UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

VINÍCIUS GARCEZ SCHAURICH

Um caso de uso do *Protocol to Access White-Space databases*

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof Dr. Juergen Rochol
Co-orientador: Prof. Dr. Cristiano Bonato Both

Porto Alegre
2015

“Ninguém caminha sem aprender a caminhar,
sem aprender a fazer o caminho caminhado,
refazendo e retocando
o sonho pelo qual se pôs a caminhar.”

— Paulo Freire

AGRADECIMENTOS

Agradeço acima de tudo a meu pai, Wilmar, pelas posições firmes diante de todas as adversidade da vida. Posições que me serviram de exemplo de perseverança, força de vontade, respeito e amor.

Agradeço a minha mãe, Iria, que mesmo não estando aqui, com certeza olha por mim de algum lugar.

Agradeço a minhas irmãs, Sabrina e Aleteya, pela presença materna, mesmo passando pelas mesmas dores que eu.

Agradeço a minha vó, Ivone, que praticamente me criou e me cuidou nesses cinco anos de curso.

Agradeço a meu orientador, Prof. Juergen Rochol, e co-orientador, Prof. Cristiano Bonato Both, pela prontidão em atender os diversos pedidos de ajuda, independentemente da hora.

Agradeço aos demais professores com quem cursei as diversas matérias do curso, por exigir o máximo de mim, sempre.

Agradeço a meus amigos do grupo de redes, que sempre estiverem à disposição para ajudar, nem que fosse pra colocar uma música esdrúxula para me incomodar. Obrigado: Luís, Bondan, Anderson, Juliano, Germano, Catarina, Naldo, Nadal, Sheep e Alexandre.

Por fim, agradeço a meus demais amigos, uns de infância, uns que fiz ao decorrer dessa estadia em Porto Alegre. Obrigado: André, Paulo, Ruan, Lucas, Sogev, Guibe, Freitas e Muratore.

RESUMO

A demanda por taxas de dados cada vez maiores em redes sem fio exige que novas tecnologias sejam criadas para uma comunicação eficiente. Uma das formas de tornar o acesso ao espectro mais eficiente é a utilização dinâmica do espectro pelos rádios. Na literatura, o uso de bases de dados de informações espectrais foi proposto como a principal técnica possibilitadora de rádios acessarem dinamicamente o espectro. Nos países onde esse tipo de acesso ao espectro é regulamentado, diversas bases de dados desse tipo têm surgido. Assim, o IETF criou o *Protocol to Access White-Space databases* (PAWS) para padronizar a comunicação entre essas bases de dados e os rádios. Por ser um protocolo novo, faltam implementações do PAWS na literatura. Assim, esse trabalho propõe uma implementação do PAWS e um caso de uso para exemplificar a atuação do protocolo.

Palavras-chave: Protocol to Access White-Space databases. IEEE 802.15.4.

An use case for the Protocol to Access White-Space databases

ABSTRACT

A higher data rate demands of wireless networks requires that new technologies use the spectrum efficiently. One way to make the spectrum usage more efficient is the spectrum's dynamic utilization by radios. The academic community points the use of white-space databases as the main technique for radios to access dynamicly the spectrum. Numerous white-space database have arised as the governmental regulators of some countries have regulated radios dynamical spectrum access via this technique. For this reason, IETF standarized the communication between radios and white-space databases as the Protocol to Access White-Space databases. As in any novel protocol, the literature lacks PAWS implementations. Therefore, this work proposes a PAWS implemenation and an use case to exemplify its operation.

Keywords: Protocol to Access White-Space databses, IEEE 802.15.4.

LISTA DE ABREVIATURAS E SIGLAS

UP	Usuário Primário
US	Usuário Secundário
WSDB	*White-Space Database*
IETF	*Internet Engineering Task Force*
PAWS	*Protocol to Access White-Space Databases*
DSA	*Dynamic Spectrum Access*
ISM	*Industrial, Scientific and Medical*
FCC	*Federal Communications Commission*
ECC	*European Commodity Clearing*
Ofcom	*Office of Communications*
CR	*Cognitive Radio*
ITM	*Irregular Terrain Model*
JSON	*JavaScript Object Notation*
USRP	*Universal Software Radio Peripheral*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Desde a criação dos primeiros órgãos que regulamentam a exploração do espectro de radiofrequências como meio de comunicação, acreditou-se que novos transmissores interfeririam com os usuários preestabelecidos em redes sem fio (STAPLE; WERBACH, 2004). Baseados nessa visão conservadora, os órgãos reguladores estipularam que todo sistema sem fio deveria possuir uma licença de operação para poder atuar. Nesse meio tempo, praticamente todo o espectro utilizável para radiocomunicação foi licenciado e essa política estática resultou na escassez de espectro para novos usuários. Por exemplo, no Brasil, cujo órgão regulador é a ANATEL, esse licenciamento é feito através de leilões, onde se alocam as faixas do espectro para as empresas vencedoras das licitações. Em contrapartida, estudos mostram que o uso do espectro é ineficiente, ou seja, é grande o número de faixas ociosas do espectro (HAN et al., 2010; DZULKIFLI; KAMARUDIN; RAHMAN, 2011). Essas faixas ociosas do espectro, chamadas na literatura de *white-spaces*, surgem pela operação intermitente das companhias de telecomunicação em suas faixas licenciadas.

Porém, com o surgimento e sucesso da tecnologia Wi-Fi, que utiliza faixas não-licenciadas do espectro, os órgãos reguladores perceberam as capacidades do compartilhamento espectral (STAPLE; WERBACH, 2004). Em redes não-licenciadas não existe regulamentação de usuários, assim, diversas tecnologias acessam e compartilham a mesma faixa de espectro, utilizando canais diferentes dentro dessa faixa ou esperando outro usuário terminar sua comunicação para poder operar. A ineficiência espectral nas faixas licenciadas do espectro poderia, então, ser superada por agregar compartilhamento espectral também às faixas licenciadas. Contudo seria necessária uma abordagem que se adequasse aos sistemas legado, baseados na política estática. Assim, o acesso oportunístico ao espectro de usuários que não possuem licença de operação em faixas licenciadas do espectro foi proposto (ZHAO; SADLER, 2007). Essa abordagem protege os usuários que adquiriram licenças de operação nas faixas licenciadas do espectro e, quando esses usuários não estiverem utilizando suas faixas do espectro, permite a comunicação de usuários que não possuem tais licenças de operação. Pela formatação hierárquica do acesso dos dispositivos nessa abordagem, usuários que possuem licenças de operação em faixas licenciadas do espectro são chamados de usuários primários (UP) e usuários que não possuem tais licenças são chamados de usuários secundários (US) na literatura.

Para que o US saiba quando pode se comunicar sem causar interferência com o UP, o US necessita conhecer o espectro vago em sua região de operação. Existem duas técnicas que agregam conhecimento espectral aos US: sensoriamento espectral e o uso de bases de dados

de *white-spaces* (WSDBs). Dentre essas técnicas, os órgãos reguladores julgaram o uso de WSDBs como a técnica mais economicamente viável para os US (KOUFOS, 2013). Assim, diversas bases de dados começaram a surgir para servir aos US nos países, onde o acesso oportunístico do espectro é regulamentado. Grandes empresas como a Google (GOOGLE, 2013) e a Microsoft (MICROSOFT, 2014), por exemplo, já disponibilizam serviços *Web* para acesso às suas WSDB. Com o crescente número de WSDBs surgindo, o *Internet Engineering Task Force* (IETF) decidiu padronizar a comunicação entre WSDBs e USs. O *Protocol to Access White-Space databases* (PAWS) (IETF, 2015) define as mensagens que o US e a WSDB devem trocar para que o usuário obtenha uma lista de canais vagos em sua região de operação. Por ser um protocolo novo, que ainda está em processo de padronização, apenas uma implementação do PAWS foi encontrada na literatura (GHOSH et al., 2015), publicado durante o desenvolvimento desse trabalho de conclusão de curso. Outras implementações proprietárias, anteriores ao desenvolvimento desse trabalho, como a implementação da Google (GOOGLE, 2013) e da iconectiv (ICONECTIV, 2013) estão defasadas e motivaram a proposta desse trabalho de conclusão de curso: uma implementação do PAWS.

A implementação do protocolo realizada pelo autor se baseia em dois módulos: o *Database Service Requester* e o *Database Service Replier*. Para que as empresas desenvolvedoras de rádios e WSDBs não necessitem implementar soluções próprias do protocolo, a implementação desses módulos tentou ser a mais transparente possível, para poderem ser agregadas aos sistemas proprietários. Apesar do protocolo ter sido concebido para ser utilizado em cenários de acesso oportunístico ao espectro, não há restrição de faixa de frequências que o PAWS deve ser utilizado. Para exemplificar o uso do protocolo é utilizada uma rede IEEE 802.15.4 (IEEE, 2011b), que opera na faixa não-licenciada de 2.4 GHz, contendo dois rádios que se comunicam em um dos canais vagos informados por uma WSDB. Esse tipo de rede foi escolhido pois seus dispositivos sofrem com interferência quando compartilham canais com dispositivos Wi-Fi e podem utilizar técnicas de conhecimento espectral para minimizar essa interferência (NARAMPANAWE et al., 2010), além de serem dispositivos baratos e de fácil obtenção.

O restante deste trabalho está estruturado da seguinte forma. O Capítulo 2 aprofunda conceitos envolvidos em acesso dinâmico ao espectro (DSA), discorrendo sobre as técnicas de conhecimento espectral e apresenta as principais características do PAWS e de redes IEEE 802.15.4. No Capítulo 3 são apresentados os trabalhos relacionados à proposta desse trabalho de conclusão de curso, abordando trabalhos que implementam o PAWS e trabalhos que agregam conhecimento espectral a redes 802.15.4. Já no Capítulo 4, a proposta do trabalho é apresentada, especificando a arquitetura dos módulos *Database Service Requester* e *Database Service*

Replier, e do sistema criado para o caso de uso. Finalmente, o Capítulo 5 conclui esse trabalho e apresenta possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esse Capítulo apresenta uma visão geral das bases teóricas necessárias para se entender a necessidade da padronização do protocolo PAWS. Primeiro, na Seção 2.1, apresenta-se DSA, como uma política de acesso ao espectro que vai de encontro à política estática em vigência na maioria dos países. Em seguida, na Seção 2.2, são abordadas as capacidades que os rádios necessitam para possibilitar essa evolução no acesso ao espectro. Na Seção 2.3, é apresentado o protocolo PAWS, abordando seu fluxo de operação, as funcionalidades definidas pelo protocolo para os dispositivos que o implementam e as mensagens trocadas pelos dispositivos definidas para cada uma dessas funcionalidades. Por fim, a Seção 2.4 apresenta as características dos dispositivos IEEE 802.15.4, em que tipo de aplicação são utilizados e os motivos pelos quais foram escolhidos para a implementação do caso de uso proposto nesse trabalho.

2.1 Acesso dinâmico ao espectro

Dada a demanda cada vez maior de conexões com altas taxas de dados por novas tecnologias, como 5G, e pelo crescente número de dispositivos sem fio, a política estática de acesso ao espectro têm ficado obsoleta (IRNICH et al., 2013). Assim, DSA tem sido cada vez mais discutido, na comunidade acadêmica, como uma possibilitadora dessa expansão do acesso ao espectro.

DSA é uma política de acesso ao espectro que se baseia na utilização dinâmica de faixas do espectro por múltiplos usuários, a fim de minimizar a ociosidade dessas faixas do espectro. Por exemplo, dois usuários podem coordenar suas operações para utilizar uma mesma faixa do espectro alternadamente, dado que não teriam necessidade de utilizar essa faixa do espectro em tempo integral. Assim, DSA é mais eficiente quanto ao uso do espectro do que a política estática. Existem três modelos em que DSA pode ser classificada (ZHAO; SADLER, 2007) e que são discutidos abaixo.

- **Modelo dinâmico de uso exclusivo –** pretende flexibilizar o acesso ao espectro mantendo a estrutura de licenciamento da política estática. Uma abordagem para esse modelo é repassar aos usuários licenciados certos direitos de propriedade sobre suas faixas do espectro. Assim, usuários licenciados poderiam comercializar suas faixas de espectro com outros usuários dado que não as utilizariam integralmente. Ainda, uma segunda abordagem é alocar dinamicamente a banda de cada usuário licenciado dadas as necessidades

desse usuário em determinado instante do tempo (LEAVES; HUSCHKE; TAFAZOLLI, 2002). Esse modelo, apesar de utilizar o espectro mais eficientemente que a política estática, não tem a possibilidade de eliminar *white-spaces*, já que sua estrutura permanece estática.

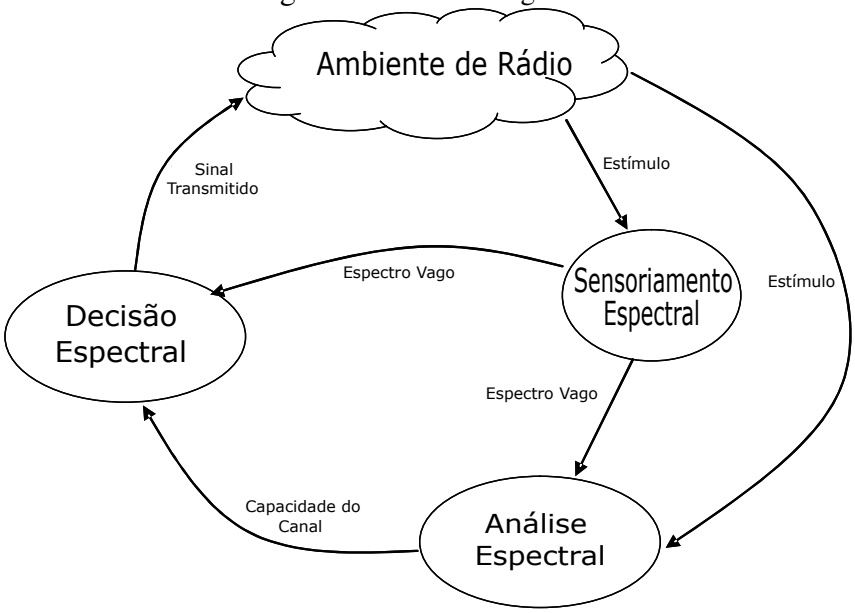
- **Modelo de compartilhamento aberto –** modelo aplicado em redes *industrial*, *scientific and medical* (ISM), como Wi-Fi e redes de sensores. Nesse modelo usuários podem alocar para si qualquer canal nessa faixa do espectro sem se preocupar com interferência. Portanto, soluções utilizando esse modelo tendem a ser tolerantes a erros e utilizar-se de *hardware* de baixa potência de transmissão. Apesar de ser o modelo ideal em DSA, por sua forma de compartilhamento completa do espectro, depende de faixas do espectro livres de usuários licenciados. Portanto, não interopera com os sistemas legados que ainda se baseiam na política estática.
- **Modelo acesso hierárquico –** baseia-se também na estrutura da política estática, ou seja, mantém o espectro alocado para usuários licenciados. Porém, pretende abrir o uso do espectro para usuários sem licença, impondo limitações na comunicação desses novos usuários. Esse modelo também possui duas abordagens distintas, sendo a primeira o acesso a qualquer banda com uma limitação de potência de transmissão abaixo da potência do ruído do UP. Assim, essa primeira abordagem é limitada a comunicações de curta distância. Já, a segunda abordagem baseia-se no uso oportunístico do espectro, ou seja, o US deve se aproveitar dos *white-spaces* deixados pelos UP em determinados instantes de tempo, para realizar sua comunicação. Um exemplo dessa abordagem é a norma IEEE 802.22 que define o uso dos *white-spaces* na faixa do espectro de TV (IEEE, 2011a). É o modelo mais amplamente estudando pela academia por se adaptar aos sistemas legados da política estática, como o modelo dinâmico de uso exclusivo, minimizando os *white-spaces*.

A abordagem de uso oportunístico do espectro, por exemplo, já é regulamentada pela *Federal Communications Commission* (FCC) dos EUA, pela *Office of Communications* (Ofcom) do Reino Unido e pela *European Commodity Clearing* (ECC) da Europa. Considerando essa abordagem, ou soluções que desejam sofrer o mínimo de interferência em redes ISM (NARAMPANAWE et al., 2010), os rádios necessitam ter a capacidade de identificar faixas do espectro que estejam ociosas e se reconfigurar para utilizar tais faixas. Rádios quem possuem tais capacidades são chamados de rádios cognitivos (CR), que são apresentados na Seção 2.2.

2.2 Rádio Cognitivo

CR é um conceito introduzido por Joseph Mitola no final dos anos 1990 (MITOLA, 2000). Tido como o futuro em dispositivos de rádio, defende que esses dispositivos devem ter a capacidade de conhecer o contexto em que estão ambientados, tomar decisões com base nos estímulos do meio e poder se reconfigurar a partir de suas decisões. Por essas características, CR é um grande habilitador de soluções que desejem acessar dinamicamente o espectro. Dadas as funcionalidades dos dispositivos cognitivos definidas por Mitola, quatro funções cognitivas principais foram enumeradas e formam o chamado ciclo cognitivo: sensoriamento, gerenciamento, mobilidade e compartilhamento espectrais (AKYILDIZ et al., 2006). O ciclo cognitivo é apresentado na Figura 2.1. Essa figura sumariza as funções de mobilidade e compartilhamento na função de análise espectral.

Figura 2.1 – Ciclo Cognitivo



Fonte: (AKYILDIZ et al., 2006)

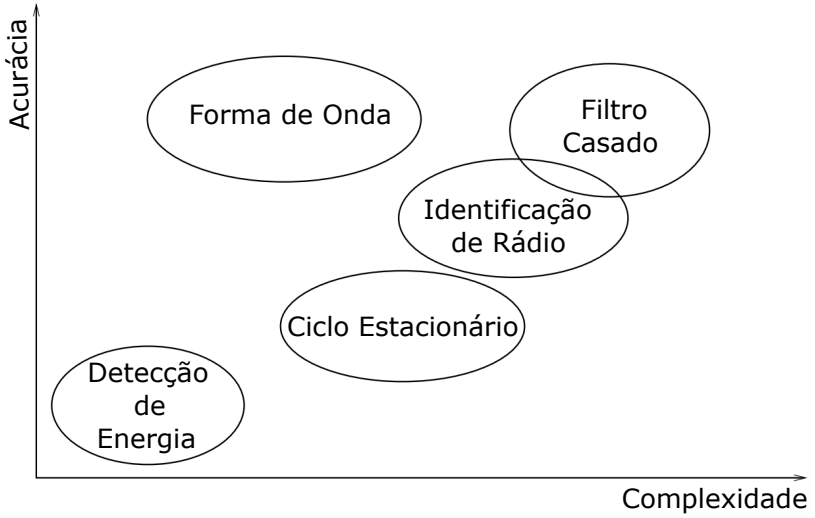
Sensoriamento é a função que torna o rádio consciente do contexto que está imerso. Nesse ponto do ciclo cognitivo, o rádio "sente" o espectro em sua área de atuação e analisa os canais sensoriados, classificando-os em vagos ou ocupados. A seguir, a função de gerenciamento tem por objetivo analisar os canais garantidamente vagos, resultantes do sensoriamento espectral e decidir qual dos canais vagos utilizar, dadas as necessidades do usuário. A terceira função no ciclo cognitivo, mobilidade, é definida pelo processo de mudança do canal de operação do usuário da forma mais transparente possível. Por fim, compartilhamento espectral define a coexistência entre múltiplos CR acessando um mesmo canal vago. Pelo enfoque desse

trabalho de conclusão de curso, a função de sensoriamento e seus principais problemas serão abordados na Subseção 2.2.1. Já, a Subseção 2.2.2 apresenta como solução para os problemas de sensoriamento espectral o uso de bases de dados de *white-spaces*.

2.2.1 Sensoriamento Espectral

Sensoriamento espectral é uma das principais funções de CR por possibilitar o rádio a conhecer o meio em que está inserido. Sem os resultados do sensoriamento, o rádio não tem a capacidade de identificar canais vagos no espectro e, consequentemente, não pode se reconfigurar para utilizá-los. Assim, os resultados de sensoriamento servem como entrada para as funções de gerenciamento e mobilidade espectrais (YUCEK; ARSLAN, 2009). Por ser uma função tão importante, foi estudada extensivamente pela comunidade acadêmica, que aponta diversas técnicas para identificação de canais vagos na região sensoriada. Cada técnica de sensoriamento espectral possui uma acurácia de identificação de canais vagos correspondente. Ao se analisar essa acurácia, observa-se uma relação direta com a complexidade da técnica utilizada, ou seja, quando mais acurada for a técnica na classificação dos canais entre vagos e ocupados, maior sua demanda de processamento. A Figura 2.2 representa essa relação entre acurácia e a complexidade de cada técnica de sensoriamento espectral.

Figura 2.2 – Relação entre acurácia e complexidade das técnicas de sensoriamento espectral



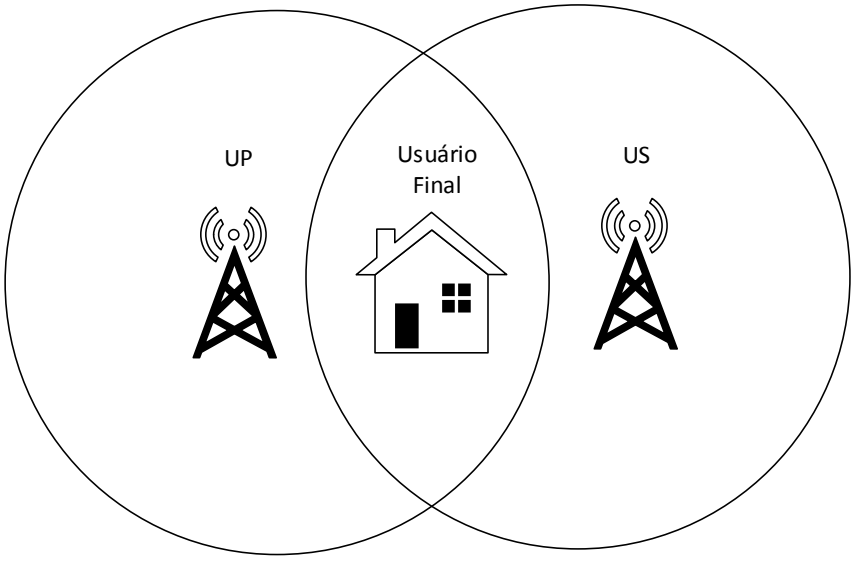
Fonte: (YUCEK; ARSLAN, 2009)

Dentre as principais técnicas de sensoriamento espectral, destacam-se duas: detecção de energia e sensoriamento baseado em forma de onda. Apesar de ser a técnica menos precisa,

detecção de energia é a mais simples e mais geral, por não necessitar de nenhuma informação prévia do sinal de UPs na região do US. Essa técnica baseia-se na comparação dos níveis de energia da região do US com um limiar, que depende dos níveis de energia do ruído da região. A determinação do limiar de energia necessário para proteção dos UPs na região de operação do US é um grande desafio na implementação dessa técnica e, por vezes, impede que o espectro seja utilizado da forma mais eficiente possível (GURNEY et al., 2008). Já, a técnica de sensoriamento baseado em forma de onda é a mais eficiente na relação entre acurácia e complexidade. Essa técnica baseia-se na correlação entre sinais do UP conhecidos pelo US e sinais sensoriados, portanto, só é aplicável a sistemas que tenham padrões de sinais conhecidos.

Independentemente da técnica, a função de sensoriamento espectral sofre com diversos desafios (YUCEK; ARSLAN, 2009). Por exemplo, para realizar o sensoriamento é necessário que o rádio possua um *hardware* poderoso, com altas taxas de amostragem de sinal, conversores analógico-digitais de alta resolução e processadores de sinal de alta velocidade. Além disso, sensoriamento sofre com o problema de nós escondidos, que pode acarretar na interferência de um US com a comunicação de um UP. Esse problema é apresentado na Figura 2.3, onde o US não enxerga o UP e tenta se comunicar com o usuário final, causando interferência em sua comunicação com o UP. Dados os problemas de sensoriamento espectral, uma nova abordagem foi introduzida para detecção de canais disponíveis para o uso de um US: a utilização de WSDBs.

Figura 2.3 – US não enxerga o UP e interfere na sua comunicação



Fonte: O Autor

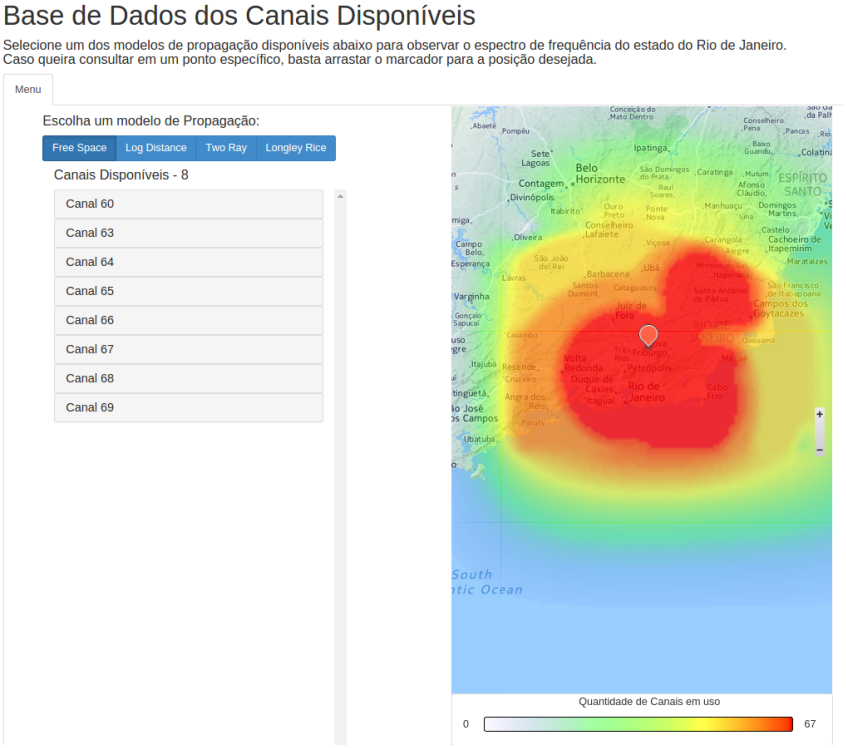
2.2.2 Bases de Dados de *White-Spaces*

WSDBs são bases de dados que possuem informações espectrais dos usuários de uma determinada região e podem ser acessadas pela *Web*. Sua principal contribuição para a solução do problema de conhecimento espectral, que antes dependia de técnicas complexas de sensoriamento, é o conhecimento global do espectro de sua região de operação. Com base nesse conhecimento, a WSDB pode estimar com precisão as faixas do espectro ociosas nessa região sem sofrer com o problema de nós escondidos. Os órgãos reguladores, citados na Seção 2.1, concluíram que o acesso espectral de USs seria economicamente viável apenas com uma abordagem centralizada de conhecimento espectral para a faixa do espectro destinada a TV (KOUFOS, 2013). Nesse cenário, USs podem fornecer uma estimativa de sua localização geográfica a uma WSDB. A WSDB, por sua vez, deve estimar os canais livres na região do US e responder às solicitações desse US por espectro ocioso.

O maior desafio na implementação de WSDBs é a modelagem da propagação dos sinais de UPs e USs para que não haja interferência. Essa modelagem varia de país para país, dadas as exigências de cada órgão regulador. *SenseLess* (MURTY et al., 2012), por exemplo, é um serviço que contém uma WSDB, a qual prevê a existência de *white-spaces* na sua região de operação e fornece um *framework* para gerenciar os dispositivo conectados à rede. Por ser um serviço subordinado ao conjunto de regras da FCC, utiliza o modelo de propagação bastante complexo, chamado de *Irregular Terrain Model* (ITM), que considera efeitos climáticos, de condutividade do solo e refração da superfície da Terra. Outro exemplo, é a WSDB Qosmo (UFRJ, 2014) que mapeia a utilização espectral do Rio de Janeiro, representada na Figura 2.4. Já que o Brasil ainda não possui uma regulamentação para uso de *white-spaces* por USs, essa WSDB implementa alguns modelos de propagação, apenas para visualização dos canais vagos seguindo cada um desses modelos.

Além dos exemplos citados acima, pela regulamentação do acesso de USs a *white-spaces* pelos órgãos reguladores citados anteriormente, diversas outras WSDB têm surgido. Grandes empresas como Google (GOOGLE, 2013) e Microsoft (MICROSOFT, 2014) já possuem WSDB para acesso de USs. Dada essa diversidade de soluções, o IETF padronizou um protocolo para comunicação entre USs e WSDBs (IETF, 2015). O protocolo PAWS, enfoque desse trabalho de conclusão de curso, define as mensagens que US e WSDB devem trocar para que o US obtenha uma lista de canais vagos em sua região de operação. A Seção 2.3 apresenta as características desse protocolo.

Figura 2.4 – Serviço *Web* da WSDB Qosmo



Fonte: (UFRJ, 2014)

2.3 *Protocol To Access White-Space Databases*

Essa seção apresenta as principais características do protocolo PAWS (IETF, 2015), enfoque maior desse trabalho de conclusão de curso. PAWS foi criado pelo grupo de trabalho paws WG, do IETF, com o objetivo de padronizar a comunicação entre USs e WSDBs. No modelo *Open Systems Interconnection* OSI, esse protocolo se localiza na camada de aplicação e suas mensagens se baseiam em objetos *JavaScript Object Notation* (JSON) trocados sobre requisições e respostas *Hyper Text Transfer Protocol Secure* (HTTPS). O tratamento de métodos POST é obrigatório, sendo opcional o tratamento de métodos GET pela WSDB. Existem três dispositivos que implementam o protocolo: o *Master Device*, o *Slave Device* e a WSDB. Tanto *Master Device* quanto *Slave Device* são dispositivos que fazem requisições por espectro vago em suas regiões, mas apenas o *Master Device* possui acesso à Internet e se comunica com a WSDB. O *Slave Device* é um dispositivo subordinado ao *Master Device* e que, quando necessita de novos canais para comunicação, solicita que o *Master Device* se comunique com a WSDB por ele. Já, a WSDB é um dispositivo que possui as característica apresentadas na Subseção 2.2.2.

O protocolo é orientado aos conjuntos de regras cujo *Master Device* e *Slave Device* são subordinados, cabe às WSDB implementar tais conjuntos de regras para poder atender às

requisições desses dispositivos. Cada conjunto de regras define parâmetros próprios exigidos pelo órgão regulador daquela região. Além disso, algumas funcionalidades, e suas respectivas mensagens, são consideradas opcionais no protocolo e se farão necessárias dependendo do conjunto de regras. A Subseção 2.3.1 apresenta as funcionalidades, obrigatórias e opcionais, do protocolo e seu fluxo de operação completo. Já, a Subseção 2.3.2 apresenta a formatação das mensagens bem como os parâmetros de cada uma.

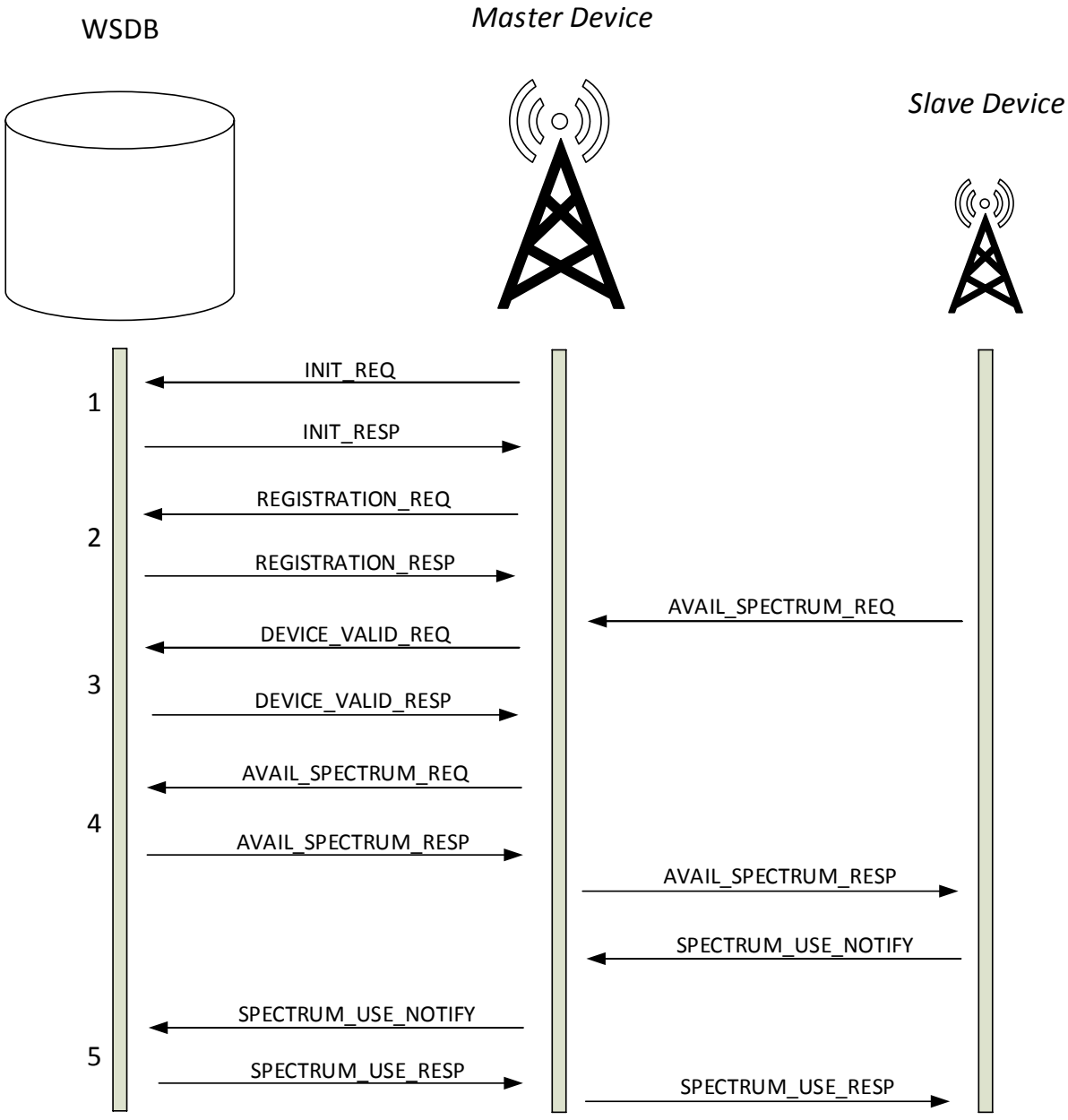
2.3.1 Funcionalidades e fluxo de operação

O grupo paws WG definiu seis funcionalidades para o protocolo PAWS: descobrimento da WSDB, inicialização, registro, validação, requisição de espectro e notificação de uso de espectro. Dentre essas funcionalidades, apenas descobrimento da WSDB, inicialização e requisição de espectro são obrigatórias, ou seja, estarão presentes na operação dos dispositivos independentemente do conjunto de regras que esses dispositivos sejam subordinados. As demais funcionalidades, registro, validação e notificação de uso de espectro, são opcionais e estarão presentes na operação dos dispositivos, apenas caso o conjunto de regras os exija. Abaixo enumeram-se as funcionalidades do protocolo e relaciona-se cada funcionalidade às mensagens que as implementam. Ainda, a Figura 2.5 apresenta o fluxo de mensagens da operação completa dos dispositivos que implementam o protocolo PAWS. As funcionalidades estão enumeradas nessa Figura como: 1 - Inicialização, 2 - Registro, 3 - Validação, 4 - Requisição de espectro, 5 - Notificação de uso de espectro. Descobrimento de WSDB não está identificado pois essa funcionalidade não gera mensagens.

- **Descobrimento da WSDB –** Ao iniciar o seu funcionamento, o *Master Device* deve descobrir uma WSDB que possua os dados de sua região de operação. O *Master Device* pode ser pré-configurado com uma ou mais *Uniform Resource Identifier* (URI) de WSDBs ou com um URI de um serviço que possua listas de URIs de WSDBs.
- **Inicialização –** Após descobrir uma WSDB, o *Master Device* deve utilizar o procedimento de inicialização. Esse procedimento tem por objetivo a troca de informações sobre as capacidades dos dispositivos, como por exemplo, o conjunto de regras que implementam. As mensagens de inicialização são:

  INIT_REQ - mensagem do *Master Device* para a WSDB, contendo a descrição do dispositivo e sua localização.

Figura 2.5 – Operação completa dos dispositivos utilizando o protocolo PAWS



Fonte: O Autor

INIT_RESP - mensagem da base de dados para o *Master Device*, contendo uma lista de regras de operação.

- **Registro –** Alguns conjuntos de regras podem exigir que o *Master Device* se registre à WSDB. Nesse caso, o *Master Device* deve realizar o procedimento de registro. Por exemplo, no caso do conjunto de regras estabelecidas pela FCC, o *Master Device* deve registrar a sua empresa proprietária, seu identificador, sua localização e a altura de sua antena. As mensagens de registro são:

  REGISTRATION_REQ - mensagem do *Master Device* para a base de dados, contendo a descrição do dispositivo, sua localização, sua empresa proprietária e características da antena, dependendo do conjunto de regras utilizado.

  REGISTRATION_RESP - mensagem da base de dados para o *Master Device*, contendo uma lista de regras de operação.

- **Validação –** Como o *Slave Device* não possui conexão à Internet e depende do *Master Device* para se comunicar com a WSDB, o *Master Device* pode requerer à WSDB a permissão de operação do *Slave Device*. A validação dos *Slave Device* depende de características desses dispositivos, como, por exemplo, o cojunto de regras que estão subordinados. Se o dispositivo é validado, o *Master Device* pode realizar a requisição de espectro para o *Slave Device*. As mensagens de validação são:

  DEV_VALID_REQ - mensagem do *Master Device* para a base de dados contendo a sua descrição e uma lista de descrições de *Slave Devices*.

  DEV_VALID_RESP - mensagem da base de dados para o *Master Device*, contendo a lista *Slave Devices* e suas respectivas validações ou invalidações.

- **Requisição de espectro –** Principal funcionalidade do protocolo, tem por objetivo possibilitar o *Master Device* e o *Slave Device* a adquirirem as informações espectrais da base de dados. As mensagens de requisição de espectro são:

  AVAIL_SPECTRUM_REQ - mensagem do *Master Device* para a base de dados, contendo a descrição do dispositivo, sua localização, sua empresa proprietária e características da antena, e outras capacidades do dispositivo. Se o *Master Device* está enviando à base de dados um AVAIL-SPECTRUM-REQ de um *Slave Device*, a mensagem também deve conter a descrição do *Master Device*.

  AVAIL_SPECTRUM_RESP - mensagem da WSDB para o *Master Device*, contendo a descrição do dispositivo que fez a requisição e uma lista de especificações de espectro. Informações como o canal e o tempo que esse canal será disponibilizado ao dispositivo fazem parte da especificação de espectro.

AVAIL_SPECTRUM_BATCH_REQ - mensagem do *Master Device* que funciona como múltiplos AVAIL_SPECTRUM_REQ antecipando novas regiões de operação do dispositivo. Nesse caso, há uma lista de localidades que o *Master Device* deseja requisitar espectro.

AVAIL_SPECTRUM_BATCH_RESP - mensagem da WSDB para o *Master Device* contendo a lista de localidades que o *Master Device* requisitou espectro, cada localidade contendo uma lista de especificações de espectro disponíveis.

- **Notificação de uso de espectro –** Caso a entidade regulamentadora exija, o *Master Device* deve notificar a base de dados o espectro que será utilizado pelo próprio *Master Device* ou pelo *Slave Device*, no caso do AVAIL_SPECTRUM_REQ ter sido feito em nome do *Slave Device*. As mensagens de notificação são:

  SPECTRUM_USE_NOTIFY - mensagem do *Master Device* para a base de dados, contendo a descrição do espectro que será utilizado e do dispositivo, bem como sua localização. Se essa notificação for feita por um *Slave Device*, é necessária também a descrição do *Master Device*.

  SPECTRUM_USE_RESP - mensagem de *acknowledgement* da base de dados para o *Master Device*.

2.3.2 Formatação das mensagens e parâmetros

O protocolo PAWS baseia-se em objetos JSON sobre mensagens de requisições e respostas HTTPS. Assim, todos os parâmetros do protocolo são objetos JSON. Existem três formatações para as mensagens do protocolo: uma formatação para mensagens do *Master Device* e duas para mensagens da WSDB. Essa diferenciação nas mensagens da WSDB se dá pela possível existência de erros nas mensagens do *Master Device*, ou seja, existe uma formatação para a operação normal do protocolo e outra para informar ao *Master Device* um erro na sua mensagem.

As mensagens de requisição do *Master Device* são formadas por quatro objetos JSON: 'jsonrpc', 'method', 'params' e 'id'. O primeiro objeto é a versão do formato JSON vigente, no caso 2.0. O segundo objeto é o nome do método que especifica qual mensagem o *Master Device* está enviando. A Tabela 2.1 apresenta a correspondência entre o nome do método com a mensagem definida no protocolo. 'Params' é a mensagem propriamente dita, onde cada parâmetro é especificado. Por fim, 'id' é o número de identificação da mensagem, que deve ser uma *string* suficientemente única.

Tabela 2.1 – Correspondência dos campos 'method' com as mensagens do *Master Device*

Método	Mensagem
spectrum.paws.init	INIT_REQ
spectrum.paws.register	REGISTRATION_REQ
spectrum.paws.getSpectrum	AVAIL_SPECTRUM_REQ
spectrum.paws.getSpectrumBatch	AVAIL_SPECTRUM_BATCH_REQ
spectrum.paws.notifySpectrumUse	SPECTRUM_USE_NOTIFY
spectrum.paws.verifyDevice	DEV_VALID_REQ

Fonte: O Autor

As mensagens de resposta da WSDB são formadas por três objetos JSON. Quando a mensagem do *Master Device* não possui erros, esses objetos são: 'jsonrpc', 'result' e 'id'. Sendo 'jsonrpc' o mesmo objeto explicado acima e 'id' uma cópia do 'id' da mensagem do *Master Device*. Já, o objeto 'result' possui os parâmetros de resposta da mensagem WSDB. Os parâmetros das mensagens de operação sem erros do *Master Device* e da WSDB são apresentados abaixo. Além disso, na Tabela 2.2, é apresentada a correspondência entre cada mensagem e seus respectivos parâmetros.

- **deviceDesc –** parâmetro que descreve o dispositivo. Os parâmetros que formam o 'deviceDesc' são todos opcionais e dependem do conjunto de regras que o dispositivo é subordinado. Por exemplo, no conjunto de regras da FCC os parâmetros que formam o 'deviceDesc' são: 'serialNumber', 'fccId' e 'fccTvbdDeviceType'. 'SerialNumber' é o número serial do dispositivo do fabricante. 'fccId' é o identificador do certificado do dispositivo da FCC. 'fccTvbdDeviceType' é o tipo de acesso a *white-spaces* do dispositivo, definido pela FCC.
- **location –** parâmetro que representa a localidade do dispositivo. Pode ser representado de duas formas: uma ponto ou um polígono de uma região. Caso o dispositivo seja representado por um ponto, ele deve conter suas informações de longitude e latitude e, caso o conjunto de regras exija, representar uma elipse de incerteza em torno desse ponto, contendo também os eixos dessa elipse. Já, se o dispositivo for representador por um polígono, uma lista de informações de latitude e longitude formam esse polígono.
- **rulesetInfos –** identifica o conjunto de regras que o dispositivo está subordinado. Contém uma *string* de duas letras identificando o país do órgão regulador, por exemplo US para os EUA, e o identificador do órgão regulador.

- **deviceOwner –** contém informações sobre os proprietários do dispositivo. É formado por vCards, definidos na RFC 6350 (IETF, 2011).
- **timestamp –** *timestamp* das mensagens de resposta da WSDB no formato YYYY-MM-DDThh:mm:ssZ, definido na RFC 3339 (IETF, 2002).
- **spectrumSpecs –** encapsula o 'rulesetInfo' do dispositivo e uma lista 'spectras' e 'eventTimes'. 'spectra' define o espectro vago que o dispositivo pode utilizar e 'eventTime' identifica o horário que o dispositivo pode iniciar sua comunicação e o horário que ele deve terminar sua comunicação naquela faixa do espectro.
- **geoSpectrumSpecs –** encapsula uma lista de 'location' e, para cada 'location', uma lista de 'spectrumSpecs'.
- **deviceDescs –** lista de 'deviceDesc'.
- **deviceValidites –** identifica a validade de um dispositivo. Possui o 'deviceDesc' do dispositivo que deve ser validado e um valor booleano, chamado 'isValid' que identifica se aquele dispositivo é válido ou não.

No caso da mensagem do *Master Device* possuir erros, ao invés do objeto 'result' na resposta da WSDB, tem-se o objeto 'error'. O objeto 'error' é formado pelos campos 'code', 'message' e 'data'. O campo 'code' é um valor inteiro negativo, que identifica o código de erro. O campo 'message' possui uma mensagem de erro qualquer que a WSDB deseje enviar, esse campo é opcional. Por fim, o campo 'data' contém parâmetros adicionais que necessitem ser comunicadas ao *Master Device* sobre o erro. A Tabela 2.3 relaciona os códigos de erro, seus nomes, suas descrições e seus parâmetros adicionais.

Tabela 2.2 – Correspondência das mensagens da operação normal do protocolo com seus parâmetros

Mensagem	Parâmetros
INIT_REQ	deviceDesc location
INIT_RESP	rulesetInfos
REGISTRATION_REQ	deviceDesc location deviceOwner
REGISTRATION_RESP	rulesetInfos
AVAIL_SPECTRUM_REQ	deviceDesc location
AVAIL_SPECTRUM_RESP	timestamp deviceDesc spectrumSpecs
AVAIL_SPECTRUM_BATCH_REQ	deviceDesc locations
AVAIL_SPECTRUM_BATCH_RESP	timestamp deviceDesc geoSpectrumSpecs
SPECTRUM_USE_NOTIFY	deviceDesc location spectra
SPECTRUM_USE_RESP	-
DEV_VALID_REQ	deviceDescs
DEV_VALID_RESP	deviceValidities

Fonte: O Autor

2.4 Redes IEEE 802.15.4

Essa seção apresenta as características dos dispositivos utilizados para implementar o caso de uso proposto nesse trabalho. A norma IEEE 802.15.4 (IEEE, 2011b) rege dispositivos de baixas taxas de dados em *Wireless Personal Area Networks* (WPANs). Baseada em comunicação de curtas distâncias, utilizam faixas de frequência livres de usuários licenciados para operar. Dispositivos 802.15.4 tem por objetivo ser de fácil instalação, transferir dados de forma confiável, ter baixo custo e possibilitar um tempo de vida de bateria prolongada, mantendo um protocolo de controle simples e flexível. Por essa características, são utilizados na implementação redes de sensores (SHON et al., 2009) e cenários de *Internet of Things* (SAJJAD; YOUSAF, 2014).

Em geral, compartilham a faixa de frequência de 2.4 GHz com dispositivos *Wireless Local Area Networks* (WLANs), porém não requerem uma infraestrutura preestabelecida para ope-

Tabela 2.3 – Correspondência dos objetos 'code', o nome do erro identificado e sua descrição

Códigos de erro	Nome	Descrição e parâmetros adicionais
-101	VERSION	A WSDB não suporta a versão do protocolo especificada.
-102	UNSUPPORTED	A WSDB não suporta dispositivo por algum dos parâmetros informados, por exemplo, seu conjunto de regras informado.
-103	UNIMPLEMENTED	A WSDB não implementa uma mensagem ou parâmetro opcional.
-104	OUTSIDE_COVERAGE	A localidade do dispositivo está fora da área de cobertura da WSDB.
-105	DATABASE_CHANGE	A WSDB mudou seu URI.
-201	MISSING	Falta algum parâmetro na requisição. A WSDB pode responder com a lista de parâmetros que faltam.
-202	INVALID_VALUE	O valor de algum parâmetro na requisição é inválido.
-301	UNAUTHORIZED	O dispositivo não é autorizado a utilizar a WSDB.
-302	NOT_REGISTERED	O conjunto de regras do dispositivo exige o registre desse dispositivo, que não foi realizado.

Fonte: O Autor

rar, ao contrário dos dispositivos WLAN. São bem mais simples e menos "agressivos" no acesso ao meio que os dispositivos WLAN, por isso, podem sofrer com interferência caso compartilhem o mesmo canal com dispositivos Wi-Fi. Nos cenários onde dispositivos IEEE 802.15.4 sofrem com interferência, o gasto de energia desses dispositivos pode crescer (NARAMPANAWE et al., 2010), diminuindo o tempo de vida das baterias dos dispositivos de uma rede de sensores, por exemplo.

Nesses cenários, onde o consumo de bateria é um ponto crítico na implementação da rede, pode-se agregar conhecimento espectral ao dispositivo para que deixem de sofrer com interferência. A Seção 3.2 do próximo capítulo apresenta trabalhos da literatura que agregam conhecimento espectral a redes IEEE 802.15.4. Porém, não foram encontrados trabalhos na

literatura que utilizem WSDBs para esse fim. Vista a necessidade de conhecimento espectral em redes IEEE 802.15.4, esses dispositivos foram propostos, no Capítulo 4, como caso de uso para exemplificar a atuação do PAWS.

3 TRABALHOS RELACIONADOS

O interesse de órgãos reguladores em DSA e, mais recentemente, no uso de WSDBs como habilitadoras da política de acesso ao espectro têm movido a academia a uma extensiva pesquisa nessa área. Sendo o enfoque desse trabalho de conclusão de curso o uso do PAWS, a Seção 3.1 apresenta trabalhos da literatura referentes ao protocolo. Já, a Seção 3.2 apresenta os trabalhos da literatura que utilizam conhecimento espectral em redes IEEE 802.15.4, visto que esse é o tipo de rede escolhido como caso de uso para utilização do PAWS nesse trabalho de conclusão de curso.

3.1 Uso do *Protocol to Access White-Space databases*

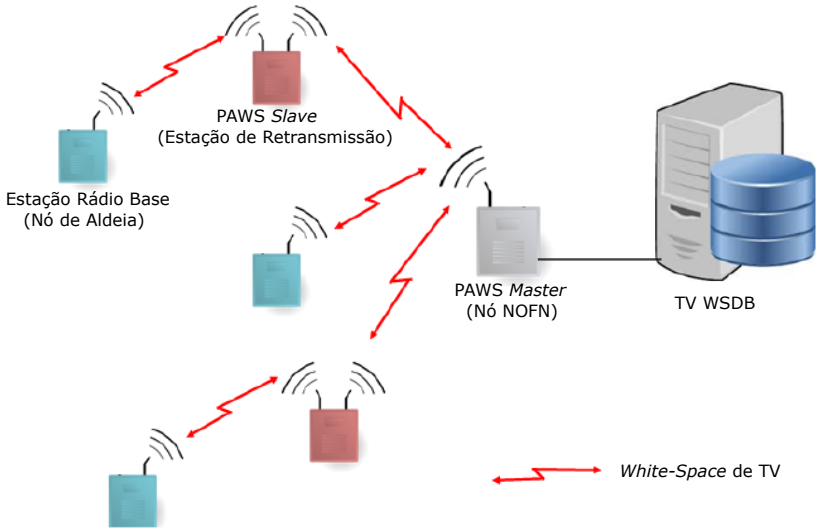
Por ser um protocolo novo, padronizado em 2015, existem poucos projetos que implementam o protocolo em suas WSDBs. Em um primeiro momento, soluções proprietárias surgiram em maior número implementando o protocolo. A Google, por exemplo, disponibiliza uma API aberta para uso experimental do PAWS para sua WSDB (GOOGLE, 2013). Entretanto, a solução da Google implementa apenas as mensagens de INIT_RESP e AVAIL_SPECTRUM_RESP, o que a torna incompleta. Outras soluções proprietárias, como a WSDB da empresa iconectiv (ICONECTIV, 2013) são fechadas e necessitam de cadastro com a empresa para serem utilizadas.

Na literatura foi encontrada apenas um trabalho que utiliza o protocolo como forma de acesso a WSDBs. OpenPAWS (GHOSH et al., 2015) é um projeto que desenvolveu e liberou para uso a primeira WSDB da Índia e foi publicado após o início do desenvolvimento desse trabalho de conclusão de curso. Essa WSDB se baseia nas informações recebidas de Doordashan, órgão governamental responsável pela radiodifusão de TV no país, que disponibilizou os parâmetros operacionais de todas as torres que utilizam a faixa de 470 - 590 MHz. Implementada em MySQL, lista todos os canais vagos na região da Índia, solicitada pelo *Master Deivce*. Sua interface HTTP opera sobre um servidor *Web* Apache2 e espera por requisições do *Master Device* utilizando um *script* PHP.

Já o *Master Device* e o *Slave Device* foram implementados usando o OpenWrt, um sistema operacional embarcado baseado em Linux. Ambos os dispositivos possuem duas interfaces de rede. O *Master Device*, para se conectar à Internet e poder acessar a WSDB, possui uma interface cabeada baseada na estrutura de fibra óptica da Índia. A segunda interface do dispositivo é sem fio e utiliza os canais informados pela WSDB para sua operação. O *Slave*

Device possui duas interfaces sem fio: uma delas para se comunicar com o *Master Device* e outra para servir clientes em lugares da Índia, onde a infraestrutura de fibra óptica não possui cobertura. A Figura 3.1 demonstra a arquitetura apresentada no artigo.

Figura 3.1 – Arquitetura do OpenPAWS



Fonte: (GHOSH et al., 2015)

OpenPAWS é um trabalho que pode servir de exemplo para os institutos de pesquisa do Brasil. Mesmo a Índia não possuindo uma regulamentação para o uso oportunístico do espectro por USs, o grupo desenvolvedor desse trabalho apresentou resultados laboratoriais que possibilitam o acesso de clientes à Internet, onde a estrutura de fibra óptica do país não possui cobertura. Além disso, utilizaram o protocolo recém padronizado para atender os requisitos futuros de dispositivos que desejam acessar WSDBs.

3.2 Conhecimento espectral em redes IEEE 802.15.4

Dispositivos IEEE 802.15.4, podem sofrer com interferência de outros dispositivos que operam na mesma faixa de espectro. Interferência pode acarretar em atrasos na comunicação desses dispositivos, fazendo com que gastem mais energia e, por conseguinte, diminuindo o tempo de vida da bateria desses dispositivos. Essa seção apresenta trabalhos da literatura que agregam conhecimento espectral a dispositivos IEEE 802.15.4 para lidar com problemas de interferência.

Na faixa de operação de 2.4 GHz é natural a coexistência entre dispositivos IEEE 802.15.4 e dispositivos IEEE 802.11. Pelas características agressivas de acesso ao espectro dos dispositivos IEEE 802.11, os dispositivos IEEE 802.15.4 podem sofrer com grandes atrasos em sua comunicação (TORABI; BHATE; LEUNG, 2013). Nesses cenários, os dispositivos IEEE 802.15.4 podem realizar sensoriamento espectral para selecionar canais livres para sua operação e evitar interferência. Narampanawe (NARAMPANAWE et al., 2010) e Torabi (TORABI; BHATE; LEUNG, 2013), por exemplo, propõem técnicas de sensoriamento espectral para redes 802.15.4. Ainda, Stabellini (STABELLINI, 2010) e Maleki (MALEKI; PANDHARIPANDE; LEUS, 2011) vão além e verificam a redução do gasto de energia de seus dispositivos ao utilizar sensoriamento espectral.

Apesar dos resultados obtidos, apenas o trabalho de Maleki se qualifica como um cenário real de rede de sensores baseada em IEEE 802.15.4. Esse trabalho utiliza rádios ZigBee para sua comunicação e realização do sensoriamento espectral, enquanto outros utilizam rádios poderosos como USRPs ou se baseiam em simulações. Nenhum trabalho, porém, utiliza WSDBs como possibilitador de conhecimento espectral em dispositivos IEEE 802.15.4. Isso se deve ao fato de não existir uma WSDB que possua informações dessa faixa do espectro (NARAMPANAWE et al., 2010).

Em contraste com os trabalhos da literatura, nesse trabalho de conclusão de curso foi utilizado um cenário real de comunicação entre dispositivos IEEE 802.15.4 que acessam uma WSDB como técnica de conhecimento espectral. O uso de WSDBs nesses cenários foi escolhido para exemplificar a utilização do PAWS, o Capítulo 4 apresenta o sistema criado para esse caso de uso. Como um possível trabalho futuro, apresentado no Capítulo 5, pode-se verificar a viabilidade do uso de WSDBs nesses cenários.

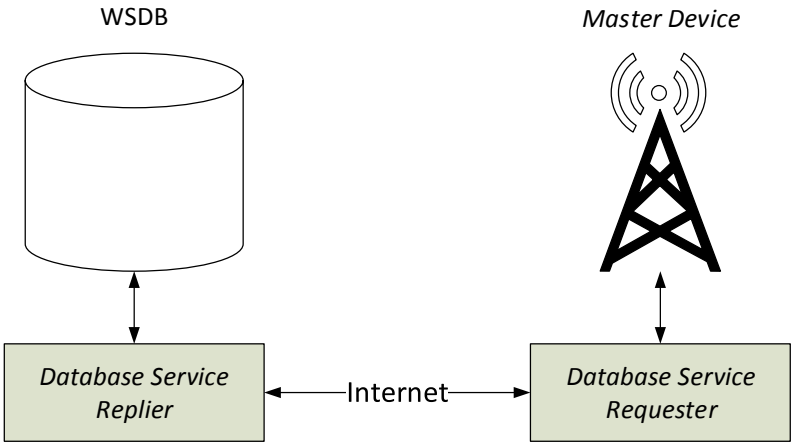
4 IMPLEMENTAÇÃO E CASO DE USO

Dado o exposto nos Capítulos anteriores, se observa a falta de implementações do PAWS na literatura. Em geral, estudos sobre o uso de WSDBs ainda não utilizam o protocolo como forma de comunicação com os dispositivos sem fio. Assim, esse trabalho de conclusão de curso foca a implementação do PAWS feita pelo autor, baseada no conjunto de regras da FCC. Este Capítulo apresenta essa proposta, bem como um caso de uso onde o protocolo pode ser inserido. Divido em duas Seções, o Capítulo está disposto da seguinte forma. A Seção 4.1 apresenta a implementação do PAWS feita pelo autor, expondo sua arquitetura, linguagem de programação utilizada, métodos que podem ser chamados pelos dispositivos e fluxogramas de operação desses métodos. Já, a Seção 4.2 apresenta o sistema criado para exemplificar o uso do PAWS, tomando como caso de uso uma rede IEEE 802.15.4. Por fim, a Seção 4.3 apresenta resultados da operação do sistema baseado nessa rede IEEE 802.15.4.

4.1 Implementação do *Protocol to Access White-Space databases*

A implementação do PAWS realizada nesse trabalho consiste em dois módulos: o *Database Service Requester* e o *Database Service Replier*. O *Database Service Replier*, módulo agregado às WSDB para que essas respondam ao *Master Device*, é detalhado na Subseção 4.1.1. Já o *Database Service Requester* tem o objetivo de ser agregado aos *Master Devices* que desejam solicitar espectro ocioso de uma WSDB, sendo detalhado na Subseção 4.1.2. Foram projetados da forma mais transparente possível para funcionar como caixas pretas, com o objetivo de eliminar a necessidade de implementação do PAWS pelas empresas que prestam esses serviços. A linguagem de programação escolhida para a implementação desses módulos foi a linguagem Python pela sua facilidade em tratar objetos JSON. A Figura 4.1 representa a arquitetura do sistema completo de comunicação entre *Master Device* e WSDB, com os módulos agregados a seus respectivos dispositivos.

Figura 4.1 – Arquitetura de um sistema de comunicação entre *Master Device* e WSDB



Fonte: O Autor

4.1.1 *Database Service Replier*

O módulo *Database Service Replier* simplesmente responde as mensagens do *Master Device*. Assim, esse módulo opera em função da mensagem recebida. Baseado em um servidor HTTPS, analisa e responde as mensagens POST recebidas. Para montar os objetos JSON de resposta às mensagens do *Master Device*, acessa a WSDB onde está agregado para obter dados como, por exemplo, o espectro disponível na região informada pelo *Master Device* ou para registrar os dados informados por esse dispositivo.

Na implementação desse módulo foram utilizadas cinco bibliotecas do Python: 'json', 'BaseHTTPServer', 'ssl', 'SocketServer' e 'datetime'. A biblioteca 'json' codifica e decodifica objetos JSON. 'BaseHTTPServer' cria um servidor HTTP e 'ssl' aplica o padrão *Security Socket Layer* sobre o servidor HTTP, transformando-o num servidor HTTPS. Da biblioteca 'SocketServer' se utiliza a classe 'ThreadingMixIn', para que a manipulação das mensagens recebidas pelo servidor funcione através de *threads*, ou seja, cada mensagem HTTPS recebida pelo servidor gera uma *thread* nova. Por fim, a biblioteca 'datetime' tem o objetivo de padronizar os *timestamps* no formato da RFC 3339 (IETF, 2002).

Com base nessas cinco bibliotecas e nas mensagens do PAWS, foram implementados seis métodos: cinco métodos relacionados com as mensagens do protocolo e um para manipular as mensagens POST recebidas do *Master Device*, chamado de 'do_POST'. Toda vez que uma mensagem POST é recebida pelo servidor, a biblioteca 'BaseHTTPServer' invoca o método 'do_POST', que decodifica o objeto JSON presente na mensagem. Com base no campo 'method' do objeto JSON, o método 'do_POST' invoca um dos cinco métodos relacionados

com as mensagens do PAWS. Esses outros métodos analisam a mensagem do protocolo e retornam uma mensagem apropriada, que é enviada pelo método 'do_POST' para o *Master Device*. A Tabela 4.1 relaciona o campo 'method' da mensagem recebida com o método a ser invocado pelo 'do_POST' e a mensagem de resposta da WSDB retornada pelo método definida no PAWS.

Tabela 4.1 – Correspondência dos campos 'method' com métodos invocados pelo método do_POST

Método	Mensagem
spectrum.paws.init	init_resp
spectrum.paws.register	reg_resp
spectrum.paws.getSpectrum	av_spec_resp
spectrum.paws.notifySpectrumUse	spec_use_not_ack
spectrum.paws.verifyDevice	dev_valid_resp

Fonte: O Autor

Dentre os métodos relacionados com as mensagens do PAWS, 'reg_resp', 'dev_valid_re sp' e 'av_spec_resp', correspondentes às funcionalidades de registro, validação e requisição de espectro, acessam a WSDB. Pelos constantes acessos do módulo à WSDB, a implementação desse módulo provou-se ser de difícil disjunção do restantes do sistema, onde esse módulo deve ser agregado.

4.1.2 *Database Service Requester*

O módulo *Database Service Requester*, por sua vez, se baseia em uma classe criada, chamada 'databaseServiceRequester'. Um objeto dessa classe pode ser instanciado pelo *Master Device* e operar da forma como lhe convier. Os parâmetros de instanciação de um objeto 'databaseServiceRequester' são: a localização do *Master Device*, o nome do operador, seu e-mail, organização que trabalha, ID da FCC e número serial do dispositivo. Para sua implementação, foram utilizadas apenas três bibliotecas: 'json', 'requests' e 'random'. Como no *Database Service Replier*, a biblioteca 'json' serve para codificar e decodificar mensagens JSON. 'requests' é utilizado para realizar requisições POST, sem o estabelecimento de conexão, e 'random' que foi utilizada para gerar números randômicos para o 'id' das mensagens do PAWS.

Baseados nessas bibliotecas, foram implementados treze métodos: cinco que fazem as requisições à WSDB, cinco que analisam as repostas da WSDB e três que servem de interface com o *Master Device*. Os dez métodos relacionados à troca de mensagens com a WSDB são

privados da classe, ou seja, apenas os três métodos de interface com o *Master Device* podem ser invocados por esse dispositivo. Esses três métodos são: 'init', 'request' e 'notify'. O método 'init' realiza as funcionalidades de inicialização e registro do PAWS e não possui parâmetros, seus valores de retorno são 'verdadeiro', em caso da operação ter sido efetuada com sucesso, ou 'falso', caso tenha ocorrido uma falha.

O método 'request' realiza a funcionalidade de requisição de espectro e, dependendo do modo de operação, pode realizar a funcionalidade de validação. Esse método recebe como parâmetros o modo de operação, que pode significar uma requisição de espectro para o *Master Device* ou para um *Slave Device*. Quando o modo de operação referencia a requisição de espectro para um *Slave Device* deve ser passado os parâmetros da mensagem DEV_VALID_REQ, especificados na Subseção 2.3.2. Seus valores de retorno são: uma lista de canais, caso a operação tenha terminado com sucesso, ou 'falso' caso tenha ocorrido uma falha. O método 'notify' realiza a funcionalidade de notificação de uso do espectro pelo dispositivo e recebe como parâmetro um objeto 'spectra', também definido na Subseção 2.3.2. Seus valores de retorno são os mesmo do método 'init'.

Tabela 4.2 – Correspondência entre os métodos de interface, métodos invocados e os objetos JSON

Método de interface	Métodos Invocados	Objetos Montados/Analisadas
init	__init_req	INIT_REQ
	__init_resp	INIT_RESP
	__reg_req	REGISTRATION_REQ
	__reg_resp	REGISTRATION_RESP
request	__dev_valid_req	DEV_VALID_REQ
	__dev_valid_resp	DEV_VALID_RESP
	__av_spec_req	AVAIL_SPECTRUM_REQ
	__av_spec_resp	AVAIL_SPECTRUM_RESP
notify	__spec_use_not	SPECTRUM_USE_NOTIFY
	__spec_use_not_ack	SPECTRUM_USE_RESP

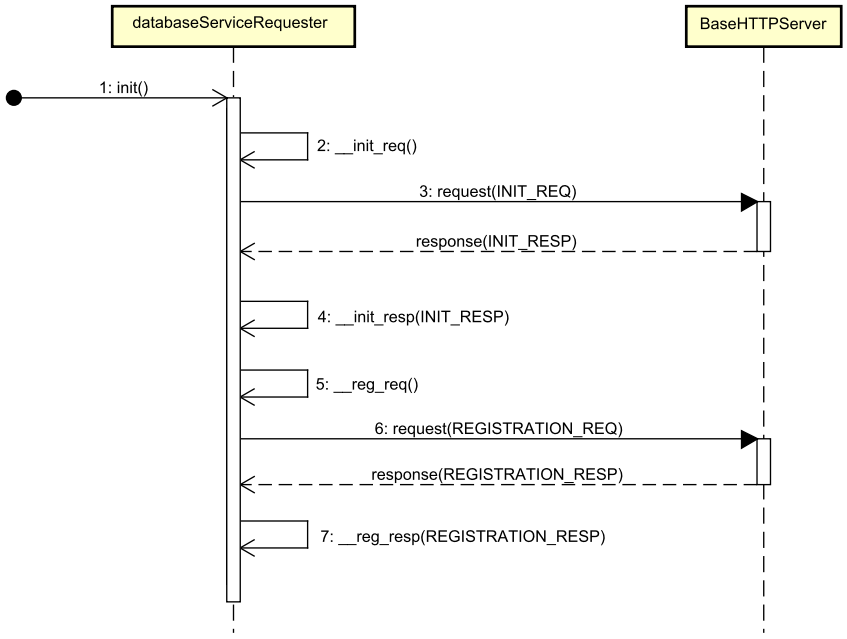
Fonte: O Autor

A Tabela 4.2 relaciona os três métodos que servem de interface entre o *Master Device* e os métodos relacionados com as mensagens do PAWS, que são invocados por cada um desses métodos. Os métodos relacionados com as mensagens do PAWS, que terminam em 'req', montam os objetos JSON das mensagens que serão enviadas à WSDB. Já os métodos que terminam em 'resp' analisam as mensagens de reposta da WSDB recebidas. Caso existam erros, esses métodos terminados em 'resp' retornam 'falso' e imprimem uma mensagem de erro para

o usuário. Na operação correta do PAWS, os métodos retornam ’verdadeiro’ e o método da próxima mensagem a ser enviada é invocado.

A Figura 4.2 apresenta a operação do método ’init’. Quando chamado pelo *Master Device*, realiza o ’__init_req’, método sem parâmetros que retorna a mensagem INIT_REQ para o método ’init’, que envia a mensagem INIT_REQ para a WSDB. Ao receber a mensagem INIT_RESP da WSDB, invoca o método ’__init_resp’, passando como parâmetro a mensagem recebida. Esse método analisa a mensagem e retorna ’verdadeiro’ ou ’falso’ dado que a operação foi correta ou aconteceu algum erro. A seguir, o método ’init’invoca o método ’__reg_req’ que retorna a mensagem de REGISTRATION_REQ. O método ’init’, então, envia a mensagem REGISTRATION_REQ para WSDB e espera pela mensagem REGISTRATION_RESP. Ao receber a resposta, invoca o método ’__reg_resp’, que analisa a mensagem REGISTRATION_RESP que retorna ’verdadeiro’ ou ’falso’ como o método ’__init_resp’. Caso a operação completa tenha ocorrido com sucesso, o método ’init’retorna ’verdadeiro’. Caso algum dos métodos ’resp’ tenha retornado ’falso’, o método ’init’retorna ’falso’ para o usuário.

Figura 4.2 – Diagrama de sequência do método ’init’

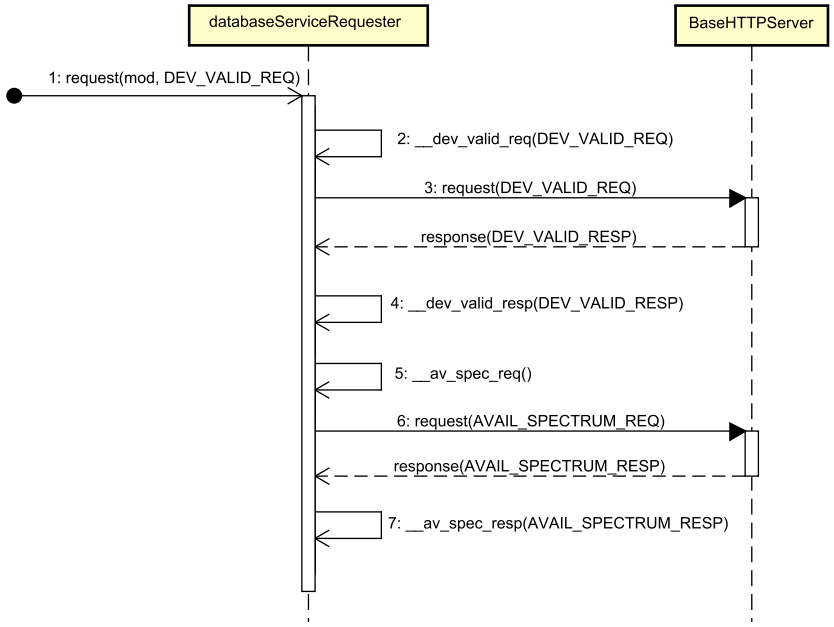


Fonte: O Autor

Já a Figura 4.3 representa a operação completa do método ’request’. Ao ser invocado pelo usuário, deve ter como parâmetro a variável ’mod’. Essa variável indica se a operação do método deve ser em função do *Master Device* ou de um *Slave Device*. Caso a variável ’mod’ indique a operação do método para um *Master Device*, o método ’request’invoca apenas o método ’__av_spec_req’, que retorna a mensagem AVAIL_SPECTRUM_REQ e envia essa mensagem

para a WSDB. Ao receber a mensagem de resposta, AVAIL_SPECTRUM_RESP, invoca o método '__av_spec_resp' para analisar a mensagem. Se a operação tiver sido concluída com sucesso, o método retorna os canais informados pela WSDB para o usuário. Caso tenha ocorrido um erro, o método retorna 'falso'. Quando a variável 'mod' indica a operação do método para um *Slave Device*, o método espera também os dados da mensagem DEV_VALID_REQ. Nesse cenário, o método 'request'invoca o método '__dev_valid_req', passando como parâmetro os dados recebidos do usuário. Esse método então retorna a mensagem DEV_VALID_REQ, que é enviada pelo método 'request'para a WSDB. Ao receber a mensagem de resposta da WSDB, invoca o método '__dev_valid_resp', que analisa a mensagem e retorna 'verdadeiro' ou 'falso' dado que a operação foi correta ou aconteceu algum erro. A seguir, o método opera como descrito acima, com os métodos '__av_spec_req' e '__av_spec_resp'.

Figura 4.3 – Diagrama de sequência do método 'request'

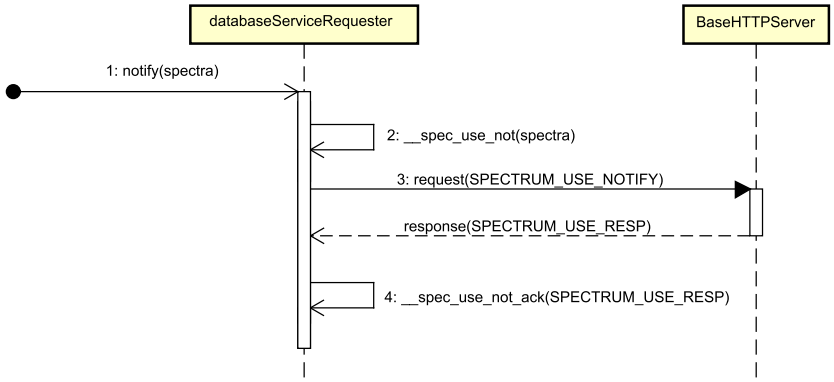


Fonte: O Autor

Por fim, a Figura 4.4 representa a operação do método 'notify'. Esse método recebe como parâmetro o canal escolhido pelo *Master Device* ou pelo *Slave Device*. O método 'notify'invoca o método '__spec_use_not', que recebe como parâmetro o canal escolhido pelo usuário. Esse método retorna a mensagem SPECTRUM_USE_NOTIFY que é enviada pelo método 'notify'para WSDB. Após receber a reposta SPECTRUM_USE_RESP da WSDB, o método '__spec_use_not_ack' é invocado com essa mensagem como parâmetro. Caso a operação tenha ocorrido com sucesso, esse método retorna 'verdadeiro' e o método 'notify', então,

também retorna 'verdadeiro'. Caso tenha ocorrido um erro os métodos '__spec_use_not_ack' e 'notify'retornam 'falso'.

Figura 4.4 – Diagrama de sequência do método 'notify'



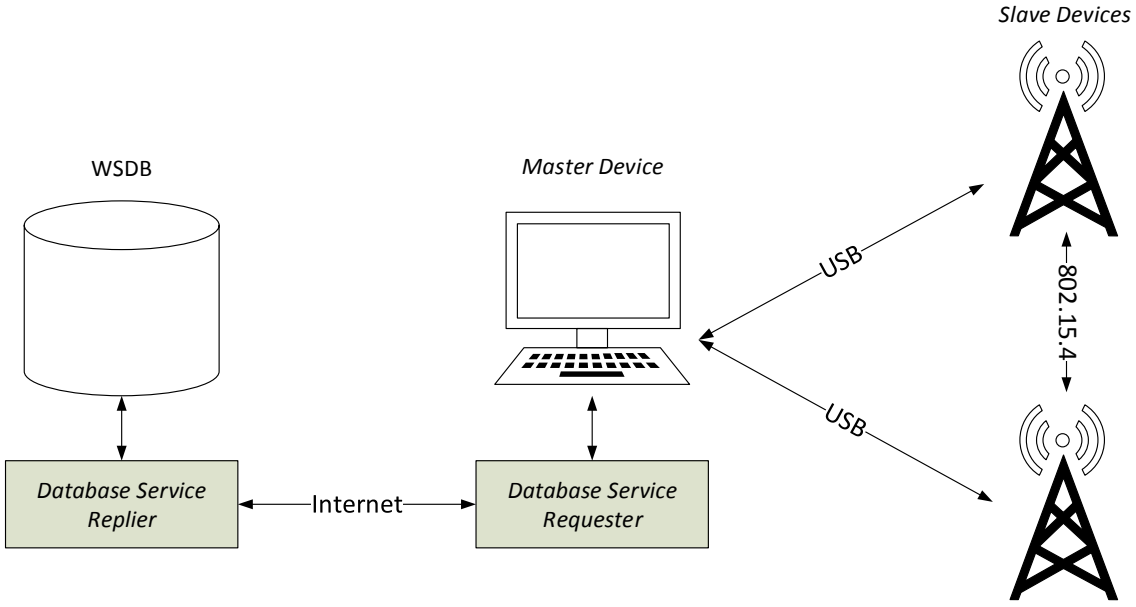
Fonte: O Autor

Ao contrário do *Database Service Replier*, a implementação desse módulo é disjunta do *Master Device*, funcionando como uma caixa preta para esse sistema. Assim, esse módulo cumpre a proposta do trabalho, podendo ser agregado a qualquer rádio que deseja obter informações espectrais de WSDBs.

4.2 Uso do PAWS em redes IEEE 802.15.4

Para validar a operação dos módulos expostos acima, foi implementada uma rede IEEE 802.15.4. Esse tipo de rede foi escolhida pelo baixo custo de seus dispositivos. Assim, dois rádios MRF24J40 (MICROSHIP, 2010) agregados a dois Alevinos (CIRCUITAR, 2012), uma implementação brasileiro do Arduino, se comunicam. Esses rádios são *Slave Devices* para o PAWS e devem solicitar ao *Master Device* que adquira adquira espectro vago da WSDB em nome deles. O *Master Device*, por sua vez, é uma aplicação Python implementado em um PC. A comunicação entre os rádios e o *Master Device* se dá através de uma das portas USB do PC. O *Master Device*, por sua vez, se comunica com a WSDB por meio do módulo *Database Service Requester*. Após adquirir tais informações da WSDB, o *Master Device* informa os canais adquiridos para os rádios, que começam a operar em um dos canais informados. A arquitetura completa do sistema é apresentada na Figura 4.5.

Figura 4.5 – Arquitetura do caso de uso implementado



Fonte: O Autor

Para a implementação dos *Slave Devices* foram utilizadas duas bibliotecas: 'ArduinoJson' (BLANCHON, 2014) e 'Mrf24j-arduino-library' (PALSSON, 2011). Para montar os objetos JSON da mensagem DEV_VALID_REQ, que são enviadas para o *Master Device*, foi utilizada a biblioteca 'ArduinoJson'. Já a biblioteca 'Mrf24j-arduino-library' tem o objetivo de comunicar os Alevinos e os rádios MRF24J40, para que esses rádios possam ser configurados e realizar sua comunicação sem fio. Ao serem ligados, os Alevinos montam o objeto JSON da mensagem DEV_VALID_REQ e o enviam para o *Master Device*. A seguir, esperam que o *Master Device* repasse a reposta AVAIL_SPECTRUM_RESP da WSDB a eles. Ao receber a mensagem AVAIL_SPECTRUM_RESP, escolhem um dos canais, na implementação é selecionado sempre o primeiro canal da lista recebida, e se reconfiguram para operar em tal canal. Por fim, os *Slave Devices* enviam a mensagem de SPECTRUM_USE_NOTIFY com o canal escolhido para o *Master Device* e esperam receber SPECTRUM_USE_RESP.

O *Master Device* foi implementado em Python, instanciando um objeto da classe 'databaseServiceRequester', exposta na Subseção 4.1.2, e utilizando a biblioteca 'serial', que ouve a porta USB do PC. O *Master Device* inicia sua operação invocando o método 'init' da objeto 'databaseServiceRequester' instanciado. A seguir, ouve a porta USB do PC, esperando receber a mensagem DEV_VALID_REQ dos rádios. Quando essa mensagem é recebida, invoca o método 'request' do objeto, usando como parâmetros o modo de operação para *Slave Devices* e a mensagem DEV_VALID_REQ. O *Master Device* envia aos *Slave Device* a lista de canais recebidas da WSDB pela interface USB do PC e volta a ouvi-la. Quando o *Slave Device*

enviar a mensagem de SPECTRUM_USE_NOTIFY, o *Master Device* deve invocar o método 'notify' do objeto com o canal utilizado pelos rádios. Por fim, com o retorno 'verdadeiro' do método 'notify', deve se enviar o SPECTRUM_USE_RESP para os *Slave Devices*.

O último dispositivo do sistema é a WSDB. Implementada usando MySQL (MySQL AB, 1995), possui duas tabelas: 'regists' e 'spectrum'. A tabela 'regists' contém as informações dos *Master Devices* registrados na WSDB e a tabela 'spectrum' possui o espectro vago de sua região de operação. O módulo *Database Service Replier* faz os acessos de escrita e leitura das tabelas. Para validar o sistema, manteve-se um dos rádios operando estaticamente em um canal. O segundo rádio, então, adquiria esse canal da WSDB e se reconfigurava. Ao observar que os rádios iniciavam sua comunicação, pôde-se concluir que o sistema está validado.

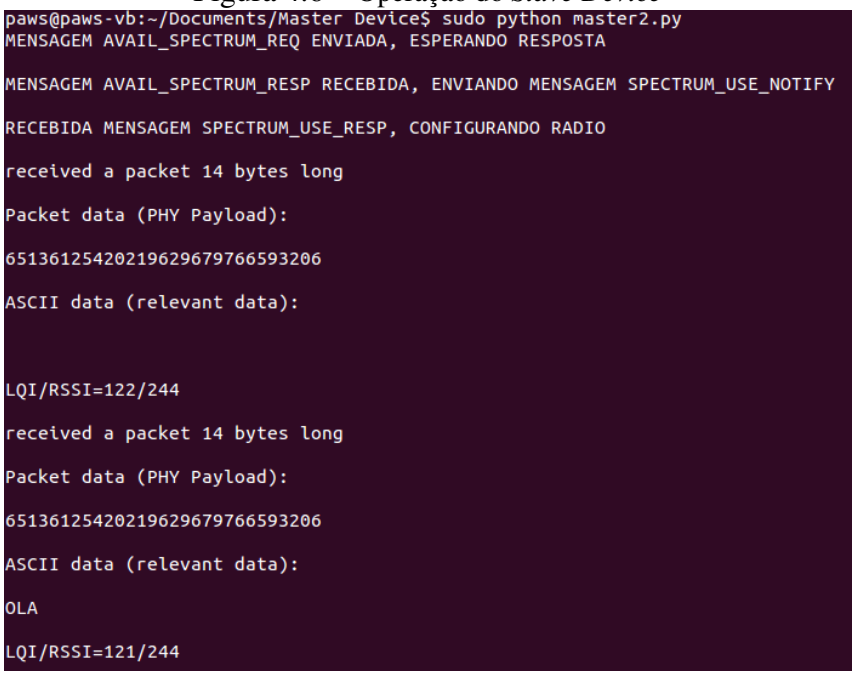
4.3 Resultados

Essa seção apresenta alguns resultados observados na operação da rede IEEE 802.15.4 implementada como caso de uso do protocolo PAWS. A Figura 4.6 representa a operação do *Slave Device* se comunicando com o *Master Device*. Ao ser iniciado, o rádio solicita que o *Master Device* adquira espectro vago da WSDB. Quando o *Master Device* responde com a lista de canais vagos, o rádio escolhe um canal e responde ao *Master Device* com a mensagem de SPECTRUM_USE_NOTIFY, indicando seu canal escolhido para comunicação. O *Slave Device* aguarda a resposta com a confirmação da notificação de uso de espectro pelo *Master Device* e passa a operar no canal escolhido. Ao se configurar para usar tal canal, o segundo rádio, que já operava estaticamente nesse canal, passa a se comunicar com o *Slave Device*.

A Figura 4.7 apresenta os acessos do *Master Device* na WSDB. À direita da figura observa-se a execução da aplicação do *Master Device*, inclusive com seu tempo de execução. À esquerda estão apresentadas as mensagens HTTPS que chegam à WSDB, representando a operação desse dispositivo. Quando o *Master Device* opera para si, são feitas quatro requisições HTTPS à WSDB, essas quatro requisições podem ser observadas na figura.

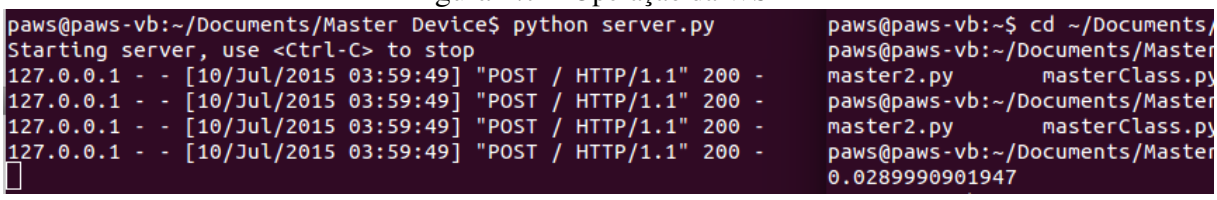
Ainda, foi feita uma comparação em diferentes modos de operação do sistema sobre duas variáveis críticas desse cenários: tempo de execução do *Master Device* e uso de memória dos Arduinos. O tempo de execução do *Master Device* foi comparado entre o modo de operação de requisição de espectro para si e para um *Slave Device*. Quando o *Master Device* opera para si, o tempo equivale ao tempo gasto na troca de mensagens com a WSDB. Como a WSDB e o

Figura 4.6 – Operação do *Slave Device*



Fonte: O Autor

Figura 4.7 – Operação da WSDB



Fonte: O Autor

Master Device foram implementados no mesmo PC, esse tempo foi bastante curto, em média 0.0289 segundos. Em contraste, quando o *Master Device* opera em nome de um *Slave Device*, o tempo de execução médio do *Master Device* sobe para 2.47 segundos. Isso se deve ao longo atraso na comunicação serial do Arduino.

A segunda variável, uso de memória dos Arduinos, foi comparada entre utilizar ou não a biblioteca 'ArduinoJson'. Quando o rádio utiliza apenas a biblioteca 'Mrf24j-arduino-library' seu uso de memória é de 4872 bytes. Quando se agrega a biblioteca 'ArduinoJson' o uso da memória sobe para 11456 bytes. Mesmo que a utilização de memória quase triplique ao utilizar a biblioteca 'ArduinoJson', essa biblioteca é necessária para que os Arduinos possam adquirir espectro vago em sua região.

5 CONCLUSÃO E TRABALHOS FUTUROS

O *Protocol to Access White-Space databases* padroniza a comunicação entre WSDBs e rádios que desejam adquirir canais ociosos em suas regiões de operação. Esse protocolo é importante para facilitar o desenvolvimento de rádios que acessam dinamicamente o espectro, tecnologia cada vez mais necessária para o progresso de redes sem fio. Por ser um protocolo novo, faltam implementações na literatura. Assim, nesse trabalho de conclusão de curso, o protocolo foi implementado. Com o intuito de auxiliar a indústria, para que as empresas que desejam acessar WSDBs não necessitem realizar implementações próprias, a implementação do PAWS foi feita por meio de dois módulos, que serviriam como caixas pretas para seus sistemas: o *Database Service Requester* e o *Database Service Replier*.

O *Database Service Requester* atingiu seu objetivo, se tornando um módulo transparente, podendo ser agregado a qualquer sistema. Porém, com o *Database Service Replier* não se obteve tal resultado, pelos constantes acessos feitos pelo módulo à WSDB. Assim, a contribuição desse trabalho de conclusão de curso é a implementação aberta do *Database Service Requester*, que pode ser utilizada pela indústria em suas soluções que desejam acessar dinamicamente o espectro.

Como trabalho futuro, se estenderá a implementação do *Database Service Requester* para aceitar outros conjuntos de regras, como o conjunto de regras da ECC. Outro trabalho futuro seria quantificar a diferença do consumo de energia ao se utilizar WSDB e sensoriamento espectral em redes IEEE 802.15.4. Verificando que o uso de WSDBs reduziria o consumo de energia dos dispositivos IEEE 802.15.4, pode-se propor o uso de WSDBs pessoais, como foi utilizada nesse trabalho, como padrão para o conhecimento espectral dos dispositivos desses cenários. Uma nova abordagem de WSDB deveria ser criada para viabilizar o acesso de dispositivos IEEE 802.15.4 a essas bases de dados. Hoje, o conceito de WSDB é uma base de dados que se baseia nas informações de rádios que cobrem grandes áreas em faixas licenciadas do espectro. Porém, diminuindo o escopo de atuação de uma WSDB para conter informações locais do espectro, por exemplo, de uma empresa, os dispositivos sem fio da empresa poderiam acessar a WSDB para adquirir canais vagos.

REFERÊNCIAS

AKYILDIZ, I. F. et al. Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. **Computer Networks**, Elsevier, v. 50, n. 13, p. 2127–2159, 2006.

BLANCHON, B. **ArduinoJson**. 2014. Disponível em <https://github.com/bblanchon/ArduinoJson>. Acessado em Abril de 2015.

CIRCUITAR. **Alevino**. 2012. Disponível em <https://www.circuitar.com.br/nanoshields/modulos/alevino/>. Acessado em Abril de 2015.

DZULKIFLI, M.; KAMARUDIN, M.; RAHMAN, T. Spectrum occupancy at uhf tv band for cognitive radio applications. In: **RF and Microwave Conference (RFM), 2011 IEEE International**. [S.l.: s.n.], 2011. p. 111–114.

GHOSH, S. et al. Openpaws: An open source paws and uhf tv white space database implementation for india. In: **Communications (NCC), 2015 Twenty First National Conference on**. [S.l.: s.n.], 2015. p. 1–6.

GOOGLE. **Spectrum Database**. Disponível em <https://www.google.com/get/spectrumdatabase/>. Acessado em Abril de 2015.: [s.n.], 2013.

GURNEY, D. et al. Geo-location database techniques for incumbent protection in the tv white space. In: **New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on**. [S.l.: s.n.], 2008. p. 1–9.

HAN, Y. et al. Spectrum occupancy measurement: Focus on the tv frequency. In: **Signal Processing Systems (ICSPS), 2010 2nd International Conference on**. [S.l.: s.n.], 2010. v. 2, p. V2–490–V2–494.

ICONECTIV. **iconectiv US TV White Space Database**. 2013. Disponível em <http://www.iconectiv.com/spectrum-mgmt/white-spaces/>. Acessado em Maio de 2015.

IEEE. **Enabling Broadband Wireless Access Using Cognitive Radio Technology and Spectrum Sharing in White Spaces Recipient of the IEEE SA Emerging Technology Award**. 2011. <https://standards.ieee.org/getieee802/download/802.22-2011.pdf>. Acessado: 02/06/2015.

IEEE. **Low-Rate Wireless Personal Area Networks (LR-WPANs)**. 2011. <https://standards.ieee.org/getieee802/download/802.15.4-2011.pdf>. Acessado: 02/06/2015.

IETF. **Date and Time on the Internet: Timestamps**. 2002. Disponível em <https://tools.ietf.org/html/rfc3339>. Acessado em Abril de 2015.

IETF. **vCard Format Specification**. 2011. Disponível em <https://tools.ietf.org/html/rfc6350>. Acessado em Abril de 2015.

IETF. **Protocol to Access White-Space databases**. 2015. Disponível em <https://datatracker.ietf.org/doc/rfc7545/>. Acessado em Maio de 2015.

IRNICH, T. et al. Spectrum sharing scenarios and resulting technical requirements for 5g systems. In: **Personal, Indoor and Mobile Radio Communications (PIMRC Workshops), 2013 IEEE 24th International Symposium on**. [S.l.: s.n.], 2013. p. 127–132.

KOUFOS, K. **Spectrum Access in White Spaces Using Spectrum Sensing and Geolocation Databases**. Thesis (PhD) — Aalto University, 2013.

LEAVES, P.; HUSCHKE, J.; TAFAZOLLI, R. A summary of dynamic spectrum allocation results from drive. In: **IST Mobile and Wireless Telecommunications**. [S.l.: s.n.], 2002. p. 245–250.

MALEKI, S.; PANDHARIPANDE, A.; LEUS, G. Energy-efficient distributed spectrum sensing for cognitive sensor networks. **Sensors Journal, IEEE**, v. 11, n. 3, p. 565–573, March 2011. ISSN 1530-437X.

MICROSHIP. **MRF24J40**. 2010. Disponível em <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en027752>. Acessado em Abril de 2015.

MICROSOFT. **Microsoft White Spaces**. 2014. Disponível em <http://whitespaces.cloudapp.net/>. Acessado em Maio de 2015.

MITOLA, J. **Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio**. Thesis (PhD) — Royal Institute of Technology, 2000.

MURTY, R. et al. Senseless: A database-driven white spaces network. **Mobile Computing, IEEE Transactions on**, v. 11, n. 2, p. 189–203, Feb 2012. ISSN 1536-1233.

MySQL AB. **MySQL database**. 1995. Disponível em <https://www.mysql.com/>. Acessado em Abril de 2015.

NARAMPANAWE, K. et al. Spectrum sensing in cognitive sensor networks. In: **Wireless And Optical Communications Networks (WOCN), 2010 Seventh International Conference On**. [S.l.: s.n.], 2010. p. 1–6.

PALSSON, K. **Mrf24j40-arduino-library**. 2011. Disponível em <https://github.com/karlp/Mrf24j40-arduino-library>. Acessado em Abril de 2015.

SAJJAD, S.; YOUSAF, M. Security analysis of ieee 802.15.4 mac in the context of internet of things (iot). In: **Information Assurance and Cyber Security (CIACS), 2014 Conference on**. [S.l.: s.n.], 2014. p. 9–14.

SHON, T. et al. Security architecture for ieee 802.15.4-based wireless sensor network. In: **Wireless Pervasive Computing, 2009. ISWPC 2009. 4th International Symposium on**. [S.l.: s.n.], 2009. p. 1–5.

STABELLINI, L. Energy-aware exploitation of white spaces in the time domain for wireless sensor networks. In: **Wireless Communication Systems (ISWCS), 2010 7th International Symposium on**. [S.l.: s.n.], 2010. p. 981–985. ISSN 2154-0217.

STAPLE, G.; WERBACH, K. The end of spectrum scarcity [spectrum allocation and utilization]. **Spectrum, IEEE**, v. 41, n. 3, p. 48–52, March 2004. ISSN 0018-9235.

TORABI, N.; BHATE, S.; LEUNG, V. C. Robust sensing strategy for dynamic spectrum access in the 2.4 ghz ism band. In: **Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on**. [S.l.: s.n.], 2013. p. 2713–2717. ISSN 2166-9570.

UFRJ. **Qosmo WSDB**. 2014. Disponível em <http://qosmo.dyndns.org:8182/home>. Acessado em Junho de 2015.

YUCEK, T.; ARSLAN, H. A survey of spectrum sensing algorithms for cognitive radio applications. **Communications Surveys Tutorials, IEEE**, v. 11, n. 1, p. 116–130, First 2009. ISSN 1553-877X.

ZHAO, Q.; SADLER, B. A survey of dynamic spectrum access. **Signal Processing Magazine, IEEE**, v. 24, n. 3, p. 79–89, May 2007. ISSN 1053-5888.

Estudo sobre o *Protocol to Access White Space databases*

Vinícius G. Schaurich[1],
Cristiano B. Both[1], **Juergen Rochol**[1]

[1]Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brasil

{vgschaurich, cbboth, juergen}@inf.ufrgs.br

Abstract. *Dynamic spectrum access techniques enable wireless communication devices to utilize channels that are vacant during certain periods of time. Among these techniques, we highlight the use of databases which stores information regarding the spectrum occupation in specific geographical areas. The Protocol to Access White Space databases has been developed to standardize the communication interface between the wireless device and the database. This paper proposes an implementation of this protocol in order to identify and mitigate failures, as well as to come up with possible extensions to the protocol.*

Resumo. *Técnicas de acesso dinâmico ao espectro tem sido aplicadas a dispositivos de comunicação sem fio para que esses dispositivos possam utilizar canais que estiverem vagos em determinados instantes de tempo. Dentre essas técnicas, destaca-se o uso de bases de dados que informam aos dispositivos sem fio as disponibilidades espectrais de suas regiões. Para padronizar a comunicação entre o dispositivo sem fio e a base de dados, o protocolo Protocol to Access White Space databases foi criado. Neste trabalho será proposta a implementação desse protocolo a fim de identificar e mitigar falhas e propôr possíveis extensões para o protocolo.*

1. Introdução

A faixa do espectro eletromagnético utilizada para radiocomunicação, que abrange as frequências de 9 kHz a 300 GHz, é chamada de espectro de radiofrequências [ANATEL 2014]. Essa faixa é, geralmente, gerenciada pelos órgãos governamentais reguladores de seus respectivos países. Tais órgãos são responsáveis pela concessão de porções do espectro para operadoras de telecomunicação. Essa concessão é estática, ou seja, apenas a operadora licenciada pode utilizar a porção alocada para si. Devido a essas estratégias de alocação adotadas pelos órgãos reguladores e ao aumento do uso de dispositivos sem fio, esse espectro tem se tornado um recurso cada vez mais escasso. Em contraste, estudos mostram uma grande subutilização desse recurso [Valenta *et al.* 2009], já que os usuários que adquirem certas porções do espectro as utilizam apenas por frações de tempo.

Para reduzir a subutilização do espectro de radiofrequências, políticas de acesso dinâmico ao espectro (*Dynamic Spectrum Access* - DSA) tem sido propostas [Zhao e Sadler 2007]. Nessa nova política, dispositivos de rede sem fio poderiam utilizar faixas do espectro que estejam ociosas (*white spaces*) de forma oportunista, com a restrição de não causar interferência aos usuários licenciados. No contexto de DSA, pelo usuário licenciado possuir prioridade na comunicação, ele comumente é chamando de usuário primário (UP) e, aquele usuário que deseja se comunicar durante a ociosidade de uma faixa do espectro, é chamado de usuário secundário (US). Para que um rádio transmita de forma

oportunista, sem interferir com um UP, tal US deve analisar a disponibilidade de faixas do espectro, através de uma técnica chamada de Sensoriamento Espectral (SE) [Yucek e Arslan 2009]. Em determinados cenários, como redes de sensores sem fio [Stabellini e Zander 2010], verificar a disponibilidade de faixas do espectro utilizando apenas SE é interessante. Entretanto, em muitos casos, provou-se que para garantir que o US não interfira com o UP essa técnica torna-se inadequada [Paisana *et al.* 2014].

Nos cenários onde somente SE é inadequado, os US podem se comunicar com bases de dados (*White Space Databases* - WSDB) que possuem informações espectrais da região onde esses US estejam operando. Com base nas informações recebidas da WSDB, os US podem escolher o canal vago que irão utilizar. Um grande número de soluções de WSDB vem sendo desenvolvidas com o objetivo de mapear a ocupação do espectro de determinadas regiões. Exemplos de sistemas criados para fazer esse mapeamento são a *Spectrum Database* [Google 2014] e a *TV White Space Database* [SpectrumBridge 2014]. Assim, dada a diversidade de WSDB que estão surgindo, foi criado um grupo de trabalho pela *Internet Engineering Task Force* (IETF) para padronizar um protocolo de comunicação entre os US e as WSDB. O *Protocol to Access White Space databases* (PAWS) [IETF 2014], define as mensagens que o US e a WSDB devem trocar para que o US obtenha uma lista de canais vagos em sua região de operação. Entretanto, por ser um protocolo novo, ainda não foram feitos estudos sobre desempenho, segurança, compatibilidade, entre outras características importantes que devem ser consideradas em um protocolo padrão.

Este trabalho de conclusão de curso propõe o estudo do protocolo PAWS através da implementação de dois módulos, um acoplado ao US e outro acoplado à WSDB, funcionado como caixas pretas para seus respectivos sistemas. O objetivo dessa implementação transparente é que empresas criadoras de US e WSDB não precisem implementar seus próprios módulos de comunicação PAWS, poderiam apenas agregar a implementação proposta nesse trabalho a seus sistemas. O módulo do US, por exemplo, receberá requisições espectrais de seu sistema e retornará a lista de canais informada pelo módulo da WSDB. Já o módulo da WSDB requisitará à WSDB uma lista de canais com os parâmetros recebidos do US. A partir desse estudo e da implementação desses dois módulos de comunicação poderá se verificar falhas ou possíveis melhorias no protocolo para uma melhor comunicação entre US e WSDB. Por fim, modificar os módulos de comunicação para atender as melhorias e mitigar as falhas identificadas.

O restante deste trabalho está estruturado da seguinte forma. A Seção 2 aprofunda conceitos envolvidos em DSA, discorrendo sobre problemas em SE e apresenta uma alternativa para esses problemas: o uso de bases de dados de *white spaces*. Na Seção 3 a proposta do trabalho é apresentada, especificado a arquitetura dos módulos do US e da WSDB. A forma com a qual se irá testar a comunicação entre os módulos para se validar, tanto a implementação como o protocolo, será descrita na Seção 4. Finalmente, a Seção 5 conclui esse trabalho dando o cronograma esperado para a segunda parte do trabalho de conclusão de curso.

2. Fundamentação Teórica

Esta seção tem o objetivo de aprofundar os conceitos de DSA e apresentar o protocolo PAWS. Na Subseção 2.1, são apontadas as principais formas de realizar DSA, focando em SE e WSDB. Na Subseção 2.2 é apresentado o fluxo de operação do protocolo PAWS, bem como as funcionalidades que foram definidas para atingir tal fluxo de operação e as

mensagens associadas a cada funcionalidade.

2.1. *Dynamic Spectrum Access*

O termo DSA remete a uma gama de abordagens que podem ser classificadas em três diferentes modelos: dinâmico de uso exclusivo, compartilhamento aberto e acesso hierárquico [Zhao e Sadler 2007]. O primeiro modelo propõe que UPs comercializem fatias temporais de seus canais para uso de US. Apesar desse compartilhamento, essa abordagem não minimiza a ociosidade dos *white spaces* já que a característica de uso exclusivo do espectro permanece. O modelo de compartilhamento aberto assume que o canal não é licenciado e os dispositivos operam de foram igual. Um exemplo de uso desse modelo são os produtos *Wi-Fi*. Já o terceiro modelo baseia-se numa hierarquia entre UP e US, onde o US transmite nos momentos que o UP não está utilizando sua porção, com a premissa que não irá interferir com o UP. Esse terceiro modelo é o mais amplamente difundido por favorecer um compartilhamento completo do espectro mesmo em canais licenciados e será o foco do trabalho.

Tomando o escopo dos modelos hierárquicos, SE surge como uma ferramenta para identificar se uma faixa do espectro está sendo utilizada pelo UP em determinado momento [Yucek e Arslan 2009]. Dentre as principais técnicas de SE, destacam-se detecção de energia e detecção de forma de onda. A primeira técnica, mais simples, mede a energia da faixa do espectro sensoriado para se tentar identificar a existência de alguma transmissão na região de operação do US. A segunda técnica, mais complexa, tenta encontrar padrões no sinal recebido que possuem alta correlação com padrões previamente configurados. Em ambos os casos, a complexidade do algoritmo é inversamente proporcional a sua acurácia, ou seja, necessita-se de um elevado poder computacional para garantir que o US identifique a transmissão de um UP e, consequentemente, não interfira na sua comunicação. Ainda assim, existem cenários onde, mesmo algoritmos mais complexos, como sensoriamento por forma de onda, não garantem a identificação da transmissão do UP. Por exemplo, o sinal de um UP degradado por ruído pode passar despercebido, mesmo por um algoritmo complexo de SE. Outro problema, ainda mais grave, é a existência de nós escondidos. Esse problema é apresentado na Figura 1, onde o US não percebe a existência de transmissão do UP e tenta se comunicar com o usuário final, causando interferência na comunicação do UP com o usuário final.

Devido à existência de cenários onde SE tem dificuldades em detectar um UP, novas abordagens foram propostas. Uma dessas abordagens é o US obter as informações espectrais de sua região através de consultas a bases de dados que contenham tais informações. O uso de WSDBs é preferível em cenários de características estáticas, onde um UP passará meses transmitindo, como na operação de transmissoras de TV digital [Gurney *et al.* 2008] para o descobrimento espectral do US. Dado que essas bases de dados tem o conhecimento integral do espectro de sua região de operação, esse tipo de solução não sofre com o problema de nós escondidos. Por suas vantagens, soluções de DSA baseadas em WSDB chamaram a atenção da comunidade e dos órgãos reguladores. Tanto a *Federal Communications Commission* (FCC) [FederalCommunicationsCommission 2014] quanto a *Electronic Communications Committee* (ECC) [ElectronicCommunicationsCommittee 2014] criaram regras para que as WSDBs pudessem estimar de forma precisa os canais em que os USs poderiam operar [Denkovska *et al.* 2011].

Dado o interesse de órgãos como o FCC e o ECC em DSA baseado em bases de dados de *white spaces*, empresas como a Google [Google 2014] e a Spectrum Bridge [Spec-

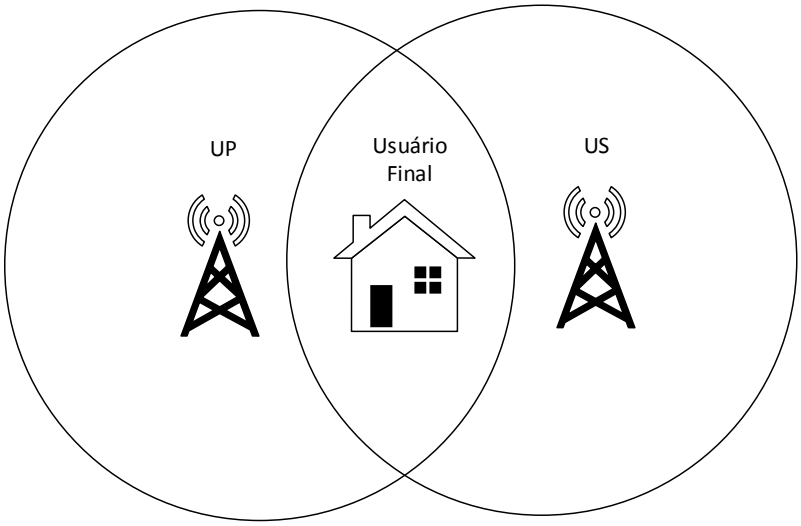


Figura 1. US não enxerga o UP e interfere na sua comunicação

trumBridge 2014] vêm desenvolvendo suas soluções em WSDB. Devido o crescente número dessas soluções, é necessário que haja uma padronização na comunicação entre US e WSDB. Com esse objetivo, o IETF criou o protocolo PAWS, apresentado na subseção 2.2.

2.2. *Protocol to Access White Space databases*

PAWS é um protocolo criado pelo grupo de trabalho *paws WG*, do IETF, com o objetivo de padronizar a comunicação entre USs e WSDBs. Nesta subseção serão apresentados os pontos principais do protocolo, focando no seu fluxo de operação, suas funcionalidades e mensagens. Outras questões como segurança, codificação, tratamento de erros e detalhamento dos parâmetros da mensagens estão presentes no *draft* e serão abordados na parte dois do trabalho de graduação. O protocolo baseia-se em três dispositivos principais:

- WSDB - Dispositivo conectado à Internet que possui a informações espectrais. Recebe e responde as requisições do *Master Device*.
- *Master Device* - Dispositivo conectado à Internet que, pode funcionar como um dispositivo fim, ou como um intermediário entre o *Slave Device* e a base de dados. Realiza requisições à base de dados para si ou para o *Slave Device*.
- *Slave Device* - Dispositivo sem acesso à Internet que realiza requisições à base de dados por meio do *Master Device*.

Definidos esses três dispositivos, pode-se dar uma visão geral sobre o protocolo. A Figura 2 apresenta o fluxo completo de operação do protocolo. Inicialmente o *Master Device* descobre uma base de dados apropriada para a sua área de operação. Essa descoberta se dá através da obtenção de um identificador uniforme de recursos (*Uniform Resource Identifier* - URI). Com esse identificador, o dispositivo estabelece uma sessão *HyperText Transfer Protocol Secure* (HTTPS) com a base de dados. Opcionalmente, o *Master Device* pode enviar uma mensagem de inicialização para a base da dados. Se a base de dados receber uma mensagem de inicialização, a WSDB deve responder essa mensagem. A base de dados pode, ainda, requerer o registro do *Master Device*.

Passada a fase de inicialização da conexão, o *Master Device* deve enviar uma mensagem de requisição de espectro para a base de dados. A requisição de espectro tanto

pode ser relacionada ao *Master Device* quanto ao *Slave Device* e, caso essa requisição seja relacionada ao segundo dispositivo, o *Master Device* pode verificar com a base de dados se o *Slave Device* é um dispositivo válido. A base de dados responde a requisição com uma mensagem de resposta de espectro. Opcionalmente, o *Master Device* pode notificar a base de dados com uma mensagem de uso de espectro e, nesse caso, a base de dados deve responder com uma mensagem de confirmação.

Dado esse fluxo de operação, os passos de registro, validação do *Slave Device* e notificação de uso de espectro são opcionais. A necessidade desses passos opcionais depende da entidade reguladora a quem o *Master Device* e o *Slave Device* estão subordinados. Para alcançar o fluxo de operação completo, o protocolo define seis funcionalidades. A lista de funcionalidades do protocolo, e a relação das mensagens que cumprem tais funcionalidades, está apresentada abaixo.

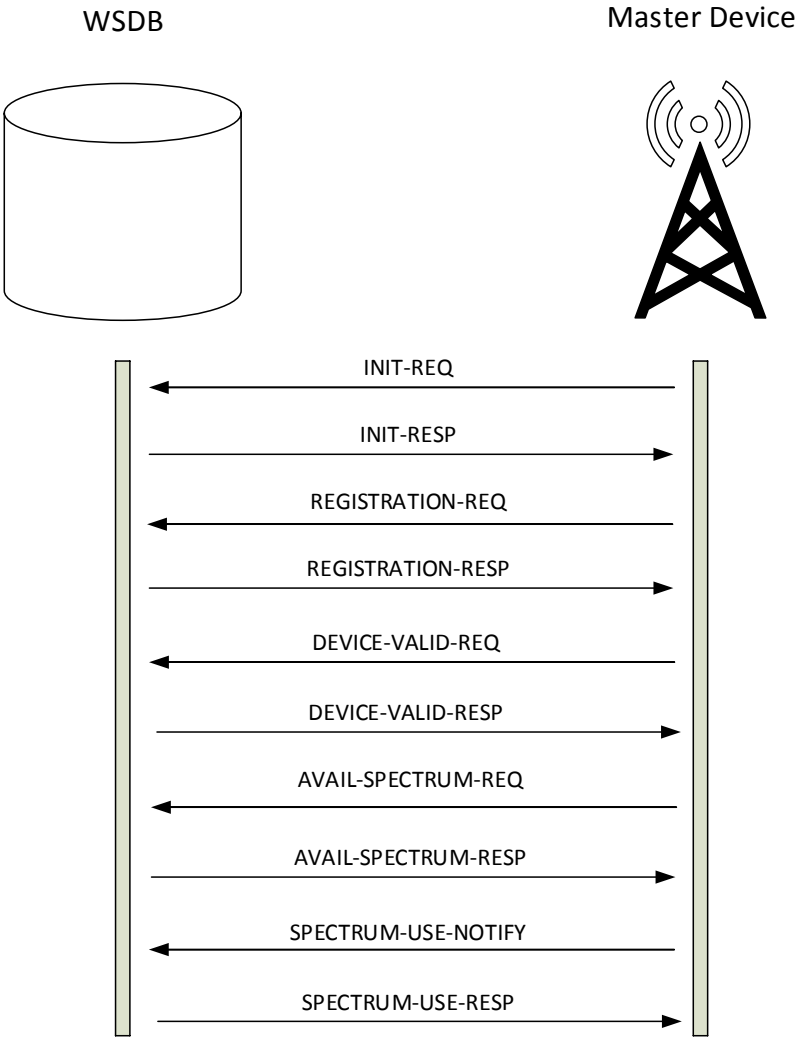


Figura 2. Troca completa de mensagens do protocolo PAWS entre WSDB e *Master Device*

- Descobrimento da base de dados - O *Master Device* pode ser pré-configurado com uma ou mais URI de bases de dados ou com uma URI de um servidor que possua listas de URIs de bases de dados.
- Inicialização - Para iniciar uma comunicação com a base de dados, o *Master Device* pode utilizar o procedimento de inicialização. Esse procedimento tem por

objetivo a troca de informações sobre as capacidades dos dispositivos, como por exemplo, o conjunto de regras que implementam (distâncias limite de operação e tempo de atualização do espectro de uso do *Master Device*).

INIT-REQ - Mensagem do *Master Device* para a base de dados, contendo a descrição do dispositivo e sua localização.

INIT-RESP - Mensagem da base de dados para o *Master Device*, contendo uma lista de regras de operação.

- Registro - Alguns conjuntos de regras podem requerer que o *Master Device* se registre à base de dados. Nesse caso, o *Master Device* deve realizar o procedimento de registro. Por exemplo, no caso do conjunto de regras estabelecidas pela FCC, o *Master Device* deve registrar a sua empresa proprietária, seu identificador, sua localização e a altura de sua antena.

  REGISTRATION-REQ - Mensagem do *Master Device* para a base de dados, contendo a descrição do dispositivo, sua localização, sua empresa proprietária e características da antena, dependendo do conjunto de regras utilizado.

  REGISTRATION-RESP - Mensagem da base de dados para o *Master Device*, contendo uma lista de regras de operação.
- Validação - Como o *Slave Device* não possui conexão à internet e depende do *Master Device* para se comunicar com a base de dados, o *Master Device* pode requerer à base de dados a permissão de operação do *Slave Device*. Se o dispositivo é validado, o *Master Device* pode realizar a requisição de espectro para o *Slave Device*.

  DEV-VALID-REQ - Mensagem do *Master Device* para a base de dados contendo a sua descrição e uma lista de descrições de *Slave Devices*.

  DEV-VALID-RESP - Mensagem da base de dados para o *Master Device*, contendo a lista *Slave Devices* e suas respectivas validações ou invalidações.
- Requisição de espectro - Principal funcionalidade do protocolo, tem por objetivo possibilitar o *Master Device* e o *Slave Device* adquiram as informações espectrais da base de dados.

  AVAIL-SPECTRUM-REQ - Mensagem do *Master Device* para a base de dados, contendo a descrição do dispositivo, sua localização, sua empresa proprietária e características da antena, e outras capacidades do dispositivo. Se o *Master Device* está enviando à base de dados um AVAIL-SPECTRUM-REQ de um *Slave Device*, a mensagem também deve conter a descrição do *Master Device*.

  AVAIL-SPECTRUM-RESP - Mensagem da base de dados para o *Master Device*, contendo a descrição do dispositivo que fez a requisição e uma lista de especificações de espectro. Informações como o canal e o tempo que esse canal será disponibilizado ao dispositivo fazem parte da especificação de espectro.
- Notificação de uso de espectro - Caso a entidade regulamentadora exija, o *Master Device* deve notificar a base de dados o espectro que será utilizado pelo próprio *Master Device* ou pelo *Slave Device*, no caso do AVAIL-SPECTRUM-REQ ter sido feito em nome do *Slave Device*.

  SPECTRUM-USE-NOTIFY - Mensagem do *Master Device* para a base de dados, contendo a descrição do espectro que será utilizado e do dispositivo, bem como sua localização. Se essa notificação for feita por um Slave Device, é necessária também a descrição do Master Device.

  SPECTRUM-USE-RESP - Mensagem de *acknowledgement* da base de dados para o *Master Device*.

Apresentadas as funcionalidades e o fluxo de operação definidos para o protocolo PAWS, já é possível modelar uma implementação e sugerir melhorias para o protocolo. As propostas de implementação e melhorias são feitas na Seção 3.

3. Proposta do trabalho

Esta seção irá detalhar a proposta do trabalho. Uma implementação do protocolo se faz necessária para encontrar falhas ou possíveis extensões. A Subseção 3.1 apresenta a arquitetura da implementação do protocolo. Já a Subseção 3.2 discute possíveis extensões para o protocolo que, mesmo sem a implementação, foram observadas.

3.1. Arquitetura

Para que a base de dados de *white spaces* e o *Master Device* possam se comunicar por meio do protocolo PAWS, ambos os dispositivos devem ser estendidos para suportar tal comunicação. Para tanto, esse trabalho propõe a implementação de dois módulos de comunicação, um para o *Master Device* e outro para a WSDB. A arquitetura da comunicação é apresentada na Figura 3. Esses módulos funcionarão como caixas pretas dentro de seus sistemas, podendo ser embarcados em qualquer solução. Por exemplo, uma empresa que desenvolve rádios baseados em DSA, poderá embarcar o módulo referente ao *Master Device*, sem necessitar desenvolver e validar um módulo próprio.

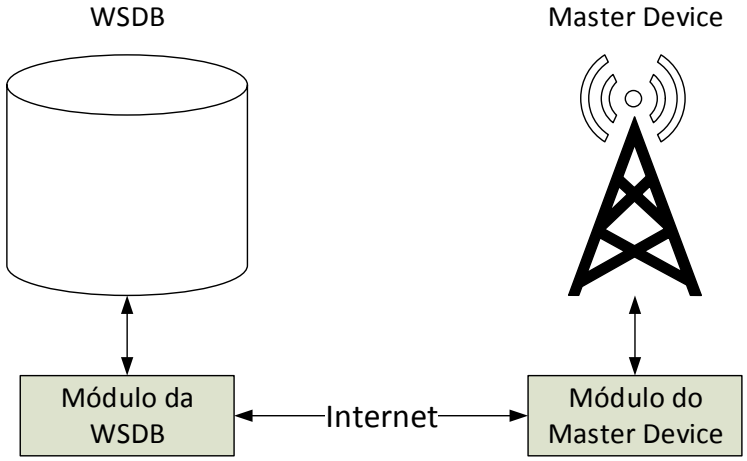


Figura 3. Arquitetura do sistema de comunicação proposto

Esses módulos funcionarão na forma de uma máquina de estados finita (*Final State Machine* - FSM). Cada um dos estados da FSM se referirá a uma funcionalidade do protocolo em que a transição entre os estados, dependerá da validade da mensagem recebida pelo módulo do outro dispositivo. Como dito na Subseção 2.2, o tratamento de erros na comunicação entre os módulos será abordado na segunda parte do trabalho de graduação, mas nesse momento, essa característica da arquitetura não será detalhada. Assim, a Figura 4 demonstra a FSM do *Master Device*, sem o tratamento de erros.

Dessa forma, como entrada de ambos os módulos, teremos as mensagens do protocolo. Ainda, outras entradas, particulares a cada módulo, também devem ser especificadas. O módulo da WSDB deve se comunicar com a WSDB para adquirir as informações espectrais da região do *Master Device*. Portanto, uma entrada particular ao módulo da WSDB é a lista de canais que será enviada ao *Master Device*. Já o módulo do *Master Device* necessita de entradas particulares para atender requisições do *Slave Device*. Essas

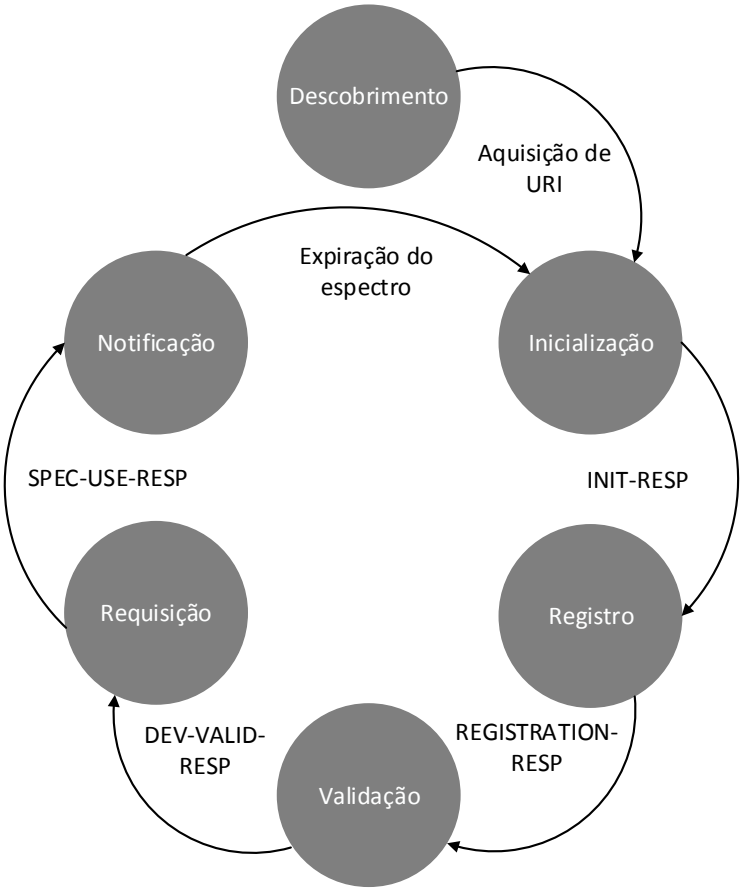


Figura 4. FSM do *Master Device*

entradas e as FSMs implementam o protocolo, porém, alguns cenários exigem extensões para o protocolo e, consequentemente, para a implementação. Possíveis extensões que já foram identificadas apenas com o estudo do protocolo serão discutidas na Subseção 3.2.

3.2. Extensões para o protocolo

Seguindo estritamente a definição do protocolo, o *Master Device* considera as informações da WSDB sempre como corretas. Então, caso a WSDB informe um canal para o *Master Device* que já esteja ocupado, o *Master Device* pode causar interferência com um UP operante na sua região. Dado esse cenário, uma possível solução seria estender o protocolo PAWS para comparar as informações adquiridas por meio da comunicação com a WSDB com as informações adquiridas localmente, por meio de SE. Ao validar as informações da base de dados, o *Master Device* segue normalmente a sua operação. Mas, verificada a inconsistência entre as informações de SE e da WSDB, alternativas como escolher outro canal da lista recebida ou realizar nova requisição à base de dados devem ser tomadas.

Outra extensão possível para o protocolo seria um tratamento diferenciado na fase de notificação. O *draft* trata a notificação de uso do espectro pelo *Master Device* com caráter apenas informativo para a WSDB. Poderia-se utilizar essa informação para atualizar as informações da WSDB para que essa contenha tanto as informações de uso de espectro de UPs como de USs. Dado que a base de dados conteria também as informações de uso espectral de USs, isso reduziria a chance de USs interferindo entre si. Adotada essa

proposta de extensão do protocolo, a funcionalidade de notificação se tornaria obrigatória.

4. Metodologia

Esta seção apresenta a metodologia que será utilizada na implementação do protocolo PAWS. Dada a proposta apresentada na Seção 3, surgem quatro componentes principais a serem implementados: a WSDB, o *Master Device* e os módulos de comunicação entre WSDB e *Master Device*.

Para implementar o *Master Device* será utilizado um *Universal Software Radio Peripheral* (USRP) [Ettus 2014]. O USRP é um dispositivo de rádio flexível, configurável através de um software *open-source* chamado GNURadio [GNURadio 2014]. Baseado em *scripts* Python, o GNURadio é uma ferramenta poderosa para soluções em rádio definido por software. O conjunto USRP e GNURadio realizará as funções do *Master Device*, como utilizar um dos canais informados pela WSDB e realizar o SE. Já a WSDB será simulada a partir de uma base de dados MySQL [MySQL 2014]. A implementação dos dois módulos se dará por meio de *scripts* Python e a comunicação entre eles será feita por *sockets* TCP.

Cada um dos componentes serão testados separadamente antes da arquitetura completa ser testada. O *Master Device* será testado utilizando um segundo USRP. Esse segundo USRP gerará tráfego em uma determinada frequência e o SE do *Master Device* deve ser capaz detectar tal transmissão. A base de dados MySQL será previamente populada com informações espectrais artificiais. Para testar a base de dados, serão requisitadas listas de canais em determinadas regiões e a WSDB deve ser capaz de retornar apenas as informações espectrais que abrangem essas regiões. Já o teste de ambos os módulos se dará de forma que, dados parâmetros de entrada que se encontrariam numa operação real do sistema, o módulo deve retornar uma saída condizente. Por exemplo, dando como entrada a mensagem REGISTRATION-REQ ao módulo da WSDB, a saída desse módulo deve ser uma mensagem do tipo REGISTRATION-RESP.

Validados todos os componentes de forma independente, se testará a arquitetura como um todo. Para tanto, se deixará previamente populada a base de dados com informações espectrais da região do USRP. Para verificar que o *Master Device* está recebendo a informação correta, o *Master Device* gerará tráfego em um dos canais informados como vagos pela base de dados. Um segundo USRP fará SE na região do *Master Device* para verificar em qual canal o *Master Device* estará transmitindo. Se o canal de transmissão do *Master Device* for o mesmo que um dos canais informados pela base de dados, a arquitetura estará validada.

5. Conclusões e Cronograma

O protocolo PAWS foi criado para padronizar a comunicação entre USs e WSDBs quando esses USs desejam acessar dinamicamente o espectro. Por esse protocolo ainda ser novo, faltam implementações para validar seu funcionamento. Essa primeira parte do trabalho de graduação apresentou uma proposta de implementação do protocolo para, estudando essa implementação, identificar melhorias e falhas do protocolo.

Acredita-se que o estudo de implementações como essa, proposta no trabalho, seja indispensável para a que indústria ganhe incentivos para investir na área de DSA. Enfim concluindo esse trabalho, apresenta-se o cronograma da segunda etapa do trabalho de graduação na tabela 1.

	Março	Abril	Maio	Junho	Julho
Implementação dos módulos	X				
Teste dos módulos	X	X			
Estudo de falhas e melhorias		X	X		
Implementação de extensões			X	X	
Teste dos módulos finais				X	X
Escrita da monografia	X	X	X	X	X

Tabela 1. Cronograma da segunda parte do trabalho de graduação

Referências

ANATEL (2014). Radiofrequência. Disponível em: <www.anatel.gov.br/Portal/>. Acesso em Setembro 2014.

Denkovska, M., Latkoski, P., e Gavrilovska, L. (2011). Geolocation database approach for secondary spectrum usage of tvws. Em *Telecommunications Forum (TELFOR), 2011 19th*, páginas 369–372.

ElectronicCommunicationsCommittee (2014). Disponível em: <www.cept.org/ecc>. Acesso em Novembro 2014.

Ettus (2014). Usrp. Disponível em: <www.ettus.com/>. Acesso em Setembro 2014.

FederalCommunicationsCommission (2014). Disponível em: <www.fcc.gov/>. Acesso em Novembro 2014.

GNURadio (2014). Disponível em: <gnuradio.org/>. Acesso em Setembro 2014.

Google (2014). Spectrum database. Disponível em: <www.google.com/get/spectrumdatabase/>. Acesso em Setembro 2014.

Gurney, D., Buchwald, G., Ecklund, L., Kuffner, S., e Grosspietsch, J. (2008). Geolocation database techniques for incumbent protection in the tv white space. Em *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, páginas 1–9.

IETF (2014). Protocol to acess white space database. Disponível em: <datatracker.ietf.org/doc/draft-ietf-paws-protocol/>. Acesso em Setembro 2014.

MySQL (2014). Disponível em: <www.mysql.com/>. Acesso em Setembro 2014.

Paisana, F., Marchetti, N., e DaSilva, L. (2014). Radar, tv and cellular bands: Which spectrum access techniques for which bands? *Communications Surveys Tutorials, IEEE*, 16(3):1193–1220.

SpectrumBridge (2014). Tv white space database. Disponível em: <http://whitespaces.spectrumbridge.com/Main.aspx>. Acesso em Setembro 2014.

Stabellini, L. e Zander, J. (2010). Energy-aware spectrum sensing in cognitive wireless sensor networks: A cross layer approach. Em *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, páginas 1–6.

Valenta, V., Fedra, Z., Marsálek, R., Baudoin, G., e Villegas, M. (2009). Towards cognitive radio networks: Spectrum utilization measurements in suburb environment. Em *Radio and Wireless Symposium, 2009. RWS '09. IEEE*, páginas 352–355.

Yucek, T. e Arslan, H. (2009). A survey of spectrum sensing algorithms for cognitive radio applications. *Communications Surveys Tutorials, IEEE*, 11(1):116–130.

Zhao, Q. e Sadler, B. (2007). A survey of dynamic spectrum access. *Signal Processing Magazine, IEEE*, 24(3):79–89.