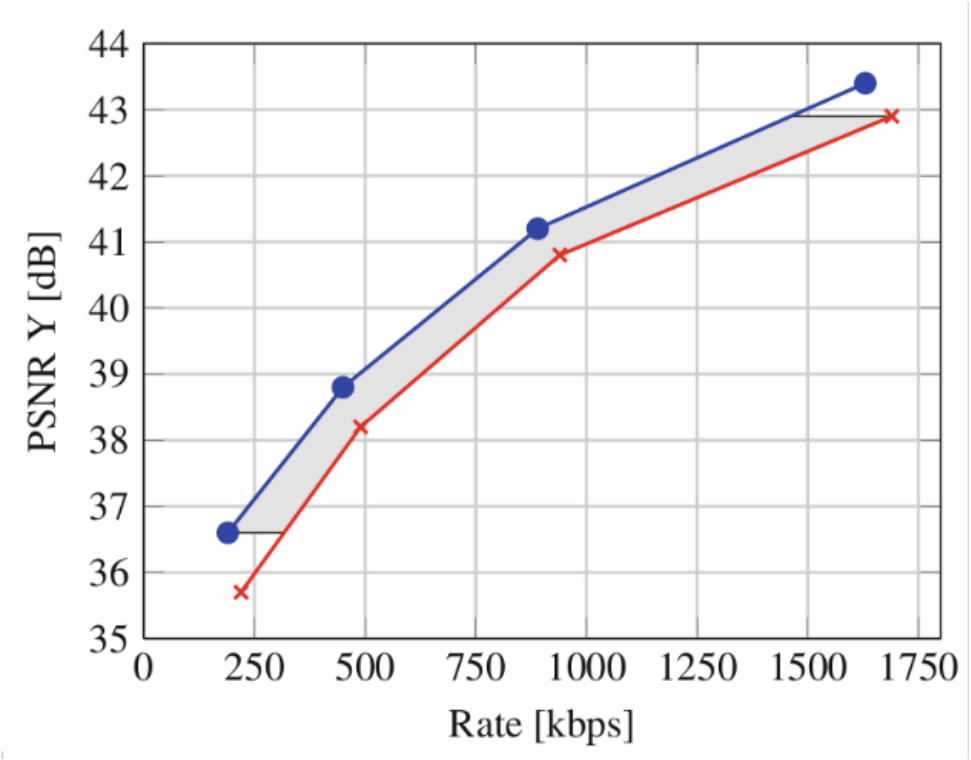


Figura 6.6 – Interpolação da Taxa BD [26].

Padrão *High Efficiency Video Coding* (HEVC)

Atualmente, o padrão UIT-T H.265 [27], também conhecido como HEVC, é um dos padrões estáveis no mercado voltado para atender as novas demandas de vídeos digitais. A proposta do HEVC apresenta uma eficiência de compressão duas vezes maior do que o seu antecessor, o H.264/*Advanced Video Coding* (AVC) e com um custo computacional de 50% a 400% mais elevado [28, 29, 30].

Entra as principais propostas presente no HEVC, estão o aumento do suporte ao paralelismo, melhorias das técnicas de predição intra e inter quadro, e um novo esquema de particionamento de imagem mais flexível e robusto, denominado *Quadtree*. Este esquema mapeia as informações de cada quadro de um vídeo em uma estrutura de árvore, que por sua vez é sub-dividida recursivamente.

Estrutura *QuadTree*

Para realizar o processo de codificação, o HEVC utiliza uma estrutura denominada *Quadtree* para mapear as informações da imagem de acordo com a etapa corrente. A estrutura básica de processamento no HEVC é a *Coding Tree Unit* (CTU), composta

pela associação de uma estrutura denominada *Coding Tree Block* (CTB) de luminância e duas CTBs de crominância. As CTUs podem ser particionadas em estruturas menores denominadas *Coding Units* (CU), de acordo com as informações carregadas nas CTUs.

As CUs são necessárias uma vez que pode haver mais detalhamento em escala de particionamento menor do que uma CTU pode comportar. Cada CU pode ser particionadas em Prediction Units (PUs), que indicam a forma de predição utilizada na respectiva CU. As PUs fornecem informações para as *Transform Units* (TU), que serão utilizadas na codificação do próximo quadro.

A Figura 6.7a ilustra a forma de particionamento de uma CTU, que pode ser dividida recursivamente em quatro ou mais estruturas de subárvores (Figura 6.7d) ou em CUs, que são compostas por informações de luminância e de crominância correspondentes aos elementos sintáticos, conforme a variação de conteúdo da imagem [31]. A CU ainda pode ser divida em PUs. As PUs podem ter tamanhos mínimo de até 4×4 pixeis e podem ser divididos de forma simétrica (Figura 6.7b) ou anti-simétrica (Figura 6.7c), em que $N \in \{4, 8, 16, 32\}$.

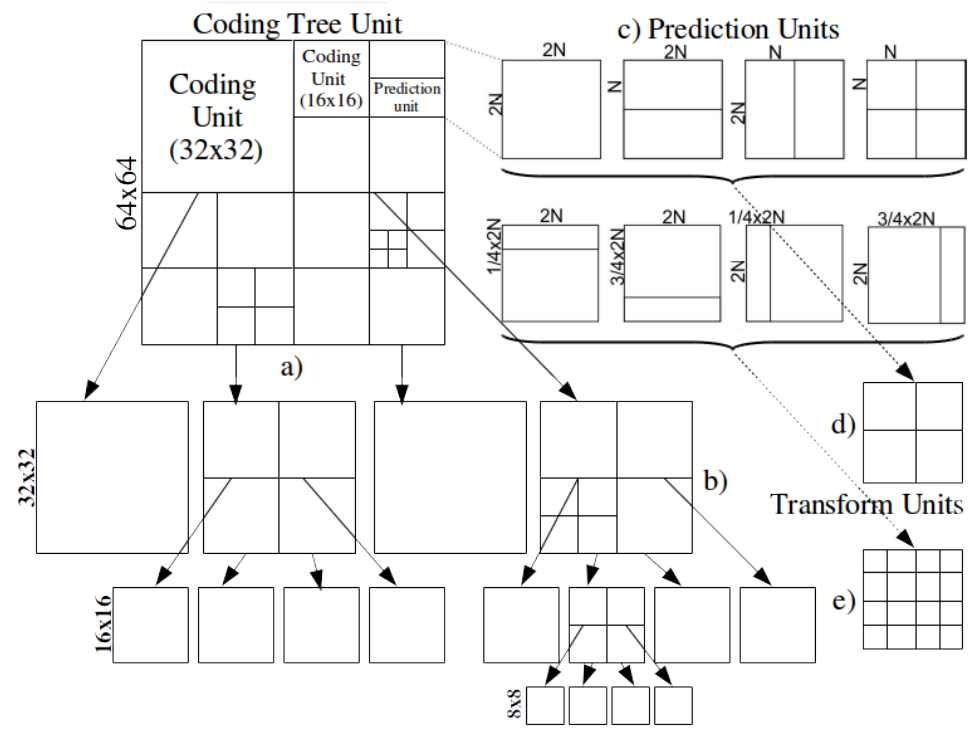


Figura 6.7 – Estrutura *Coding Tree Unit* dividida em *Coding Units* e *Coding Units* dividida em *Prediction Units*.

Cada CTU pode ser dividida em múltiplas CUs, com blocos com dimensões de 8×8, 16×16, 32×32 ou 64×64. Cada CU pode ainda ser dividida em PUs menores, inclusive nos modos *SKIP* e *MERGE*, que resultam em um bloco de dimensões $2N \times 2N$ (Figura 6.7e). Nesse processo, o codificador verifica 4^i partições, $i \in [0, 3]$. Com isso, uma CTU pode ter $(2^4 + 1)^4 + 1 = 83522$ diferentes combinações de tamanhos de unidades de codificação. Esta estrutura apresenta um aumento considerável na flexibilidade em que a imagem é

particionada, porém demanda um alto custo computacional em relação à estrutura utilizada pelo H264/AVC [21, 32].

HEVC: Predição Intra

A predição intra é utilizada para explorar a redundância espacial em uma imagem, para isso é utilizado as informação dos blocos vizinhos para identificar o modo de predição adequado. No HEVC são aplicados 33 modos de predição angular, como ilustrado na Figura 6.8, além do modo DC e o modo planar que são direcionados para áreas planas ou áreas com pouca estrutura da imagem [32, 33].

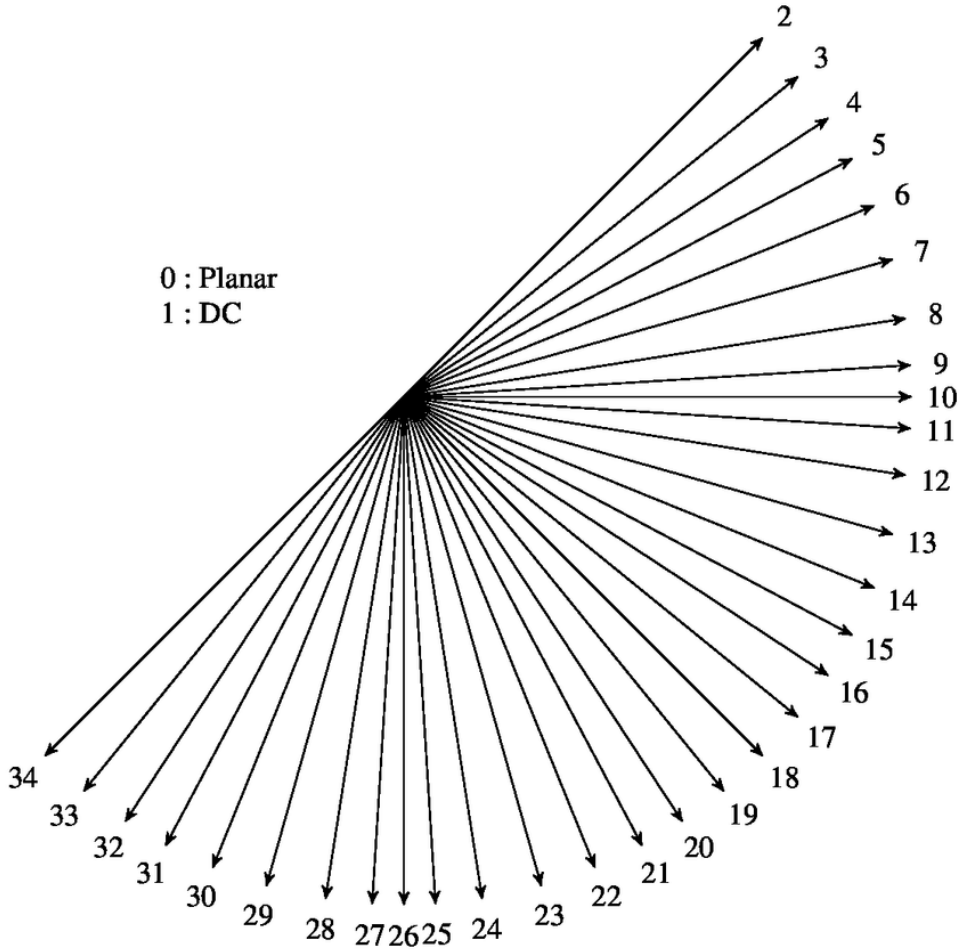


Figura 6.8 – Modo de Predição Intra [33]

A predição intra é aplicada em blocos de diferentes tamanhos de acordo com o tamanho da respectiva TU. Esses tamanhos podem variar em 4×4, 8×8, 16×16, 32×32 e 64×64.

HEVC: Predição Inter

A predição inter atua na redundância temporal que existe entre os quadros sucessivos. Para isso é realizada uma estimação de movimento utilizando sequências de quadros. O HEVC usa vetores de movimento de um quadro de referência (uni-predição) ou dois quadros

de referência (bi-predição). Esses vetores proveem blocos de pixeis que são reaproveitados no processo de compressão do quadro atual, como ilustra a Figura 6.9.

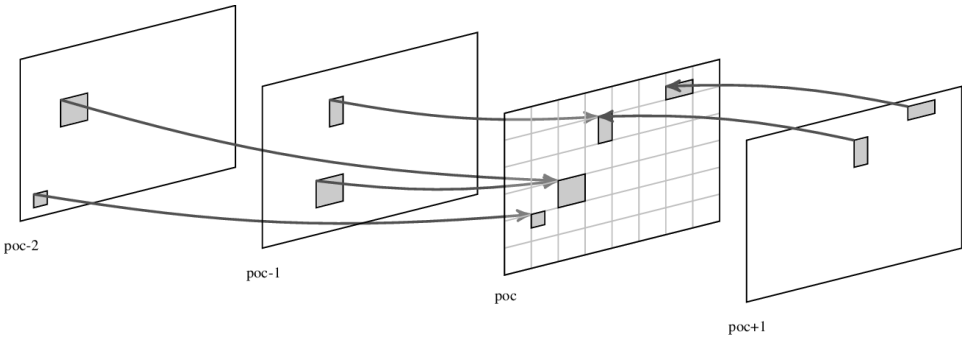


Figura 6.9 – Modo de Predição Inter [26]

O tamanho do bloco previsto, chamado de PU, é determinado pelo tamanho da CU e seu modo de particionamento. Cada bloco pode ter tamanho de até 64x64 pixeis, e pode ser dividido recursivamente utilizando uma *QuadTree*, como ilustra a Figura 6.10. Com esta abordagem é possível ter maior flexibilidade de particionamento a um custo do aumento respectivo da complexidade [34].

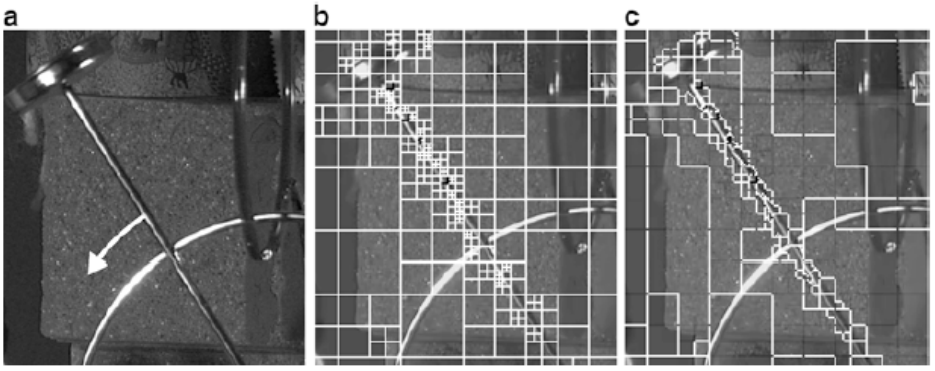


Figura 6.10 – Modo de Predição Inter [34].

A Figura 6.10 ilustra uma sequência de imagens a), b) e c) de um pendulo, utilizando particionamento QuadTree, é possível observar o aumento da intensidade dos particionamentos nos blocos na região de movimentos. A Figura 6.10 c) também ilustra o uso de mesclagem de folha redundantes posterior ao particionamento baseado em QuadTree. Essa etapa permite mesclar um bloco de folha com um dos blocos de folha vizinhos, principalmente se o bloco vizinho for de um galho pai diferente [35].

HEVC: Processamento Paralelo

Diferente do padrão H.264/AVC, em que o paralelismo foi pouco explorado, o HEVC buscou constantemente explorar a capacidade *multi-core* da maioria dos dispositivos do mercado atual. Para aumentar a eficiência por meio de paralelismo, o HEVC utiliza uma

estratégia de divisão dos quadros (imagens do vídeo), em que cada parte resultante pode ser manipulada paralelamente sem interferir na qualidade do particionamento ou na forma com as predições atuam sobre a imagem [36].

No HEVC, há duas principais técnicas de paralelismo: usando *Tiles* e o modo *Wavefront*. Na técnica de *Tiles*, em que a imagem é dividida como uma grade de blocos retangulares, não é realizada a codificação no modo intra-predição nas bordas dos retângulos. No modo *Wavefront* cada linha da CTU pode ser codificada e decodificada por uma *thread*, para isso as linhas da imagem são sinalizadas e sincronizadas na codificação e decodificação, possibilitando que cada CTU seja adequadamente manipulada por sua respectiva *thread*.

Implementação de Referência

Para promover a pesquisa de otimização do HEVC, o comitê JCT-VC fornece o código fonte de codificador de referência (HM), que possui uma implementação completa de especificações HEVC, o que permite realizar estudos sobre eficiência e desempenho do codificador e desenvolver novos recursos e otimizações [37]. No entanto, a avaliação de desempenho dos codificadores desenvolvidos com base no código-fonte de referência HM é um processo demorado, uma vez que cada teste implica na execução de várias instâncias do codificador avaliado e do codificador de referência, podendo durar várias horas e até dias. Após a execução, é necessário comparar os vídeos usando métricas objetivas (como as discutidas na Seção 6, ou subjetivas. Com as comparações é possível avaliar a eficiência e o desempenho dos novos codificadores propostos [38].

Analisar código do HM pode ajudar no entendimento prático do funcionamento do HEVC. A implementação segue os conceitos do paradigma da orientação objetos, além de apresentar uma boa documentação, o que pode ajudar no entendimento. O código e documentação podem ser acessados em: `https://hevc.hhi.fraunhofer.de`.

Versatile Video Coding (VVC)

A *Joint Video Exploration Team* (JVET), uma equipe colaborativa formada pelo MPEG e pelo VCEG do Grupo de Estudos 16 do ITU-T, iniciou em 2015 uma implementação do HM denominada *Joint Exploration Model* (JEM). O aumento da eficiência de compressão constatado no JEM em relação ao HEVC fundamentaram o que vem a ser considerado o mais novo padrão de codificação de vídeo, denominado *Versatile Video Coding* (VVC) [39]. O VVC foi oficialmente apresentado em San Diego - California, em abril de 2018, durante a 122° reunião do MPEG. A reunião contou com a presença de 32 organizações envolvidas diretamente ou indiretamente no desenvolvimento de mecanismos de codificação.

Foi destacado um ganho de eficiência de compressão de 40% em relação ao HEVC, além do suporte para tecnologias de mídias imersivas 360° omnidirecional e também para vídeos HDR. Os resultados apresentados foram suficientes para a criação do modelo de teste para experimentos e simulação e um conjunto de *benchmarks*. O novo padrão deve

permite a entrega de serviços UHD a taxas de bits que hoje são usadas para transportar vídeos HDTV. Como alternativa, o uso do VVC permitiria que o dobro do conteúdo de vídeo seja armazenado ou enviado por meio de um serviço de *streaming* [40].

Entre as principais propostas apresentadas na implementação de referência JEM estão [39]:

- Estrutura de bloco;
  - A nova estrutura de particionamento da imagem, denominada *Quadtree plus binary tree* (QTBT).
- Modificações na predição Intra;
  - Aumento de direções da predição intra, de 33 no HEVC para 65 no JEM;
  - Aprimoramento da técnica de predição intra por meio *Cross-component Linear Model* (CCLM);
  - Uso do algoritmo *Position Dependent intra Prediction Combination* (PDPC).
- Modificações na predição Inter;
  - Sub-PU diretamente relacionada ao nível do vetor de predição;
  - Uso de *Locally Adaptive Motion Vector Resolution* (LAMVR);
  - Precisão de 1/16 para decisão do armazenamento de vetor de movimento;
  - Uso de *Local Illumination Compensation* (LIC);
  - Uso de *Overlapped Block Motion Compensation* (OBMC);
  - Derivação de vetor de movimento compatível com o padrão;
  - Uso de *Decoder-side Motion Vector Refinement* (DMVR).
- Transformação;
  - *Transform-Blocks* maiores e com alta frequência de zeros;
  - Transformações secundárias não separáveis dependentes de modo.
- Filtro dentro do loop;
  - Uso de *Adaptive Loop Filter* (ALF).
- Modificações do design do Context Adaptive Binary Arithmetic Coding (CABAC).
  - Seleção de modelo de contexto para níveis de coeficiente de transformação;
  - Uso de *Multi-hypothesis* utilizando estimação probabilística;
  - Inicialização para modelos de contexto.

É importante acrescentar que, apesar do JEM ser considerada a base para formulação do VVC, a documentação do padrão ainda está sendo definida pela JVET. Muitas das propostas implementadas no JEM ainda estão em fase de análise ou de otimização. Até o momento a única estrutura presente em ambas as documentações é a nova forma de particionamento de imagem *Quad Tree plus Binary Tree*.

Estrutura *Quad Tree plus Binary Tree*

Entre as principais inovações do padrão VVC está a nova forma de particionamento de imagem denominada *Quad Tree plus Binary Tree* (QTBT). Enquanto na definição do HEVC são propostos vários tipos de particionamentos (*Coding Unit, Prediction Units e Transform Units*) na estrutura *QuadTree*, na implementação do JEM é utilizada uma proposta na qual as formas de particionamento são combinadas em uma única estrutura QTBT, ilustrada na Figura 6.11.

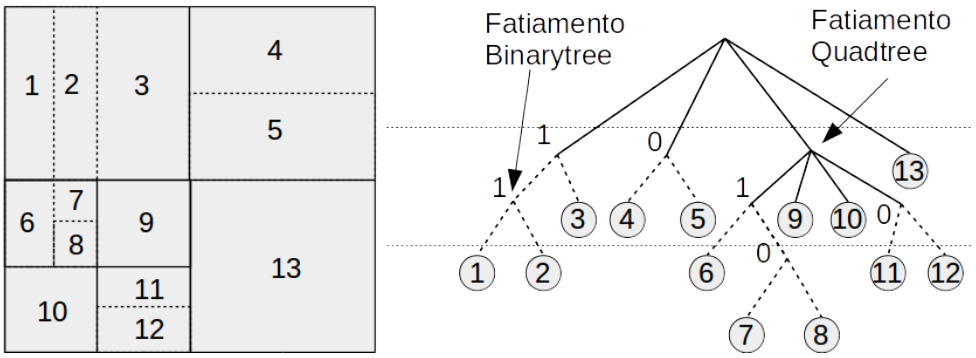


Figura 6.11 – Estrutura de Árvore QTBT [41].

A estrutura QTBT presente no JEM é composta pela junção da estrutura *QuadTree* (Figura 6.11, lado direito, linhas contínuas) e *BinaryTree* (Figura 6.11, lado direito, linhas tracejadas). Primeiro, é realizado o particionamento recursivo da estrutura *Quadtree* para obter blocos menores, semelhante à proposta do HEVC. A partir dos nós folhas da *Quadtree*, é realizado o particionamento em árvores binárias (BinaryTree). A *BinaryTree* realiza a divisão recursiva dos blocos em dois modos, horizontal e vertical, de acordo com as informações de luminância e crominância. Essa divisão resulta em estruturas de formatos retangulares, que podem ser divididos novamente em sub-árvores. Para determinar o modo como o particionamento é realizado, um sinalizador que determina se o nó será particionado no modo horizontal (com valor 0) ou vertical (com valor 1) é utilizado [39].

A profundidade das árvores *QuadTree* e *BinaryTree* são controladas por três parâmetros: MinQTSize, que representa o tamanho de nó da folha QuadTree mínima permitida e indica quando a partição *QuadTree* deve ser terminada. MaxBTSize, que determina o tamanho máximo permitido do nó folha da *BinaryTree* e MaxBTDepth, que limita a profundidade máxima da *BinaryTree* [39].

A estrutura QTBT na implementação JEM apresenta uma profundidade de particionamento de 3 níveis. Essa profundidade limita o tamanho mínimo e máximo da

Coding Unit, logo um possível particionamento distinto pode ser desperdiçado. Porém, a estrutura simplificada do JEM em relação ao HEVC possibilita uma representação mais eficiente da estrutura de árvore [41].

Na Tabela 6.1 são apresentadas as principais diferenças entre a estrutura de particionamento do HEVC (HM) e do JEM.

Algoritmo	QuadTree	QTBT
Modos de Particionamentos	Coding Unit Prediction Unit Transform Unit	Coding Unit
Tamanho da *Coding Tree Unit*	64x64	128x128
Tamanho da *Transform Unit* para luminância	4x4, 8x8, 16x.16, 32x32	4x4, 4x8, 4x16, 4x32 8x4, 8x8, 8x16, 8x32 16x4, 16x8, 16x16, 16x32 32x4, 32x8, 32x16, 32x32 64x64
Tamanho da *Transform Unit* para crominância	4x4, 8x8, 16x16	2x2, 2x4, 2x8, 2x16 4x2, 4x4, 4x8, 4x16, 4x32 8x2, 8x4, 8x8, 8x16, 8x32 16x2, 16x4, 16x8, 16x16, 16x32 32x4, 32x8, 32x16, 32x32

Tabela 6.1 – Comparação entre as estruturas do HEVC e do JEM.

Comparação de Eficiência

Na Figura 6.12 é apresentado o gráfico de eficiência de compressão dos codificadores JPEG, H.261, H.262, H.264 ,H.265 (HEVC) e do MPEG-VVC, destacando a eficiência da implementação do VVC. O gráfico foi utilizado para apresentação da proposta na 122º reunião MPEG meeting. Foi utilizado o vídeo Foreman com 10 Hz, QCIF e 100 quadros. Para testes do VVC, foi utilizada a implementação JEM 7 [39].

Considerações Finais

Neste trabalho é apresentado o cenário atual das principais tecnologias de compressão de vídeo. Para isso, foram destacados alguns desafios presentes no mercado atual como tecnologias de vídeos UHD, vídeo 360 etc. Também foram apresentadas as técnicas de compressão de vídeo utilizadas pelo HEVC e as respectivas melhorias propostas no novo padrão VVC. Foi constatado que as novas estruturas presentes no VVC proporcionam um alto ganho em eficiência de compressão além de prover um amplo suporte para as tecnologias de vídeos digitais. Inerente ao aumento de eficiência, também foi observado um constante aumento do custo computacional, porém os avanços tecnológicos e o aumento

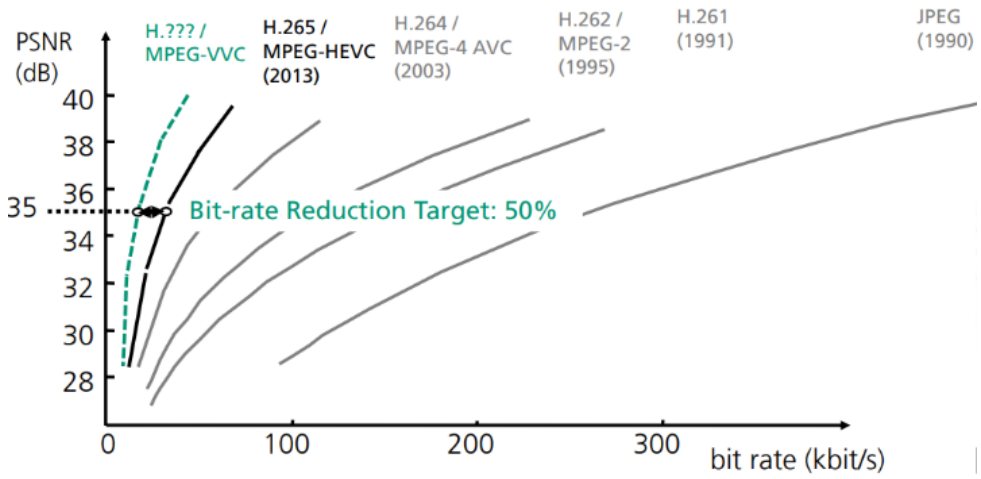


Figura 6.12 – Comparação de eficiência [40]

da disponibilidade de recursos computacionais possibilitam um cenário favorável para implantação do novo padrão VVC.

Referências Bibliográficas

[1] CISCO. Cisco vni forecast and methodology, 2015-2020. Technical report, CISCO, 2016. Disponible: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html, Access: 8 January 2016.

[2] Manri Cheon and Jong-Seok Lee. Objective quality comparison of 4k uhd and up-scaled 4k uhd videos. In *2014 IEEE International Symposium on Multimedia*, pages 78–81, Dec 2014.

[3] Ruan Delgado Gomes, Yuri Gonzaga Gonçalves da Costa, Lucenildo Lins Aquino Júnior, Manoel Gomes da Silva Neto, Alexandre Nóbrega Duarte, and Guido Lemos de Souza Filho. A solution for transmitting and displaying uhd 3d raw videos using lossless compression. In *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web*, WebMedia '13, pages 173–176, New York, NY, USA, 2013. ACM.

[4] Iain Richardson. Historical timeline of video coding standards and formats. Technical report, CODEX, 2014. Disponible: https://www.vcodex.com/historical-timeline-of-video-coding-standards-and-formats/, Access: 8 June 2018.

[5] Leandro Ribeiro Moreira. Digital video introduction. Technical report, 2017. Disponible: https://github.com/leandromoreira/digital_video_introduction, Access: 8 June 2018.

[6] M. Silva Neto A. Duarte R. Costa L. Aquino JÃºnior, R. D. Gomes and G. Souza Filho. A software-based solution for distributing and displaying 3d uhd films. *IEEE MultiMedia*, 20(1):60–68, 2013.

[7] Was Ion T. A. Lopes Marcelo s. Alencar and Francisco Madeiro t. History of television in brazil. *Telecommunications Conference (HISTELCON), 2010 Second IEEE Region 8 Conference on the History of. IEEE*, pages 1–4, 2010.

[8] 4k.com. 4k video ultra hd content: Youtube, vimeo, and ways to create it using smartphones and cameras. Technical report, apr 2016.

[9] 4k.com. Tv makers might love hdr for another reason, to boost their bottom line on 4k tvs. Technical report, apr 2016.

[10] R. Diaz, S. Blinstein, and S. Qu. Integrating hevc video compression with a high dynamic range video pipeline. *SMPTE Motion Imaging Journal*, 125(1):14–21, January 2016.

[11] Sansung. Samsung reimagines the way moments are captured and shared with gear 360. Technical report, Feb 2016.

[12] R. Skupin, Y. Sanchez, D. Podborski, C. Hellge, and T. Schierl. Viewport-dependent 360 degree video streaming based on the emerging omnidirectional media format (omaf) standard. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4592–4592, Sept 2017.

[13] Klara Nahrstedt. Distribution systems for 3d teleimmersive and video 360 content: Similarities and differences. Technical report, Jun 2018.

[14] ITU. INTERNATIONAL TELECOMMUNICATION UNION: ITU-T recommendation p.910, subjective video quality assess- ment methods for multimedia applications. Recommendation, Sep 1999.

[15] C. R. D. Estrada. Avaliação automática de qualidade de video-conferências de alta definição. Technical report, 2009.

[16] José V. M. Cardoso, Augusto C. S. Mariano, Carlos D. M. Regis, and Marcelo S. Alencar. Comparação das métricas objetivas de qualidade de vídeos baseadas na similaridade estrutural e na sensibilidade ao erro. *Revista de Tecnologia da Informação e Comunicação*, 1(2):33–40, Apr 2012.

[17] A. Banitalebi-Dehkordi, M. Azimi, M. T. Pourazad, and P. Nasiopoulos. Compression of high dynamic range video using the hevc and h.264/avc standards. In *10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, pages 8–12, Aug 2014.

[18] C. Danilo de Miranda Regis, I. de Pontes Oliveira, J. Vinicius de Miranda Cardoso, and M. Sampaio de Alencar. Design of objective video quality metrics using spatial and temporal informations. *IEEE Latin America Transactions*, 13(3):790–795, March 2015.

[19] Jose Vinicius de Miranda Cardoso, Augusto C. S. Mariano, Carlos D. M. Regis, and Marcelo S. Alencar. Comparação das métricas objetivas de qualidade de vÃdeos baseadas na similaridade estrutural e na sensibilidade ao erro. *Revista de Tecnologia da Informação e Comunicação*, 2012.

[20] Zhou Wang, Ligang Lu, and A. C. Bovik. Video quality assessment using structural distortion measurement. In *Proceedings. International Conference on Image Processing*, volume 3, pages III–65–III–68 vol.3, 2002.

[21] J. F. de Oliveira and M. S. Alencar. Online learning early skip decision method for the hevc inter process using the svm-based pegasos algorithm. *Electronics Letters*, 52(14):1227–1229, 2016.

[22] C. Danilo de Miranda Regis, I. de Pontes Oliveira, J. Vinicius de Miranda Cardoso, and M. Sampaio de Alencar. Design of objective video quality metrics using spatial and temporal informations. *IEEE Latin America Transactions*, 13(3):790–795, March 2015.

[23] A. Furnari, G. M. Farinella, A. R. Bruna, and S. Battiato. Generalized sobel filters for gradient estimation of distorted images. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 3250–3254, Sept 2015.

[24] Jean Felipe F. de Oliveira, Carlos Danilo Miranda Regis, and Marcelo Sampaio de Alencar. Performance evaluation of the pwssim metric for hevc and h.264. *International Conference on Communication, Management and Information Technology*, 65(9):115–124, 2015.

[25] G. BjÃ¸ntegaard. Document vceg-m33: Calculation of average psnr differences between rd-curves. In *ITU-T VCEG Meeting*, 2001.

[26] Mathias Wien. High efficiency video coding: Coding tools and specification. *Signals and Communication Technology*, 2015.

[27] International Telecommunication Union. Recommendation h.265: High efficiency video coding. Recommendation, ITU, 2016. Disponible: https://www.itu.int/rec/T-REC-H.265, Access: 18 January 2016.

[28] F. Bossen, B. Bross, K. Suhring, and D. Flynn. Hevc complexity and implementation analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1685–1696, Dec 2012.

[29] J. S. Yoon, C. Lee, C. Park, G. Lee, K. Lee, S. Roh, M. Jeon, Y. Jung, J. Oh, and J. A. Lee. An h.265/hevc codec for uhd(3840 x00d7;2160) capturing and playback. In *2013 International SoC Design Conference (ISOCC)*, pages 218–220, Nov 2013.

[30] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz. Performance and computational complexity assessment of high-efficiency video encoders. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1899–1909, Dec 2012.

[31] Y. Zhang, S. Kwong, X. Wang, H. Yuan, Z. Pan, and L. Xu. Machine learning-based coding unit depth decisions for flexible complexity allocation in high efficiency video coding. *IEEE Transactions on Image Processing*, 24(7):2225–2238, July 2015.

[32] Jean Felipe Fonseca de Oliveira and Marcelo Sampaio de Alencar. Padrão hevc "novas tecnologias para aplicações de elevadas taxas de compressão de vídeo. *Revista de Tecnologia da InformaÃ§Ã£o e ComunicaÃ§Ã£o*, 4(2):54–62, 2014.

[33] H. Hamout and A. Elyousfi. Low complexity intra mode decision algorithm for 3d-hevc. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1475–1479, Aug 2017.

[34] Vivienne Sze, Madhukar Budagavi, and Gary J. Sullivan. High efficiency video coding (hevc): Algorithms and architectures. In *Springer*, July 2014.

[35] Fraunhofer Research Groups. Block merging for quadtree-based block partitioning. Article, Fraunhofer, 2016. Disponible: https://www.hhi.fraunhofer.de/en/departments/vca/research-groups/image-video-coding/research-topics/block-merging-for-quadtree-based-block-partitioning.html, Access: 18 April 2018.

[36] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl. Parallel scalability and efficiency of hevc parallelization approaches. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1827–1838, Dec 2012.

[37] International Telecommunication Union. Hevc test model (hm). Recommendation, ITU, 2016.

[38] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. Technical report, Netflix, 2016. Disponible: http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html, Access: 10 January 2016.

[39] Jianle Chen, Elena Alshina, Gary J. Sullivan, Jens-Rainer Oh, and Jill Boyce. Algorithm description of joint exploration test model 7 (jem 7). In *Joint Video Exploration Team-G1001-v1*, pages 1–47, July 2017.

[40] MPEG. Versatile video coding (vvc) project starts strongly in the joint video experts team. Technical report, apr 2018.

[41] Z. Wang, S. Wang, J. Zhang, and S. Ma. Local-constrained quadtree plus binary tree block partition structure for enhanced video coding. In *2016 Visual Communications and Image Processing (VCIP)*, pages 1–4, Nov 2016.

CAPÍTULO

7 Protocolos de Comunicação para Dispositivos de Saúde na Internet das Coisas

Danilo F. S. Santos (UFCG)

Introdução

Com os últimos avanços tecnológicos em comunicação sem fio e sistemas embarcados, o número de dispositivos conectados em rede é cada vez maior, além disso, estes dispositivos são cada vez menores, autônomos e inteligentes. Portanto, esses dispositivos embarcados, sejam eles sensores ou atuadores, tornam-se capazes de se conectar entre si e com a Internet, viabilizando a chamada Internet das Coisas (do ingês *Internet of Things – IoT*) [1]. A IoT vislumbra uma rede constituída por dispositivos e não apenas por pessoas. Uma tendência natural desses dispositivos na IoT é a sua interação com o mundo físico. Por exemplo, esses dispositivos podem ser sensores que coletam informações do mundo físico, como um sensor de temperatura, mas também podem ser atuadores que alteram o ambiente, como uma válvula que controla a circulação de ar em uma casa. Esse cenário onde sensores, atuadores, e unidades computacionais interagem com o mundo físico são viáveis através de sistemas físico-cibernéticos [2] [3]. Deste modo, com a concretização da IoT, uma tendência é a integração dos sistemas físico-cibernéticos com a Internet.

Dado o cenário apresentado, um dos desafios da IoT com os sistemas físico-cibernéticos está em como fazer dispositivos heterogêneos comunicarem-se de maneira eficaz. Esses dispositivos, de maneira geral, apresentam restrições no que diz respeito ao consumo de bateria, processamento e armazenamento. Com isso, o uso de tecnologias de comunicação eficientes e de baixo consumo é essencial. Seguindo essa linha de avaliação, recentemente têm sido desenvolvidas e aprimoradas tecnologias de comunicação e sensoriamento de baixo custo, manutenção e consumo de energia, como o Bluetooth [1], Near-Field Communication

[1] http://www.bluetooth.org

(NFC) [2], ZigBee [3] e o recente Bluetooth Low Energy (BLE), apenas para mencionar algumas.

Concomitantemente, o crescente número de dispositivos conectados, como *smartphones* e *smartwatches*, viabiliza o desenvolvimento de novos serviços e aplicações, permitindo que a IoT se integre com tecnologias de uso pessoal. Nessa linha de pesquisa e desenvolvimento, novos protocolos estão sendo desenvolvidos e avaliados em diversas camadas. Protocolos como o *Constrained Application Protocol* (CoAP) [4] e o *Message Queue Telemetry Transport* (MQTT) [5] apresentam soluções na camada de aplicação que podem ser utilizadas por dispositivos embarcados com poucos recursos computacionais e de armazenamento energia, como sensores. Em comum, esses protocolos são executados sobre o Protocolo de Internet (do inglês, *Internet Protocol* - IP), viabilizando seu transporte na Internet e com dispositivos pessoais.

Em paralelo ao cenário da IoT, recentemente, com o aumento de custos com a saúde pessoal e a demanda crescente por novos serviços para o tratamento de doenças crônicas, novos desafios e oportunidades também são criados para os serviços de saúde [6]. Neste cenário, o uso da tecnologia para o monitoramento de pacientes vem crescendo ano após ano. Esse interesse por tecnologias de monitoramento de saúde impulsiona o desenvolvimento de novos Dispositivos Pessoais de Saúde (DPS) com interfaces de comunicação embutidas. Exemplos de DPS incluem medidores de pressão arterial ou glicosímetros conectados.

Com esses DPS, dados pessoais de saúde são coletados e enviados para a Internet através de suas interfaces de comunicação. Deste modo, profissionais da área de saúde podem monitorar a evolução do estado de saúde dos pacientes remotamente, e tomar ações com antecedência, evitando complicações futuras. Este processo, em conjunto com suas tecnologias envolvidas, é conhecido como Saúde Conectada [7]. Em termos gerais, um sistema de Saúde Conectada viabiliza um cenário onde DPS se conectam com a Internet para exportar seus dados, portanto, construindo a Internet das Coisas para a área de saúde. Podem-se vislumbrar casos de uso onde um DPS faz a coleta dos dados de saúde do paciente, e estes dados são compartilhados automaticamente com o seu médico transparentemente.

Um sistema de Saúde Conectada é composto por múltiplos componentes, desde o DPS até o serviço de Monitoramento Remoto de Pacientes (MRP) na Internet. Em termos gerais, o primeiro passo nesse sistema é a coleta de Informações Pessoais de Saúde (IPS) através de um DPS, e o seu compartilhamento pela Internet através de uma interface de comunicação. Por exemplo, a realização de uma coleta de dados de pressão arterial através de um tensiômetro com tecnologia Bluetooth. A partir dessa coleta e compartilhamento de dados, um dispositivo agregador, ou um Gateway, recebe essas IPS e as encaminha para o serviço de MRP na Internet [8].

Como introduzido anteriormente, esses Gateways podem ser dispositivos pessoais portáteis, como *smartphones* ou computadores pessoais. Portanto, desde o DPS até o serviço em nuvem, a informação é transportada por diversos meios de comunicação até o seu

[2]http://www.nfc-forum.org
[3]http://www.zigbee.org

destino, desde uma rede pessoal ou corporal (do inglês *Body Area Network* - BAN) até uma rede de larga escala (do inglês *Wide Area Network* - WAN) como a Internet.

Alguns desafios devem ser considerados na implantação de sistemas de Saúde Conectada para o MRP. A depender do público alvo, diferentes tipos de dispositivos poderão ser utilizados. Por exemplo, se o caso de uso for a realização de um MRP contínuo em diferentes localizações, o uso de Gateways portáteis e pervasivos, como *smartphones*, torna-se obrigatório. Em conjunto com esses Gateways, sensores corporais podem ser utilizados como DPS, como por exemplo, uma pulseira que faz aferições de frequência cardíaca continuamente. Do ponto de vista tecnológico, além de utilizar diferentes tecnologias de transmissão, como BLE, esses dispositivos e Gateways precisam definir protocolos para a troca de dados de saúde. Em relação a esse ponto, boa parte das soluções e fabricantes definem seus próprios protocolos, criando soluções verticais onde seus DPS conversam apenas com seus Gateways e serviços de saúde [9].

Em relação a esses desafios, associações e grupos de trabalhos definiram padrões de interoperabilidade para diversos níveis da cadeia de comunicação de um sistema de Saúde Conectada. No nível de DPS, na família de padrões ISO/IEEE 11073 define-se como esses dispositivos devem trocar dados com outras entidades (o padrão ISO/IEEE 11073:20601 [10]), e como as informações de saúde devem ser representadas (o padrão ISO/IEEE 11073:10101 [11]). Em relação ao compartilhamento de dados de saúde com serviços na Internet, o Continua Health Alliance[4] e o IHE[5], apresentam recomendações para o compartilhamento de dados de saúde entre DPS, agregadores de dados e serviços de armazenamento de dados de saúde [7]. Essas recomendações têm como objetivo viabilizar um cenário onde DPS compartilham dados com a Internet através de Gateways e agregadores padronizados.

Considerando o cenário descrito, esse capítulo apresenta detalhes sobre as principais características de sistemas de Saúde Conectada para a Internet das Coisas, tendo como foco a apresentação dos principais protocolos de comunicação a serem utilizados em tais sistemas. Nesse sentido, na primeira sessão, o artigo apresenta inicialmente as principais características de um sistema de Saúde Conectada para a IoT em uma abordagem baseada em interfaces, onde cada nível de comunicação tem uma descrição de interface diferente. Em seguida, na segunda sessão, são apresentados os principais protocolos de comunicação para as interfaces de primeiro nível, ou seja, as interfaces de comunicação para os dispositivos pessoais de saúde. Após a descrição dos protocolos para DPS, são descritos os principais protocolos para as interfaces de serviços, ou seja, as interfaces entre os sistemas de informação na nuvem, como o HL7 na e o FHIR, e novos arcabouços de IoT. Por fim, é realizada uma descrição de como tais protocolos podem ser integrados a depender das características dos dispositivos para saúde na IoT disponíveis no sistema, de modo a manter a interoperabilidade do sistema com outros serviços de saúde.

[4]http://www.continuaalliance.org
[5]http://www.ihe.org

Saúde Conectada na Internet das Coisas

Considerando o domínio de Saúde Conectada, torna-se importante discutir como a indústria e a comunidade científica estão utilizando padrões e tecnologias de comunicação em prol da interoperabilidade. Vários grupos de trabalhos, incluindo o grupo ISO/IEEE 11073, apresentaram padrões e recomendações para diversos níveis da cadeia de comunicação relativa a dados de saúde. Em especial, a associação *Continua Health Alliance* [6] desenvolveu recomendações para intercomunicação de sistemas e dispositivos de saúde na Internet [7]. Essas recomendações, chamadas de *Continua Design Guidelines (CDG)* foram adotadas como recomendação ITU-T H.810 [12]. O CDG apresenta uma arquitetura de referência onde interfaces são definidas para os diferentes níveis de comunicação, como ilustrado na Figura 7.1, e descritas a seguir:

- TAN-IF é a interface onde os DPS estão a um raio muito próximo de um agregador de Saúde (*Application Host Device* - AHD) em uma rede TAN (*Touch Area Network*), onde dados são trocados através de toque entre os dispositivos;
- PAN-IF é a interface onde os DPS estão a um raio próximo ao AHD do usuário, portanto, em uma rede pessoal PAN (*Personal Area Network*), como uma uma rede Bluetooth;
- LAN-IF é a interface entre o DPS e o AHD, onde ambos se comunicam a partir de uma rede local LAN (*Local Area Network*);
- WAN-IF é a interface entre o AHD e serviços na Internet através de uma rede WAN (*Wide Area Network*), também conhecida como Interface de Serviços;
- HRN-IF é a interface entre um serviço de saúde WAN e serviços de uma *Health Record Network* (HRN), ou seja, serviços de armazenamento de dados de saúde.

Em termos de tecnologias de transmissão, o CDG define o *Near-Field Communication* (NFC)[7] como tecnologia TAN, o Bluetooth e o Bluetooth Low-Energy [13] como tecnologias PAN e o ZigBee[8] como LAN. Nesses níveis, o protocolo ISO/IEEE 11073 é utilizado como base para a comunicação entre os DPS e os agregadores.

Nas interfaces restantes, são utilizados perfis disponibilizados pelo *Integrating the Healthcare Enterprise (IHE)*[9] [14]. Esse perfis, de maneira geral, fazem uso dos padrões disponibilizados pelo *Health Level 7 (HL7)* [10], os quais compreendem os padrões de maior adoção para a troca, gerenciamento, e integração de informações relativas a saúde em diferentes níveis, desde o nível clínico ao administrativo. Apesar de bastante amplo, o

[6] http://www.continuaalliance.org
[7] http://www.nfc-forum.com
[8] http://www.zigbee.org
[9] http://www.ihe.net
[10] http://www.hl7.org

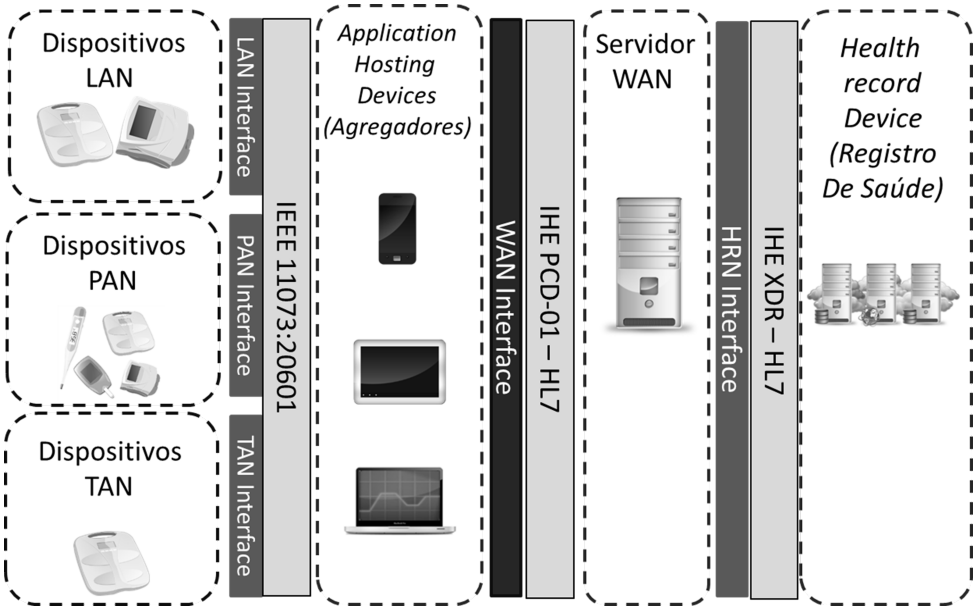


Figura 7.1 – Arquitetura de Referência do Continua Health Alliance.

HL7 não define mensagens apropriadas ao uso por DPS, portanto, por isso da adoção do ISO/IEEE 11073 nos primeiros níveis de rede pelo CDG.

Considerando o contexto desse trabalho, o CDG apresenta uma arquitetura de referência para sistemas de Saúde Conectada e MRP. Entretanto, o CDG ainda considera o uso de agregadores de dados de saúde, os quais podem ser *smartphones* ou *tablets* por exemplo, para a coleta de dados dos DPS, adaptação do formato da informação e encaminhamento para a Internet. Portanto, o CDG não apresenta recomendações que permitam que os DPS compartilhem dados diretamente com serviços na Internet. Nesse sentido, na próxima seção apresenta-se uma classificação de DPS em relação ao seu modo de compartilhamento de dados com a Internet.

Dispositivos Pessoais de Saúde

Considerando um sistema de Saúde Conectada, o primeiro passo para seu funcionamento é a coleta e compartilhamento automatizado dessas informações utilizando interfaces de comunicação como Bluetooth, ZigBee ou USB. Após a coleta, a informação deve ser enviada para a nuvem, e neste ponto podem-se classificar o DPS por seu modo de envio de informação:

- *DPS preparado para a Internet*. São dispositivos que geram a informação de saúde preparada para Internet, ou seja, já encapsulam a informação em um formato IP, de modo que não é necessária qualquer alteração para o tráfego de dados na Internet;
- *DPS dependente de Gateway ou Agregador de Dados de Saúde*. São dispositivos que geram informações de saúde e as compartilham utilizando um Gateway coletor, ou um agregador, o qual encapsula e transforma a informação para que esta seja enviada para a Internet.

A maioria dos DPS atuais disponíveis no mercado depende de Gateways de Dados de Saúde para enviar a informação para a Internet, como ilustrado na Figura 7.2. É importante perceber que os dados de saúde podem ser alterados durante o encapsulamento de dados no Gateway. Essa operação pode gerar perda ou alteração semântica dos dados, o que do ponto de vista de dados de saúde não é tolerado.

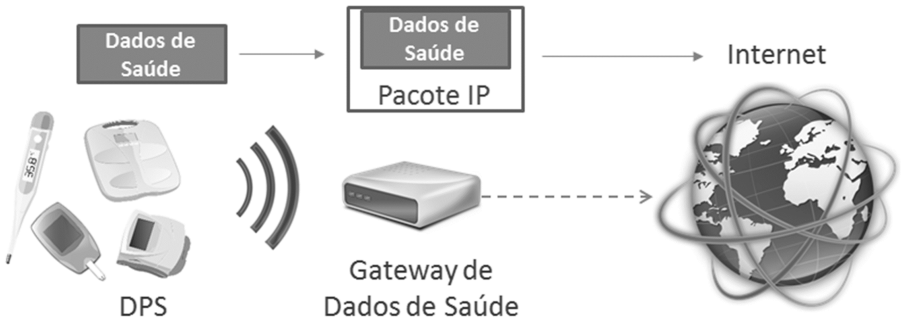


Figura 7.2 – Comunicação entre um DPS e a Internet através de um agregador de dados de saúde.

Considerando DPS preparados para a Internet, o transporte dos dados é mais simples, como ilustrado na Figura 7.3. A informação é gerada e compartilhada em um formato pronto para tráfego na Internet, portanto, os dados apenas vão trafegar por Gateways de Internet, os quais não alteram a informação, reduzindo a probabilidade de erros na manipulação dos dados.

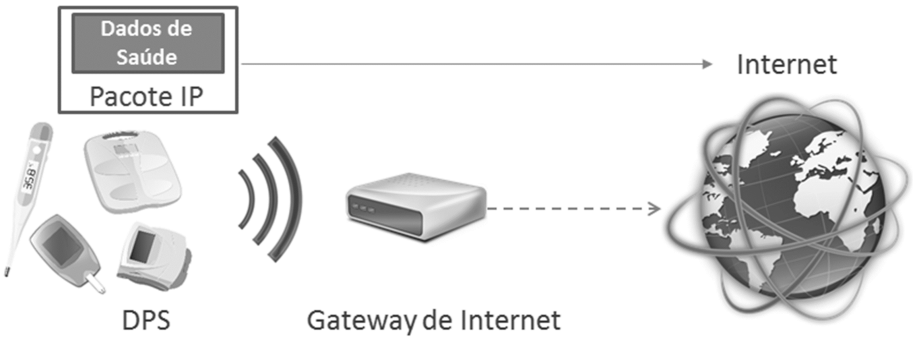


Figura 7.3 – Comunicação entre um DPS e a Internet através de um Gateway de Internet.

Com essa definição de tipos, pode-se avaliar a estrutura interna de um DPS através de sua modularização em três módulos:

- o *Módulo de Sensoriamento*, o qual é responsável pela coleta das informações do meio físico através de sensores, e do envio para o módulo de controle;
- o *Módulo de Controle*, o qual é responsável pela manipulação e processamento dos dados recebidos dos sensores;
- o *Módulo de Comunicação*, o qual é responsável pelo envio e compartilhamento das informações de saúde coletadas pelo DPS.

No decorrer desse artigo, protocolos e soluções para os *Módulos de Controle e Comunicação* serão apresentados considerando os dois tipos de DPS, com ou sem agregadores.

Protocolos de Comunicação para Dispositivos

Considerando os protocolos no primeiro nível de comunicação, ou seja, as interfaces TAN-IF e PAN-IF, alguns protocolos de comunicação sem fio e na camada de transporte se destacam na área de saúde. Nas próximas seções são descritos dois desses principais protocolos: a família de especificações ISO/IEEE 11073 e os perfis de comunicação para Bluetooth Low- Energy (BLE).

O Padrão ISO/IEEE 11073 para Dados de Saúde

Essa seção apresenta uma revisão simples sobre o padrão ISO/IEEE 11073. Dois tipos de dispositivos são definidos pelo ISO/IEEE 11073: agentes e agregadores. Agentes são produtores de dados, tipicamente dispositivos sensores como um DPS. Agregadores, por sua vez, são os coletores de dados. A conexão entre os dispositivos pode ocorrer em qualquer direção, entretanto, normalmente, o agente tem a iniciativa de conexão, pois o mesmo tem ciência de quando os dados dos sensores estão disponíveis, por exemplo, quando um paciente faz uma medição de glicose utilizando um DPS. O ISO/IEEE 11073 tem como base o requisito de que agentes são dispositivos sensores com poucos recursos computacionais e com limitações de bateria, enquanto um agregador é tipicamente um dispositivo com mais recursos computacionais e está conectado a uma fonte de energia maior. Portanto, a maior complexidade computacional do ISO/IEEE 11073 está no agregador.

O documento base do protocolo é o ISO/IEEE 11073:20601 [10]. Além deste, entretanto, existem documentos que definem especializações para dispositivos. Por exemplo, o documento IEEE 11073:10404 define a especialização para um dispositivo oxímetro de pulso. Estas especializações definem como informações específicas destes dispositivos são transportadas pelo ISO/IEEE 11073. Também é definida que tipo de informação o dispositivo suporta, por exemplo, quais dados um glicosímetro deve salvar internamente.

Em especial, relativo ao objetivo deste trabalho, o ISO/IEEE 11073 é um protocolo independente de meio de transporte. Portanto, dados ISO/IEEE 11073 podem ser transportados por praticamente qualquer tecnologia de transmissão baseada em pacotes, como TCP/IP, Bluetooth ou USB. Vários DPS disponíveis no mercado fazem uso do Bluetooth HDP (*Bluetooth Health Device Profile*) como tecnologia de transporte. Outros dispositivos, por exemplo, fazem uso do perfil USB PHDC (*Personal Health Device Class*). Em ambos os casos esses perfis oferecem meios para o transporte de dados de saúde definidos pelo ISO/IEEE 11073.

A especificação do protocolo ISO/IEEE 11073 é definida utilizando a linguagem ASN.1 e os dados são trafegados entre os dispositivos através de APDUs (*Application Protocol Data Units*) [11]. Em uma camada superior é definido um DIM (*Domain Information Model*) [15], o qual define uma estrutura de dados para um agente específico. Um dispositivo agente instancia um conjunto de classes ISO/IEEE 11073, as quais têm atributos que definem medições, unidades, etc. Todo agente tem um objeto MDS (*Medical Device System*). Um objeto MDS contém atributos com informações do fabricante, especializações do agente, ID do sistema, entre outros. Atributos MDS podem ser obtidos e alterados através de operações de *Get/Set*. Um agregador, normalmente, coleta informações sobre o MDS do agente via relatórios de eventos e configurações.

Como a maioria dos protocolos, o fluxo de controle do ISO/IEEE 11073 é governado por uma máquina de estados. O diagrama da Figura 7.4 apresenta uma representação simplificada dessa máquina de estados. Dois estados principais são definidos, *Disconnected e Connected*. Quando no estado *Connected*, agente e agregador devem iniciar um procedimento de *Association*, passando pelos os estados de *Associating*, *Associated*, *Disassociating* e *Unassociated*. Durante o estado de *Associated*, o agente e o agregador iniciam a troca de informações de configuração, para então entrar em operação (*Operating*), onde eles realmente trocam eventos com medições.

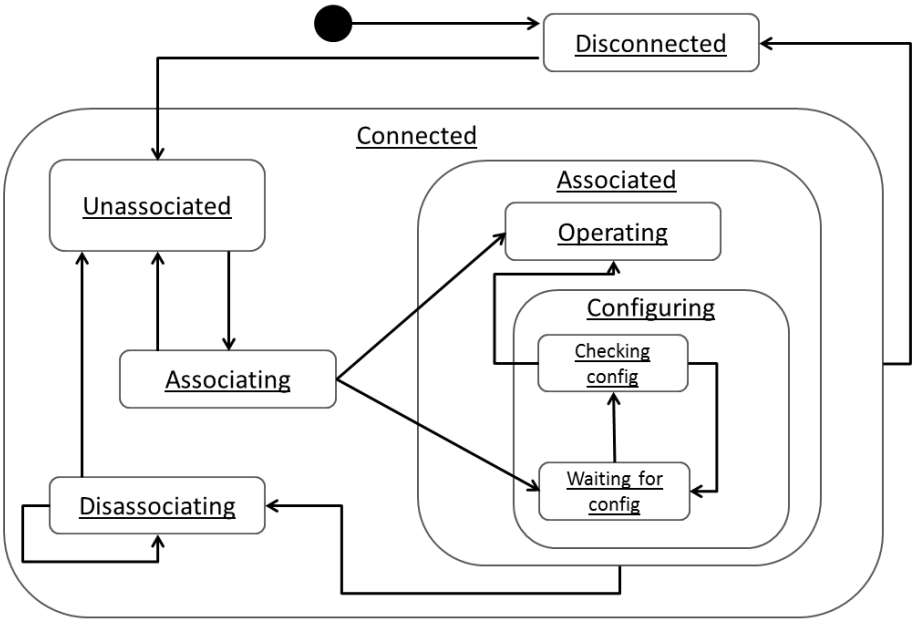


Figura 7.4 – Máquina de Estados do ISO/IEEE 11073.

Após o procedimento de *Association*, mas antes de entrar em operação, o agente envia sua configuração ao agregador através de um APDU de evento. A configuração descreve detalhes do agente, como quantos objetos MDS existem e quais são seus atributos. Com essa informação, o agregador é capaz de interpretar eventos futuros do agente e, portanto, interpretar medições transmitidas pelo mesmo. Com isso, o agregador não precisa ter conhecimento prévio sobre o agente, e pode aprender detalhes do mesmo na fase de configuração.

Bluetooth, o Perfil GATT e o Protocolo ATT

Seguindo essa linha de avaliação, recentemente têm sido desenvolvidas e aprimoradas tecnologias de comunicação e sensoriamento de baixo custo, manutenção e consumo de energia, como o Bluetooth [11], Near-Field Communication (NFC) [12], ZigBee [13] e o recente Bluetooth Low Energy (BLE), apenas para mencionar.

Considerando especificamente o domínio de saúde, o BLE apresenta-se como uma das principais tecnologias balizadoras da saúde conectada. Isso se faz devido a sua grande adoção por dispositivos para usuários finais, como smartphones, e sua arquitetura de comunicação baseada em um protocolo de atributos.

Semelhantemente ao Bluetooth convencional (BR/EDR), o BLE opera na faixa de frequência ISM de 2.4GHz [13]. O BLE também faz uso de *frequency hopping*, e oferece dois esquemas para múltiplo acesso à camada física, o *Frequency Division Multiple Access* (FDMA) e *Time Division Multiple Access* (TDMA). Quarenta (40) canais são utilizados no esquema FDMA, dos quais, três (3) são utilizados para *advertising* e 37 para troca de dados. O esquema TDMA faz com que cada dispositivo tenha um tempo pré-determinado para a transmissão de pacotes. O canal físico, então, é dividido em unidades de tempo conhecido como eventos de conexão (*connection events*). Dados são transmitidos entre os dispositivos BLE dentro desses eventos. Após a conexão ser estabelecida, o iniciador se torna o mestre (*master*) de uma rede *piconet*, e o outro dispositivo se torna seu escravo (*slave*), como ilustrado na Figura 7.5.

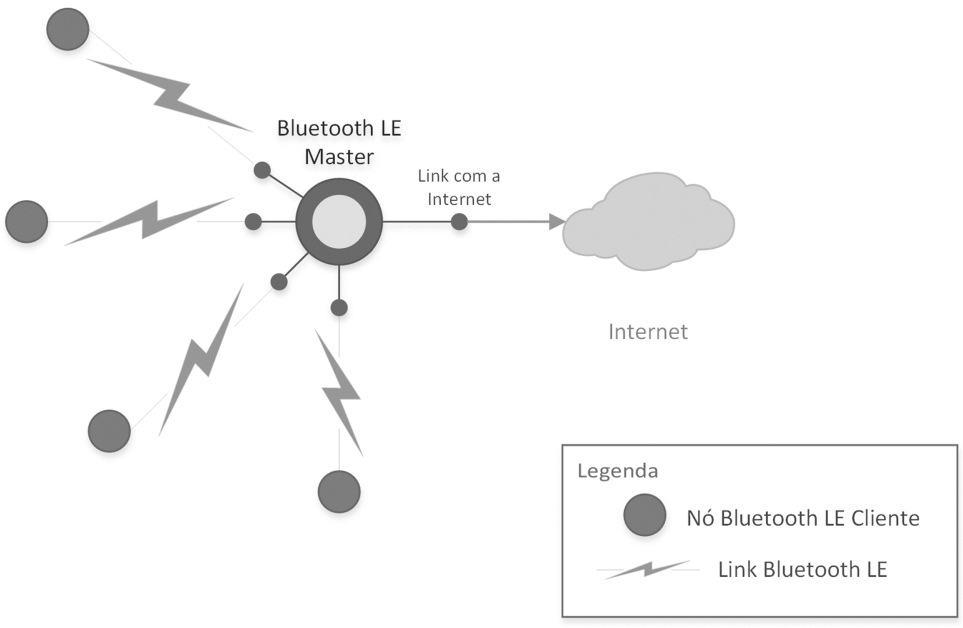


Figura 7.5 – Formação de uma rede Piconet BLE.

Sobre o canal físico existem os conceitos de enlace, canais e protocolos de controle. A hierarquia é composta pelo canal físico, enlace físico, transporte lógico, enlace lógico e canal

[11] http://www.bluetooth.org
[12] http://www.nfc-forum.org
[13] http://www.zigbee.org

L2CAP (*Logical Link Control and Adaptation Protocol*). É interessante observar que tanto a camada física quanto a camada de enlace fazem parte do lado controlador de um dispositivo Bluetooth. O lado controlador, fazendo uma definição simples, pode ser considerado a parte de hardware do dispositivo Bluetooth propriamente dito. Por exemplo, um adaptador USB ou PCI Bluetooth é o controlador, enquanto o computador pessoal é o hospedeiro. Nessa divisão, a pilha de protocolos Bluetooth acaba sendo definida por uma interface HCI (*Host Controller Interface*), onde parte do protocolo é controlado pelo hospedeiro, e a outra parte pelo controlador, como ilustrado no diagrama da Figura 7.6.

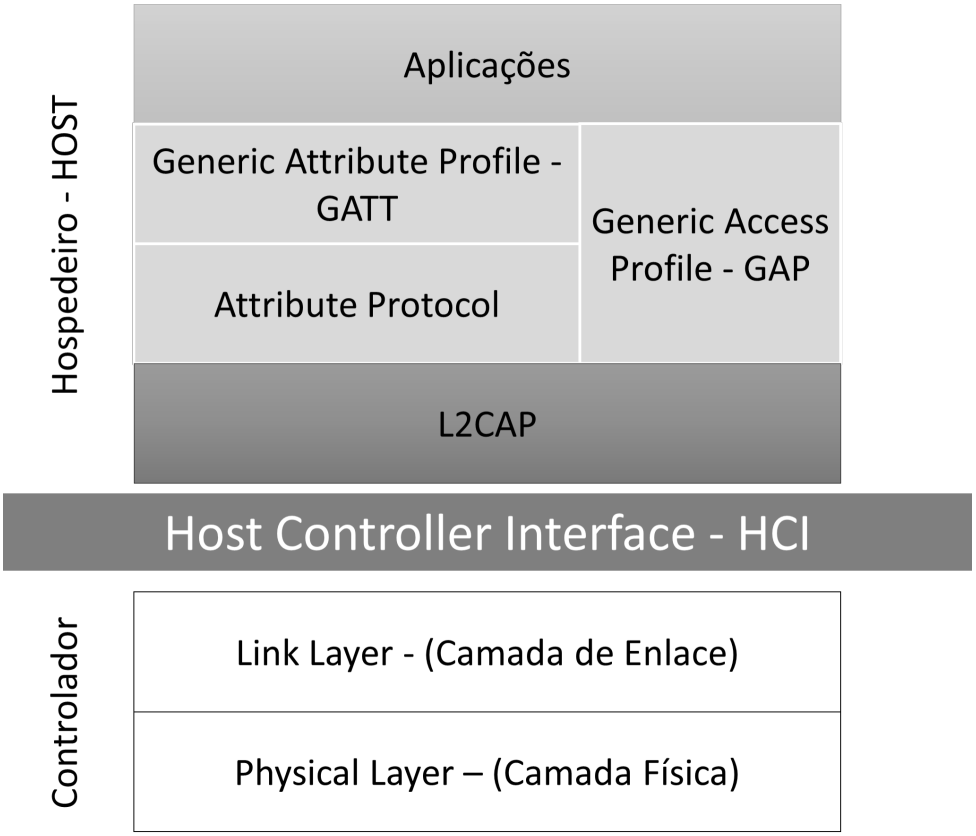


Figura 7.6 – Pilha de protocolos para o BLE.

Especificamente para o BLE, o mesmo tem como principal protocolo de nível superior o do *Generic Attribute Profile* (GATT), o qual faz um uso de um protocolo de atributos (ATT) para realizar a troca de informações entre dispositivos BLE. O ATT define dois papéis, cliente e servidor, e realiza a comunicação entre esses dispositivos através de um canal na camada de enlace. O servidor disponibiliza um conjunto de atributos que podem ser acessados por um cliente. Operações de escrita, leitura e notificação são oferecidas pelo ATT. Com o perfil GATT, um dispositivo é capaz de descobrir serviços e realizar a troca de características. Características são organizadas em valor e propriedades. Serviços e características GATT são armazenados em atributos do protocolo ATT. Portanto, o GATT realiza a definição de como identificar um serviço ou característica através do ATT [13].

Com isso, o GATT define uma estrutura de entidades que permite a interação entre os atributos de um dispositivo. No nível mais alto da hierarquia dessa estrutura é definido o perfil GATT, o qual é composto por um ou mais serviços necessários para o uso do perfil. Um

serviço, por sua vez, é compost por características ou outros serviços. Uma característica consiste de um tipo (representado por um identificador único UUID), um valor, um conjunto de propriedade que indicam que tipo de operações a citada característica suporta, e um conjunto de permissões relacionadas a segurança. Além disso, uma característica pode incluir um ou mais descritores (metadados) relativos as configurações da mesma.

O GATT agrupa esses serviços para encapsular o comportamento de parte de um dispositivo, descrevendo um caso de uso com papéis para cada serviço. Portanto, perfis GATT são utilizados para a definição de vários serviços e perfis para dispositivos simples. Por exemplo, existem definições BLE GATT para monitores cardíacos, termômetros, balanças, entre outros [14].

Em relação a serviços de saúde, diversos perfis suprem as necessidades para os dispositivos pessoais de saúde, dentre eles destacam-se os seguintes perfis e serviços disponibilizados pelo Bluetooth SIG[15]:

- Pressão arterial - *Blood Pressure Profile and Service*;
- Termômetro - *Health Thermometer Profile and Service*;
- Glicose - *Glucose Profile and Service*;
- Batimento Cardíaco - *Heart Rate Profile and Service*;
- Oxímetro - *Pulse Oximeter Profile and Service*;
- Balança - *Weight Scale Profile and Service*.

Além da descrição dos perfis e serviços GATT são definidos os valores identificadores únicos (UUID) das características do ATT de cada um desses perfis e serviços. Esses valores são reservados e gerenciados pelo Bluetooth SIG. Por exemplo, o serviço de *Weight Scale* tem um valor definido de 0x181D, e o serviço de *Thermometer* tem o valor definido de 0x1809. Esses valores são utilizados para identificar o que cada característica lida através do protocolo ATT se refere, ou seja, serve para identificar o que aquele dispositivo representa e ao que cada atributo se refere.

Por exemplo, considerando um dispositivo medidor de pressão arterial e seu perfil *Blood Pressure – GATT*, são definidos dois papéis para o processo de comunicação: Sensor e Coletor de *Blood Pressure*, como apresentado na Figura 7.7.

Com isso, o dispositivo que faz a aferição de pressão arterial é o Sensor *Blood Pressure* e o dispositivo que recebe as medições de pressão arterial é o Coletor *Blood Pressure*, o qual pode ser um smartphone por exemplo. Com isso, o Sensor deve ser um servidor GATT e o Coletor um cliente GATT. Um Sensor *Blood Pressure* é composto por diversas características, como as apresentadas na Figura 7.8 a seguir.

[14]https://www.bluetooth.org/en-us/specification/adopted-specifications
[15]http://www.bluetooth.org

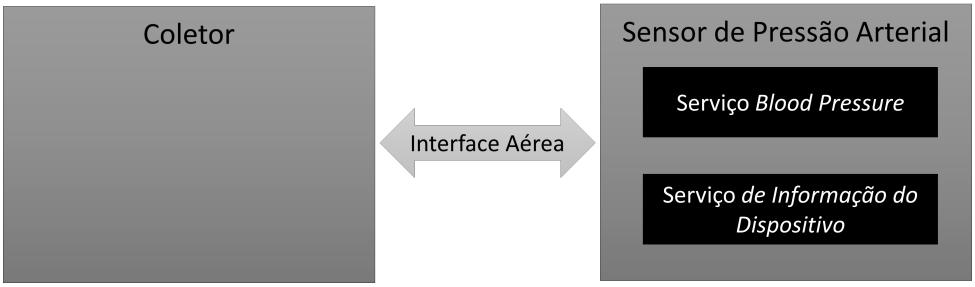


Figura 7.7 – Papéis de comunicação em um Perfil GATT Blood Pressure.

Characteristic Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
Blood Pressure Measurement	M	Indicate		None
Blood Pressure Measurement - Client Characteristic Configuration descriptor	M	Read, Write		None
Intermediate Cuff Pressure	O	Notify		None
Intermediate Cuff Pressure - Client Characteristic Configuration descriptor	C.1	Read, Write		None
Blood Pressure Feature	M	Read		None

Figura 7.8 – Tabela de características de um Sensor *Blood Pressure* em sua especificação. Fonte: Bluetooth SIG

Cada característica apresentada é representada por um identificador único, o que permite que o protocolo ATT o encontre e estabeleça a comunicação para a troca de mensagens. Por exemplo, o serviço *Blood Pressure* tem um identificar de atributo 0x1810, e a característica *Blood Pressure Measurement* tem o identificador de atributo 0x2A35. Portanto, um cliente GATT vai buscar o atributo que descreve o serviço e identificar que o valor 0x1810 representa um serviço de *Blood Pressure*. Através do protocolo ATT, o mesmo cliente irá recupera a lista de atributos disponíveis e identificar o valor 0x2A35, portanto, obtendo ciência que o valor daquele atributo representa um valor de pressão arterial.

Com essa dinâmica, vários dispositivos de saúde podem ser representados e implementados utilizando um modelo de representações por atributos e o perfil GATT do BLE.

Transcodificação BLE

Como será discutido nas próximas seções, os padrões ISO/IEEE 11073, e especificamente suas definições semânticas (ex. nomenclatura), são utilizados como base para diversas outras interfaces de comunicação. Nesse sentido, foi identificada a necessidade de mapear os dados (atributos) dos perfis GATT do BLE para o formato semântico do ISO/IEEE 11073 através de um mecanismo de transcodificação de dados [16].

A transcodificação define como dados enviados de Sensores de saúde GATT devem ser

representados e codificados no Coletor no GATT utilizando a nomenclatura e modelo de DIM do ISO/IEEE 11073:20601 [10], portanto, permitindo a interoperabilidade entre os dois padrões. Portanto, o documento de transcodificação do Bluetooth SIG define como as características de perfis GATT podem ser mapeadas de maneira consistente a nomenclatura, objetos e atributos equivalentes ao ISO/IEEE 11073:20601.

Como introduzido anteriormente, essa compatibilidade de formatos permite que dispositivos BLE-GATT possam ser utilizados em diversos sistemas de saúde, como por exemplos, sistemas baseados nas recomendações do Continua Health Alliance ou em padrões baseados em HL7. Todos os atributos mandatórios definidos em cada especialização IEEE 11073-104xx, como por exemplo a especialização ISO/IEEE 11073:10415 para Balanças [17] e a especialização ISO/IEEE 11073:10407 para medidores de pressão arterial [18], são mapeados por atributos equivalentes em seus perfis BLE-GATT. Uma importante característica do mecanismo de transcodificação é que todos os tipos são mapeados no Coletor GATT sem qualquer perda de precisão. Para tanto, o documento de transcodificação define um conjunto de requisitos específicos para cada tipo de dispositivo a serem seguidos pelo dispositivo Coletor. É importante destacar que a transcodificação define como dados de serviços BLE-GATT podem ser mapeados para a nomenclatura e o modelo DIM do IEEE 11073, entretanto, ele não exige que os dispositivos Coletores criem de fato um DIM com todos os seus objetos, atributos e pacotes APDUs associados a transações IEEE 11073.

Com isso, o processo de transcodificação pode ser descrito a partir do diagrama apresentado na Figura 7.9. Nesse processo, as características de um sensor BLE-GATT são interpretadas e mapeadas através das regras de transcodificação do BLE no Coletor. Após esse mapeamento, esses novos dados podem ser compartilhados como um agente IEEE 11073 para um agregador IEEE 11073, como apresentado na Seção 7, ou podem ser encapsulados em outro padrão de comunicação e enviados para um serviço de informação em saúde, como por exemplo utilizando o formato HL7 apresentado na Seção 7.

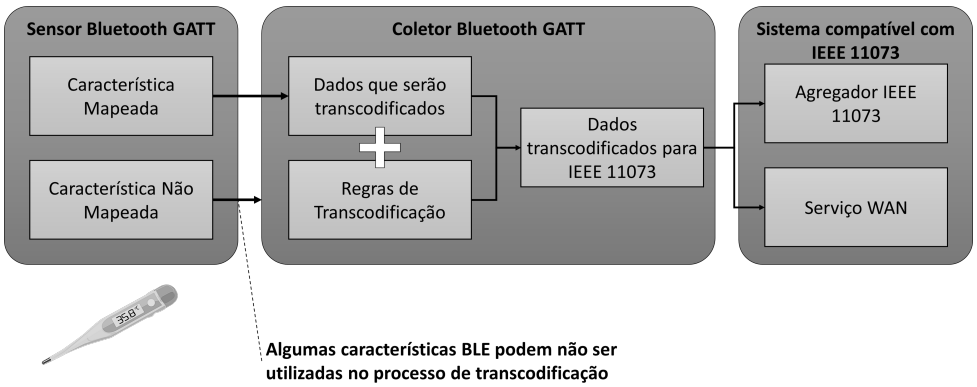


Figura 7.9 – Exemplo de processo de transcodificação. [16]

Em um exemplo prático das regras de transcodificação, considerando um termômetro de saúde definido no perfil BLE-GATT *Health Thermometer*[16] o qual deve ser mapeado na especialização ISO/IEEE 11073:10408 [19], no processo de transcodificação são definidos

[16]https://www.bluetooth.org/en-us/specification/adopted-specifications

como cada característica no GATT deve ser mapeada em um atributo ASN.1 equivalente do IEEE 11073:20601, como apresentado na Tabela 7.1.

Atributo IEEE 11073:20601	Característica equivalente no BLE	Serviço BLE	Atributo ASN.1	Tipo do Atributo IEEE 11073:20601
Metric-Id	Temperature Measurement	Health Thermometer	OID-Type	INT-U16
Unit-Code	Temperature Type and Measurement	Health Thermometer	OID-Type	INT-U16
Simple-Nu-Observed-Value	Temperature Measurement	Health Thermometer	SimpleNuObsValue	FLOAT-Type

Tabela 7.1 – Tabela de transcodificação para um termômetro de saúde IEEE 11073:10408.

Observa-se nessa tabela que, por exemplo, o *timestamp* que estar incluso na característica *Temperature Measurement* deve ser mapeado em um *Absolute-Time-Stamp* do IEEE 11073. Também, em conjunto com essa tabela de mapeamento, o documento de transcodificação apresenta tabelas específicas para mapear as unidades das medidas. Por exemplo a Tabela 7.2 apresenta o mapeamento das unidades de temperatura entre o IEEE 11073:20601 e o BLE-GATT.

Unidades de Temperatura no IEEE 11073:20601	Unidades de Temperatura no BLE	Descrição da Unidade de Temperatura
MDC_DIM_DEGC	0	Celsius
MDC_DIM_FAHR	1	Fahrenheit

Tabela 7.2 – Mapeamento das unidades de temperatura entre o IEEE 11073:20601 e o BLE-GATT.

Com esse conjunto de regras de mapeamento e conversão entre os perfis BLE-GATT de saúde e o IEEE 11073:20601, é possível utilizar dispositivos BLE de modo interoperável com diversos sistemas de informação em saúde que tem como base o IEEE 11073.

Protocolos de Comunicação para Serviços

Dentre os principais protocolos e padrões de comunicação utilizados na interface de serviços (WAN) em sistemas de saúde conectada, podem-se destacar a família de padrões HL7 com as recomendações IHE, e o padrão mais recente HL7-FHIR. A seguir são descritas as principais características desses padrões.

HL7

A sigla HL7 significa *Health Level 7*, e dá nome ao grupo de trabalho (organização)[17] que trabalha na especificação de padrões para informações de saúde de forma eletrônica. Mais especificamente, o HL7 oferece um arcabouçou de padrões para a troca, integração, compartilhamento e recuperação de informações eletrônicas em saúde. Esses padrões definem como a informação é empacotada e transportada entre diferentes entidades a partir

[17] http://www.hl7.org

da definição semântica dos dados, ou seja, a partir da definição da linguagem, estrutura e tipos de dados necessários para a integração entre diferentes sistemas.

Os padrões HL7 suportam a prática clínica e os procedimentos de gerenciamento, entrega e avaliação de serviços de saúde, fazendo que os mesmos sejam reconhecidos como os principais padrões de interoperabilidade em saúde do mundo. Os padrões HL7 são agrupados em sete (7) categorias:

- Categoria 1: Padrões Primários – São os principais padrões utilizados na integração de sistemas, interoperabilidade e sistemas de regulação.
- Categoria 2: Padrões Fundacionais – Definem os principais blocos de construção (fundação) que são utilizados para a especificação dos padrões. Esses padrões definem como todos os outros padrões do HL7 devem ser definidos.
- Categoria 3: Domínios Clínicos e Administrativos – Documentos específicos para especialidades clínicas. São normalmente implementados após os padrões primários serem implantados.
- Categoria 4: Perfis EHR – São padrões que oferecem modelos funcionais que permitem a construção e gestão de prontuários eletrônicos de saúde (EHR - *Electronic Health Records*).
- Categoria 5: Recomendações de Implementação – São materiais suplementares que apresentam recomendações e documentos de suporte a serem utilizados com outros padrões.
- Categoria 6: Regras e Referências – Especificações técnicas, estruturas de programação, e recomendações para o desenvolvimento de software.
- Categoria 7: Educação e Conhecimento – Documentos, ferramentas e outros recursos que ajudam no entendimento dos padrões do HL7.

Os padrões HL7, especialmente a primeira versão pronta para produção, a HL7 v2, foram definidas para suprir as necessidades das práticas clínicas reais. Nesse sentido, o HL7 utilizou uma abordagem onde 80

Essa possibilidade de personalização fez que com que surgissem problemas na adoção do HL7, especificamente quando pensamos na integração de diferentes sistemas de diferentes áreas. Por exemplo, é comum observar prontuários eletrônicos e hospitais utilizarem seus próprios códigos, o que faz com diferentes sistemas HL7 tenham que realizar uma pesquisa e mapeamento entre tais códigos, o que leva a aumento de custos tanto de implementação quanto de operação. Além disso, de maneira geral, as principais funcionalidades não suportadas ou oferecidas pelo HL7 incluem questões relacionadas à Segurança e Controle de acesso, Privacidade e Confidencialidade, auditorias de sistemas. Essas funcionalidades, entretanto, são importantes para a real implementação de sistemas de informação aplicados a saúde.

Com isso, na prática, a integração de diferentes sistemas de saúde advindos de diferentes domínios, como exames laboratoriais e radiologia, pode se tornar um procedimento complexo. Nesse sentido, padrões como o HL7 podem ser aplicados como referência, entretanto, ao se implementar um sistema integrador de informações em saúde, ambiguidades e interpretações conflitantes podem ocorrer. Nesse sentido, outras organizações, como o Continua Health Alliance e o IHE, a ser descrito na próxima seção, apresentam recomendações para a implementação desses padrões de interoperabilidade.

Integração via IHE

O IHE (*Integrating the Healthcare Enterprise*) é uma organização internacional colaborativa de provedores, fabricantes, agências reguladoras e experts independentes que trabalham para aprimorar a interoperabilidade no processamento e troca de informações médicas em diversas áreas, como radiologia e cuidados com a saúde. O IHE foi criado com o objetivo de integrar sistemas de imagens médicas com os sistemas de informação de hospitais, e desde então trabalha para inserção de informações de diferentes domínios em uma única infraestrutura de IT.

Considerando o domínio IHE focado em dispositivos médicos conectados, o domínio PCD (*Patient Care Domain*, o mesmo é aberto e gratuito a ser utilizado por qualquer organização que tem como objetivo obter a interoperabilidade entre dispositivos. Como introduzido anteriormente, apesar de vários padrões de comunicação relevantes existirem para a saúde, como os padrões HL7, normalmente o implementador (ou implantador) não tem em mãos um modelo de implantação claro de como tais padrões devem ser utilizados em uma situação específica. Com isso, o IHE não foi criado para definir novos padrões, e sim para definir perfis de integração, os quais apresentam recomendações de como tais padrões devem ser utilizados em cenários de uso específicos.

Nesse contexto, o IHE-PCD promove o cuidado efetivo e seguro em comunicações clínicas quando dispositivos médicos são utilizados, onde medições, configurações, eventos e dados de controle são compartilhados entre dispositivos e sistemas de saúde. O IHE-PCD tem um foco na utilização dentro de ambientes clínicos, ou seja, dentro de hospitais. Entretanto, seu uso pode ser facilmente adaptado para sistemas de saúde conectada de uso pessoal, ou seja, em casa. Esse cenário de uso é onde o Continua Health Alliance se encaixa, utilizando as recomendações IHE-PCD e as adaptando o seu uso a dispositivos pessoais de saúde. Na próxima seção são apresentados usos práticos da aplicação dos padrões HL7 utilizando as recomendações IHE, o que pode ser utilizado tanto para dispositivos médicos em ambientes clínicos, como por dispositivos pessoais de saúde.

Utilizando o HL7 com as recomendações IHE

O primeiro passo para iniciar a transmissão dos dados utilizando HL7 e as recomendações IHE é estabelecer o canal de comunicação. Esse canal pode ser um socket

TCP com um protocolo de descrição simples, ou através de interfaces RESTful [20] e SOAP (*Simple Object Access Protocol*) [21], como são utilizadas na Internet.

Em sistemas na Internet, pode-se utilizar servidores web e oferecer APIs de serviços utilizando o protocolo SOAP e seguindo as recomendações do Continua Health Alliance para a implementação da interface de serviços. Essa interface define que os dados devem ser formatados de acordo com o perfil IHE-PCD, e especificamente seguindo o formato de transações IHE PCD-01: Communication PCD Data [14]. O arcabouço técnico do PCD (*PCD technical framework*), o qual especifica tais perfis, define que mensagens HL7 V2.6 com nomenclaturas ISO/IEEE 11073, requerendo o que observações médicas sejam compartilhadas utilizando mensagens ORU (*Unsolicited Observation Result*). Um exemplo de mensagem completa HL7 v2.6 é apresentado a seguir.

```
1 MSH|^~\&|AcmeInc^ACDE48234567ABCD^EUI-64|||20090713090030+0000||ORU^R01^
      ORU_R01|
2      MSGID1234|P|2.6|||NE|AL|||||IHE PCD ORU-R01 2006^HL7
      ^2.16.840.1.113883.9.n.m^HL7
3 PID|||789567^^^Imaginary Hospital^PI ||Doe^John^Joseph^^^^L^A|||M
4 OBR|1|AB12345^AcmeAHDInc^ACDE48234567ABCD^EUI-64|CD12345^AcmeAHDInc^
      ACDE48234567ABCD^EUI-64
5     |182777000^monitoring of patient^SNOMED-CT|||20090813095715+0000
6 OBX|1|CWE|68220^MDC_TIME_SYNC_PROTOCOL^MDC|0.0.0.1|532224^
      MDC_TIME_SYNC_NONE^MDC|||||R
7 OBX|2||528391^MDC_DEV_SPEC_PROFILE_BP^MDC|1|||||||X|||||||0123456789ABCDEF^
      EUI-64
8 OBX|3||150020^MDC_PRESS_BLD_NONINV^MDC|1.0.1|||||||X|||20090813095715+0000
9 OBX|4|NM|150021^MDC_PRESS_BLD_NONINV_SYS^MDC|1.0.1.1|120|266016^
      MDC_DIM_MMHG^MDC|||||R
10 OBX|5|NM|150022^MDC_PRESS_BLD_NONINV_DIA^MDC|1.0.1.2|80|266016^MDC_DIM_MMHG
      ^MDC|||||R
11 OBX|6|NM|150023^MDC_PRESS_BLD_NONINV_MEAN^MDC|1.0.1.3|100|266016^
      MDC_DIM_MMHG^MDC|||||R
12 OBX|7|DTM|67975^MDC_ATTR_TIME_ABS^MDC|1.0.0.1|20091028123702||||||R
      |||20091028173702+0000
```

Utilizando uma abordagem mais simples através da conexão ponto a ponto via um socket TCP e do uso do protocolo MLLP - *Minimal Lower Layer protocol* como base para delimitar as mensagens HL7 enviadas, pode-se considerar a estrutura a seguir para uma mensagem:

```
1 <SB> + <Mensagem HL7> + <EB> + <CR>
2
3 Onde:
4 <SB> = Inicio do bloco (0x0B)
5 <Mensagem HL7> = HL7 compativel com os perfis do IHE PCD
6 <EB> =  Fim do bloco (0x1C)
```

```
7 <CR> = Carriage return (0x0D)
```

Uma vez configurado esse canal, o Exportador HL7 inicia uma conexão cliente TCP com o servidor do HL7 Gateway como descrito na Figura 7.10 a seguir.

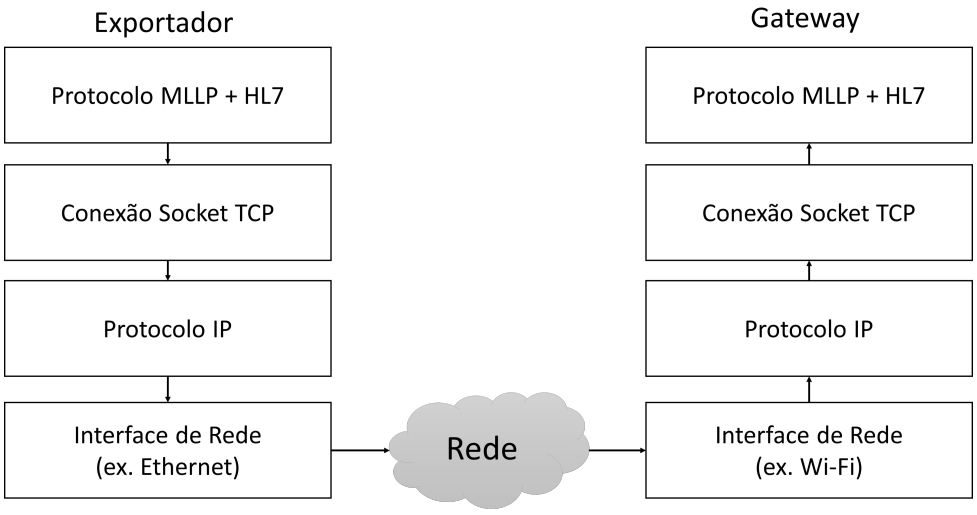


Figura 7.10 – Modelo de Comunicação MLLP com HL7.

Portanto, utilizando o MLLP, o qual envia as mensagens descritas anteriormente através de canais TCP (socket), uma mensagem HL7 segue o formato apresentado a seguir.

```
1 <SB>
2 MSH|^~\&|ZIS|1^AHospital|||199605141144||ADT^A01|20031104082400|P|2.3|||
3 AL|NE|||8859/15|<CR>EVN|A01|20031104082400.0000+0100|20031104082400
4 PID||""|10||Vries^Danny^D.^^de||19951202|M|||Rembrandlaan^7^Leiden^^7301TH
      ^""
5 ^^P||""|""||""|||||||""|""<CR>PV1||I|3w^301^""^01|S|||100^van den Berg^^A.S
      .
6 ^^""^dr|""||9||||H||||20031104082400.0000+0100<CR>
7 <EB><CR>
```

Como descrito anteriormente, as mensagens HL7 são formatadas de acordo com os perfis do IHE PCD - *Patient Care Devices*, os quais definem recomendações de como representar dados processados por dispositivos de saúde. Especificamente relativo ao HL7, os segmentos que compõem as mensagens são organizados como apresentado na Figura 7.11 a seguir.

Onde:

- MSH - Cabeçalho da mensagem
- PID - Identificação do paciente
- PV1 - Informação sobre a visita do paciente (Sala, leito e etc.)

Segmento	Significado	Uso	Cardinalidade
MSH	Cabeçalho da Mensagem	Obrigatório	[1..1]
{			
PID	Identificação do Paciente	Obrigatório	[1..1]
PV1	Identificação da Visita ao Paciente	Opcional	[0..1]
{			
OBR	Requisição de Observação	Obrigatório	[1..1]
[			
{OBX}	Parâmetro de Observação	Obrigatório	[1..*]
]			
}			
}			

Figura 7.11 – Estrutura de uma mensagem ORU com HL7.

- OBR - Requisição de observação
- OBX - Resultados da observação

Além do perfil PCD, outros perfis são definidos pelo IHE, como os seguintes perfis:

- *Device Enterprise Communication* (DEC). O perfil DEC é utilizado para reportar resultados como SpO2, Frequência cardíaca e etc.
- *Alert Communication Management* (ACM). O perfil ACM é utilizado para notificar alertas.
- *Infusion Pump Event Communication* (IPEC). O perfil IPEC é utilizado para notificar eventos relacionados a bombas de infusão (ex: infusão iniciada, infusão finalizada e etc.).

FHIR

Os padrões HL7 descritos anteriormente não são uma solução perfeita, mas suprem a maioria dos requisitos da indústria da saúde. Entretanto, quando observamos sua adoção com tecnologias voltadas para a Internet e a Web, novos requisitos surgem. Esses novos requisitos levaram a definição de um novo padrão, o FHIR- *Fast Healthcare Interoperability Resources*. O FHIR foi criado para adicionar melhorias em termos de segurança e comunicação para a web, incluindo um novo modelo de dados que facilita a customização.

Uma das principais características do FHIR é o seu modelo de comunicação baseado em REST, o que o faz ser facilmente adotado por organizações e desenvolvedores habituados a sistemas web. Esse modelo de comunicação semelhante aos adotados por sistemas na Internet faz com que sua adoção seja mais simples e rápida, reduzindo custos de adaptação e manutenção do sistema se comparado ao HL7 clássico. Entretanto, vale ressaltar que

o FHIR ainda estar em seus primeiros estágios de adoção, significando que o HL7 ainda continua sendo um dos sistemas de maior adoção na indústria de saúde.

Além de seguir um modelo de comunicação baseado em REST, ou seja, onde recursos são trocados entre entidades representando seus estados atuais, o FHIR permite que tais recursos sejam representados através de um formato JSON, o qual é amplamente utilizado em sistemas web modernos. É importante destacar que o FHIR tem como objetivo simplificar a implementação de sistemas de informação em saúde sem sacrificar a integridade da informação. O FHIR aproveita os modelos lógicos e teóricos existentes para fornecer um mecanismo consistente, fácil de implementar e rigoroso para a troca de dados entre sistemas de saúde. O FHIR possui mecanismos integrados de rastreabilidade para o HL7 RIM (*Reference Information Model*) e outros importantes modelos de conteúdo. Isso garante o alinhamento aos padrões e melhores práticas definidos anteriormente pelo HL7, sem exigir que o desenvolvedor tenha conhecimento profundo do RIM ou de quaisquer derivações do HL7 v3.

De maneira geral, a especificação FHIR é dividida em um conjunto de módulos:

- Fundamentos: A infraestrutura básica de definição e troca na qual o restante da especificação é definido;
- Suporte ao Desenvolvedor: Serviços para ajudar os desenvolvedores a usar a especificação;
- Segurança e Privacidade: Documentação e serviços para criar e manter a segurança, integridade e privacidade;
- Conformidade: como testar a conformidade com a especificação e definir guias de implementação;
- Terminologia: Uso e suporte de terminologias e artefatos relacionados;
- Dados vinculados: como usar o RDF (*Resource Definition Framework*) e como ontologias são usadas ao definir o conteúdo do FHIR;
- Administração: Recursos básicos para rastrear pacientes, profissionais, organizações, dispositivos, substâncias, etc;
- Clínica: Conteúdo clínico principal, como alergias e o processo de cuidados com saúde e muito mais;
- Medicamentos: gerenciamento de medicação e rastreamento de imunização;
- Diagnóstico: Observações, relatórios de diagnóstico e solicitações e conteúdo relacionado;
- Fluxo de trabalho: gerenciamento do processo de cuidado com saúde e artefatos técnicos relacionados ao gerenciamento de tarefas;

- Financeiro: suporte ao faturamento e reclamações;
- Raciocínio Clínico: Apoio à Decisão Clínica e Medidas de Qualidade.

Em uma arquitetura utilizando o FHIR, normalmente é definido uma interface de acesso em um exportador FHIR. Portanto, uma entidade remota (ex. dispositivo ou servidor de dados de saúde) oferece uma série de pontos de acessos a recursos de saúde seguindo as definições do FHIR. Por exemplo, um exportador FHIR pode suportar duas interfaces de exportação de dados:

- *Patient* – interface que retorna e altera informações sobre os pacientes armazenados em um sistema;
- *Observation* – interface que retorna e recebe recursos que descrevem observações realizadas sobre um paciente. Observações nesse contexto são recursos de sinais vitais do paciente.

Outras interfaces podem ser adicionadas, como por exemplo interface de manipulação de medicamentos, entre outros. Além disso, é necessário criar um mecanismo de autenticação e gestão de usuários que irão acessar esses dados.

Considerando o acesso a interface *Observation*, um cliente pode acessar um servidor exportador FHIR e, através de uma API REST, recuperar as observações clínicas de um paciente, como por exemplo, batimentos cardíacos, saturação de oxigênio (SpO2), pressão arterial (Sistólica, Diastólica, Média), frequência respiratória, temperatura corporal, entre outros. A seguir, na Figura 7.12, é apresentado um diagrama de sequência com as requisições que devem ser realizadas para o Exportador FHIR a fim de recuperar as observações.

Na resposta da requisição pode ser utilizados códigos LOINC (*Logical Observation Identifier Names and Codes*) [22] para identificar os sinais vitais coletados. As unidades das medidas coletadas são definidas de acordo com o sistema UCUM (*Unified Code for Units of Measure*) [23]. Um exemplo de resposta é apresentado a seguir, onde a informação é representada utilizando o formato JSON.

```
{
.....
  "entry": [
    {
      "fullUrl": "https://hostaddress:9443/fhir/Observation/42697",
      "resource": {
        "resourceType": "Observation",
        "id": "42697",
        "contained": [
          {
            "resourceType": "Patient",
```

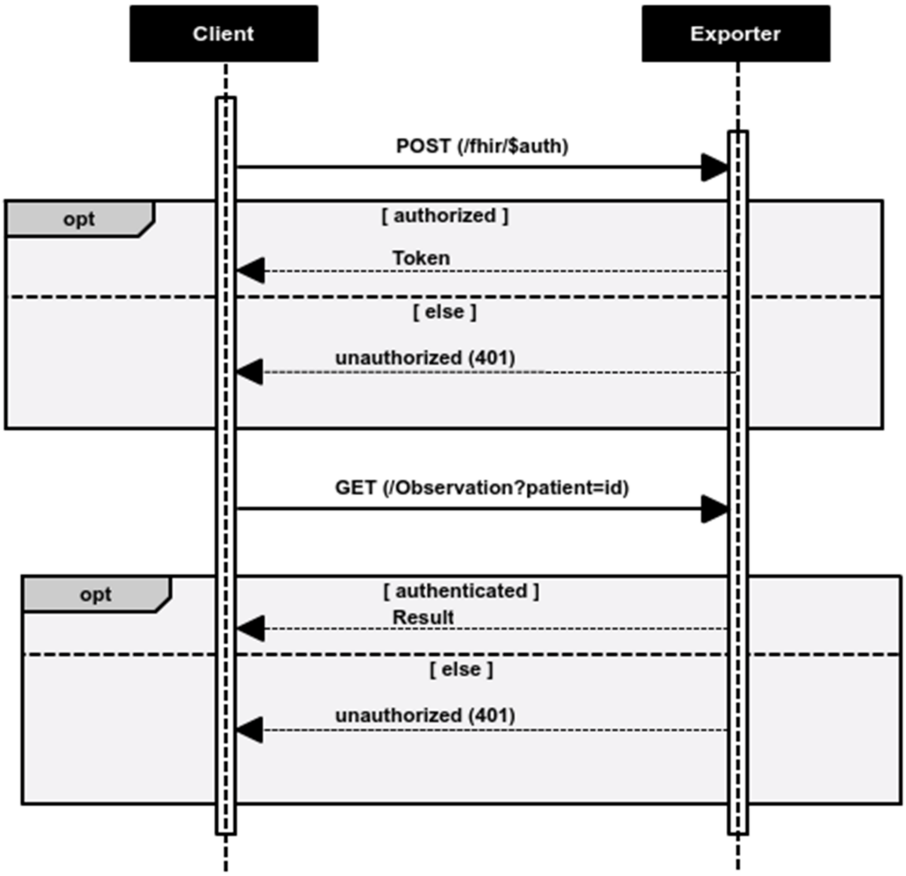


Figura 7.12 – Sequência de requisições para a recuperação de observações.

```
            "id": "12",
            "name": [
              {
                "given": [
                  "Andre"
                ]
              },
              {
                "given": [
                  "Matias"
                ]
              }
            ]
          },
          {
            "resourceType": "Device",
            "id": "59",
            "lotNumber": "SERIAL NUMBER",
            "model": "DEVICE MODEL"
          }
        ],
```

```
        "component": [
          {
            "code": {
              "coding": [
                {
                  "system": "http://loinc.org",
                  "code": "8867-4",
                  "display": "Heart rate"
                }
              ]
            },
            "valueQuantity": {
              "value": 80.0,
              "unit": "beats/minute",
              "system": "http://unitsofmeasure.org",
              "code": "/min"
            }
...
        ]
...}
```

Neste exemplo observa-se inicialmente a identificação do paciente através do recurso (*resourceType*) *PatientId*, em seguida a identificação do dispositivo através do recurso *Device*, e por fim uma lista de observações representadas por *components*, os quais apresentam o código da medida utilizando o formato LOINC e, em seguida, seus valores.

Com esse processo de comunicação utilizando FHIR, um desenvolvedor ou integrador de sistemas pode recuperar o estado de um paciente, de suas observações e informações clínicas, através da leitura e escrita de recursos FHIR, utilizando uma interface RESTful e formatação de dados JSON.

Novos arcabouços para a Internet das Coisas

Além dos protocolos e padrões de comunicação previamente citados, novo arcabouços integradores estão sendo definidos de modo a viabilizar uma melhor inserção dos dispositivos de saúde na Internet das Coisas. Esses arcabouços têm como objetivo funcionarem como vetores integradores para que dispositivos de diversas áreas compartilhem dados entre si. Como um exemplo, permitir que dispositivos multimídia e de saúde compartilhem dados de modo interoperável, permitindo a criação de novos tipos de serviços e aplicações. Dentre esses novos arcabouços pode-se destacar dois grandes fóruns de discussão da indústria, os quais definiram dois arcabouços de comunicação:

- O UPnP – Universal Plug and Play[18].

[18]https://openconnectivity.org/developer/specifications/upnp-resources

- O OCF – OpenConnectivity Foundation[19].

O UPnP define um conjunto de padrões os quais dependem de uma arquitetura central de compartilhamento de serviços através de tecnologias como HTTP e SOAP. O UPnP foi um dos arcabouços de compartilhamento de serviços pervasivos de maior sucesso, sendo utilizado por vídeo games, smart-tvs, entre outros aparelhos. Em relação a saúde conectaca, o UPnP definiu uma especificação de serviço chamada de *IoT Management and Control* a qual permite que sensores para o IoT se integrem a redes UPnP, incluindo sensores de saúde [24]. Apesar de definir um novo padrão para a comunicação entre sensores na IoT, o padrão UPnP faz uso da nomenclatura IEEE 11073 para sensores de saúde, portanto, permitindo a interoperabilidade com outros padrões.

Já o arcabouço definido pelo OCF apresenta uma arquitetura moderna, e baseada em recursos ao invés de serviços. Essa mudança de paradigma na definição da arquitetura permite uma melhor inserção do OCF com os novos sistemas na Internet baseados em REST, como também simplifica o desenvolvimento de novos dispositivos com a utilização de protocolos como o CoAP *Constrained Application Protocol* [4]. Por ser baseado em recursos, o OCF define uma biblioteca de recursos que é utilizada para a especificação de novos dispositivos, incluindo dispositivos pessoais de saúde. Essa biblioteca, chamada de OneIota[20], lista recursos como temperatura, oximetria, batimento cardíaco, altura, peso, etc. Ao se especificar um novo dispositivo, por exemplo, um oxímetro de pulso, o desenvolvedor precisa exportar apenas os recursos que representam aquele dispositivo, como por exemplo os recursos de oximetria e batimento cardíaco.

Em termos históricos, desde 2015 as especificações do UPnP fazem parte do portfólio de protocolos do OCF, permitindo a interoperabilidade entre os dois arcabouços.

Discussão e Aplicação Tecnológica

Considerando os protocolos e modelos de comunicação apresentados e as definições de tipos de dispositivos de saúde pessoal para a Internet das Coisas apresentados nas sessões anteriores, um desenvolver integrador de sistemas de saúde deve considerar diferentes cenários de implantação. Por exemplo, deve-se levar em consideração que tipo de dispositivo pessoal de saúde tem-se disponível, e que tipo de serviço de informação em saúde pretende-se realizar a integração.

Em um cenário onde o desenvolvedor irá realizar o desenvolvimento do dispositivo pessoal de saúde e do serviço de informação na Internet das Coisas, uma abordagem a ser utilizada é a utilização da nomenclatura e modelo de informação do ISO/IEEE 11073:20601 desde a transmissão dos dados no DPS, como apresentado em [25] e [26]. Nesses exemplos, pacotes ISO/IEEE 11073 são encapsulados utilizando o protocolo CoAP (*Constrained Application Protocol*), e enviados diretamente ao serviço em nuvem na Internet.

[19] http://www.openconnectivity.org
[20] http://www.oneiota.org

Essa abordagem permite que os dados enviados e recebidos no serviço em nuvem possam ser facilmente encapsulados em mensagens HL7 ou FHIR, e integrados com outros serviços em nuvem, como apresentado na Figura 7.13(b).

Ao mesmo tempo, caso o objetivo seja integrar também dispositivos legados, ou seja, dispositivos que utilizem o protocolo ISO/IEEE 11073 com o Bluetooth HDP ou dispositivos BLE-GATT com perfis em saúde, pode-se integrar um agregador de dados ao sistema, o qual pode implementar o transcodificador apresentado anteriormente e utiliza o protocolo HL7 com as recomendações IHE. Deste modo, pode-se obter um sistema integrado de Saúde Conectada para a Internet das coisas, como o apresentado na Figura 7.13, onde o lado (a) representa a integração dos dispositivos de saúde legados, e o lado (b) apresenta a integração de dispositivos preparados para a Internet das Coisas.

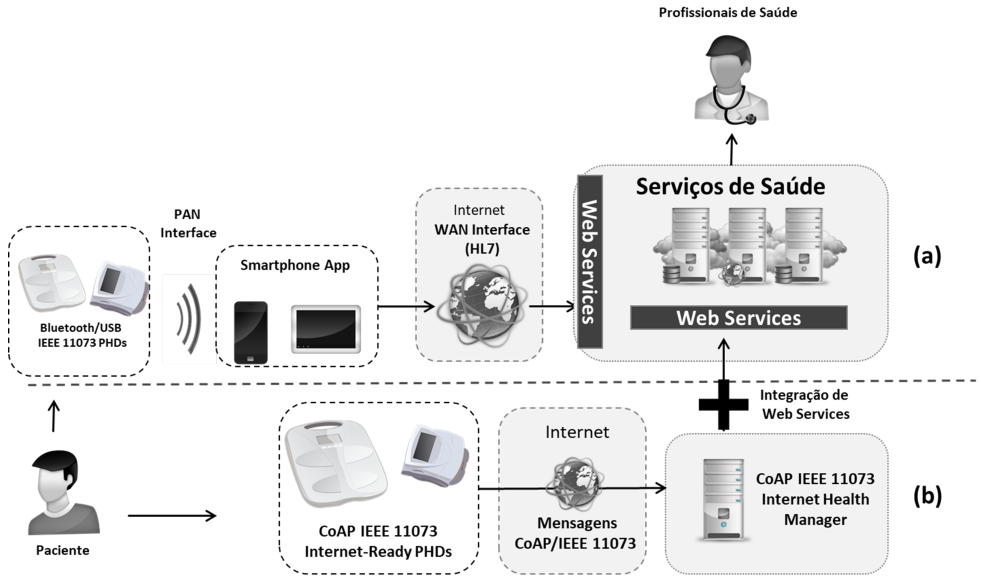


Figura 7.13 – Visão integrada de um sistema de Saúde Conectada para a Internet das Coisas.

Como cenário de integração paralelo, outras opções podem ser consideradas, como por exemplo a utilização de novos arcabouços de integração para a Internet das Coisas, como os apresentados anteriormente, os quais permitem a integração de dispositivos diretamente com o serviço em nuvem utilizando outros protocolos de transporte, como o OCF. Nesse cenário, recomenda-se realizar o mapeamento das estruturas de dados do OCF para formatos compatíveis com o HL7v2 ou FHIR, de modo que o sistema de saúde conectada desenvolvido seja compatível com outros sistemas de saúde existentes.

Conclusões

Novas tecnologias voltadas para a Internet das Coisas (IoT) estão sendo criadas, as quais viabilizam a criação de novos serviços e aplicações em diversas áreas. Em paralelo, tecnologias de comunicação estão sendo estendidas com o objetivo de viabilizar a conexão da maior variedade de dispositivos com a Internet, como por exemplo, Dispositivos Pessoais de Saúde (DPS). Nesse sentido, o uso adequado de padrões de interoperabilidade e

seus protocolos associados devem ser explorados e bem aplicados no desenvolvimento de serviços de Saúde Conectada para a Internet das Coisas.

Considerando esse cenário, a definição de uma arquitetura de IoT para sistemas de saúde deve seguir um modelo de divisão de interfaces de comunicação simples e clara, de modo a dividir os campos de atuação dos implantadores e desenvolvedores. Além disso, o uso de protocolos de interoperabilidade, como discutido no decorrer desse trabalho, permite que tanto os dispositivos pessoais de saúde quanto os sistemas na Internet possam compartilhar informações de modo seguro e simples. Nesse sentido o uso adequado de tais protocolos de comunicação deve ser incentivado, e até regulamentado, como iniciativas de diversos países, como o Brasil, o qual determinou o uso de padrões de interoperabilidade para a comunicação em seu sistema único de saúde, o e-SUS, através da Portara de Interoperabilidade 2073/GM/MS de 31 de agosto de 2011.

Referências Bibliográficas

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[2] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. Lee & Seshia, 2011.

[3] Fang-Jing Wu, Yu-Fen Kao, and Yu-Chee Tseng. From wireless sensor networks towards cyber physical systems. *Pervasive and Mobile Computing*, 7(4):397–413, 2011.

[4] Zach Shelby, Klaus Hartke, Carsten Bormann, and B Frank. RFC 7252: The constrained application protocol (CoAP). *Internet Engineering Task Force*, 2014.

[5] Dave Locke. MQ telemetry transport (mqtt) v3. 1 protocol specification. *IBM developerWorks Technical Library], available at http://www. ibm. com/developerworks/webservices/library/ws-mqtt/index. html*, 2010.

[6] David P.L. Simons. Consumer electronics opportunities in remote and home healthcare. *2008 Digest of Technical Papers - International Conference on Consumer Electronics*, pages 1–2, 2008.

[7] R Carroll, R Cnossen, M Schnell, and D Simons. Continua: an interoperable personal healthcare ecosystem. *IEEE Pervasive Computing*, pages 90–94, 2007.

[8] Danilo Freire de Souza Santos, Aldenor FalcÃ£o Martins, Hyggo Almeida, and Angelo Perkusich. IEEE 11073 and Connected Health: Preparing Personal Health Devices for the Internet. In *Proceedings of ICCE 2014*. IEEE, 2014.

[9] Danilo Freire de Souza Santos, A. Perkusich, and H.O. Almeida. A personal connected health system for the internet of things based on the constrained application protocol. *Computers and Electrical Engineering Journal*, 2015.

[10] IEEE. *ISO/IEEE 11073-20601: Health informatics - Point-of-care medical device communication - Part 20601:Optimized exchange protocol Standards*, 2010.

[11] IEEE. *ISO/IEEE 11073-10101: Health informatics - Point-of-care medical device communication - Part 10101:Nomenclature*, 2008.

[12] ITU-T. *H.810 : Interoperability design guidelines for personal health systems*, 2013.

[13] Bluetooth SIG. Bluetooth specification version 4.2. *Bluetooth Special Interest Group*, 2014.

[14] John G Rhoads, Todd Cooper, Ken Fuchs, Paul Schluter, and Raymond Peter Zambuto. Medical device interoperability and the Integrating the Healthcare Enterprise (IHE) initiative. *Biomed Instrum Technol*, pages 21–7, 2010.

[15] IEEE. *ISO/IEEE 11073-10201: Health informatics - Point-of-care medical device communication - Part 10201:Domain information model*, 2008.

[16] R Gonzá¡lez Gómez, R. D. Hughes, D. Bogia, K. Shingala, L. Kermes, M. Lima, R. Herbster, L. Pátzenreuter, E.and Ott, J. R. Barr, G. Schatz, and R. Strickland. Personal health devices transcoding white paper, v1. 4. *Bluetooth SIG*, 2012.

[17] Iso/ieee health informatics–personal health device communication–part 10415: Device specialization–weighing scale. *ISO/IEEE 11073-10415:2010(E)*, pages 1–52, May 2010.

[18] Iso/ieee health informatics personal health device communication part 10407: Device specialization blood pressure monitor. *ISO/IEEE 11073-10407:2010(E)*, pages 1–52, May 2010.

[19] Iso/ieee health informatics personal health device communication part 10408: Device specialization thermometer. *ISO/IEEE 11073-10408:2010(E)*, pages 1–48, May 2010.

[20] Leonard Richardson and Sam Ruby. *RESTful web services*. "O'Reilly Media, Inc.", 2008.

[21] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. Simple object access protocol (soap) 1.2. *World Wide Web Consortium*, 2003.

[22] Clement J McDonald, Stanley M Huff, Jeffrey G Suico, Gilbert Hill, Dennis Leavelle, Raymond Aller, Arden Forrey, Kathy Mercer, Georges DeMoor, John Hook, et al. Loinc, a universal standard for identifying laboratory observations: a 5-year update. *Clinical chemistry*, 49(4):624–633, 2003.

[23] Gunther Schadow and Clement J McDonald. The unified code for units of measure. *Regenstrief Institute and UCUM Organization: Indianapolis, IN, USA*, 2009.

[24] Danilo Freire de Souza Santos, Russell Berkoff, Clarke Stevens, Park Jangwoong, and Paul Jeon. Iot management and control:1 device for UPnP, version 1.0, standardized DCP (SDCP). *UPnP Forum Standardized DCPs, available at http://upnp.org/specs/smgt/smgt1/*, 2013.

[25] Danilo Freire de Souza Santos, Frederico Bublitz, Hyggo Almeida, and Angelo Perkusich. Integrating IEEE 11073 and Constrained Application Protocol for Personal Health Devices. In *Proceedings of the 2014 ACM Symposium On Applied Computing*. ACM, 2014.

[26] Danilo Freire de Souza Santos, A. Perkusich, and H.O. Almeida. Standard-based and distributed health information sharing for mHealth IoT systems. In *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pages 94–98, 2014.

editoraIFPB