PROTOCOLO DE DECISÃO DO ESPECTRO PARA REDES DE SENSORES SEM FIO COGNITIVAS

ERASMO EVANGELISTA DE OLIVEIRA

PROTOCOLO DE DECISÃO DO ESPECTRO PARA REDES DE SENSORES SEM FIO COGNITIVAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Luiz Henrique Andrade Correia
Coorientador: Antônio Alfredo Ferreira Loureiro.

Belo Horizonte
17 de junho de 2011

Oliveira, Erasmo Evangelista de

O48o Protocolo de Decisão do Espectro para Redes de Sensores Sem fio Cognitivas / Erasmo Evangelista de Oliveira. — Belo Horizonte, 2011
xxv, 133 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais.Departamento de Ciência da Computação.
Orientador: Luiz Henrique Andrade Correia
Coorientador: Antônio Alfredo Ferreira Loureiro.

1. Computação - Teses. 2. Redes de Computadores – Teses. 3. Redes de Sensores Sem Fio – Teses.
I. Orientador. II. Coorientador. III. Título.

CDU 519.6*22 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Protocolo de decisão do espectro para redes de sensores sem fio

ERASMO EVANGELISTA DE OLIVEIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. LUIZ HENRIQUE ANDRADE CORREIA - Orientador
Departamento de Ciência da Computação - UFLA

PROF. ANTONIO ALFREDO FERREIRA LOUREIRO - Co-orientador
Departamento de Ciência da Computação - UFMG

PROF. RICARDO AUGUSTO RABELO OLIVEIRA
Departamento de Ciência da Computação - UFOP

PROF. DANIEL FERNANDES MACEDO
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 17 de junho de 2011.

Dedico primeiramente ao Divino Pai Eterno, minha fonte inexorável de força e perseverança nos momentos difíceis; aos meus amados pais Joaquim e Irene que sempre me deram amor, carinho e incentivo; aos meus queridos irmãos Iron e Ronis pelo afeto e admimiração que guardam por mim; a minha amada esposa Elaine, companheira fiel das batalhas cotidianas; e aos meus estimados filhos João Pedro, Vinícius e Hugo que são a razão da minha vida.

Agradecimentos

Agradeço primeiramente ao Pai Eterno que me deu força, saúde e perseverança nos momentos em que fraquejei e pensei em desistir.

Agradeço a minha esposa Elaine, pelo incentivo e suporte que eu precisei ao longo desta caminhada. A você toda minha gratidão pelo amor incondicional, paciência e carinho que você teve comigo.

Agradeço os meus filhos, por terem compreendido os momentos de ausência e as vezes que deixei de cumprir com o dever de pai para cuidar dos trabalhos e compromissos relacionados ao mestrado.

Agradeço os colegas e professores que contribuíram com esse trabalho, principalmente ao colega de pesquisa Pedro Moura e aos professores orientadores Prof. Luiz Herique e Prof. Antônio Loureiro por acreditarem no meu potencial e terem me aceito como orientado.

Agradeço também os amigos Rafael Frinhani e Frank Nobre, por compartilharem as angústias dessa caminhada, pelo incentivo e pela companhia nas noites de estudos em que adentramos.

Agradeço os meus pais por acreditarem sempre em mim e que apesar de todas as dificuldades, propiciaram minha formação acadêmica.

Agradeço os colegas da DGTI/UFLA pela colaboração nos momentos em que tive que me ausentar, em especial o Alexandre, secretário e fiel escudeiro nas batalhas do dia a dia; o Anderson pela vezes em que me subistitui como diretor, a Cássia e o Haroldo que colaboraram comigo na escrita deste trabalho e nas atividades administrativas do setor. Enfim, a cada funcionário da DGTI pela dedicação e afinco com que desempenharam suas tarefas e de certa forma me ajudaram a concluir essa empreitada.

Agradeço o Manoel, chefe do setor de transportes da UFLA, na pessoa de quem estendo meus agradecimentos a cada motorista da universidade, que sempre foram muito gentis e prestativos nas constantes viagens à UFMG.

Por último agradeço o Prof. Scolforo, vice-reitor da UFLA, pelo incentivo, compreensão e apoio na conclusão dessa empreitada.

"Não concordo com uma palavra do que dizes, mas defenderei até o ultimo instante seu direito de dizê-la."
(Voltaire)

Resumo

Neste trabalho é proposto a adaptação de um protocolo MAC com foco em decisão do espectro para redes de sensores sem fio cognitivas (RSSFC). O protocolo proposto emprega mecanismos distribuídos de seleção do melhor canal disponível. O objetivo é explorar de forma oportunista o espectro de frequências e permitir que nós pertecentes a redes de sensores sem fio em bandas não licenciadas se adaptem a condições de densidade, ruído, tráfego e *deployment*. O protocolo de decisão do espectro para RSSF proposto neste trabalho se baseou em dois mecanismos de escolha do melhor canal. O primeiro se baseia na escolha do melhor canal em função da força do sinal recebido (RSSI); enquanto que o segundo além de utilizar o RSSI, faz a coleta dos parâmetros de SINR, ruído base e atraso fim a fim, para escolher o melhor canal com base numa técnica clássica de tomada de decisão conhecida como *Analytical Hierarchical Process* (AHP). Esses métodos foram adicionados ao protocolo T-MAC disponível no simulador de RSSF Castalia para serem avaliados sob as métricas taxa de entrega, latência, overhead de transmissão, número de trocas e consumo de energia. Além dos aspectos relacionados a decisão do espectro, foi necessário desenvolver funcionalidades de sensoriamento e mobilidade do espectro. Simulações mostraram que o uso de mecanismos de decisão do espectro possibilitou o aumento da taxa de entrega e manutenção da latência, em cenários de médio e alto ruído, sem impactar consideravelmente no aumento de consumo de energia.

Palavras-chave: Redes de Sensores sem Fio Cognitivas, Gerenciamento do Espectro, Decisão do Espectro, Protocolos MAC Cognitivos, Seleção Dinâmica Distribuída de Canal.

Abstract

This work proposes to adaptate a MAC protocol, focusing on the decision spectrum for Cognitive Radio Sensor Networks (CRSN). This protocol uses dynamic allocation distributed mechanisms of the best channel available. These mechanisms were based on a cognitive adaptation of the T-MAC protocol and on dynamic allocation channels distributed mechanisms. Besides, these protocols treat the spectrum sensing and mobility. Were applied two methods to choose the best channel: the first one is based on the choice of the best channel by the received strength signal indicator (RSSI); the second uses RSSI and obtain the SINR parameters, noise floor and delay, choosing the best channel based on the AHP decision making method. Simulations have shown that the use of these mechanisms improved the packet delivery rate, decreasing the delay, without impacting power consumption.

Keywords: Cognitive Radio Sensor Networks, Spectrum Management, Cognitive MAC Protocols, Dynamic Channel Selection, Spectrum Decision.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Capítulo 1

Introdução

Neste trabalho o foco principal está na decisão do espectro para redes de sensores sem fio cognitivas (RSSFC). Nesse sentido é proposto um protocolo que emprega um mecanismo distribuído de seleção dinâmica do melhor canal disponível, visando a eficiência do uso do espectro e a coexistência de redes de sensores sem fio cognitivas (RSSFC) com outras redes sem fio em faixas do espectro não licenciadas. Como principais contribuições serão apresentados: um algoritmo de decisão do espectro e a adaptação de um arcabouço de simulação para avaliar esse algoritmo em um simulador de RSSF.

1.1 Contextualização

O uso do espectro de rádio-frequências é regulamentado por agências reguladoras de telecomunicações de cada país, como por exemplo a *Federal Communications Commission* (FCC) nos Estados Unidos; Agência Nacional de Telecomunicações (Anatel) no Brasil e o *Office of Communications* (Ofcom) no Reino Unido. Todos esses órgãos definem uma política de alocação estática das faixas de frequências, o que pode implicar em um gerenciamento ineficiente do uso do espectro.

Isso acontece devido a política de concessão definida pelas agências reguladoras, que divide o espectro entre usuários primários (licenciados) e secundários (não licenciados)[1]. As licenças de uso são concedidas observando-se uma faixa de frequência, uma abrangência geográfica e um período de exploração. As concessionárias exploram essas bandas por um longo período de tempo e em regiões geográficas abrangentes.

[1]Usuários primários ou licenciados são usuários que tem a concessão de uso de determinada faixa de frequência outorgada pela agência reguladora de telecomunicação. Os órgãos de regulamentação determinam que os usuários primários não devem sofrer interferências de usuários secundários, e limitam o uso de faixas de frequência mesmo em regiões que não possuem serviços de comunicação utilizando estas faixas.

As faixas de frequências não licenciadas são destinadas às utilizações industriais, científicas e médicas; e podem ser utilizadas livremente. Estas faixas, chamadas de ISM (*Industrial, Scientific, Medical*), foram reservadas inicialmente para dispositivos usando rádios de curto alcance e de baixa potência de sinal. Os equipamentos de radiocomunicação que trabalham nessas faixas, podem operar em três faixas, 900 MHz, 2.4 GHz e 5 GHz [Anatel, 2008]. Essas faixas são de uso livre e possibilitaram a expansão do uso de redes locais de computadores sem fio, e o provimento de acesso a Internet via radiofrequência.

No entanto, o uso indiscriminado das faixas de frequência não licenciadas causou uma poluição do espectro, na qual a ocupação do espectro atinge um patamar de 90% [McHenry, 2005], o que tem inviabilizado a comunicação nessa faixas de frequências, sobretudo na frequência de 2.4 GHz. Por outro lado, as faixas de frequências conhecidas como primárias, ou licenciadas, apresentam em determinadas regiões uma baixa taxa de ocupação do espectro, chegando a menos de 1% [FCC, 2003].

Nos Estados Unidos, a DARPA *(Defense Advanced Research Projects Agency)* propôs um novo paradigma de redes, denominado redes de nova geração ou xG *(NeXtGeneration)*. Esse novo paradigma envolve técnicas de acesso dinâmico ao espectro *(Dynamic Spectrum Access - DSA)* baseadas em rádios inteligentes, também conhecidos como rádios cognitivos [Akyildiz et al., 2006]. Além disso, algumas modificações na política de alocação estática de frequências reguladas pelo FCC têm sido proposta de forma a homologar novos dispositivos que façam alocação dinâmica do espectro [FCC, 2005].

Essas redes de nova geração, também denominadas redes de rádios cognitivos (RRC), devem ser capazes de aprender, monitorando as condições de uso de frequências em determinada região, planejando e atuando de acordo com as suas observações [Akyildiz et al., 2008]. As RRC devem ainda, respeitar a premissa de que um sistema secundário não deve interferir em sistemas primários, e que caso isso ocorra o rádio cognitivo deve atuar prontamente na liberação da frequência licenciada ocupada.

As RRC impõem desafios devido à natureza flutuante do espectro disponível, bem como os diversos requisitos de qualidade de serviço das aplicações. A fim de enfrentar esses desafios, cada usuário na RRC deve [Akyildiz et al., 2008]:

- Determinar quais porções do espectro estão disponíveis;
- Escolher o melhor canal disponível;
- Coordenar o acesso a este canal com outros usuários;
- Desocupar o canal quando um usuário primário é detectado.

Ademais, as RRC devem considerar os requisitos da aplicação, como largura de banda, atraso e vazão. Sendo assim, uma RRC deve possuir um arcabouço de gerenciamento do espectro que realize as seguintes funções: sensoriamento, decisão, compartilhamento e mobilidade do espectro [Akyildiz et al., 2008].

Em linhas gerais, os principais desafios envolvidos no projeto de rádios cognitivos são o sensoriamento e a decisão do espectro, flexibilidade para adaptar os parâmetros de transmissão para maximizar a capacidade do sistema, coexistência com redes sem fio legadas e a necessidade de atender a requisitos de qualidade de serviço das classes de aplicações sem interferir nos usuários primários.

No âmbito das redes de sensores sem fio (RSSF), as técnicas de acesso dinâmico ao espectro usadas nas redes de rádio cognitivo, podem trazer benefícios a essas redes. Em [Akan et al., 2009] é proposto um novo paradigma de redes, no qual ocorre a junção do conceito de rádio cognitivo e redes de sensores sem fio, constituindo Redes de Sensores Sem Fio Cognitivas (RSSFC)[2].

Segundo Akan et al. [2009], esse novo paradigma de rede constituirá a próxima geração das redes de sensores sem fio. O objetivo dessa mescla de tecnologias, é identificar como a incorporação da tecnologia de rádio cognitivo às redes de sensores sem fio (RSSF) tradicionais poderá trazer benefícios a comunicação, como por exemplo a disponibilização de um maior número de canais, menor número de colisões, coexistência de redes sobrepostas e aumento do ciclo de vida.

1.2 Definição do Problema

Neste trabalho o foco central é o tratamento da eficiência de uso do espectro e a coexistência de redes sensores sem fio que utilizem a banda de frequência não licenciada. Dessa forma, o problema consiste em: dados os requisitos da aplicação e os parâmetros da camada física (canal, ruído base, SINR, RSSI) escolher o canal mais eficiente que melhore a taxa de entrega, minimize o ação do ruído base e o consumo de energia dos nós sensores de uma dada região.

Assim, a hipótese de pesquisa é que o uso de um mecanismo de seleção dinâmica de canal, bem como o ajuste dos parâmetros do rádio e o acesso oportunista ao espectro, permitirá que os nós que compõem uma rede sem fio possam coexistir com outros nós de redes heterogêneas (coexistência de nós inter-rede), diminuindo o número de colisões e melhorando o desempenho da rede em termos de consumo de energia e pacotes entregues.

[2]Traduzido do Inglês CRSN - Cognitive Radio Sensor Network.

Nesse sentido, deve-se tratar a interferência mútua entre nós pertencentes a arquitetura de redes distintas operando na mesma faixa de frequência, conforme ilustrado na Figura 1.1.

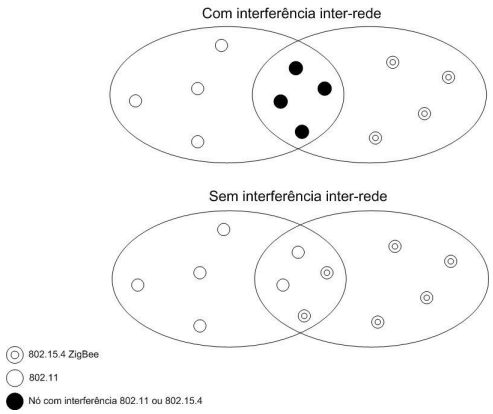

Figura 1.1. Problema da coexistência de nós em redes distintas e a interferência inter-rede.

Conforme pode ser observado na Figura 1.1, o problema de coexistência inter-redes deve considerar regiões de nós cuja a intercessão é constituída por tecnologias de redes que trabalham na mesma frequência. Tais tecnologias devem ser implantadas em determinados locais e faixas do espectro sobrepostos, de maneira a coexistir com mínima interferência possível.

Esse problema pode ser solucionado por meio de técnicas de salto de frequência FHSS (*Frequency Hopping Spread Spectrum*) na qual a informação é transmitida em um espectro de frequência abrangente, com saltos de canais pré-estabelecidos, permitindo que menos interferências ocorram. No entanto, esse tipo de abordagem não elimina completamente a sobreposição de canais, uma vez que os saltos de canais podem coincidir temporariamente com canais já utilizados em em outras redes.

Um exemplo que ilustra essa situação pode ser observado na Figura 1.2, que apresenta um cenário no qual nós pertencentes a redes de diferentes padrões, IEEE 802.15.4 (Zigbee - FHSS) e o IEEE 802.11 (Wi-Fi), divididas em canais cujos os intervalos de frequência são 5 MHz e 22MHz respectivamente, se sobrepõe na maior parte do espectro de frequência analisado.

1.3 Justificativa do Trabalho

Um dos grandes desafios propostos pela NSF (*National Science Foundation*) é a definição de parâmetros de transmissão, como controle de potência, antenas inteligentes

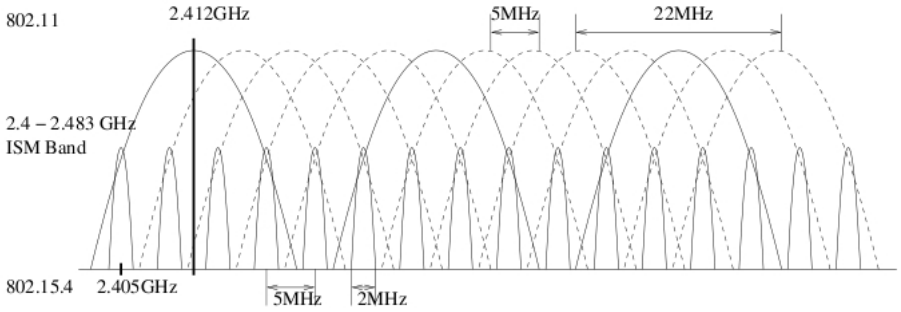

Figura 1.2. Coexistência entre nós pertencentes a redes IEEE 802.15.4 e o IEEE 802.11.

e ciclo de operação [McHenry, 2005]. Com exceção do projeto de antenas inteligentes, os demais podem ser resolvidos por meio de rádios cognitivos.

A pesquisa na área de rádios cognitivos conta com muitos problemas em aberto para serem explorados. Dos quais, destacamos a busca de métodos eficazes de coordenação e ocupação das faixas de frequência e o compartilhamento do espectro entre os usuários, de maneira que os níveis de interferência fiquem em um patamar tolerável.

Nesse sentido, as técnicas de acesso dinâmico ao espectro surgem como uma abordagem promissora e eficiente para redes de sensores sem fio, visto que, RSSF podem gerar tráfego em rajadas e ocasionar um excesso de colisões entre os nós dessa rede além de um consumo desnecessário de energia e perda de desempenho.

Dessa forma, o uso de técnicas de rádio cognitivos em RSSF pode melhorar a taxa de entrega de pacotes, permitir a implantação de redes de sensores sobrepostas, suportar a coexistência com redes convencionais e evitar longos períodos de contenção em regiões onde a densidade de nós é muito alta.

Segundo [Akan et al., 2009], as capacidades dos rádios cognitivos podem ser exploradas por RSSF, que tradicionalmente empregam atribuição fixa do espectro e possuem *hardware* com recursos limitados, visando se beneficiar de algumas vantagens potenciais, que são:

- Acesso Dinâmico ao Espectro: a maioria dos nós sensores utilizam alocação fixa das faixas de frequência não licenciadas, que já estão sobrecarregadas [Zhou et al., 2006].

- A utilização oportunista do canal para tráfego em rajadas: um grande número de nós sensores ao detectar um evento, podem gerar tráfego em rajadas e tentar adquirir o canal para enviar suas leituras. Isso aumenta a probabilidade de col-

isões e perdas de pacotes. O acesso oportunista a vários canais alternativos pode minimizar esse problema.

- Adaptabilidade para reduzir o consumo de energia: a natureza dinâmica do canal sem fio faz com que aumente o consumo de energia devido às perdas de pacotes e retransmissões. Usando nós sensores dotados de rádios cognitivos, é possível adaptá-los às condições variáveis do canal, consequentemente aumentar a eficiência da transmissão e, portanto reduzir a energia usada para transmissão e recepção.

- Implantação sobreposta de múltiplas RSSF: o gerenciamento dinâmico do espectro pode contribuir significativamente para a coexistência e sobreposição espacial de redes de sensores sem fio.

- Comunicação por meio de diferentes faixas do espectro: uma certa faixa de frequência disponível em uma determinada região ou país pode não estar disponível em outra, devido à variação na regulamentação do uso do espectro. Nós sensores equipados com capacidade cognitiva podem superar este problema.

Por fim, há muitas questões de investigação em aberto, inerentes ao desenvolvimento de técnicas de decisão do espectro para RSSFC, como por exemplo, a definição de parâmetros a serem utilizados na decisão do espectro (relação sinal-ruído, capacidade de canal, atrasos, etc) e algoritmos que permitam a escolha do melhor canal disponível. Além disso, a escolha de métodos energeticamente eficientes e distribuídos também é outro aspecto que deve ser cuidadosamente investigado.

1.4 Solução Proposta

A solução proposta permeia questões relacionadas ao projeto dos rádios cognitivos, no que tange à capacidade de cognição e reconfiguração dinâmica dos parâmetros dos transceptores. As técnicas empregadas em rádios cognitivos levam em consideração os requisitos da aplicação, como por exemplo o tipo de tráfego gerado e atraso fim a fim, além de variações do ambiente, interferências em frequências adjacentes, potência de transmissão e qualidade do enlace.

Dessa forma, os rádios cognitivos devem aprender por meio de observação, sensoriando o espectro e planejando qual a frequência a ser utilizada para reduzir ou mesmo eliminar as interferências na rede. Assim, a proposta deste trabalho é tratar aspectos relacionados a decisão do espectro, o sensoriamento e a mobilidade do espectro.

A decisão do espectro se baseia em parâmetros sensoriados pelos rádios, tais como frequência e canal livre, relação sinal interferência ruído (SINR - *Signal Noise Ratio*), ruído base do meio de transmissão, força do sinal recebido (RSSI - *Received Signal Strength Indicator*) e em requisitos da aplicação.

O algoritmo de decisão do espectro proposto é capaz de ler esses parâmetros, e fazer a escolha do melhor canal, para diferentes classes de tráfegos (exponencial, periódico, aleatório) da aplicação e se ajustar a variações do atraso fim a fim imposto a determinado canal. Além disso, emprega um mecanismo distribuído de seleção dinâmica do melhor canal disponível que foi incorporado ao protocolo T-MAC [van Dam & Langendoen, 2003] dando origem a uma adaptação cognitiva do protocolo T-MAC.

Foram empregados dois métodos para escolha do melhor canal, o primeiro se baseia na escolha do melhor canal em função da força do sinal recebido (RSSI); enquanto que o segundo além de utilizar o RSSI, coleta os parâmetros de SINR, ruído base e atraso fim a fim, escolhendo o melhor canal com base no método de tomada de decisão AHP[Saaty, 2000].

Assim, o que é comum em ambos os métodos é o mecanismo de seleção dinâmica de canal e reconfigurabilidade dos parâmetros do rádio, para permitir a escolha do melhor canal disponível para os nós sensores que desejarem comunicar com seus vizinhos.

Para avaliar o algoritmo de decisão e seus dois métodos de seleção do melhor canal disponível, foi utilizado o simulador de RSSF Castalia [Boulis, 2010].

1.5 Organização do Trabalho

No capítulo 2 são apresentados os fundamentos dos sistemas de rádio baseados na cognição, bem como alguns trabalhos relacionados a rádio cognitivo, redes de rádio cognitivo (RRC) e protocolos MAC para rádio cognitivo.

No capítulo 3 é feita uma revisão bibliográfica do paradigma rede de sensores fio cognitiva (RSSFC), descrevendo as principais funcionalidades do arcabouço de gerenciamento do espectro, bem como discutindo aspectos da coexistência de RSSFC.

No capítulo 4 é apresentado o processo metodológico para avaliação da solução proposta. No capítulo 5 o algoritmo de decisão do espectro e os dois métodos de escolha do melhor canal disponível são apresentados, bem como os modelos de interferência e de rádio cognitivo utilizados no arcabouço de simulação.

No capítulo 6 são apresentadas as métricas, os cenários avaliados e a discussão dos resultados obtidos. Por fim, no capítulo 7 o trabalho é arrematado com as conclusões

que se pode tirar da avaliação da solução proposta, bem como sinalizar para trabalhos futuros não contemplados neste trabalho.

Capítulo 2

Referencial Teórico

Nesse capítulo são apresentados os conceitos de rádio cognitivo, redes de rádio cognitivo, arcabouço de gerenciamento do espectro, redes de sensores sem fio e protocolos MAC para RSSF. A compreensão desses conceitos se faz necessária, sobretudo para contextualizarmos que parte do arcabouço de gerenciamento do espectro será abordada na solução proposta neste trabalho.

2.1 Rádio Cognitivo

Na últimas décadas ocorreu uma mudança de paradigma, na qual os rádios que antes eram baseados apenas em *hardware*, passaram por mudanças na sua concepção e se tornaram rádios híbridos baseados em uma combinação de *hardware* e *software*. Em meados da década de noventa do século passado, o pesquisador Joseph Mitola denominou esses novos rádios como *Software Defined Radio* (SDR) [Mitola, 1993].

O SDR é um rádio que inclui um transmissor no qual parâmetros operacionais como frequência, tipo de modulação ou potência máxima de saída pode ser alterada por meio de software sem fazer quaisquer alterações nos componentes de *hardware* responsáveis pela transmissão em rádio-frequência [FCC, 2001].

Dessa forma, o SDR permite a implementação de rádios reconfiguráveis que aliada a capacidade de aprender com o ambiente, torna possível o desenvolvimento rádios cognitivos [Haykin, 2005]. Um rádio cognitivo pode reconfigurar seus parâmetros (frequência de operação, largura de banda, potência de transmissão ou modulação) para melhorar seu desempenho.

A maioria dos rádios cognitivos provavelmente são SDR, porém nem todos os rádios cognitivos têm como requisitos a necessidade de um rádio em software ou serem reprogramáveis. Por exemplo, um telefone sem fio na faixa 43,71-44,49 MHz é uma

forma simples de rádio cognitivo, pois esses telefones devem possuir um mecanismo de seleção automática de canais para evitar a operação nos canais ocupados [FCC, 2003].

As técnicas de rádio cognitivos fornecem a capacidade de compartilhar o espectro de uma forma oportunista. Formalmente, um rádio cognitivo (RC) é definido como um rádio que pode alterar os parâmetros de seu transmissor baseado na interação com o meio ambiente [FCC, 2003].

Essa interação pode envolver a negociação ativa ou a comunicação com outros usuários do espectro e/ou detecção passiva com tomada de decisão dentro do rádio.

Duas características principais do rádio cognitivo são a capacidade cognitiva e a reconfigurabilidade, ambas são definidas a seguir [Haykin, 2005].

2.1.1 Capacidade Cognitiva

Cognição[1] é o *"ato ou efeito de conhecer, processo ou faculdade de adquirir um conhecimento"*, podendo significar ainda percepção e conhecimento. Dessa forma, a cognição é a aquisição de conhecimento por meio da percepção e a capacidade cognitiva é o ato de aprender com o ambiente. A capacidade cognitiva busca a modelagem de mecanismos de cognição, que possam tomar decisões de maneira eficiente, reagindo a novas situações e aprendendo com a experiência.

2.1.2 Reconfigurabilidade

Um rádio cognitivo pode ser programado para transmitir e receber em frequências variadas, e utilizar diferentes tecnologias de acesso suportadas pelo seu projeto de *hardware* [Jondral, 2005]. Por meio deste recurso, a melhor banda do espectro e os parâmetros de operação mais adequados podem ser selecionados e reconfigurados.

Ao alterar dinamicamente seus parâmetros operacionais, o rádio cognitivo tem a capacidade de ser programado para transmitir em uma ampla gama de frequências disponíveis de forma oportunista, melhorando a utilização do espectro total.

Com esses recursos, o rádio cognitivo pode operar tanto em faixas licenciadas, quanto nas faixas não licenciadas. Desta forma, tantos usuários primários (usuários que tem a outorga de uso da faixa de espectro) quanto os usuários secundários podem utilizar as faixas de frequências licenciadas, desde que o usuário secundário não cause interferência ao usuário primário.

[1]Definição extraída do dicionário Houaiss.

Uma vez caracterizado as duas principais características de um rádio cognitivo, é necessário descrever como esses dispositivos interagem. Na seção seguinte o conceito de ciclo cognitivo define como essa interação ocorre.

2.1.3 Ciclo Cognitivo

Quando um usuário primário inicia a comunicação, o usuário de rádio cognitivo deve detectar as frequências potencialmente livres (sensoriamento ou detecção do espectro), para decidir para qual canal deve mover-se (decisão espectro) e, finalmente, adaptar o seu transceptor para que a comunicação ativa continue no novo canal escolhido (*handoff spectrum*). Essa sequência de operações apresenta um ciclo cognitivo [Haykin, 2005], que também pode ser aplicado sobre uma faixa não licenciada por todos os usuários de rádio cognitivo.

Todas essas operações listadas anteriormente devem ser executadas de maneira a tornar eficiente a utilização do espectro eletromagnético. Para que isto ocorra, essas operações devem seguir uma dinâmica de operação denominada ciclo cognitivo [Mitola & Maguire, 1999]. O ciclo cognitivo é mostrado na Figura 2.1.

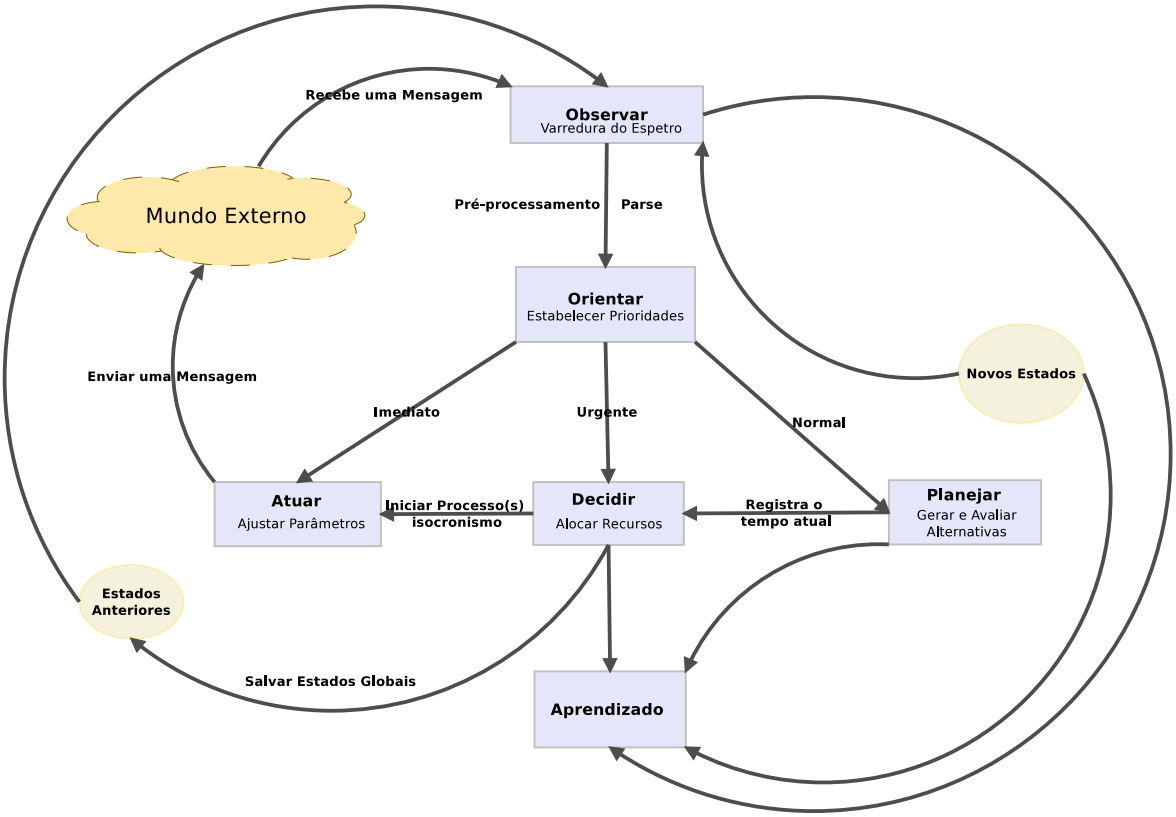

Figura 2.1. Ciclo Cognitivo.

O ciclo cognitivo permite que um nó ao obter conhecimento do contexto, seja capaz de estar ciente de seu ambiente operacional, a fim de detectar os espaços de frequência não utilizados e usá-los de forma inteligente e eficaz [Mitola & Maguire, 1999].

O ciclo cognitivo é composto por seis estágios principais:

- Observar,
- Orientar,
- Agir,
- Decidir,
- Planejar,
- Aprender.

No estado Observar, ocorre o monitoramento do ambiente operacional. Em seguida, o estado Orientar determina a importância e a prioridade em que as ações serão tomadas. No caso de prioridade imediata, o próximo estágio é Agir, se a prioridade definida for urgente o próximo estágio é Decidir. Se a prioridade for normal o próximo estágio é Planejar. O estado Decidir determina a próxima ação, enquanto o estado Planejar gera e avalia alternativas e traça um plano de ações a longo prazo. No estado Agir, a ação escolhida é executada, e suas consequências são armazenadas no estágio Aprender.

Após apresentar o conceito de ciclo cognitivo, é necessário descrever a organização e a arquitetura de redes de rádio cognitivo (RRC), bem com os aspectos relacionados ao gerenciamento do espectro nas seções a seguir.

2.2 Redes de Rádio Cognitivo

As redes de rádio cognitivo RRC são redes sem fio que não estão licenciadas para operar em uma determinada faixa frequência, porém por meio do acesso oportunista ao espectro, compartilham a banda de espectro com usuários primários (licenciados). As RRC impõem desafios únicos devido à coexistência com as redes primárias e carecem de funções de gerenciamento do espectro que serão descritas mais adiante em um arcabouço de gerenciamento do espectro [Akyildiz et al., 2008].

Redes de rádio cognitivo (RRC) devem fornecer banda larga a usuários móveis por meio de arquiteturas sem fio heterogêneas e técnicas de acesso dinâmico ao espectro. No entanto, as RRC impõem desafios devido à natureza flutuante do espectro disponível, bem como os diversos requisitos de qualidade de serviço das aplicações diversas. Funções de gerenciamento do espectro podem enfrentar esses desafios para a realização deste novo paradigma de rede [Akyildiz et al., 2008].

Para promover à coexistência entre usuários primários e secundários um arcabouço de gerenciamento do espectro deve levar em consideração os seguintes desafios de projeto:

- Evitar interferência com redes primárias;
- Sensibilidade à QoS: encontrar uma faixa de espectro adequada que suporte comunicações sensíveis à QoS, considerando o ambiente heterogêneo e dinâmico do espectro;
- Comunicação transparente: prover transparência na comunicação independentemente da existência de usuários primários.

Para enfrentar esses desafios algumas funcionalidades devem estar presentes nas RRC. Para tanto, o processo de gerenciamento do espectro deve consistir de quatro etapas principais, conforme pode ser visto na Figura 2.2 e descritas como [Akyildiz et al., 2008] :

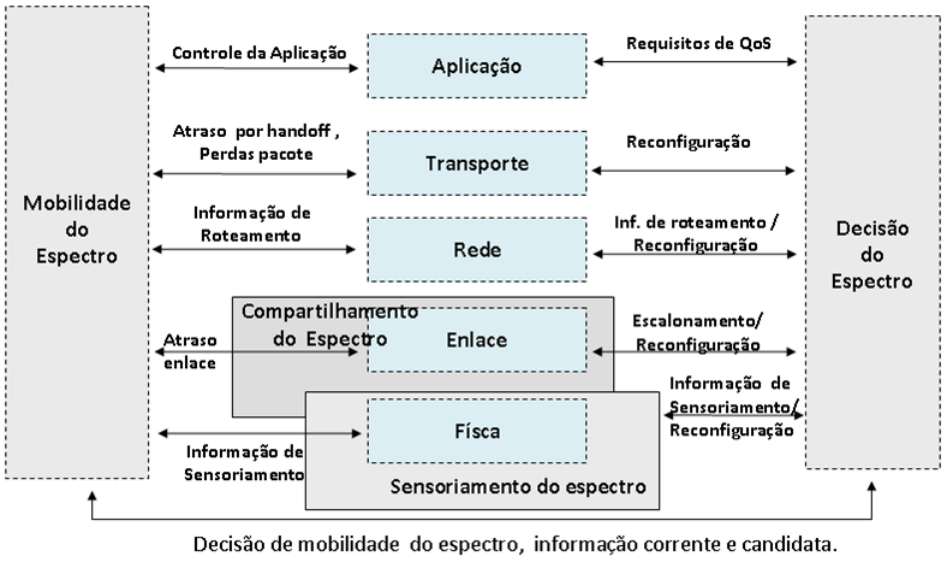

Figura 2.2. Arcabouço de Gestão do Espectro. *[Akyildiz et al., 2008].*

- Sensoriamento ou detecção do espectro: um usuário RC (rádio cognitivo) pode alocar somente uma faixa não usada do espectro. Desse modo, um usuário RC deve monitorar o espectro, capturar sua informação e detectar faixas não utilizadas do espectro.

- Decisão do espectro: baseado na disponibilidade do espectro, usuários RC podem alocar um canal. Essa alocação depende também de políticas internas e até de políticas externas;

- Compartilhamento do espectro: como podem existir vários usuários RC tentando acessar o espectro, o acesso à RRC deve ser coordenado de modo a prevenir a colisão de múltiplos usuários em faixas do espectro;

- Mobilidade do espectro: usuários RC são considerados visitantes do espectro. Assim, se uma faixa específica do espectro é requisitada por um usuário primário, a comunicação do usuário RC deve ser realizada em outra faixa disponível de espectro.

O arcabouço ilustrado na Figura 2.2 mostra que, devido ao número significativo de interações entre as funções de gerenciamento do espectro, seu projeto exige uma abordagem em camada. Nas seções seguintes são apresentadas as quatro principais funções de gerenciamento do espectro.

2.2.1 Sensoriamento do Espectro

As técnicas de sensoriamento de espectro podem ser classificadas em três grupos: detecção de transmissor primário, detecção de receptor primário e gerenciamento de interferência de temperatura [Akyildiz et al., 2008]. Existem vários problemas abertos que devem ser investigados para o desenvolvimento de técnicas de detecção de espectro:

- Medição da interferência: devido à falta de interação entre as redes primárias e as RRC, geralmente um usuário RC não pode estar certo da localização precisa dos receptores primários;

- Sensoriamento de espectro em redes multiusuários: as redes multiusuários (usuários primários e usuários RC) tornam mais difíceis a detecção de buracos no espectro e a estimativa de interferência;

- Sensoriamento eficiente de espectro: a detecção não pode ser realizada enquanto pacotes estiverem sendo transmitidos. Usuários RC tem que parar de transmitir enquanto estiver ocorrendo o sensoriamento do espectro. Isso faz com que

a eficiência do espectro seja diminuída. Por esta razão, equilibrar a eficiência do uso do espectro e a precisão do sensoriamento é um importante desafio de pesquisa. Ademais, como o tempo de detecção afeta diretamente o desempenho da transmissão, então os algoritmos de sensoriamento do espectro devem ser desenvolvidos de tal forma que o tempo de detecção seja minimizado dentro de uma determinada precisão que não comprometa a transmissão.

2.2.2 Decisão do Espectro

Redes de rádio cognitivo devem decidir qual é a melhor banda de espectro entre as bandas disponíveis de acordo com os requisitos de qualidade de serviço das aplicações. A decisão de espectro constitui um tema bastante importante, mas ainda inexplorado em redes RC [Akyildiz et al., 2008].

A decisão do espectro consiste de duas etapas: primeiro, cada faixa do espectro é caracterizada, com base em observações locais, não só dos usuários RC, mas também em informações estatísticas de usuários primários. Então, com base nestas características, a banda de espectro mais adequada pode ser escolhida.

A disponibilidade de espectro possui características que variam com o tempo, cada faixa disponível deve ser caracterizada considerando tanto o ambiente dinâmico e parâmetros do espectro como a largura de banda e frequência. Por isso, é essencial definir parâmetros para diferenciar cada faixa do espectro disponível [Akyildiz et al., 2008]. Alguns problemas permanecem sem solução na questão de decisão de espectro:

- Modelo de decisão: apenas a estimativa de capacidade do espectro usando a taxa sinal-ruído (SNR) não é suficiente para caracterizar a banda de espectro em RRC. Além disso, deve-se considerar os diferentes requisitos de qualidade das aplicações. Assim, o projeto de aplicação e de modelos adaptativos da decisão do espectro é uma questão em aberto que carece de pesquisa [Akyildiz et al., 2008].

- Cooperação com reconfiguração: técnicas de rádio cognitivo devem permitir a reconfiguração de parâmetros de transmissão para o funcionamento ideal em certa banda de espectro. Assim, se a relação sinal ruído for alterada, é desejável que a taxa de transmissão e a taxa de erro sejam mantidas. Dessa forma, um mecanismo de cooperação entre os rádios que permita a reconfiguração é necessária na tarefa de decisão do espectro.

- Decisão de espectro sobre bandas de espectro heterogêneas: algumas faixas do espectro são atribuídas para propósitos diferentes, enquanto algumas bandas per-

manecem não licenciadas. Assim, uma RRC deverá apoiar as operações de decisão do espectro, tanto nas bandas licenciadas quanto nas não licenciadas.

2.2.3 Compartilhamento do Espectro

O compartilhamento do espectro deve realizar a coordenação de tentativas de transmissão entre os usuários RC, além de incluir muitas das funcionalidades de um protocolo MAC e promover a coexistência de usuários RC com usuários licenciados no intervalo de espectro disponível. Os trabalhos existentes no compartilhamento do espectro podem ser classificados em quatro aspectos: a arquitetura, o comportamento de alocação do espectro, a técnica de acesso ao espectro e o alcance [Akyildiz et al., 2008].

Com relação a arquitetura, o compartilhamento pode ser centralizado: a alocação de espectro e os procedimentos de acesso são controlados por uma entidade central. Um procedimento distribuído de sensoriamento pode ser usado e as medidas de alocação do espectro são encaminhadas para a entidade central, construindo-se um mapa de alocação de espectro. Na arquitetura distribuída cada nó define o acesso ao espectro de forma distribuída.

De acordo, com o comportamento de alocação, o acesso ao espectro pode ser cooperativo e não cooperativo. Na alocação cooperativa são exploradas as medidas de interferência de cada nó, tal que o efeito da comunicação de um nó em outros nós é considerado. Uma técnica comum usada nesse tipo alocação é o compartilhamento das informações sobre interferências locais causadas pelo nós pertencentes a determinados grupos *clusters*. Esta operação localizada é mais equilibrada que uma solução totalmente centralizada ou uma solução distribuída.

Na alocação não-cooperativa, apenas um único nó é considerado e a interferência causada por outros nós não é considerada. Soluções não-cooperativas podem resultar na utilização reduzida do espectro. No entanto, estas soluções não requerem frequentes trocas de mensagens entre vizinhos como em soluções cooperativas.

Com relação a tecnologia de acesso, o compartilhamento do espectro pode feito por sobreposição (*Overlay*), no qual nós de acesso à rede usando uma faixa do espectro não usado por usuários licenciados. Uma outra alternativa é o compartilhamento de espectro subjacente (*Underlay*), no qual as técnicas de espalhamento espectral são exploradas de tal forma que a transmissão de um nó de RC é considerado como ruído pelos usuários licenciados.

Por último, as redes podem ser classificadas de acordo com a localização dos nós: dentro de uma rede RRC (intra-rede) e entre as várias RRC coexistentes (inter-rede), como mostrado na Figura 2.3 [Akyildiz et al., 2008].

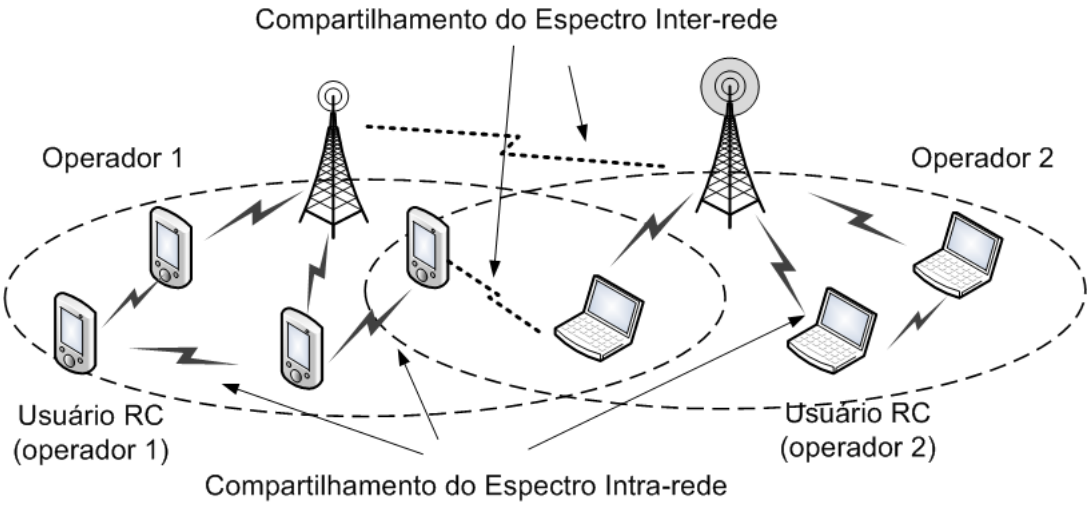

Figura 2.3. Compartilhamento do espectro. *[Akyildiz et al., 2008]*

O compartilhamento do espectro intra-rede trata a alocação do espectro entre as entidades de uma RRC, de forma que os usuários de uma RRC devem acessar o espectro disponível sem causar interferência para os usuários primários. O compartilhamento do espectro intra-rede impõe desafios únicos que não têm sido consideradas em sistemas de comunicação sem fio.

No compartilhamento inter-rede, a arquitetura de rádios cognitivos permitem que múltiplos sistemas sejam implantados em determinados locais e faixas do espectro sobrepostos. Atualmente, as soluções de compartilhamento de espectro inter-rede fornecem uma visão mais ampla do conceito de compartilhamento do espectro, incluindo as políticas de determinado operador. Por último, destaca-se os potenciais desafios e questões de investigação em aberto, referentes ao compartilhamento do espectro [Akyildiz et al., 2008]:

- Controle do Canal Comum (CCC): para permitir a cooperação entre os usuários de RC, um CCC é necessário para a troca de informações sobre o espectro e coordenar o acesso ao espectro [Akyildiz et al., 2009]. Alguns desafios são impostos pelo projeto de CCC, dentre eles destacam-se a escolha do canal, a definição da área de cobertura e o *overhead* causado pela mobilidade do espectro. Com relação a escolha do canal para a CCC, é importante que a transmissão não seja interrompida durante longos períodos de tempo. Logo, a principal dificuldade é identificar um canal uniformemente aceitável sobre grande parte da rede. Além disso, deve-se tomar cuidado para assegurar que a CCC não diminua a eficiência de utilização de espectro em cenários de tráfego baixo, quando uma faixa

exclusiva do espectro é reservada para o controle de mensagens[2].

- Alcance dinâmico de rádio: devido à interdependência entre o alcance do rádio e a frequência, os vizinhos de um nó podem mudar assim que a frequência de operação muda.

- Unidade de espectro: quase todas as técnicas de decisão e compartilhamento de espectro consideram o canal como a unidade básica de espectro. Uma definição de um canal como um espectro é crucial no desenvolvimento de algoritmos.

- Localização da informação: na maioria das vezes, assumimos que os usuários secundários conhecem a localização e potência de transmissão de usuários primários, e então o cálculo de interferência pode ser realizado facilmente. Entretanto, essa suposição não é válida sempre.

2.2.4 Mobilidade do Espectro

Quando um usuário primário acessa um canal usado por um usuário secundário, este usuário deve desocupar o espectro para dar lugar ao usuário licenciado. O principal objetivo é manter a transparência na comunicação. A mobilidade entre uma faixa do espectro e outra deve ser feita de maneira eficiente minimizando a degradação de desempenho. Os problemas para a eficiência da mobilidade de espectro em RRC são [Akyildiz et al., 2008]:

- Mobilidade de espectro no tempo: como os canais disponíveis variam com o tempo, habilitar QoS nesse ambiente é um problema;

- Mobilidade de espectro no espaço: a disponibilidade de banda também muda quando um usuário move de um lugar para o outro. Por isso, alocação contínua de espectro é um problema.

Nas RRC, a identificação dos recursos disponíveis por meio do sensoriamento do espectro, a decisão sobre os tempos ótimos de sensoriamento e transmissão, bem como a coordenação com os outros usuários para acesso ao espectro são funcionalidades importantes tratadas pelos protocolos de controle de acesso ao meio (MAC) [Cormio & Chowdhury, 2009].

A seguir são apresentadas, as características de alguns protocolos MAC cognitivos investigados para ambas as arquiteturas de rede: infra-estruturadas e redes *ad hoc*. Em seguida, uma classificação dos protocolos MAC cognitivos é apresentada.

[2] *Out-of-band CCC.*

2.3 Protocolos MAC para RRC

A principal motivação para a pesquisa em protocolos MAC de rádios cognitivos (MAC-RC) é o gerenciamento eficiente do uso do espectro. O objetivo é fornecer meios eficazes de sensoriamento do canal para determinar a sua ocupação e o compartilhamento do espectro entre os usuários do RC e usuários, mantendo os níveis de interferência em patamares toleráveis.

O projeto de protocolos MAC-RC seguiu duas abordagens, esforços de padronização, levando à formação do grupo de trabalho IEEE 802.22 [Cordeiro et al., 2005], e a aplicação/cenário específico. A primeira abordagem consiste principalmente em redes infra-estruturadas, em que um coordenador centralizado ou estação rádio base gerencia a alocação e o compartilhamento de espectro entre os usuários. Os usuários RC, no entanto, podem participar com o sensoriamento do espectro fornecendo informações sobre o canal para o controlador central. Os esforços de padronização visam levar a uma uniformidade no projeto de protocolos e políticas, permitindo que vários operadores de RC possam coexistir [Cormio & Chowdhury, 2009].

Na segunda abordagem, a aplicação ou cenário de protocolos específicos é otimizada para um tipo particular de ambiente ou usuário. Esta abordagem tem sido cada vez mais utilizada em protocolos MAC-RC distribuídos, que operam sem o apoio de uma entidade de controle centralizado. Como exemplo, nós em uma rede *ad hoc* podem apresentar altos níveis de mobilidade e tornar difícil a coordenação de sensoriamento [Cormio & Chowdhury, 2009].

Para esses casos, o protocolo MAC pode considerar a mobilidade e determinar quais regiões abrangidas pelo nó devem ser consideras durante o seu movimento, informando os níveis de atividade de usuários primários. A disponibilidade de espectro de uma região pode ser aprendida pelo nó ao longo do tempo, de tal forma que a escolha do espectro de transmissão pode ser melhorada.

Além disso, como os dados coletados por um nó é baseado em suas próprias observações, o tempo dedicado ao sensoriamento na fase inicial do processo de aprendizagem pode ser maior do que o tempo útil de transmissão de dados. Apesar disso, proporcionar maior proteção aos usuários primários no longo prazo pode impactar em redução na taxa de transferência [Cormio & Chowdhury, 2009].

A tarefa de sensoriamento dos espectro realizada pelos protocolos MAC-RC permite explorar oportunidades em faixa espectro vagas e evitar interferências com os usuários primários. As duas abordagens apontadas por [Akyildiz et al., 2006], para o sensoriamento são: sensoriamento do transmissor primário e sensoriamento do receptor primário.

A detecção do transmissor primário baseia-se na detecção do sinal fraco de um primário transmissor através de observações locais de usuários RC. A detecção do receptor primário visa encontrar os usuários primários que estão recebendo dados dentro do alcance de comunicação de um usuário do RC.

A maioria das pesquisas atuais sobre sensoriamento do espectro, focaliza a detecção do transmissor primário. Segundo [Cormio & Chowdhury, 2009] as principais questões investigadas são:

- Qual o tempo o ideal de sensoriamento e durações das transmissões (otimização da duração do sensoriamento do espectro e da transmissão).
- Em que ordem o espectro deve ser varrido para minimizar o tempo de encontrar a faixa de espectro disponível (otimização da sequência de busca do espectro).

Enquanto que os principais desafios no sensoriamento do espectro são:

- coordenação do sensoriamento;
- propagação de informações de sensoriamento;
- duração ótima do sensoriamento;
- integração do sensoriamento com outros parâmetros de transmissão [Cormio & Chowdhury, 2009].

O acesso ao espectro permite a múltiplos usuários RC compartilharem os recursos do espectro por meio da determinação de quem irá acessar o canal (alocação aleatória do canal) ou quando um usuário vai acessar o canal (alocação do canal por meio de intervalos discretos).

Enquanto ambos os esquemas de acesso, intervalos de tempo discreto e acesso aleatório, podem ser utilizados nas redes infra-estruturadas, a dificuldade de sincronização dos tempos em toda extensão das redes móveis *ad hoc*, torna inviável a adoção de protocolos completamente discretizados.

As abordagens existentes em protocolos de acesso aleatório, protocolos de tempo discretizado, e protocolos híbridos, pode ser visto na Figura 2.4 [Cormio & Chowdhury, 2009]. Além disso, o número de transmissores de rádio também determinam o funcionamento do protocolo MAC. A classificação proposta divide os protocolos em acesso aleatório, tempo discretizado e híbrido:

- Protocolos de acesso aleatório: nesta classe de protocolos não há a necessidade de sincronização de tempo, e são geralmente baseados no protocolo de acesso

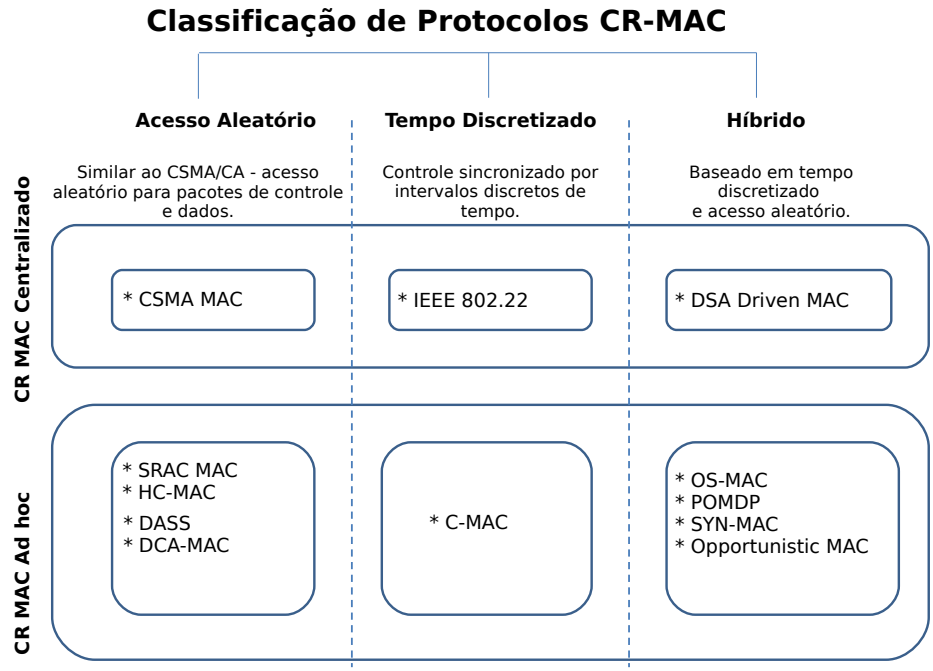

Figura 2.4. Classificação dos protocolos MAC-RC.[Cormio & Chowdhury, 2009]

múltiplo com detecção de portadora e prevenção de colisões (CSMA/CA [3]). Dessa forma, o usuário RC monitora a banda do espectro para detectar quando não há transmissão do RC de outros usuários e transmite após um período de contenção, denominado *backoff*, para prevenir transmissões simultâneas.

- Protocolos de tempo discretizado: estes protocolos MAC têm a necessidade de sincronização dos tempos de cada nó da rede, de maneira que o tempo seja dividido em intervalos de tempos discretos (*slots*) tanto para o controle do canal quanto para a transmissão de dados.

- Protocolos híbridos: neste tipo de protocolo o controle de sinalização geralmente ocorre durante intervalos de tempo sincronizados. No entanto, a transmissão de dados podem ter esquemas de acesso aleatório do canal, sem sincronização de tempo. Em uma outra abordagem, as durações das sinalizações de controle e da transferência de dados pode ter tempos pré-definidos comum a todos os usuários da rede. Dentro de cada período de controle ou de dados, o acesso ao canal pode ser completamente aleatória.

[3]Carrier sense multiple access with collision avoidance é um método de transmissão que visa diminuir a ocorrência de colisões em uma rede. Para tanto, antes de transmitir um pacote, a estação sinaliza sua intenção de transmitir.

Os principais desafios de pesquisa em protocolos de acesso ao espectro são descritos a seguir:

- Projeto do canal de controle: o acesso ao espectro envolve o controle de sinalização entre os dois usuários RC em ambos as extremidades do enlace. Esta troca de mensagens deve ser ininterrupta tanto pela atividade de usuários primários vizinhos, como pela troca de informações de sensoriamento e coordenação do acesso ao canal. Para isso, deve ser concebido um mecanismo de mudança dos canais de controle de maneira dinâmica e confiável [Cormio & Chowdhury, 2009].

- Adaptação às características de transmissão dos usuários primários: alguns usuários primários têm padrões de transmissão regulares ou podem ter acesso ocasional ao canal. Nessas situações, o protocolo MAC-RC pode contribuir para a inferência da natureza do tráfego do usuário primário e adaptar a sua transmissão. Com conhecimento prévio do tipo de tráfego pode-se realizar o controle dinâmico de potência e o escalonamento de transmissão dos usuários de RC [Cormio & Chowdhury, 2009].

Esses desafios impõe a necessidade de se desenvolver protocolos MAC-RC eficientes que tratem a questão do acesso ao espectro. A seguir são apresentados o padrão IEEE 802.22 – que estabelece especificações das camadas físicas e MAC para rádio cognitivo – bem como alguns dos principais protocolos MAC-RC.

2.3.1 Padrão 802.22

O IEEE 802.22 é a padronização das redes regionais ou *Wireless Regional Area Networks* (WRAN). O padrão visa explorar de forma oportunista os espaços não utilizados ou em branco dentro das bandas de televisão entre 54 e 862 MHz, especialmente nas zonas rurais, onde o uso do espectro pode ser menor. Trata-se do primeiro padrão a estabelecer especificações das camadas físicas e MAC para rádio cognitivo [Cordeiro et al., 2005].

Esse grupo de trabalho criado em 2004, tinha como objetivo levar acesso banda larga sem fio a áreas rurais. De maneira geral, pode-se dizer que o padrão trata questões de eficiência espectral e limiares de detecção, no intuito de compartilhar o espectro eletromagnético alocado aos canais de TV nas faixas VHF[4] e UHF[5].

[4]VHF é a sigla para o termo inglês *Very High Frequency* que designa a faixa do espectro radio frequências de 30 a 300 MHz.

[5] VHF - sigla para o termo inglês *Very High Frequency*, que designa a faixa de frequências do espectro compreendidas entre 300 MHz e 3 GHz.

O padrão IEEE 802.22 especifica ainda, uma interface de comunicação sem fio ponto-multi-ponto (P2MP) pelo qual uma estação base controla sua própria célula e todos os usuários associados. Segundo [Cordeiro et al., 2005], o sistema é composto por estações base, também chamados de pontos de acesso, e por equipamentos clientes dessas estações, denominados CPE[6].

As estações base irão controlar o meio de acesso para todos os CPE ligados a ela. Uma característica fundamental das estações base WRAN é que elas serão capazes de realizar uma detecção distribuída, na qual o CPE irá detectar o espectro e enviará relatórios periódicos às estações base informando-as sobre o que foi detectado.

A partir da informação coletada, a estação base vai avaliar se é necessária uma mudança no canal utilizado ou se deve manter transmissão e recepção no mesmo canal. Atualmente, o sistema WRAN deve detectar a presença do sinal de TV digital, TV analógica e microfones sem fio.

Outro parâmetro importante definido pelo padrão é a área de cobertura. Essa característica distingue o IEEE 802.22 WRAN dos demais padrões IEEE 802 existentes, pois prevê uma ampla cobertura da estação-base. O valor nominal desse parâmetro em uma WRAN é de 33 km, a uma potência isotrópica efetivamente irradiada (EIRP)[7] de 4W. Caso a regulamentação permita aumentar os limites de potência, o alcance pode ir até aos 100 km.

Além disso, uma WRAN IEEE 802.22 deve possuir uma largura de banda por canal de 18 Mbps, taxas máximas de download de 1,5 Mbps e 384 Kbps de upload por usuário. A Tabela 2.1, mostra outros parâmetros de capacidade e cobertura de uma WRAN IEEE 802.22 [Akyildiz et al., 2006].

Para promover a eficiência espectral, é necessário que a camada MAC, seja capaz de se adaptar dinamicamente às mudanças no ambiente utilizando detecção do espectro. Para tanto, duas estruturas de quadros são empregadas: *frame* e *superframe*.

Um *superframe* é formado por vários quadros sendo constituído por um cabeçalho SCH[8] e um preâmbulo. Eles serão enviados pela estação base em cada canal que seja possível transmitir sem causar interferências. Quando um CPE é ativado, ele irá detectar o espectro, descobrir quais canais estão disponíveis e receberá todas as informações necessárias para se juntar à célula de uma estação base. Para tanto, dois tipos diferentes de medição do espectro serão feitas pelo CPE: *in-band* e *out-of-band*.

[6]CPE - Customer-Provided Equipment

[7]EIRP - *effective isotropic radiated power*, potência equivalente a um sinal transmitido por um radiador isotrópico (antena omnidirecional). Normalmente, o EIRP é igual ao produto da potência do transmissor pelo ganho da antena [IEEE, 2003a].

[8]SCH - *Superframe Controle Header*.

Tabela 2.1. Especificações de capacidade e cobertura do protocolo IEEE 802.22.

Parâmetro	Valor
Frequência de Rádio por Canal	6 MHz
Média da eficiência do espectro	3 Bit/s/Hz
Capacidade do canal	18 Mbit/s
Capacidade por assinante (envio)	1.5 Mbit/s
Capacidade por assinante (recebimento)	384 Kbit/s
Relação entre Taxa Envio/Taxa Recebimento	3.9
Número de assinantes por canal	600
Número mínimo de assinantes	90
Número potencial de assinantes	1800
Número hipotético de pessoas por domicílio	2.5 Pessoas
Número total de pessoas por área de cobertura	4500 Pessoas
EIRP da Estação Base	98.3 Watts
Raio de cobertura	30.7 Km
Densidade mínima de população abrangida	1.5 Pessoas/Km^2

A medição *in-band* consiste na detecção do próprio canal que está sendo utilizado pela estação base e pelo CPE. Enquanto a medição *out-of-band* consistirá na detecção dos demais canais [IEEE, 2011].

Independentemente da detecção ser *in-band* ou *out-of-band*, a camada MAC fará dois tipos de sensoriamento: sensoriamento rápido (*fast sensing*) e sensoriamento refinado (*fine sensing*). O sensoriamento rápido consiste na detecção do espectro a taxa de 1ms por canal. Essa detecção é realizada pelo CPE e pela estação base, que reunirá todas as informações e irá decidir se há algo novo a ser feito.

O *fine sensing* leva cerca de 25ms ou mais por canal e é utilizado com base no resultado do mecanismo de detecção do *fast sensing*. Esses mecanismos de detecção são utilizados principalmente para identificar se existe uma transmissão acontecendo, e se há necessidade de evitar a interferência com essa transmissão [IEEE, 2011].

2.3.2 OS-MAC

O protocolo OS-MAC utiliza períodos (janelas) de contenção pré-determinados para realizar a coordenação do acesso ao espectro entre os usuários RC e fazer o intercâmbio de informações de controle. No entanto, dentro de cada janela, o acesso ao espectro é aleatório e, portanto, este é um protocolo híbrido, conforme classificação proposta em [Cormio & Chowdhury, 2009].

As bandas do espectro utilizado para comunicação de dados são consideradas não-sobrepostas e um canal de controle comum (CCC) separado é assumido para a

troca de pacotes de controle entre usuários em diferentes faixas. Ele usa um único rádio que precisa mudar entre a banda de transmissão de dados dados e a transmissão de pacotes de controle no CCC. O funcionamento básico do protocolo é constituído das seguintes fases [Hamdaoui & Shin, 2008]:

- **Fase de Inicialização Rede:** nessa etapa os usuários RC formam grupos (*clusters*), de tal forma que todos os membros do mesmo grupo são nós que desejam comunicar uns com os outros. O usuário tem uma nova opção de formar seu próprio grupo ou juntar-se a um dos grupos existentes. Durante toda esta fase de agrupamento, o usuário RC mantém seu rádio sintonizado para o CCC. Em um dado momento, somente um usuário RC, denominado delegado, está ativo no *cluster*.

- **Fase de Inicialização da Sessão:** nessa etapa, o delegado ativo escolhe uma banda do espectro para ele e para todos os membros do grupo comunicarem.

- **Fase de Comunicação de Dados:** os membros do grupo usam a função de coordenação distribuída (DCF) do padrão IEEE 802.11 para acessar a banda do espectro. Ao mesmo tempo, o delegado ativo monitora o CCC para coletar informações sobre o ambiente do espectro. Em seguida, informa aos membros do seu grupo sobre a mudança na faixa de espectro, se necessário.

- **Fase de Atualização:** cada delegado de *cluster* agora envia as informações de tráfego de seu próprio *cluster* para os outros delegados por meio do CCC, e retorna a banda de espectro utilizada no final da transferência.

- **Fase de Seleção:** ao obter as estatísticas de utilização do espectro dos grupos vizinhos, o delegado de *cluster* pode iniciar a mudança do espectro utilizado no *cluster*. Isto é feito usando uma período de espera menor entre pacotes consecutivos, de forma que o delegado ganhe a disputa e transmita a escolha do novo espectro com maior prioridade.

- **Fase de Delegação:** o papel do delegado é agora transferido para outro usuário RC no mesmo *cluster* para o próximo ciclo de operação do protocolo.

Todas essas fases ocorrem sequencialmente e têm uma duração determinada pela janela de seus respectivos temporizadores. Mais ainda, esses prazos são flexíveis, e podem ser escolhidos de modo que cada usuário RC no *cluster* possa acessar o espectro de uma forma justa.

Porém o protocolo OS-MAC tem alguns inconvenientes. Um deles é que a participação dos usuários do RC nos *clusters* é baseado no pressuposto de que cada usuário já sabe a qual *cluster* deve se juntar. Como os grupos são formados baseados na comunicação entre os membros de cada grupo, é inviável fazer isso sem a troca de informações detalhadas do *cluster*.

Outro inconveniente, é que o RC delegado não coordena os outros *clusters* para a detecção eficiente do espectro, já que cada *cluster* funciona de forma independente, sem cumprir períodos de silêncio.

Além dos inconvenientes acima, o protocolo não leva em consideração a proteção aos usuários primários, por meio da adaptação dos parâmetros de transmissão, como por exemplo o controle de potência.

2.3.3 CSMA MAC

O CSMA MAC é um protocolo de acesso ao meio que usa um único transceptor e uma sinalização dentro da banda (*in-band*), ou seja transmissão de dados e sinalização de controle no mesmo canal. Este protocolo usa um esquema de transmissão adaptativa em que é possível permitir a um usuário secundário (usuário RC) transmitir, mesmo quando um usuário primário está transmitindo. Assim a convivência entre os usuários do RC e usuários primários é assegurada por meio da adaptação da potência e da taxa de transmissão da RRC [Lien et al., 2008].

Nesse protocolo, o RC e as estações dos usuários primários são separados, embora possam ter áreas de cobertura sobrepostas. Os usuários RC e os usuários primários podem estabelecer conexões diretas (*single-hop*) com as suas respectivas estações. As transmissões simultâneas dos usuários do RC e usuários primários que por ventura forem detectadas, devem levar em consideração a interferência causada a esses últimos, respeitando-se um limiar predefinido.

Sendo assim, a rede primária segue o protocolo CSMA clássico, de forma que o usuário primário encarrega-se de detectar a portadora por um período de tempo τ_p antes de enviar um quadro de solicitação de envio (RTS) para a sua estação. Enquanto que a estação primária pode responder com um quadro de controle (CTS) avisando que está pronta para receber o quadro de dados.

Porém, os usuários RC tem um tempo de detecção de portadora τ_s tal que ($\tau_s \gg \tau_p$), de modo que a prioridade de acesso ao espectro é dada aos usuários primários. Com base na distância dos usuários secundários à estação rádio base cognitiva, e de acordo com a potência do ruído, a estação rádio base decide os parâmetros de transmissão (potência de transmissão e taxa de dados) [Lien et al., 2008]. O usuário RC tem

permissão para enviar apenas um pacote em uma rodada dessa negociação, a fim de minimizar o risco de interferência com outros usuários primários.

Segundo [Lien et al., 2008] os resultados obtidos com esse protocolo mostram que sua taxa de transferência é 36% e 100% melhor quando comparado com a CSMA convencional e com as operações de RC convencionais, respectivamente.

2.3.4 MOAR

O MOAR (*Multi-channel Opportunistic Auto-rate*) é um protocolo MAC para múltiplos canais e taxas de transferência que permitem trabalhar com redes sem fio *ad hoc* IEEE 802.11. A idéia principal da MOAR é explorar a natureza variável do canal sem fio de forma distribuída e oportunista por meio de salto nas frequências dos canais, permitindo a busca de um canal de melhor qualidade [Kanodia et al., 2004].

Quando as métricas de qualidade do canal estão baixas em determinada frequência, o MOAR permite que o receptor e o transmissor possam negociar uma nova faixa de frequência em que se possa transmitir com melhor qualidade. Uma vez que o padrão IEEE 802.11 apresenta diferentes canais de frequência, há uma alta probabilidade de que o nó saltando de frequência irá encontrar melhores condições de canal em um dos outros canais de frequência.

Em teoria pode-se buscar indefinidamente pelo canal de frequência com a maior taxa de transmissão possível. No entanto, em sistemas reais, onde as informações de estado de canal não estão disponíveis, *a priori*, cada salto de frequência incorre em decisão que traz uma sobrecarga de processamento devido a medição da qualidade canal.

Assim os ganhos de rendimento conquistados por meio de saltos oportunistas de frequência podem diminuir a cada salto. Além disso, quando as condições do canal médio são pobres, a probabilidade de encontrar um canal com maior taxa de dados é muito baixa.

Assim, a fim de maximizar o ganho na taxa de transferência é fundamental equilibrar as trocas entre o ganho adicional trazido pelo salto de frequência e o tempo gasto para medir a qualidade dos canais. Por conseguinte, deve-se limitar o número de vezes que um nó salta em busca de um melhor canal. Finalmente, [Kanodia et al., 2004] mostra por meio de simulações que o MOAR, supera o estado-da-arte de protocolos que trabalham com múltiplas taxas de transferência, em 20% a 25% de desempenho.

A exemplo do MOAR, a maioria dos protocolos MAC-RC visam apenas desempenho em métricas como taxas de transferência, taxa de pacotes entregue, atraso fim a fim e não tem como preocupação a questão do consumo de energia. Sendo assim,

a seguir são apresentados os protocolos MAC específicos para RSSF, no intuito de investigar como a questão da eficiência energética é tratada nesses protocolos.

2.4 Protocolos MAC para RSSF

Os protocolos de controle de acesso ao meio para RSSFs, têm como principal objetivo reduzir o consumo de energia. Em geral, esses protocolos são desenvolvidos para aplicações específicas e buscam um equilíbrio entre as métricas de latência, vazão e consumo de energia [Correia et al., 2007].

Ademais, os protocolos MAC para RSSFs são diferentes dos empregados em redes *ad hoc* sem fio. As características da aplicação influenciam os requisitos da subcamada MAC, de forma que os protocolos são especializados para certos tipos de redes [Correia et al., 2007].

Nesta seção são apresentados os principais protocolos para RSSF que levam em consideração a implementação de técnicas de acesso dinâmico ao espectro, a fim de apresentar o estado da arte e os principais conceitos relacionados. Estes protocolos foram pesquisados para auxiliar na escolha do protocolo que será adaptado nesta dissertação, levando em consideração os aspectos de cognição e reconfigurabilidade.

2.4.1 S-MAC

O S-MAC (*Sensor-MAC*), é um protocolo MAC explicitamente concebido para redes de sensores sem fio, cuja a redução do consumo de energia é o principal objetivo. Além disso é destinado a aplicações orientadas a eventos, insensíveis a latência e com baixa taxa de envio de mensagens.

O S-MAC atinge boa escalabilidade e evita colisões, utilizando um método de acesso ao meio com as seguintes características [Ye et al., 2004]:

- Alocação dinâmica de canal;
- Sincronização para coordenar os modos de operação de rádio(*idle/sleep*);
- Usado em RSSF orientadas a eventos, com coleta periódica de dados, insensível a latência da rede e com baixa taxa de envio de mensagens.
- Uso eficiente de energia.

Para atingir a eficiência energética, foram identificadas as principais fontes de desperdício de energia [Ye et al., 2004], e propostos mecanismos para contornar estes problemas:

- Colisão: para evitar as colisões, o problema do terminal escondido e o problema da estação exposta, o S-MAC faz reserva de canal usando quadros de controle *RTS-CTS* e um vetor de alocação de rede (NAV - *Network Allocation Vector*) para detecção de portadora virtual, como no IEEE 802.11. Caso ocorra colisão, o S-MAC utiliza o algoritmo BEB (*Binary Exponential Backoff*).

- *Overhearing*: coloca o nó em modo *sleep* quando verifica que o pacote não é destinado a ele.

- *Overhead*: os pacotes de controle são usados para reserva do canal de comunicação, reconhecimento de pacotes de dados e sincronização. O S-MAC reduz os pacotes de controle para diminuir o *overhead*, pois estes pacotes de controle aumentam o tráfego da rede.

- *Idle listening*: o nó fica ouvindo o meio de transmissão mesmo quando não existe tráfego na rede. O S-MAC utiliza um ciclo de operação reduzido com tempos fixos para os nós ficarem acordados e dormindo. Visando a economia de energia, o tempo de atividade é menor que o tempo de repouso, cerca de 10%.

Com relação ao ciclo de operação, no período em que está acordado *(listen)*, o nó envia e recebe mensagens. No período em que o nó está dormindo (*sleep*) as mensagens são armazenadas e transferidas em rajada no início do próximo período em que o nó acordar.

Todos os nós tem a liberdade de alternar entre o período *listen/sleep*. Porém, para reduzir o *overhead* os nós vizinhos são sincronizados. Um nó informa para seus vizinhos o tempo de seu período de *listen*, deste modo assume-se que todos os vizinhos podem comunicar-se até mesmo com tempos diferentes [Ye et al., 2004].

Apesar de empregar alocação dinâmica de canal e um método de controle de acesso aleatório (CSMA/CA), o S-MAC utiliza sincronização para coordenar os modos de operação do rádio. A sincronização dos nós é feita por meio de *clusters* virtuais formados entre nós vizinhos que possuem o mesmo ciclo de operação. Para manter os nós sincronizados, são enviados quadros de sincronização em difusão para todos os nós da vizinhança. Os nós que estão na fronteira de dois *clusters* seguem as duas escalas de sincronização [Ye et al., 2004].

A comunicação dentro de cada *clusters* segue um ciclo de operação com períodos de atividade e repouso. No intervalo de atividade os nós escutam o canal de modo a detectar transmissões de outros nós. Caso detectem transmissões, e essas sejam destinadas especificamente àquele nó, um quadro de dados é recebido, caso contrário

o nó entra em estado de repouso. Durante o intervalo de repouso o nó desliga o rádio para economizar energia [Ye et al., 2004].

A eficiência energética é alcançada pelo uso de ciclo de operação reduzido, negociado em detrimento da latência. Quanto maior o período de repouso maior será a latência na comunicação.

O protocolo S-MAC emprega uma comunicação multissaltos e considera os requisitos de uma rede densa e homogênea para ser eficiente em energia e permitir a adaptação dos nós às condições da rede. Para tal, o S-MAC é programado para se autoconfigurar à variabilidade do ambiente na inserção ou remoção de nós, bem como a detecção de novos agrupamentos dentro do seu alcance de rádio.

Em resumo, o protocolo S-MAC, é um protocolo cujo ciclo de operação é reduzido, usado em RSSF dirigidas a eventos, com coleta periódica, insensível a latência da rede e com baixa taxa de envio de mensagens. A principal vantagem desse protocolo é que prolonga consideravelmente o tempo de vida da rede. A principal desvantagem é a latência ocasionada pelo período em que o nó fica dormindo.

2.4.2 B-MAC

O B-MAC (Berkeley-MAC) é um protocolo simples, composto por um núcleo pequeno que concentra as funcionalidades modularizadas. Com isso o B-MAC consegue suportar uma grande variedade de aplicações. Polastre et al. [2004] mostraram que essa arquitetura é eficaz e útil para RSSFs.

O protocolo B-MAC foi projetado para aplicações dirigidas a eventos em RSSFs. Seus principais objetivos são a eficiência energética, evitar colisões na rede, possuir menor tamanho de código e fornecer interfaces para permitir reconfiguração de parâmetros pela aplicação.

O B-MAC possui as seguintes características, essenciais para um protocolo MAC:

- Operação em baixa potência e redução do consumo de energia, aumentando o tempo de vida da rede.
- Prevenção de colisões, evitando retransmissões e economizando energia através de períodos de *backoff*.
- Operação simples e intuitiva: facilita a personalização do protocolo para aplicações específicas.
- Ocupa pouca memória, disponibilizando mais espaço da memória do nó sensor para aplicação.

- Tolerância a mudanças.
- Escalável para grande número de nós.

O Protocolo B-MAC implementa controle *carrier sense multiple access* (CSMA), avaliando o canal livre, atraso de transmissão (*backoff*) e quadros de confirmação, sendo possível desabilitar e regular parâmetros destes mecanismos pela aplicação.

O B-MAC utiliza um rádio monocanal e alocação dinâmica com coordenação aleatória do canal. Esse protocolo não utiliza quadros de sincronização ou de reserva de banda no intuito de reduzir o tamanho de código [Polastre et al., 2004].

Para evitar colisão de quadros na rede, o B-MAC utiliza uma heurística que verifica se existe atividade no canal (livre ou ocupado). Essa heurística é conhecida como CCA (*Clear Channel Assessment*), que amostra a força do sinal recebido do meio de transmissão quando não existe tráfego na rede. A partir dessas amostras é determinado o valor máximo de ruído do meio (ruído base) [Polastre et al., 2004].

Dessa forma, quando um nó deseja transmitir ele amostra o nível de sinal do meio e o compara com o ruído base. Caso o sinal amostrado seja maior que o ruído base, é presumido que o canal está ocupado e o nó não poderá transmitir. Então, a transmissão será atrasada por meio de um algoritmo de *backoff* aleatório. Se o sinal amostrado está próximo do ruído base então o meio está livre para a transmissão. O CCA provê um método de detecção de portadora antes da transmissão pelo nó, não necessitando de sincronização entre os nós vizinhos ou reserva de canal [Polastre et al., 2004].

O B-MAC utiliza ciclo de operação para minimizar os períodos de *idle listening*. O nó escuta o canal periodicamente para coletar as amostras do nível de sinal e verificar se existe alguma transmissão em progresso. Se alguma transmissão é identificada, o nó entra em modo de recepção e pesquisa um preâmbulo no sinal recebido. Para garantir a escuta dos quadros, os preâmbulos enviados são ajustados de maneira que seu tempo de transmissão seja maior que o intervalo de tempo de repouso dos nós. Dessa forma, o tempo de vida da rede é calculado em função do intervalo de amostragem, do tamanho do preâmbulo e do tempo de repouso do nó [Polastre et al., 2004].

Outra característica do B-MAC é a adaptabilidade de parâmetros pelos serviços da aplicação em função das condições do enlace, do número de vizinhos e do tráfego da rede [Polastre et al., 2004]. Essa capacidade de adaptação do B-MAC permite simplicidade e redução do tamanho do código, além de suportar vários tipos de tráfego e aumentar o tempo de vida da rede.

Os serviços da aplicação podem controlar adaptativamente os parâmetros do B-MAC habilitando ou desabilitando o CCA, configurando períodos de *backoff* para ne-

gociar vazão e justiça. O protocolo também permite configurar um serviço confiável de enlace por meio de quadros de controle de mensagem recebida (ACK), o serviço pode escolher entre retransmitir quadros perdidos e reconfigurar o tamanho do preâmbulo. O tamanho do preâmbulo está relacionado com o tempo de repouso do nó, de maneira que seja possível que o nó detecte o preâmbulo a cada ciclo de operação. No B-MAC o tamanho do preâmbulo é calculado dinamicamente em função do tempo de repouso, de modo a minimizar o custo da transmissão do preâmbulo.

Em suma, o B-MAC emprega mecanismos para reduzir o consumo de energia, maximizar a vazão e promover a justiça dos mecanismos de acesso ao meio da rede. Porém, nenhum requisito de qualidade de serviço é considerado no protocolo B-MAC.

2.4.3 T-MAC

O protocolo *Time-out-MAC* (T-MAC) é baseado no mecanismo de contenção do IEEE 802.11 para o controle de acesso ao meio em RSSFs. O T-MAC foi desenvolvido para aplicações orientadas a eventos que possuam baixa taxa de entrega de mensagens e sejam insensíveis a latência. Trabalha com transmissão contínua ou periódica de dados. A exemplo do S-MAC, tem com principal objetivo a eficiência de energia considerando as limitações de hardware do nós sensor e as trocas de mensagens entre os nós vizinhos [van Dam & Langendoen, 2003].

Tal qual o S-MAC o ciclo de operação do T-MAC é reduzido, possuindo tempos de atividade (*listen*) e de repouso (*sleep*) variáveis, adaptativos à carga da rede. A variação dinâmica do tempo ativo é obtida pela implementação de um temporizador que desliga o rádio do nó ao verificar que não existe transmissão durante um intervalo de tempo.

Outra característica do T-MAC é que ele utiliza alocação dinâmica de canal e os nós identificam as transmissões usando métodos de detecção das portadoras física e virtual. A comunicação é realizada por meio de um rádio monocanal que permite manipulação de seus modos de operação. Da mesma forma que no S-MAC, o protocolo T-MAC utiliza um quadro de sincronização para formar *clusters* virtuais, sendo enviado periodicamente em *broadcast* para os vizinhos.

Além disso, o T-MAC utiliza um ciclo de operação dinâmico que se adapta à variação da carga da rede. O período de atividade é controlado por um temporizador T_A (*time-out activity*), que ao expirar coloca o rádio em repouso. As mensagens recebidas durante o tempo de repouso são armazenadas e transferidas em rajadas de tamanho variável, no início do tempo ativo, conforme mostrado na figura 2.5. A técnica de notificação é por escuta síncrona do canal, que ocorre durante o período ativo, quando

o nó pode transmitir e receber dados. Os nós utilizam quadros de confirmação e de reserva de canal *RTS-CTS* para evitar colisões e incrementar a probabilidade da recepção de quadros. Se um quadro de *RTS* ou *CTS* é recebido, o tempo T_A é renovado por um período suficiente para receber um quadro de dados. O mecanismo de *backoff* é calculado em função do tráfego instantâneo da rede.

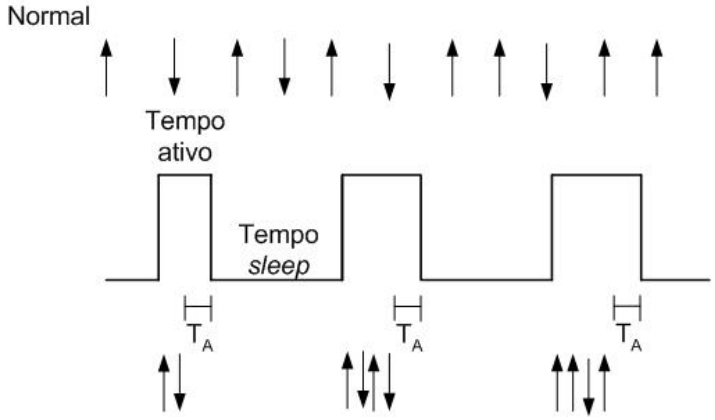

Figura 2.5. Ciclo adaptativo do protocolo T-MAC.

Os protocolos que empregam ciclos de operação, como o T-MAC, estão sujeitos ao problema de dormir cedo (*early sleeping*),como apresentado na figura 2.6. Este problema ocorre quando um nó dorme enquanto um outro nó ainda tem mensagem para ele. Este problema pode ser resolvido de duas formas:

- Na primeira, um nó ao escutar um pacote *CTS* destinado a outro nó envia imediatamente aos seus vizinhos um pacote designado de *FRTS* (*Future RTS*), que evita que os nós vizinhos entrem em modo de repouso.

- A outra forma é usar um esquema que prioriza o esvaziamento do *buffer* quando ele estiver perto de sua capacidade limite. Um nó ao receber um *RTS* ao invés de responder com um *CTS*, transmite todas as mensagens armazenadas em seu *buffer* para o nó de destino.

O protocolo T-MAC emprega autoconfiguração para identificar seus vizinhos durante a inicialização da rede e para permitir a inserção ou remoção de nós. Esse protocolo também se adapta à variação de carga na rede reduzindo o período ativo.

O T-MAC consegue ser mais eficiente em energia que o S-MAC, mas é extremamente limitado em largura de banda não sendo recomendado para aplicações sensíveis a este requisito. Isso se deve ao uso de temporizador para reduzir o período de atividade

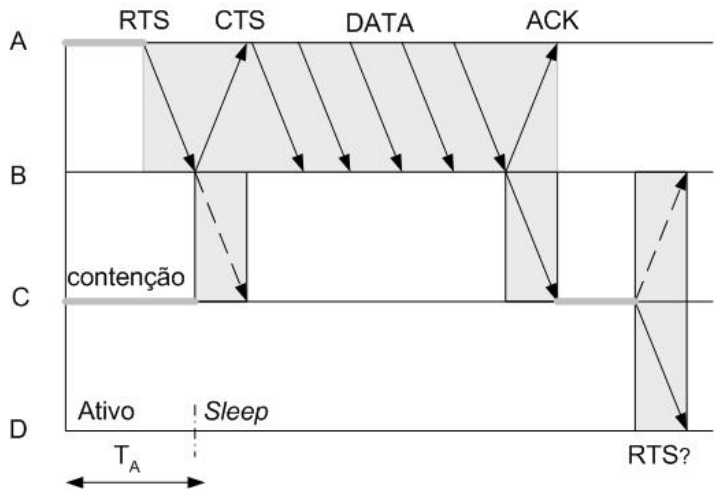

Figura 2.6. O nó *D* dorme antes de *C* enviar um RTS.

do nó, quando não existe atividade na rede, limita a largura de banda e a transferência de grandes quantidades de dados.

2.5 RSSFC - Redes de Sensores Sem Fio Cognitivas

2.5.1 Definição

Em geral, uma rede de sensores fio cognitiva (RSSFC) pode ser definida como uma rede distribuída de nós sensores sem fio baseadas em rádio cognitivos. Essas RSSFC detectam sinais de eventos, colaboram entre entre si comunicando as suas leituras de forma dinâmica, em faixas de frequências disponíveis, por meio de múltiplos saltos, para satisfazer os requisitos de uma aplicação específica [Akan et al., 2009].

As RSSFC possuem desafios semelhantes aos das RRC e as estratégias envolvidas nas soluções são similares às das RRC. Em uma RSSF, cada nó envia e recebe dados ou está em estado ocioso. No entanto, em um RSSFC, haveria outro estado chamado de estado de sensoriamento no qual os nós sensores fariam uma varredura do espectro a fim de encontrar oportunidades ou faixas de frequências livres em que este nós pudessem transmitir.

A Figura 2.7 mostra os diferentes estados para ambas as redes. Entre as várias tarefas para cada nó sensor, a transmissão e recepção de dados são tarefas que mais consomem energia. Pode-se observar que além dos três estados convencionais de uma RSSF, temos o estado de sensoriamento do espectro, em que as oportunidades de transmissão em frequências primárias são identificadas. As tarefas de detecção do espectro

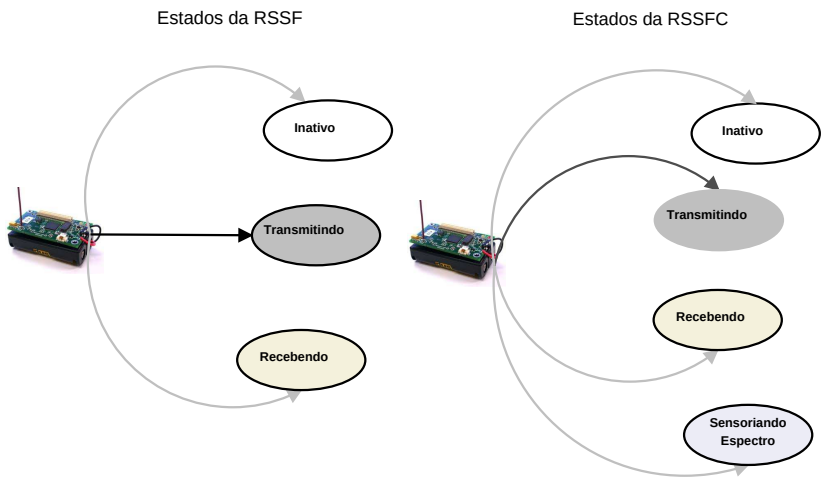

Figura 2.7. Estados de uma RSSFC.

em uma RSSFC podem ser realizadas por estratégias distribuída ou centralizada.

Em uma estratégia distribuída, cada sensor concorre com outros sensores pelo acesso ao espectro disponível [Cavalcanti et al., 2008]. Assim, cada sensor deve ter a capacidade de sensorear o canal por completo, e determinar uma política para maximizar seus benefícios, tais como o número de transmissões ao longo do tempo.

Em um esquema centralizado, as oportunidades de espectro são detectados por uma única entidade chamada coordenador da rede [Gao et al., 2007]. A Figura 2.8 mostra a comparação entre um esquema de sensoriamento distribuído e um esquema centralizado. No primeiro esquema (distribuído), a detecção do espectro é feita por cada nó, enquanto que no segundo (centralizado) uma estação coordena a detecção do espectro.

Uma vez introduzido o conceito de RSSFC é necessário apresentar a arquitetura, o arcabouço de gerenciamento e alguns trabalhos relacionados na área de redes de sensores sem fio que utilizam técnicas de gerenciamento dinâmico do espectro.

2.5.2 Arquitetura de uma RSSFC

A arquitetura de comunicação típica de um RSSFC é ilustrado na figura 2.9. Dependendo da disponibilidade de espectro, os nós sensores transmitem suas leituras de forma oportunista para os próximos saltos e, finalmente, para o nó sink. O coletor pode também ser equipado com a capacidade cognitiva de rádio (ou seja, nó sink de rádio cognitivos). Além das leituras evento, os sensores podem trocar informações de controle para a formação de grupos, a alocação do espectro, e a determinação da rota de espectro ciente do *handoff* de frequências em função da topologia específica.

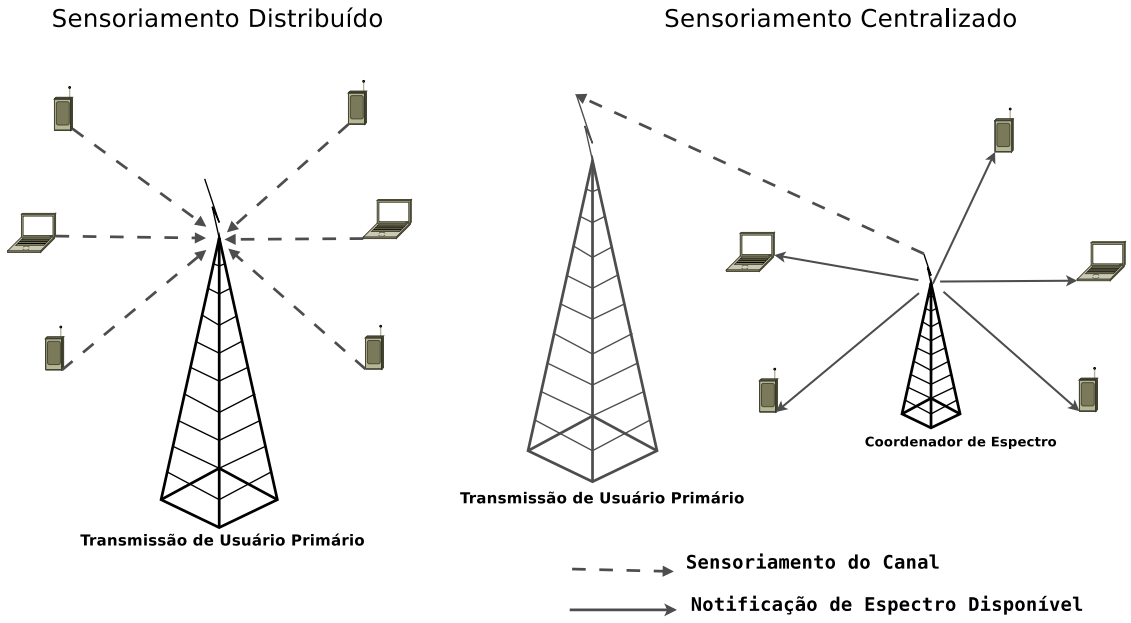

Figura 2.8. Sensoriamento centralizado e distribuído em uma RSSFC.

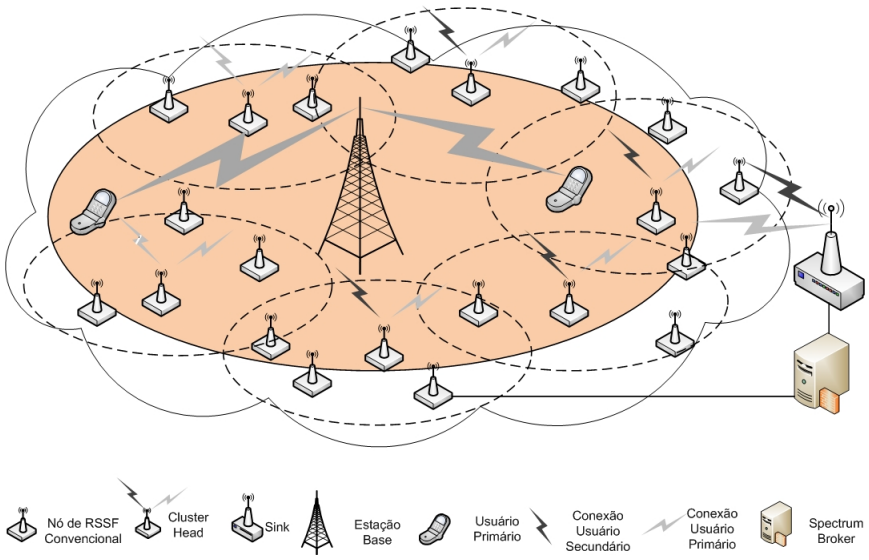

Figura 2.9. Arquitetura de uma RSSFC [Akan et al., 2009].

Observe que a figura 2.9 ilustra a conexão de usuários secundários e usuários primários, bem como a coordenação dos nós sensores convencionais por *clusters heads*.

2.5.3 Gerenciamento Dinâmico do Espectro em RSSFC

O arcabouço de gerenciamento do espectro em RSSFC é composto de três etapas principais: sensoriamento, decisão e mobilidade do espectro. Em suma o que distingue as funcionalidades deste arcabouço com o gerenciamento do espectro em RRC convencionais é a preocupação adicional com o consumo de energia. A figura 2.10 ilustra esse arcabouço e a interação entre as camadas para o seu gerenciamento. Nas subseções a seguir, cada uma dessas funcionalidades apresentadas na figura são melhor detalhadas.

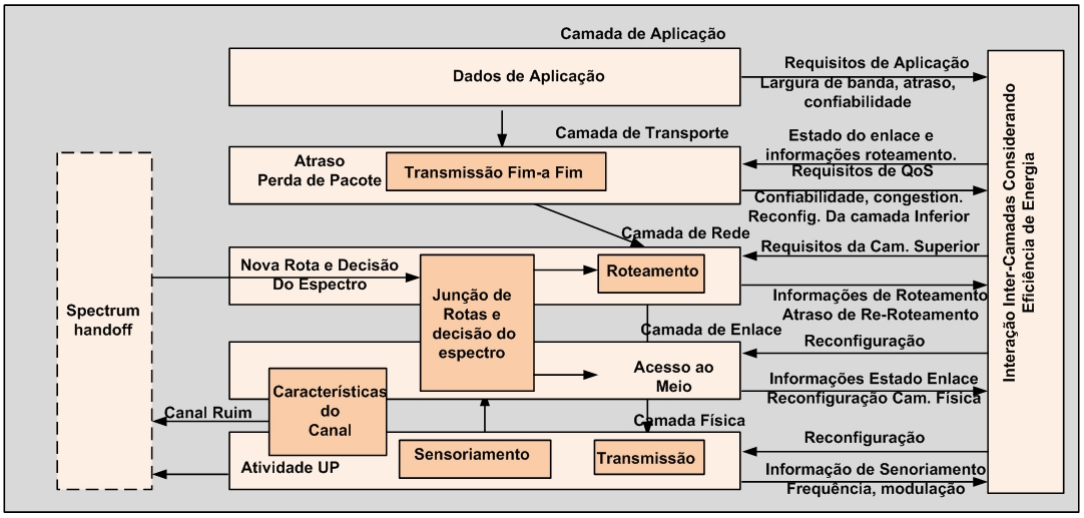

Figura 2.10. Arcabouço de Gerenciamento do Espectro em RSSFC - Interação entre as Camadas [Akan et al., 2009]

2.5.3.1 Sensoriamento do Espectro

O sensoriamento do espectro é uma das funcionalidades que distinguem as RSSFC das RSSFs tradicionais, uma vez que seus nós devem coletar informações de utilização do espectro antes da transmissão. No entanto, apesar das vantagens do acesso oportunista ao espectro (mais largura de banda, menor taxa de erro devido à capacidade de mudar para o melhor canal, menor atraso contenção), o consumo de energia adicional imposto pelo sensoriamento do espectro e a distribuição das informações coletadas deve ser levados em consideração.

O sensoriamento em RSSFC pode ser baseado em uma arquitetura distribuída baseada em redes de sensores totalmente dedicados para executar o sensoriamento do espectro, na qual são distinguidos dois tipos de redes: a rede de sensores responsáveis pelo sensoriamento do espectro e uma ou mais redes sem fio operacionais [Shankar et al., 2005].

A rede de detecção seria composta por um conjunto de sensores utilizados na área-alvo em que o sensoriamento do espectro deve ser feito (contínua ou periodicamente) e comunicar os resultados (que podem ser submetidos a algum processamento, como fusão de dados, etc) para um nó sorvedouro conhecido. O nó pode processar os dados recolhidos e disponibilizar as informações sobre a ocupação do espectro na área sensoriada para todas as redes operacionais. As redes operacionais, por outro lado, são responsáveis pela transmissão de dados e uso oportunista do espectro, e aceitaria as informações sobre o mapa de ocupação do espectro para determinar o canal a utilizar, quando utilizar e por quanto tempo.

Dado que o sensoriamento do espectro é um processo repetitivo, que consome energia extra da bateria sensores de potência, a aplicação de sensoriamento em todos os nós em uma RSSF pode não ser eficiente em termos de consumo de energia.

Assim, deve ser considerado o impacto do tempo de sensoriamento do espectro no consumo de energia, uma vez que todo o tráfego está suspenso enquanto a detecção do espectro é feita.

Na literatura há vários métodos de detecção, que serão apresentados a seguir em termos de como eles podem ser aplicados a RSSFCs [Akan et al., 2009]:

- Filtro Combinado: este é um método ótimo de detecção do espectro baseado no ruído gaussiano, no entanto, requer um conhecimento a priori sobre a transmissão do usuário primário e de hardware adicional para os nós da RSSFC sincronizarem com o usuário primário.

- Detecção de Energia: se a energia medida em um canal estiver abaixo de um valor limiar, o canal é considerado disponível. Sua simplicidade e baixa exigência de processamento de sinal torna o método muito atraente para RSSFCs. Entretanto, a detecção de energia requer mais tempo de medição, o que implica em um maior consumo de energia.

- Detecção de Características: esse método pode ser usado quando certas características da transmissão do usuário (frequência portadora e prefixos cíclicos) são conhecidos. Ela explora a correlação do espectro do sinal do usuário primário e, portanto, é muito robusto contra as variações do ruído. No entanto, tem complexidade muito alta, portanto, pode não ser adequado para RSSFCs.

- Temperatura de Interferência: baseado no cálculo de quanta interferência um nó pode causar no receptor do usuário primário, ajusta-se sua potência de transmissão tal que a sua interferência mais o limite inferior de ruído não ultrapasse um determinado nível de interferência. Este método requer a priori informações

sobre a localização do usuário primário, e é computacionalmente muito complexo para um nó RSSFC.

Esses métodos de sensoriamento de espectro muitas das vezes não são feitos sob medida para nós sensores. Portanto, há várias questões em aberto sobre o sensoriamento do espectro em RSSFC:

- Técnicas híbridas de sensoriamento: técnicas de sensoriamento, explorando as vantagens principais de métodos de sensoriamento de espectro levando em consideração o compromisso entre precisão e complexidade, podem ser investigados.
- Sensoriamento Cooperativo: mecanismo distribuídos de sensoriamento do espectro cooperativo precisam encontrar um equilíbrio entre precisão e tempo de sensoriamento
- Sensoriamento com base em estatísticas de colaboração de usuários primários: fazer uso de estatísticas de utilização do canal pelos usuários primários, devem ser exploradas para melhorar a acurácia da detecção.

2.5.3.2 Decisão Espectro

Os nós das RSSFC devem analisar os dados de sensoriamento e tomar uma decisão sobre o canal e os parâmetros de transmissão (por exemplo, a potência de transmissão e modulação). Os métodos de decisão espectro proposto para as redes de rádio cognitivo consideram o consumo de energia como uma questão secundária. Capacidades de processamento e quantidade de pacotes de controle extra para transmitir são quase sempre ignoradas. Claramente, estas definições não combinam com as características das RSSFC devido às limitações inerentes aos nós sensores [Akan et al., 2009].

Por outro lado, os resultados do sensoriamento do espectro serão semelhantes em um determinado local [Zhao et al., 2005]. Ao tentar acessar o canal, dependendo apenas dos seus resultados locais, uma decisão individual do espectro aumentaria a probabilidade de colisão.

Além disso, uma vez que os nós executaram o mesmo algoritmo, quando ocorre uma colisão, todos eles tentam trocar de canal, deixando o canal anterior vazio e colidindo novamente no novo canal.

Portanto, a decisão do espectro em RSSFCs deve ser coordenada para aumentar a utilização global e maximizar a eficiência de energia.Além disso, um mecanismo decisão de espectro para RSSFCs devem ter baixa complexidade.

A decisão do espectro pode ter centralizada ou distribuída. Um nó central pode ser designado como entidade centralizadora para decidir a faixo do espectro ideal em

toda a rede, que impõe também o tráfego adicional, resultando em consumo excessivo de energia.

No entanto, na decisão distribuída do espectro, os nós compartilham os resultados de sensoriamento e decisão apenas com seus vizinhos imediatos ou dentro de pequenos grupos.

Esta abordagem leva a utilização sub ótima, que ainda pode estar perto da solução global ótima [Zhao et al., 2005]. Ademais, é mais simples de implementar e acarreta menos *overhead* de comunicação e energia do que uma abordagem centralizada.

Claramente, há muitas questões de investigação aberto para o desenvolvimento de métodos de decisão do espectro decisão para RSSFCs:

- Parâmetros de decisão do espectro: parâmetros a serem utilizados na decisão do espectro (por exemplo, a relação sinal-ruído, a capacidade do canal, os atrasos e os tempos de exploração de pus) devem ser explorado, e novos algoritmos que geram decisões sobre o espectro ideal devem ser projetados.

- Canal de controle de distribuído: usando um canal de controle comum em toda a rede de decisão geralmente não é viável, no entanto, considerando uma determinada região temos possibilidade de encontrar um canal de coordenação para um subconjunto. Em nosso trabalho optaremos por um método de decisão do espectro inicialmente usando um canal comum para toda rede e depois o modelo deve ser extrapolado para mais de um canal de controle.

2.5.3.3 Mobilidade *(Handoff)* do Espectro

Quando um usuário começa a usar um canal disponível, os nós da RSSFC devem verificar esta atividade dentro de um determinado período de tempo através de métodos de sensoriamento de espectro. Em seguida, eles imediatamente se deslocam para outro canal disponível escolhido por um mecanismo eficaz de decisão do espectro de decisão.

Os nós também podem querer mudar de canal, se as condições do canal piorar, reduzindo o desempenho da comunicação. Esta funcionalidade fundamental do rádio cognitivo é chamado *handoff* do espectro. Em [Byun et al., 2008], um esquema de alocação de espectro central que tenta minimizar a entrega do espectro tem sido proposto para RSSFCs. No entanto, não considerar os desafios colocados pelas limitações inerentes à RSSFCs. Minimizando-se o efeito do *handoff* do espectro em várias camadas de comunicação, o gerenciamento dinâmico do espectro dinâmico,torna-se mais complexo.

Uma vez apresentadas as funcionalidades do arcabouço de gerenciamento do espectro em RSSFC, no capítulo seguinte será apresentado o processo metodológico usado para avaliar o protocolo de decisão do espectro proposto neste trabalho.

RSSFs. Algumas questões em aberto para continuar a ser explorados incluem a escalabilidade da detecção histórico e procedimentos de recuperação, bem como a coexistência entre os vários RSSFC.

2.5.4 Trabalhos Relacionados em RSSFC

Em Zhou et al. [2006] são apresentados experimentos que demonstraram como dispositivos eletrônicos que operam em 2,4 GHz podem causar interferência e até mesmo deixar inoperante uma RSSF baseada no padrão 802.15.4. Para evitar esse problema, os autores propuseram uma abordagem multicanal para as RSSF, já que os nós sensores atuais possuem rádios multi-frequências. Ainda foi proposto um *middleware* entre as camadas física e MAC para as RSSF de forma a suportar uma abordagem multicanal. Apesar disso, esse modelo apresenta somente aspectos de reconfigurabilidade multicanal e nenhum experimento ou simulação foi realizado.

No trabalho [Cavalcanti et al., 2008] é apresentado um projeto conceitual de RSSFs baseadas em RC que identifica as principais vantagens e desafios do uso da tecnologia de RC, bem como possíveis soluções para superar esses desafios. O mesmo artigo apresenta resultados de simulações, analisando o desempenho de uma RSSFC aplicada à automação e controle de aplicações residenciais e comerciais. Os resultados de simulação compararam o desempenho de uma RSSFC com um padrão ZigBee/802.15.4 para RSSF. De acordo com as análises e simulações, os resultados demonstraram as vantagens do sistema baseado em RC em termos de extensão, alcance e eficiência do protocolo.

Nas simulações feitas, foram utilizados o padrão ZigBee/802.15.4 disponível na plataforma OPNET[9] operando na banda de 2.4 GHz, e construído um modo RC estendido baseado na pilha de protocolo ZigBee/802.15.4 . Neste trabalho os autores utilizaram o mecanismo de acesso CSMA básico do 802.15.4 no modo *non-beacon*, acrescentado-se características de detecção e mudança de canal na presença de usuário primário.

Em [Ge et al., 2009] um método de decisão do espectro desenvolvido para redes sem fio é proposto. O método emprega informações estatísticas da presença de usuários primários na região monitorada, que são alimentadas a um classificador multi-

[9]Software comercial para análise de desempenho em redes de computadores e aplicações. Disponível em http://www.opnet.com/.

parâmetros baseado em AHP (*Analytic Hierarchy Process*) Saaty [2000]. O método considera a probabilidade da chegada de usuários primários durante o monitoramento, os requisitos da aplicação e a influência de múltiplos parâmetros de entrada (capacidade do canal, atraso, perda de pacotes e *jitter*).

Além disso, é proposto um mecanismo, baseado em entropia, que automaticamente determina os pesos dos parâmetros de entrada no método de decisão. Apesar de apresentar resultados satisfatórios nas simulações, a falta de caracterização real dos modelos de rádio, da topologia da rede e de cenários realísticos tornaram essa abordagem abstrata. Nenhuma referência aos problemas encontrados na diversidade de antenas, nos protocolos MAC ou na camada física foi localizada.

Recentemente, Kusy et al. [2011] propuseram uma arquitetura de nós sensores multi-rádio para aumentar a confiabilidade na comunicação. A plataforma IRIS e o padrão IEEE 802.15.4 foram modificados para operarem com dois rádios com frequências de 900 MHz e 2,4 GHz. Experimentos demonstraram a viabilidade da solução proposta, apresentando melhorias da taxa de entrega e da estabilidade do enlace ao custo de um incremento moderado no consumo de energia. Apesar disso, a abordagem restringiu-se a uma implementação mono canal, em que os rádios foram utilizados de forma independente, sem apresentar quaisquer funcionalidades de sensoriamento ou decisão do espectro.

Capítulo 3

Protocolo de Decisão do Espectro para RSSF

O protocolo de decisão do espectro para RSSF proposto neste trabalho consiste em uma adaptação cognitiva ao protocolo T-MAC [van Dam & Langendoen, 2003]. Essa adaptação é baseada em dois métodos de seleção dinâmica do melhor canal disponível:

- CogTMAC: realiza a escolha do melhor canal com base apenas na força do sinal recebido (RSSI);
- AhpTMAC: além de utilizar o RSSI, esse método faz a coleta dos parâmetros de SINR, ruído base e atraso fim a fim, para escolher o melhor canal com base no método de tomada de decisão AHP *Analytic Hierarchy Process* apresentado na seção 3.3.2.2.

Na solução proposta, além dos aspectos relacionados a decisão do espectro, foram consideradas funcionalidades de sensoriamento e mobilidade do espectro. Para tanto, alguns pressupostos foram assumidos na caracterização do problema. Primeiramente, assumiu-se que a rede é composta apenas por usuários secundários (não licenciados). Tais usuários, contam com nós sensores multi-rádios, operando nas frequências de 2.4 GHz e 900 MHz[1].

No rádio de 2.4 GHz o espectro é dividido em 11 canais sobrepostos[2], sendo que três canais são totalmente não interferentes, como no IEEE 802.11. Os rádios de 900

[1] O rádio de 900 MHz não foi utilizado na simulação, apenas o rádio 2.4 GHz IEEE 802.15.4 CC2420 [Texas Instruments, 2006].A utilização de dois rádios foi implementada para uso em trabalhos futuros em protocolos multi-rádio. No entanto, não compromete a avaliação da solução, uma vez que a unidade básica do protocolo de decisão é o canal e bastaria levar em consideração também os dois canais do rádio de 900 MHz na escolha do melhor canal.

[2] A sobreposição de canais se deve a interferência dos canais adjacentes.

MHz, são providos de dois canais de comunicação. A unidade fundamental considerada é um canal físico, associado a uma região do espectro ou tecnologia de rádio que deverá ser atribuído para comunicação de dois nós sensores vizinhos.

Assumiu-se também que os nós sensores são estáticos[3] e que se comunicam selecionando o mesmo canal. Cada usuário utiliza um transceptor *half-duplex*, e só pode transmitir ou receber em um canal de cada vez. Apenas um rádio pode ser invocado por um dispositivo multi-rádio ao mesmo tempo.

A qualidade do canal flutua e a perda média de percurso (*Average Path Loss*) no canal sem fio é estimada em função da distância entre dois nós. Em RSSF, onde a separação de nós é da ordem de dezenas de metros, o modelo de sombreamento *lognormal* é adequado para dar estimativas mais precisas para a perda média de percurso. Tal modelo é descrito pela equação 3.1 que retorna a perda de caminho em dB em função da distância entre dois nós, e é expressa como:

$$PL(d) = PL(d_0) + 10.\eta.log\left(\frac{d}{d_0}\right) + X_\sigma \tag{3.1}$$

onde $PL(d)$ é a perda de percurso, calculada em função de uma distância d entre um nó transmissor e um nó receptor, $PL(d0)$ é a perda de caminho conhecida, para uma distância de referência $d0$, η é o expoente de perda de sinal, e X_σ é uma variável aleatória gaussiana que descreve uma média com desvio padrão σ. Todos esses quatro parâmetros, são definidos como parâmetros do módulo de canal sem fio do simulador Castalia [Boulis, 2010].

Periodicamente são coletados parâmetros deste canal sem fio, que descrevem a qualidade do canal, como por exemplo interferência e ruído com base nos modelos disponíveis no simulador Castalia. Os erros de transmissão e os parâmetros da camada física, como frequência de operação, nível de interferência e taxa de erros são informados para cada canal com base no modelo de rádio disponível no simulador.

Uma vez considerados esses pressupostos, o passo seguinte é definir como serão implementados o modelo de interferência, o modelo de rádio e as adaptações realizadas no módulo MAC, bem como a classe geradora de tráfego, empregada para avaliar o desempenho da solução implementada no simulador Castalia.

[3] Nos experimentos realizados não foram considerados cenários de mobilidade.

3.1 Modelo de Interferência a Dois Saltos

A interferência é uma das principais causas de degradação de desempenho em redes multi-salto. Logo, para melhorar o desempenho destas redes, é necessário ter algum conhecimento de como os enlaces de uma rede podem interferir entre si e em que medida. No entanto, o problema de estimar a interferência entre os nós de uma rede multi-salto sem fio é um desafio [Padhye et al., 2005].

Dadas essa dificuldade, a maior parte dos trabalho sobre redes sem fio multi salto tem assumido que a informação sobre a interferência na rede ou é conhecida, ou se aplica um modelo de interferência mais simplificado, como por exemplo um modelo de interferência a dois saltos.

Em um cenário de redes de sensores sem fio, o desempenho de uma rede pode ser sensivelmente afetado pela interferência entre os nós. Dessa forma, dois enlaces interferentes entre si não podem realizar suas transmissões se estiverem utilizando o mesmo canal. Sendo assim, será adotado na adaptação do protocolo MAC um modelo de interferência a dois saltos, nos moldes do que é proposto em Padhye et al. [2005].

No modelo proposto, dois enlaces só serão considerados interferentes se estiverem a dois saltos de distância um do outro. A escolha desse modelo é adequada pois os rádios dos nós só podem comunicar com um vizinho (a um salto) por vez.

Assim os vizinhos a um salto não devem ser considerados interferentes, uma vez que a interferência entre eles é tratada por mecanismos de contenção baseados no envio de pacotes de controle RTS e CTS, enviados no canal comum de controle e disponíveis no protocolo MAC adotado.

3.2 Modelo de Rádio Cognitivo

O hardware encontrado nas atuais RSSF possui limitações de processamento, energia, memória e de transmissão. Em geral, os nós sensores são equipados somente com um transceptor que opera na frequência livre ISM e a transmissão é a tarefa que mais consome energia [Correia & Nogueira, 2008].

Dessa forma, evitar colisões dos quadros transmitidos reduz a quantidade de retransmissões na rede e consequentemente o consumo de energia [Silva et al., 2009]. Isso demonstra que o emprego de rádios cognitivos nas RSSF pode reduzir o consumo de energia.

No modelo de rádio cognitivo adotado, são tratadas apenas as funcionalidades do arcabouço de gerenciamento inerentes ao sensoriamento, decisão e a mobilidade do espectro. O modelo empregado para atender essas funcionalidades se baseou no

desenvolvimento de módulos e adaptações de protocolos para o simulador de RSSF Castalia [Boulis, 2010].

Primeiramente, foi feita uma adequação na camada física (rádio) do simulador. Para tanto, foram inseridos múltiplos modelos de rádio e um controlador de rádio responsável por selecionar qual rádio estará ativo, conforme pode ser visto na Figura 3.1. Isso é muito útil para comutar entre diferentes rádios de acordo com os requisitos de banda e qualidade de serviço.

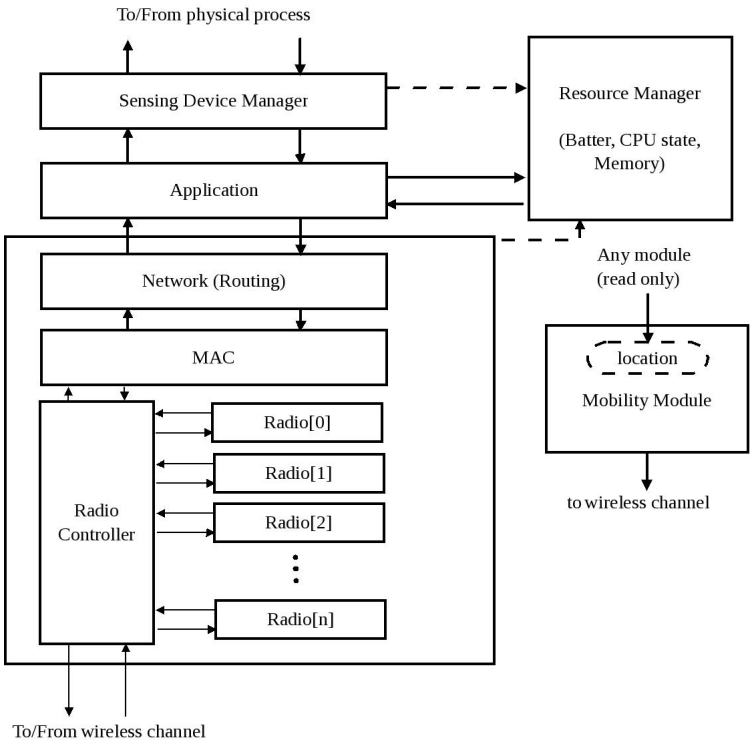

Figura 3.1. Diagrama de implementação de múltiplos rádios no simulador. *Adaptado de [Boulis, 2010].*

Como demonstrado na Figura 3.1, inicialmente, o controlador de rádio ativa temporariamente um dos rádios que opera em determinada frequência e escuta se existem transmissões em andamento em canais específicos. Esse processo é realizado com todos os rádios modelados no simulador.

Dessa forma, por observação das frequências utilizadas em uma determinada região, é possível determinar quais frequências e canais estão livres. Esse sensoriamento do espectro deve ser realizado periodicamente para evitar interferências e modificar dinamicamente a escolha do canal, conforme pode ser visto na Figura 3.2.

Após o controlador ativar o rádio, a decisão do espectro será feita por um método de seleção dinâmica de canais, que em uma primeira abordagem (CogTMAC), levará

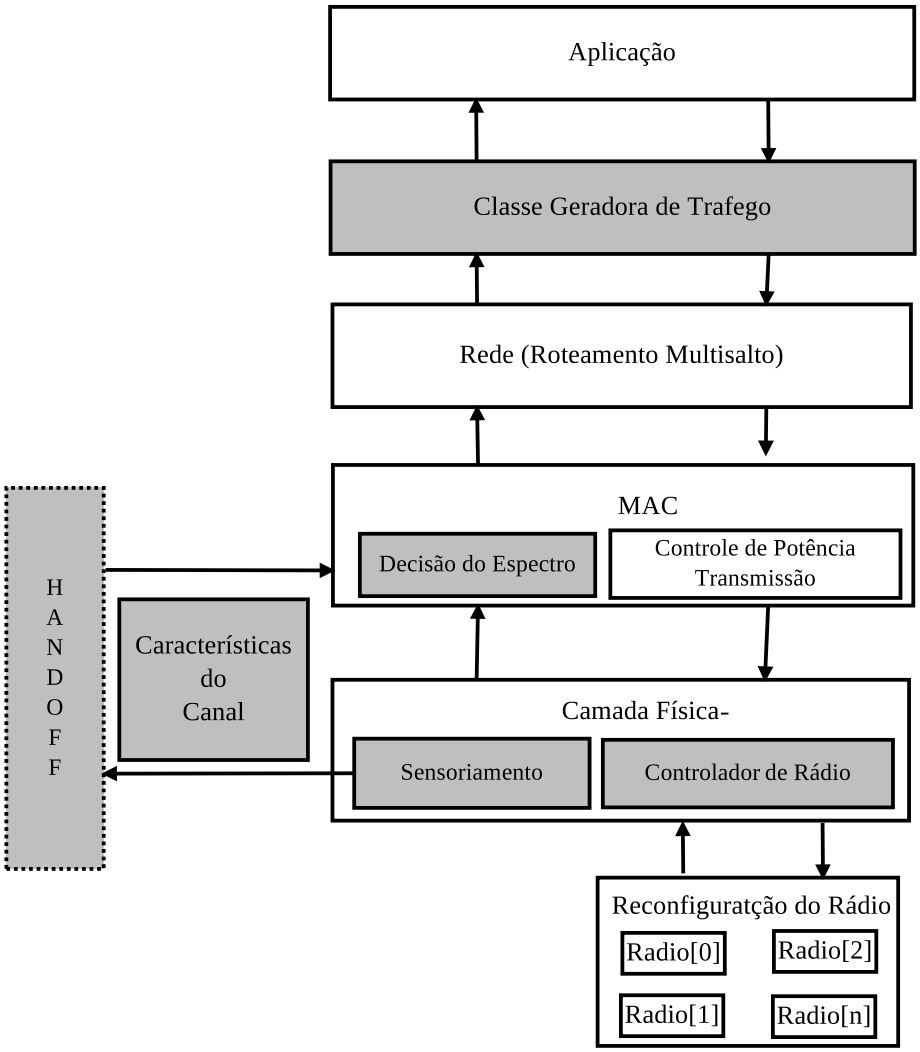

Figura 3.2. Protocolo de controle de acesso ao meio.

em consideração apenas uma das características que afetam a qualidade do canal, a força do sinal recebido (RSSI).

Em uma segunda abordagem, o procedimento de troca de canais levará em consideração não somente o RSSI, mas também a relação sinal interferência ruído (SINR - *Signal Interference Noise Ratio*), o ruído do base, e o atraso fim-a-fim (AhpTMAC). Neste segundo método, tais parâmetros constituirão a entrada para a escolha do melhor canal, baseada em uma técnica de tomada de decisão conhecida como AHP (*Analytical Hierarchical Process*) [Saaty, 2000].

Esse arcabouço de decisão, bem como o modelo de mobilidade de canais em função da alteração das características que afetam a qualidade do canal e o modelo de sensoriamento, hachurados na Figura 3.2 , demandam que sejam feitas adaptações na

camada MAC do simulador.

Portanto, nas seções a seguir são tratados os aspectos inerentes a essas adaptações, tais como leitura dos parâmetros do rádio, mobilidade dos canais (*handoff*) e notificação do canais escolhidos, que deverão ser considerados e implementados no módulo MAC do simulador, conforme mostra a Figura 3.2.

3.3 Implementação Proposta

Nesta seção serão apresentadas as implementações das funcionalidades do arcabouço de gerenciamento do espectro proposto por Akyildiz et al. [2009]: sensoriamento, decisão, mobilidade e compartilhamento do espectro, conforme ilustrado na Figura 3.3, bem como um mecanismo de controle de canal comum para sincronizar dois nós operando em canais diferentes, que desejarem comunicar[4].

Para atender à essas funcionalidades foram desenvolvidas as funções e procedimentos elencados a seguir:

1. Descoberta de vizinhos;
2. Descoberta de vizinhos a dois saltos;
3. Escolha do método de classificação dos canais;
    - Mecanismo de seleção dinâmica de canal baseada no RSSI (CogTMAC);
    - Classificação dos canais de acordo com o AHP (AhpTMAC);
4. Notificação do canal escolhido;
5. Mecanismo de sincronização de canais (Controle de Canal Comum).

Esses procedimentos são realizados pelos nós de uma RSSF, no momento em que são ligados e repetidos quando o temporizador de escolha do melhor canal expira. Assim, quando um nó é ligado, ele deve encontrar os seus vizinhos e em seguida a sua lista de vizinhos a dois saltos.

De posse destas duas listas, cada nó escolhe o melhor canal disponível com base em um dos dois métodos de decisão (CogTMAC ou AhpTMAC). Após ter sido escolhido o canal, a notificação deve ser feita e o processo repetido quando o temporizador atingir seu tempo limite.

[4]As implementações simulador Castalia estão disponíveis no Anexo A.

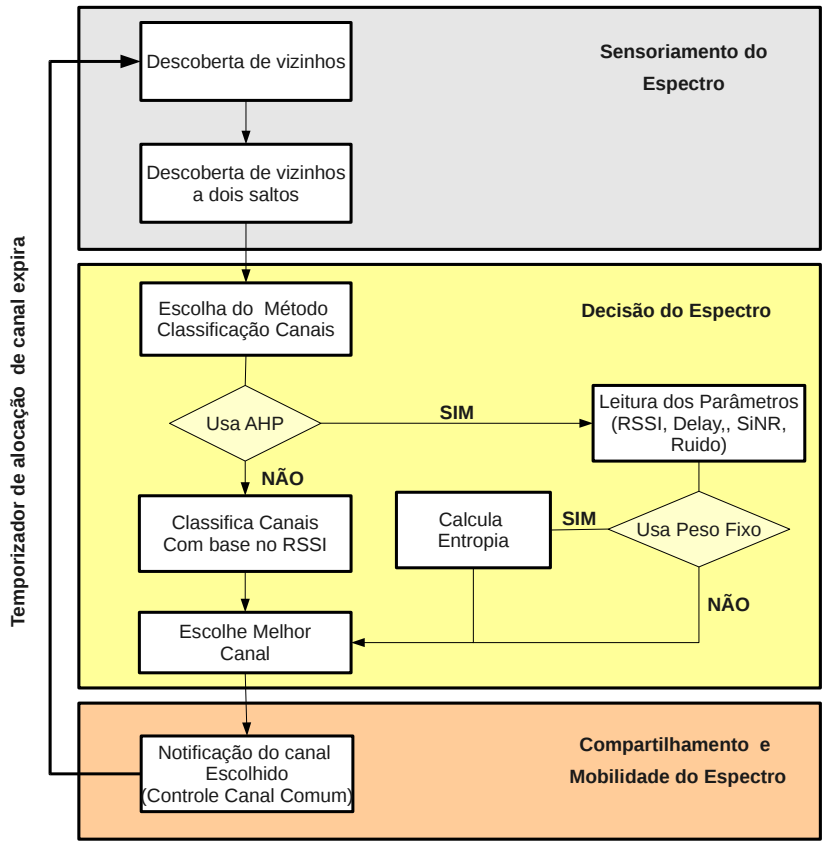

Figura 3.3. Implementações propostas nas funcionalidades do arcabouço de gerenciamento do espectro.

Por último, se faz necessário a presença de um mecanismo de sincronização dos canais, pois os dois métodos de decisão propostos constituem mecanismos distribuídos e assíncronos que executam em cada nó de forma independente. Dessa forma, a decisão do espectro pode levar a situações nas quais um par de nós que desejam comunicar estejam com diferentes canais selecionados. Nas seções a seguir cada um desses procedimentos e funções são descritos.

3.3.1 Sensoriamento do espectro

Nesta etapa do gerenciamento do espectro foram implementadas procedimentos para descoberta de vizinhos a um e dois saltos. O modelo de interferência empregado, conforme descrito na seção 3.1, considera que dois enlaces interferentes entre si não podem realizar suas transmissões se estiverem utilizando o mesmo canal [Padhye et al.,

2005].

Sendo assim na fase de sensoriamento do espectro os nós devem realizar periodicamente os procedimentos de descoberta de vizinhos imediatos e a dois saltos. O método de descoberta de vizinhos imediatos, aqueles dentro do alcance de transmissão, é baseado no Algoritmo 1 apresentado na seção 3.3.1.1. Enquanto que o procedimento de descoberta de vizinhos a dois saltos é descrito na seção 2.

3.3.1.1 Procedimento de Descoberta de Vizinhos

O procedimento de descoberta de vizinhos utilizado segue um paradigma clássico de computação distribuída, conhecido na literatura como algoritmos de sonda/resposta [5] para computações distribuídas em grafos [Andrews, 1991].

Tal paradigma foi empregado devido a natureza orientada a eventos do simulador Castalia, em que cada nó realiza o processamento do algoritmo distribuído proposto. Dessa forma, as mensagens de sonda são enviadas de forma assíncrona a todos os vizinhos, que após receberem a mensagem de sonda, trabalham de forma concorrente entre si.

Assim, o procedimento de descoberta de vizinhos é definido por um método que é chamado quando um temporizador no simulador expira. Nesse caso, o temporizador definido no código para proceder o envio de mensagens de *broadcast* para descoberta dos vizinhos é o FIND_NEIGHBORS, cujo a implementação pode ser vista no Anexo A.

O temporizador cria um pacote[6] MAC e o armazena no *buffer* de envio de pacotes da camada MAC do nó. A mensagem solicitando a descoberta de vizinhos é enviada a partir de cada instância do nó no ato de sua inicialização.

Na inicialização da rede uma frequência e canal de instalação são conhecidos e o nó envia em *broadcast* um quadro FN (*Find Neighbors*) aos seus vizinhos a um salto, que respondem à requisição com um quadro NF (*Neighbor Find*). Quando o nó recebe a resposta NF, ele adiciona o endereço (ID) do nó remetente à sua lista de vizinhos (LVN_i). O procedimento de descoberta de vizinhos imediatos é realizado periodicamente, mas o *broadcast* do quadro FN (*Find Neighbors*) é feito em um canal escolhido e compartilhado pelos nós (descrito na seção 3.3.3). O Algoritmo 1 descreve a formação da lista de vizinhos.

[5] Traduzido do inglês *probe/echo*. *Probe* é uma mensagem de sonda enviada por um nó ao seu vizinho no grafo. *Echo* é a resposta do vizinho ao nó do qual recebeu a mensagem de sonda.

[6] Apesar da camada ter como unidade de dados a denominação quadro, e tipicamente o nome pacote ser usado apenas na camada de redes, devido a questões de compatibilidade com a nomenclatura utilizada pela Castalia, as unidades de dados enviados e recebidos por qualquer camada serão tratadas pelo nome pacote.

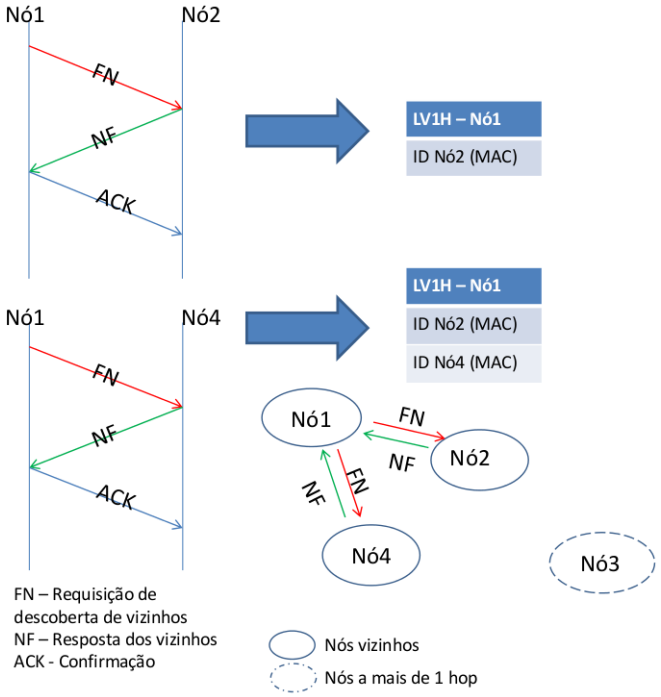

Figura 3.4. Procedimento de descoberta de vizinhos

Função DescobreVizinhos(N_i);
/* Procedimento para obter a lista vizinhos de um determinado N_i */
/* LVN_i: Lista de vizinhos a 1 salto do nó i */
/* N_i, N_j: Dois nós vizinhos quaisquer */
/* NosRSSF: Conjunto dos nós sensores */
início
    $LVN_i \leftarrow \varnothing$;
    para cada $N_i \in NosRSSF$ **faça**
        N_i Envia broadcast para N_j;
        se N_j *Responde broadcast para* N_i **então**
            $LVN_i \leftarrow LVN_i \cup N_i$;
        fim
    fim
    retorna LVN_i
fim

Algoritmo 1: Função que Descobre os Vizinhos de um nó.

Conforme pode ser visto na Figura 3.4, quando o nó 1 envia uma mensagem FIND_NEIGHBOOR em *broadcast* para seus vizinhos a um salto (nós 2 e 4), esses nós respondem a requisição (sonda) com uma mensagem NEIGHBOR_FIND (resposta). Quando o nó 1 recebe esta mensagem NEIGHBOR_FIND, este adiciona o nó remetente na sua lista de vizinhos.

3.3.1.2 Procedimento de Descoberta dos Vizinhos a Dois Saltos.

O modelo de interferência apresentado na seção 3.1, considera que dois nós são interferentes se são separados por dois saltos na RSSF considerada. Dessa forma é necessário um procedimento de descoberta de vizinhos a dois saltos.

Tal procedimento, deve varrer a RSSF e montar uma lista ($LV2HN_i$) para cada nó (N_i) pertencente a essa rede. A implementação desse procedimento no simulador, é invocada por meio de um temporizador (FIND_NEIGHBORS_2_HOPS), cujo o pseudocódigo é descrito no Algoritmo 2.

Esse procedimento é executado no momento em que esse temporizador é ativo, e faz com que um nó (N_i) ao receber a lista de vizinhos a um salto (LVN_i), verifique se cada um dos nós (N_i) pertencentes a essa lista é ele mesmo ou um de seus vizinhos a um salto.

Os nós que não pertencem a lista de vizinhos de um determinado nó N_i são adicionados a lista de vizinhos a dois saltos ($LV2HN_i$). Tal procedimento é realizado para identificar a lista dos nós potencialmente interferentes.

```
1 Função ListadeVizinhos2Hops(N_i, LVN_i);
  /* Procedimento para obter a lista vizinhos a dois saltos de um
     determinado N_i                                              */
  /* LVN_i: Lista de vizinhos a 1 salto do nó i                   */
  /* N_i: Nó cujo os vizinhos são os nós N_j                      */
  /* N_j: Nó j ∈ LVN_i                                            */
2 início
3 |  LV2HN_i ← ∅;
4 |  para cada N_j ∈ LVN_i faça
5 |  |  LV2HN_i ← LV2HN_i ∪ (LVN_j − LVN_i) − N_i ;
6 |  fim
7 |  retorna LV2HN_i
8 fim
```

Algoritmo 2: Função que obtem a lista de vizinhos a dois saltos de um nó.

Após a descoberta de vizinhos, cada nó realiza o sensoriamento do espectro, aferindo os parâmetros de entrada para os mecanismos de decisão.

3.3.2 Método de Decisão

Nessa fase, o nó escolhe em que canal irá transmitir em função dos parâmetros de RSSI, SINR, ruído base e do atraso fim a fim. Conforme citado anteriomente, a escolha do melhor canal disponível pode utilizar dois métodos de decisão CogTMAC e AhpTMAC, descritos a seguir.

3.3.2.1 CogTMAC: Mecanismo de Seleção de Canais Baseada no RSSI

O método CogTMAC escolhe o melhor canal disponível em função do ruído local observado pelo nó. O ruído local é estimado pelo valor do RSSI do canal. O CogTMAC utiliza o recurso de julgamento de canal livre CCA (*Clear Channel Assessment*), disponível no rádio CC2420 e implementado no Castalia, que permite ao rádio verificar se o canal está livre para transmissões (linha 3 do Algoritmo 3).

O canal escolhido será aquele que esteja livre e possua o menor valor de RSSI, e que portanto ofereça a menor interferência (linhas de 3 a 5). Uma vez escolhido o melhor canal, ele deve ser marcado como ocupado (linha 6) e uma notificação é enviada aos vizinhos (linhas 7 e 8). O método de notificação é detalhado na seção 3.3.3.

```
1  Procedure escolheCanalRSSI(ListaCanais, LVN_i);
2  para cada C_i ∈ ListaCanais faça
3      se C_i estiver livre então
4          se C_i.RSSI ≤ melhorCanal.RSSI então
5              melhorCanal ← C_i;
6          fim
7      fim
8      Atualiza ListaCanaisAtribuidos;
       /* Notifica os vizinhos a um salto                    */
9      NotificaLV1Hop();
       /* Notifica os vizinhos a dois saltos                 */
10     NotificaLV2Hop();
11 fim
```

Algoritmo 3: Mecanismo de seleção do canal com base no RSSI.

3.3.2.2 AhpTMAC: Mecanismo de Seleção dos Canais de Acordo com o AHP

Conforme citado no referencial teórico deste trabalho, na seção 2.2.2, um modelo de decisão baseado apenas em parâmetros do canal, como por exemplo a relação sinal-ruído, ou na força do sinal recebido, conforme proposto no mecanismo descrito na seção

3.3.2.1, pode não ser suficiente para escolher o melhor canal ou frequência disponíveis [Akyildiz et al., 2008].

Além disso, deve-se considerar os diferentes requisitos de qualidade de serviço da aplicação [Akyildiz et al., 2008], como por exemplo definições de atraso fim a fim e largura de banda mínimos exigidos, bem como a definição de aplicações com requisitos de tempo real ou insensíveis ao atraso.

Sendo assim, a classificação dos canais e a escolha das frequências e rádios escolhidos devem levar consideração não somente os parâmetros físicos dos canais, mas também os parâmetros da aplicação. Então, se faz necessário um modelo de decisão que colete os parâmetros da camada de aplicação e da camada física, e com base nesses parâmetros coletados decida qual canal, rádio e frequência utilizar.

O AHP (*Analytic Hierarchy Process*) é um classificador multi-parâmetros que tem sido usado para decisão de espectro [Ge et al., 2009]. Neste trabalho, adaptamos esse método ao protocolo T-MAC, classificando os canais a partir de parâmetros das camadas física e de aplicação. Dessa forma, o protocolo AhPTMAC coleta parâmetros físicos do rádio como RSSI, SNIR e ruído local e considera requisitos da aplicação para classificação de canais, tais como o atraso. Para tanto, é necessário definir os pesos dos diferentes parâmetros. Alguns métodos especificam esses pesos manualmente de acordo com critérios estabelecidos pelo usuário. Porém, a maioria dos usuários não tem conhecimento prévio de quanto cada parâmetro influencia na escolha do melhor canal.

O protocolo AhPTMAC calcula de forma automática os pesos dos diferentes parâmetros, usando o conceito de entropia de *Shannon* e o método proposto por Ge et al. [2009]. Para o cálculo dos pesos é utilizada uma Matriz de Parâmetros $M_{C\times P}$, na qual cada linha C é uma leitura referente aos parâmetros de um canal e P são as colunas referentes aos parâmetros (Ruído, RSSI, SINR, Atraso) desses canais.

O primeiro passo, descrito no Algoritmo 4, consiste em calcular uma constante de entropia C_{entr} (linha 2), dada em função da quantidade de leituras dos parâmetros de um canal C. Em seguida é calculada a entropia para cada parâmetro da matriz $M_{C\times P}$. Logo, para cada parâmetro P_k é calculada a entropia por meio de um fator F_{ji}, cujo valor individual de cada parâmetro P_{ki} na coluna k é dividido pelo somatório de todos os valores da coluna (linha 5). Após ter sido calculado o fator F_{ji}, é calculado o vetor de entropia H_i, cujos elementos são somados e multiplicados pela constante C_{entr} (linhas 4–7).

Finalmente, o último passo é calcular o vetor de pesos W_i em função do vetor de entropias H_i e do número de parâmetros k analisados. Então, o peso de cada parâmetro é calculado (linhas 8–10). O valor máximo do peso é escolhido (linha 10). O algoritmo

```
1  Procedimento AhPTMAC com pesos dinâmicos(Canal, M_{C×P});
2  C_entr ← (−1 ∗ (1/ log(qtde_linhas C)) ;
3  para i ← 1 até P faça
4  |   para j ← 1 até C faça
   |   |   /* Soma os valores de um parâmetro i                    */
5  |   |   F_ji ← B_ji/soma_coluna_i;
6  |   |   H_i ← F_ji ∗ log(F_ji);
7  |   |   H_i ← C_entr ∗ soma(H_i);
8  |   fim
   |   /* Cálculo dos pesos W                                     */
9  |   para i ← 1 até k faça
10 |   |   W_i ← (1 − H_i)/(k − soma(H));
11 |   fim
12 |   Escolhe max(W_i)
13 fim
14 retorna Entropia, Peso, Parâmetro_Maior_Peso
```

Algoritmo 4: Método AHP com pesos dinâmicos.

retorna os valores da Entropia, Peso e Parâmetro Maior Peso (linha 11) para que o método de decisão escolha o melhor canal disponível.

```
1  Função AHPTMAC(TabelaDeValores, ListaDeCanais);
2  pesos = PesosAhPTMAC(TabelaDeValores)
3  normalizaMetricas(ListaDeCanais)
4  para cada Canal_i ∈ ListaDeCanais faça
5  |   Canal.indiceDeQualidade ←
   |   pesos.RSSI ∗ Canal_i.RSSI + pesos.SNIR ∗ Canal_i.SNIR +
   |   pesos.ruidoBase ∗ Canal_i.ruidoBase + pesos.atraso ∗ Canal.atraso
6  fim
7  melhorCanal ← arg max_i(Canal_i.indiceDeQualidade)
8  retorna (melhorCanal)
```

Algoritmo 5: Escolha do melhor canal a partir do pesos.

Esses pesos serão usados como entrada do AHP, Algoritmo 5, para a escolha do melhor canal. A tabela de pesos é normalizada de acordo com os canais (linhas 2–3). Um índice de qualidade é definido para cada canal de acordo com os pesos das métricas (linhas 4–5). O canal com o maior índice de qualidade é selecionado (linha 6) e retornado (linha 7).

3.3.3 Compartilhamento do canal escolhido

Um nó, ao escolher o melhor canal, deve notificar seus vizinhos imediatos e a dois saltos. A mensagem de notificação é enviada em *broadcast*, mas para evitar que sejam feitas notificações repetidas o nó notificador procura identificar interseções entre as listas de vizinhos. Tal procedimento é descrito no Algoritmo 6.

```
1 Procedimento ReencaminhaBroadcast(LVN_i, LV2HN_i)
2 ListaNosParaVisitar ← LV2HN_i;
3 N ← ordena(LVN_i);
4 para cada N_j ∈ N faça
5 |   se ListaNosParaVisitar ∩ LVN_j ≠ ∅ então
6 |   |   ListaNosParaVisitar ← ListaNosParaVisitar − LVN_j;
7 |   |   envia(N_j);
8 |   fim
9 fim
```

Algoritmo 6: Procedimento de repasse de broadcast.

O algoritmo apresenta um mecanismo similar ao de *flooding*. Inicialmente é criada uma lista dos nós que irão repassar, via *broadcast*, a informação do canal escolhido a seus vizinhos (linha 2). O remetente encaminha a mensagem para os seus vizinhos com o maior grau de conectividade, a fim de reduzir a quantidade de pacotes repassados. Os nós com a menor quantidade de vizinhos são adicionados a este conjunto a cada iteração (linhas 4-7). O processo é concluído quando todos os nós a dois saltos forem visitados.

3.3.4 Mobilidade do espectro

Conforme foi visto na seção 2.4.3, o protocolo T-MAC, se baseia no mecanismo de contenção CSMA/CA, do padrão IEEE 802.11 e tenta evitar a ocorrência de colisões definindo a comunicação entre dois nós por meio do envio de pacotes RTS/CTS/DATA/ACK (*Request-To-Send/Clear-To-Send/Data*) [van Dam & Langendoen, 2003].

No entanto, apesar do T-MAC utilizar um procedimento de alocação dinâmica de canal, o mesmo não consegue restabelecer a sincronização entre os nós em função da mudança de canais provocada pelo mecanismo de escolha do melhor canal proposto neste trabalho. Essa falta de sincronismo dos canais se dá quando um nó origem estiver transmitindo em um canal diferente do nó destino.

Esse problema de comunicação entre os nós, foi resolvido por meio de um mecanismo para sincronizar os canais de um par de nós comunicantes. Por meio desse sincro-

nismo, o nó de destino da mensagem escuta no canal certo durante a transmissão. Esse mecanismo foi adicionado ao protocolo T-MAC presente no simulador. Dessa forma, um nó transmissor antes de enviar um pacote RTS, verifica na lista de vizinhos o canal em que o destino está escutando e muda para esse canal no rádio.

Ao enviar o pacote de RTS, o nó de origem envia o canal escolhido no qual será transmitida a mensagem. Ao receber o RTS o nó de destino ativa um temporizador CHANGE_CHANNEL_TO_TRANSMISSION e envia um quadro CTS.

Ao receber o CTS o nó transmissor retorna ao seu canal original (C_j) de transmissão e envia o pacote. Ao receber o pacote o nó destino envia um ACK e retorna ao canal de origem (C_k). Esse mecanismo de sincronização de canais é ilustrado na figura 3.5.

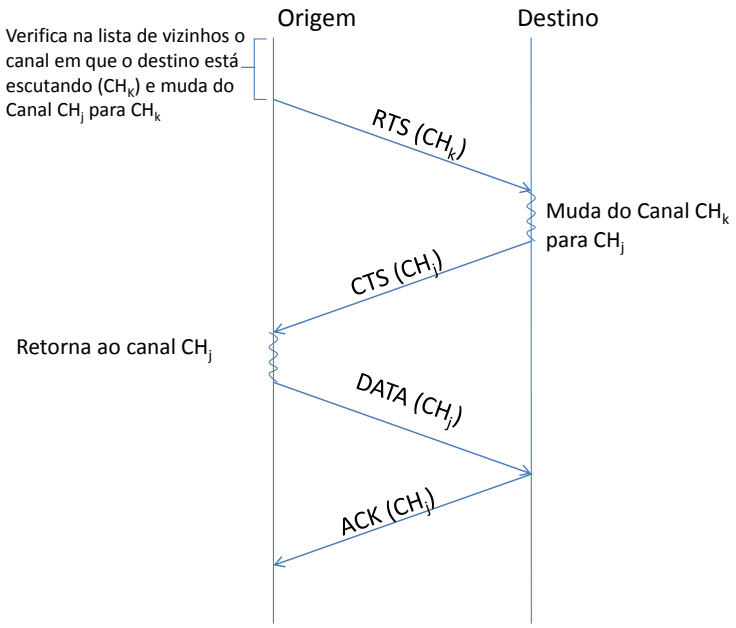

Figura 3.5. Mecanismo de sincronização do canal de transmissão.

Em outras palavras, a mudança dinâmica de canais pode provocar a ruptura da comunicação entre os nós, que pode ocorrer quando um nó estiver transmitindo em um canal diferente daquele do nó destino. Assim, um nó transmissor, antes de enviar um quadro de RTS, verifica na lista de vizinhos o canal em que o destino está escutando e muda para esse canal ($CH_j \rightarrow CH_k$). Um temporizador também é ativado de forma

que o emissor retorne para o canal escolhido antes da chegada do quadro de CTS.

No quadro de RTS é inserido o canal escolhido pelo nó origem, pelo qual será transmitida a mensagem. O nó destino, ao receber o RTS, muda de canal ($CH_k \rightarrow CH_j$) e envia um quadro de CTS. Ao receber o CTS, o nó transmissor já estará no seu canal original (CH_j), e assim transmite o quadro DATA. Ao receber o DATA, o nó destino envia um ACK e retorna ao seu canal de origem (CH_k).

3.3.5 Análise de complexidade

No intuito de analisar a solução proposta, será apresentada a seguir a análise de complexidade de tempo e espaço, de cada uma das funções, procedimentos e algoritmos descritos anteriormente nesta seção 3.3; visando avaliar esses algoritmos sob o ponto de vista de escalabilidade e ordem de complexidade.

A função *DescobreVizinhos*, descrita no Algoritmo 1 é responsável por obter a lista de vizinhos LVN_i de um nó N_i e, equivale ao problema de se encontrar os vértices adjacentes V_j de cada um dos vértices V_i pertencentes a um grafo $G(V, E)$, onde V corresponde ao nós N_i e E a lista de vizinhos LVN_i.

Logo, esse procedimento é análogo a busca em profundidade (DFS) em um grafo, pois ele tem que visitar todos os nós para descobrir a topologia da rede, por difusão das mensagens de sonda e coletar as informações da topologia, por meio do algoritmo de coleta com sonda/resposta. Assim no pior caso, o cálculo da lista de vizinhos de todos os nós obtida por meio dessa função, tem complexidade de tempo $\Theta(|V + E|)$ e de espaço $\Theta(|V|)$.

O procedimento de descoberta dos Vizinhos a Dois Saltos, descrito no Algoritmo 2 na seção 3.3.1.2 retorna a lista $LV2HN_i$ dos vizinhos dos vizinhos de um N_i. Para tanto, a lista de vizinhos LVN_i a um salto do N_i é varrida a um custo de no máximo $O(n)$ e para cada um dos nós N_j pertencentes a essa lista é verificado se o seus vizinhos já fazem parte da lista de vizinhos do N_i também a um custo máximo de $O(n)$.

No entanto, se o custo para varrer a lista LVN_i, no pior caso é $O(n)$ implica em dizer que todos os demais $N - 1$ nós da rede são vizinhos do N_i e nesse caso ele não teria vizinhos a dois saltos. Se os vizinhos do N_i é igual ao número de nós da rede menos dois, o custo para se calcular a $LV2HN_i$ é $O(1)$. Por último, se a cardinalidade da LVN_i é um, o custo para se calcular a $LV2HN_i$ é $O(n)$. Dessa forma, podemos concluir que a complexidade de tempo para se obter a lista de vizinhos a dois saltos é no máximo $O(n)$.

A complexidade do algoritmo de classificação dos canais baseada no AHP, conta dois laços de repetição aninhados em um laço mais externo com p iterações. Como p

corresponde a um número constante de parâmetros, podes dizer que a complexidade do método é linear já que os laços mais internos tem complexidade de tempo $O(p+k)$.

A notificação do canal escolhido está essencialmente ligada ao mecanismos de descoberta de vizinhos a um e dois dois saltos. Assim, a complexidade algorítmica dos métodos é no máximo $O(|V| + |E|)$, em que $|V|$ corresponde ao total de nós e $|E|$ corresponde ao número de vizinhos alcançáveis por um nó N_i.

Esse número cresce com o aumento da densidade de nós da rede e consequentemente com a LVN_i, porém é independente do número total de nós. Desta forma, se considerarmos a complexidade algorítmica não há problemas de escalabilidade na solução proposta.

Capítulo 4

Resultados e Discussões

Neste capítulo são apresentados os resultados obtidos com a adaptação do protocolo T-MAC definida no Capítulo 3, bem como o método de avaliação e análise dos resultados. Dessa forma o capítulo foi divido em quatro partes: metodologia de avaliação e simulação, resultados obtidos e análise dos resultados. A primeira parte é apresentada na Seção 4.1 e descreve os cenários e as métricas de desempenho avaliados, bem como descreve uma classe geradora de tráfego utilizada na simulação para avaliar os resultados de acordo com diferentes tipos de tráfego. Os resultados obtidos nas simulações considerando-se a combinação de cenários e métricas propostas são apresentados na Seção 4.2. Por fim, na Seção 4.3 é apresentada a análise e discussão dos resultados.

4.1 Metodologia de Avaliação e Simulação

A pesquisa quantitativa e a qualitativa são coletivamente chamadas de pesquisa empírica [Wainer, 2007]. Neste trabalho, esse tipo de metodologia de pesquisa será adotado, dando ênfase na análise de resultados, lançando-se mão de ferramentas de simulações e técnicas estatísticas para avaliar os resultados.

Devido a dificuldades na montagem de experimentos reais em uma RSSF, optou-se por empregar a simulação como método de avaliação da solução proposta. Essa escolha se justifica, principalmente devido aos custos e tempos de execução demandados em cenários nos quais se deseja avaliar redes de sensores de alta densidade. Assim, a escolha do ambiente de simulação, levou em consideração a necessidade de se ter um arcabouço para tratar protocolos de acesso ao meio para RSSF e que melhor se adequasse aos requisitos do problema avaliado. Dentro dessa perspectiva, o simulador adotado foi o Castalia [Boulis, 2010].

Foram realizadas 33 simulações para cada um dos cenários e métricas que serão apresentados na seção 4.1.1. Em cada uma das simulações foi feito o sorteio de k nós ($k \mid 1 \leq k \leq 4$), de forma que um nó N_i escolha aleatoriamente um nó vizinho $N_j(N_j \mid N_j \in LVN_i)$ para estabelecer um fluxo entre eles, isto significa que existirão de um a quatro fluxos simultâneos na rede.

O número de nós usados na simulação foi de 18, cada um equipado com um rádio CC2420 e distribuído no espaço de simulação de acordo com os cenários de *deployment* citados na seção 4.1.1.

O tempo de simulação foi de $500s$ e o temporizador de seleção do melhor canal disponível, consiste em um evento disparado de 30 em $30s$. O intervalo entre o envio de pacotes obedecem as classes de tráfegos definidas na seção 4.1.2.

Para simular o comportamento de um nó sensor, aquisitar os parâmetros de operação do rádio e avaliar a eficiência da solução proposta foram necessárias adaptações nos módulos do rádio, e na aplicação; bem como alterações no protocolo de acesso ao meio T-MAC [van Dam & Langendoen, 2003] e o desenvolvimento de um mecanismo gerador de tráfego.

A adaptação no módulo de rádio do Castalia foi necessária para permitir que múltiplos rádios, de vários modelos fossem avaliados. Para tanto foi necessário o desenvolvimento de um controlador de rádio encarregado de selecionar qual rádio deveria ser ativado.

A adaptação do protocolo de acesso ao meio T-MAC no simulador de RSSF Castalia 3.0, levou em consideração duas abordagens com foco na decisão do espectro e escolha do melhor canal: avaliação da força do sinal (RSSI) no canal amostrado e avaliação do canal com base em parâmetros físicos do canal (ruído base, RSSI e SINR) e o atraso fim a fim, constituindo-se assim uma abordagem *cross layer*. Ambas as abordagens foram descritas no capítulo 3.

Para melhor compreender e interpretar os resultados obtidos nestas duas abordagens, bem como a reproduzir o estudo em questão, adotou-se uma metodologia para avaliar a solução proposta. Tal metodologia é apresentada na Figura 4.1. Conforme pode ser observado, a aplicação simulada é definida por classes de tráfego. O roteamento pode ou não ser considerado, no entanto neste trabalho não foi tratado, focamos apenas na subcamada MAC, onde os algoritmos de decisão do espectro foram avaliados segundos as métricas e cenários descritos na subseção 4.1.1.

Conhecer o tipo de tráfego gerado pela aplicação passa pela implementação de uma classe geradora de tráfego que proverá a aplicação de um modelo de tráfego. A implementação de uma classe geradora de tráfego é muito importante para avaliar qual a influência do tipo de tráfego no comportamento da solução proposta, permitindo

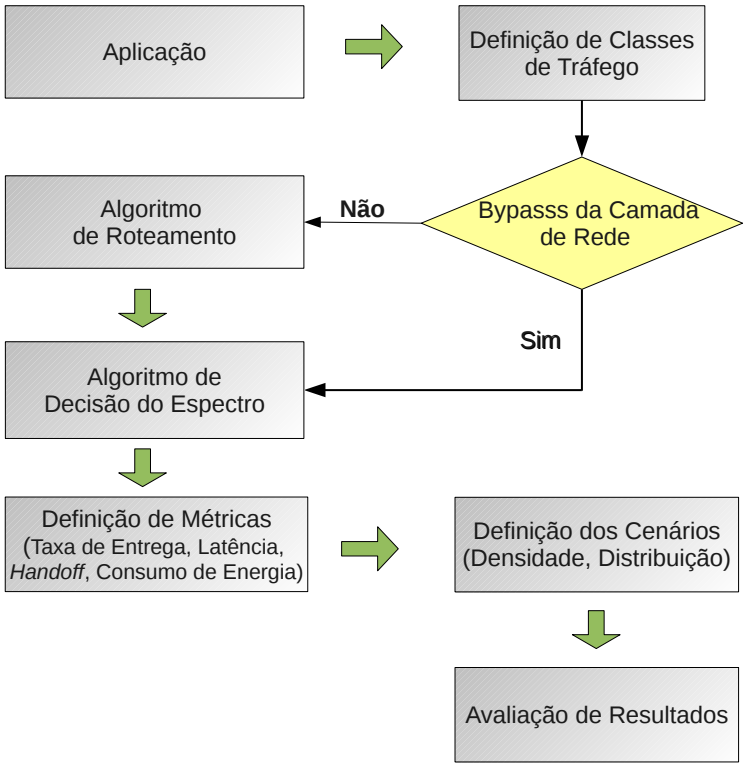

Figura 4.1. Metodologia de avaliação do algoritmo de decisão.

avaliar a taxa de entrega para diferentes tipos de aplicação.

Além disso, permite que sejam estabelecidos modelos de transmissão de dados, para criar cargas de tráfego de acordo com os eventos gerados em uma RSSF. A geração da carga de tráfego foi feita por uma classe geradora de tráfego que segue um escalonamento pré-definido, disciplinando o envio de pacotes de acordo com modelos de tráfego. Em particular, essa classe envia um pedido para transmitir uma carga de pacotes em um período de tempo especificado de acordo com os seguintes modelos de tráfego:

- Tráfego constante (*Constant Bit Rate* - CBR): envio de pacotes de tamanho fixo, a uma taxa constante, ou seja, o intervalo entre o envio de dois pacotes é fixo.

- Tráfego aleatório: envio de pacotes de tamanho fixo, cujo intervalo entre o envio de dois pacotes é aleatório;

- Tráfego em exponencial: envio de pacotes de tamanho fixo em intervalos de tempos exponencialmente distribuídos.

4.1.1 Métricas e Cenários Avaliados

Os resultados obtidos nesse trabalho se basearam nas seguintes métricas de avaliação de desempenho:

- Taxa média de entrega de pacotes;
- Atraso fim a fim (milissegundos);
- Consumo de energia;
- *Handoff* (trocas de canais);
- *Overhead* de transmissão (pacotes retransmitidos);

Os cenários avaliados combinam parâmetros de densidade de nós, níveis de ruído do canal, distribuição (*deployment*) dos nós no espaço e classes de tráfego. A Tabela 4.1 descreve as possíveis instâncias destes cenários.

Tabela 4.1. Resumo dos Cenários Avaliados

Tráfego	Distribuição	Ruído	Densidade
Periódico	Grade, Aleatória Uniforme	Baixo,Médio,Alto	Baixa, Alta
Exponencial	Grade, Aleatória Uniforme	Baixo,Médio,Alto	Baixa, Alta

Com relação a densidade foram estabelecidos dois níveis de densidade, variando-se a área do cenário avaliado. No cenário de baixa densidade a área sensoreada é de $75m \times 30m$, enquanto que no cenário de alta densidade a área tem suas dimensões reduzidas para $50m \times 20m$. Isso possibilita analisar o impacto que a alta e baixa densidade dos nós tem no mecanismo de seleção dinâmica do canal, levando-se em consideração as métricas avaliadas.

Já o parâmetro ruído do canal pode ser variado de acordo três níveis baixo (-100 dBm), médio (-95 dbm) e alto (-92 dBm)[1], possibilitando avaliar o impacto que esse parâmetro, em conjunto com a densidade nós, tem nas medidas citadas anteriormente.

Também foram avaliados cenários, nos quais os nós são distribuídos (*deployment*) de acordo com os seguintes critérios :

- Distribuição aleatória uniforme, na qual os nós são colocados no campo usando uma distribuição aleatória uniforme;

[1]Estes níveis de ruídos foram obtidos empiricamente, por meio de experimentos com nós reais e testando-se cada um desses limiares de ruídos no simulador.

- Distribuição em grade: nós equidistantes, colocados em uma grade de $N \times M$;
- Distribuição aleatória em grade: implantação da rede com nós variando suas posições de maneira aleatória em cada célula de uma grade de dimensões $N \times M$.

Por último, a combinação dos cenários de *deployment*, densidade e ruído foram avaliados levando-se em consideração três classes de tráfego, periódico (CBR), exponencial e aleatório. Para simular uma aplicação que gerasse esses três tipos de tráfego foi desenvolvida uma classe geradora de tráfego que é descrita na seção a seguir.

4.1.2 Classe Geradora de Tráfego

A classe geradora de tráfego, cujo a implementação no Castalia pode ser vista no Anexo B, recebe como parâmetro o número de fluxos e o intervalo de transmissão. Esta classe tem um temporizador chamado SEND_PACKET que é ativado periodicamente de acordo com o parâmetro de intervalo de transmissão. Quando o temporizador SEND_PACKET é ativado, são sorteados N fluxos aleatórios entre um par de nós vizinhos [2].

Em seguida são enviadas mensagens para esses nós indicando que eles devem iniciar o envio de pacotes. Ao receber a mensagem, o nó sorteia um vizinho e envia um pacote para ele. Os fluxos sorteados consideram apenas pares de nós vizinhos, uma vez que de acordo com a metodologia proposta para este trabalho o roteamento foi desconsiderado (*bypass routing*).

Os intervalos de transmissão entre os envios de pacotes obedecem a três classes de tráfego: periódico, aleatório ou exponencialmente distribuído, de acordo com o cenário escolhido. Para a classe de tráfego periódico, foi considerado o intervalo de transmissão entre os pacotes igual a dois segundos ($IT_{trans} = 2s$). Para a classe de tráfego aleatório esse intervalo entre o envio de pacotes é escolhido ao acaso entre zero e dois segundos ($0 \leq IT_{trans} \leq 2s$). Por último, na classe exponencial esse intervalo de transmissão obedece a um tráfego distribuído exponencialmente no tempo de acordo com a função de distribuição:

$$f(x) = \begin{cases} 1 - \frac{1}{b}e^{-(x-a)/b} & \text{se } x \geq a \\ 0 & \text{caso contrário} \end{cases} \tag{4.1}$$

onde os parâmetros a e b foram configurados no arquivo de simulação com os valores $a = 0$ e $b = 0.5$. Um exemplo de distribuição frequência para os intervalos de tempos

[2] Não foi considerado o envio de fluxos para nós que não são vizinhos, devido a limitações do simulador que não apresentou um algoritmo de roteamento adequado aos cenários que foram avaliados.

entre os envio de pacotes, de acordo com essa distribuição, pode ser visto no histograma representado na Figura 4.2.

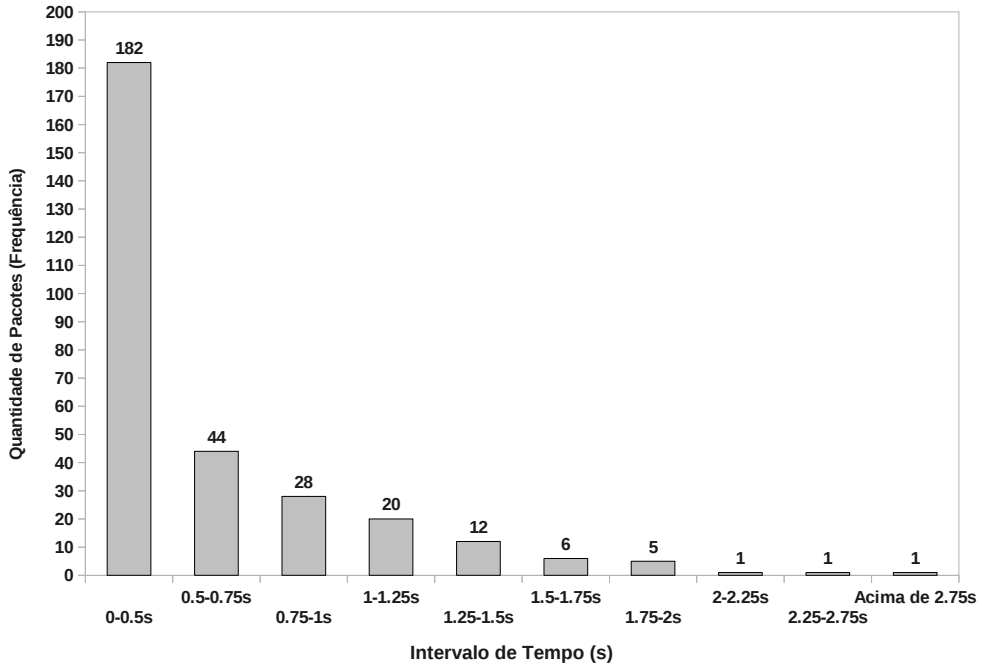

Figura 4.2. Intervalo de envio entre pacotes distribuídos exponencialmente no tempo.

Essa função de distribuição foi implementada na classe geradora de tráfego, utilizando-se uma classe que contém várias distribuições estatísticas contínuas e discretas disponíveis que fornece uma função densidade de probabilidade arbitrária [Saucier, 2000].

No entanto, vale ressaltar que mesmo que o intervalo entre o envio de pacotes seja disciplinado pela classe geradora de tráfego, eles também devem ser enfileirados no *buffer* de saída e obedecerem ao ciclo de operação do protocolo T-MAC.

Assim, levando em consideração a combinação das classes de tráfego com os cenários e métricas elencados anteriormente, foi possível comparar o desempenho do protocolo de decisão do espectro e suas variações (CogTMAC e AhpTMAC), com o desempenho do protocolo T-MAC. Tais resultados são apresentados na seção 4.2 a seguir.

4.2 Resultados Obtidos

Os valores obtidos para cada métrica representam os valores médios das 33 simulações realizadas para cada um dos métodos usando os protocolos T-MAC, CogTMAC e AhPTMAC. Em cada uma das simulações foi feito o sorteio de quatro nós, de forma que um nó N_i escolha aleatoriamente um nó vizinho N_j ($N_j \mid N_j \in LVN_i$) para estabelecer fluxos entre eles. Assim, existirão quatro fluxos simultâneos na rede, simulando a interferência de múltiplas redes.

Apesar da implementação de múltiplos rádios multicanais realizada no Castalia, simulamos cada nó com apenas um rádio multicanal. A justificativa é que sensorear n-rádios com m-canais seria equivalente a sensorear $n \times m$ canais do espectro.

Foram implementados três modelos de tráfego, mas devida ao grande número de combinações de cenários serão apresentados somente os resultados para tráfego periódico (típico das RSSF) e tráfego exponencial, com distribuição dos nós da rede em grade.

A escolha destes cenários não trouxe prejuízo para análise dos resultados. Tendo em vista que, o cenário de tráfego aleatório teve comportamento similar ao cenário de tráfego periódico e os demais cenários de *deployment* tiveram resultados similares ao cenário de distribuição em grade.

Os gráficos mostrados a seguir representam os valores médios de 33 simulações com sementes diferentes, e considerando-se um intervalo de confiança de 99%, para cenários de baixa densidade de nós (representado no gráfico pela legenda $< protocolo > -B$) e para alta densidade de nós (representado pela legenda $< protocolo > -A$), com níveis de ruído baixo, médio e alto.

4.2.1 Tráfego Periódico

Nesta seção serão avaliadas as adaptações no protocolo proposto, em um cenário de tráfego periódico (CBR), cujos os 18 nós foram distribuídos de forma equidistante em uma grade de dimensões 6×3. Os pacotes são enviados periodicamente com intervalo de envio de 2s.

A periodicidade de sensoriamento do canal foi estimada em 30s. Foram simulados 500s de operação da rede, de forma que foram geradas 600 transmissões durante as simulações. A largura de banda utilizada foi a mesma provida pelo rádio CC2420, 250 kbps, com um alcance de transmissão de 250m.

Os valores do ruído do canal foram considerados como baixo (-100 dBm), médio (-95 dbm) e alto (-92 dBm). Além disso, simulamos duas densidades diferentes

ao variar o tamanho da área simulada: Baixa (75x30m) e Alta (50x20m). A seguir são exibidos os principais resultados, referentes as variações de densidade e ruído, considerando-se as métricas de taxa de entrega, atraso fim a fim, consumo de energia, número de trocas de canais e pacotes retransmitidos.

4.2.1.1 Taxa de entrega no tráfego periódico

A Figura 4.3 apresenta a taxa média de entrega para os três protocolos. Nos cenários de baixa densidade de nós observa-se que o protocolo T-MAC possui uma taxa de entrega superior aos protocolos AhPTMAC e CogTMAC somente quando o ruído local é baixo, cerca de 8% e 9% respectivamente.

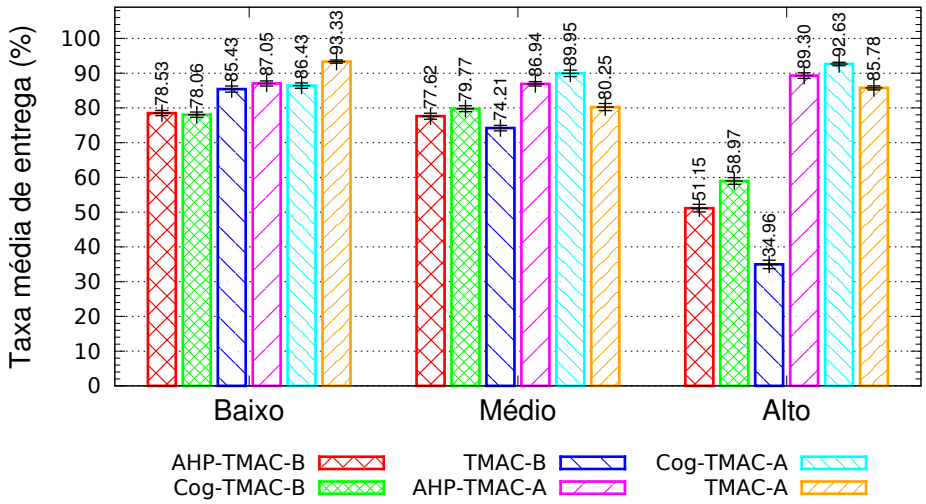

Figura 4.3. Taxa média de entrega de pacotes no tráfego periódico.

Para valores de ruído médio os protocolos cognitivos possuem uma taxa de entrega superior ao T-MAC, cerca de $4,6\%$ para o AhP-TMAC e de $7,5\%$ para o CogTMAC. Esses valores são mais acentuados quando o ruído local é alto. O protocolo CogTMAC continuou a apresentar o melhor resultado, cerca de $68,7\%$ em relação ao T-MAC e de $15,3\%$ em relação ao AhPTMAC.

O comportamento da taxa de entrega é semelhante para os cenários de alta e baixa densidade. Quando o ruído local é baixo, os protocolos cognitivos possuem uma taxa de entrega inferior ao T-MAC, cerca de $7,2\%$ e de 8% para os protocolos AhPTMAC e CogTMAC respectivamente. O desempenho inferior dos protocolos cognitivos em relação ao T-MAC em cenários de baixo ruído se justifica devido às mudanças periódicas de canal.

Durante a troca de canais, os protocolos cognitivos estão sujeitos à sincronização dos nós e a perdas de pacotes. Nos cenários de médio e alto ruído, observa-se que o protocolo CogTMAC obteve a melhor taxa de entrega. Isso pode ser justificado pela escolha de melhores canais de transmissão em função da variação do ruído.

4.2.1.2 Atraso fim a fim no tráfego periódico

A Figura 4.4 apresenta o atraso fim a fim. O protocolo T-MAC apresentou o maior atraso para ruído baixo, cerca de $8,5\%$ superior aos protocolos cognitivos. Isso pode ser justificado pela maior taxa de entrega do T-MAC.

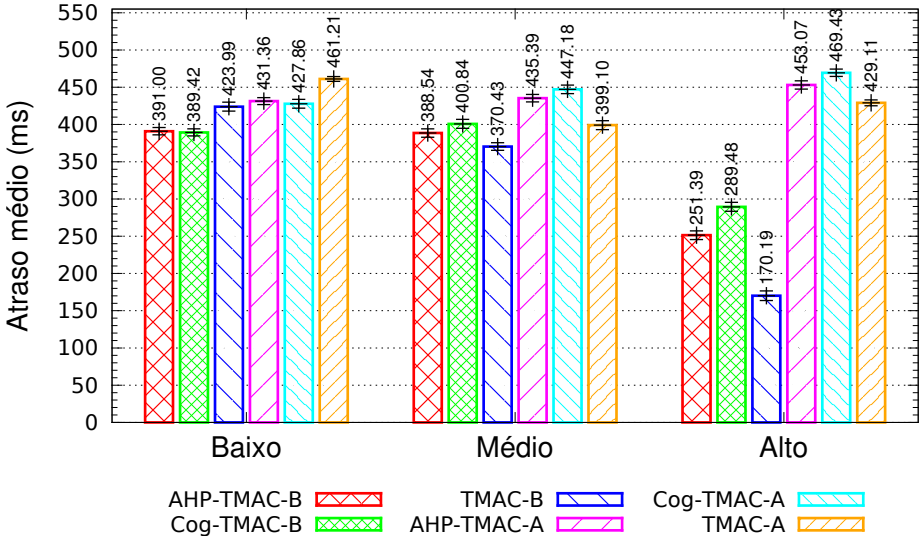

Figura 4.4. Atraso fim a fim no tráfego periódico.

Nos cenários de ruído médio, o CogTMAC obteve o maior atraso, cerca de 3% e 8% para baixa densidade e $2,7\%$ e 12% para alta densidade, em relação aos protocolos AhPTMAC e T-MAC respectivamente. Esse incremento é devido ao maior número de trocas de canal do CogTMAC.

No cenário de baixa densidade com ruído alto, o protocolo CogTMAC tem o pior desempenho, com um atraso 15% e 70% superior aos protocolos AhPTMAC e T-MAC, respectivamente.

É importante observar que, para esse cenário, a taxa de entrega é reduzida e o número de trocas de canal do CogTMAC é superior ao AhPTMAC. Entretanto, no cenário de ruído alto e alta densidade, o protocolo CogTMAC continua com um atraso maior, mas a diferença é menos acentuada, cerca de 3% e 9% em relação ao AhPTMAC e T-MAC.

4.2.1.3 Consumo de energia no tráfego periódico

O consumo de energia para as operações do rádio é apresentado na Figura 4.5. Os três protocolos tiveram um consumo de energia muito semelhante para os cenários de baixa e alta densidade. A maior variação ocorreu no cenário de alta densidade com ruído alto, no qual o protocolo T-MAC consumiu cerca de 1% e 1, 41% menos energia que os protocolos AhPTMAC e CogTMAC, respectivamente.

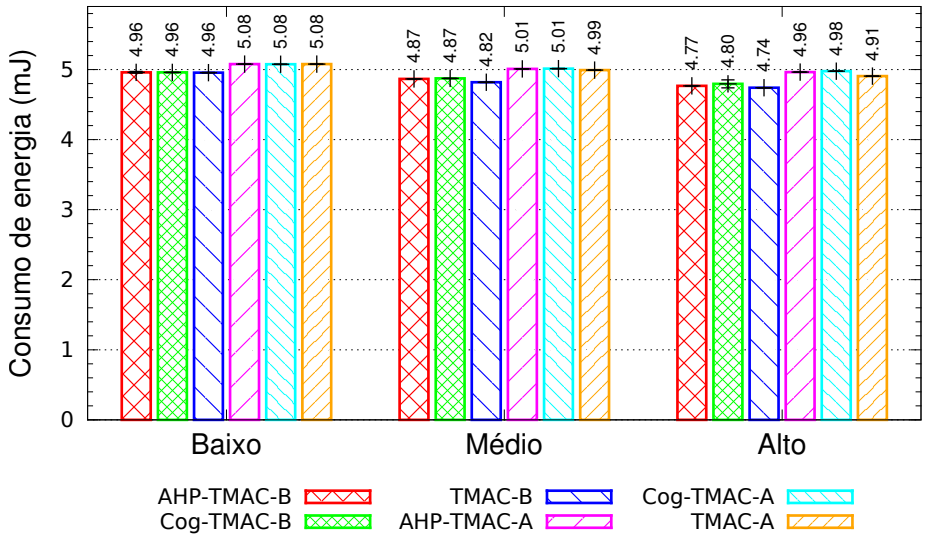

Figura 4.5. Consumo de energia no tráfego periódico.

Esse leve incremento nos protocolos cognitivos se justifica pelas trocas de canais e pela comunicação entre os nós para escolher o melhor canal. Entretanto, a diferença de consumo de energia entre o protocolo T-MAC e os protocolos CogTMAC e AhpTMAC é pouco significativa, considerando-se uma margem de erro máxima de $\pm 0,06$ *mJoule*, para um intervalo de confiança de 99%.

4.2.1.4 Troca de canais no tráfego periódico

A Figura 4.6 mostra o número médio de trocas de canais. Nesta figura apresentamos somente os resultados para os protocolos cognitivos, já que o T-MAC não realiza troca de canais. O protocolo CogTMAC efetuou o maior número de trocas para todos os cenários avaliados.

Para o cenário de baixa densidade de nós o maior número médio de trocas ocorreu para o caso de ruído alto, sendo que o CogTMAC foi superior em 80% ao protocolo AhPTMAC. Apesar do maior número de trocas de canal, o CogTMAC não sofre um incremento significativo das trocas à medida que o ruído é aumentado.

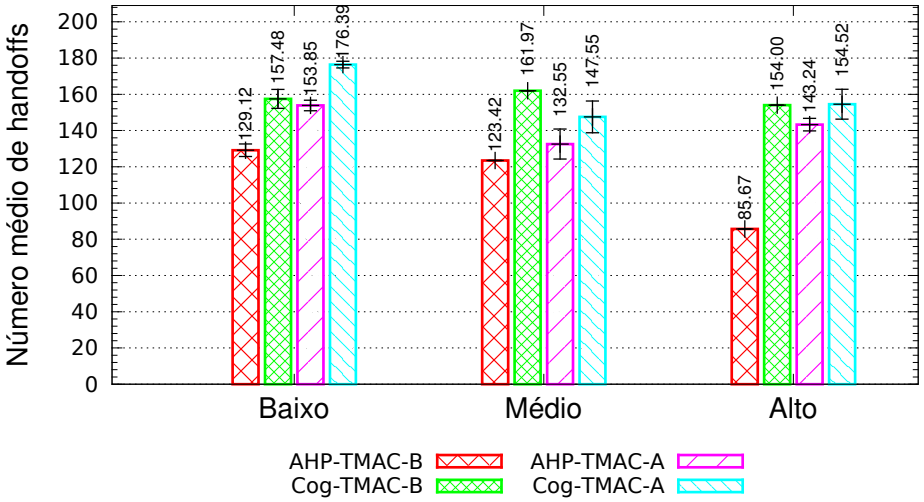

Figura 4.6. Troca de canais (*handoff*) no tráfego periódico.

No cenário de alta densidade o CogTMAC continua a ter o maior número médio de trocas, mas a diferença é menos acentuada. O maior número de trocas ocorreu para ruído baixo, sendo que o CogTMAC efetuou 14% mais trocas que o AhPTMAC. Entretanto, observa-se uma tendência de que o número médio de trocas de canais entre os protocolos seja reduzida à medida que o ruído aumenta.

4.2.1.5 Pacotes retransmitidos no tráfego periódico

O gráfico da Figura 4.7, mostra que em ambos os cenários de densidade de nós, o *overhead* médio de transmissão de pacotes dos protocolos CogTMAC e AhpTMAC é bem maior que o protocolo T-MAC, em todos os níveis de ruído. Além disso, pode-se observar que o desempenho do AhpTMAC é praticamente igual ao do CogTMAC. O maior número de pacotes retransmitidos nas versões cognitivas, se deve ao fato das tentativas de transmissões entre nós dessincronizados.

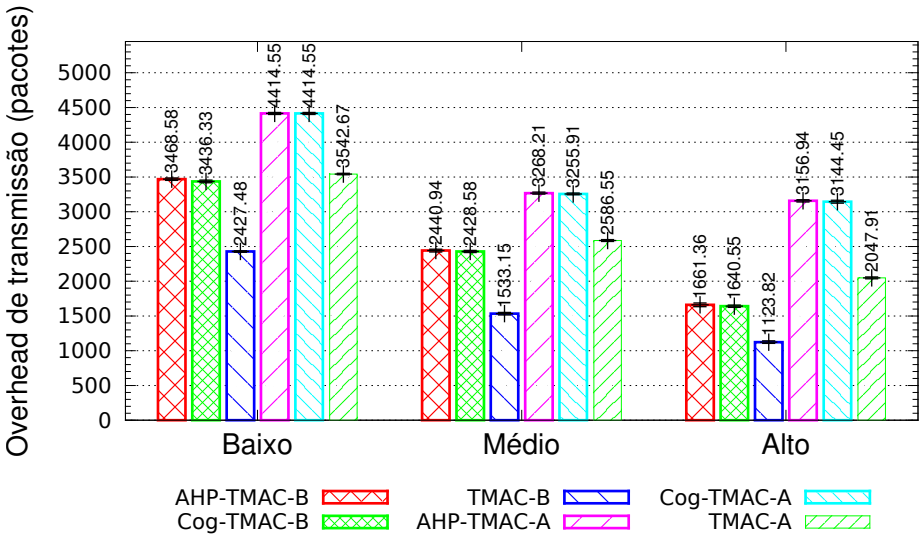

Figura 4.7. Pacotes retransmitidos no tráfego periódico.

4.2.2 Tráfego Exponencial

Nesta seção os protocolos foram avaliados em um cenário no qual o intervalo entre o envio de pacotes de um nó, obedece a uma distribuição de frequência exponencial, conforme visto na Seção 4.1.2. Com relação ao tipo de *deployment*, os nós foram distribuídos de forma equidistante em uma grade de dimensões 6×3. A seguir são exibidos os resultados de desempenho para cada uma das métricas avaliadas, considerando-se também as variações de densidade e ruído.

4.2.2.1 Taxa de entrega no tráfego exponencial

O gráfico apresentado na Figura 4.8, indica que em um cenário de baixa densidade de nós, a diferença entre as taxas médias de entrega de pacotes dos protocolos cognitivos e o T-MAC fica em torno de $0,9\%$ a $3,8\%$, para níveis médios de ruído. Para cenários de ruído alto, considerando-se uma margem de erro entre $\pm 0,56\%$ e $\pm 1,21\%$, a taxa de entrega é cerca de 50% melhor nas versões cognitivas.

Para níveis de ruído baixo, observa-se o mesmo comportamento do tráfego periódico, onde o T-MAC tem melhor taxa de entrega do que suas versões cognitivas. Outra característica em comum com cenário de tráfego periódico , é que o aumento do ruído base no cenário de baixa densidade de nós, faz com que o T-MAC sofra uma diminuição mais acentuada na taxa de entrega de pacotes do que a que ocorre com o CogTMAC e AhpTMAC.

Para o cenário de alta densidade de nós, verifica-se que os níveis de ruído não

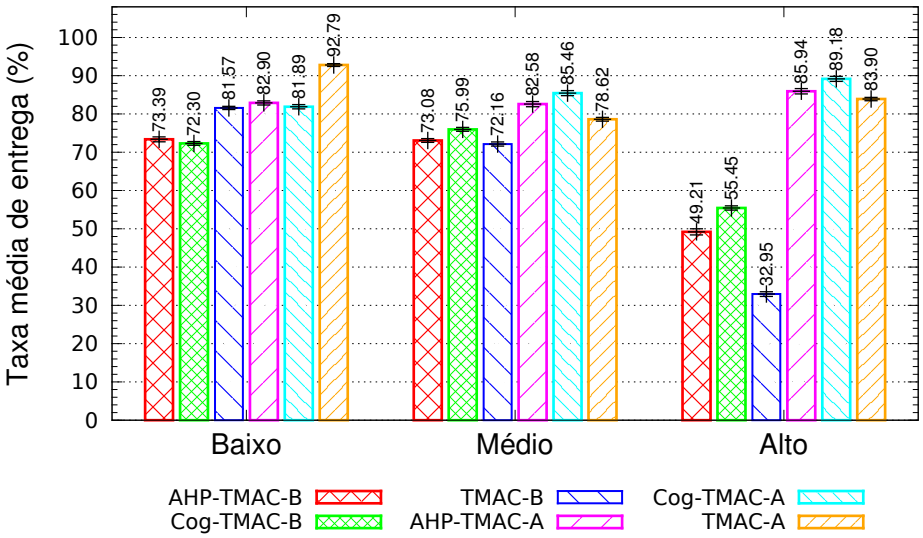

Figura 4.8. Taxa média de entrega de pacotes no tráfego exponencial.

impuseram uma variação acentuada na taxa de entrega dos três protocolos avaliados. Em média, a taxa de entrega dos pacotes atingiu patamares entre $78,6\%$ e $92,8\%$, variando mais em função do número de fluxos simultâneos entre os nós e do protocolo MAC utilizado, do que em função do ruído base.

Ainda no cenário de alta densidade, observa-se também que as taxas de entrega para o CogTMAC e AhpTMAC, onde os níveis de ruído estão entre médio e alto, tem taxas de entrega de 5%cerca a 8% melhor que o T-MAC.

4.2.2.2 Atraso fim a fim no tráfego exponencial

Para um cenário de baixa densidade de nós, o atraso fim a fim médio apresentou valores similares para os três protocolos analisados. A Figura 4.9, mostra que apesar dos protocolos AhpTMAC e CogTMAC apresentarem uma pequena vantagem em relação ao T-MAC, essa diferença fica dentro da margem de erro situada entre ± 2.88 e ± 5.88, para um intervalo de confiança de 99%.

No cenário de alta densidade da rede observa-se comportamento similar ao do cenário de baixa densidade, em todos as configurações de ruído. Assim, os três protocolos apresentaram atrasos fim a fim médios praticamente iguais, se for considerada a margem de erro situada entre entre ± 2.92 e ± 4.64, para um intervalo de confiança de 99%.

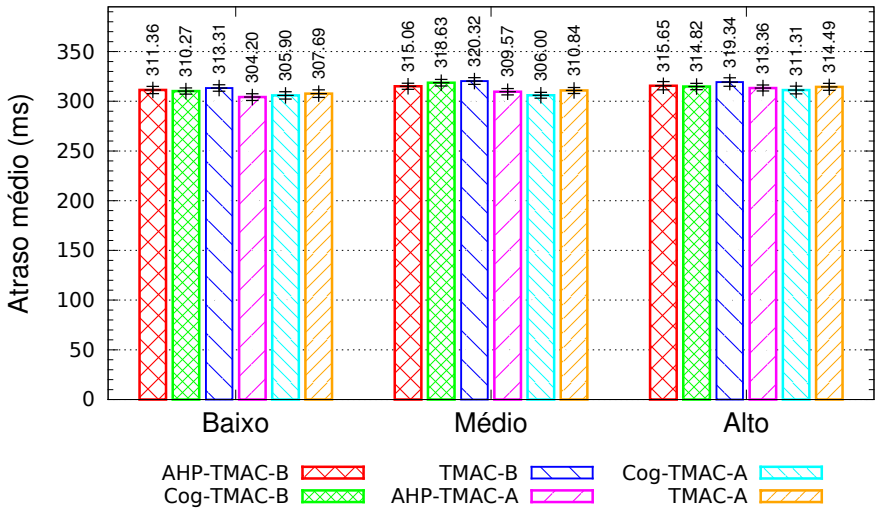

Figura 4.9. Atraso fim a fim no tráfego exponencial.

4.2.2.3 Consumo de energia no tráfego exponencial

Conforme mostra o gráfico da Figura 4.10, tanto em um cenário de baixa densidade quanto em alta densidade de nós, os três protocolos analisados (CogTMAC, AhpTMAC e T-MAC) apresentaram consumo de energia semelhante para as simulações realizadas.

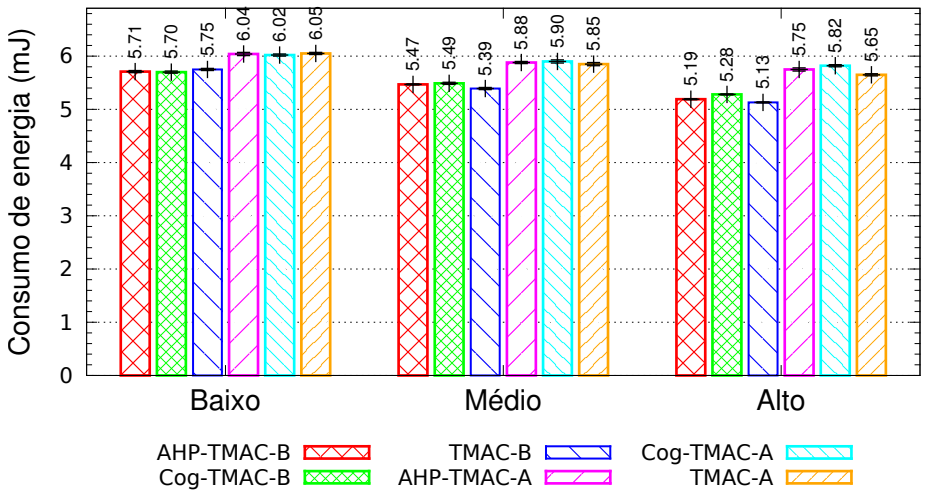

Figura 4.10. Consumo de energia no tráfego exponencial.

4.2.2.4 Troca de canais no tráfego exponencial

Conforme pode ser observado no gráfico da Figura 4.11, em um cenário de baixa densidade de nós, o número de trocas de canais efetuadas pelo protocolo AhpTMAC é menor que o CogTMAC em todos os cenários de ruído. No entanto, a diferença é mais acentuada para o nível de ruído alto, cujo o número de trocas do CogTMAC é cerca de 45% maior que a quantidade de trocas canais efetuadas pelo protocolo AhpTMAC.

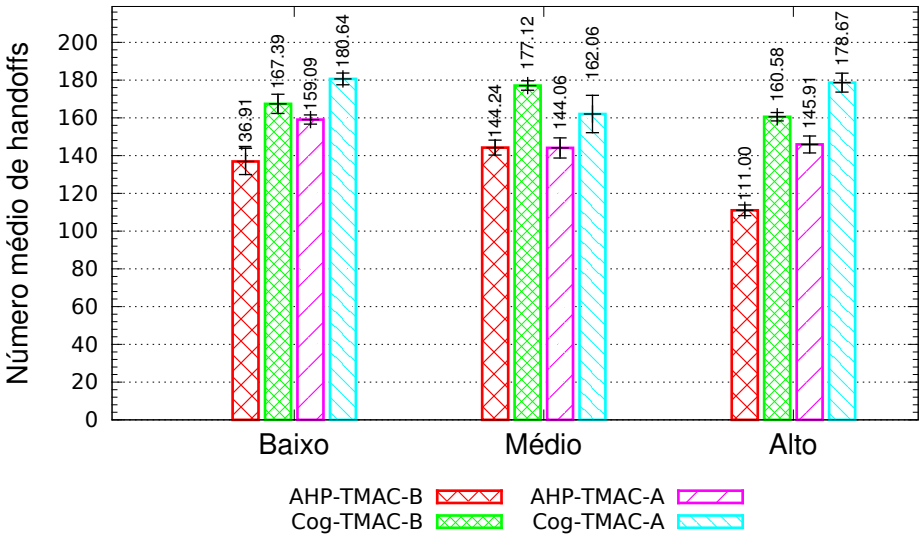

Figura 4.11. Troca de canais no tráfego exponencial.

Em uma área de alta densidade de nós, em todas as configurações de ruído o protocolo AhpTMAC também apresentou um número de trocas menor que o protocolo CogTMAC. Porém essa diferença foi menor do que a percebida no cenário de baixa densidade de nós, sendo que o CogTMAC apresentou cerca 22% mais trocas que o AhpTMAC. Tal comportamento se explica pelo fato do protocolo AhpTMAC levar em consideração uma gama maior de parâmetros que o CogTMAC para decidir se haverá troca de canais. Ou seja, como o CogTMAC se baseia apenas no RSSI, oscilações neste parâmetro são mais sentidas por este protocolo, o que resulta num maior numero de trocas de canais.

4.2.2.5 Pacotes retransmitidos no tráfego exponencial

Conforme pode ser observado na Figura 4.12, em um cenário de baixa densidade de nós, o número total de pacotes retransmitidos pelo protocolo T-MAC é ligeiramente menor que os protocolos CogTMAC e AhpTMAC, para o nível de ruído baixo, considerando-se uma margem de erro situada entre $\pm 25,94$ e $\pm 235,43$ pacotes de diferença. Nas

situações em que o nível ruído do canal atinge patamares de médio a alto o T-MAC também apresenta desempenho muito próximo aos dos protocolos AhpTMAC e Cog-MAC.

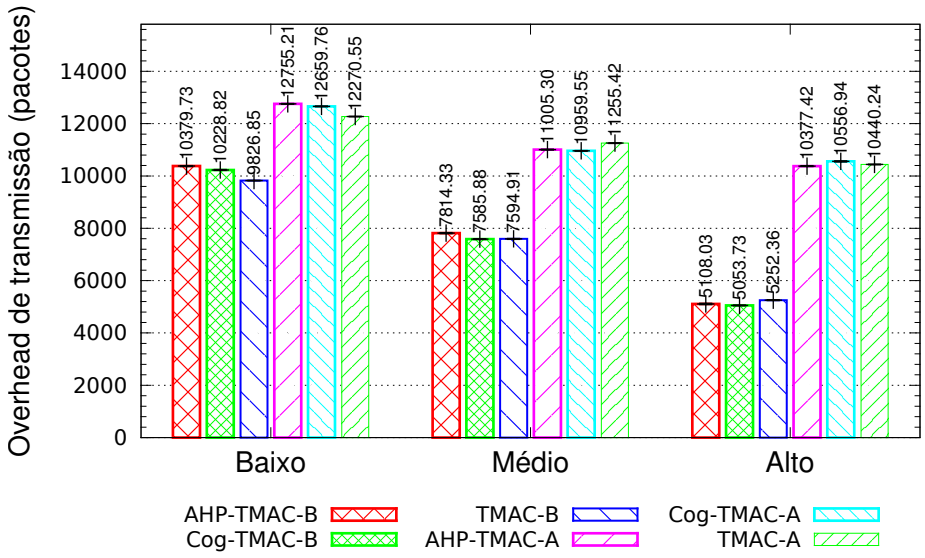

Figura 4.12. Pacotes Retransmitidos no tráfego exponencial.

Para um cenário de alta densidade de nós, observa-se no gráfico da Figura 4.12, que o desempenho dos três protocolos avaliados é similar, no que se refere ao número de pacotes retransmitidos. Pois considerando-se uma margem de erro situada entre ± 27.55 e ± 334.81 pacotes retransmitidos, pode-se afirmar que a diferença entre os protocolos comparados é insignificante.

4.3 Discussão dos Resultados

À partir dos resultados apresentados anteriormente, foi feita uma análise para identificar os comportamentos semelhantes entre as métricas avaliadas para cada cenário. Além disso, são discutidas questões de escalabilidade e potencial para prototipação em aplicações específicas das versões cognitivas do protocolo T-MAC.

4.3.1 Discussão do Desempenho nas Métricas Avaliadas

De acordo com a combinação de cenários proposta na Tabela 4.1, observou-se comportamentos similares para as métricas avaliadas. Tais comportamentos são descritos a seguir.

4.3.1.1 Taxa de entrega

Para todos os cenários avaliados as taxas médias de entrega de pacotes dos protocolos cognitivos (CogTMAC e AhpTMAC), apresentaram desempenho melhor que o protocolo T-MAC para níveis ruído médio e alto. O T-MAC superou as versões cognitivas apenas quando o nível de ruído foi baixo.

Outro ponto comum a todos os cenários avaliados foi que os protocolos CogTMAC e AhpTMAC apresentaram variações menores que o T-MAC nas taxas de entrega de pacotes, quando os níveis de ruído foram aumentados.

Observou-se também em todos os cenários avaliados que a baixa densidade de nós é mais suscetível a ação do ruído, pois os nós estão mais distantes. Assim, a taxa de entrega para níveis de ruído alto foi bem menor em cenários de baixa densidade de nós, se comparada aos cenários de alta densidade de nós.

4.3.1.2 Atraso Fim a Fim

Conforme descrito na seção 2.4.3, o protocolo T-MAC apresenta baixas variações no atraso fim a fim. Isso ficou comprovado nos resultados obtidos. Outro situação que se repetiu na maioria dos cenários avaliados, foi que o atraso fim a fim medido apresentou valores similares para os três protocolos analisados. A explicação para isto é que o CogTMAC e AhpTMAC, são adaptações do protocolo T-MAC, que não alteraram o ciclo de operação imposto pelo protocolo original.

4.3.1.3 Consumo de Energia

Com relação a métrica de consumo de energia, conforme citado anteriomente foi considerado apenas o consumo inerente a transmissão, sendo desconsiderado o consumo relativo ao processamento. Assim, em todos os cenários avaliados, a energia consumida em cada um dos protocolos avaliados foi muito parecida.

No entanto, o protocolo T-MAC apresentou ligeira vantagem em relação aos demais. Esse consumo um pouco maior dos protocolos cognitivos se justifica pelo maior número de trocas de canais efetuadas, uma vez que esses protocolos trocam de canal de tempos em tempos, em função dos parâmetros de qualidade do canal, mas isso é compensado pela melhor taxa de entrega. Dessa forma, em termos de eficiência energética, os protocolos cognitivos são melhores.

4.3.1.4 Número de Trocas de Canais

Conforme dito anteriormente, o T-MAC não realiza trocas de canais. Assim no comparativo feito entre os protocolos cognitivos, em todos os cenários avaliados o número de trocas realizados pelo AhpTMAC foi inferior ao número de trocas efetuadas pelo protocolo CogTMAC.

A explicação para esse comportamento é que para o protocolo AhpTMAC o método AHP atribui pesos para os parâmetros de qualidade do canal (Ruído Base, RSSI, SINR, Delay) na escolha do menor canal. Dessa forma, acontecem menos trocas em função da variação de RSSI, já que esse parâmetro contribui parcialmente para a decisão de qual canal utilizar, devido ao peso atribuído a esse parâmetro no método de decisão AHP.

4.3.1.5 Quantidade de Pacotes Retransmitidos

Para todos os cenários avaliados o número de pacotes retransmitidos foi maior nas versões cognitivas do protocolo T-MAC. No entanto em alguns cenários de baixa densidade de nós, o número total de pacotes retransmitidos pelo protocolo T-MAC foi praticamente igual aos dos protocolos CogTMAC e AhpTMAC, quando o nível de ruído apresentado era baixo.

4.3.2 Escalabilidade da Solução e Viabilidade de Prototipação

A avaliação da escalabilidade visa identificar o impacto do crescimento do número de nós pertencentes a RSSFC, avaliando se a solução proposta é sensível ao número de nós e se as métricas de desempenho citadas anteriormente são afetadas por esse crescimento. Os métodos de decisão propostos foram implementados em algoritmos distribuídos e conforme visto na análise de complexidade feita na seção 3.3.5, de maneira geral não haveria problemas de escalabilidade na solução proposta.

Considerando-se os pacotes retransmitidos (*overhead* de transmissão) e o número de trocas, a abordagem apesar de distribuída pode enfrentar problemas com as constantes trocas e necessidade de retransmissão. A maior quantidade de fluxos simultâneos, também teve grande impacto nessas métricas, pois existe um aumento do *overhead* de pacotes de controles necessários para escolha do melhor canal e para o sincronismo de nós ouvindo em canais diferentes.

Apesar dos valores absolutos coletados para essa métrica serem piores do que os obtidos para o T-MAC, a expectativa é que para redes maiores essa abordagem se

sobressaia, já que a porcentagem de pacotes entregues permaneceu inalterada independente dos cenários de densidade variada.

Com relação ao AhpTMAC, a restrição é que algoritmo apresenta cálculos de entropia que utilizam operações de ponto flutuante (cálculos baseados em logaritmos), o que pode ser um empecilho para uma possível prototipação, já que o hardware dos nós sensores atuais não apresentam a opção para trabalhar com ponto flutuante.

Portanto, a decisão de qual desses protocolos adotar, considerando o tamanho e a topologia da rede, deve ser feito considerando uma aplicação específica. O que se procurou neste trabalho foi testar a solução em diverso cenários de densidade, ruído, tráfego e *deployment*. Assim o desenvolvimento de protocolo de propósito geral em RSSF é um grande desafio, o que se percebe atualmente é uma tendência de se desenvolver protocolos para aplicações específicas.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho foi proposta uma adapatação cognitiva do protocolo T-MAC, por meio de mecanismos distribuídos de alocação dinâmica de canais, com foco em decisão do espectro para redes de sensores sem fio cognitivas (RSSFC).

Porém, além dos aspectos relacionados a decisão do espectro, as técnicas de implementação seguiram as demais fases do arcabouço de gerenciamento do espectro: sensoriamento, compartilhamento e mobilidade do espectro.

Tal adaptação, deu origem a dois mecanismos distribuídos de seleção dinâmica do melhor canal disponível: CogTMAC e AhpTMAC. O primeiro se baseia na escolha do melhor canal em função da força do sinal recebido (RSSI); enquanto que o segundo além de utilizar o RSSI, coleta os parâmetros de SINR, ruído base e atraso fim a fim, escolhendo o melhor canal com base em uma técnica clássica de tomada de decisão conhecida como *Analytical Hierarchical Process* (AHP). Ambos os métodos foram adicionados ao protocolo T-MAC disponível no simulador de RSSF Castalia.

A avaliação desses métodos, levou em consideração variações na densidade e distribuição dos nós, bem como diferentes níveis de ruído e classes de tráfego periódico, exponencial e aleatório. Com base nestes cenários foram avaliadas as métricas de taxa de entrega, latência, *overhead* de transmissão, número de trocas e consumo de energia.

As simulações mostraram que as adaptações cognitivas do protocolo T-MAC, denominadas neste trabalho CogTMAC e AhpTMAC, apresentaram um desempenho de até 69% mais pacotes entregues que a versão original do protocolo. Com relação ao atraso fim a fim percebeu-se que não houve melhora sensível, mas isto pode ser reflexo do T-MAC apresentar baixas variações de latência. Avaliando-se as métricas consumo de energia, as versões cognitivas, apresentam valores bem próximos ao do T-MAC, cuja a grande vantagem é o desempenho medido nessa métrica. Com relação ao número de trocas de canais, o AhpTMAC foi melhor, pois não considera apenas as variações de

intensidade de sinal.

Outra constatação, foi que o CogTMAC, mesmo sendo um modelo de decisão baseado apenas em um único parâmetro físico do canal (RSSI), apresentou um melhor desempenho que o AhpTMAC nas taxas de pacotes entregues. Porém essa questão cabe ser investigada mais a fundo, já que Akyildiz et al. [2008] afirmam que um modelo de decisão não deve baseado apenas em parâmetros físicos do canal e que somente esses parâmetros não são suficientes para escolher o melhor canal. Dessa forma, observou-se que a adição de parâmetros da aplicação como atraso fim a fim no método de decisão proposto no AhpTMAC, não implicou em melhora de desempenho das métricas avaliadas.

Como proposta de trabalhos futuros vislumbra-se implementações do protocolo de decisão proposto, baseando-se em outros protocolos MAC , como por exemplo o B-MAC que apresenta estrutura modular. Outra alternativa, é testar outros métodos mais complexos de classificação de canal com base na predição e algoritmos evolucionários, como alternativa ao AHP na fase de escolha de canal.

Outra situação de trabalho futuro que se deparou nessa pesquisa, foi a necessidade de implementação de um algoritmo de roteamento, que permita a avaliação dos métodos de decisão em redes multissaltos, no intuito de analisar a interação da subcamada MAC com a camada de rede.

Por fim, a pesquisa pode ser extrapolada para levar em consideração uma série de oportunidades e desafios que foram identificados ao longo do trabalho:

- testes com múltiplos rádios, uma vez que foi implementado um módulo controlador de rádio, mas apenas o rádio de 2.4 Ghz foi utilizado;
- controle de potência dos nós, no intuito de restringir a interferência causada pelos nós;
- cenários de mobilidade dos nós;
- testes em nós sensores reais, para verificar a viabilidade de prototipação dos nós.

Anexo A

Modulo CognitiveMAC Desenvolvido no Castalia

```
/***************************************************************************
 *  Copyright: National ICT Australia,  2007 - 2010                        *
 *  Developed at the ATP lab, Networked Systems research theme             *
 *  Author(s): Athanassios Boulis, Yuriy Tselishchev                       *
 *  This file is distributed under the terms in the attached LICENSE file.  *
 *  If you do not find this file, copies can be found by writing to:        *
 *                                                                         *
 *      NICTA, Locked Bag 9013, Alexandria, NSW 1435, Australia            *
 *      Attention:  License Inquiry.                                       *
 *                                                                         *
 ***************************************************************************/

#include <cmath>
#include "CMAC.h"

Define_Module (CMAC);

void
CMAC::startup ()
{
  printStateTransitions = par ("printStateTransitions");
  ackPacketSize = par ("ackPacketSize");
  //just convert msecs to secs
  frameTime = ((double) par ("frameTime")) / 1000.0;
  syncPacketSize = par ("syncPacketSize");
  rtsPacketSize = par ("rtsPacketSize");
  ctsPacketSize = par ("ctsPacketSize");
```

```
  resyncTime = par ("resyncTime");
  allowSinkSync = par ("allowSinkSync");
  // just convert msecs to secs
  contentionPeriod = ((double) par ("contentionPeriod")) / 1000.0;
 // TA: just convert msecs to secs (15ms default);
  listenTimeout = ((double) par ("listenTimeout")) / 1000.0;
  waitTimeout = ((double) par ("waitTimeout")) / 1000.0;
  useFRTS = par ("useFrts");
  useRtsCts = par ("useRtsCts");
  useAHP = par ("useAHP");
  maxTxRetries = par ("maxTxRetries");

  disableTAextension = par ("disableTAextension");
  conservativeTA = par ("conservativeTA");
  collisionResolution = par ("collisionResolution");
  if (collisionResolution != 2 && collisionResolution != 1
      && collisionResolution != 0)
    {
      trace () <<
        "Unknown value for parameter 'collisionResolution', will default to 1";
      collisionResolution = 1;
    }
  //Initialise state descriptions used in debug output
  if (printStateTransitions)
    {
      stateDescr[100] = "MAC_STATE_SETUP";
      stateDescr[101] = "MAC_STATE_SLEEP";
      stateDescr[102] = "MAC_STATE_ACTIVE";
      stateDescr[103] = "MAC_STATE_ACTIVE_SILENT";
      stateDescr[104] = "MAC_STATE_IN_TX";
      stateDescr[110] = "MAC_CARRIER_SENSE_FOR_TX_RTS";
      stateDescr[111] = "MAC_CARRIER_SENSE_FOR_TX_DATA";
      stateDescr[114] = "MAC_CARRIER_SENSE_FOR_TX_SYNC";
      stateDescr[115] = "MAC_CARRIER_SENSE_BEFORE_SLEEP";
      stateDescr[120] = "MAC_STATE_WAIT_FOR_DATA";
      stateDescr[121] = "MAC_STATE_WAIT_FOR_CTS";
      stateDescr[122] = "MAC_STATE_WAIT_FOR_ACK";
    }

  phyDataRate = par ("phyDataRate");
  phyDelayForValidCS = (double) par ("phyDelayForValidCS") / 1000.0;
  //parameter given in ms in the omnetpp.ini
  phyLayerOverhead = par ("phyFrameOverhead");
```

```
//try to obtain the value of isSink parameter from application module
if (getParentModule ()->getParentModule ()->
    findSubmodule ("nodeApplication") != -1)
  {
    cModule *tmpApplication =
      getParentModule ()->getParentModule ()->
      getSubmodule ("nodeApplication");
    isSink =
      tmpApplication->hasPar ("isSink") ? tmpApplication->
      par ("isSink") : false;
  }
else
  {
    isSink = false;
  }

syncPacket = NULL;
rtsPacket = NULL;
ackPacket = NULL;
ctsPacket = NULL;

macState = MAC_STATE_SETUP;
scheduleTable.clear ();
primaryWakeup = true;
needResync = 0;
currentFrameStart = -1;
activationTimeout = 0;

declareOutput ("Sent packets breakdown");

if (isSink && allowSinkSync)
  setTimer (SYNC_CREATE, 0.1);
else
  setTimer (SYNC_SETUP, allowSinkSync ? 3 * frameTime : 0.1);
//setTimer(TEST_NEIGHBORS,1);
radioModule =
  check_and_cast <
  Radio *
  >(getParentModule ()->
    getSubmodule ("Radio", radioController->getActiveRadio ()));
ChannelList = radioModule->getChannelList ();
best = radioModule->getChannel ();
//best->state=STATE_BUSY;
```

```
  toRadioLayer (createRadioCommand (SET_CS_INTERRUPT_ON));
  //ativa os temporizadores de descoberta de vizinhos
  setTimer (FIND_NEIGHBORS, (self / 2) + 101);
  setTimer (FIND_NEIGHBORS_2_HOPS, (self / 2) + 119);

  setTimer (CHOOSE_CHANNEL, (self * 3) + 130);
  setTimer (CHANGE_CHANNEL, 200);
  setTimer (TRACE_CHANNEL, 200);
}

void
CMAC::timerFiredCallback (int timer)
{
  switch (timer)
    {
    case TRACE_CHANNEL:
      {
        trace () << "Canal=" << best->channel << " Radio=" << radioModule->
          getChannel ()->channel;
        setTimer (TRACE_CHANNEL, 1);
        break;
      }
    case CHANGE_CHANNEL:
      {
        toRadioLayer (createRadioCommand
                      (SET_CHANNEL, (int) (ChannelToChange)));
        trace () << "Channel changed, new channel is " << ChannelToChange;
        //handoff++;
        break;
      }
    case RESTORE_CHANNEL:
      {
        if (best->channel != radioModule->getChannel ()->channel)
          {
            toRadioLayer (createRadioCommand (SET_CHANNEL, best->channel));
          }
        break;
      }
      //temporizador que envia mensagem para descoberta de vizinhos
    case FIND_NEIGHBORS:
      {
        trace () << "Sent mac packet to find neighbors";

        //cria um pacote da camada MAC do tipo FIND_NEIGHBORS_CMAC_PACKET
```

```
    CMacPacket *macPkt =
      new CMacPacket ("cognitiveMAC data packet", MAC_LAYER_PACKET);
    macPkt->setType (FIND_NEIGHBORS_CMAC_PACKET);
    macPkt->setSource (SELF_MAC_ADDRESS);
    macPkt->setDestination (BROADCAST_MAC_ADDRESS);
    macPkt->setSequenceNumber (txSequenceNum);
    macPkt->setNav (TX_TIME (ctsPacketSize));
    macPkt->setByteLength (ctsPacketSize);

    //adiciona pacote no buffer
    if (bufferPacket (macPkt))
      {                     // this is causing problems
        if (TXBuffer.size () == 1)
          checkTxBuffer ();
      }
    else
      {
        // cancelAndDelete(macPkt);
        //We could send a control message to upper layer to inform of full buffer
      }
    break;
  }
  //temporizador que envia mensagem para descoberta de vizinhos a 2 hops
case FIND_NEIGHBORS_2_HOPS:
  {
    //envia uma mensagem com a sua lista de vizinhos
    if (!neighbors.isEmpty ())
      {
        CMacPacket *macPkt =
          new CMacPacket ("cognitiveMAC data packet", MAC_LAYER_PACKET);
        macPkt->setType (NEIGHBOR_LIST_CMAC_PACKET);
        macPkt->setSource (SELF_MAC_ADDRESS);
        macPkt->setDestination (BROADCAST_MAC_ADDRESS);
        macPkt->setNeighborListArraySize (neighbors.getSize ());
        neighbor *n = neighbors.getHead ()->getNext ();
        for (int i = 0; i < neighbors.getSize (); i++)
          {
            macPkt->setNeighborList (i, n->getAdress ());
            n = n->getNext ();
          }
        macPkt->setSequenceNumber (txSequenceNum);
        macPkt->setNav (TX_TIME (ctsPacketSize));
        macPkt->setByteLength (ctsPacketSize);
```

```
          //adiciona pacote no buffer
          if (bufferPacket (macPkt))
            {                 // this is causing problems
              if (TXBuffer.size () == 1)
                checkTxBuffer ();
            }
          else
            {
              // cancelAndDelete(macPkt);
              //We could send a control message to upper layer to inform of full buffer
            }
        }

      break;
    }
  case CHOOSE_CHANNEL:
    {
      channelNowFree = best;
      if (!useAHP && PacketList.size () < 2)
        {                     //&& PacketList.size() < 2){
          //verfica o canal com menor RSSI e seleciona como o melhor canal
          list < Channel_type >::iterator c;
          for (c = ChannelList->begin (); c != ChannelList->end (); c++)
            {
              trace () << "RSSI do canal " << c->
                channel << " = " << addPower_dBm (c->RSSI,
                                                  c->
                                                  noiseFloor) << " state "
                << c->state;
              if (c->state == STATE_CLEAR)
                {
                  if (addPower_dBm (c->RSSI, c->noiseFloor) <=
                      addPower_dBm (best->RSSI, best->noiseFloor))
                    {
                      best = c;
                    }
                }
            }
        }
      else
        {
          //chama ahp
          double somaColunas[4];
          trace () << "Entrada do ahp";
```

```
    list < packet_type >::iterator pkt = PacketList.begin ();
    pesosAtuais = ahp (PacketList, 4, somaColunas);
    PacketList.clear ();
    //usa os pesos para calcular a qualidade do canal
    list < Channel_type >::iterator c;
    for (c = ChannelList->begin (); c != ChannelList->end (); c++)
      {
        c->channelQuality =
          pesosAtuais.metric[rssi] * (100 -
                                      ((c->RSSI / somaColunas[rssi]) *
                                       100)) +
          pesosAtuais.metric[noisefloor] *
          ((c->noiseFloor / somaColunas[noisefloor]) * 100) +
          pesosAtuais.metric[snir] * ((c->SINR / somaColunas[snir]) *
                                      100) +
          pesosAtuais.metric[delay] * (100 -
                                       ((c->delay.dbl () /
                                         somaColunas[delay]) * 100));
      }

    //escolhe o melhor canal a partir da qualidade do canal
    for (c = ChannelList->begin (); c != ChannelList->end (); c++)
      {
        trace () << "Canal " << c->channel << " state " << c->state;
        if (c->state == STATE_CLEAR)
          {
            if (c->channelQuality >= best->channelQuality)
              {
                best = c;
              }
          }
      }
    trace () << "Melhor canal = " << best->channel;
  }


ChannelToChange = best->channel;
if (channelNowFree != best)
  handoff++;
trace () << "Best channel =" << best->channel;
if (neighbors.getSize () <= 1)
  break;

//-----------Notificacao do canal escolhido
```

```
neighborList *resendBroadcast = new neighborList ();

//marca todos os nos como nao visitados
neighbors.clearVisited ();
twoHopNeighbors.clearVisited ();

//os nos marcados indicam nos que serao notificados

//enquanto todos os nos nao forem visitados
while (!neighbors.allVisited () && !twoHopNeighbors.allVisited ())
  {

    //escolhe o vizinho com maior grau de conectividade para ser visitado no momento
    neighbor *higher = neighbors.higherDegreeNotVisited ();
    if (higher == NULL)
      break;
    //marca como visitado
    higher->setVisited (true);

    //twoHopNeighborsFromSource eh a lista de vizinhos a 1 hop do no sendo visitado agora
    neighborList *twoHopNeighborsFromSource =
      twoHopNeighbors.neighborOfSource (higher->getAdress ());

    if (!twoHopNeighborsFromSource->isEmpty ())
      {
        neighbor *n1;
        neighbor *n2 = twoHopNeighbors.getHead ()->getNext ();

        //todos os nos que estao na lista de vizinhos a dois hops e na lista
        //twoHopNeighborsFromSource serao notificados pelo no sendo visitado no momento
        for (int i = 0; i < twoHopNeighbors.getSize (); i++)
          {
            n1 = twoHopNeighborsFromSource->getHead ()->getNext ();
            for (int j = 0; j < twoHopNeighborsFromSource->getSize ();
                 j++)
              {
                if (n1->getAdress () == n2->getAdress ())
                  {
                    n2->setVisited (true);
                  }
                n1 = n1->getNext ();
              }
            n2 = n2->getNext ();
          }
```

```
          //o no com maior grau de conectividade sendo visitado no momento
          //eh adicionado na lista de reencaminhadores de broadcast
          resendBroadcast->add (higher->getAdress ());
        }
    }

  /* Cria um pacote da camada mac contendo o canal a ser notificado como ocupado e
   * a lista de reencaminhadores de broadcast*/
  if (!neighbors.isEmpty ())
    {
      CMacPacket *macPkt =
        new CMacPacket ("cognitiveMAC data packet", MAC_LAYER_PACKET);
      macPkt->setType (UPDATE_CHANNEL_1_HOP_PACKET);
      macPkt->setSource (SELF_MAC_ADDRESS);
      macPkt->setDestination (BROADCAST_MAC_ADDRESS);
      macPkt->setNeighborListArraySize (resendBroadcast->getSize ());
      neighbor *n = resendBroadcast->getHead ()->getNext ();
      for (int i = 0; i < resendBroadcast->getSize (); i++)
        {
          macPkt->setNeighborList (i, n->getAdress ());
          n = n->getNext ();
        }
      macPkt->setChannelNowFree (channelNowFree->channel);
      macPkt->setNotifyingNode (self);
      macPkt->setNotifiedChannel (best->channel);
      macPkt->setSequenceNumber (txSequenceNum);
      macPkt->setNav (TX_TIME (ctsPacketSize));
      macPkt->setByteLength (ctsPacketSize);

      if (bufferPacket (macPkt))
        {                 // this is causing problems
          if (TXBuffer.size () == 1)
            checkTxBuffer ();
        }
      else
        {

          // cancelAndDelete(macPkt);
          //We could send a control message to upper layer to inform of full buffer
        }
    }
  setTimer (CHOOSE_CHANNEL, 30);
  break;
}
```

```
case SYNC_SETUP:
  {
    /* Timeout to hear a schedule packet has expired at this stage,
     * MAC is able to create its own schedule after a random offset
     * within the duration of 1 frame
     */
    if (macState == MAC_STATE_SETUP)
      setTimer (SYNC_CREATE, genk_dblrand (1) * frameTime);
    break;
  }

case SYNC_CREATE:
  {
    /* Random offset selected for creating a new schedule has expired.
     * If at this stage still no schedule was received, MAC creates
     * its own schedule and tries to broadcast it
     */
    if (macState == MAC_STATE_SETUP)
      createPrimarySchedule ();
    break;
  }

case SYNC_RENEW:
  {
    /* This node is the author of its own primary schedule
     * It is required to rebroadcast a SYNC packet and also
     * schedule a self message for the next RESYNC procedure.
     */
    trace () << "Initiated RESYNC procedure";
    scheduleTable[0].SN++;
    needResync = 1;
    setTimer (SYNC_RENEW, resyncTime);
    break;
  }

case FRAME_START:
  {
    /* primaryWakeup variable is used to distinguish between primary and secondary schedules
     * since the primary schedule is always the one in the beginning of the frame, we set
     * primaryWakeup to true here.
     */
    primaryWakeup = true;

    // record the current time and extend activation timeout
```

```
    currentFrameStart = activationTimeout = getClock ();
    extendActivePeriod ();

    // schedule the message to start the next frame. Also check for frame offsets
    // (if we received a RESYNC packet, frame start time could had been shifted due to
    // clock drift - in this case it is necessary to rebroadcast this resync further)
    setTimer (FRAME_START, frameTime);
    if (scheduleTable[0].offset != 0)
      {
        //trace() << "New frame started, shifted by " << scheduleTable[0].offset;
        scheduleTable[0].offset = 0;
        needResync = 1;
      }
    // schedule wakeup messages for secondary schedules within the current frame only
    for (int i = 1; i < (int) scheduleTable.size (); i++)
      {
        if (scheduleTable[i].offset < 0)
          {
            scheduleTable[i].offset += frameTime;
          }
        setTimer (WAKEUP_SILENT + i, scheduleTable[i].offset);
      }

    // finally, if we were sleeping, need to wake up the radio. And reset the internal MAC
    // state (to start contending for transmissions if needed)
    if (macState == MAC_STATE_SLEEP)
      toRadioLayer (createRadioCommand (SET_STATE, RX));
    if (macState == MAC_STATE_SLEEP || macState == MAC_STATE_ACTIVE
        || macState == MAC_STATE_ACTIVE_SILENT)
      {
        resetDefaultState ("new frame started");
      }
    break;
  }

case CHECK_TA:
  {
    /* Activation timeout fired, however we may need to extend the timeout
     * here based on the current MAC state, or if there is no reason to
     * extend it, then we need to go to sleep.
     */

    //if disableTAextension is on, then we will behave as SMAC - simply go
    //to sleep if the active period is over
```

```
    if (disableTAextension)
      {
        primaryWakeup = false;
        // update MAC and RADIO states
        toRadioLayer (createRadioCommand (SET_STATE, SLEEP));
        setMacState (MAC_STATE_SLEEP, "active period expired (SMAC)");
      }
    //otherwise, check MAC state and extend active period or go to sleep
    else if (conservativeTA)
      {
        if (macState != MAC_STATE_ACTIVE
            && macState != MAC_STATE_ACTIVE_SILENT
            && macState != MAC_STATE_SLEEP)
          {
            extendActivePeriod ();
          }
        else
          {
            performCarrierSense (MAC_CARRIER_SENSE_BEFORE_SLEEP);
          }
      }

    else
      {
        primaryWakeup = false;
        // update MAC and RADIO states
        toRadioLayer (createRadioCommand (SET_STATE, SLEEP));
        setMacState (MAC_STATE_SLEEP, "active period expired (CMAC)");
      }
    break;
  }

case CARRIER_SENSE:
  {
    /* First it is important to check for valid MAC state
     * If we heard something on the radio while waiting to start carrier sense,
     * then MAC was set to MAC_STATE_ACTIVE_SILENT. In this case we can not transmit
     * and there is no point to perform carrier sense
     */
    if (macState == MAC_STATE_ACTIVE_SILENT
        || macState == MAC_STATE_SLEEP)
      break;

    // At this stage MAC can only be in one of the states MAC_CARRIER_SENSE_...
```

```
if (macState != MAC_CARRIER_SENSE_FOR_TX_RTS
    && macState != MAC_CARRIER_SENSE_FOR_TX_SYNC
    && macState != MAC_CARRIER_SENSE_FOR_TX_DATA
    && macState != MAC_CARRIER_SENSE_BEFORE_SLEEP)
  {
    trace () <<
      "WARNING: bad MAC state for MAC_SELF_PERFORM_CARRIER_SENSE";
    break;
  }
radioModule =
  check_and_cast <
  Radio *
  >(getParentModule ()->
    getSubmodule ("Radio", radioController->getActiveRadio ()));
switch (radioModule->isChannelClear ())
  {

  case CLEAR:
    {
      carrierIsClear ();
      break;
    }

  case BUSY:
    {
      carrierIsBusy ();
      break;
    }

  case CS_NOT_VALID_YET:
    {
      setTimer (CARRIER_SENSE, phyDelayForValidCS);
      break;
    }

  case CS_NOT_VALID:
    {
      if (macState != MAC_STATE_SLEEP)
        {
          toRadioLayer (createRadioCommand (SET_STATE, RX));
          setTimer (CARRIER_SENSE, phyDelayForValidCS);
        }
      break;
    }
```

```
        }
      break;
    }

  case TRANSMISSION_TIMEOUT:
    {
      resetDefaultState ("timeout");
      //toRadioLayer(createRadioCommand(SET_CHANNEL, best->channel));
      break;
    }

  case WAKEUP_SILENT:
    {
      /* This is the wakeup timer for secondary schedules.
       * here we only wake up the radio and extend activation timeout for listening.
       * NOTE that default state for secondary schedules is MAC_STATE_ACTIVE_SILENT
       */

      activationTimeout = getClock ();
      extendActivePeriod ();
      if (macState == MAC_STATE_SLEEP)
        {
          toRadioLayer (createRadioCommand (SET_STATE, RX));
          resetDefaultState ("secondary schedule starts");
        }
      break;
    }

  default:
    {
      int tmpTimer = timer - WAKEUP_SILENT;
      if (tmpTimer > 0 && tmpTimer < (int) scheduleTable.size ())
        {
          activationTimeout = getClock ();
          extendActivePeriod ();
          if (macState == MAC_STATE_SLEEP)
            {
              toRadioLayer (createRadioCommand (SET_STATE, RX));
              resetDefaultState ("secondary schedule starts");
            }
        }
      else
        {
          trace () << "Unknown timer " << timer;
```

```
          }
      }
    }
}

int
CMAC::handleRadioControlMessage (cMessage * msg)
{
  RadioControlMessage *radioMsg =
    check_and_cast < RadioControlMessage * >(msg);
  if (radioMsg->getRadioControlMessageKind () == CARRIER_SENSE_INTERRUPT)
    carrierIsBusy ();

  toNetworkLayer (msg);
  return 1;
}

void
CMAC::fromNetworkLayer (cPacket * netPkt, int destination)
{
  // Create a new MAC frame from the received packet and buffer it (if possible)
  CMacPacket *macPkt = new CMacPacket ("CMAC data packet", MAC_LAYER_PACKET);
  macPkt->setType (DATA_CMAC_PACKET);
  macPkt->setSource (SELF_MAC_ADDRESS);
  macPkt->setTimestamp (simTime ());
  if (!neighbors.isEmpty ())
    {
      if (!neighbors.isMyNeghbor (destination))
        {
          trace () << "AFF Nao esta na lista de vizinhos";
        }
      else
        {
          trace () << "AEE Esta na lista de vizinhos";
        }
    }
  macPkt->setDestination (destination);
  macPkt->setSequenceNumber (txSequenceNum);
  encapsulatePacket (macPkt, netPkt);

  if (bufferPacket (macPkt))
    {                           // this is causing problems
      if (TXBuffer.size () == 1)
        checkTxBuffer ();
```

```
    }
  else
    {
      // cancelAndDelete(macPkt);
      //We could send a control message to upper layer to inform of full buffer
    }
}

void
CMAC::finishSpecific ()
{
  if (packetsSent.size () > 0)
    {
      trace () << "Sent packets breakdown: ";
      int total = 0;
      for (map < string, int >::iterator i = packetsSent.begin ();
           i != packetsSent.end (); i++)
        {
          trace () << i->first << ": " << i->second;
          total += i->second;
        }
      trace () << "Total: " << total << "\n";
    }
}

void
CMAC::finish ()
{
  string GraphName = string ("handoff.txt");
  graph (GraphName) << handoff;
  DebugInfoWriter::closeStream ();
  Graph::closeStream ();

  GraphName = string ("txRetries.txt");
  graph (GraphName) << totalRetries;
  DebugInfoWriter::closeStream ();
  Graph::closeStream ();

}

/* This function will reset the internal MAC state in the following way:
 * 1 -  Check if MAC is still in its active time, if timeout expired - go to sleep.
 * 2 -  Check if MAC is in the primary schedule wakeup (if so, MAC is able
 *      to start transmissions of either SYNC, RTS or DATA packets after a
```

```
 *      random contention offset.
 * 3 -  IF this is not primary wakeup, MAC can only listen, thus set
 *      state to MAC_STATE_ACTIVE_SILENT
 */
void
CMAC::resetDefaultState (const char *descr)
{
  //if (descr)
  //trace() << "Resetting MAC state to default, reason: " << descr;

  if (activationTimeout <= getClock ())
    {
      if (disableTAextension)
        {
          toRadioLayer (createRadioCommand (SET_STATE, SLEEP));
          setMacState (MAC_STATE_SLEEP, "active period expired (SMAC)");
        }
      else
        {
          performCarrierSense (MAC_CARRIER_SENSE_BEFORE_SLEEP);
        }
    }
  else if (primaryWakeup)
    {
      simtime_t randomContentionInterval =
        genk_dblrand (1) * contentionPeriod;
      if (needResync)
        {
          if (syncPacket)
            cancelAndDelete (syncPacket);
          syncPacket = new CMacPacket ("CMAC SYNC packet", MAC_LAYER_PACKET);
          syncPacket->setType (SYNC_CMAC_PACKET);
          syncPacket->setDestination (BROADCAST_MAC_ADDRESS);
          syncPacket->setSyncId (scheduleTable[0].ID);
          syncPacket->setSequenceNumber (scheduleTable[0].SN);
          syncPacket->setSync (currentFrameStart + frameTime - getClock () -
                               TX_TIME (syncPacketSize) -
                               randomContentionInterval);
          syncPacket->setByteLength (syncPacketSize);
          performCarrierSense (MAC_CARRIER_SENSE_FOR_TX_SYNC,
                               randomContentionInterval);
          return;
        }
```

```
      while (!TXBuffer.empty ())
        {
          if (txRetries <= 0)
            {
              trace () << "Transmission failed to " << txAddr;
              popTxBuffer ();
            }
          else
            {
              totalRetries++;
              if (useRtsCts && txAddr != BROADCAST_MAC_ADDRESS)
                {
                  performCarrierSense (MAC_CARRIER_SENSE_FOR_TX_RTS,
                                       randomContentionInterval);
                }
              else
                {
                  performCarrierSense (MAC_CARRIER_SENSE_FOR_TX_DATA,
                                       randomContentionInterval);
                }
              return;
            }
        }
      setMacState (MAC_STATE_ACTIVE, "nothing to transmit");
    }
  else
    {
      //primaryWakeup == false
      setMacState (MAC_STATE_ACTIVE_SILENT,
                   "node is awake not in primary schedule");
    }
}

/* This function will create a new primary schedule for this node.
 * Most of the task is delegated to updateScheduleTable function
 * This function will only schedule a self message to resycnronise the newly created
 * schedule
 */
void
CMAC::createPrimarySchedule ()
{
  updateScheduleTable (frameTime, self, 0);
  setTimer (SYNC_RENEW, resyncTime);
}
```

```
/* Helper function to change internal MAC state and print
 * a debug statement if neccesary */
void
CMAC::setMacState (int newState, const char *descr)
{
  if (macState == newState)
    return;
  if (printStateTransitions)
    {
      if (descr)
        trace () << "state changed from " << stateDescr[macState] <<
          " to " << stateDescr[newState] << ", reason: " << descr;
      else
        trace () << "state changed from " << stateDescr[macState] <<
          " to " << stateDescr[newState];
    }
  macState = newState;
}

/* This function will update schedule table with the given values
 * for wakeup time, schedule ID and schedule SN
 */
void
CMAC::updateScheduleTable (simtime_t wakeup, int ID, int SN)
{
  // First, search through existing schedules
  for (int i = 0; i < (int) scheduleTable.size (); i++)
    {
      //If schedule already exists
      if (scheduleTable[i].ID == ID)
        {
          //And SN is greater than ours, then update
          if (scheduleTable[i].SN < SN)
            {

              //Calculate new frame offset for this schedule
              simtime_t new_offset =
                getClock () - currentFrameStart + wakeup - frameTime;
              //trace() << "Resync successful for ID:" << ID << " old offset:" <<
              //              scheduleTable[i].offset << " new offset:" << new_offset;
              scheduleTable[i].offset = new_offset;
              scheduleTable[i].SN = SN;
```

```
            if (i == 0)
              {
                //If the update came for primary schedule, then the next frame
                //message has to be rescheduled
                setTimer (FRAME_START, wakeup);
                currentFrameStart += new_offset;
              }
            else
              {
                //This is not primary schedule, check that offset value falls within the
                //interval: 0 < offset < frameTime
                if (scheduleTable[i].offset < 0)
                  scheduleTable[i].offset += frameTime;
                if (scheduleTable[i].offset > frameTime)
                  scheduleTable[i].offset -= frameTime;
              }

          }
        else if (scheduleTable[i].SN > SN)
          {
            /* CMAC received a sync with lower SN than what currently stored in the
             * schedule table. With current CMAC implementation, this is not possible,
             * however in future it may be neccesary to implement a unicast sync packet
             * here to notify the source of this packet with the updated schedule
             */
          }
        //found and updated the schedule, nothing else need to be done
        return;
      }
  }

//At this stage, the schedule was not found in the current table, so it has to be added
CMacSchedule newSch;
newSch.ID = ID;
newSch.SN = SN;
trace () << "Creating schedule ID:" << ID << ", SN:" << SN << ", wakeup:" <<
  wakeup;

//Calculate the offset for the new schedule
if (currentFrameStart == -1)
  {
    //If currentFrameStart is -1 then this schedule will be the new primary schedule
    //and it's offset will always be 0
    newSch.offset = 0;
```

```
    }
  else
    {
      //This schedule is not primary, it is necessary to calculate the offset from primary
      //schedule for this new schedule
      newSch.offset = getClock () - currentFrameStart + wakeup - frameTime;
    }

  //Add new schedule to the table
  scheduleTable.push_back (newSch);

  //If the new schedule is primary, more things need to be done:
  if (currentFrameStart == -1)
    {
      //This is new primary schedule, and since SYNC packet was received at this time, it is
      //safe to assume that nodes of this schedule are active and listening right now,
      //so active period can be safely extended
      currentFrameStart = activationTimeout = getClock ();
      currentFrameStart += wakeup - frameTime;
      extendActivePeriod ();

      //create and schedule the next frame message
      setTimer (FRAME_START, wakeup);

      //this flag indicates that this schedule has to be rebroadcasted
      needResync = 1;
      primaryWakeup = true;

      //MAC is reset to default state, allowing it to initiate and accept transmissions
      resetDefaultState ("new primary schedule found");
    }
}

/* This function will handle a MAC frame received from the lower layer (physical or radio)
 * We try to see if the received packet is CMacPacket, otherwise we discard it
 * CMAC ignores values of RSSI and LQI
 */
void
CMAC::fromRadioLayer (cPacket * pkt, int channel, double frequency,
                      double noiseFloor, double RSSI, double LQI, double snr)
{
  CMacPacket *macPkt = dynamic_cast < CMacPacket * >(pkt);
  if (macPkt == NULL)
    return;
```

```
int source = macPkt->getSource ();
int destination = macPkt->getDestination ();
simtime_t nav = macPkt->getNav ();

if (destination != SELF_MAC_ADDRESS && destination != BROADCAST_MAC_ADDRESS)
  {
    if (macState == MAC_CARRIER_SENSE_FOR_TX_RTS && useFRTS)
      {
        //FRTS would need to be implemented here, for now just keep silent
      }
    if (collisionResolution != 2 && nav > 0)
      setTimer (TRANSMISSION_TIMEOUT, nav);
    extendActivePeriod (nav);
    setMacState (MAC_STATE_ACTIVE_SILENT, "overheard a packet");
    return;
  }

switch (macPkt->getType ())
  {
    //recebido um pacote de requisicao de vizinhos
  case FIND_NEIGHBORS_CMAC_PACKET:
    {
      //cria um pacote da camada mac
      CMacPacket *macPkt =
        new CMacPacket ("CMAC data packet", MAC_LAYER_PACKET);
      macPkt->setType (NEIGHBOR_FOUND_CMAC_PACKET);
      macPkt->setSource (SELF_MAC_ADDRESS);
      macPkt->setDestination (BROADCAST_MAC_ADDRESS);
      macPkt->setSequenceNumber (txSequenceNum);
      macPkt->setNav (TX_TIME (ctsPacketSize));
      macPkt->setByteLength (ctsPacketSize);
      if (bufferPacket (macPkt))
        {                     // this is causing problems
          if (TXBuffer.size () == 1)
            checkTxBuffer ();
        }
      else
        {
          //cancelAndDelete(macPkt);
          //We could send a control message to upper layer to inform of full buffer
        }
      //setTimer(DONE_MESSAGE,0.5);
      break;
```

```
  }
case NEIGHBOR_FOUND_CMAC_PACKET:
  {
    trace () << "macPkt" << macPkt;
    neighbors.add (macPkt->getSource (), this->self);
    /*trace()<<"Lista de vizinhos tamanho"<< neighbors.getSize();
       neighbor *n=neighbors.getHead()->getNext();
       for (int i=0;i<neighbors.getSize();i++){
       trace()<<n->getSource()<<"->"<<n->getAdress();
       n=n->getNext();
       } */
    //trace() << "Neihgbor found message returned";
    break;
  }
  /* A lista de vizinhos a 2 hops deve ser obtida para
   * idetificar os nos interferentes na rede                      */
case NEIGHBOR_LIST_CMAC_PACKET:
  {

    //verifica se o nó é vizinho
    if (neighbors.isMyNeghbor (macPkt->getSource ()))
      {
        neighbor *n = neighbors.getNeighbor (macPkt->getSource ());
        n->setConnectivityDegree (macPkt->getNeighborListArraySize ());
        //acha os vizinhos dos vizinhos que nao estao na minha lista de
        // vizinhos que nao sao o proprio no
        for (unsigned int i = 0; i < macPkt->getNeighborListArraySize ();
             i++)
          {
            if (!neighbors.isMyNeghbor (macPkt->getNeighborList (i))
                && macPkt->getNeighborList (i) != self)
              {
                //adiciona a lista de vizinhos a 2 hops
                twoHopNeighbors.add (macPkt->getNeighborList (i),
                                     macPkt->getSource ());
              }
          }
      }
    break;
  }
case UPDATE_CHANNEL_1_HOP_PACKET:
  {
    trace () << "received message UPDATE_CHANNEL_1_HOP_PACKET " <<
      macPkt->getNotifiedChannel ();
```

```
    if (neighbors.isMyNeghbor (macPkt->getSource ()))
      neighbors.getNeighbor (macPkt->getSource ())->setChannel (
      macPkt-> getNotifiedChannel());
    for (unsigned int i = 0; i < macPkt->getNeighborListArraySize (); i++)
      {
        if (macPkt->getNeighborList (i) == self)
          {
            CMacPacket *macPkt2 =
              new CMacPacket ("cognitiveMAC data packet",
                              MAC_LAYER_PACKET);
            macPkt2->setType (UPDATE_CHANNEL_2_HOP_PACKET);
            macPkt2->setSource (SELF_MAC_ADDRESS);
            macPkt2->setDestination (BROADCAST_MAC_ADDRESS);
            macPkt2->setNotifyingNode (macPkt->getSource ());
            macPkt2->setChannelNowFree (macPkt->getChannelNowFree ());
            macPkt2->setNotifiedChannel (macPkt->getNotifiedChannel ());
            macPkt2->setSequenceNumber (txSequenceNum);
            macPkt2->setNav (TX_TIME (ctsPacketSize));
            macPkt2->setByteLength (ctsPacketSize);

            if (bufferPacket (macPkt2))
              {             // this is causing problems
                if (TXBuffer.size () == 1)
                  checkTxBuffer ();
              }
            else
              {

                // cancelAndDelete(macPkt);
                //We could send a control message to upper layer to inform of full buffer
              }
          }
      }
    break;
  }
case UPDATE_CHANNEL_2_HOP_PACKET:
  {
    trace () << "received message UPDATE_CHANNEL_2_HOP_PACKET channel" <<
      macPkt->getNotifiedChannel ();
    list < Channel_type >::iterator c;
    for (c = ChannelList->begin (); c != ChannelList->end (); c++)
      {
        if (c->channel == macPkt->getChannelNowFree ())
          {
```

```
            c->state = STATE_CLEAR;
          }
        if (c->channel == macPkt->getNotifiedChannel ())
          {
            c->state = STATE_BUSY;
            trace () << "Channel " << c->channel << " state " << c->state;
          }
      }
    //}
    break;
  }
  /* received a RTS frame */
case RTS_CMAC_PACKET:
  {
    //If this node is the destination, reply with a CTS, otherwise
    //set a timeout and keep silent for the duration of communication
    ChannelToChange = macPkt->getRTSChannel ();
    if (ctsPacket)
      cancelAndDelete (ctsPacket);
    ctsPacket = new CMacPacket ("CMAC CTS packet", MAC_LAYER_PACKET);
    ctsPacket->setType (CTS_CMAC_PACKET);
    ctsPacket->setSource (SELF_MAC_ADDRESS);
    ctsPacket->setDestination (source);
    ctsPacket->setNav (nav - TX_TIME (ctsPacketSize));
    ctsPacket->setByteLength (ctsPacketSize);

    // Send CTS packet to radio
    toRadioLayer (ctsPacket);
    toRadioLayer (createRadioCommand (SET_STATE, TX));

    packetsSent["CTS"]++;
    collectOutput ("Sent packets breakdown", "CTS");
    ctsPacket = NULL;

    // update MAC state
    setMacState (MAC_STATE_WAIT_FOR_DATA, "sent CTS packet");

    // create a timeout for expecting a DATA packet reply
    setTimer (TRANSMISSION_TIMEOUT,
              TX_TIME (ctsPacketSize) + waitTimeout);
    trace () << "Recebeu RTS agendou troca de canal";
    sendDelayed (createRadioCommand
                 (SET_CHANNEL, macPkt->getRTSChannel ()),
                 TX_TIME (ctsPacketSize), "toRadioModule");
```

```
    //setTimer(CHANGE_CHANNEL, TX_TIME(ctsPacketSize));
    setTimer (RESTORE_CHANNEL, 1);
    break;
  }

  /* received a CTS frame */
case CTS_CMAC_PACKET:
  {
    if (macState == MAC_STATE_WAIT_FOR_CTS)
      {

        toRadioLayer (createRadioCommand
                      (SET_CHANNEL, (int) (best->channel)));
        trace () << "Recebeu CTS e trocou canal para " << (int) (best->
                                                                 channel);
        //handoff++;
        if (TXBuffer.empty ())
          {
            trace () <<
              "WARNING: invalid MAC_STATE_WAIT_FOR_CTS while buffer is empty";
            resetDefaultState ("invalid state while buffer is empty");
          }
        else if (source == txAddr)
          {
            cancelTimer (TRANSMISSION_TIMEOUT);
            sendDataPacket ();
          }
        else
          {
            trace () << "WARNING: recieved unexpected CTS from " <<
              source;
            resetDefaultState ("unexpected CTS");
          }
      }
    setTimer (RESTORE_CHANNEL, 1);
    break;
  }

  /* received DATA frame */
case DATA_CMAC_PACKET:
  {
    // Forward the frame to upper layer first
    toNetworkLayer (decapsulatePacket (macPkt));
    // If the frame was sent to broadcast address, nothing else needs to be done
```

```
if (destination == BROADCAST_MAC_ADDRESS)
  break;
//macPkt->
// If MAC was expecting this frame, clear the timeout
if (macState == MAC_STATE_WAIT_FOR_DATA)
  cancelTimer (TRANSMISSION_TIMEOUT);
// Create and send an ACK frame (since this node is the destination for DATA frame)
if (ackPacket)
  cancelAndDelete (ackPacket);
if (useAHP)
  {
    if (LQI > 0)
      {
        packet_type pk;
        pk.metric[rssi] = RSSI;
        pk.metric[noisefloor] = noiseFloor;
        pk.metric[snir] = LQI;
        pk.metric[delay] =
          simTime ().dbl () - macPkt->getTimestamp ().dbl ();
        PacketList.push_front (pk);
      }
  }

list < Channel_type >::iterator c;
radioModule->getChannel ()->RSSI = RSSI;
radioModule->getChannel ()->noiseFloor = noiseFloor;
radioModule->getChannel ()->SINR = LQI;
radioModule->getChannel ()->delay =
  simTime ().dbl () - macPkt->getTimestamp ().dbl ();

ackPacket = new CMacPacket ("CMAC ACK packet", MAC_LAYER_PACKET);
ackPacket->setSource (SELF_MAC_ADDRESS);
ackPacket->setDestination (source);
ackPacket->setType (ACK_CMAC_PACKET);
ackPacket->setByteLength (ackPacketSize);
ackPacket->setRSSI (RSSI);
ackPacket->setNoiseFloor (noiseFloor);
ackPacket->setSNIR (LQI);
ackPacket->setDelay (simTime () - macPkt->getTimestamp ());

// Send ACK packet to the radio
toRadioLayer (ackPacket);
toRadioLayer (createRadioCommand (SET_STATE, TX));
```

```
    packetsSent["ACK"]++;
    collectOutput ("Sent packets breakdown", "ACK");
    ackPacket = NULL;

    // update MAC state
    setMacState (MAC_STATE_IN_TX, "transmitting ACK packet");

    // create a timeout for this transmission - nothing is expected in reply
    // so MAC is only waiting for the RADIO to finish the packet transmission
    setTimer (TRANSMISSION_TIMEOUT, TX_TIME (ackPacketSize));
    extendActivePeriod (TX_TIME (ackPacketSize));
    ChannelToChange = best->channel;
    //setTimer(CHANGE_CHANNEL, TX_TIME(ackPacketSize));
    sendDelayed (createRadioCommand (SET_CHANNEL, best->channel),
                 TX_TIME (ackPacketSize), "toRadioModule");
    break;
  }

  /* received ACK frame */
case ACK_CMAC_PACKET:
  {
    best->RSSI = macPkt->getRSSI ();
    best->SINR = macPkt->getSNIR ();
    best->noiseFloor = macPkt->getNoiseFloor ();
    best->delay = macPkt->getDelay ();
    if (macPkt->getSNIR () > 0)
      {
        packet_type pk;
        pk.metric[rssi] = macPkt->getRSSI ();
        pk.metric[noisefloor] = macPkt->getNoiseFloor ();
        pk.metric[snir] = macPkt->getSNIR ();
        pk.metric[delay] = macPkt->getDelay ().dbl ();
        PacketList.push_front (pk);
      }
    if (macState == MAC_STATE_WAIT_FOR_ACK && source == txAddr)
      {
        trace () << "Transmission succesful to " << txAddr;
        cancelTimer (TRANSMISSION_TIMEOUT);
        popTxBuffer ();
        resetDefaultState ("transmission successful (ACK received)");
      }
    ChannelToChange = best->channel;
    toRadioLayer (createRadioCommand (SET_CHANNEL, best->channel));
    break;
```

```
      }

      /* received SYNC frame */
    case SYNC_CMAC_PACKET:
      {
        // Schedule table is updated with values from the SYNC frame
        updateScheduleTable (macPkt->getSync (),
                             macPkt->getSyncId (),
                             macPkt->getSequenceNumber ());

        break;
      }

    default:
      {
        trace () << "Packet with unknown type (" << macPkt->getType () <<
          ") received: [" << macPkt->getName () << "]";
      }
    }
}

void
CMAC::carrierIsBusy ()
{
  /* Since we are hearing some communication on the radio we need to do two things:
   * 1 - extend our active period
   * 2 - set MAC state to MAC_STATE_ACTIVE_SILENT unless we are actually expecting
   *  to receive something (or sleeping)
   */
  if (macState == MAC_STATE_SETUP || macState == MAC_STATE_SLEEP)
    return;
  if (!disableTAextension)
    extendActivePeriod ();
  if (collisionResolution == 0)
    {
      if (macState == MAC_CARRIER_SENSE_FOR_TX_RTS
          || macState == MAC_CARRIER_SENSE_FOR_TX_DATA
          || macState == MAC_CARRIER_SENSE_FOR_TX_SYNC
          || macState == MAC_CARRIER_SENSE_BEFORE_SLEEP)
        {
          resetDefaultState ("sensed carrier");
        }
    }
  else
```

```
    {
      if (macState != MAC_STATE_WAIT_FOR_ACK
          && macState != MAC_STATE_WAIT_FOR_DATA
          && macState != MAC_STATE_WAIT_FOR_CTS)
        setMacState (MAC_STATE_ACTIVE_SILENT, "sensed carrier");
    }
}

/* This function handles carrier clear message, received from the radio module.
 * That is sent in a response to previous request to perform a carrier sense
 */
void
CMAC::carrierIsClear ()
{
  switch (macState)
    {

      /* MAC requested carrier sense to transmit an RTS packet */
    case MAC_CARRIER_SENSE_FOR_TX_RTS:
      {
        if (TXBuffer.empty ())
          {
            trace () <<
              "WARNING! BUFFER_IS_EMPTY in MAC_CARRIER_SENSE_FOR_TX_RTS, will reset state";
            resetDefaultState ("empty transmission buffer");
            break;
          }
        // create and send RTS frame
        if (rtsPacket)
          cancelAndDelete (rtsPacket);
        if (neighbors.isMyNeghbor (txAddr))
          {
            trace () << "Channel changed to send RTS packet " << (int) (best->
                                                                         channel)
              << "->" << (int) (neighbors.getNeighbor (txAddr)->
                                getChannel ()) << "to node " << txAddr;
            toRadioLayer (createRadioCommand
                          (SET_CHANNEL,
                           (int) (neighbors.getNeighbor (txAddr)->
                                  getChannel ())));
          }
        else
          {
            trace () << "Não era vizinho pra mandar RTS";
```

```
      }

    //handoff++;
    rtsPacket = new CMacPacket ("RTS message", MAC_LAYER_PACKET);
    rtsPacket->setRTSChannel (best->channel);
    rtsPacket->setSource (SELF_MAC_ADDRESS);
    rtsPacket->setDestination (txAddr);
    rtsPacket->setNav (TX_TIME (rtsPacketSize) + TX_TIME (ctsPacketSize) +
                       TX_TIME (ackPacketSize) +
                       TX_TIME (TXBuffer.front ()->getByteLength ()));
    rtsPacket->setType (RTS_CMAC_PACKET);
    rtsPacket->setByteLength (rtsPacketSize);
    toRadioLayer (rtsPacket);
    toRadioLayer (createRadioCommand (SET_STATE, TX));

    if (useRtsCts)
      txRetries--;
    packetsSent["RTS"]++;
    collectOutput ("Sent packets breakdown", "RTS");
    rtsPacket = NULL;

    // update MAC state
    setMacState (MAC_STATE_WAIT_FOR_CTS, "sent RTS packet");

    // create a timeout for expecting a CTS reply
    setTimer (TRANSMISSION_TIMEOUT,
              TX_TIME (rtsPacketSize) + waitTimeout);
    break;
  }

  /* MAC requested carrier sense to transmit a SYNC packet */
case MAC_CARRIER_SENSE_FOR_TX_SYNC:
  {
    // SYNC packet was created in scheduleSyncPacket function
    if (syncPacket != NULL)
      {
        // Send SYNC packet to radio
        toRadioLayer (syncPacket);
        toRadioLayer (createRadioCommand (SET_STATE, TX));

        packetsSent["SYNC"]++;
        collectOutput ("Sent packets breakdown", "SYNC");
        syncPacket = NULL;
```

```
            // Clear the resync flag
            needResync = 0;

            // update MAC state
            setMacState (MAC_STATE_IN_TX, "transmitting SYNC packet");

            // create a timeout for this transmission - nothing is expected in reply
            // so MAC is only waiting for the RADIO to finish the packet transmission
            setTimer (TRANSMISSION_TIMEOUT, TX_TIME (syncPacketSize));

          }
        else
          {
            trace () <<
              "WARNING: Invalid MAC_CARRIER_SENSE_FOR_TX_SYNC while syncPacket undefined";
            resetDefaultState ("invalid state, no SYNC packet found");
          }
        break;
      }

      /* MAC requested carrier sense to transmit DATA packet */
    case MAC_CARRIER_SENSE_FOR_TX_DATA:
      {
        sendDataPacket ();
        break;
      }

      /* MAC requested carrier sense before going to sleep */
    case MAC_CARRIER_SENSE_BEFORE_SLEEP:
      {
        // primaryWakeup flag is cleared (in case the node will wake up in the current frame
        // for a secondary schedule)
        primaryWakeup = false;

        // update MAC and RADIO states
        toRadioLayer (createRadioCommand (SET_STATE, SLEEP));
        setMacState (MAC_STATE_SLEEP, "no activity on the channel");
        break;
      }
    }
}

void
CMAC::sendDataPacket ()
```

```
{
  if (TXBuffer.empty ())
    {
      trace () <<
        "WARNING: Invalid MAC_CARRIER_SENSE_FOR_TX_DATA while TX buffer is empty";
      resetDefaultState ("empty transmission buffer");
      return;
    }
  // create a copy of first message in the TX buffer and send it to the radio
  toRadioLayer (TXBuffer.front ()->dup ());
  packetsSent["DATA"]++;
  collectOutput ("Sent packets breakdown", "DATA");

  //update MAC state based on transmission time and destination address
  double txTime = TX_TIME (TXBuffer.front ()->getByteLength ());

  if (txAddr == BROADCAST_MAC_ADDRESS)
    {
      // This packet is broadcast, so no reply will be received
      // The packet can be cleared from transmission buffer
      // and MAC timeout is only to allow RADIO to finish the transmission
      popTxBuffer ();
      setMacState (MAC_STATE_IN_TX, "sent DATA packet to BROADCAST address");
      setTimer (TRANSMISSION_TIMEOUT, txTime);
    }
  else
    {
      // This packet is unicast, so MAC will be expecting an ACK
      // packet in reply, so the timeout is longer
      // If we are not using RTS/CTS exchange, then this attempt
      // also decreases the txRetries counter
      // (NOTE: with RTS/CTS exchange sending RTS packet decrements counter)
      if (!useRtsCts)
        txRetries--;
      setMacState (MAC_STATE_WAIT_FOR_ACK,
                   "sent DATA packet to UNICAST address");
      setTimer (TRANSMISSION_TIMEOUT, txTime + waitTimeout);
    }
  extendActivePeriod (txTime);

  //update RADIO state
  toRadioLayer (createRadioCommand (SET_STATE, TX));
}
```

```
/* This function will set a timer to perform carrier sense.
 * MAC state is important when performing CS, so setMacState is always called here.
 * delay allows to perform a carrier sense after a choosen delay (useful for
 * randomisation of transmissions)
 */
void
CMAC::performCarrierSense (int newState, simtime_t delay)
{
  setMacState (newState);
  setTimer (CARRIER_SENSE, delay);
}

/* This function will extend active period for MAC, ensuring that the remaining active
 * time it is not less than listenTimeout value. Also a check TA message is scheduled here
 * to allow the node to go to sleep if activation timeout expires
 */
void
CMAC::extendActivePeriod (simtime_t extra)
{
  simtime_t curTime = getClock ();
  if (conservativeTA)
    {
      curTime += extra;
      while (activationTimeout < curTime)
        {
          activationTimeout += listenTimeout;
        }
      if (curTime + listenTimeout < activationTimeout)
        return;
      activationTimeout += listenTimeout;
    }
  else if (activationTimeout < curTime + listenTimeout + extra)
    {
      activationTimeout = curTime + listenTimeout + extra;
    }
  setTimer (CHECK_TA, activationTimeout - curTime);
}

/* This function will check the transmission buffer, and if it is not empty, it will update
 * current communication parameters: txAddr and txRetries
 */
void
CMAC::checkTxBuffer ()
{
```

```
  if (TXBuffer.empty ())
    return;
  CMacPacket *macPkt = check_and_cast < CMacPacket * >(TXBuffer.front ());
  txAddr = macPkt->getDestination ();
  txRetries = maxTxRetries;
  txSequenceNum = 0;
}

/* This function will remove the first packet from MAC transmission buffer
 * checkTxBuffer is called in case there are still packets left in the buffer to transmit
 */
void
CMAC::popTxBuffer ()
{
  cancelAndDelete (TXBuffer.front ());
  TXBuffer.pop ();
  checkTxBuffer ();
}

neighbor *
neighbor::getNext ()
{
  return next;
}

neighbor *
neighborList::getNeighbor (int adress)
{
  neighbor *n = head.getNext ();
  for (int i = 0; i < this->size; i++)
    {
      if (n->getAdress () == adress)
        {
          return n;
        }
      n = n->getNext ();
    }
  return NULL;
}

void
neighborList::add (neighbor n)
{
  if (!isMyNeghbor (n.getAdress ()))
```

```
    {
      if (this->size == 0)
        {
          head.setNext (&n);
          size = 1;
        }
      else
        {
          n.setNext (head.getNext ());
          head.setNext (&n);
          size++;
        }
    }
}

void
neighborList::add (int adress, int source)
{
  neighbor *n2 = head.getNext ();
  bool isNeighborSameSource = false;
  for (int i = 0; i < this->size; i++)
    {
      if (n2->getAdress () == adress && n2->getSource () == source)
        {
          isNeighborSameSource = true;
        }
      n2 = n2->getNext ();
    }
  if (!isNeighborSameSource)
    {
      neighbor *n = new neighbor (adress, source);
      if (this->size == 0)
        {
          head.setNext (n);
          size = 1;
        }
      else
        {
          n->setNext (head.getNext ());
          head.setNext (n);
          size++;
        }
    }
}
```

```
bool
neighborList::isMyNeghbor (int adress)
{
  neighbor *n = head.getNext ();
  for (int i = 0; i < this->size; i++)
    {
      if (n->getAdress () == adress)
        {
          return true;
        }
      n = n->getNext ();
    }
  return false;
}

bool
neighborList::allVisited ()
{
  neighbor *n = head.getNext ();
  for (int i = 0; i < this->size; i++)
    {
      if (n->isVisited () == false)
        {
          return false;
        }
      n = n->getNext ();
    }
  return true;
}

neighbor *
neighborList::higherDegreeNotVisited ()
{
  neighbor *n;
  if (this->isEmpty ())
    {
      return NULL;
    };
  cout << "Erro 1" << endl;
  neighbor *higher = head.getNext ();
  cout << "Erro 2" << endl;
  while (higher->isVisited () && higher->getNext () != NULL)
    {
```

```
      cout << "Erro 3" << endl;
      higher = higher->getNext ();
      cout << "Erro 4" << endl;
    }
  if (higher->isVisited () && higher->getNext () != NULL)
    {
      cout << "Erro 5" << endl;
      return NULL;
      cout << "Erro 6" << endl;
    }
  if (higher->getNext () != NULL)
    n = higher->getNext ();
  cout << "Erro 7" << endl;
  do
    {
      if (n->getConnectivityDegree () > higher->getConnectivityDegree ())
        {
          if (!n->isVisited ())
            {
              higher = n;
            }
        }
      if (n->getNext () != NULL)
        n = n->getNext ();
      cout << "Erro 8" << endl;
    }
  while (n->getNext () != NULL);
  //higher->setVisited(true);
  return higher;
}

void
neighborList::clearVisited ()
{
  neighbor *n = head.getNext ();
  for (int i = 0; i < this->size; i++)
    {
      n->setVisited (false);
      n = n->getNext ();
    }
}

bool
neighborList::isEmpty ()
```

```
{
  if (this->size == 0)
    {
      return true;
    }
  else
    {
      return false;
    }
}


neighborList *
neighborList::intersection (neighborList n)
{
  neighbor *n1;
  neighbor *n2;
  neighborList *result = new neighborList ();
  n1 = n.head.getNext ();
  for (int i = 0; i < n.getSize (); i++)
    {
      n2 = this->head.getNext ();
      for (int j = 0; j < this->getSize (); j++)
        {
          if (n1->getAdress () == n2->getAdress ())
            {
              result->add (n1->getAdress ());
            }
          n2 = n2->getNext ();
        }
      n1 = n1->getNext ();
    }
  return result;
}

neighborList *
neighborList::neighborOfSource (int source)
{
  neighbor *n = head.getNext ();
  neighborList *result = new neighborList ();
  for (int i = 0; i < this->size; i++)
    {
      if (source == n->getSource ())
        {
```

```
          result->add (n->getAdress (), n->getSource ());
        }
      n = n->getNext ();
    }
  return result;
}

neighbor::neighbor (int adress, int source)
{
  this->adress = adress;
  this->channel = 1;
  this->connectivityDegree = 0;
  this->source = source;
  this->visited = false;
  next = NULL;
}

neighbor
neighbor::copyInstance ()
{
  neighbor n;
  n.adress = this->adress;
  n.channel = this->channel;
  n.connectivityDegree = this->connectivityDegree;
  n.source = this->source;
  n.visited = this->visited;
  return n;
}

packet_type
CMAC::ahp (list < packet_type > PacketList, int numberOfParams,
           double *somaColunas)
{
  double constant = -1 * (1 / log ((double) PacketList.size ()));
  cout << "Constante:" << constant << endl;
  list < packet_type >::iterator pkt;

  //calcula as somas das colunas
  double somaColuna[numberOfParams];
  for (int i = 0; i < numberOfParams; i++)
    {
      somaColuna[i] = 0;
    }
  pkt = PacketList.begin ();
```

```
for (unsigned int i = 0; i < PacketList.size (); i++)
  {
    somaColuna[rssi] += pkt->metric[rssi];
    somaColuna[delay] += pkt->metric[delay];
    somaColuna[snir] += pkt->metric[snir];
    somaColuna[noisefloor] += pkt->metric[noisefloor];
    pkt++;
  }
for (int i = 0; i < numberOfParams; i++)
  {
    somaColunas[i] = somaColuna[i];
  }

pkt = PacketList.begin ();
for (unsigned int i = 0; i < PacketList.size (); i++)
  {
    pkt->metric[noisefloor] =
      (pkt->metric[noisefloor] / somaColuna[noisefloor]) * 100;
    pkt->metric[snir] = (pkt->metric[snir] / somaColuna[snir]) * 100;
    pkt->metric[rssi] =
      100 - ((pkt->metric[rssi] / somaColuna[rssi]) * 100);
    pkt->metric[delay] =
      100 - ((pkt->metric[delay] / somaColuna[delay]) * 100);
    pkt++;
  }
for (int i = 0; i < numberOfParams; i++)
  {
    somaColuna[i] = 0;
  }
pkt = PacketList.begin ();
for (unsigned int i = 0; i < PacketList.size (); i++)
  {
    somaColuna[noisefloor] += pkt->metric[noisefloor];
    somaColuna[snir] += pkt->metric[snir];
    somaColuna[rssi] += pkt->metric[rssi];
    somaColuna[delay] += pkt->metric[delay];
    pkt++;
  }

//obtem os pesos em funcao da entorpia
double HL[PacketList.size ()];
double entropia[numberOfParams];
for (int i = 0; i < numberOfParams; i++)
  {
```

```
      pkt = PacketList.begin ();
      for (unsigned int j = 0; j < PacketList.size (); j++)
        {
          double bjL = pkt->metric[i];
          double fjL = bjL / somaColuna[i];
          HL[j] = (fjL * log (fjL));
          pkt++;
        }
      double somaHL = 0;
      for (int k = 0; k < numberOfParams; k++)
        {
          somaHL += HL[k];
        }
      entropia[i] = somaHL * constant;
    }
  double somaEntropias = 0;
  for (int k = 0; k < numberOfParams; k++)
    {
      somaEntropias += entropia[k];
    }
  packet_type weights;
  for (int i = 0; i < numberOfParams; i++)
    {
      weights.metric[i] =
        (1 - entropia[i]) / (numberOfParams - somaEntropias);
    }
  return weights;
}
```

Anexo B

Módulo da Classe Geradora de Tráfego

```
/*************************************************************************
 *  Copyright: National ICT Australia,  2007 - 2010                      *
 *  Developed at the ATP lab, Networked Systems research theme           *
 *  Author(s): Athanassios Boulis, Yuriy Tselishchev                     *
 *  This file is distributed under the terms in the attached LICENSE file.  *
 *  If you do not find this file, copies can be found by writing to:     *
 *                                                                       *
 *      NICTA, Locked Bag 9013, Alexandria, NSW 1435, Australia          *
 *      Attention:  License Inquiry.                                     *
 *                                                                       *
 *************************************************************************/

#include "TrafficGenerator.h"

Define_Module (TrafficGenerator);

void
TrafficGenerator::initialize (int stage)
{
  trafficType = par ("trafficType");
  randomMaxInterval = par ("randomMaxInterval");
  exponentialAlpha = par ("exponentialAlpha");
  exponentialBeta = par ("exponentialBeta");
  flows = par ("flows");
  transmissionInterval = par ("transmissionInterval");
  cMessage *sendPacket = new cMessage ("noise", SEND_PACKET_TIMER);
  scheduleAt (200, sendPacket);
```

```
  numNodes = getParentModule ()->par ("numNodes");
  srand (time (0));
}

void
TrafficGenerator::handleMessage (cMessage * msg)
{
  switch (msg->getKind ())
    {

    case SEND_PACKET_TIMER:
      {
        if (numNodes > 1)
          {
            int sendersAndReceivers[numNodes];
            for (int i = 0; i < numNodes; i++)
              {
                sendersAndReceivers[i] = i;
              }
            for (int i = 0; i < numNodes * 200; i++)
              {
                int randomNum1 = rand () % numNodes;    //(int)genk_intrand(0, numNodes);
                int randomNum2 = rand () % numNodes;    //(int)genk_intrand(0, numNodes);
                int aux = sendersAndReceivers[randomNum1];
                sendersAndReceivers[randomNum1] =
                  sendersAndReceivers[randomNum2];
                sendersAndReceivers[randomNum2] = aux;
              }
            for (int i = 0; i < flows; i++)
              {
                TrafficGeneratorMessage *sendPkt =
                  new TrafficGeneratorMessage ("x",
                                               TRAFFIC_GENERATOR_MESSAGE);
                sendPkt->setDestination (sendersAndReceivers[i * 2]);
                sendPkt->
                  setTrafficGeneratorMessageKind
                  (TRAFFIC_GENERATOR_SEND_PACKET);
                send (sendPkt, "toNode", sendersAndReceivers[(i * 2) + 1]);
                trace () << "TrafficGen funciona";
              }
          }
        if (trafficType == 1)
          {
            cMessage *sendPacket = new cMessage ("Timer", SEND_PACKET_TIMER);
```

```
            scheduleAt (simTime () + transmissionInterval, sendPacket);
            trace () << "Periodical traffic";
          }
        else if (trafficType == 2)
          {
            cMessage *sendPacket = new cMessage ("Timer", SEND_PACKET_TIMER);
            scheduleAt (simTime () + (rand () % randomMaxInterval),
                        sendPacket);
            trace () << "Random traffic";
          }
        else if (trafficType == 3)
          {
            cMessage *sendPacket = new cMessage ("Timer", SEND_PACKET_TIMER);
            scheduleAt (simTime () +
                        statisticalRandom.exponential (exponentialAlpha,
                                                       exponentialBeta),
                        sendPacket);
            trace () << "Exponential traffic";
          }

        break;
      }
    default:
      {
        opp_error ("ERROR: TrafficGen received unknown message kind=%i",
                   msg->getKind ());
        break;
      }
    }
  delete msg;
}

void
TrafficGenerator::finishSpecific ()
{
  DebugInfoWriter::closeStream ();
  Graph::closeStream ();
}
```

Referências Bibliográficas

Akan, O.; Karli, O. & Ergul, O. (2009). Cognitive radio sensor networks. *Network, IEEE*, 23(4):34 –40.

Akyildiz, I.; Lee, W.-Y.; Vuran, M. & Mohanty, S. (2008). A survey on spectrum management in cognitive radio networks. *Communications Magazine, IEEE*, 46(4):40–48.

Akyildiz, I. F.; Lee, W.-Y. & Chowdhury, K. R. (2009). Crahns: Cognitive radio ad hoc networks. *Ad Hoc Netw.*, 7(5):810--836.

Akyildiz, I. F.; Lee, W.-Y.; Vuran, M. C. & Mohanty, S. (2006). Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. *Comput. Netw.*, 50(13):2127--2159.

Anatel (2008). Regulamento sobre equipamentos radiocomunicação de radiação restrita. URL: http://www.anatel.gov.br/Portal/verificaDocumentos/ Acessada em 11/09/2009.

Andrews, G. R. (1991). Paradigms for process interaction in distributed programs. *ACM Comput. Surv.*, 23:49--90.

Boulis, A. (2010). Castalia: A simulator for wireless sensor networks and body area networks. user's manual. URL: http://castalia.npc.nicta.com.au/pdfs/ Acessada em 21/09/2010.

Byun, S.-S.; Balasingham, I. & Liang, X. (2008). *Dynamic Spectrum Allocation in Wireless Cognitive Sensor Networks: Improving Fairness and Energy Efficiency*. IEEE.

Cavalcanti, D.; Das, S.; Wang, J. & Challapali, K. (2008). Cognitive radio based wireless sensor networks. Em *Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on*, pp. 1 –6.

Cavalcanti, D.; Schmitt, R. & Soomro, A. (2007). Achieving energy efficiency and qos for low-rate applications with 802.11e. Em *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pp. 2143 –2148.

Cordeiro, C.; Challapali, K.; Birru, D. & Sai Shankar, N. (2005). Ieee 802.22: the first worldwide wireless standard based on cognitive radios. Em *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 328–337.

Cormio, C. & Chowdhury, K. R. (2009). A survey on mac protocols for cognitive radio networks. *Ad Hoc Netw.*, 7(7):1315--1329.

Correia, L. & Nogueira, J. (2008). Transmission power control techniques for mac protocols in wireless sensor networks. Em *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pp. 1049 –1054.

Correia, L. H.; Macedo, D. F.; dos Santos, A. L.; Loureiro, A. A. & Nogueira, J. M. S. (2007). Transmission power control techniques for wireless sensor networks. *Computer Networks*, 51(17):4765 – 4779.

FCC (2001). Et docket no. 00-47et docket no. 03-322.authorization and use of software defined radios. URL: http://hraunfoss.fcc.gov/edocs_public/attachmatch/FCC-01-264A1.pdf. Acessada em 25/01/2010.

FCC (2003). Et docket no. 03-322. notice of proposed rule making and order.

FCC (2005). Et docket no. 03-108. facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies (tech. rep.).

Gao, S.; Qian, L.; Vaman, D. & Qu, Q. (2007). Energy efficient adaptive modulation in wireless cognitive radio sensor networks. Em *Communications, 2007. ICC '07. IEEE International Conference on*, pp. 3980 –3986.

Ge, Y.; Sun, Y. & Lu, S. (2009). ADSD: An Automatic Distributed Spectrum Decision method in Cognitive Radio networks. *Networks, 2009. ICFIN 2009*, pp. 0--5.

Hamdaoui, B. & Shin, K. (2008). Os-mac: An efficient mac protocol for spectrum-agile wireless networks. *Mobile Computing, IEEE Transactions on*, 7(8):915–930.

Haykin, S. (2005). Cognitive radio: brain-empowered wireless communications. *Selected Areas in Communications, IEEE Journal on*, 23(2):201–220.

Howitt, I. & Gutierrez, J. (2003). Ieee 802.15.4 low rate - wireless personal area network coexistence issues. Em *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pp. 1481 –1486 vol.3.

IEEE (2003a). Ieee 802.11h. URL: http://www.dcc.unicamp.br/ wainer/papers/metod07.pdf. Acessada em 09/02/2010.

IEEE, S. (2003b). Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans). *IEEE Std 802.15.4-2003*, pp. 1--670.

IEEE, S. (2011). Ieee draft standard for information technology -telecommunications and information exchange between systems - wireless regional area networks (wran) - specific requirements - part 22: Cognitive wireless ran medium access control (mac) and physical layer (phy) specifications: Policies and procedures for operation in the tv bands. *IEEE P802.22/D2.0, February 2011*, pp. 1–698.

Jondral, F. K. (2005). Software-defined radio: basics and evolution to cognitive radio. *EURASIP J. Wirel. Commun. Netw.*, 2005(3):275--283.

Kanodia, V.; Sabharwal, A. & Knightly, E. (2004). Moar: A multi-channel opportunistic auto-rate media access protocol for ad hoc networks. Em *BROADNETS '04: Proceedings of the First International Conference on Broadband Networks*, pp. 600--610, Washington, DC, USA. IEEE Computer Society.

Kusy, B.; Richter, C.; Hu, W.; Afanasyev, M.; Jurdak, R.; Brunig, M.; Abbott, D.; Huynh, C. & Ostry, D. (2011). Radio diversity for reliable communication in WSNs. Em *Proceedings of the 10th International Conference on Information Processing in Sensor Networks*, IPSN '11, pp. 270--281.

Lien, S.-Y.; Tseng, C.-C. & Chen, K.-C. (2008). Carrier sensing based multiple access protocols for cognitive radio networks. Em *Communications, 2008. ICC '08. IEEE International Conference on*, pp. 3208–3214.

McHenry, M. A. (2005). Nsf spectrum occupancy measurements project summary. http://www.sharedspectrum.com/inc/content/measurements/nsf/NSF_Project_Summary.pdf. Acessada em 21/01/2010.

Mitola, J., I. (1993). Software radios: Survey, critical evaluation and future directions. *Aerospace and Electronic Systems Magazine, IEEE*, 8(4):25–36.

Mitola, J., I. & Maguire, G.Q., J. (1999). Cognitive radio: making software radios more personal. *Personal Communications, IEEE*, 6(4):13–18.

Padhye, J.; Agarwal, S.; Padmanabhan, V. N.; Qiu, L.; Rao, A. & Zill, B. (2005). Estimation of link interference in static multi-hop wireless networks. Em *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, IMC '05, pp. 28--28, Berkeley, CA, USA. USENIX Association.

Polastre, J.; Hill, J. & Culler, D. (2004). Versatile low power media access for wireless sensor networks. Em *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pp. 95--107, New York, NY, USA. ACM.

Pollin, S.; Tan, I.; Hodge, B.; Chun, C. & Bahai, A. (2008). Harmful coexistence between 802.15.4 and 802.11: A measurement-based study. Em *Cognitive Radio Oriented Wireless Networks and Communications, 2008. CrownCom 2008. 3rd International Conference on*, pp. 1 –6.

Saaty, T. (2000). *Fundamentals of the Analytic Hierarchy Process*. RWS Publications, 4922 Ellsworth Avenue, Pittsburgh, PA 15413.

Saucier, R. (2000). Computer generation of statistical distributions. URL: http://ftp.arl.mil/random/random.pdf Acessada em 01/01/2011.

Shankar, N.; Cordeiro, C. & Challapali, K. (2005). Spectrum agile radios: utilization and sensing architectures. Em *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 160 –169.

Silva, R.; Sa Silva, J.; Simek, M. & Boavida, F. (2009). A new approach for multi-sink environments in wsns. Em *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, pp. 109 –112.

Texas Instruments, C. (2006). Data sheet do rádio cc2420. URL: http://www-mtl.mit.edu/Courses/6.111/labkit/datasheets/CC2420.pdf Acessada em 05/01/2010.

van Dam, T. & Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. Em *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pp. 171--180, New York, NY, USA. ACM.

Wainer, J. (2007). Métodos de pesquisa quantitativa e qualitativa para a ciência da computação. URL: http://www.dcc.unicamp.br/ wainer/papers/metod07.pdf. Acessada em 15/10/2009.

Ye, W.; Heidemann, J. & Estrin, D. (2004). Medium access control with coordinated adaptive sleeping for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(3):493 – 506.

Zhao, J.; Zheng, H. & Yang, G.-H. (2005). Distributed coordination in dynamic spectrum allocation networks. Em *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 259–268.

Zhou, G.; Stankovic, J. A. & Son, S. H. (2006). Crowded spectrum in wireless sensor networks. Em *in Proceedings of Third Workshop on Embedded Networked Sensors (EmNets.*