Lucas Jeferson da Costa Silva

Tiúba Wi-Fi Manager: Uma proposta para o gerenciamento da rede Wi-Fi da Vila de Contêineres da UEMA

São Luís

2019

Lucas Jeferson da Costa Silva

Tiúba Wi-Fi Manager: Uma proposta para o gerenciamento da rede Wi-Fi da Vila de Contêineres da UEMA

Projeto apresentado ao curso de Mestrado Profissional em Engenharia da Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para a obtenção do título de mestre sob orientação do Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva.

Universidade Estadual do Maranhão

Centro de Ciências Tecnológicas

Programa de Pós-graduação em Engenharia de Computação e Sistemas

Orientador: Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva

Coorientador: Prof. Dr. Rogério Moreira Lima Silva

São Luís

2019

Silva, Lucas Jeferson da Costa.
Tiúba wi-fi manager: uma proposta para o gerenciamento da rede wi-fi da vila de contêineres da Universidade Estadual do Maranhão / Lucas Jeferson da Costa Silva. – São Luís, 2019.

109f

Dissertação (Mestrado) – Curso de Engenharia de Computação e Sistemas, Universidade Estadual do Maranhão, 2019.

Orientador: Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva.

1.Gerenciamento. 2.OpenWrt. 3.wi-fi. 4.Segurança.. I.Título

CDU: 004.738(812.1)

Elaborado por Giselle Frazão Tavares- CRB 13/665

Lucas Jeferson da Costa Silva

Tiúba Wi-Fi Manager: Uma proposta para o gerenciamento da rede Wi-Fi da Vila de Contêineres da UEMA

Projeto apresentado ao curso de Mestrado Profissional em Engenharia da Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para a obtenção do título de mestre sob orientação do Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva.

Trabalho aprovado. São Luís, 10 de setembro de 2019:

Prof. Dr. Leonardo Henrique Gonsioroski Furtado da Silva
Presidente

Prof. Dr. Rogério Moreira Lima Silva
Convidado Interno

Prof. Dr. Luís Carlos Costa Fonseca
Convidado Interno

Prof. Dr. Lúcio Flávio de Albuquerque Campos
Convidado Externo

São Luís
2019

Este trabalho é dedicado à minha família
que sempre me incentivou e me deu o suporte necessário
para a realização desse sonho.

Agradecimentos

Os agradecimentos principais são, em primeiro lugar, a Deus que nos dá a graça diária de crescer através das dificuldades e nos ensina a ter confiança em Sua providência mesmo quando tudo parece não ter solução.

Agradeço aos meus familiares e à minha namorada, que foram o meu suporte em todos os momentos em que precisei de incentivo e motivação para continuar firme nessa caminhada.

Agradeço aos meus orientadores por me ajudarem a ver qual o melhor caminho a seguir e por darem o apoio intelectual e criativo para produzir este trabalho.

"E disse-me: A minha graça te basta, porque o meu poder se aperfeiçoa na fraqueza. De boa vontade, pois, me gloriarei nas minhas fraquezas, para que em mim habite o poder de Cristo.
(Bíblia Sagrada, 2 Coríntios 12, 9)

Resumo

Este trabalho trata sobre o problema de acesso à rede Wi-Fi na Vila de Contêineres da UEMA, devido a dificuldade de propagação de sinal de rádio da tecnologia em questão, o alto custo para implantação utilizando a solução de rede Wi-Fi proprietária adotada pela universidade e o problema da utilização de dispositivos não gerenciados (como forma de suprir a necessidade de acesso Wi-Fi). Diante disso, foi desenvolvido uma solução envolvendo o desenvolvimento do protótipo de um software para realizar gerenciamento de dispositivos Wi-Fi com hardware menos robusto e baseado em um sistema derivado do Linux, o Openwrt. Para entender melhor como a solução funciona, foi realizada uma fundamentação teórica com o objetivo de introduzir o leitor nas tecnologias empregadas na solução e também foi dada uma breve explicação sobre o funcionamento do software desenvolvido e como ele interage com os dispositivos baseados em Openwrt. Este trabalho se torna relevante ao mercado por obordar um método de configuração de dispositivos wi-fi de forma não convencional, utilizando tecnologias de código livre, possibilitando implementação de novas funcionalidades e produção de propriedades intelectuais. Ele também traz um impacto à comunidade em geral, pois oferece uma forma de gerenciamento de pontos de acesso wi-fi mais barato, acessível para usuários residenciais e de pequenas empresas.

Palavras-chave: Wi-Fi; Gerenciamento; Openwrt.

Abstract

This paper deals with the problem of access to the Wi-Fi network in UEMA's Container Village, due to the difficulty of radio signal propagation of the technology in question, the high cost for deployment using the proprietary Wi-Fi network solution adopted by university and the problem of using unmanaged devices (as a way to meet the need for Wi-Fi access). Therefore, a solution was developed involving the prototype development of software to manage Wi-Fi devices with less robust hardware and based on a Linux-derived system, Openwrt. To better understand how the solution works, a theoretical grounding was introduced with the aim of introducing the reader to the technologies employed in the solution and a brief explanation of how the software developed and how it interacts with Openwrt-based devices. This work becomes relevant to the market because it obeys a method of configuring wireless devices in an unconventional way, using open source technologies, enabling the implementation of new features and production of intellectual properties. It also has an impact on the wider community by providing a cheaper way of managing Wi-Fi hotspots that is affordable for home and small business users.

Keywords: Wi-Fi; Management; Openwrt.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AES	Advanced Encryption Standart
AGA	Assessoria de Gestão Ambiental
AP	Ponto de Acesso - *Access Point*
API	Application Programming Interface
BIOPESQ	Laboratórios de Biologia Pesqueira
BSS	Basic Service Set
BPSK	Binary Phase Shift Keying
CCMP	Counter Mode with CBC-MAC
CHAP	Challenge Handshake Authentication Protocol
CSI	Camera Serial Interface
DFC	Dynamic Frequency Control
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DRY	Don't Repeat Yourself
DS	Distribution System
DSI	Display Serial Interface
DSS	Distribution System Service
DSSS	Direct-Sequence Spread Spectrum
EAP	Extensible Authentication Protocol
EAPOL	Extensible Authentication Protocol over LAN
ESS	Extended Service Set
ETSI	European Telecommunications Standards Institute
FHSS	Frequency-Hopping Spread Spectrum

FISIOMAR	Fisioecologia, Reprodução e Cultivo de Organismos Marinhos
FTP	File Transfer Protocol
GPL	General Public License
GPU	Graphics Processing Unit
GTC	Generic Token Card
HPE	Hewlett-Packard Enterprise
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IAP	Instant Access Point
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
IR	Infra Red
IV	Initialization Vector
JSON	JavaScript Object Notation
LAN	Local Area Network
LCC	Logical Link Control
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MAN	Rede Metropolitana
MVC	Model-view-controller
NAS	Network Access Server
NAT	Network Address Translation
NIS	Network Information Service
NVRAM	Non-Volatile Random Access Memory
NVT	Network Virtual Terminal

OWASP	Open Web Application Security Project
OFDM	Orthogonal Frequency Division Multiplexing
PAM	Pluggable Authentication Modules
PEAP	Protected Extensible Authentication Protocol
PHY	Physical Layer
PPP	Point to Point Protocol
PSK	Pre Shared Key
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RADIUS	Remote Authentication Dial In User Service
RPC	Remote Procedure Call
RSNA	Robust Security Network Association
RSS	Rich Site Summary
SoC	System on a Chip
SQL	Structured Query Language
SS	Station Service
SSH	Secure Shell
SSID	Service Set Identifier
SSL	Secure Socket Layers
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
TPC	Transmit Power Control
TTLS	Tunneled Transport Layer Security
TWM	Tiúba Wi-Fi Manager
UCI	Unified Configuration Interface

$UEMA$	Universidade Estadual do Maranhão
$UNII$	Unlicensed National Information Infrastructure
URL	Uniform Resource Locator
VAP	Virtual Access Point
$VLAN$	Virtual Local Area Network
WEP	Wired Equivalent Privacy
$WLAN$	Rede Local Sem Fio
WPA	Wi-Fi Protected Access
$WSGI$	Web Server Gateway Interface
XML	eXtensible Markup Language
XSS	Cross Site Scripting

Sumário

1 INTRODUÇÃO

As redes sem fio estão sendo cada vez mais utilizadas na comunicação entre dispositivos dos mais variados tipos e tamanhos e em diferentes ambientes. Por permitirem a mobilidade, estas redes facilitam a ubiqüidade do poder computacional, tornando transparente a disseminação da informação e a cooperação dos dispositivos na realização das mais variadas tarefas (VIEIRA, 2005).

As WLANs (*Wireless Local Area Network*) fazem com que seus usuários se conectem de forma flexível à redes sem cabos conectados e se tornou uma das melhores opções para obter comunicação sem fio.

Tendo em vista o fato de a mobilidade no acesso à rede de dados ser um fator que facilita e flexibiliza a pesquisa, a produção acadêmica, e a gestão administrativa no ambiente universitário, observamos a importância de oferecer acesso à rede Wi-Fi na Universidade Estadual do Maranhão - UEMA. Porém, alguns fatores como perda de sinal por conta de obstáculos que dificultam a propagação de rádio frequência e o preço da solução atualmente adotada pela universidade, dificultam a implantação de uma LAN sem fio de qualidade na região da Vila de Contêineres da UEMA.

Na visão do usuário, a necessidade é ter acesso à rede Wi-Fi com qualidade satisfatória, porém não podemos esquecer as necessidades dos administradores da rede, que trabalham para manter a infraestrutura de dados funcionando corretamente, para oferecer um ótimo serviço aos usuários. O administradores de rede sentem a necessidade de gerenciamento da rede como um todo, inclusive da rede Wi-Fi.

A partir disso, mostramos neste trabalho a proposta do *"Tiúba Wi-Fi Manager"*, um sistema para gerenciamento de dispositivos Wi-Fi baseados em OpenWrt. Tudo isso visando atender a necessidade de acesso à rede de dados com qualidade por parte do usuário, gerenciamento dos dispositivos e usuários conectados por parte dos administradores de redes, tudo isso com um custo menor que a solução Wi-Fi adotada como padrão na UEMA.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo auxiliar o entendimento no que se refere às tecnologias envolvidas no desenvolvimento do protótipo e no funcionamento da solução proposta para resolver o problema de cobertura Wi-Fi e gerenciamento de dispositivos Wi-fi na Vila de Contêineres da UEMA (*Universidade Estadual do Maranhão*), por meio

do desenvolvimento do sistema *Tiúba Wi-Fi Manager*.

1.1.2 Objetivos Específicos

•Desenvolver uma interface de gerenciamento dos APs (*Access Points*), através de uma interface web, possibilitando a configuração de dispositivos individualmente ou em grupos e a visualização dos usuários conectados em todos so APs;

•Fornecer segurança na comunicação entre o sistema Tiúba Wi-Fi Manager e os APs vinculados a ele por meio de protocolos criptográficos;

•Implementar funcionalidade de VLAN aliadas à múltiplos SSIDs, onde cada SSID está atrelado a uma VLAN.

1.2 Metodologia

A metodologia utilizada neste trabalho consiste na pesquisa de tecnologias de software livre e de código aberto, para serem embarcado em hardwares de baixo custo (menos que R$150,00) e utilizado na implementação do sistema que irá compor a solução proposta. Através de um estudo bibliográfico feito por meio de artigos, livros e documentos técnicos, definir quais melhor se adequam à implementação dos requisitos quem abrangem a necessidade em uma melhor cobertura de sinal Wi-Fi na região da Vila de Contêineres da UEMA, assim como a necessidade de gerenciamento centralizado dos equipamentos que serão utilizados. Com o estudo realizado, será mostrado como foi desenvolvido e como funciona o protótipo chamado do *Tiúba Wi-Fi Manager*, o sistema, ainda em desenvolvimento, que compõe a solução proposta para atender o problema em questão.

1.3 Justificativa

O que serviu como motivação para desenvolver esse trabalho foi o alto custo para se atender com qualidade a Vila de Contêineres da UEMA utilizando a solução de rede Wi-Fi adotada pela universidade, onde seriam necessários uma grande quantidade de APs para se atender um local com problemas de propagação de sinal e com uma baixa densidade de usuários.

As soluções comerciais oferecem uma grande quantidade de funcionalidades, além de ser composto por hardware com alto desempenho. Isso tudo acaba elevando o preço da maioria das soluções de rede Wi-Fi, como é o caso da solução oferecida pela Aruba Networks, que foi a adotada pela UEMA.

1.4 Aplicabilidade

Com a popularização de dispositivos móveis e o alto consumo de dados, devido ao crescente avanço nos serviços oferecidos pela internet, estar conectado passa a ser algo essencial na vida do ser humano. Esse projeto tem como objetivo oferecer uma solução de acesso à rede Wi-Fi aliada a um gerenciamento simplificado, voltado para ambientes onde há deficiência de cobertura de sinal com a presença de um único AP, fazendo-se necessária presença de múltiplos APs Wi-Fi.

1.5 Estrutura do Documento

O trabalho está estruturado da seguinte maneira. No Capítulo 2 é apresentado padrão IEEE 802.11, seus derivados e formas de funcionamento. O Capítulo 3 apresenta o sistema para dispositivos embarcados OpenWrt. O Capítulo 4 apresenta o framework Web2py, utilizado no desenvolvimento do sistema. O Capítulo 5 faz um aprofundamento sobre o protocolo SSH. O Capítulo 6 apresenta as Remote Procedure Calls. O Capítulo 7 nos ajuda a conhecer um pouco sobre a Vila de Contêineres da UEMA. O Capítulo 8 conheceremos o sistema desenvolvido, o Tiúba Wi-Fi Manager. Por fim, nos dois capítulos seguintes serão mostradas as conclusões do trabalho e os trabalhos futuros.

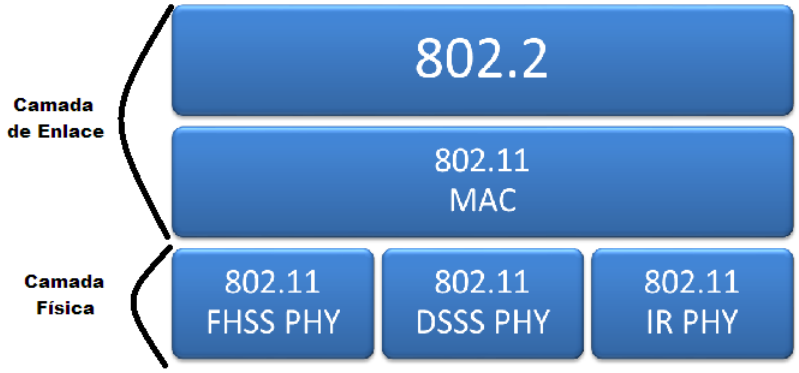
2 PADRÃO IEEE 802.11

A comunicação sem fios é uma das tecnologias que mais tem crescido nos últimos anos. A demanda pela conexão de dispositivos sem a utilização de cabos aumentou vertiginosamente em todo o mundo. Atualmente, as LANs sem fios são encontradas em campus universitários, escritórios de empresas e em áreas públicas (FOROUZAN, 2006).

> "Cada vez mais organizações descobrem que as LANs sem fio são um complemento indispensável para as LANs com fio tradicionais, para atender necessidades de mobilidade, relocação, interligação, rede provisória e cobertura de locais difíceis de ligar" (STALLINGS, 2005).

A equipe de trabalho do IEEE divulgou em 1997 as especificações do protocolo definidas pelos padrões IEEE 802.11, que fazem parte da família de padrões 802, responsável por especificar padronizações para redes LAN e MAN. No padrão 802.11 é feito o detalhamento da camada física (PHY) e da subcamada MAC (camada de enlace no padrão IEEE 802), como pode ser vista na Figura 1.

Figura 1 – Camada MAC e PHY



Fonte: Adaptado de GARCIA (2003).

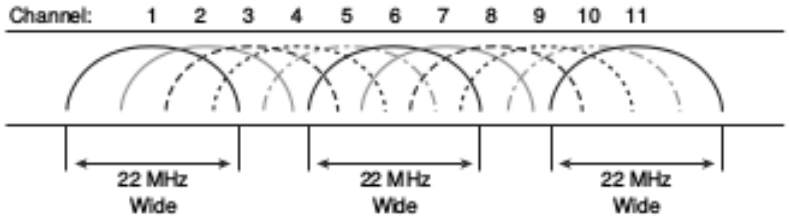
O padrão IEEE 802.11 prevê que a camada física poderá usar até três tipos de métodos de transmissão: FHSS, DSSS[1] e o Infra Vermelho (IR). Os padrões FHSS e DSSS operam na frequência de 2.4 GHz, sendo que a maioria das aplicações utilizam a técnica de DSSS (VINCI; FERREIRA, 2007).

Segundo CARROLL (2009), a faixa de frequência de 2.4GHz é, provavelmente, a mais utilizada em WLANs. Sendo utilizada nos padrões IEEE 802.11, 802.11b, 802.11g e

[1] *Frequency-Hopping Spread Spectrum* e *Direct-Sequence Spread Spectrum*: são técnicas de codificação para transmissão digital de sinais

802.11n. A faixa de frequência de 2,4 GHz vai de 2400 MHz a 2483.5 MHz ou 2495 MHz (dependendo da regulamentação país). Nos Estados Unidos e no Brasil é utilizado a faixa de 2400 MHz a 2483.5 MHz, dispostos em 11 canais de largura de banda de 22 MHz, como pode ser visto na Figura 2. Nesse cenário, estão disponíveis 11 canais, onde alguns deles se sobrepõem, podendo causar interferência entre si. Por essa razão, os canais 1, 6 e 11 são os mais utilizados, por eles não estarem sobrepostos.

Figura 2 – Canais em 2.4GHz



Fonte: (CARROLL, 2009).

A faixa de 5 GHz é utilizada pelos padrões IEEE 802.11a, 802.11n e 802.11ac. Essa faixa de frequência foi utilizada inicialmente no padrão 802.11a, onde as taxas de transmissão variam de 6 Mbps a 54 Mbps. Algum tempo depois, a faixa de 5 GHz foi utilizada nos padrões 802.11n, que poderia usar tanto a faixa de 2.4 GHz quanto a de 5 GHz, e no padrão 802.11ac, que utiliza exclusivamente essa frequência de transmissão.

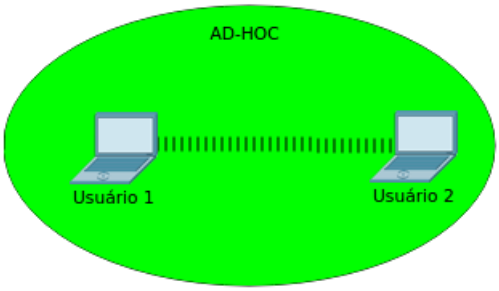
2.1 Arquitetura IEEE 802.11

A arquitetura definida para este padrão é celular, onde cada célula, denominada de *Basic Service Set* (BSS), é controlada por um *Access Point* (VINCI; FERREIRA, 2007).

Existem três tipos de topologia nas quais é possível formar uma rede WLAN, o tipo de topologia é classificado de acordo com a forma que os dispositivos se comunicam. Os três tipos de topologias são IBSS (também conhecido como AD-HOC), BSS (o mais utilizado nas nossas residências) e o ESS (mais utilizadas em ambientes mais extensos ou com muita interferência). A seguir temos uma descrição mais detalhada de cada uma dessas topologias:

- **IBSS (*Independent Basic Service Sets*):** trata-se de um grupo de estações que se comunicam de forma direta entre si, sem a necessidade de um AP para controlar o acesso, como pode ser visto na Figura 3. Esse tipo de topologia é também conhecida como AD-HOC.

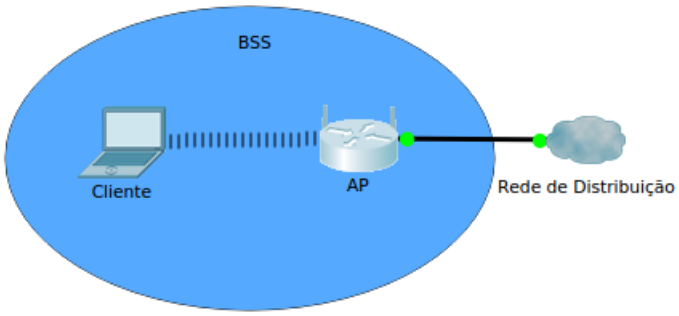
Figura 3 – Representação de um IBSS



Fonte: O autor.

- **BSS (*Basic Service Set*):** é um grupo de estações móveis que se comunicam em uma célula, tendo o AP como um ponto de conexão comum, como pode ser visto na Figura 4 Esse AP deve estar conectado à rede de distribuição por meio de uma rede cabeada ou até mesmo por conexões sem fios.

Figura 4 – Representação de uma BSS

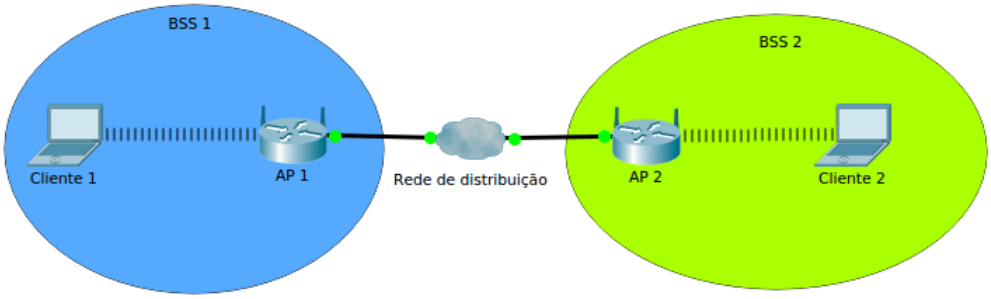


Fonte: O autor.

- **ESS (*Extended Service Sets*):** nessa topologia, temos duas ou mais estruturas de BSS conectadas através de uma rede de distribuição (uma infraestrutura de rede que pode ser cabeada, sem fios ou fibra óptica), como pode ser visto na Figura 5.

Dependendo de como os APs estão dispostos no ambiente, é possível que as estações possam passar da área de cobertura de uma BSS para a área de cobertura de outra de forma transparente, processo esse denominado de *roaming*.

Figura 5 – Representação de uma ESS



Fonte: O autor.

2.2 Serviços IEEE 802.11

O padrão IEEE 802.11 especifica nove serviços diferentes, divididos em duas categorias: Serviços de Estação (SS - *Station Services*) e os serviços de sistema de distribuição (DSS - *Distribution System Service*), ambos os serviços utilizados pela camada MAC (VINCI; FERREIRA, 2007).

2.2.1 Serviços de Estação

Os serviços de estação devem ser implementados em todas as estações que seguem o padrão IEEE 802.11. Como há casos em que os APs também podem ser estações, se faz necessário a implementação desse serviço nos mesmos (VINCI; FERREIRA, 2007).

Nas redes cabeadas, a segurança é relativamente garantida devido ao meio de transmissão confinado (cabos) e pela segmentação do domínio de colisão oferecido pelos *switches*. Essa segurança não pode ser garantida de forma tão natural nas redes WLAN, por esse motivo, os seguintes serviços são necessários:

- **Autenticação:** Usada para estabelecer a identidade das estações, de uma para outra. Em uma rede LAN, geralmente se considera que o acesso a uma conexão física significa autorização para se conectar à LAN. Essa não é uma suposição válida para uma WLAN, na qual a conectividade é obtida simplesmente com uma antena corretamente sintonizada. O serviço de autenticação é usado pelas estações para estabelecer sua identidade em estações com as quais elas irão se comunicar. O padrão

IEEE 802.11 não exige qualquer esquema de autenticação específico, que pode variar de um *handshaking* a esquemas de criptografia de chave pública (STALLINGS, 2005).

- **Desautenticação:** Uma notificação por parte de uma estação ou AP de que uma associação existente será finalizada. Assim como é necessário a autenticação para que a estação tenha acesso efetivo à rede, é necessário que a estação emita uma notificação quando for deixar a área da BSS ou desligar. Entretanto, o recurso de gerenciamento do MAC tem a função de se proteger contra estações que desaparecem sem enviar a notificação de desautenticação (STALLINGS, 2005).

- **Privacidade:** Com o objetivo de impedir que as trocas de mensagens entre estação e AP possam ser lidos por outros dispositivos além do destinatário pretendido. Para garantir a privacidade na troca de mensagens, o padrão sugere o uso de criptografia (STALLINGS, 2005).

- ***Delivery*:** Garantia de entrega das informações entre os dispositivos, através da identificação por endereço MAC (VINCI; FERREIRA, 2007).

2.2.2 Serviços de Sistema de Distribuição

Em redes de estrutura do tipo ESS, a responsabilidade de encontrar os destinatário dos dados que trafegam pela rede é de responsabilidade do sistema de distribuição, mas que o envio e recebimento de dados pelos dispositivos da rede *wireless* ocorra é necessário que se os APs ligados ao sistema de distribuição implementem os serviços de que trabalham nas subcamadas LCC e MAC. Estes serviços são:

- **Associação:** Esse serviço é o responsável por permitir que uma estação envie e receba dados pela rede através da associação à um AP, onde o processo de associação é sempre iniciado pela estação. Ainda que uma estação possa estar autenticada em mais de um AP, ele pode estar associado somente a um AP, que é o mais importante para o sistema de distribuição quando este necessita descobrir em qual AP a estação está associada.

- **Desassociação:** Este serviço serve para informar ao AP quando uma estação não continuará associada à ele, podendo ser efetuado tanto pela estação como pelo AP. Esse serviço é importante para que não ocorra o envio de *frames* de forma equivocada para um AP que não possui mais a estação destinatária associada a ele.

- **Reassociação:** Geralmente por questões de economia de energia, uma estação pode não ficar continuamente associada a um AP, onde se faz necessário a existência do serviço de reassociação, que é responsável por informar ao sistema de distribuição o atual AP ao qual a estação está associada, sendo invocado sempre pela estação quando esta se reassocia à um AP.

- **Integração:** O serviço de integração tem a função de conectar as redes do padrão IEEE 802.11 à outros tipos de redes LAN, podendo ser uma ou mais LANs ou outra WLAN. Este serviço consiste em traduzir *frames* IEEE 802.11 para *frames* da rede destino e vice-versa (O'HARA; PETRICK, 2005).
- **Distribuição:** determina como será a entrega dos *frames*. Esse serviço determina que o sistema de distribuição ficará responsável pela entrega dos frames aos seus respectivos destinatários, mesmo que a estação destino esteja associada ao mesmo AP em que a estação de origem está associada.

2.2.3 Estados de uma estação

O padrão IEEE 802.11 define que os dispositivos que sigam esse padrão devem implementar os serviços anteriormente citados. Com isso, em algum momento do seu funcionamento, toda estação poderá assumir um dos três estados que podem ser visualizados na Figura 6, onde é possível ver melhor a função de alguns dos serviços que já foram mostrados.

Figura 6 – Estados que uma estação pode assumir

Fonte: (VINCI; FERREIRA, 2007).

- **Estado 1:** Quando a estação encontra-se nesse estado, ela não está associada e nem autenticada a nenhum AP. O próximo passo seria realizar autenticação em algum AP disponível no local.
- **Estado 2:** Nesse estado, a estação já realizou a autenticação com sucesso, porém ainda não está associada ao AP. Quando está nesse estado, a estação tem a opção de realizar a associação para realizar a transmissão dos dados ou realizar a desautenticação para que ele volte para o Estado 1.

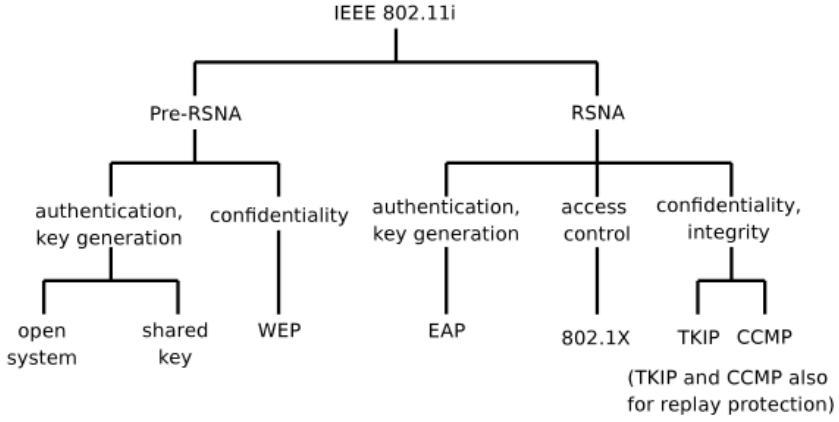
- **Estado 3:** É nesse estado que a estação está apta a transmitir dados aos outros dispositivos da rede, onde a estação encontra-se autenticada e associada à um AP. Para sair desse estado, a estação pode ser desautenticada para voltar ao Estado 1 ou ser desassociado para voltar ao Estado 2, em ambos os casos o serviço pode ser solicitado tanto pela estação quanto pelo AP.

2.3 Autenticação em redes WLAN

Inicialmente o padrão IEEE 802.11 não especificava nenhum método de que garantisse segurança no acesso e transmissão dos dados, onde o controle de acesso era feito por endereço MAC em dispositivos de alguns fabricantes. Como fazer o controle de acesso por endereço MAC era um muito trabalhoso e de difícil manutenção, foi introduzido no padrão IEEE 802.11b o método de segurança WEP (*Wired Equivalent Privacy)*, que sofria de várias falhas. Com esses problemas sofridos pelo WEP, o IEEE resolveu desenvolver o RSNA (*Robust Security Network Association*), especificado pelo padrão 802.11i, que servia como um aperfeiçoamento para prover uma forma mais robusta de segurança no controle de acesso. O RSNA foi idealizado para prover autenticação, geração de chaves criptografadas, integridade e confidencialidade.

O padrão 802.11i demorou um tempo para ser totalmente especificado e, enquanto isso não acontecia, a Wi-Fi Alliance criou uma solução temporária para substituir o WEP, foi aí que nasceu o WPA (WiFi *Protected Access*). O WPA já implementava parte das especificações do padrão 802.11i, com os algoritmos de segurança que foram denominados Pre-RSNA. Quando o padrão 802.11i foi finalizado, a Wi-Fi Alliance lançou o método de segurança WPA2, que é conhecido até hoje pela a sua eficácia e amplamente utilizado nos diversos padrões IEEE 802.11 desenvolvidos até então.

Figura 7 – Resumo do padrão IEEE 802.11i



Fonte: (HOLT; HUANG, 2010).

Na Figura 7 temos um resumo de como o padrão IEEE 802.11i está organizado no seu objetivo de prover autenticação, geração de chaves criptografadas, integridade e confidencialidade.

2.3.1 Autenticação Aberta

Conhecida como *open system*, este método de autenticação não possui nenhuma forma de segurança no controle de acesso das estações à rede. Para fazer a autenticação, considerando dois dispositivos A e B, o dispositivo A afirma sua identidade para o dispositivo B através de uma requisição de autenticação. O dispositivo B envia para A o resultado da requisição, que pode ser *"success"* ou *"failure"*. O dispositivos estarão mutuamente autenticados caso o resultado seja *"sucess"* (HOLT; HUANG, 2010).

Como não foi definido originalmente nenhum tipo de segurança para controlar a autenticação das estações, foi implementado por alguns fabricantes uma lista de autorização acesso baseada em endereço MAC, onde somente os dispositivos com o endereço MAC contido na lista recebiam a resposta *"sucess"* pela requisições de autenticação feitas ao AP. Essa implementação não está contida na especificação original do padrão IEEE 802.11.

2.3.2 Wired Equivalent Privacy (WEP)

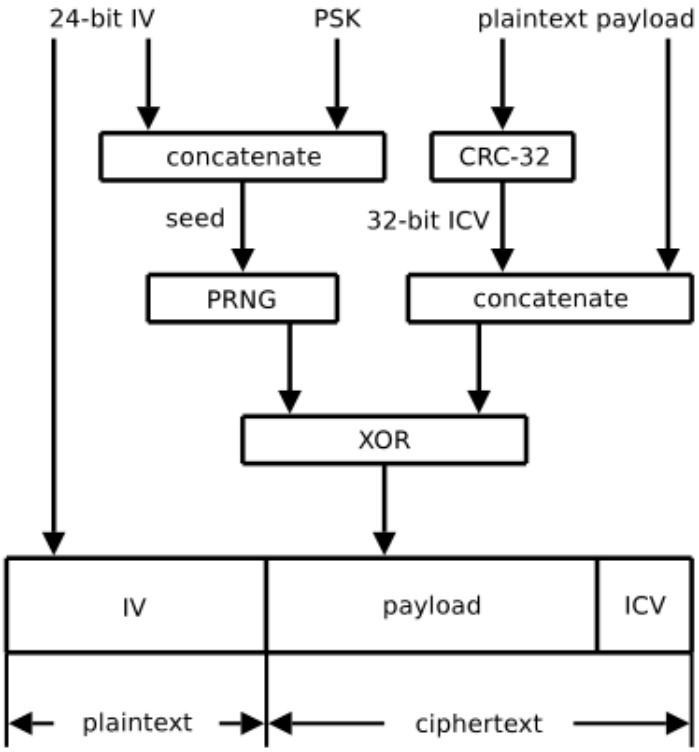
Com o intuito original de prover o mesmo nível de confidencialidade que era extinta nas redes cabeadas tradicionais, o método de autenticação WEP foi inicialmente especificado no padrão IEEE 802.11b com o objetivo de fornecer confidencialidade aos dados por meio de um algoritmo criptográfico de chave secreta, ser eficiente a ponto de ser capaz de ser processado rapidamente por software e ser exportável a ponto de poder ser utilizado em outros países além das fronteiras americanas (MORAES, 2010).

Segundo MORAES (2010), o WEP trabalha com o algoritmo RC4[2] da RSA que é uma cifra baseada em *stream* de fluxo, onde um algoritmo simétrico uma única chave para criptografar e descriptografar os dados.

Na Figura 8 é possível ver como é montado o *frame* WEP, onde a chave utilizada é uma chave de 40 bits que é concatenada com uma vetor de inicialização de 24 bits para formar a chamada *seed* ou semente em tradução livre. Toda vez que um texto claro vai ser enviado, ele é submetido à uma operação XOR com uma *stream* de dados pseudorrandômicos gerado pela semente, onde o resultado dessa operação é novamente concatenado com o vetor de inicialização e transmitido. O processo de decriptação é inverso ao que acabou de ser descrito.

[2] É um algoritmo simétrico de criptografia de fluxo mais usado no *software* e utilizado nos protocolos mais conhecidos, como *Secure Socket Layers* (SSL)

Figura 8 – Montagem do *frame* WEP



Fonte: (HOLT; HUANG, 2010).

Por conter algumas vulnerabilidades que comprometiam a segurança das informações que trafegam na rede, o WEP foi definido pela Universidade de Berkeley na Califórnia e pela Universidade de Maryland como inadequado para prover privacidade em redes sem fio em camada de enlace.

2.3.3 WiFi Protected Access (WPA)

Criado em 2003 pela Wi-Fi Alliance, o WPA tinha como objetivo encontrar e tentar corrigir as falhas do WEP, garantindo uma forma temporária de melhorar a segurança das WLANs enquanto o padrão IEEE 802.11i era finalizado. Para isso, foi implementada uma melhor forma de gerenciamento de chaves utilizando cifra RC4 através do TKIP (*Temporal Key Integrity Protocol*).

O funcionamento do TKIP é baseado na criação de uma chave temporal de 128 bits, que é compartilhada entre todos as estações e o AP. Essa chave temporal é combinada com o endereço MAC da estação para ajudar compor o vetor de inicialização, que será utilizado para a encriptação dos dados. Para melhorar a segurança, ocorre uma distribuição dinâmica dessas chaves temporais, onde as chaves temporais são trocadas a cada 10.000 pacotes.

O WPA já seguia algumas especificações de uma versão não terminada do padrão 802.11i, que definia que o WPA deveria trabalhar de duas formas:

- **WPA *Personal*:** utiliza o método PSK (*Pre Shared Key*), que é uma chave pré-compartilhada, previamente instalada no AP e nas estações para que se faça a autenticação das estações, seguido do processo de troca de chaves através do TKIP.
- **WPA *Enterprise*:** esse método de autenticação utiliza o padrão IEEE 802.1x, que é um protocolo de autenticação, baseado em porta, que utiliza o EAP (*Extensible Authentication Protocol*) (MORAES, 2010).

2.3.3.1 Extensible Authentication Protocol (EAP)

Especificado pela RFC 3748, o EAP foi muito utilizado nos anos de 1990 para prover uma forma segura de autenticação para o protocolo PPP(Point-to-Point Protocol). Apesar de ser mais utilizado no PPP, o EAP é um simples encapsulamento que pode funcionar sobre qualquer protocolo da camada de enlace, podendo utilizar uma ampla variedade de métodos de autenticação, das quais podemos citar:

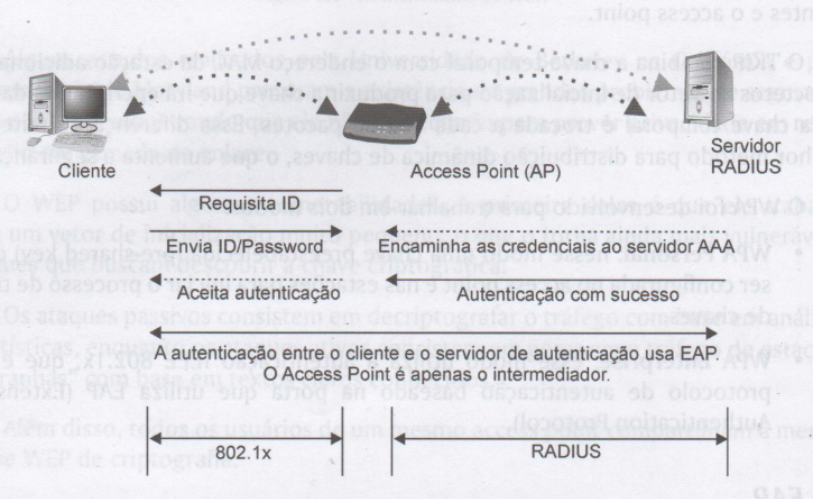
- **MD5 *Challenge*:** funciona como um *handshake* aplicado ao EAP;
- **GTC:** utiliza cartões com códigos *token* para prover autenticação;
- **TLS:** utilização de certificados, tanto do lado do servidor como do lado do cliente, para prover autenticação mútua;
- **TTLS:** utilização de certificados para autenticação apenas do lado do servidor, somado a utilização de tunelamento para garantir um canal seguro para o tráfego de dados. Esse método pode ser utilizado em conjunto com outros métodos de autenticação mais fracos;
- **PEAP:** similar ao TTLS, podendo, também, ser utilizado em conjunto com outros métodos de autenticação mais fracos;
- **EAP-SIM:** autenticação utilizada em telefonia móvel para garantir a identidade do assinante;
- **MS-CHAP-V2:** autenticação por senha criptografada desenvolvida pela Microsoft e amplamente utilizada em suas tecnologias.

No âmbito de redes sem fio, o EAP é utilizado para prover a segurança na autenticação especificada pelo padrão 802.1x, onde o EAP é encapsulado em camada 2 e fica conhecido como EAPOL (*EAP over LAN*).

2.3.3.2 IEEE 802.1x

Segundo HOLT e HUANG (2010), o 802.1x é um protocolo de camada 2 que suporta controle de acesso à rede baseado em porta.

Figura 9 – Processo de autenticação do 802.1x



Fonte: (MORAES, 2010).

O processo de autenticação do 802.1x, descrito na Figura 9, mostra a estação do cliente fazendo a requisição para acessar os recursos da rede e o AP fazendo o papel de intermediador da autenticação, responsável por fazer o controle do acesso à rede e enviar as credenciais ao servidor RADIUS. O servidor RADIUS, por sua vez, fará a validação dessas credenciais e, de acordo com a sua base de dados, retornará o resultado ao AP. De acordo com a resposta fornecida pelo servidor RADIUS, o AP fará ou não a liberação do acesso aos recursos da rede para a estação cliente.

O processo de comunicação entre a estação cliente e o AP é feita através do protocolo 802.1x baseado no EAP, já a comunicação entre o AP e o servidor de autenticação é feita através do protocolo RADIUS.

Uma das vantegens de utilizar o RADIUS é que ele oferece suporte a uma variedade de diferentes bases de usuários. Além de utilizar base de dados locais, o RADIUS server pode ser usado como um gateway para diretórios LDAP, faz autenticações UNIX como NIS e PAM, domínios Kerberos, *Active Directory* da Microsoft, ou até mesmo em outros servidores RADIUS (GAST, 2005).

2.3.4 Wi-Fi Protected Access 2 (WPA2)

Baseado na especificação final do padrão IEEE 802.11i, o WPA2 foi lançado pela Wi-Fi Alliance em 2004, fornecendo à usuários domésticos e empresas um maior grau de

segurança, mantendo a compatibilidade com o WPA, ou seja, implementa o algoritmo TKIP e a autenticação por 802.1x.

A novidade é que o WPA2 passou a utilizar o algoritmo CCMP (*Counter Mode with CBC-MAC*), que é considerado pelo governo dos Estados Unidos como um dos mais seguros existentes por ser baseado na especificação final do AES (*Advanced Encryption Standart*).

Assim como WPA, o WPA2 oferece dois métodos de autenticação, um baseado em uma chave pré-compartilhada e outro baseado em autenticação 802.1x.

2.4 Família de protocolos do padrão IEEE 802.11

De acordo com (CARROLL, 2009), a padronização da rede sem fio começou com o protocolo 802.11 original em 1997, e cada protocolo depois disso simplesmente contribuiu para o benefício das tecnologias sem fio. A seguir, analisaremos as famílias de protocolos do 802.11, sua história e como elas operam. Os protocolos 802.11 funcionam nos ranges de frequência não licenciada de 2,4GHz e 5GHz.

2.4.1 O protocolo 802.11 original

(CARROLL, 2009) afirma que o protocolo 802.11 original era onde as LANs sem fio encontravam seus primeiros passos. É raro encontrar este protocolo original em novo hardware hoje, provavelmente porque ele opera somente em 1 e 2 Mbps. O padrão 802.11 descreve a utilização do espectro de propagação por saltos de frequência (FHSS), que opera somente em 1 e 2 Mbps. O padrão também descreve a utilização do espectro de dispersão de seqüência direta (DSSS), que opera apenas em 1 e 2 Mbps. Se um cliente opera em qualquer outra taxa de dados, ele é considerado não-compatível com 802.11, mesmo que possa usar as taxas de 1 e 2 Mbps.

O protocolo 802.11 original, de acordo com (CARROLL, 2009), se enquadra nas faixas industriais, científicas e médicas (ISM) e opera somente na faixa de 2,4 GHz. A faixa de 2,4 GHz tem até 14 canais, dependendo do país em que você está. Nos Estados Unidos, a FCC permite que os canais 1 a 11 sejam usados. Isso fornece 3 canais não sobrepostos: 1, 6 e 11. Isso é importante porque você não deseja que APs e clientes que operam no mesmo canal sejam colocados próximos uns dos outros por motivos de interferência. A Tabela 1 mostra as principais características do protocolo:

Tabela 1 – Características do protocolo 802.11 original

Ratificado	1997
Tecnologia de RF	FHSS e DSSS
Espectro de Frequência	2.4-GHz

Fonte: Adaptado de (CARROLL, 2009).

2.4.2 O protocolo 802.11a

De acordo com (CARROLL, 2009), o 802.11a foi ratificado em 1999 e opera na faixa de frequência de 5 GHz. Isso faz com que seja incompatível com 802.11, 802.11be 802.11g, evitando a interferência desses dispositivos, além de microondas, dispositivos Bluetooth e telefones sem fio. O 802.11a teve adoção tardia no mercado, por isso não é tão amplamente implementado quanto os protocolos 802.11b e 802.11g. Outra diferença é que o 802.11a suporta de 12 a 23 canais não sobrepostos, em oposição aos 3 canais não sobrepostos do 802.11b/g. Como o OFDM é usado, os subcanais podem se sobrepor. O 802.11a requer que as taxas de dados de 6, 12 e 24 Mbps sejam suportadas, mas permite taxas de dados de até 54 Mbps.

(GROUP et al., 1999b) afirma que este protocolo foi inicialmente voltado para as faixas de estruturas nacionais não licenciadas de 5.15–5.25, 5.25–5.35 e 5.725–5.825 GHz (U-NII), conforme regulamentado nos Estados Unidos pelo Código de Regulamentos Federais, Título 47, Seção 15.407. O sistema OFDM fornece uma LAN sem fio com capacidade de comunicação que varia entre 6, 9, 12, 18, 24, 36, 48 e 54Mbit/s. Com suporte para transmitir e receber em taxas de 6, 12 e 24 Mbit/s é obrigatório. O sistema usa 52 subportadoras que são moduladas usando (BPSK/QPSK), 16-QAM ou 64-QAM. Codificação de correção de erro de encaminhamento (codificação convolucional) é usada com uma taxa de codificação de 1/2, 2/3 ou 3/4.

Na Tabela 2 podemos ver as principais características do protocolo 802.11a:

Tabela 2 – Características do protocolo 802.11a

Ratificado	1999
Tecnologia RF	OFDM
Espectro de Freqüência	5,0 GHz
Codificação	convolução
Modulação	BPSK, QPSK, 16-QAM, 64-QAM dependendo da subportadora.

Continua

Tabela 2 – *Continuação*

Taxas de dados	6, 9, 12, 18, 24, 36, 48, 54 Mbps com canais OFDM não sobrepostos Cada banda tem um 4; o meio 8 é usado com 52 subportadoras em cada canal.

Fonte: Adaptado de (CARROLL, 2009).

(CARROLL, 2009) afirma que as regras das especificações do ETSI são um pouco diferentes. O ETSI permite 19 canais e requer que o controle de frequência dinâmica (DFC) e o controle de potência de transmissão (TPC) sejam usados. O que torna o 802.11a único é o modo como a banda de frequência de 5 GHz é dividida em várias partes. Essas partes, a Infraestrutura Nacional de Informações Não Licenciada (UNII), foram projetadas para diferentes usos. O UNII-1 foi projetado para uso interno com uma antena permanente. O UNII-2 foi projetado para uso interno ou externo com uma antena externa e o UNII-3 foi projetado para pontes externas e antenas externas. Na Tabela 3 podemos ver como são divididas as bandas de frequência UNII.

Tabela 3 – Bandas de frequência UNII

Banda	**Frequência**	**Uso**
UNII-1	5.15–5.25 GHz	(UNII Indoor) O FCC permite uso interno e externo.
UNII-2	5,25–5,35 GHz	(UNII Low) Exterior / interior com DFC e TPC.
UNII-3	5.725–5.825 GHz	(U-NII / ISM) O FCC permite uso interno e externo.

Fonte: Adaptado de (CARROLL, 2009).

2.4.3 O protocolo 802.11b

De acordo com (CARROLL, 2009), o 802.11b é um complemento do protocolo 802.11. O 802.11 foi rapidamente superado porque as redes com fio ofereciam 10 Mbps contra o 1 e 2 Mbps do 802.11. Os fornecedores desenvolveram métodos para obter taxas de dados mais altas. O perigo nos protocolos criados pelos fornecedores, é claro, é a interoperabilidade. O trabalho do IEEE era simplesmente definir um padrão que todos os fornecedores pudessem seguir com base nas implementações proprietárias que estavam usando.

(GROUP et al., 1999a) afirma que o IEEE 802.11b fornece taxas de dados de até 11Mb/s a 2,4GHz usando DSSS com modulação de codificação de código complementar (CCK) ou o algoritmo de codificação convolucional binária de pacote (PBCC) (que foi oficialmente ratificado pelo IEEE como uma alternativa ao CCK).

Também de acordo com (CARROLL, 2009), este padrão foi ratificado em setembro de 1999. Os Estados Unidos têm 11 canais, o mesmo que o 802.11. Na Europa, o ETSI define 13 canais, e o Japão tem 14. O 802.11b permite a mudança dinâmica de taxa (DRS) para permitir que os clientes mudem as taxas para taxas mais baixas à medida que se afastam de um AP e taxas mais altas à medida que se aproximam de um AP. A Tabela 4 mostra mais características do padrão 802.11b.

Tabela 4 – Características do protocolo 802.11b

Ratificado	1999
Tecnologia de RF	DSSS
Espectro de Frequência	2.4GHz
Codificação	Barker 11 e CCK
Modulação	DBPSK e DQPSK
Taxas de dados	1, 2, 5.5, 11 Mbps
Canais não sobrepostos	1, 6, 11

Fonte: Adaptado de (CARROLL, 2009).

2.4.4 O protocolo 802.11g

De acordo com (CARROLL, 2009), o IEEE ratificou o 802.11g em junho de 2003. Além das quatro taxas de dados do 802.11b, ele acrescentou mais oito. A taxa de dados máxima de 54 Mbps coloca 802.11g na mesma faixa de velocidade que 802.11a; no entanto, permanece no intervalo de frequência de 2,4 GHz.

Segundo (VASSIS et al., 2005), enquanto o IEEE 802.11b usa apenas tecnologia DSSS, o IEEE 802.11g usa DSSS, OFDM ou ambos na banda ISM de 2,4 GHz para fornecer altas taxas de dados de até 54 Mb/s. O uso combinado de DSSS e OFDM é obtido através do fornecimento de quatro diferentes camadas físicas. Essas camadas, definidas no padrão como físico de taxa estendida (ERPs), coexistem durante uma troca de quadros, de modo que o emissor e o receptor têm a opção de selecionar e usar uma das quatro camadas, desde que ambas a suportem. As quatro diferentes camadas físicas definidas no padrão IEEE 802.11g são as seguintes:

- **ERP-DSSS/CCK**: Esta é a camada física usada pelo IEEE 802.11b. A tecnologia DSSS é usada com modulação CCK. As taxas de dados fornecidas são as do IEEE 802.11b;
- **ERP-OFDM**: Esta é uma nova camada física, introduzida pelo IEEE 802.11g. O OFDM é usado para fornecer taxas de dados IEEE 802.11a na banda de 2,4 GHz;

- **ERP-DSSS/PBCC**: Esta camada física foi introduzida no IEEE 802.11b e fornece as mesmas taxas de dados que a camada física DSSS/CCK. Ele usa a tecnologia DSSS com o algoritmo de codificação PBCC. O IEEE 802.11g estendeu o conjunto de taxas de dados adicionando as de 22 e 33 Mb/s;

- **DSSS-OFDM**: foi criada uma nova camada física que usa uma combinação híbrida de DSSS e OFDM. O cabeçalho físico do pacote é transmitido usando o DSSS, enquanto a carga útil do pacote é transmitida usando o OFDM. O escopo dessa abordagem híbrida é cobrir os aspectos de interoperabilidade, conforme explicado para 15 slots, mesmo que o atributo ERP esteja desabilitado, o mais tardar. tecnologias.

(CARROLL, 2009) afirma que ainda existem apenas três canais não sobrepostos. Com OFDM, você deve ter cuidado com as saídas de energia; a energia precisa ser reduzida para lidar com os picos da técnica de modulação e ainda estar dentro dos regulamentos governamentais. A Tabela 5 mostra alguns detalhes sobre o 802.11g:

Tabela 5 – Características do protocolo 802.11g

Ratificado	junho de 2003
Tecnologia de RF	DSSS e OFDM
Espectro de Frequência	2,4 GHz
Codificação	Barker 11 e CCK
Modulação	DBPSK e DQPSK
Taxas de dados	1, 2, 5,5, 11Mbps com DSSS 6, 9, 12, 18, 24, 36, 48, 54Mbps com OFDM
Canais não sobrepostos	1, 6, 11

Fonte: Adaptado de (CARROLL, 2009).

(VASSIS et al., 2005) afirma que a sobrecarga do pacote de camada física de um pacote IEEE 802.11 consiste em duas partes: o preâmbulo do Protocolo de Convergência de Camada Física (PLCP) usado para sincronização e o cabeçalho PLCP que contém informações de pacote relacionadas à camada física. O grupo IEEE 802.11b percebeu que o preâmbulo PLCP é muito longo e adiciona uma sobrecarga considerável em um sistema WLAN. Portanto, uma opção para suportar um tipo mais curto de preâmbulo (chamado de preâmbulo curto, em contraste com o preâmbulo antigo) foi introduzida para reduzir a sobrecarga de pacotes e melhorar o desempenho da rede. Se tanto o remetente quanto o receptor suportarem esta opção, a comunicação é realizada usando o preâmbulo curto. O IEEE 802.11g recomenda o uso obrigatório da opção de preâmbulo curto.

Também de acordo com (VASSIS et al., 2005), em uma rede IEEE 802.11g, as

estações podem escolher entre 14 taxas de dados diferentes e quatro camadas físicas para transmitir um pacote da maneira mais eficiente. Essa multiplicidade de taxas de dados e camadas físicas aumenta os problemas de interoperabilidade. Para isso, é essencial classificar os diferentes tipos de estações que podem existir em uma rede IEEE 802.11g:

- **Estações ERP**: as estações que suportam a camada física do ERP-OFDM. Essas estações são equipadas com uma interface sem fio IEEE 802.11g pura;
- **Estações não-ERP que suportam preâmbulo curto**: Estas estações são equipadas com uma interface sem fio IEEE 802.11b de uma versão mais nova, que suporta até 11 Mb/s, mas seu firmware é atualizado para suportar o uso de preâmbulo curto;
- **Estações não-ERP que não suportam preâmbulo curto**: Estas estações são equipadas com uma interface sem fio IEEE 802.11b de uma versão mais antiga ou uma interface sem fio IEEE 802.11 que não suporta o uso de preâmbulo curto.

2.4.5 O protocolo 802.11n

Segundo (GROUP et al., 2009), o protocolo de rede Wi-Fi de geração sob o padrão IEEE 802.11n oferece novos desafios e oportunidades para fornecer dados em redes sem fio, sendo até quatro a seis vezes mais rápido do que os dispositivos Wi-Fi legados 802.11a/b/g anteriores. O IEEE 802.11n-2009 é uma emenda ao padrão de rede sem fio IEEE 802.11-2007. Destina-se a melhorar o rendimento da rede em relação aos padrões anteriores com um aumento significativo na taxa de dados líquida máxima de 54Mbit/s para 600Mbit/s (taxa de bits bruta ligeiramente maior incluindo, por exemplo, códigos de correção de erros e throughput máximo ligeiramente inferior) através da utilização de quatro fluxos espaciais em uma largura de canal de 40MHz.

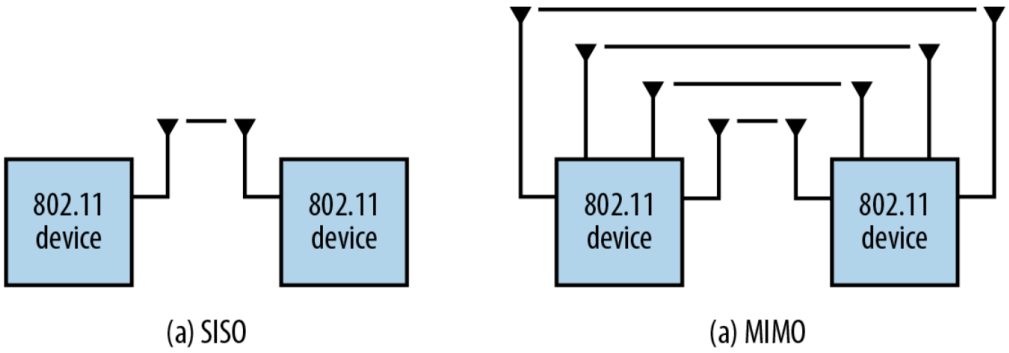
(SKORDOULIS et al., 2008) afirma que quando o IEEE 802.11g foi lançado para compartilhar a banda de espectro com os dispositivos legados 802.11a/b existentes, ele forneceu formas de garantir a coexistência entre os dispositivos legados e sucessores. O padrão IEEE 802.11n estende o gerenciamento de coexistência para proteger suas transmissões dos dispositivos legados, que incluem 802.11g, 802.11be 802.11a. Existem três mecanismos de proteção de nível MAC e PHY:

- **Proteção de nível PHY**: Proteção de Formato de Modo Misto (também conhecida como Proteção TXOP L-SIG): No modo misto, cada transmissão 802.11n é sempre incorporada em uma transmissão 802.11a ou 802.11g. Para transmissões de 20 MHz, essa incorporação cuida da proteção com 802.11a e 802.11g. No entanto, os dispositivos 802.11b ainda precisam de proteção CTS.

- **Proteção de nível PHY**: Transmissões usando um canal de 40 MHz na presença de clientes 802.11a ou 802.11g exigem o uso da proteção CTS nas metades de 20 MHz do canal de 40 MHz, para evitar a interferência com dispositivos legados.
- **Proteção do nível MAC**: Uma troca de quadros RTS/CTS ou transmissão de quadros CTS em taxas herdadas pode ser usada para proteger a transmissão 802.11n subsequente.

De acordo com (CARROLL, 2009), a compatibilidade retroativa e a capacidade de velocidade do 802.11n vêm do uso de múltiplas antenas e de uma tecnologia chamada MIMO (Multiple-Input, Multiple-Output). O MIMO, usa diferentes antenas para enviar e receber, aumentando assim o rendimento e realizando mais de uma operação full duplex. A Figura 10 mostra graficamente como é uma trnasmissão utilizando MIMO em comparação com uma transmissão usando SISO.

Figura 10 – Comparação entre uma transmissão SISO e uma Transmissão MIMO

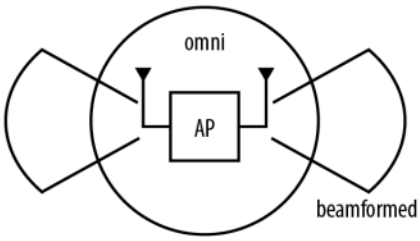


Fonte: (GAST, 2012).

Ainda de acordo com (CARROLL, 2009), o 802.11n e os outros padrões de protocolo 802.11 também são diferentes de outras maneiras. Por exemplo, na camada física, a forma como um sinal é enviado considera reflexões e interferências uma vantagem em vez de um problema. Outra maneira de aumentar o throughput é pela agregação de canais. No 802.11n, dois canais são agregados para aumentar o rendimento. O 802.11n usa canais de 20 MHz e 40 MHz. Os canais de 40 MHz no 802.11n são na verdade dois canais de 20 MHz que são adjacentes entre si e ligados.

Outra tecnologia que veio para aumentar a taxa de transferência nos dispositivos do padrão 802.11n é o *beamforming*. De acordo com (GAST, 2012), ele permite que uma matriz de antenas concentre a energia na direção de um dispositivo cliente. Usando a mesma potência, mas focando-a no receptor, é possível aumentar a relação sinal/ruído no receptor. Maior relação sinal-ruído permite o uso de codificação mais agressiva e, portanto, maior velocidade. A Figura 11 mostra a ideia básica de *beamforming*:

Figura 11 – Ideia de funcionamento do *beamforming*



Fonte: (GAST, 2012).

2.4.6 O protocolo 802.11ac

De acordo com (GAST, 2013), o Grupo de Tarefas AC foi autorizado a construir um padrão de gigabit que pudesse suportado em frequências de menor que 6 GHz, o que o torna compatível com as bandas de frequência existentes utilizadas pelo 802.11. No início do processo de desenvolvimento, foi decidido restringir o 802.11ac às bandas de frequência de 5 GHz usadas pelo 802.11n e pelo 802.11a, e não suportar a banda de frequência de 2,4 GHz usada pelo 802.11b e 802.11g. Na mesma época, o Grupo de Tarefas AD foi autorizado a criar um padrão de gigabit em uma faixa de frequência de 60 GHz. Embora seja uma tecnologia interessante, ela requer mudanças significativas na maneira como as redes são planejadas e construídas, e o alcance é dramaticamente mais curto com uma frequência tão alta.

(GAST, 2013) afirma que conceitualmente, 802.11ac é uma evolução da 802.11n e não uma inovação revolucionária. Muitas das técnicas usadas para aumentar a velocidade no 802.11ac são familiares após a introdução do MIMO. Ao contrário do 802.11n, que desenvolveu novos recursos MAC principais para melhorar a eficiência, o 802.11ac usa técnicas já conhecidas e as leva a um novo nível, com uma exceção. Em vez de usar o MIMO apenas para aumentar o número de fluxos de dados enviados para um único cliente, o 802.11ac é pioneiro em um método de MIMO para vários usuários que permite que um ponto de acesso envie para vários clientes ao mesmo tempo. A Tabela 6 apresenta as diferenças entro o padrão 802.11n e o padrão 802.11ac.

Tabela 6 – Diferenças entre 802.11n e 802.11ac

802.11n	**802.11ac**
Suporta canais de 20 e 40 MHz	Adiciona canais de 80 e 160 MHz
Suporta bandas de frequência de 2,4 GHz e 5 GHz	Suporta apenas 5 GHz

Continua

Tabela 6 – *Continuação*

802.11n	**802.11ac**
Suporta BPSK, QPSK, 16-QAM e 64-QAM	Adiciona 256-QAM
Suporta muitos tipos de *beamforming* explícito	Suporta somente *beamforming* explícito de pacote de dados nulo (NDP) Suporta até quatro fluxos espaciais Suporta até oito fluxos espaciais (AP); dispositivos clientes até quatro fluxos espaciais
Suporta somente transmissão de usuário único	Adiciona transmissão multiusuário
Inclui aprimoramentos significativos de MAC (A-MSDU, A-MPDU)	Suporta aprimoramentos MAC semelhantes, com extensões para acomodar altas taxas de dados

Fonte: Adaptado de (GAST, 2013).

Ainda de acordo com (GAST, 2013), o padrão 802.11ac inclui as seguintes funcionalidades:

- **Canais mais amplos:** 802.11ac apresenta dois novos tamanhos de canal: 80 MHz e 160 MHz. Assim como no 802.11n, canais maiores aumentam a velocidade. Em algumas áreas, 160 MHz de espectro contíguo será difícil de encontrar, então o 802.11ac introduz duas formas de canais de 160 MHz: um único bloco de 160 MHz, e um canal "80 + 80 MHz" que combina dois canais de 80 MHz e entrega a mesma capacidade;

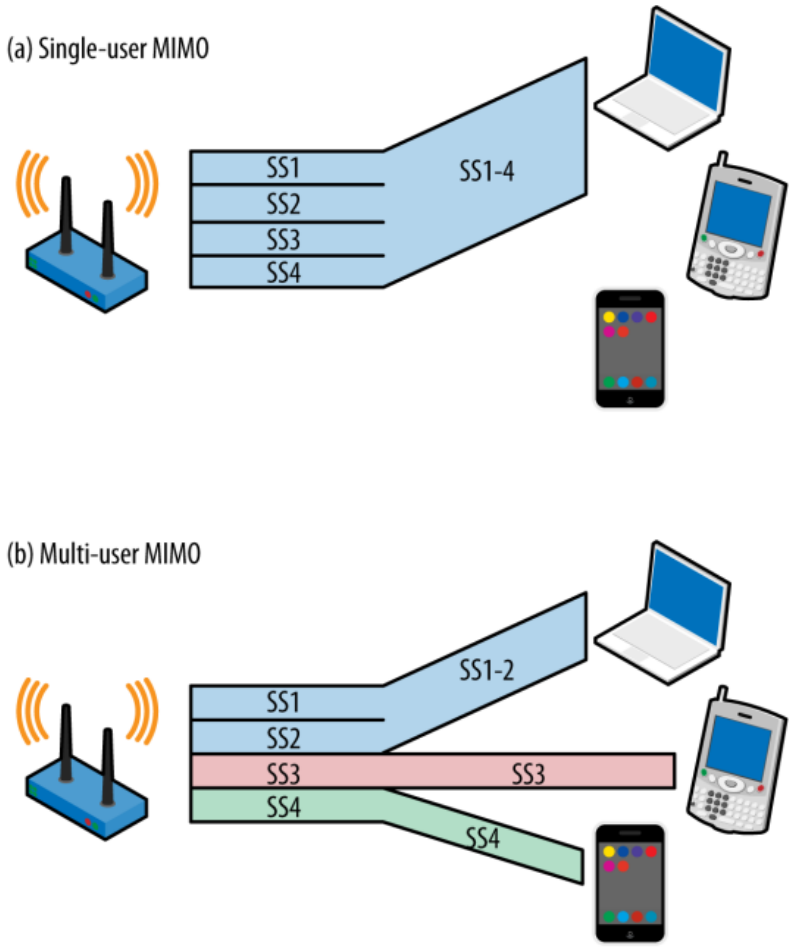
- **256-QAM:** como as alterações anteriores do 802.11, o 802.11ac transmite uma série de símbolos, cada um representando um padrão de bits. Antes do 802.11ac, os dispositivos de LAN sem fio transmitiam seis bits em um período de símbolo. Usando uma modulação mais complexa que suporta mais bits de dados, é possível enviar oito bits por período de símbolo, um ganho de 30%. A extensão até a qual o 256-QAM pode ser usado de maneira confiável em implementações do mundo real é uma questão em aberto para o 802.11ac neste momento;

- **Beamforming:** o 802.11ac simplifica radicalmente as especificações de *beamforming* para um método preferencial. O *beamforming* no 802.11n exigia dois dispositivos para implementar funções de *beamforming* mutuamente aceitáveis a partir das opções disponíveis. Pouquíssimos fornecedores implementaram as mesmas opções e, como resultado, quase não havia compatibilidade de *beamforming* entre dispositivos

de fabricantes diferentes. Com os principais recursos do 802.11ac, dependendo do *beamforming*, entretanto, foi necessária uma simplificação para facilitar essa interação;

- **Múltiplos fluxos espaciais e MIMO multiusuário (MU-MIMO):** o 802.11ac especifica até oito fluxos espaciais, em comparação com os quatro fluxos espaciais do 802.11n, no ponto de acesso. Os fluxos espaciais extra podem ser usados para transmitir para vários clientes ao mesmo tempo. Com a capacidade de transmitir em alta velocidade para vários clientes simultaneamente, o 802.11ac acelerará as redes ainda mais do que seria possível, simplesmente observando a taxa de dados.

Figura 12 – Comparativo entre SISO e MU-MIMO



Fonte: (GAST, 2013).

A Figura 12(a) mostra o funcionamento do SISO, onde os fluxos espaciais são direcionados para um único dispositivo, entregando uma taxa de dados bem alto, o que foi uma inovação protagonizada pelo 802.11n. Em seguida temos a Figura 12(b) que mostra como funciona o MU-MIMO, onde quatro fluxo espaciais são transmitidos para três dispositivos diferentes. Segundo (GAST, 2013), para manter as três transmissões separadas,

o ponto de acesso usa *beamforming* para focar cada uma das transmissões em direção ao seu respectivo receptor. Para que esse tipo de cenário funcione, é necessário que os receptores estejam localizados em direções diferentes o suficiente para que as transmissões focadas não interfiram umas nas outras. Devido ao potencial de interferência inter-stream, as transmissões multi-usuário exigem feedback mais atualizado.

3 OPENWRT

> "O OpenWrt é uma distribuição do GNU/Linux, altamente customizável, direcionada a sistemas embarcados (frequentemente roteadores *wireless*). Ao contrário de muitas outras distribuições para roteadores, o OpenWrt é um sistema operacional facilmente modificável e repleto de recursos." (OPENWRT, 2019).

Sua função é substituir os *firmwares* dos roteadores WiFi vindos de fábrica, dando a eles maior quantidade de funcionalidades e fazendo com o *software* que executa nesses dispositivo deixe de ser simplesmente uma ferramenta pronta e estática, atribuindo mais flexibilidade através da possibilidade de se incluir e excluir novos pacotes de funcionalidades. Isso significa, na prática, que é possível ter tudo o que se precisa, sem desperdícios, com base e um *kernel* Linux que é mais recente do que a maioria das outras distribuições.

Os roteadores de linhas domésticas da grande maioria dos fabricantes oferecem funcionalidades limitadas. Essa limitações são feitas pelo *firmware* padrão de fábrica, que disponibilizam apenas alguma funções ao usuário, sem oferecer o máximo que os *hardwares* desses dispositivos podem oferecer. Como exemplo de funcionalidades que geralmente não são oferecidas de fábrica e que ficam disponíveis com a instalação do OpenWrt temos:

- VLAN (*Virtual Local Area Networks*);
- Múltiplos SSIDs (podendo ser cada SSID em uma VLAN diferente);
- Configuração por Telnet e SSH (*Secure Shell*);
- Servidor e cliente VPN;
- Inclusão e exclusão de novas funcionalidades e funcionalidades já existentes;

O OpenWrt pode ser utilizado tanto como um substituto dos firmwares de alguns pontos de acesso de múltiplos fabricantes conhecidos como sendo o sistema operacional vindo direto de fábrica, em alguns casos bastante comuns em hardwares fabricados na China e disponíveis para importação dos sites de vendas chineses.

3.1 Histórico do OpenWrt

O projeto OpenWrt foi iniciado em janeiro de 2004. As primeiras versões OpenWRT foram baseadas em códigos fontes do Linksys WRT54G para GPL e um BuildRoot[1] do

[1] Ferramenta simples, eficiente e de fácil utilização para a compilação de sistemas embarcados.

projeto uClibc[2]. Esta versão era conhecida como OpenWrt *"stable release"* (versão estável) e foi amplamente colocada em uso (OPENWRT, 2019).

No início de 2005, alguns novos desenvolvedores se associaram à equipe. Depois de alguns meses de desenvolvimento, a equipe decidiu publicar as primeiras versões *"experimental"* (experimentais) de OpenWrt. As versões experimentais usam um sistema de construção fortemente personalizado com base no BuildRoot2 do projeto *uClibc*. OpenWrt usa fontes do *kernel* oficiais GNU/Linux e só adiciona *patches* para o SoC e *drivers* para interfaces de rede (OPENWRT, 2019).

De 2007 até os dias atuais foram lançadas várias versões do OpenWrt, onde as versões estáveis lançadas são:

- **White Russian v0.9 (2007):**Lançada em 2007, essa foi a primeira versão do OpenWrt estável que pôde ser utilizada pelo público. O nome que o batizou veio de um famoso coquetel, onde as versões subsequentes também são batizadas com nomes de coquetéis;
- **Kamikaze v7.09 - v8.09.2 (2006-2010):** Nessa versão foram feitas melhorias substanciais para o ambiente de compilação sobre um *fork* do *BuildRoot-NG* em agosto de 2006, onde estes foram implementados no ramo principal de desenvolvimento do *Kamikaze* e tornou-se o primeiro lançamento oficial da versão estável do *Kamikaze*. As versões 7 e 8 do OpenWrt são da corrente *Kamikaze* e lançados durante o período de 2007 a 2008;
- **Backfire v10.03 - v10.03.1 (2010-2011):** A primeira versão do OpenWrt *Blackfire* 10.03 foi lançado em 2010 e sua versão de manutenção (10.03.1) ficou disponível no fim de 2011;
- **Attitude Adjustment v12.03 (2013):** foi lançada em abril de 2013, onde foram adicionadas importantes atualizações de segurança e, por esse motivo, é recomendado que essa versão seja utilizada em dispositivos com 4MB de memória flash ou mais;
- **Barrier Breaker v14.07 (2014):** versão lançada em julho de 2014;
- **Chaos Calmer v14.07 (2015-2016):** é atual versão do OpenWrt, que usa o *kernel* 3.18 LTS como base. Essa versão foi lançada em setembro de 2015;
- **LEDE 17.01 (2017):** O projeto LEDE foi fundado como um spin-off do projeto OpenWrt e compartilhou muitos dos mesmos objetivos. O projeto visava a criação de uma distribuição Linux embarcada que tornasse mais fácil para desenvolvedores, administradores de sistemas ou outros entusiastas do Linux construir e customizar

[2] Um Sistema Linux de código aberto criado em 1998 e completamente voltado a Sistemas Embarcados, com *kernel* de apenas 900Kb e suporte para vários protocolos de rede.

software para dispositivos embarcados, especialmente roteadores sem fio. O nome LEDE significa Linux Embedded Development Environment;

- **OpenWrt 18.06 (2018):** sendo a versão mais recente disponível, série OpenWrt 18.06 se concentra na modernização de muitas partes do sistema, no backporting do suporte de download para alvos elegíveis e no estabelecimento de bases para atualizações regulares de lançamentos futuros.

3.2 O sistema UCI (Unified Configuration Interface)

Até a versão *White Russian* as configurações eram baseadas em NVRAM (*Non-Volatile* RAM), onde a NVRAM é uma memória não volátil (que persiste mesmo sem alimentação elétrica) responsável por guardar as configurações de inicialização do sistema. O UCI veio como substituto da NVRAM, oferecendo muito mais do um simples método de armazenar os dados de forma persistente, ele veio para oferecer uma forma de configuração fácil e simples, tornando a configuração dos dispositivos menos complicada.

UCI pode ser visto como uma interface de configuração principal do OpenWrt para as configurações mais importantes do sistema. Normalmente, essas são as definições que são cruciais para o funcionamento do dispositivo, tais como são normalmente encontrados na interface web de roteadores e dispositivos embarcados. Exemplos disso, são o de configuração de interfaces de rede, as configurações de rede sem fio, funcionalidade de *log* e de configuração de acesso remoto (OPENWRT, 2019).

Por ser um sistema baseado em Linux, o OpenWrt segue a mesma estrutura de arquivos de configuração, além de aceitar a instalação de programa de terceiros que, por sua vez, tem seus próprios arquivos de configuração espalhados pelos diretórios do sistema, onde cada arquivo de configuração de cada programa pode seguir uma sintaxe particular.

Para facilitar a configuração de alguns programas de terceiros, foi feita a compatibilidade desse programas com o UCI, dessa maneira você não tem que se preocupar com qualquer um dos arquivos de configuração deles, basta modificar os arquivos de configuração da UCI, que seguem uma sintaxe padrão. Claro, a maioria dos *softwares* utilizados não terão sido preparados para a configuração via UCI, o que pode ser algo bom, porque muitas vezes é necessário utilizar o poder da própria interface de configuração de um aplicativo, planejado pelos desenvolvedores. Portanto, apenas alguns programas selecionados pelos mantenedores dos pacotes OpenWRT foram beneficiados com da disponibilidade de uma configuração centralizada feita via UCI. A seguir, temos alguns dos arquivos de configuração com suporte ao UCI:

- **/etc/config/dhcp:** configurações de DNS e DHCP;

- **/etc/config/dropbear:** configurações do servidor SSH;
- **/etc/config/firewall:** configurações *firewall*, NAT, filtragem de pacotes e redirecionamento de portas;
- **/etc/config/network:** configurações de *switch*, interfaces e rotas;
- **/etc/config/wireless:** configurações de rede WiFi;
- **/etc/config/qos:** configurações de QoS (*Quality Of Service*);
- **/etc/config/samba:** configurações do Samba (simulador de gerenciador de arquivos e impressão da Microsoft).

3.3 Sintaxe nos arquivos de configuração

Os arquivos de configuração da UCI seguem uma estrutura simples e de fácil entendimento, onde essa estrutura é composta, basicamente, por um ou mais blocos de instruções de configurações, que por usa vez são formados por uma ou mais declarações de opções e a definição de seus valores.

Figura 13 – Exemplo de estrutura de um arquivo de configuração UCI

```
package 'example'

config 'example' 'test'
        option   'string'      'some value'
        option   'boolean'     '1'
        list     'collection'  'first item'
        list     'collection'  'second item'
```

Fonte: (OPENWRT, 2019).

Na Figura 13, é possivel ver um exemplo de como é estruturado um arquivo de configuração da UCI, exemplificando a inicialização do bloco de configuração indicada por:

- **config 'example' 'teste' :** indica o início de um bloco de configurações (ou seção);
- **option 'string' 'some value' :** definição de um opção do tipo *string*;
- **option 'boolean' '1' :** definição de um opção do tipo *boolean*, que pode receber os valores '0', *'no'*, *'off'*, *'false'* ou *'disabled'* para representar um valor falso e '1' , *'yes'*, *'on'*, *'true'* ou *'enabled'* para representar verdadeiro;

- **list :** define uma lista de valores, que são considerados como sendo da mesma lista de acordo com o tipo (que nesse exemplo é definido pelo valor *'collection'*).

Na definição dos tipos e dos valores pode ser usado aspas simples ou aspas duplas, mas o uso dessas aspas não são obrigatórias, elas são utilizadas apenas para especificar valores que contém espaços em sua composição.

3.4 Configuração por linha de comando

Até agora, foi mostrado que para alterar as configurações era necessário, simplesmente, mudar os valores nos arquivos de configuração do serviço que desejamos modificar. No entanto, quando for necessário fazer configurações via scripts, é possível modificar qualquer configuração UCI através do utilitário de linha de comando UCI, que pode ser utilizado pelos desenvolvedores tanto para definir novas configurações, modificar configurações existentes e analisar as configurações feitas. Essa última função pode ser muito útil quando o desenvolvedor precisa analisar um arquivo de configuração UCI para tomar outra decisão, tornando o uso dos comandos *awk* ou *grep* totalmente desnecessários.

A estrutura base dos comandos UCI é: **"uci [<options>] <command> [<arguments>]"**

Na estrutura base dos comandos UCI temos as opções (*options*), que servem como comandos opcionais para controlar como os resultados serão mostrados ou para especificar como o comando será executado. A Tabela 7 mostra os comandos UCI padrões, sua utilização e a descrição de cada um.

Tabela 7 – Detalhes dos comandos UCI

Comando	**Forma de usar**	**Descrição**
batch	-	Executar *scripts* de comandos UCI.
export	[<config>]	Exportar arquivos de configuração na sintaxe UCI.
import	[<config>]	Importar arquivos de configuração na sintaxe UCI.
changes	[<config>]	Listar as alterações feitas, antes de dar o *commit*.
commit	[<config>]	Escreve os comandos nos arquivos de configuração.

Continua

Tabela 7 – *Continuação*

Comando	Forma de usar	Descrição
add	<config> <section-type>	Adicionar uma seção ao arquivo de configuração.
add_list	<config>.<section>.<option>=<string>	Adicionar uma string à uma lista de já existente.
del_list	<config>.<section>.<option>=<string>	Remover uma *string* de uma lista já existente.
show	[<config>[.<section>[.<option>]]]	Mostrar uma opção, seção ou configuração em notação comprimida.
get	<config>.<section>[.<option>]	Buscar um valor de opção, seção ou configuração.
set	<config>.<section>[.<option>]=<value>	Atribuir um valor à uma opção, seção ou configuração.
delete	<config>[.<section[.<option>]]	Deletar uma determinada seção ou opção.
rename	<config>.<section>[.<option>]=<name>	Renomear uma determinada seção ou opção.
revert	<config>[.<section>[.<option>]]	Reverter uma determinada opção, seção ou arquivo de configuração.

Fonte: Adaptado de (OPENWRT, 2019).

Por ser uma interface de linha de comando, os comandos UCI são utilizados quando é feito o acesso remoto ao dispositivo que executa o OpenWrt. Para permitir o acesso remoto aos dispositivos, o OpenWrt oferece nativamente servidores Telnet e SSH.

3.5 Pacotes OpenWrt

> "O sistema OpenWrt é mantido e distribuído como uma coleção de pacotes. Quase todos os softwares encontrados em uma imagem típica de firmware OpenWrt são fornecidos por um pacote desse tipo, com exceção notável do próprio kernel do Linux."(OPENWRT, 2019).

De acordo com (OPENWRT, 2019), o termo pacote OpenWrt pode referir-se a um pacote fonte OpenWrt que essencialmente é um diretório que consiste em pelo menos um pacote Makefile OpenWrt descrevendo os procedimentos de aquisição, construção

e empacotamento de um software, um diretório suplementar opcional com correções de pacotes OpenWrt que modificam o adquirido código-fonte e, opcionalmente, mais, arquivos suplementares estáticos enviados junto com o pacote, como arquivos de script init, configurações padrão, scripts ou outros arquivos de suporte.

(OPENWRT, 2019) afirma que um pacote OpenWrt também pode se referir a um pacote binário OpenWrt que é um arquivo compatível com GNU tar contendo artefatos executáveis binários juntamente com arquivos de controle de pacotes para instalação em um sistema em execução, semelhante ao Debian .deb ou Linux .rpm distribuições como CentOS, SuSE etc. O pacote binário do OpenWrt é quase exclusivamente produzido a partir de pacotes de código-fonte invocando o pacote OpenWrt ou o OpenWrt SDK para traduzir as descrições do Makefile do pacote fonte para artefatos binários executáveis adaptados para um determinado sistema de destino.

Segundo (OPENWRT, 2019), os pacotes de código-fonte são desenvolvidos em vários feeds de pacotes do OpenWrt hospedados em diferentes locais e para diferentes finalidades. Cada feed de pacote é uma coleção de definições de pacote de origem que residem em um repositório de código-fonte alcançável pública ou privadamente. Pacotes de fontes descrevem como um software deve ser baixado, corrigido, compilado e empacotado para formar um artefato de software binário adequado para uso em um sistema de destino em execução. Eles também descrevem as relações com outros pacotes de origem necessários no tempo de construção ou no tempo de execução. Cada pacote de código-fonte deve ter um nome exclusivo globalmente parecido com o nome do software descrito por ele. O OpenWrt geralmente segue o exemplo de outras distribuições ao decidir sobre a nomenclatura de pacotes e adota as mesmas convenções de nomenclatura em muitos casos.

Existe uma alta disponibilidade de pacotes com múltiplas funcionalidades disponíveis para o OpenWrt. Dentre os quais podemos ver na Tabela 8 alguns poucos exemplos de pacotes e suas funcionalidades, que estão disponíveis para serem baixados ou que já podem vir na própria imagem do OpenWrt baixada ou criada pelo usuário.

Tabela 8 – Detalhes dos comandos UCI

Pacote	**Função**
uhttpd	Servidor HTTP/HTTPS.
iptables	ferramenta de interface de usuário que permite a criação de regras de firewall e NATs.
opkg	Gerenciador de pacotes do OpenWrt bem leve, capaz remover e instalar pacotes .ipk e .deb.
rpcd	Habilita a interface de comunicação por meio de Remote Procedure Calls

Continua

Tabela 8 – *Continuação*

Pacote	**Função**
uci	Serviço que habilita a interface de linha de comando UCI.
squid	O Squid é um proxy de armazenamento em cache para a Web que suporta HTTP, HTTPS, FTP e muito mais. Reduz a largura de banda e melhora os tempos de resposta, armazenando em cache e reutilizando páginas da Web solicitadas com frequência.
mariadb-server	O MariaDB é um servidor de banco de dados SQL multi-usuário, multithreaded rápido, estável e verdadeiro.
tcpdump	Conhecida ferramenta de monitoramento de rede e aquisição de dados.
openssh-server	Servidor SSH.

Fonte: (OPENWRT, 2019).

Para que seja realizada a instalação de um pacote no sistema OpenWrt, é necessário utilizar o utilitário de gerenciamento de pacotes, chamado OPKG. O Através do OPKG é possivel instalar, atualizar e remover pacotes no sistema, além de gerenciar a atualização de informações buscadas no repositório onde esses pacotes são baixados. O exemplo a seguir, mostra como pode ser instalado o pacote uhttpd (servidor de páginas HTTP) no sistema:

```
opkg install uhttpd
```

O OpenWrt oferece vários pacotes de serviços que são muito utilizados, o que pode abrir muito as possibilidades de aplicações em que se torna possível fazer uso dessa distribuição Linux para sistemas embarcados.

4 FRAMEWORK WEB2PY

Desenvolvido inicialmente como uma ferramenta de ensino por Massimo Di Pierro, o Web2py é um *web framework* livre e *open source*, escrito em Python e programável em Python, para desenvolvimento ágil de aplicações *web* seguras baseadas em banco de dados. O Web2py é um *framework "full stack"*, o que significa que ele contém todos os componentes que você precisa para desenvolver completamente aplicações *web* funcionais, altamente interativas e escaláveis.

Em seu nível mais fundamental, uma aplicação *web* é composta de um conjunto de funções (ou métodos) que são executadas quando a URL correspondente é visitada. A saída do programa é retornada para o visitante e transmitido pelo navegador. O objetivo dos *frameworks web* é permitir que desenvolvedores criem novos aplicativos de forma ágil, fácil e sem erros. Isso é feito por fornecimento de APIs[1] (*Application Programming Interface*) e ferramentas que reduzem e simplificam a quantidade de codificação que é necessária (PIERRO, 2019).

Além de permitir a construção de sistemas *web* baseado no padrão MVC (*Model View Controller*), o Web2py permite a implementação de módulos pelo desenvolvedor, que podem ser importados para qualquer parte do sistema de forma muito simples.

O DAL (*Database Abstraction Layer*) é uma API escrita em Python que mapeia objetos Python em objetos de banco de dados, tais como consultas, tabelas e registros, com o objetivo de simplificar a utilização de diversos bancos de dados em aplicações Web2py.

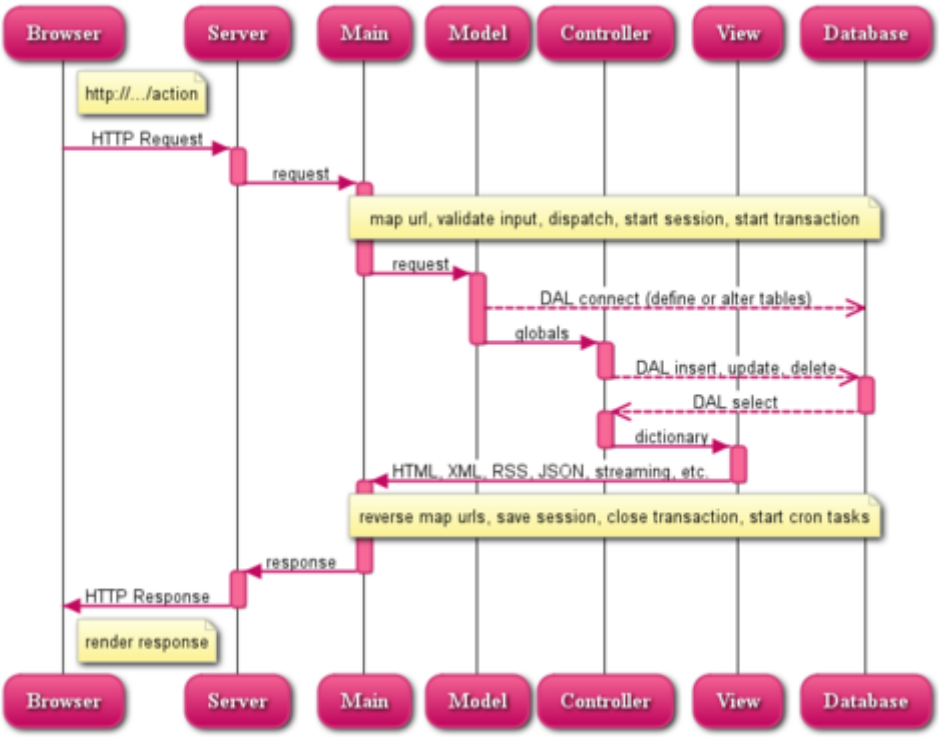
Esse *framework* foi projetado para orientar um desenvolvedor *web* a seguir as boas práticas da engenharia de *software*, tal como a utilização do padrão MVC. Web2py separa a representação dos dados (*model*) da apresentação dos dados (*view*) e também da lógica de aplicação e do fluxo de trabalho (*controller*). Além disso, o Web2py fornece bibliotecas para ajudar na criação, implementação, e teste do projeto, cada uma dessas três partes desenvolvidas separadamente, mas trabalhando em conjunto(PIERRO, 2019).

A linguagem Python tipicamente segue os seguintes princípios:

- Não se repita (DRY - *Don't Repeat Youself*);
- Deve existir apenas uma maneira óbvia de para fazer algo;
- Explícito é melhor do que implícito.

[1] É um conjunto de rotinas e padrões estabelecidos por um *software*, que se resume a utilizar seus serviços, sem implementar nada na própria API.

Figura 14 – Caminho de uma requisição HTTP



Fonte:(PIERRO, 2019).

O Web2py procura seguir rigorosamente os dois primeiros princípios por forçar o desenvolvedor a utilizar boas práticas de engenharia de software, utilizando o amplamente difundido padrão de projeto MVC e incentivando o reaproveitamento de código. O framework guia o desenvolvedor facilitando as tarefas mais comuns em desenvolvimento para web(PYTHON, 2019).

A Figura 14 mostra um exemplo de uma requisição HTTP feita à um sistema web desenvolvido com Web2py, onde o elemento *Main* é a aplicação WSGI[2] (*Web Server Gateway Interface*) e a partir dela o sistema interage entre os *models*, logo após com os *controllers* e, por fim, com as *views*, que são responsáveis por dar a resposta em diversos formatos possíveis (HTML, XML, RSS, JSON, streaming, etc).

A estrutura do Web2py pode ser vista na Figura 15, que nos mostra que ele é composto pelos seguintes componentes:

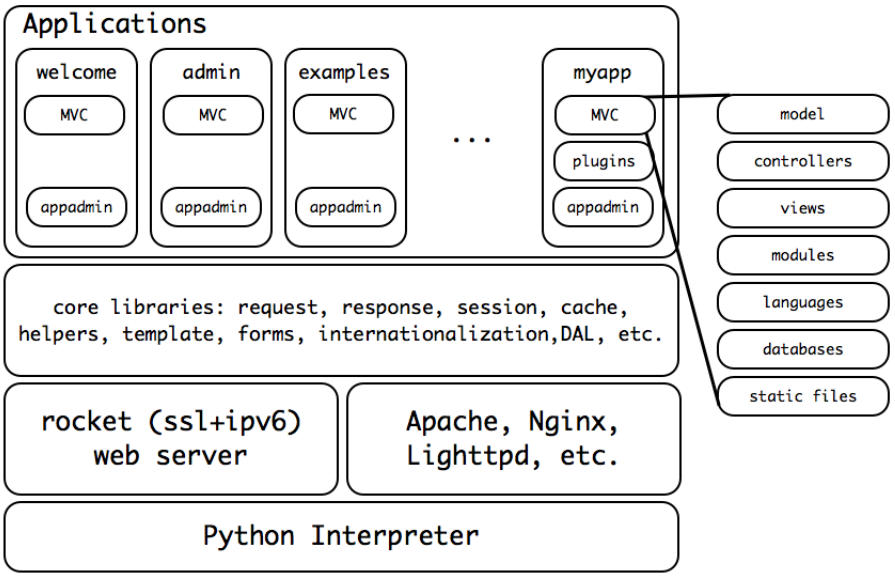
- **Aplicação welcome:** é uma das aplicações padrão do Web2py, que serve como base para o desenvolvimento de novas aplicações;
- **Aplicação examples:** contém exemplos interativos, documentações sobre o Web2py

[2] É uma especificação de uma interface simples e universal usada para manter a comunicação entre servidores web e aplicações ou frameworks que utilizam linguagem Python

e um clone exato do site oficial do Web2py;

- **Aplicação admin:** oferece uma completa IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) que é utilizada para criar, projetar e gerenciar as aplicações Web2py;
- **Aplicação próprias:** são aplicações desenvolvidas pelo usuário;
- **Bibliotecas:** é o núcleo do Web2py, de onde vem todas as funcionalidades do *framework*;
- **Servidor Web:** é o servidor usado para receber e responder às requisições. O servidor padrão é Rocket, mas pode ser substituído pelo Apache através de algumas modificações;
- **Interpretador Python:** responsável por executar os código em Python(atualmente é distribuído com a versão 2.5 do Python).

Figura 15 – Estrutura do Web2py



Fonte:(PIERRO, 2019).

No decorrer do desenvolvimento do Web2py também houve uma grande preocupação em prover segurança das aplicações Web2py. Para isso, o os desenvolvedores resolveram consultar a OWASP (Open Web Application Security Project), que é uma comunidade mundial livre e de código aberto focado em melhorar a segurança de aplicações de software.

A OWASP listou os dez principais problemas de segurança responsáveis por expor aplicações web ao risco de ataque de crackers. Nessa lista há vários tipos de vulnerabilidades

como o SQL *injection*, o *Cross Site Scripting* (XSS), conexões não criptografadas e outras mais. Através dessa lista, os desenvolvedores do Web2py focaram em evitar que as aplicações desenvolvidas com Web2py sofressem dessas mais famosas e perigosas vulnerabilidades.

Com todas as facilidades ofertadas pelo Web2py, aliadas à segurança que o *framework* insere nas aplicações desenvolvidas, faz com que o Web2py seja considerado uma ótima ferramenta para desevolvimento ágil e seguro de aplicações web em linguagem Python.

4.1 Web2py Scheduler

Essa é a solução web2py para executar tarefas em segundo plano (e, portanto, longe do processo do servidor web). O *scheduler* web2py funciona muito bem como a fila de tarefas com as seguintes características:

- Ele fornece um mecanismo padrão para criar, agendar e monitorar tarefas;
- Não há um único processo de segundo plano, mas um conjunto de processos de trabalho;
- O trabalho dos nós do *worker* pode ser monitorado através do seu estado, assim como o estado das tarefas, já que eles são armazenados no banco de dados;
- Funciona sem Web2py, ou seja, funciona em paralelo, em um serviço diferente, mas em conjunto com o Web2py.

No *scheduler*, uma tarefa é simplesmente uma função definida em um modelo (ou em um módulo e importada por um modelo). Por exemplo:

```
def task_add(a, b):
        return a + b
```

De acordo com (PIERRO, 2019), as tarefas sempre serão chamadas no mesmo ambiente visto pelos controladores e, portanto, elas verão todas as variáveis globais definidas nos modelos, incluindo conexões de banco de dados(db). As tarefas diferem de uma ação do controlador porque elas não estão associadas a uma solicitação HTTP. Além disso, as tarefas podem acessar outra variável de ambiente que não estão presente nas solicitações normais.Para ativar o *scheduler*, você deve instanciar a classe *Scheduler* em um modelo. A maneira recomendada de ativar o *scheduler* para seu aplicativo é criar um arquivo de modelo chamado scheduler.py e definir sua função lá. Após as funções, você pode colocar o seguinte código no modelo:

```
from gluon.scheduler import Scheduler
scheduler = Scheduler(db)
```

A tarefa pode ser agendada da seguinte maneira:

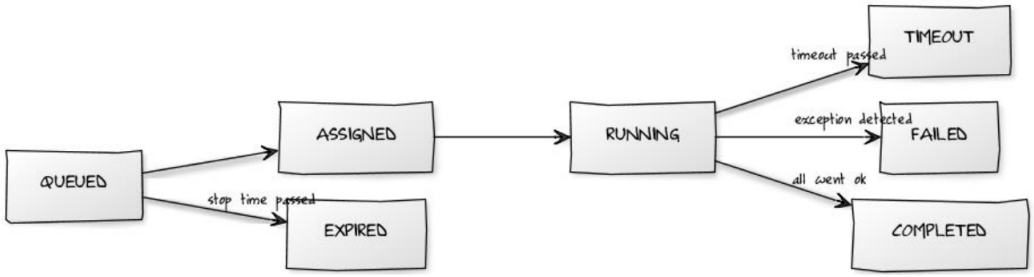
```
scheduler.queue_task(task_add, pvars=dict(a=1, b=2))
```

Para iniciar o scheduler junto com a o servido web que vem como o Web2py, é preciso dao seguinte comando:

```
python web2py.py -a Senha -K meuApp -X
```

Onde, neste caso, o texto "Senha"deve ser substituído pela senha que o usuário deseja utilizar para acessar a área administrativa do Web2py e o texto "meuApp" deve ser substituído pelo nome da aplicação que irá utilizar o *scheduler*.

Figura 16 – Ciclo de vida de uma tarefa



Fonte: (PIERRO, 2019).

Na Figura 16, pode ser visto como funciona o ciclo de vida de uma tarefa no *scheduler* do Web2py. Assim que a tarefa entra na fila, ela pode ficar em dois estados *expired* ou *assigned*, o primeiro quer dizer que a tarefa expirou por ultrapassar o tempo de espera da tarefa e o segundo é quando a tarefa é assumida por um *worker* e passa para o estado *running*. No estado *running* a tarefa é executada e pode assumir três estados: *timeout*(quando o tempo de execução ultrapassa o definido pelo desenvolvedor ou padrão do sistema), *failed*(quando ocorre algum erro não tratado durante a execução da tarefa) ou *completed*(quando a tarefa é realizada com sucesso dentro do tempo permitido).

5 SSH (Secure Shell)

Segundo (MENEZES et al., 2002), o serviço de ssh permite fazer o acesso remoto ao console de uma máquina, como se estivesse conectado localmente ao console da mesma, desenvolvido para substituir ao telnet, rlogin e rsh. A principal diferença com relação ao serviço telnet padrão, rlogin e rsh é que toda a comunicação entre cliente/servidor é feita de forma encriptada usando chaves públicas/privadas RSA para criptografia, garantindo uma transferência segura de dados pela rede.

Também segundo (MENEZES et al., 2002), em conexões sem criptografia (telnet, rsh, rlogin) os dados trafegam de forma desprotegida e caso exista algum programa que faça monitoramento de pacotes instalado em sua rota com a máquina destino, todo o que fizer poderá ser capturado (incluindo senhas).

5.1 Componentes principais do SSH

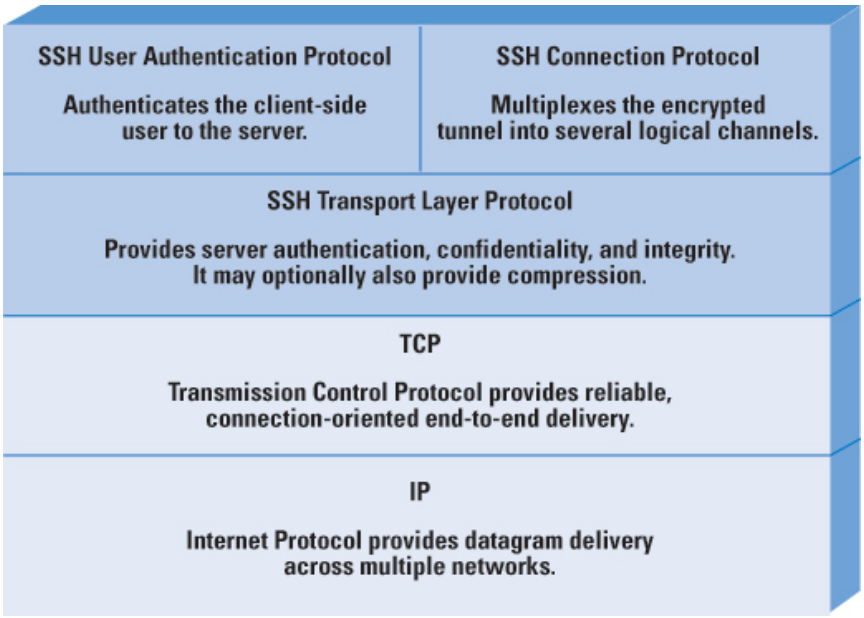
De acordo com (YLONEN; LONVICK, 2006b), o protocolo Secure Shell (SSH) é um protocolo para login remoto seguro e outros serviços de rede segura em uma rede insegura. Como mostra a Figura 17, o protocolo SSH consiste em três componentes principais executando sobre a pilha do protocolo TCP: *O Protocolo de Camada de Transporte* fornece autenticação, confidencialidade e integridade do servidor com perfeito sigilo antecipado. *O Protocolo de Autenticação do Usuário* autentica o cliente para o servidor. *O Protocolo de Conexão* multiplexa o encapsulamento criptografado em vários canais lógicos.

5.1.1 Protocolo de Camada de Transporte SSH

De acordo com (STALLINGS, 2009), a autenticação do servidor ocorre na camada de transporte, com base no servidor que possui um par de chaves pública-privada. Um servidor pode ter várias chaves de host usando vários algoritmos de criptografia assimétrica diferentes. Vários hosts podem compartilhar a mesma chave de host. Em qualquer caso, a chave de host do servidor é usada durante a troca de chaves para autenticar a identidade do host. Para que essa autenticação seja possível, o cliente deve ter conhecimento presuntivo da chave do host público do servidor. A RFC 4251 (YLONEN; LONVICK, 2006b) dita dois modelos de confiança alternativos que podem ser usados:

- O cliente possui um banco de dados local que associa cada nome de host (conforme digitado pelo usuário) com a chave do host público correspondente. Esse método requer infraestrutura administrada centralmente e coordenação de terceiros. A

Figura 17 – Pilha do protocolo SSH



Fonte: (STALLINGS, 2009).

desvantagem é que o banco de dados de associações de nome para chave pode se tornar oneroso para manter;

- A associação de nome-chave do host é certificada por uma autoridade de certificação (CA) confiável. O cliente conhece apenas a chave raiz da CA e pode verificar a validade de todas as chaves do host certificadas pelas CAs aceitas. Essa alternativa facilita o problema de manutenção, porque, idealmente, apenas uma única chave de CA precisa ser armazenada com segurança no cliente. Por outro lado, cada chave de host deve ser adequadamente certificada por uma autoridade central antes que a autorização seja possível.

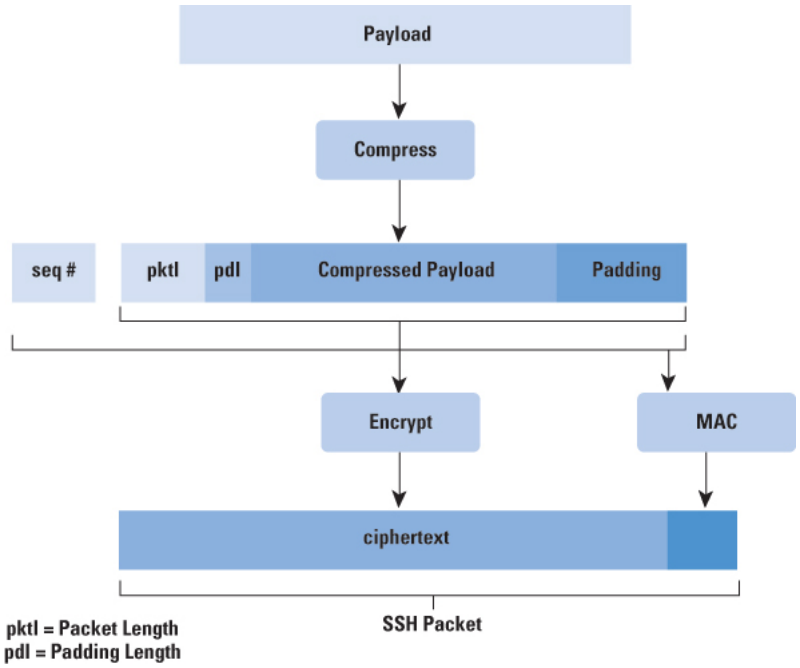
Podemos ver na Figura 18 como é formada a estrutura do pacote da Camada de Transporte SSH, formado pelos seguintes campos:

- **Comprimento do pacote**: O comprimento do pacote é o tamanho do pacote em bytes, não incluindo os campos Comprimento do Pacote e Código de Autenticação de Mensagem;

- **Comprimento de Preenchimento**: o comprimento de preenchimento é o comprimento do campo de preenchimento aleatório;

- **Payload**: Payload constitui o conteúdo útil do pacote. Antes da negociação do algoritmo, este campo é descompactado. Se a compactação for negociada, nos pacotes subsequentes esse campo será compactado;
- **Preenchimento Aleatório**: depois que um algoritmo de criptografia é negociado, esse campo é adicionado. Ele contém bytes aleatórios de preenchimento de forma que o comprimento total do pacote (excluindo o campo MAC) seja um múltiplo do tamanho do bloco de cifra ou 8 bytes para uma cifra de fluxo;
- **Código de Autenticação de Mensagem** (no inglês: Message Authentication Code - MAC): Se a autenticação de mensagem foi negociada, este campo contém o valor MAC. O valor do MAC é calculado sobre o pacote inteiro mais um número de sequência, excluindo o campo MAC. O número de sequência é uma sequência de pacotes implícita de 32 bits que é inicializada em zero para o primeiro pacote e incrementada para cada pacote. O número de sequência não está incluído no pacote enviado pela conexão TCP.

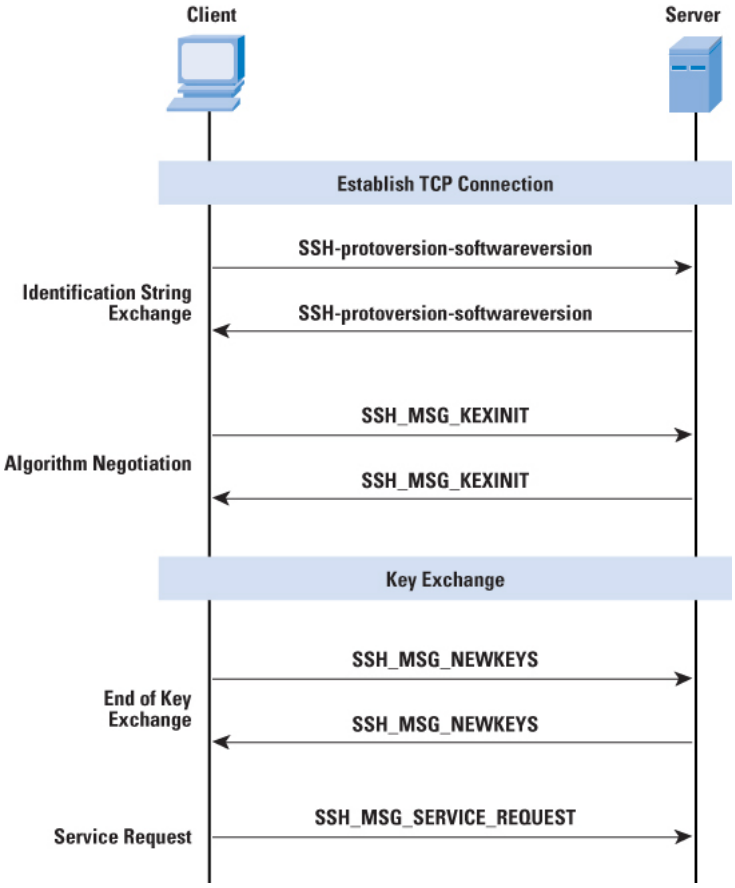
Figura 18 – Estrutura de um pacote SSH



Fonte: (STALLINGS, 2009).

A Figura 19 mostra como funciona a sequência de eventos no protocolo SSH Transport Layer. Inicialmente, o cliente SSH estabelece uma conexão TCP com o servidor SSH por meio do protocolo TCP (que não faz parte do protocolo da camada de transporte). Quando a conexão é estabelecida, o cliente e o servidor trocam dados, chamados de pacotes, no campo de dados de um segmento TCP.

Figura 19 – Sequência eventos da Camada de Transporte do protocolo SSH



Fonte: (STALLINGS, 2009).

De acordo com (STALLINGS, 2009), a troca de pacotes da Camada de Transporte SSH consiste em uma sequência de etapas (Figura 19). O primeiro passo, a troca da string de identificação, começa com o cliente enviando um pacote com uma string de identificação do formulário no seguinte formato:

$$"SSH - versãoProtocolo - versãoSoftwareSPcomentáriosCRLF"$$

Onde SP, CR e LF são caractere de espaço, retorno de carro e alimentação de linha, respectivamente.

Um exemplo de uma *string* válida é:

$$"SSH - 2.0 - billsSSH\_3.6.3q3 < CR >< LF > "$$

O servidor responde com sua própria cadeia de identificação. Essas cadeias de caracteres são usadas na troca de chaves Diffie-Hellman.

Stallings (STALLINGS, 2009) afirma que em seguida vem a negociação de algoritmo. Cada lado envia um mensagem "**SSH_MSG_KEXINIT**"contendo listas de algoritmos suportados na ordem de preferência do remetente. Cada tipo de algoritmo criptográfico possui uma lista. Os algoritmos incluem troca de chaves, criptografia, algoritmo MAC e algoritmo de compressão. A Tabela 9 mostra as opções permitidas para criptografia. Para cada categoria, o algoritmo escolhido é o primeiro algoritmo na lista do cliente que também é suportado pelo servidor.

Tabela 9 – Detalhes dos comandos UCI

Criptografia	**Descrição**
3des-cbc	Padrão de criptografia digital tríplice de três chaves (3DES) no modo CBC (codificação de bloco)
blowfish-cbc	Blowfish no modo CBC
twofish256-cbc	Twofish no modo CBC com uma chave de 256 bits
twofish256-cbc	Twofish no modo CBC com uma chave de 256 bits
twofish192-cbc	Twofish com uma chave de 192 bits
twofish128-cbc	Twofish com uma chave de 128 bits
aes256-cbc	Advanced Encryption Standard (AES) no modo CBC com uma chave de 256 bits
aes192-cbc	AES com uma chave de 192 bits
aes128-cbc	AES com uma chave de 128 bits
Serpent256-cbc	Serpent no modo CBC com uma chave de 256 bits
Serpent192-cbc	Serpent com uma chave de 192 bits
Serpent128-cbc	Serpent com uma chave de 128 bits
arcfour	RC4 com uma chave de 128 bits
cast128-cbc	CAST-128 no modo CBC
hmac-sha1	HMAC-SHA1; Comprimento do digest = comprimento da chave = 20
hmac-sha1-96	Primeiros 96 bits do HMAC-SHA1; Comprimento do digest = 12; Comprimento da chave = 20
hmac-md5	HMAC-SHA1; Comprimento do resumo = Comprimento da chave = 16
hmac-md5-96	Primeiros 96 bits do HMAC-SHA1; Comprimento do digest = 12; Comprimento da chave = 16

Fonte: Adaptado de (STALLINGS, 2009).

Ainda de acordo com (STALLINGS, 2009), o próximo passo é a troca de chaves. A especificação permite métodos alternativos de troca de chaves, mas no momento apenas

duas versões da troca de chaves Diffie-Hellman são especificadas. Ambas as versões são definidas por (HARKINS; CARREL et al.,) e exigem apenas um pacote em cada direção. As etapas a seguir estão envolvidas na troca. Neste, C é o cliente; S é o servidor; p é um prime grande e seguro; g é um gerador para um subgrupo de $GF(p)$; q é a ordem do subgrupo; V_S é a string de identificação de S; V_C é a string de identificação C; K_S é a chave do host público S; I_C é a mensagem C "**SSH_MSG_KEXINIT**"; e I_S é a mensagem S "**SSH_MSG_KEXINIT**"que foi trocada antes desta parte começar. Os valores de p, g e q são conhecidos pelo cliente e pelo servidor como resultado da negociação de seleção de algoritmo. A função hash() também é decidida durante a negociação do algoritmo.

1. C gera um número aleatório $x(1 < x < q)$ e calcula $e = g^x \bmod p$. C envia e para S;

2. S gera um número aleatório $y(0 < y < q)$ e calcula $f = g^y \bmod p$. S recebe e. Ele calcula $K = e^y \bmod p$, $H = \text{hash}\ (V_C \mid\mid V_S \mid\mid I_C \mid\mid I_S \mid\mid K_S \mid\mid e \mid\mid f \mid\mid K)$ e assinatura s em H com sua chave de host privada. S envia $(K_S \mid\mid f \mid\mid s)$ para C. A operação de assinatura pode envolver uma segunda operação de hash;

3. C verifica se K_S realmente é a chave do host para S (por exemplo, usando certificados ou um banco de dados local). C também pode aceitar a chave sem verificação; no entanto, isso tornará o protocolo inseguro contra ataques ativos (mas pode ser desejável por razões práticas no curto prazo em muitos ambientes). C então calcula $K = fx \bmod p$, $H = \text{hash}\ (V_C \mid\mid V_S \mid\mid I_C \mid\mid I_S \mid\mid K_S \mid\mid e \mid\mid f \mid\mid K)$, e verifica a assinatura s em H.

Stallings (STALLINGS, 2009) afirma que como resultado dessas etapas, os dois lados agora compartilham uma chave mestra K. Além disso, o servidor foi autenticado para o cliente, porque o servidor usou sua chave privada para assinar sua metade da troca Diffie-Hellman. Finalmente, o valor de hash H serve como um identificador de sessão para essa conexão. Quando computado, o identificador de sessão não é alterado, mesmo que a troca de chave seja executada novamente para essa conexão para obter chaves novas. O fim da troca de chaves é sinalizado pela troca de pacotes **"SSH_MSG_NEWKEYS"**. Neste ponto, ambos os lados podem começar a usar as chaves geradas a partir de K, como discutido posteriormente.

A etapa final é a solicitação de serviço, onde o cliente fica responsável por enviar um pacote **"SSH_MSG_SERVICE_REQUEST"** para solicitar a autenticação do usuário ou o protocolo de conexão. Após essa solicitação, todos os dados são trocados como a carga útil de um pacote da Camada de Transporte SSH, protegido por criptografia e MAC.

As chaves usadas para criptografia e MAC (e quaisquer IVs necessários) são geradas a partir da chave secreta compartilhada K, o valor de hash da troca de chaves H e o identificador de sessão, que é igual a H, a menos que tenha havido uma troca de chave subsequente após a troca de chave inicial. Considerando que $HASH()$ é a função hash determinada durante a negociação do algoritmo, os valores são calculados da seguinte forma:

- IV inicial do Cliente para o servidor: HASH(K | | H || $"A"$ || session_id);
- IV inicial do Servidor para o cliente: HASH(K || H || $"B"$ || session_id);
- Chave de criptografia cliente para servidor: HASH(K || H || $"C"$ || session_id);
- Servidor de chave de criptografia para cliente: HASH(K || H || $"D"$ || session_id);
- Cliente-chave de integridade para servidor: HASH(K || H || $"E"$ || session_id);
- Servidor de chaves de integridade para cliente: HASH(K || H || $"F"$ || session_id).

5.1.2 Protocolo de Autenticação do Usuário

Segundo (YLONEN; LONVICK, 2006b), a finalidade deste protocolo é executar a autenticação do usuário do cliente. Ele pressupõe que isso seja executado em um protocolo de camada de transporte seguro, que já autenticou a máquina do servidor, estabeleceu um canal de comunicação criptografado e calculou um identificador de sessão exclusivo para essa sessão. Vários métodos de autenticação com diferentes características de segurança são permitidos. Cabe à política local do servidor decidir quais métodos (ou combinações de métodos) está disposto a aceitar para cada usuário.

O servidor pode entrar em um período de suspensão após repetidas tentativas de autenticação malsucedidas para dificultar a busca de chaves pelos invasores. As solicitações de autenticação do cliente possuem o seguinte formato:

Tabela 10 – Requisições de Autenticação

byte	SSH_MSG_USERAUTH_REQUEST(50)
string	nome de usuário
string	nome do serviço
string	nome do método
...	campos específicos do método

Fonte: Adaptado de (STALLINGS, 2009).

Como mostra a Tabela 10, o nome de usuário é a identidade de autorização que o

cliente está reivindicando, nome de serviço é o recurso para o qual o cliente está solicitando acesso (geralmente o Protocolo de Conexão SSH) e o nome do método é o método de autenticação usado nessa solicitação. O primeiro byte tem o valor decimal 50, que é interpretado como SSH_MSG_USERAUTH_REQUEST.

Se o servidor rejeitar a solicitação de autenticação ou aceitar a solicitação, mas exigir um ou mais métodos de autenticação adicionais, o servidor enviará uma mensagem com o formato mostrado na Tabela 11, onde a lista de nomes é uma lista de métodos que podem continuar produtivamente a caixa de diálogo. Se o servidor aceitar autenticação, ele enviará uma mensagem de byte único, SSH_MSG_USERAUTH_SUCCESS(52).

Tabela 11 – Mensagem de rejeição de autenticação

byte	SSH_MSG_USERAUTH_FAILURE (52)
lista de nomes	autenticações que podem continuar
booleano	sucesso parcial

Fonte: Adaptado de (STALLINGS, 2009).

De acordo com (STALLINGS, 2009), a troca de mensagens do processo de autenticação de usuários no SSH envolve as seguintes etapas:

1. O cliente envia uma mensagem SSH_MSG_USERAUTH_REQUEST sem nenhum método solicitado;

2. O servidor verifica se o nome de usuário é válido. Caso contrário, o servidor retornará uma mensagem SSH_MSG_USERAUTH_FAILURE com 'falso' no valor do campo de sucesso parcial. Se o nome de usuário for válido, o servidor prosseguirá para a etapa 3;

3. O servidor retorna a mensagem SSH_MSG_USERAUTH_FAILURE com uma lista de um ou mais métodos de autenticação a serem usados;

4. O cliente seleciona um dos métodos de autenticação aceitáveis e envia uma mensagem SSH_MSG_USERAUTH_REQUEST com o nome desse método e os campos dos métodos específicos necessários. Nesse ponto, pode haver uma sequência de trocas de mensagens para executar o método;

5. Se a autenticação for bem-sucedida e mais métodos de autenticação forem necessários, o servidor prosseguirá para a etapa 3, usando um valor 'verdadeiro' no campo de sucesso parcial. Se a autenticação falhar, o servidor prosseguirá para a etapa 3, usando um valor 'falso' no campo de sucesso parcial;

6. Quando todos os métodos de autenticação necessários forem bem-sucedidos, o servidor enviará uma mensagem SSH_MSG_USERAUTH_SUCCESS e o Protocolo de Autenticação será encerrado.

Stallings (STALLINGS, 2009) também afirma que o servidor pode exigir um ou mais dos seguintes métodos de autenticação:

- **Chave Pública**:os detalhes desse método dependem do algoritmo de chave pública escolhido. Em essência, o cliente envia uma mensagem para o servidor que contém a chave pública do cliente, com a mensagem assinada pela chave privada do cliente. Quando o servidor recebe essa mensagem, ele verifica se a chave fornecida é aceitável para autenticação e, em caso afirmativo, verifica se a assinatura está correta.
- **Password**: o cliente envia uma mensagem contendo uma senha de texto sem formatação, que é protegida por criptografia pelo protocolo da camada de transporte.
- **Baseda em host**: a autenticação é executada no host do cliente, em vez do próprio cliente. Assim, um host que suporta vários clientes forneceria autenticação para todos os seus clientes. Esse método funciona fazendo com que o cliente envie uma assinatura criada com a chave privada do host do cliente. Assim, em vez de verificar diretamente a identidade do usuário, o servidor SSH verifica a identidade do host do cliente e, em seguida, acredita no host quando diz que o usuário já foi autenticado no lado do cliente.

5.1.3 Protocolo de Conexão

Segundo (STALLINGS, 2009), o protocolo de conexão SSH é executado na parte superior do protocolo SSH Transport Layer e presume que uma conexão de autenticação segura esteja em uso. Essa conexão de autenticação segura, chamada de túnel, é usada pelo Protocolo de Conexão para multiplexar vários canais lógicos.

Em (YLONEN; LONVICK, 2006a), "O protocolo de conexão Secure Shell (SSH)", afirma que o protocolo de conexão é executado em cima dos protocolo de camada de transporte e o protocolo de autenticação de usuário. Já em (YLONEN; LONVICK, 2006b), "Arquitetura do Protocolo SSH", declara que o Protocolo de Conexão é executado sobre o Protocolo de Autenticação do Usuário. (STALLINGS, 2009) afirma que na verdade, o Protocolo de Conexão é executado sobre o Protocolo da Camada de Transporte, mas pressupõe que o Protocolo de Autenticação do Usuário tenha sido invocado anteriormente.

(STALLINGS, 2009) também afirma que todos os tipos de comunicação usando SSH, como uma sessão de terminal, são suportados usando canais separados. Qualquer um dos lados pode abrir um canal. Para cada canal, cada lado associa um número de canal

único, que não precisa ser o mesmo em ambas as extremidades. Os canais são controlados por fluxo usando um mecanismo de janela. Nenhum dado pode ser enviado a um canal até que uma mensagem seja recebida para indicar que o espaço da janela está disponível. A vida de um canal progride através de três etapas: abertura de um canal, transferência de dados e fechamento de um canal.

De acordo com (STALLINGS, 2009), quando um dos lados deseja abrir um novo canal, ele aloca um número local para o canal e envia uma mensagem do formulário com o formato semelhante ao mostrado na Tabela 12, onde uint32 significa inteiro de 32 bits sem sinal. O tipo de canal identifica o aplicativo para este canal, conforme descrito posteriormente. O canal emissor é o número do canal local. O tamanho inicial da janela especifica quantos bytes de dados de canal podem ser enviados ao remetente desta mensagem sem ajustar a janela. O tamanho máximo do pacote especifica o tamanho máximo de um pacote de dados individual que pode ser enviado ao remetente. Por exemplo, pode-se querer usar pacotes menores para conexões interativas para obter melhor resposta interativa em links lentos.

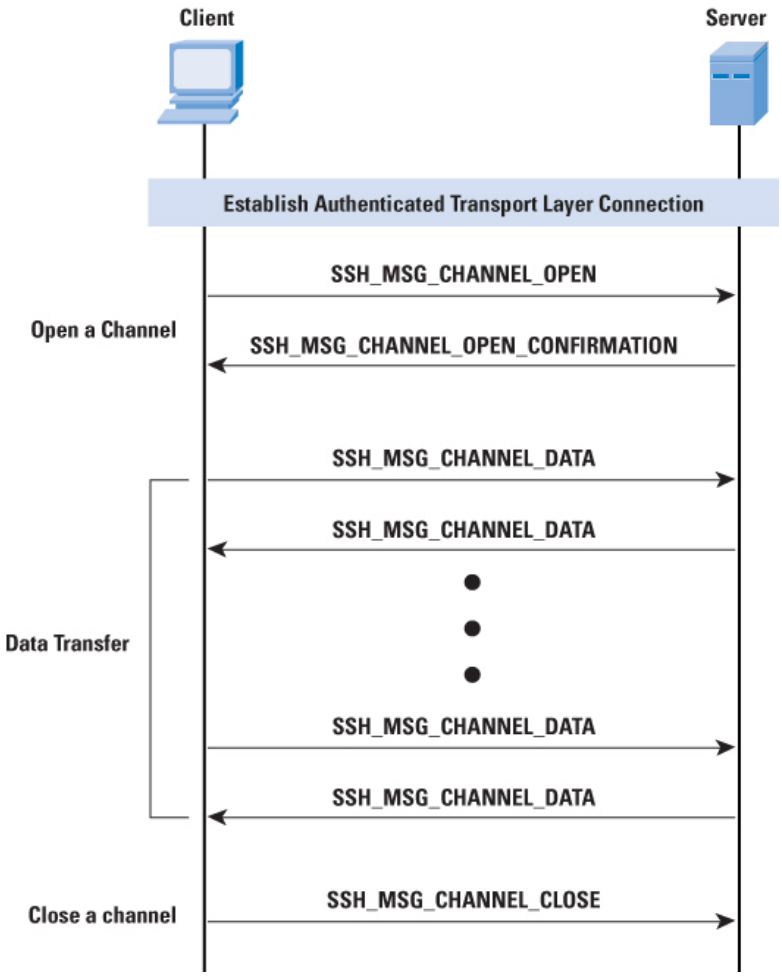
Tabela 12 – Mensagem de alocação de novo canal SSH

byte	SSH_MSG_CHANNEL_OPEN
string	tipo de canal
uint32	canal remetente
uint32	tamanho inicial da janela
uint32	tamanho máximo do pacote
....	seguir dados específicos do tipo de canal

Fonte: Adaptado de (STALLINGS, 2009).

Se o lado remoto puder abrir o canal, de acordo com (STALLINGS, 2009), ele retornará uma mensagem SSH_MSG_CHANNEL_OPEN_CONFIRMATION, que inclui o número do canal do remetente, o número do canal do destinatário e os valores de tamanho da janela e do pacote para o tráfego de entrada. Caso contrário, o lado remoto retornará uma mensagem SSH_MSG_CHANNEL_OPEN_FAILURE com um código indicando o motivo da falha. Depois que um canal é aberto, a transferência de dados é realizada usando uma mensagem SSH_MSG_CHANNEL_DATA, que inclui o número do canal do destinatário e um bloco de dados. Essas mensagens, em ambas as direções, podem continuar enquanto o canal estiver aberto. Quando qualquer um dos lados deseja fechar um canal, envia uma mensagem SSH_MSG_CHANNEL_CLOSE, que inclui o número do canal do destinatário. A Figura 20 fornece um exemplo de troca de mensagens no Protocolo de Conexão.

Figura 20 – Troca de mensagens após o estabelecimento da conexão SSH



Fonte: (STALLINGS, 2009).

Quatro são as possibilidades de tipos de canais conhecido e especificados por (YLONEN; LONVICK, 2006a):

- **Sessão**: refere-se à execução remota de um programa. O programa pode ser um shell, um aplicativo como transferência de arquivos ou e-mail, um comando do sistema ou algum subsistema interno. Quando um canal de sessão é aberto, as solicitações subsequentes são usadas para iniciar o programa remoto;
- **x11**: Este tipo de canal refere-se ao X Window System, um sistema de software de computador e protocolo de rede que fornece uma GUI para computadores em rede.

X permite que os aplicativos sejam executados em um servidor de rede, mas sejam exibidos em uma máquina de desktop;

- **Encaminhamento de porta TCP/IP**: fornece a capacidade de converter qualquer conexão TCP insegura em uma conexão SSH segura. Também é conhecido como tunelamento SSH. Uma porta é um identificador de um usuário do TCP. Portanto, qualquer aplicativo executado em cima do TCP possui um número de porta. O tráfego TCP de entrada é entregue ao aplicativo apropriado com base no número da porta. Um aplicativo pode empregar vários números de porta. Por exemplo, para o protocolo SMTP, o lado do servidor geralmente escuta na porta 25, para que uma solicitação SMTP de entrada use TCP e direcione os dados para a porta de destino 25. O TCP reconhece que esse endereço é o endereço do servidor SMTP e encaminha os dados para o aplicativo do servidor SMTP;

- **TCP/IP local**: Quando uma conexão chega a uma porta TCP/IP encaminhada localmente, o seguinte pacote é enviado para o outro lado. Observe que essas mensagens também podem ser enviadas para portas para as quais nenhum encaminhamento foi explicitamente solicitado. O lado receptor deve decidir se permite o encaminhamento.

5.1.4 Protocolo SFTP (Secure File Trasfer Protocol)

De acordo com (DEAN, 2012), o SFTP é uma versão segura do FTP, que usa o SSH para criptografia e às vezes é chamado de FTP sobre SSH ou SSH FTP. O SFTP pode ser configurado para escutar em qualquer porta, embora normalmente use a porta 22 do SSH.

Segundo (STAHNKE, 2005), sftp pode ser usado de maneira muito semelhante ao FTP, embora por baixo dos panos o FTP e o sftp sejam muito diferentes. O sftp é um mecanismo de transferência de arquivos totalmente diferente, originalmente desenvolvido no SSH Communications Security para o protocolo SSH versão 2. O cliente sftp fala com o subsistema sftp-server dentro da conexão SSH. Isso significa que os comandos enviados via ssh são passados para o subsistema sftp-server.

(STAHNKE, 2005) também afirma que os usuários finais aproveitam a simplicidade e o conforto do sftp. Eles podem usar muitos comandos dentro de uma sessão sftp que eles estão acostumados a partir de uma sessão tradicional do cliente ftp. Uma vez que um usuário familiarizado com o FTP é apresentado ao sftp, normalmente a transição é bem simples. O sftp, no entanto, transfere arquivos conforme encontrado no sistema de arquivos. Normalmente, no FTP, o modo ASCII converterá os arquivos DOS em UNIX e vice-versa; Ao usar o sftp, o arquivo é transferido exatamente sem qualquer conversão.

6 RPC (Remote Procedure Call)

Segundo (BIRRELL; NELSON, 1984), Remote Procedure Calls (RPC) é um paradigma útil para fornecer comunicação através de uma rede entre programas escritos em uma linguagem de alto nível. A idéia de chamadas do RPC é bastante simples, baseia-se na observação de que as chamadas de procedimento são um mecanismo bem conhecido e bem compreendido para a transferência de controle e dados dentro de um programa executado em um único computador. Portanto, propõe-se que esse mesmo mecanismo seja estendido para fornecer transferência de controle e dados através de uma rede de comunicação.

Ainda de acordo com (BIRRELL; NELSON, 1984), quando um RPC é invocado, o ambiente de chamada é suspenso, os parâmetros são passados pela rede para o ambiente onde o procedimento deve ser executado (ao qual nos referiremos como o chamado) e o procedimento desejado é executado lá. Quando o procedimento termina e produz seus resultados, os resultados são passados para o ambiente de chamada, onde a execução continua como se estivesse retornando de uma simples chamada de máquina única. Enquanto o ambiente de chamada está suspenso, outros processos nessa máquina podem (possivelmente) ainda ser executados (dependendo dos detalhes do paralelismo desse ambiente e da implementação de RPC).

6.1 JSON-RPC

De acordo com (JSONRPC, 2019), o JSON-RPC é um protocolo de chamada de procedimento remoto (RPC) leve e sem estado. Primeiramente, esta especificação define várias estruturas de dados e as regras em torno de seu processamento. Ele é independente de transporte, pois os conceitos podem ser usados dentro do mesmo processo, em sockets, em http ou em vários ambientes de passagem de mensagens. Ele usa JSON como formato de dados, definido em (CROCKFORD, 2006).

Como o JSON-RPC utiliza o JSON, ele possui o mesmo sistema de tipos, de acordo com (CROCKFORD, 2006). O JSON pode representar quatro tipos primitivos (*Strings*, *Numbers*, *Booleans* e *Null*) e dois tipos estruturados (Objetos e *Arrays*). O termo "primitivo"nesta especificação referencia qualquer um desses quatro tipos primitivos de JSON. O termo "Estruturado"faz referência aos tipos JSON estruturados.

Uma chamada RPC é representada enviando um objeto *Request* para um servidor. De acordo com (JSONRPC, 2019), o objeto *Request* possui os seguintes membros:

- **jsonrpc:** Uma *String* especificando a versão do protocolo JSON-RPC. Deve ser exatamente "2.0";
- **method:** Uma *String* contendo o nome do método a ser invocado. Os nomes dos métodos que começam com a palavra rpc seguida por um caractere de período (U + 002E ou ASCII 46) são reservados para métodos e extensões internos da RPC e não devem ser usados para mais nada;
- **params:** Um valor estruturado que contém os valores de parâmetro a serem usados durante a invocação do método. Este membro pode ser omitido;
- **id:** Um identificador estabelecido pelo Cliente que deve conter um valor String, Number ou NULL, se incluído. Se não estiver incluído, é considerado uma notificação. O valor normalmente não deve ser nulo e os números não devem conter partes fracionárias.

Quando uma chamada RPC é feita, o servidor deve responder com uma resposta, exceto no caso de notificações. De acordo com (JSONRPC, 2019), a resposta é expressa como um único Objeto JSON, com os seguintes membros:

- **jsonrpc:** Uma *String* especificando a versão do protocolo JSON-RPC. Deve ser exatamente "2.0";
- **result:** Este membro é necessário em caso de sucesso. Este membro não deve existir se houver um erro ao invocar o método. O valor desse membro é determinado pelo método chamado no servidor;
- **error:** Este membro é necessário por erro. Esta membro não deve existir se não houve erro acionado durante a invocação. O valor para este membro deve ser um objeto;
- **id:** Este membro é necessário. Deve ser o mesmo que o valor do membro id no objeto de solicitação. Se ocorreu um erro ao detectar o ID no objeto Solicitação (por exemplo, Erro de análise/Solicitação inválida), ele deve ser Nulo.

De acordo com (JSONRPC, 2019), quando uma chamada RPC encontra um erro, o objeto de resposta deve conter o membro de erro com um valor que é um objeto com os seguintes membros:

- **code:** Um número que indica o tipo de erro que ocorreu. Deve ser do tipo Inteiro;
- **message:** Uma *String* fornecendo uma breve descrição do erro. A mensagem deve ser limitada a uma única sentença concisa;

- **data:** Um valor primitivo ou estruturado que contém informações adicionais sobre o erro, podendo ser omitido. O valor desse membro é definido pelo servidor (por exemplo, informações detalhadas de erro, erros aninhados etc.).

A Tabela 13 mostra os erros pré-definidos pelo JSON-RPC. De acordo com (JSONRPC, 2019), os códigos de erro de -32768 a -32000 são reservados para erros pré-definidos. Qualquer código dentro deste intervalo, mas não definido explicitamente abaixo, é reservado para uso futuro.

Tabela 13 – Erros do JSON-RPC

código	**mensagem**	**significado**
-32700	Erro de análise	JSON inválido foi recebido pelo servidor. Ocorreu um erro no servidor durante a análise do texto JSON.
-32600	Pedido inválido	O JSON enviado não é um objeto de solicitação válido.
-32601	Método não encontrado	O método não existe/não está disponível.
-32602	Params inválidos	Parâmetros de método inválidos.
-32603	Erro interno	Erro JSON-RPC interno.
-32000 a -32099	Erro do servidor	Reservado para erros de servidor definidos pela implementação.

Fonte: Adaptado de (JSONRPC, 2019).

6.1.1 JSON-RPC sobre HTTP

Segundo (JSONRPC, 2019), com alguma limitação, as solicitações HTTP podem ser usadas como um transporte para comunicação entre pares. Uma comunicação entre pares, sendo um cliente HTTP e outro um servidor HTTP, pode abranger várias solicitações HTTP. Um par do lado do cliente pode enviar uma ou mais solicitações, notificações ou respostas ao seu par enviando uma solicitação HTTP POST contendo todos os objetos serializados.

(JSONRPC, 2019) também afirma que o par do lado do servidor deve responder com respostas a todas as solicitações enviadas e pode enviar solicitações ou notificações próprias. O par do lado do cliente deve responder às solicitações recebidas enviando outro HTTP POST. Para dar ao servidor uma oportunidade de enviar mensagens para o peer do lado do cliente, o peer do lado do cliente pode reabrir a comunicação enviando um HTTP POST vazio. Uma solicitação não válida deve resultar no fechamento da conexão.

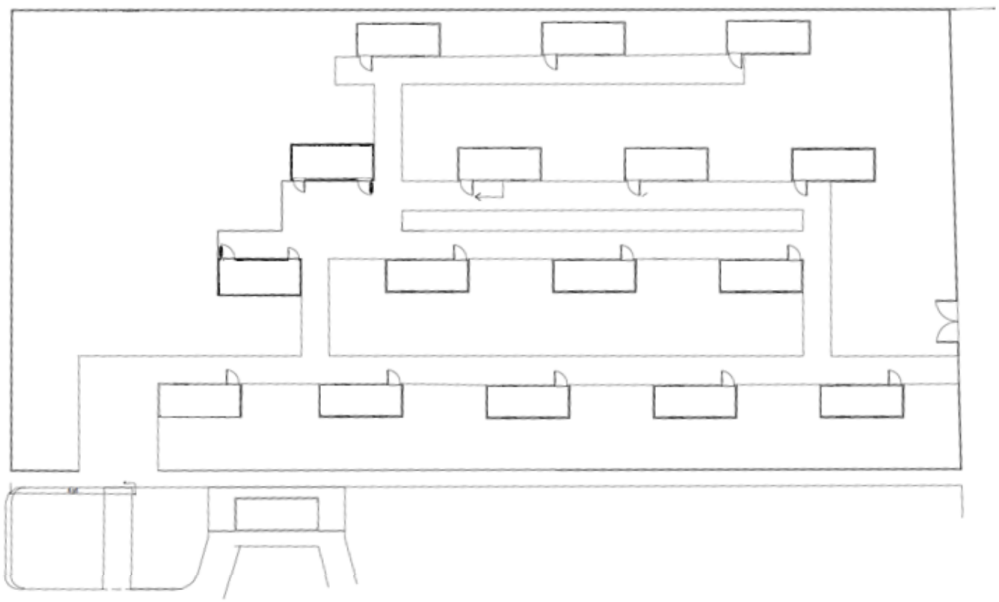
Uma resposta inválida deve gerar uma exceção para todas as solicitações não atendidas no cliente. Fechar uma conexão deve gerar uma exceção para todas as solicitações não atendidas no cliente.

7 Vila de Contêineres da UEMA

A Vila de Contêineres da Universidade Estadual do Maranhão tem o objetivo de servir de instalações de instalações para laboratórios de empresas juniores de alguns cursos da universidade, oferecendo um ambiente climatizado e com isolamento térmico para utilização da comunidade acadêmica.

O projeto consiste na instalação de 16 contêineres, como mostra a Figura 21, sendo que 2 desses contêineres são utilizados como banheiros, 1 é utilizado como suporte para operações de segurança e abrigo para a infraestrutura de rede de dados da própria Vila de Contêineres e os outros 13 são utilizados para fins acadêmicos.

Figura 21 – Projeto da Vila de Contêineres da UEMA



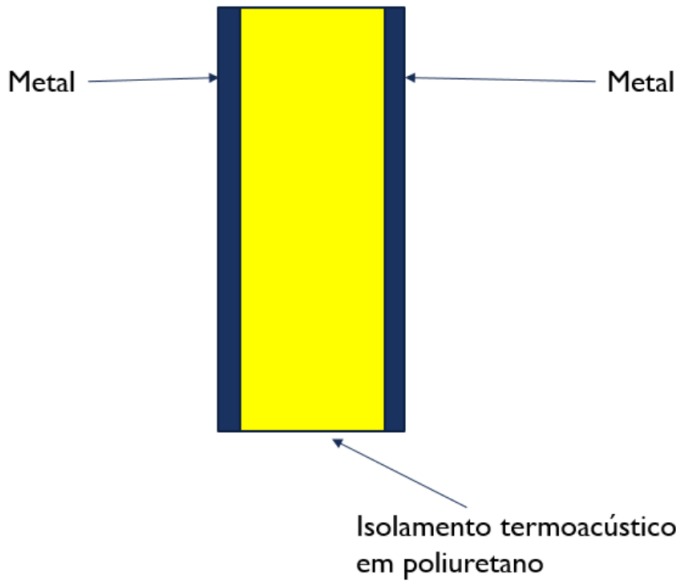
Fonte: O autor.

Segundo (UEMA, 2019), a vila científica contemplará os laboratórios de biologia pesqueira - BIOPESQ, Fisioecologia, Reprodução e Cultivo de Organismos Marinhos - FISIOMAR, Genética e Reprodução de Peixes, Preservação de Gametas de Peixes, Engenharia Mecânica, Engenharia de Produção, Coleção de Tecidos e DNA da Fauna Maranhense; AGA; Empresas Juniors dos cursos de Administração e Engenharia da Computação).

Segundo (MODULARES, 2019), os contêineres têm as dimensões de 6 metros de comprimento e 2,65 metros de largura, onde a sua estrutura é feita de aço galvanizado de 2 milimetros e as paredes são compostas de painéis termoacústicos feitos em placas de de

metal na parte externa e miolo composto de uma espuma de Poliuretano, como está sendo representado na Figura 22.

Figura 22 – Estrutura do painel termoacústico



Fonte: O autor.

7.1 Problema da Cobertura Wi-Fi

Em alguns testes, observou-se que a perda por penetração causada pelos painéis termoacústicos das paredes variavam de 15dB a 17dB, prejudicando bastante a propagação do sinal da rede Wifi, fazendo com que se fizesse necessária a utilização de um número considerável de pontos de acesso para oferecer uma qualidade aceitável (próximo a -67dBm) em todos os contêineres, de acordo com o padrão utilizado no projeto de implantação da rede Wi-Fi da UEMA.

Pudemos afirmar isso utilizando o software de predição de sinal Ekahau Site Survey para simular o ambiente da Vila de Contêineres. Como esse software é possível fazer uma predição da propagação do sinal, simulando o ambiente dos contêineres, levando em consideração os materiais no qual os contêineres são compostos, a escala da planta utilizada como base para a simulação e os pontos de acesso da solução adotada como padrão da rede institucional UEMA.

Foram feitas três simulações no Ekahau Site Survey na frequência de 2.4GHz e potência de transmissão de 100mW, nas quais foram utilizadas as seguintes abordagens:

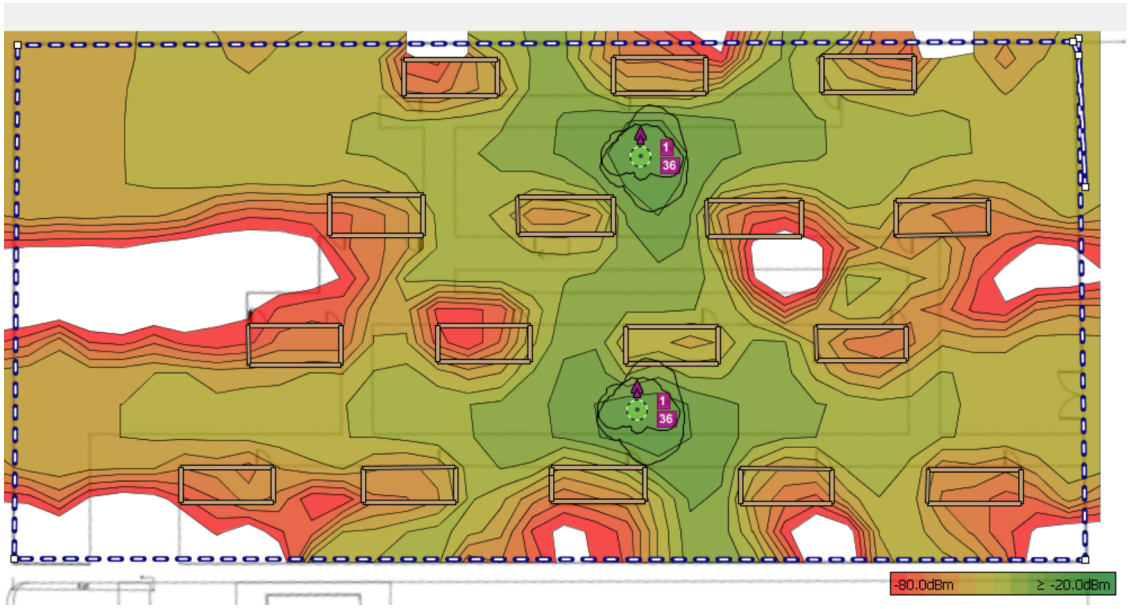
- **Abordagem outdoor:** foram utilizados dois pontos de acesso HPE Aruba IAP-275 (específico para ambientes externos) posicionados estrategicamente entre os

contêineres;

- **Abordagem indoor alternado:** foram utilizados seis pontos de acesso HPE Aruba IAP-205 (específico para ambientes internos) posicionados entro de contêineres alternados, de forma que se tentou obter o melhor sinal possível nos contêineres que não foram colocados pontos de acesso;
- **Abordagem indoor coompleto:** foram utilizados quatorze pontos de acesso HPE Aruba IAP-205 (específico para ambientes internos) posicionados entro de cada contêiner (com exceção dos banheiros), com o objetivo de deixar todos os contêineres com sinal de qualidade.

Os dois modelos usados como referência para a realização da simulação (IAP-275 e IAP-205) são os pontos de acesso mais básicos adquiridos pela UEMA na mais recente licitação de compra de pontos de acesso Wi-Fi e que estavam disponíveis para serem instalados.

Figura 23 – Simulação da propagação de sinal, utilizando o software Ekahau Site Survey em frequência de 2.4GHz, utilizando abordagem outdoor.

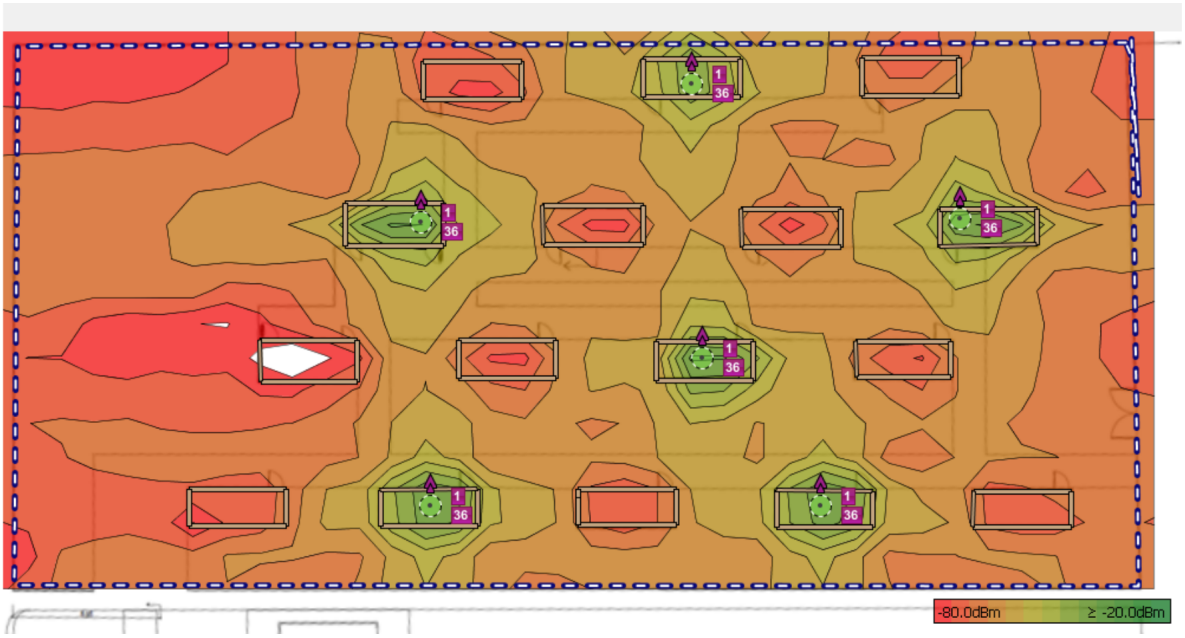


Fonte: O autor.

Na Figura 23 observamos que (de acordo com a legenda de intensidade de sinal) alguns contêineres ficaram com uma qualidade de sinal baixa (acima de -70dBm), o que poderia prejudicar a experiência de conexão do usuário em alguns contêineres.

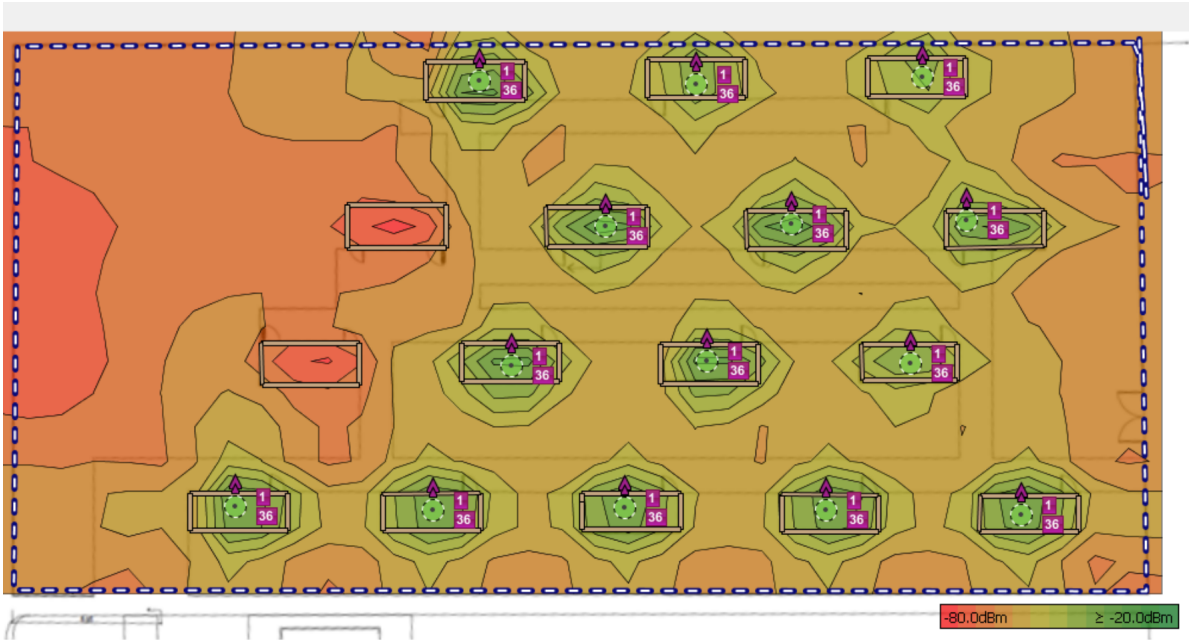
A Figura 24 nos mostra situação em que foi pensado a possibilidade de posicionamento dos pontos de acessos dentro dos contêineres, em distâncias médias, alternando espaçadamente os contêineres onde seriam instalados os pontos de acesso.

Figura 24 – Simulação da propagação de sinal, utilizando o software Ekahau Site Survey em frequência de 2.4GHz, utilizando abordagem indoor alternado.



Fonte: O autor.

Figura 25 – Simulação da propagação de sinal, utilizando o software Ekahau Site Survey em frequência de 2.4GHz, utilizando abordagem indoor completo.



Fonte: O autor.

De acordo com a legenda de intensidade de sinal da Figura 24, alguns contêineres ficaram com uma qualidade de sinal baixa (acima de -70dBm, chegando até -80dBm), o

que poderia prejudicar a experiência de conexão do usuário em alguns contêineres. Isso ocorre pela necessidade de o sinal precisar ultrapassar duas paredes para chegar ao interior dos contêineres onde não há pontos de acesso instalados.

Podemos ver na Figura 25 a terceira abordagem de simulação realizada, levando em consideração a instalação de um ponto de acesso em cada contêiner (com exceção dos banheiros). Nessa abordagem, o sinal dentro de todos os contêineres atingem a qualidade mínima exigida, como já era de se esperar, uma vez que não há barreiras para atenuar o sinal de forma significativa.

7.1.1 Custos de implantação

Outro fator importante a se considerar é o valor que cada ponto de acesso irá custar aos cofres públicos e o custo total de implantação, incluindo apenas os pontos de acesso e as licenças necessárias para fazer com que o ponto de acesso funcione com a infraestrutura que a universidade já dispõe.

Tabela 14 – Preços de pontos de acessos e licenças utilizados como base para as simulações.

Modelo do ponto de acesso	**Preço**
IAP-275-RW : ponto de acesso para ambiente externo	$1995.00
IAP-205-RW : ponto de acesso para ambiente interno	$695.00
LIC-AP: licença obrigatória para vincular o ponto de acesso na controladora Wi-Fi da UEMA	$75.00

Fonte: (ITPRICE, 2019).

Na Tabela 14 podemos ver um resumo desses valores, detalhando cada item e custo unitário em dolar. Esses valores (em dólar) são baseados nos dados encontrados no site (ITPRICE, 2019), responsável por guardar as price lists divulgadas pelos fabricantes, tomando como referência o ano de 2019.

Podemos ver na Tabela 15 as informações referentes aos preços obtidos do (ITPRICE, 2019) e a quantidade de pontos de acesso utilizados em cada abordagem que realizamos a simulação.

Tabela 15 – Custo estimado das abordagens simuladas.

Abordagem	**Preço por licença**	**Preço por ponto de acesso**	**Quantidade**	**Preço total**
Outdoor	$75,00	$1.995,00	2	$4.140,00
Indoor alternado	$75,00	$695,00	6	$4.620,00
Indoor completo	$75,00	$695,00	14	$10.780,00

Fonte: (ITPRICE, 2019).

Com essas informações, podemos calcular o custo de implantação aproximado das três abordagens, onde pudemos observar que a abordagem com menor custo foi a abordagem outdoor com o custo de $4.140,00 dólares, enquanto que a mais cara (e com a melhor qualidade de sinal entregue ao usuário) foi a abordagem indoor completa com $10.780,00 dólares.

8 Tiúba Wi-Fi Manager

O Tiúba Wi-Fi Manager - TWM é um sistema web responsável por gerenciar pontos de acesso Wi-Fi que tem o OpenWrt como sistema operacional. Ele foi desenvolvido como uma proposta para gerenciar pontos de acesso com um custo reduzido ou até mesmo já disponíveis nos local(por meio da substituição do firmware original).

Através do TWM é possível gerenciar algumas funcionalidades de rede, de wireless, gerenciamento de pontos de acesso por grupos, múltiplos SSIDs, múltiplas VLANs, vincular SSID à uma VLAN específica e verificar os usuários conectados aos pontos de acesso.

Atualmente, o TWM gerencia pontos de acesso com o OpenWrt versão 18.06.4 instalado, mas não basta que o sistema esteja instalado, é necessário que alguns pacotes e alguns arquivos de configuração sejam modificados no ponto de acesso para que tenhamos como fazer a gerencia do dispositivo pelo TWM. Para entendermos melhor, veremos a seguir como é feita a configuração do ponto de acesso para que seja possível o pleno funcionamento com o Tiúba.

8.1 Preparando o ponto de acesso Wi-Fi

Para este trabalho, estamos considerando que o ponto de acesso já está com o Openwrt instalado e funcionando com as configurações padrões. Caso o hardware utilizado seja de um fabricante que usa firmware próprio, será necessário a substituição do firmware, caso o ponto de acesso já venha com o OpenWrt instalado, é interessante restaurar as configurações do ponto de acesso para o padrão.

Com o OpenWrt pronto para ser configurado, a primeira coisa a ser feita é a configuração de um endereço IP (evitando conflitos) e uma senha padrão para o usuário root, de escolha do administrador da rede. Após isso, para o funcionamento da integração entre o ponto de acesso e o sistema Tiúba Wi-Fi Manager é necessário que dois pacotes sejam instalados no OpenWrt:

- **uhttpd-mod-ubus:** habilita a interface ubus do OpenWrt, por onde nós iremos realizar o acesso às informações do ponto de acesso por meio do protocolo HTTP;
- **rpcd-mod-iwinfo:** habilita o acesso do pacote iwinfo por meio de Remote Procedure Calls - RPC, onde este será utlizado junto com o pacote anterior para a requisição de dados.

No script a seguir(que pode ser executado via terminal) faremos uso do OPKG, o

gerenciador de pacotes do OpenWrt, para realizar as modificações necessárias para que possamos concluir parte do processo de configuração inicial dos pontos de acesso:

```
opkg update
opkg install uhttpd-mod-ubus rpcd-mod-iwinfo
opkg remove uhttpd --force-depends
rm /etc/config/uhttpd
opkg install uhttpd
```

No script acima, podemos observar na primeira linha que é dado o comando *"opkg update"*, responsável por realizar as buscas no repositório de pacotes oficial do OpenWrt para verificar os pacotes disponíveis. Na segunda linha temos a instalação dos pacotes *uhttpd-mod-ubus* e *rpcd-mod-iwinfo*. Da terceira linha até a quinta, é realizada, consecutivamente, a desinstalação do pacote *uhttpd*, a remoção do arquivo de configuração deste mesmo pacote e depois é realizado a instalação do pacote novamente. A objetivo desta última etapa do script é forçar a atualização do pacote e a criação de um novo arquivo de configuração, para evitar que uma versão antiga do pacote *uhttpd* e seu arquivo de configuração no padrão antigo sejam utilizados.

A próxima etapa consistem na configuração das permissões de acesso por RPC, por meio da criação de um arquivo de configuração de permissões de usuários. Para isso, criamos o arquivo *"superuser.json"* no diretório *"/usr/share/rpcd/acl.d/"* e colocamos o seguinte texto no arquivo:

```
{
        "superuser": {
                "description": "Super_user_access_role",
                "read": {
                        "ubus": {
                                "*": [ "*" ]
                        },
                        "uci": [ "*" ]
                },
                "write": {
                        "ubus": {
                                "*": [ "*" ]
                        },
                        "uci": [ "*" ]
                }
        }
}
```

Após a realização destes procedimentos, é necessário reiniciar o ponto de acesso. Após o processo de boot, o ponto de acesso com OpenWrt configurado já estará apto a funcionar em conjunto com o Tiúba Wi-Fi Manager com todas as funcionalidades disponíveis no protótipo desenvolvido.

8.2 Conhecendo o Tiúba Wi-Fi Manager

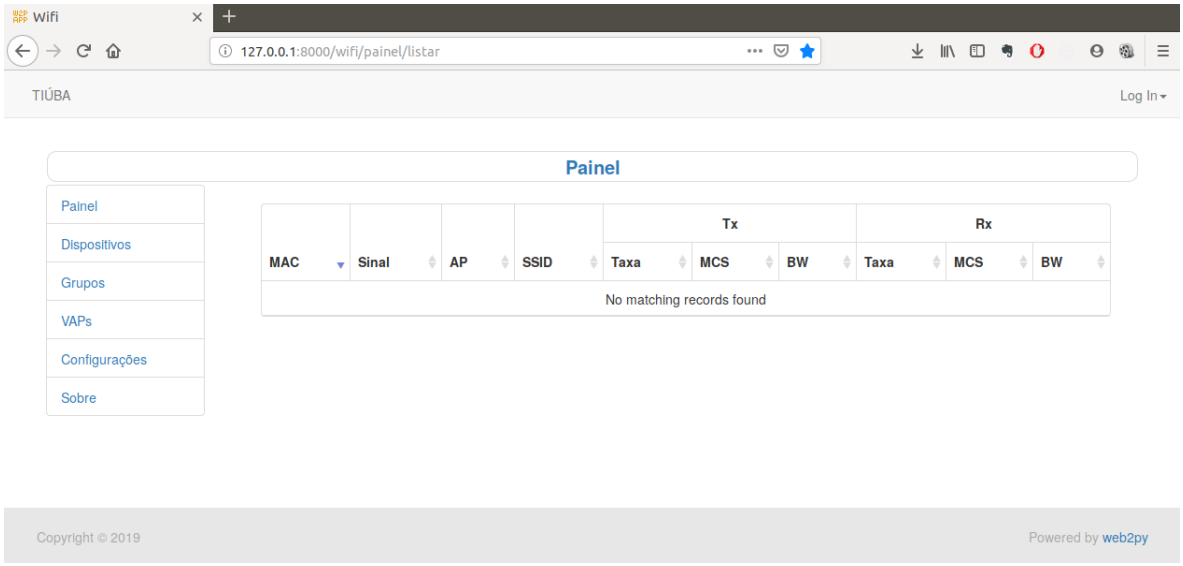
Atualmente o protótipo do Tiúba Wi-Fi Maneger é dividido em seis seções, nas quais podemos acessar as funcionalidades disponíveis para realizar o gerenciamento da rede sem fio. A seis seções são: Painel, Dispositivos, Grupos, VAPs, Configurações e Sobre.

A seguir, cada seção dessa será explicada melhor, exemplificando como elas estão dispostas no protótipo.

8.2.1 Painel

A seção Painel é o local onde são exibidos os usuários conectados aos pontos de acesso gerenciados pelo TWM.

Figura 26 – Painel de clientes conectados do Tiúba Wi-Fi Manager



Fonte: O autor.

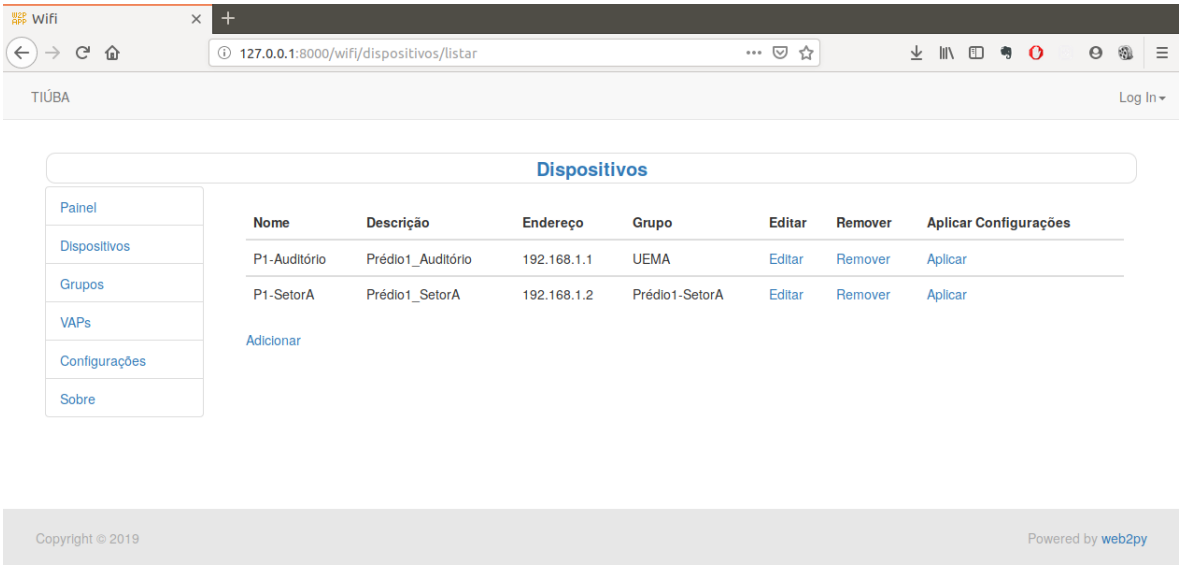
A Figura 26 mostra a seção chamada Painel, nela é possível ver informações sobre os dispositivos conectados aos pontos de acesso gerenciados pelo TWM. Para cada dispositivo conectado, é possível ver informações como endereço MAC, a potência de sinal recebido, o ponto de acesso ao qual ele está conectado, o SSID ao qual ele está conectado, as taxas de tx e rx, MCS (tx e rx) e a largura de banda do canal (tx e rx) que eles estão utilizando.

Tudo isso disponível em uma única tabela que pode ser ordenada por qualquer dos itens citados anteriormente, de acordo com a escolha do usuário do sistema.

8.2.2 Dispositivos

Nesta seção é onde o TWM disponibiliza a funcionalidade de administrar os dispositivos gerenciados pelo mesmo. É nesta parte que o usuário pode, no âmbito dos pontos de acesso, adicionar, editar, remover os dispositivos no sistema e aplicar as configurações (que será abordado melhor em outra seção deste trabalho).

Figura 27 – Painel de dispositivos gerenciados pelo Tiúba Wi-Fi Manager



Fonte: O autor.

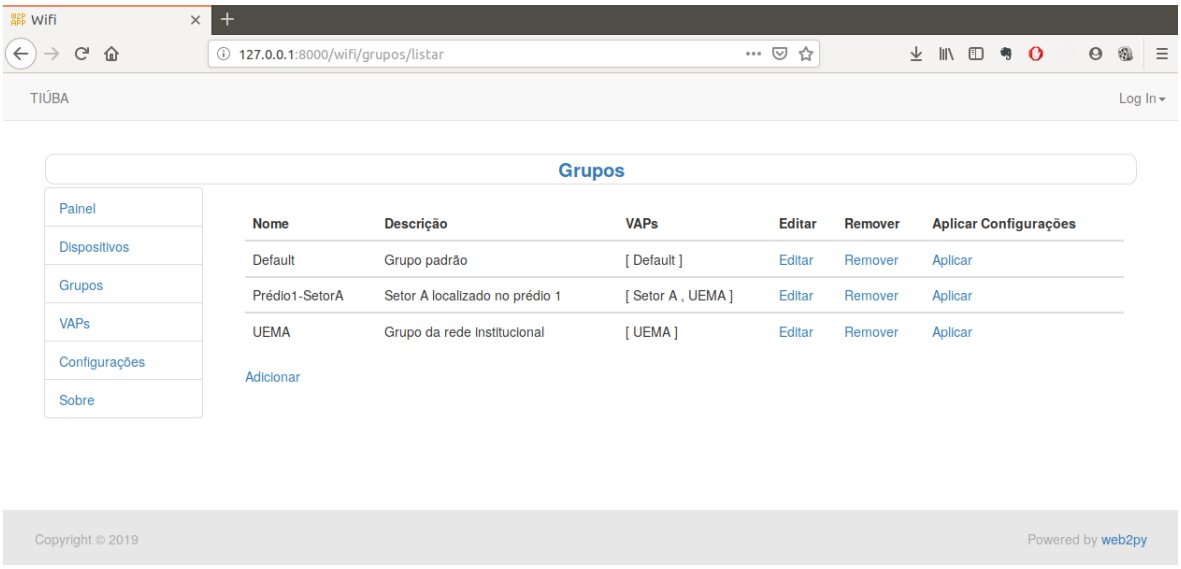
Na Figura 27 podemos observar a tela principal da seção Dispositivos, nela temos uma tabela contendo todos os pontos de acesso cadastrados no sistema, junto com algumas informações sobre os mesmos, dentre as quais podemos citar:

- **Nome:** é nome pelo qual se identifica o ponto de acesso Wi-Fi;
- **Descrição:** é uma forma descrever melhor os dispositivos, passando mais informações básicas importantes sobre o ponto de acesso;
- **Endereço:** endereço IP dos dispositivo a ser gerenciado pelo TWM;
- **Grupo:** é o grupo ao qual um determinado ponto de acesso está vinculado. Todo ponto de acesso deve estar vinculado à um grupo previamente criado na seção de Grupos.

8.2.3 Grupos

Na seção Grupos é onde o usuário do sistema poderá gerenciar os grupos de pontos de acesso disponíveis no sistema. É nesta parte que o usuário, no âmbito de grupos de pontos de acesso, pode adicionar, editar, remover os dispositivos no sistema e aplicar as configurações (que será abordado melhor em outra seção deste trabalho).

Figura 28 – Painel de grupos gerenciados pelo Tiúba Wi-Fi Manager



Fonte: O autor.

Na Figura 28 podemos observar a tela principal da seção Grupos, nela temos uma tabela contendo todos os grupos cadastrados no sistema, junto com algumas informações sobre os mesmos, dentre as quais podemos citar:

- **Nome:** é nome pelo qual se identifica o grupo de pontos de acesso;
- **Descrição:** é uma forma descrever melhor os grupos de pontos de acesso, passando mais informações básicas importantes sobre o grupo;
- **VAPs (Virtual Access Points):** são os pontos de acesso virtuais (será explicado com mais detalhes na próxima seção) cadastrados no sistema e que podem ser vinculados a cada grupo.

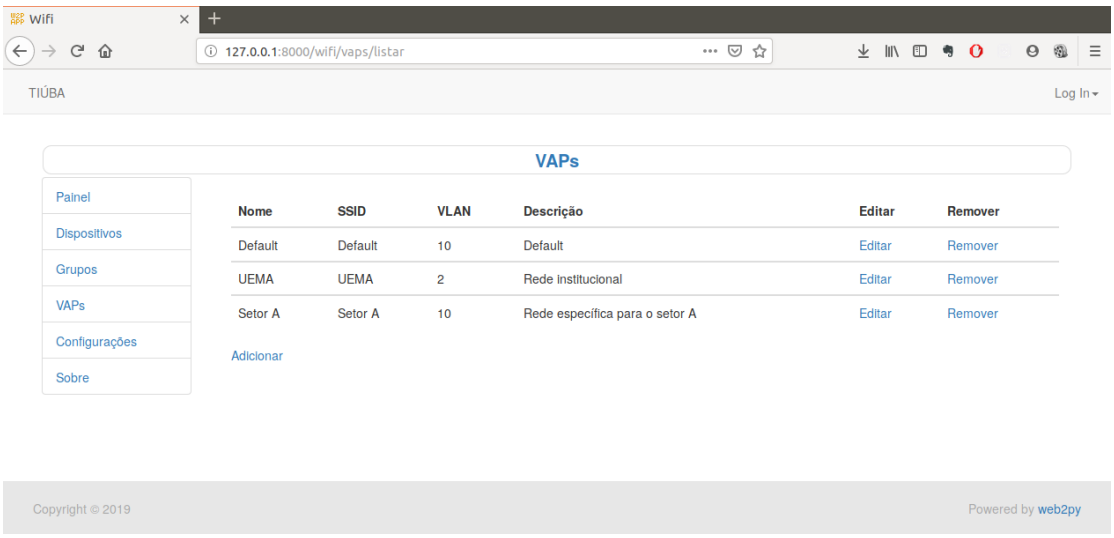
8.2.4 VAPs - Virtual Access Points

Na seção VAPs é onde o usuário do sistema poderá gerenciar os perfis de pontos de acesso virtuais disponíveis no sistema. O OpenWrt pode permitir que sejam criados múltiplas redes Wi-Fi (chamados Virtual Access Point - VAP) em único ponto de acesso,

essa funcionalidade depende do hardware utilizado. Geralmente os equipamentos permitem a criação de até quatro porntos de acesso virtuais em cada ponto de acesso real.

Nesta seção que o usuário, no âmbito de VAPs, pode adicionar, editar, remover os dispositivos no sistema e aplicar as configurações (que será abordado melhor em outra seção deste trabalho).

Figura 29 – Painel de VAPs gerenciados pelo Tiúba Wi-Fi Manager



Fonte: O autor.

Na Figura 29 podemos observar a tela principal da seção VAPs, nela temos uma tabela contendo todos as VAPs cadastradas no sistema, junto com algumas informações sobre as mesmas, dentre as quais podemos citar:

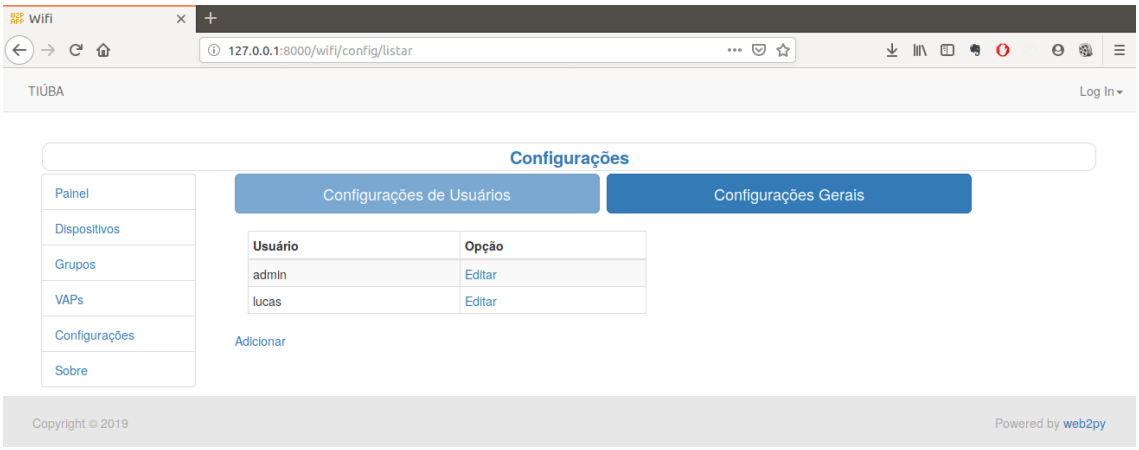
- **Nome:** define o nome pelo qual se identifica os pontos de acesso virtuais;
- **Descrição:** é uma forma descrever melhor os VAPs, passando mais informações básicas importantes sobre o VAP;
- **SSID:** define como será identificada a rede nos dispositivos Wi-Fi clientes;
- **VLAN:** define qual VLAN está vinculada ao determinado perfil de VAP.

8.2.5 Configurações

Nesta seção o usuário do sistema poderá gerenciar as configurações do próprio sistema. Ela está dividida em duas partes: Configurações de Usuários e Configurações Gerais.

A Figura 30 mostra o painel de configuração de usuários do sistema. nesta seção é possível adicionar, remover e editar os usuários. Cada usuário pode pertencer a um

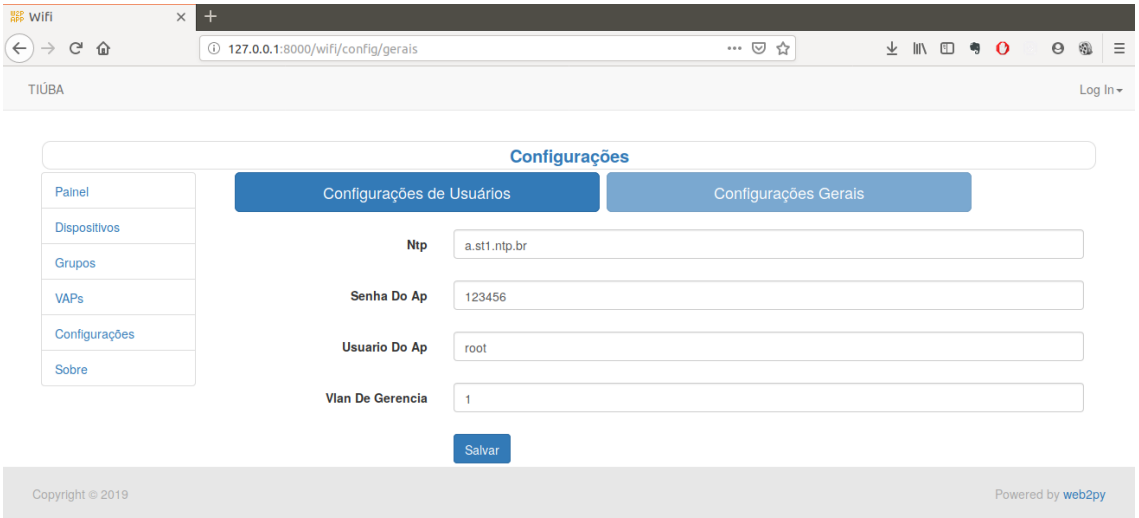
Figura 30 – Configurações de usuários do Tiúba Wi-Fi Manager



Fonte: O autor.

grupo, ou *"admin"* ou *"user"*. O usuário do grupo *"user"* tem acesso apenas ao painel de dispositivos conectados, já o usuário do grupo *"admin"* tem acesso a realizar todas as modificações disponível no sistema.

Figura 31 – Configurações gerais do Tiúba Wi-Fi Manager



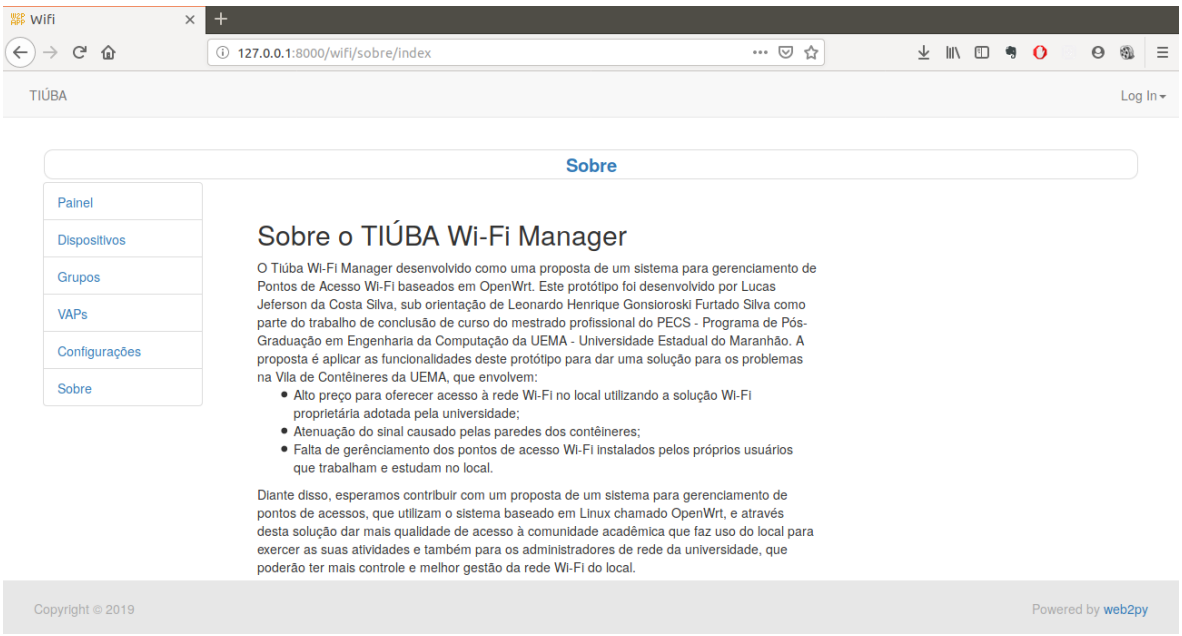
Fonte: O autor.

A Figura 31 exibe o formulário de configurações gerais do sistema. Atualmente (por se tratar de uma versão beta) apenas quatro itens estão disponíveis para serem modificados nesta seção, os itens descritos aqui fazem referência a informações sensíveis para o funcionamento correto do sistema.

8.2.6 Sobre

Esta seção, como mostra a Figura 32, serve apenas para mostrar conteúdos informativos sobre o sistema, dando ũm resumo sobre o objetivo e o histórico do Tiúba Wi-Fi Manager.

Figura 32 – Informações gerais sobre o Tiúba Wi-Fi Manager



Fonte: O autor.

8.2.7 Aplicação de Configurações e Persistência dos Dados

Algo importante sobre o funcionamento do TWM é que os registros inseridos, desde a seção de dispositivos até a seção de configurações, são salvos em um banco de dados do sistema e ficam persistentes após o usuário solicitar que os mesmos sejam salvos. Mas isso não implica que essas modificações já passam a funcionar nos pontos de acesso Wi-Fi a que as modificações se referem.

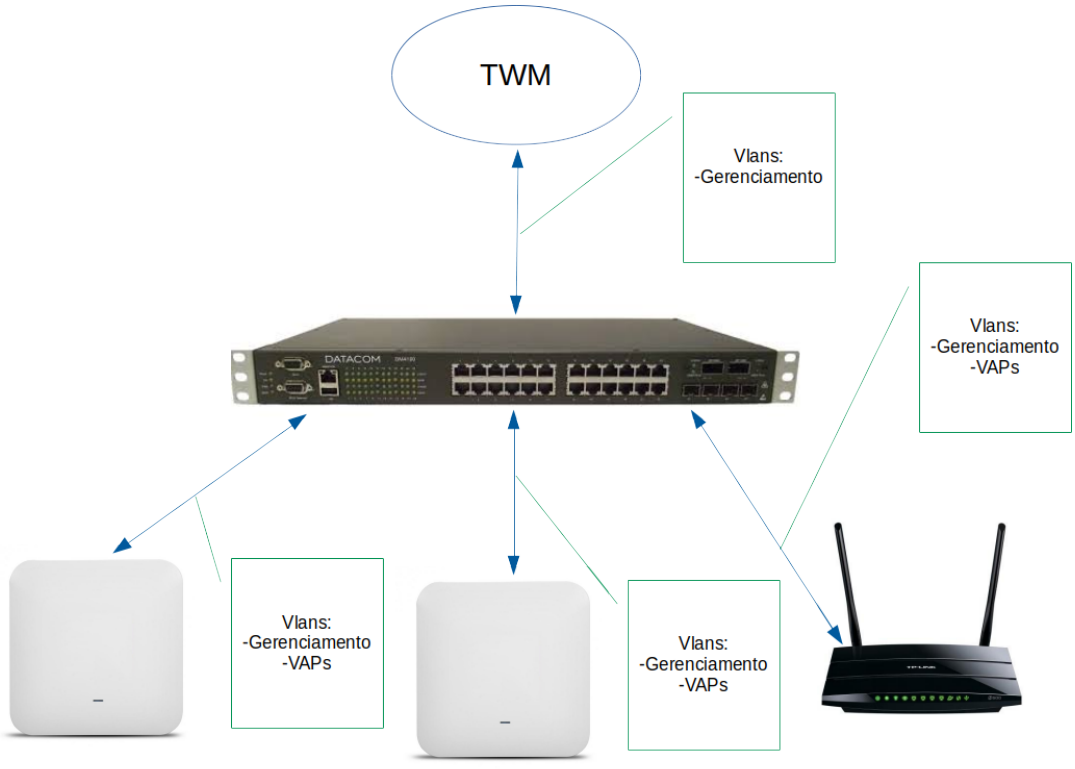
Para que as modificações salvas recentemente no sistema passe a vigorar nos pontos de acesso, é necessário que o usuário escolha duas formas de aplicar as configurações: para todo o grupo de pontos de acesso ou para apenas um ponto de acesso. Como podemos ver na Figura 27 e na Figura 28, estas configurações ainda precisam ser aplicadas de forma manual pelo usuário.

8.3 Funcionamento do Tiúba Wi-Fi Manager

Por se tratar de um sistema para gerenciar dispositivos Wi-Fi, o TWM não foi inicialmente desenvolvido com o intuito de oferecer outros serviços de rede, como é o caso dos serviços de DHCP, firewall e outros. Além disso, o TWM foi desenvolvido como um possível solução para resolver o problema de acesso Wi-Fi na Vila de Contêineres da UEMA e como a própria rede da UEMA já conta com estes serviços citados, não foi uma preocupação levada em consideração no decorrer do desenvolvimento do projeto, o que não exclui a possibilidade de serem feitas futuras inclusões de funcionalidades no projeto.

Para que toda a solução funcione corretamente, é necessário ter cuidado de configurar os ativos de rede corretamente e de deixar passando, nas portas onde os pontos de acesso estão conectados e onde o , as VLANs necessárias para que os dados possam trafegar corretamente sem nenhum problema.

Figura 33 – Arquitetura de funcionamento do Tiúba Wi-Fi Manager



Fonte: O autor.

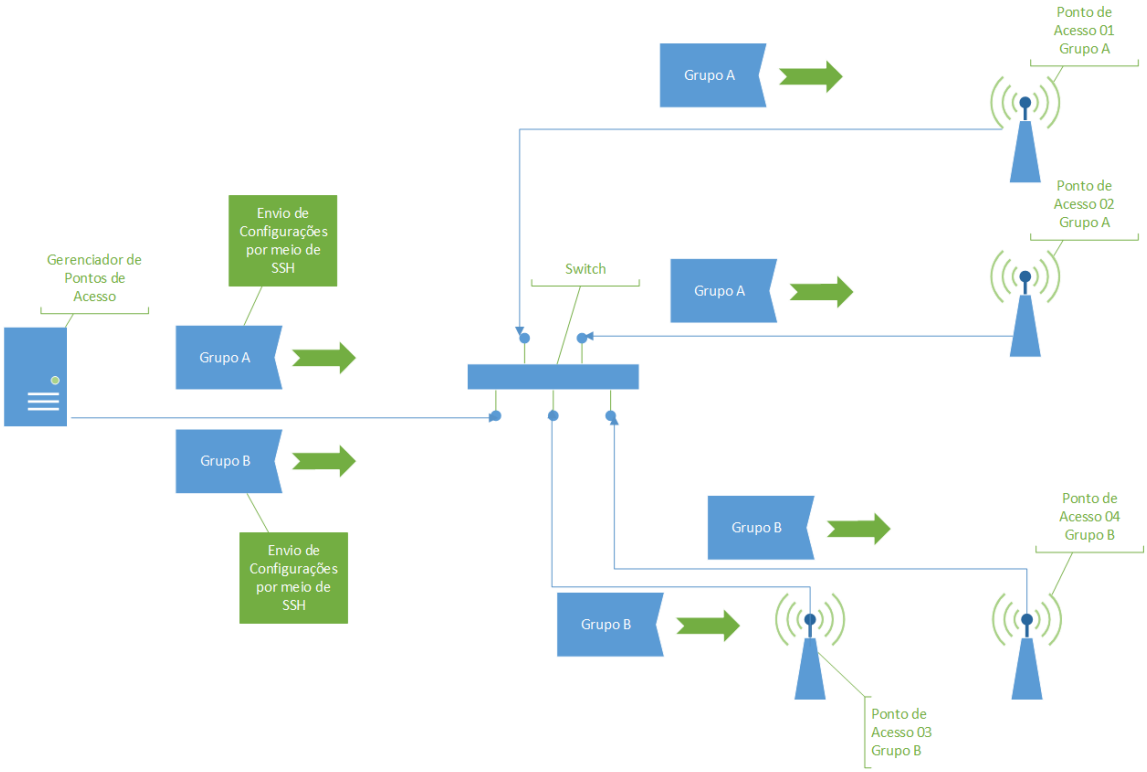
A Figura 33 mostra um exemplo simplificado de como deve ser a arquitetura da solução de rede sem fio que está sendo proposta. Nela podemos ver o que podemos entender como o host onde o TWM está hospedado estando conectado à um switch gerenciável, que está conectado aos pontos de acesso. A Figura 33 também mostra, nas conexões entre

o TWM e os pontos de acesso ao switch, as VLANs que devem estar configuradas para trafegar nas suas respectivas portas.

Necessário salientar que, em situações onde se faz uso de múltiplos VAPs, todas as VLANS, inclusive a de gerência, precisam passar como *"tagged"* nas portas dos switches onde os pontos de acesso estarão conectados. Caso só seja utilizado um VAP e o usuário não queira investir recursos na compra de switches gerenciáveis, basta utilizar tudo em VLAN 1 como *"untagged"*.

Após efetuadas as configurações de portas de switches e realizadas todas as configurações descritas acima, o TWM estará pronto para ser utilizado para ajudar no gerenciamento da rede sem fio do ambiente onde ele for instalado. A Figura 34 mostra como é feito o envio das configurações por meio do protocolo SSH, onde o usuário é o responsável por decidir quando as configurações serão enviadas para os pontos de acesso e o TWM escolhe quais configurações cada ponto de acesso irá receber, de acordo com o grupo ao qual eles pertencem.

Figura 34 – Envio de configurações por SSH

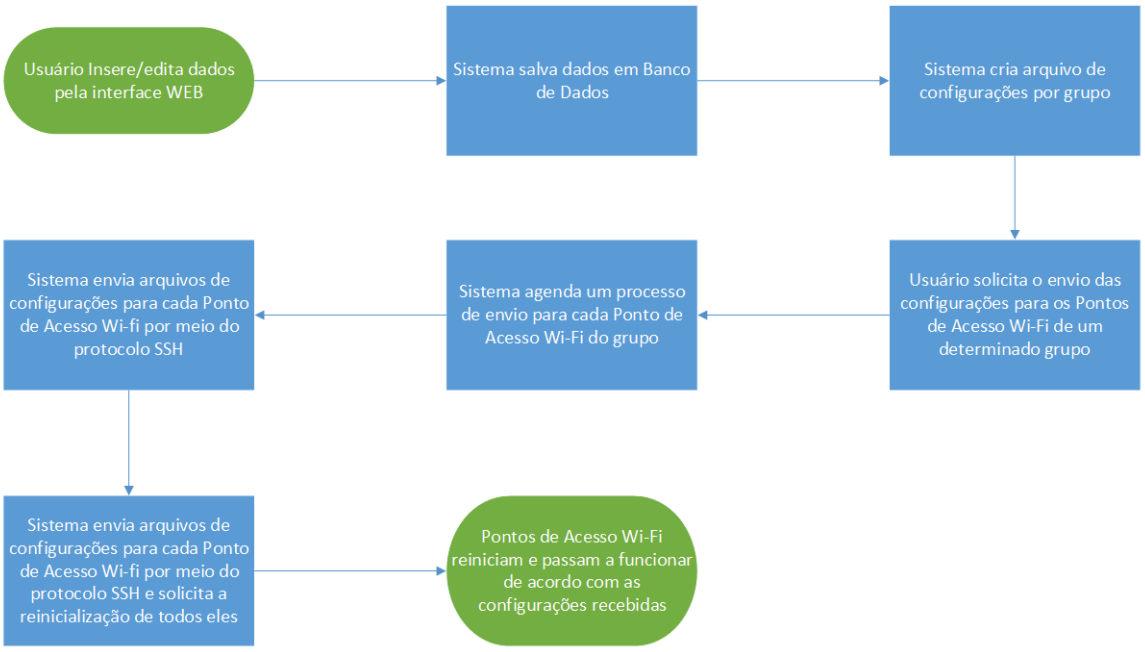


Fonte: O autor.

Podemos ver na Figura 35 um fluxograma referente ao processo de configuração de um ponto de acesso no TWM desde o início (inserção de dados no sistema por parte do usuário), passando pela parte de comunicação de rede através do protocolo SSH até

chegar no seu fim com a efetivação da configuração e funcionamento dos pontos de acesso de acordo com que está no TWM.

Figura 35 – Fluxograma do processo de configuração dos pontos de acesso por meio do protocolo SSH.

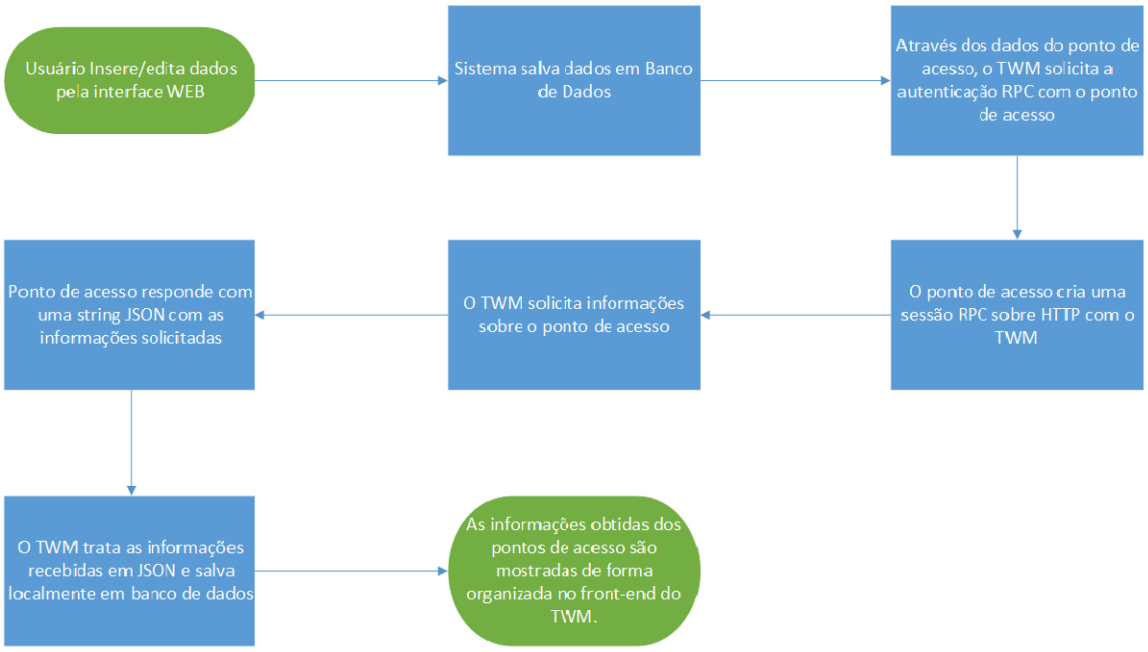


Fonte: O autor.

Já a Figura 36 nos dá uma visão, por meio de outro fluxograma, de como funciona o processo de aquisição periódica de dados que o TWM faz para obter informações dos pontos de acessos cadastrados. Este processo se dá por meio de RPC - Remote Procedure Calls (no português: chamadas de procedimento remoto), que se utilizam da estrutura do JSON para organizar e estruturar os dados e funciona sobre o protocolo de rede HTTP.

Por meio deste processo mostrado na Figura 36 é possível ter uma visão de como funciona o JSON-RPC e assim entender como é feita essa aquisição de dados dos pontos de acesso (informações de rede, configurações do dispositivo, dados de usuários conectados, etc), que se inicia desde a inserção dos dados no sistema por parte do usuário, passando pelo processo autenticação e autorização entre o TWM e cada ponto de acesso e finalizando com o TWM mostrando as informações de forma organizada ao usuário no front-end do sistema.

Figura 36 – Fluxograma do processo de aquisição periódica de dados dos pontos de acesso por meio de JSON-RPC sobre HTTP.



Fonte: O autor.

Este processo de aquisição de dados ocorre de forma periódica e atinge todos os pontos de acesso, sendo possível modificar a periodicidade com que esse processo acorre de acordo com a necessidade. Para a visualização de clientes conectados em cada ponto de acesso, foi utilizado um processo de aquisição de dados que ocorre a cada 10 segundos.

8.4 Custos de implantação

É importante mencionarmos quais são os custos envolvidos na implantação desta solução, para ter noção do quanto isso poderá custar aos cofre públicos. Como o TWM funciona em conjunto com dispositivos baseados em Openwrt, é possível obter um dispositivo compatível das seguintes maneiras: comprando um dispositivo que vem de fábrica com o Openwrt instalado ou instalando o Openwrt em um equipamento (já comprado ou que ainda será comprado) que tenha suporte para o Openwrt.

Tabela 16 – Custo de aquisição dos pontos de acesso Wi-Fi baseados em Openwrt.

Ponto de acesso	**Preço**
Ponto de acesso já comprado e com suporte para Openwrt	$0.00
Ponto de acesso (802.11n 2.4GHz MIMO-2x2 300Mbps) de linha doméstica de marcas conhecida	$25.00 (R$100)
Ponto de acesso (802.11n 2.4GHz MIMO-2x2 300Mbps) já com o Openwrt de fábrica	$20.00 (R$80)

Fonte: O autor.

Na Tabela 16 é possível observar os custos unitários de aquisição dos pontos de acesso. Considerando as abordagens mostradas no Capítulo 7, para esta solução, escolhemos utilizar a abordagem indoor completa, onde é sugerida a instalação de um ponto de acesso por contêiner.

A escolha dessa abordagem se deu pela melhor qualidade de sinal oferecida e o custo por ponto de acesso ser reduzido, facilitando assim a implantação da solução. Como a abordagem indoor completa sugere a utilização de 14 unidades de pontos de acesso, na pior das situações o custo com equipamentos seria de $350,00 dólares (ou R$1400,00 reais) ou, na melhor das hipóteses, a custo de aquisição zero no caso da utilização de equipamentos compatíveis com o Openwrt já disponíveis para uso na universidade.

8.5 Equipamentos utilizados nos testes

Para a realização dos testes durante a implementação do protótipo do TWM, foram utilizados dois tipos de hardwares distintos. O da Figura 37 é o equipamento modelo WDR3500 da fabricante Tp-Link, que teve o seu firmware substituído pelo Openwrt, para que pudesse funcionar em conjunto com o TWM.

Na Figura 38 temos a imagem do ponto de acesso OEM importado da China. Este equipamento já vem com Openwrt instalado, sendo necessário apenas realizar a atualização e configuração do Openwrt, para a versão já especificada anteriormente no início do capítulo.

Figura 37 – Roteador Wi-Fi modelo WDR3500 da fabricante Tp-Link.

Fonte: (TPLINK, 2019).

Figura 38 – Roteador Wi-Fi OEM importado da China com o Openwrt de fábrica.

Fonte: O autor.

9 Conclusão

Os objetivos deste trabalho foram alcançados. Como foi proposto, foi desenvolvido um sistema para gerenciamento de pontos de acesso Wi-Fi com interface web para prover gerenciamento centralizado dos pontos de acesso, possibilitando a configuração de múltiplos equipamento de forma automatizada e a visualização dos dispositivos conectados. Essa comunicação é feita por meio de uma conexão SSH, que provê uma comunicação ponto-a-ponto com segurança garantida pela criptografia utilizada para evitar que dados sensíveis possam a vir cair nas mãos de pessoas mau intencionadas.

O que também separa a solução proposta da grande maioria dos pontos de acesso simples, destinados a ambientes domésticos e comercializado pela maioria dos fabricantes, é a capacidade de configuração de múltiplos perfis de redes Wi-Fi (mostrando múltiplos SSIDs) e a possibilidade de atrelar cada perfil a uma VLAN diferente, com o objetivo de separar serviços e clientes, para oferecer ambientes diferentes em um mesmo ponto de acesso.

Por estar em fase inicial, o TWM não consegue competir em oferta de recursos e funcionalidades com a solução proprietária adquirida e definida como padrão da UEMA, mas oferece (a custos bem mais atraentes) uma opção para situações em que a utilização da solução padrão da UEMA se torna muito cara para ser usada e em situações que se deseja evitar a utilização de pontos de acesso de linha domésticas, sem gerenciamento centralizado.

Através deste trabalho deu-se início mais sólido a um projeto que poderá ser ampliado e expandido para diversas situações da vida real, ajudando muitas pessoas e empresas que desejam ter acesso à um serviço de gerenciamento de rede Wi-Fi com recursos um pouco mais sofisticados, a um custo mais acessível. E com isso, foi possível dar entrada em dois processos de propriedade intelectual, um processo de registro de software e um processo de registro de patente.

O processo de registro de software do Tiúba Wi-Fi Manager foi solicitado visando garantir a propriedade intelectual do sistema desenvolvido. Foram encontrados alguns softwares semelhantes, como é o caso do Openwisp, que é um projeto de código aberto, e que utilizada JSON-RPC em todos os processos, tanto de configuração quanto de requisição de dados. Já o Tiúba Wi-Fi Manager utiliza o SSH para poder realizar a configuração dos pontos de acesso baseados em Openwrt e utiliza JSON-RPC apenas na requisição de informações dos pontos de acesso.

Também está sendo realizado um processo de requisição de patentes, visando registrar o processo de configuração de pontos de acesso baseado em Openwrt utilizando

o protocolo SSH. Para a realização deste processo, foram encontrados algumas patentes existentes que oferecem soluções parecidas, algumas delas podem ser vistas a seguir.

A patente CN105430688A, que define uma arquitetura de LAN sem fio baseada em redes definidas por software, baseado na utilização do protocolo OpenFlow, da ferramenta Open vSwitch e Openwrt, onde o mesmo realiza um controle a nível de camada física, facilitar o processo de gerenciamento centralizado e de balanceamento de carga, sem se preocupar em apresentar uma solução mais voltada para a segurança no gerenciamento. Enquanto isso, a presente invenção tem como o objetivo facilitar o processo de configuração de múltiplos pontos de acesso baseados em Openwrt, utilizando um protocolo diferente, no caso o protocolo SSH, onde não se tem o controle tão robusto e versátil quanto no protocolo OpenFlow, mas inclui-se a vantagem da segurança, que é provido por meio da comunicação criptografada do protocolo SSH.

O processo descrito na patente USOO7305240B2 se difere da nossa solução por ser uma solução voltada para redes MAN sem fio, geralmente utilizado em telefonia móvel celular. Além disso, o processo descrito na patente em questão utiliza-se do protocolo SNMP (diferente do utilizado na presente invenção) para realizar o gerenciamento das estações rádio base. A presente invenção se difere pelo fato de ter como foco o gerenciamento de dispositivos de LAN sem fio e fazer uso de um protocolo diferente, que faz toda a diferença no processo de como é feito gerenciamento dos equipamentos.

O processo descrito na patente WO/2014/056437 tem como objetivo fazer o gerenciamento de pacotes wi-fi para realizar um bypass no gerenciamento dos clientes de uma rede sem fio, visando tirar do ponto de acesso a funcionalidade de gestão de gerenciamento de IP, cobrança, autenticação e outros processos. A presente invenção se difere da patente citada devido o fato de ela não ter o objetivo remover a gestão das funcionalidades citadas do ponto de acesso, mas sim configurar múltiplos pontos de acesso de forma centralizada, mantendo no ponto de acesso as funcionalidades citadas.

O processo descrito na patente WO/2014/094483 descreve o processo de configuração de múltiplos Service Set Identifiers – SSIDs, visto que isto é feito por meio de um WEBUI. A presente invenção não se limita a apenas configuração de SSIDs de pontos de acesso, mas também a parâmetros de rede, parâmetros de rádio e parâmetros de serviços do sistema Openwrt em geral, que vão mais além do processo mostrado na patente em questão.

Com essas informações é possível concluir que o projeto tem relevância tanto no sentido acadêmico como no comercial, abrindo portar para a pesquisa de soluções envolvendo gerenciamento de WLANs utilizando tecnologias gratuitas em suas amplas possibilidades, além de abrir possibilidade de comercialização de um produto resultante do desenvolvimento mais forte da solução proposta neste trabalho e também de pesquisas que podem vir a ser desenvolvidas utilizando este trabalho como base. Além de todas

as possibilidades, este trabalho trouxe um retorno para a UEMA e para o Programa de Pós-Graduação em Engenharia da Computação e Sistemas - PECS, por meio do processo de registro de software, que já foi encaminhado e do processo de registro de patente, que está sendo desenvolvido.

10 Trabalhos Futuros

Como a proposta do trabalho era atender a Vila de Contêineres da UEMA e na universidade já existe uma política consolidada de DHCP e firewall, o foco do desenvolvimento do protótipo do Tiúba Wi-Fi Manager acabou sendo em oferecer as funcionalidades proposta e a comunicação dos clientes com outros recursos da rede. Não foi direcionado esforços para o desenvolvimento de outras funcionalidades como DHCP, firewall, recursos de dignóstico de rede e outras ferramentas que poderiam facilitar a vida do usuário do TWM.

Sendo assim, foram definidos como trabalhos futuros as seguintes oportunidades:

- **Adição de recursos:** inclusão de funcionalidades como DHCP, firewall, rotas, ferramentas de diagnóstico e outras funcionalidades, podem abrir o leque de utilização do WPM para clientes domésticos e pequenas empresas;
- **Criação de uma build própria baseada no OpenWrt:** este trabalho poderá facilitar o processo de configuração da solução, facilitando a vida de quem vai configurar e abrindo a possibilidade de comercialização de um equipamento pronto de fábrica para utilização junto com o PWM.

Referências

BIRRELL, A. D.; NELSON, B. J. **Implementing remote procedure calls**. *ACM Transactions on Computer Systems (TOCS)*, ACM, v. 2, n. 1, p. 39–59, 1984. Citado na página 77.

CARROLL, B. J. ***CCNA Wireless Official Exam Certification Guide***. Indianapolis: Cisco Press, 2009. Citado 8 vezes nas páginas 29, 30, 40, 41, 42, 43, 44 e 46.

CROCKFORD, D. ***The application/json Media Type for JavaScript Object Notation (JSON)***. [S.l.], 2006. Citado na página 77.

DEAN, T. ***Network+ guide to networks***. [S.l.]: Cengage Learning, 2012. Citado na página 76.

FOROUZAN, B. ***Comunicação de Dados e Redes de Computadores***. Porto Alegre: ed. 3, Bookman, 2006. Citado na página 29.

GARCIA, L. G. U. ***REDES 802.11 (Camada de Enlace)***. 2003. Disponível em: <http://www.gta.ufrj.br/grad/01_2/802-mac/>. Acesso em: 10.05.2015. Citado na página 29.

GAST, M. ***802.11 Wireless Networks: The Definitive Guide***. [S.l.]: ed. 2, O'Reilly, 2005. ISBN 0596100523. Citado na página 39.

GAST, M. ***802.11n: A Survival Guide***. [S.l.]: "O'Reilly Media, Inc.", 2012. Citado 2 vezes nas páginas 46 e 47.

GAST, M. ***802.11 ac: a survival guide: Wi-Fi at gigabit and beyond***. [S.l.]: "O'Reilly Media, Inc.", 2013. Citado 3 vezes nas páginas 47, 48 e 49.

GROUP, I. . W. et al. **Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications: higher-speed physical layer extension in the 2.4 GHz band**. *ANSI/IEEE Std 802.11*, 1999. Citado na página 42.

GROUP, I. 802.11a W. et al. **IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band**. *IEEE Std 802.11a-1999*, IEEE, p. 1–102, Dec 1999. Citado na página 41.

GROUP, I. 802.11n W. et al. **IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput**. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, p. 1–565, Oct 2009. Citado na página 45.

HARKINS, D.; CARREL, D. et al. ***The internet key exchange (IKE)***. [S.l.]. Citado na página 70.

HOLT, A.; HUANG, C. ***802.11 wireless networks security and analysis***. London: Springer, 2010. ISBN 978-1-84996-274-2. Citado 4 vezes nas páginas 35, 36, 37 e 39.

ITPRICE. 2019. Disponível em: <https://www.itprice.com>. Citado 2 vezes nas páginas 85 e 86.

JSONRPC. 2019. Disponível em: <https://www.jsonrpc.org/specification>. Citado 3 vezes nas páginas 77, 78 e 79.

MENEZES, A. F. et al. ***Linux: Administração de Redes***. [S.l.]: Alfamídia, 2002. Citado na página 65.

MODULARES, M. 2019. Disponível em: <http://mondialmodulares.com.br/wp-content/uploads/2017/06/arquivo.pdf>. Citado na página 81.

MORAES, A. ***Segurança em Redes: Fundamentos***. São Paulo: ERICA, 2010. ISBN 9788536503257. Citado 3 vezes nas páginas 36, 38 e 39.

O'HARA, B.; PETRICK, A. ***IEEE 802.11 Handbook: A Designer's Companion***. [S.l.]: Wiley, 2005. (IEEE standards wireless networks series). Citado na página 34.

OPENWRT. 2019. Disponível em: <https://openwrt.org/docs/start>. Citado 7 vezes nas páginas 51, 52, 53, 54, 56, 57 e 58.

PIERRO, M. D. ***Web2py: Complete Reference Manual***. Chicago, IL: Experts4Solutions, 2019. Citado 5 vezes nas páginas 59, 60, 61, 62 e 63.

PYTHON. ***Web2pyt***. 2019. Disponível em: <http://http://wiki.python.org.br/>. Citado na página 60.

SKORDOULIS, D. et al. **IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput WLANs**. *IEEE Wireless Communications*, v. 15, n. 1, p. 40–47, February 2008. ISSN 1536-1284. Citado na página 45.

STAHNKE, M. ***Pro OpenSSH***. [S.l.]: Apress, 2005. Citado na página 76.

STALLINGS, W. ***Redes de Sistemas de Comunicação de Dados: Teoria e Aplicações Corporativas***. Rio de Janeiro: ed. 5, Elsevier, 2005. Citado 2 vezes nas páginas 29 e 33.

STALLINGS, W. **Protocol Basics: Secure Shell Protocol**. *The Internet Protocol Journal*, v. 12, n. 4, p. 18–29, 2009. Citado 11 vezes nas páginas 65, 66, 67, 68, 69, 70, 71, 72, 73, 74 e 75.

TPLINK. ***Roteador Wireless WDR3500***. 2019. Disponível em: <https://www.tp-link.com/us/home-networking/wifi-router/tl-wdr3500/>. Citado na página 100.

UEMA. *UEMA e GOVERNO juntos em favor da educação superior no Maranhão*. 2019. Disponível em: <https://www.uema.br/2017/10/uema-e-governo-juntos-a-favor-da-educacao-superior-no-maranhao/>. Citado na página 81.

VASSIS, D. et al. **The IEEE 802.11g standard for high data rate WLANs**. *IEEE network*, IEEE, v. 19, n. 3, p. 21–26, 2005. Citado 2 vezes nas páginas 43 e 44.

VIEIRA, D. **IEEE 802.11**. 2005. Universidade Federal do Rio de Janeiro. Citado na página 25.

VINCI, O.; FERREIRA, P. **WiIP Sistema de Comunicação VoIP Wi-Fi**. São Paulo, 2007. Citado 5 vezes nas páginas 29, 30, 32, 33 e 34.

YLONEN, T.; LONVICK, C. ***The Secure Shell (SSH) Connection Protocol***. [S.l.], 2006. Citado 2 vezes nas páginas 73 e 75.

YLONEN, T.; LONVICK, C. ***The Secure Shell (SSH) Protocol Architecture***. [S.l.], 2006. Citado 3 vezes nas páginas 65, 71 e 73.