

**FS-MAC: UM SISTEMA PARA FLEXIBILIZAÇÃO  
DA SUBCAMADA MAC EM REDES SEM FIO**



JEFFERSON RAYNERES SILVA CORDEIRO

**FS-MAC: UM SISTEMA PARA FLEXIBILIZAÇÃO  
DA SUBCAMADA MAC EM REDES SEM FIO**

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

ORIENTADOR: LUIZ FILIPE MENEZES VIEIRA  
COORIENTADOR: DANIEL FERNANDES MACEDO

Belo Horizonte  
Fevereiro de 2017

© 2017, Jefferson Rayneres Silva Cordeiro.  
Todos os direitos reservados.

Cordeiro, Jefferson Rayneres Silva

C794f FS-MAC: Um sistema para flexibilização da subcamada MAC em redes sem fio / Jefferson Rayneres Silva Cordeiro. — Belo Horizonte, 2017  
xxv, 105 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação.

Orientador: Luiz Filipe Menezes Vieira  
Coorientador: Daniel Fernandes Macedo

1. Computação — Teses. 2. Redes de sensores sem fio. 3. Sistemas de computação sem fio. 4. Subcamada MAC. I. Orientador. II. Coorientador. III. Título.

CDU 519.6\*22 (043)



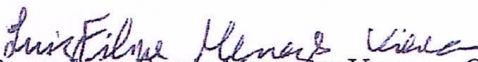
UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO


## FOLHA DE APROVAÇÃO

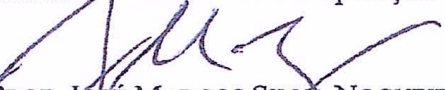
Fs-mac: um sistema para flexibilização da subcamada mac em redes sem fio

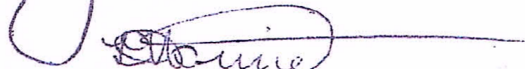
**JEFFERSON RAYNERES SILVA CORDEIRO**

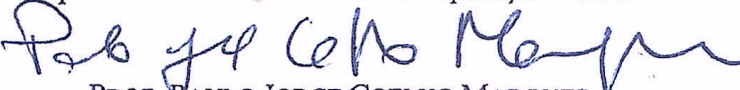
Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

  
PROF. LUIZ FILIPE MENEZES VIEIRA - Orientador  
Departamento de Ciência da Computação - UFMG

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

  
PROF. JOSÉ MARCOS SILVA NOGUEIRA  
Departamento de Ciência da Computação - UFMG

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

  
PROF. PAULO JORGE COELHO MARQUES  
Escola Superior de Tecnologia - IPCB

Belo Horizonte, 21 de fevereiro de 2017.



*Dedico este trabalho à minha família, parte fundamental do que me tornou um curioso incurável.*





# Agradecimentos

Começo agradecendo à minha família, que é a parte maior do que me tornei ao longo dos anos. Em especial a meus pais Dirce e Juvenal, que doaram com tanta dedicação o seu tempo e esforço para que eu pudesse alçar voo um pouco mais seguro pela vida. Agradeço ainda, com igual grandeza, a minha querida companheira Lettícia, por ter compreendido com amor e paciência as minhas ausências e me apoiado quando o fardo parecia pesado demais.

Agradeço imensamente aos meus orientadores Daniel Macêdo e Luiz Filipe, pelo conhecimento valioso compartilhado e pelas discussões científicas que me proporcionaram enorme crescimento. Ao colegas do laboratório Winet pelas discussões e apoio, em especial aos amigos Erik, Alisson, Pablo e Luiz pelo apoio humano quando tudo parecia dar errado.

Agradeço aos grandes amigos Eduardo e Fábio, minhas referências pessoais, que mesmo tendo pouco contato, mantêm comigo uma amizade inabalável. Ao professor Loureiro, grande incentivador, por me fazer enxergar que eu posso ir mais longe do que eu achei que poderia.

Agradeço à Esthefanie pelo apoio com os testes em momentos críticos. Ao GRUBI (Grupo de Redes Ubíquas) e ao FUTEBOL (Federated Union of Telecommunications Research Facilities for an EU-Brazil Open Laboratory) por cederem gentilmente seus equipamentos para que eu pudesse realizar meus experimentos. Agradeço também ao laboratório empresa Synergia, sem o qual eu não teria condições de permanecer no curso.

Por fim, agradeço a todos aqueles que contribuíram, mesmo que de forma discreta, para que eu pudesse concluir essa jornada.



*“De modo geral, não é a observação de fenômenos raros e escondidos que só são apresentáveis por meio de experimentos que serve para a descoberta das mais importantes verdades, mas a observação daqueles fenômenos que são evidentes e acessíveis a todos. Por isso a tarefa não é ver o que ninguém viu ainda, mas pensar aquilo que ninguém pensou a respeito daquilo que todo mundo vê.”*

*(Arthur Schopenhauer)*



# Resumo

As redes sem fio atuais são muito dinâmicas e abrigam uma grande diversidade de aplicações com necessidades bem diferentes. Como consequência, existem vários padrões para lidar com as características e problemas específicos dessas redes como alta interferência, meio variável com o tempo e mobilidade. Além disso, dentro da mesma rede podem conviver aplicações com necessidades diversas. Aplicações de voz e vídeo, por exemplo, exigem uma latência baixa para que haja qualidade na transmissão, mas suportam taxas razoáveis de erro. Por outro lado, aplicações de dados como serviços de mensagem e web são pouco tolerantes a erros, mas não possuem baixa latência como um requisito essencial. Esse cenário de diversidade faz surgir a necessidade por equipamentos e redes mais flexíveis, que se adaptem às características do contexto.

Na subcamada MAC (*Medium Access Control*), um dos problemas que ainda persistem é o fato de os protocolos não serem eficientes em todos os cenários. Protocolos baseados em TDMA (*Time Division Multiple Access*), por exemplo, possuem boa vazão quando o meio é muito disputado, mas devido ao *overhead* das mensagens de controle usadas na alocação dos *slots* de tempo, não são adequados quando o meio está descongestionado. Já os protocolos baseados em CSMA (*Carrier sense multiple access*) são adequados para redes esparsas, pois não possuem esse *overhead*, mas perdem qualidade com o aumento de colisões de mensagens quando o meio está congestionado.

O presente trabalho aborda esse problema flexibilizando o controle de acesso ao meio. Para isso propomos uma arquitetura chamada FS-MAC (*Flexible System for Medium Access Control*), FS-MAC foi baseada na estrutura de funcionamento de rádios cognitivos e seu objetivo é permitir que mais de um protocolo possa ser utilizado na rede, colocando cada um em atividade na situação em que é mais eficiente. Além disso, a solução utiliza um sistema *Fuzzy* para tomar decisões e permite que novos protocolos possam ser acrescentados ao conjunto já existente sem que para isso seja necessário refazer toda a subcamada. A partir da arquitetura, implementamos um protótipo utilizando SDR (*Software Defined Radio*), uma tecnologia que permite a programação de rádios, possibilitando a criação de protocolos e aplicações com alto

grau de flexibilidade.

O protótipo foi testado em um *testbed* onde variamos a carga e o número de estações transmitindo. Em relação à vazão, os resultados mostraram que FS-MAC foi capaz de reconhecer o melhor cenário para cada protocolo e colocar cada um em operação no momento em que é mais eficiente. Realizamos também testes onde modificamos o objetivo da rede, priorizando a minimização do número de retransmissões. Nos dois casos, FS-MAC apresentou desempenho similar ao protocolo mais eficiente em cada momento, dispendo apenas de um *overhead* em torno de 2%.

Além dos resultados apresentados, o trabalho gerou como contribuições as seguintes publicações: *FS-MAC: Uma Plataforma para a Flexibilização da Subcamada MAC em Redes Sem Fio* [Cordeiro et al., 2017], *Experimental Wireless Networking Research using Software-Defined Radios* [Souza et al., 2017] e *Introdução a Rádios Definidos por Software com aplicações em GNU Radio* [Silva et al., 2015];

**Palavras-chave:** Redes sem fio, Subcamada MAC, Flexibilização, Rádio Definido por Software.

# Abstract

Current wireless networks are very dynamic and have a wide variety of applications with very different requirements. As a consequence, there are many communication standards that deal with the specific problems and characteristics of each network, such as high interference, time varying channels and mobility. Moreover, applications with diverse requirements usually coexist within the same network. Applications such as voice and video, for example, require low latency to ensure quality in the transmission, but tolerate reasonable error rates. On the other hand, data applications such as web and message services have low error tolerance, however low latency is not an essential requirement. This diversity raises the need for more flexible equipment as well as networks that adapt to the context of the network.

In the MAC sublayer, one of the problems that still persist is that the protocols are not efficient in all scenarios. Protocols based on TDMA (*Time Division Multiple Access*), for example, have good throughput when the medium is congested. However, due to the overhead of control messages used to allocate the slots, they are not suitable when the medium is not congested. In contrast, protocols based on CSMA (*Carrier sense multiple access*) are suitable for sparse networks because they have low overhead, but their performance decreases due to frequent message collisions when the medium is congested.

This master thesis addresses proposes to increase the flexibility on the medium access control. We propose an architecture called FS-MAC (*Flexible System for Medium Access Control*), which is based on the working structure of cognitive radio. It allows more than one protocol to be used on the network, activating each one in the situation where it is most efficient. In addition, the solution uses a Fuzzy system to make decisions and allows new MAC protocols to be added to the existing set without changing the entire sublayer. We implemented a prototype of the architecture using Software Defined Radios (SDR), a technology that allows the programming of wireless protocols and applications with a high degree of flexibility.

The prototype was tested in a testbed where we varied the load and number of

connected stations. Regarding throughput, the results showed that FS-MAC was able to recognize the best scenario for each protocol and to activate each one when it is most efficient. We also performed tests where we modified the optimization objective, prioritizing the minimization of the number of retransmissions. In both cases, FS-MAC showed similar performance to the most efficient protocol at any given time, having an overhead of around 2%.

In addition to the presented results, this work generated as contributions the following publications: *FS-MAC: Uma Plataforma para a Flexibilização da Subcamada MAC em Redes Sem Fio* [Cordeiro et al., 2017], *Experimental Wireless Networking Research using Software-Defined Radios* [Souza et al., 2017] and *Introdução a Rádios Definidos por Software com aplicações em GNU Radio* [Silva et al., 2015];

**Keywords:** Wireless networks, MAC sublayer, Flexibilization, Software Defined Radio.



# Lista de Figuras

2.1	Métodos de acesso múltiplo. [Busch et al., 2004] . . . . .	8
2.2	Exemplo de um SDR reconfigurável. Silva et al. [2015] . . . . .	14
2.3	Arquitetura básica SDR. Silva et al. [2015] . . . . .	18
4.1	Arquitetura FS-MAC . . . . .	33
4.2	Correspondência entre Rádios cognitivos e FS-MAC . . . . .	35
4.3	Detalhes do módulo de Decisão usando duas métricas e dois protocolos . .	38
4.4	Implementação FS-MAC . . . . .	43
4.5	Função de pertinência para número de transmissores. . . . .	47
4.6	Função de pertinência para latência média de entrega dos pacotes (CSMA).	48
4.7	Função de pertinência para latência média de entrega dos pacotes (TDMA).	48
4.8	Função de pertinência para adaptabilidade dos protocolos. . . . .	48
4.9	Troca de protocolo em operação na rede . . . . .	50
5.1	Diagrama de topologia e comunicação dos experimentos. . . . .	55
5.2	Organização das USRPs durante os experimentos. . . . .	56
5.3	Montagem dos blocos no GNU Radio para um nó Comum. . . . .	58
5.4	Etapas do Experimento com inclusão gradativa de transmissores . . . . .	59
5.5	Superquadro do TDMA . . . . .	60
5.6	Vazão total da rede por número de transmissores . . . . .	61
5.7	Taxa de entrega média dos pacotes por número de transmissores . . . . .	62
5.8	Taxa de retransmissão média por número de transmissores . . . . .	63
5.9	Latência média de entrega dos pacotes por número de transmissores . . . .	64
5.10	Vazão total da rede por número de transmissores . . . . .	66
5.11	Taxa de entrega média dos pacotes por número de transmissores . . . . .	68
5.12	Taxa de retransmissão média por número de transmissores . . . . .	68
5.13	Latência média de entrega dos pacotes por número de transmissores . . . .	68
5.14	Função de pertinência para número de transmissores. . . . .	69

5.15	Função de pertinência para latência média (CSMA).	70
5.16	Função de pertinência para latência média (TDMA).	70
5.17	Função de pertinência para adaptabilidade dos protocolos.	70
5.18	FS-MAC com regras que priorizam menor taxa de perda	71
5.19	Impacto na vazão causado por sequência de trocas consecutivas para cinco transmissores	73
5.20	Desempenho do FS-MAC com múltiplas trocas em relação aos outros protocolos	74
5.21	Desempenho do FS-MAC com múltiplas trocas em relação ao FS-MAC normal	75
A.1	Superquadro do TDMA	97
A.2	Fluxograma do CSMA - Transmissor	102
A.3	Fluxograma do CSMA - Receptor	103
A.4	Fluxograma do TDMA - Nó Comum	104
A.5	Fluxograma do TDMA - Nó coordenador	105

# Lista de Tabelas

3.1	Comparação entre os principais trabalhos . . . . .	30
5.1	Equipamento utilizado nos experimentos . . . . .	55
5.2	Parâmetros CSMA . . . . .	57
5.3	Parâmetros TDMA . . . . .	57
5.4	Média de pacotes confirmados em relação a pacotes de controle . . . . .	66
5.5	Total de pacotes confirmados e rendimento em relação ao FS-MAC . . . . .	67
5.6	Tempo de operação dos protocolos durante o experimento . . . . .	73
5.7	<i>Overhead</i> do FS-MAC modificado em relação ao FS-MAC operando normalmente para o cenário com cinco transmissores . . . . .	73



# Lista de Algoritmos

1	Procedimento de troca no nó Coordenador . . . . .	40
2	Procedimento de troca no nó Comum . . . . .	40



# Sumário

Agradecimentos	ix
Resumo	xiii
Abstract	xv
Lista de Figuras	xvii
Lista de Tabelas	xix
<b>1 Introdução</b>	<b>1</b>
1.1 O problema . . . . .	3
1.2 Objetivos . . . . .	3
1.3 Contribuições . . . . .	4
1.4 Organização da dissertação . . . . .	5
<b>2 Conceitos Fundamentais</b>	<b>7</b>
2.1 Protocolos MAC . . . . .	7
2.1.1 CSMA/CA . . . . .	9
2.1.2 TDMA . . . . .	11
2.1.3 DFWMAC . . . . .	12
2.2 Rádio Definido por Software . . . . .	13
2.2.1 História . . . . .	14
2.2.2 Aplicações . . . . .	16
2.2.3 Funcionamento do SDR . . . . .	17
2.2.4 Plataformas e desenvolvimento . . . . .	18
2.3 Rádios cognitivos . . . . .	20
<b>3 Trabalhos relacionados</b>	<b>23</b>
3.1 Plataformas para MAC flexível . . . . .	23

3.2	Otimização de protocolos específicos . . . . .	25
3.3	Flexibilização com abordagem híbrida . . . . .	27
3.4	Discussão . . . . .	29
<b>4</b>	<b>FS-MAC: <i>Flexible System for Medium Access Control</i></b>	<b>31</b>
4.1	Arquitetura . . . . .	32
4.1.1	FS-MAC e Rádios cognitivos . . . . .	34
4.1.2	Sensoriamento . . . . .	34
4.1.3	Decisão . . . . .	36
4.1.4	Troca . . . . .	37
4.1.5	Conjunto de protocolos . . . . .	40
4.2	Implementação . . . . .	41
4.2.1	Protocolos . . . . .	44
4.2.2	Sensoriamento . . . . .	45
4.2.3	Decisão . . . . .	47
4.2.4	Troca . . . . .	49
<b>5</b>	<b>Avaliação experimental</b>	<b>53</b>
5.1	Convenções . . . . .	53
5.2	Metodologia . . . . .	54
5.2.1	Equipamentos e sistema . . . . .	54
5.2.2	Topologia . . . . .	54
5.2.3	Carga de dados e modos de operação . . . . .	56
5.2.4	Parâmetros . . . . .	56
5.2.5	Dinâmica dos experimentos . . . . .	57
5.3	Avaliação e caracterização dos protocolos . . . . .	59
5.3.1	Vazão em relação à variação do número de transmissores (CSMA e TDMA) . . . . .	59
5.4	Avaliação da plataforma FS-MAC . . . . .	64
5.4.1	Vazão em relação à variação do número de transmissores (FS-MAC) . . . . .	65
5.4.2	Flexibilização das regras FS-MAC para priorizar a menor taxa de retransmissão de pacotes . . . . .	69
5.4.3	Comportamento da vazão na ocorrência de múltiplas trocas . . . . .	72
<b>6</b>	<b>Conclusão e Trabalhos Futuros</b>	<b>77</b>
	<b>Referências Bibliográficas</b>	<b>81</b>



<b>Apêndice A</b>	<b>Descrição detalhada da implementação</b>	<b>89</b>
A.1	Aspectos comuns . . . . .	89
A.1.1	Bibliotecas e Estruturas de dados . . . . .	90
A.1.2	Temporizações . . . . .	92
A.2	CSMA . . . . .	93
A.2.1	<i>Carrier sense</i> . . . . .	94
A.2.2	Receptor . . . . .	94
A.2.3	Transmissor . . . . .	95
A.3	TDMA . . . . .	96
A.3.1	Mensagens de controle . . . . .	97
A.3.2	Funcionamento dos nós . . . . .	98
A.4	Interface de troca . . . . .	99



# Capítulo 1

## Introdução

As redes sem fio atuais são dinâmicas e suas características possibilitam a existência de uma grande diversidade de aplicações. As redes de telecomunicação móvel, por exemplo, se difundiram globalmente e são utilizadas hoje por bilhões de usuários para a transmissão de voz e dados de diversas formas [Labib et al., 2016]. As redes de sensores sem fio também possibilitaram a criação de várias soluções nas áreas de segurança, saúde, agricultura, indústria, urbanismo, entre outras [Rawat et al., 2014]. Surgiram também muitas aplicações voltadas para atividades militares [Đurišić et al., 2012]. Além disso, difundiram-se as aplicações que utilizam, separadamente ou combinadas, tecnologias como RFID, GPS, Wi-Fi, *Bluetooth*, etc. [Dao et al., 2014].

Com as várias possibilidades que surgiram, multiplicaram-se também os padrões utilizados pelas redes para se comunicar. Esses padrões foram agrupados em conjuntos chamados famílias. Cada grupo sintetiza as necessidades de um contexto específico, contemplando características como mobilidade, tamanho da rede, demanda por energia, processamento, memória, dentre outras. Algumas dessas famílias tratam especificamente dos problemas referentes às camadas Física e Enlace de redes sem fio.

A família 802.11, por exemplo, trata das redes WLAN (*Wireless Local Area Network*). Essa família possui várias versões como 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac e 802.11ad, com variações de faixa frequência utilizada e principalmente de largura de banda. As redes Wi-Fi são baseadas nessa família e são bastante utilizadas em ambientes locais como redes domésticas, aeroportos, universidades e escritórios. A família 802.16 trata de redes WMAN (*Metropolitan Area Network*), possuem alcance de alguns quilômetros e banda na ordem de Mb/s. As redes de telefonia móvel WiMax são baseadas nessa família. Já a família 802.15 trata de redes WPAN (*Wireless Personal Area Network*). Essa família possui várias sub-divisões como 802.15.1 (*Bluetooth*), desenhado para dispositivos fixos ou móveis, funciona na faixa de 2,4 a 2,485 GHz e

integra aparelhos como *smartphones*, mouses e teclados; 802.15.3 (*High Rate WPAN*), projetado para transferências multimídia, atinge taxas na ordem de Gb/s e são utilizadas em dispositivos como *smart TVs* e *home theaters*; 802.15.4 *ZigBee*, pensado para baixas taxas de transferência e em compensação, grande economia de energia. Opera na faixa de 2,4 a 2,48 GHz e é utilizado em várias aplicações que envolvem redes de sensores como segurança, monitoramento ambiental e monitoramento de redes elétricas.

Além dessa diversidade de padrões, dentro da mesma rede, várias aplicações podem ter necessidades distintas que exijam um certo comportamento dessa rede. Consideremos, por exemplo, uma rede de celular que promove a comunicação entre diversos dispositivos, entre eles, *smartphones*. Esses aparelhos possuem aplicações que executam chamadas de voz, chamadas de vídeo, exibição de páginas web, aplicativos de troca de mensagens, etc. As aplicações de chamada de voz e vídeo possuem uma tolerância razoável a erros, pois a perda de alguns pacotes não compromete o entendimento da mensagem por parte do usuário. Por outro lado, essas aplicações exigem uma latência pequena, pois no nível do usuário, atrasos grandes prejudicam muito este tipo de comunicação. Já as aplicações web e de mensagem de texto por exemplo, suportam latências maiores, pois em geral não há grandes problemas em “esperar a mensagem chegar” ou “esperar a página carregar”, por exemplo. No entanto, a tolerância a erros nesses casos é pequena, pois a falta mínima de alguns dados pode comprometer a exibição e o entendimento do conteúdo. Assim, torna-se desejável que essa rede seja flexível o bastante para adaptar o seu comportamento de acordo com as necessidades das aplicações, alterando dinamicamente parâmetros de um protocolo, por exemplo.

Na subcamada MAC há também essa demanda por flexibilidade, pois os protocolos existentes que controlam o acesso ao meio não são eficientes em todos os contextos. Gummalla & Limb [2000] mostram que propriedades inerentes ao meio sem fio, tais como *Operação half-duplex*, *Canal variável com o tempo* e *Canal com erros em rajada* tornam a construção de protocolos MAC para redes sem fio mais desafiadora do que no caso de sistemas com fio. Nessa camada, é possível utilizar protocolos com abordagens diferentes, com contenção, sem contenção, utilizando ou não controle de fluxo, com *Automatic Repeat Request* (ARQ) ou com *Forward Error Correction* (FEC), por exemplo. O problema é que nenhum protocolo é conveniente em todos os contextos da rede, sendo eficientes quando a rede está em determinados estados e ineficientes em outros. Na maioria dos casos, tanto os dispositivos quanto os protocolos utilizados possuem pouca ou nenhuma flexibilidade para lidar com essa situação.

Protocolos baseados em CSMA por exemplo, trabalham com verificação do meio, tentam evitar colisões, possuem métodos de contenção para lidar com os conflitos de

tentativa de acesso ao meio. Na abordagem desses protocolos, a tentativa de transmissão é direta, dependendo apenas da verificação do meio e da espera de algumas temporizações previamente definidas, neste caso, quando não há colisões, o tempo é bem aproveitado. No entanto, com essa abordagem, o número de colisões tende a crescer com o aumento de dispositivos tentando se comunicar. Em certo momento, o número de colisões pode ser grande o suficiente para prejudicar o funcionamento da rede [Takagi & Kleinrock, 1985; Ziouva & Antonakopoulos, 2002].

Utilizando outra abordagem, os protocolos baseados em TDMA dividem o tempo em *slots* onde cada dispositivo faz a sua transmissão individualmente no tempo. Como um *slot* é reservado para apenas um dispositivo, não há colisões, não importa quantos dispositivos estejam tentando se comunicar. Portanto, é uma boa abordagem quando a rede possui muitos nós tentando transmitir. Apesar dessa vantagem, quando há poucas tentativas de transmissão, as mensagens de controle trocadas no procedimento de alocação dos *slots* consomem um tempo relativo grande. Assim, esses protocolos são melhores quando a rede está cheia, mas perdem desempenho quando há poucos nós tentando se comunicar.

Diante dos aspectos apresentados, percebemos uma necessidade por redes e dispositivos mais flexíveis que possam se adaptar ao contexto da comunicação para maximizar o aproveitamento dos recursos e atender corretamente aos requisitos.

## 1.1 O problema

O problema que nos motivou decorre da percepção de que os protocolos atuais de controle de acesso ao meio não são eficientes em todos os cenários e as implementações desses protocolos são pouco flexíveis a mudanças e adaptações. Diante do que foi apresentado, nos perguntamos se seria possível promover controle de acesso ao meio em redes sem fio de forma flexível o suficiente para aproveitar o melhor de cada protocolo.

## 1.2 Objetivos

A fim de melhorar o acesso ao meio flexibilizando o uso dos protocolos na subcamada MAC, os objetivos deste trabalho são:

1. Elaborar detalhadamente uma arquitetura que atenda satisfatoriamente os requisitos que tornem possível:
  - Utilizar um conjunto de protocolos para controle de acesso ao meio.

- Incluir facilmente ao conjunto novos protocolos implementados sem prejudicar o funcionamento dos já existentes e sem a necessidade de alterá-los.
  - Configurar facilmente protocolos incluídos para que sejam selecionados para utilização na situação em que são eficientes e evitados nos cenários em que são ineficientes.
  - Monitorar as diversas características da rede para apoiar a decisão de qual protocolo utilizar em cada momento.
  - Alternar entre protocolos automaticamente sem prejuízo para a comunicação.
2. Implementar um protótipo baseado na arquitetura elaborada utilizando a tecnologia SDR. O protótipo deve monitorar no mínimo duas características da rede e utilizar pelo menos dois protocolos diferentes, um baseado em contenção e outro livre de contenção.
  3. Testar o protótipo em *testbed* com número não trivial de estações. Nos testes, analisar a capacidade do protótipo de determinar o melhor cenário para cada protocolo e sua eficácia em colocá-los em operação no momento de ocorrência desse cenário. É também um objetivo analisar a flexibilidade quanto à mudança das regras que realizam a decisão de qual protocolo é mais adequado em cada momento.

### 1.3 Contribuições

Nessa dissertação apresentamos a elaboração de uma arquitetura com alta flexibilidade chamada FS-MAC para a subcamada MAC de redes sem fio. Implementamos um protótipo que permite a utilização de mais de um protocolo nessa subcamada e a configuração de regras que determinam a sua utilização<sup>1</sup>.

As principais contribuições deste trabalho são:

1. A criação de uma arquitetura para a subcamada MAC de redes sem fio chamada FS-MAC, que pode abrigar vários protocolos e possui alta flexibilidade para utilizar cada um na situação em que é mais eficiente.
2. A generalização das funções de rádios cognitivos para que possam ser utilizadas na solução de outros problemas com características similares.

---

<sup>1</sup>Código disponível em <https://github.com/jeffRayneres/FS-MAC>

3. Avaliação da arquitetura FS-MAC em um cenário onde há a troca de utilização entre um protocolo baseado em contenção e outro sem contenção.
4. Publicação dos trabalhos:
  - FS-MAC: Uma Plataforma para a Flexibilização da Subcamada MAC em Redes Sem Fio [Cordeiro et al., 2017]
  - *Experimental Wireless Networking Research using Software-Defined Radios* [Souza et al., 2017]
  - Introdução a Rádios Definidos por Software com aplicações em GNU Radio [Silva et al., 2015]

## 1.4 Organização da dissertação

Esta dissertação é composta por cinco capítulos além da Introdução. No capítulo 2 abordamos os conceitos que julgamos fundamentais para o desenvolvimento deste trabalho e discutimos alguns aspectos necessário para o entendimento da solução. No capítulo 3 analisamos os trabalhos que se relacionam com a nossa proposta, dividimos esses trabalhos em três categorias e discutimos cada uma delas em relação à abordagem adotada por nós. No capítulo 4 apresentamos a plataforma FS-MAC, explicamos a sua arquitetura e as decisões envolvidas na sua elaboração. Nesse capítulo descrevemos também a implementação de um protótipo construído a partir da arquitetura apresentada. No capítulo 5 descrevemos a avaliação da plataforma utilizando o protótipo implementado, explicamos detalhadamente cada caso de uso e discutimos os resultados dos experimentos realizados. No capítulo 6 concluímos o trabalho e trazemos alguns tópicos que futuramente podem ser explorados por meio da solução proposta nesta dissertação.





# Capítulo 2

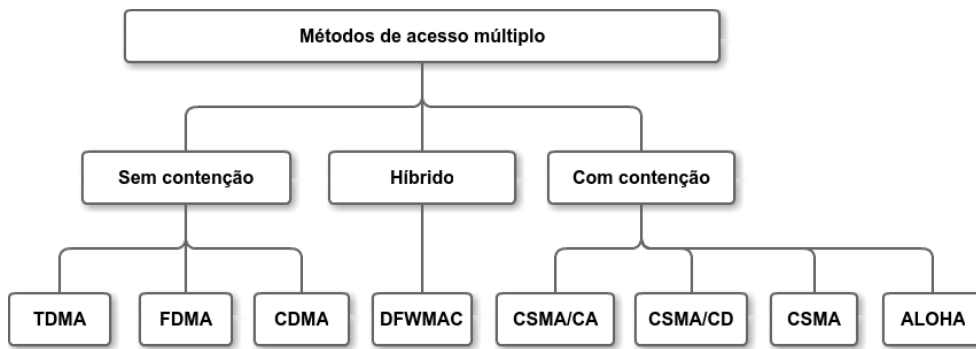
## Conceitos Fundamentais

Neste capítulo, apresentamos aqueles conceitos que julgamos fundamentais para chegarmos à solução apresentada no Capítulo 4 e cujos detalhes precisam ser compreendidos para que a apresentação da solução desenvolvida fique clara. Na Seção 2.1 apresentamos os protocolos e técnicas da subcamada MAC, mostramos uma divisão baseada no uso ou não de mecanismos de contenção, falamos brevemente sobre o seu funcionamento e sobre as vantagens e desvantagens de cada abordagem. Os métodos apresentados como exemplo são aqueles utilizados na implementação do nosso trabalho. Na Seção 2.2 falamos sobre *Rádios Definidos por Software*, apresentamos sua história, suas aplicações, funcionamento e as plataformas utilizadas para desenvolvimento usando a tecnologia. SDR é utilizado nesse trabalho como plataforma para a criação de uma subcamada MAC flexível; na Seção 2.3 discutimos o conceito de Rádios cognitivos, apresentamos a ideia geral, os requisitos envolvidos e as funções de uma rede que os utiliza. As definições e mecanismo envolvidos na organização e funcionamento de rádios cognitivos são apropriadas por nós como apoio ao desenvolvimento do FS-MAC.

### 2.1 Protocolos MAC

Os métodos de acesso múltiplo em redes sem fio são utilizados quando o meio é compartilhado e é necessária alguma forma de organização para que os diferentes nós consigam transmitir satisfatoriamente seus dados. Existem diversos métodos que cumprem esse papel e eles podem ser divididos em duas grandes categorias: métodos de acesso múltiplo com contenção e sem contenção, conforme mostra a Figura 2.1.

Os métodos com contenção são aqueles em que os transmissores não têm restrições fortes para acessar o meio, podendo acessá-lo mais livremente. O acesso nesse caso não se dá de forma coordenada e a decisão de transmitir ou não é tomada localmente. Não



**Figura 2.1.** Métodos de acesso múltiplo. [Busch et al., 2004]

há também uma delimitação de tempo específica dentro da qual a transmissão deve ser realizada. Por essas razões, são adotadas medidas de contenção para diminuir o número de colisões e possibilitar que o canal não seja saturado e assim a rede funcione satisfatoriamente. De forma geral, quando acontece uma colisão, o nó espera um tempo aleatório e repete a tentativa de transmissão. As primeiras técnicas a tentarem essa abordagem foram ALOHA puro e *slotted* ALOHA. Essa abordagem foi melhorada incluindo a verificação do meio antes da tentativa de transmissão, surgindo assim o CSMA *Carrier sense multiple access* e mais tarde as suas variações com detecção de colisões (CSMA/CD) e com prevenção de colisão (CSMA/CA).

Os métodos sem contenção são aqueles em que o meio é acessado de forma coordenada e que não é necessário haver mecanismos para resolver conflitos de acesso e colisões. O controle de alocação do canal nesse caso pode ser feito por uma entidade centralizada, que coordena a ordem de transmissão, ou por algum mecanismo distribuído como *Passagem de ficha* ou *Fila distribuída*. Alguns exemplos de técnicas que não utilizam mecanismos de contenção são *Frequency Division Multiple Access* (FDMA), *Time Division Multiple Access* (TDMA) e *Code Division Multiple Access* (CDMA) [Busch et al., 2004].

Existe ainda uma abordagem híbrida, que utiliza métodos com contenção e sem contenção combinados. Nessa abordagem, cada modo opera durante um determinado tempo e há uma alternância entre os dois ao final de cada período. No período com contenção *Contention Period* (CP), a organização do acesso ao meio é distribuída, e no período livre de contenção *Contention Free Period* (CFP), a utilização do meio é coordenada por um ponto de acesso. Um exemplo que utiliza essa abordagem é o DFWMAC [Diepstraten & WCND-Utrecht, 1993].

A seguir descrevemos um exemplo de cada grupo, CSMA/CA como exemplo de acesso múltiplo com utilização de mecanismos de contenção, o TDMA como exemplo

de acesso sem os mecanismos de contenção e o DFWMAC como exemplo de abordagem híbrida. Mostramos suas características gerais e destacamos suas vantagens e desvantagens.

### 2.1.1 CSMA/CA

A técnica *Carrier sense multiple access with collision avoidance* é uma variação da CSMA [Tanenbaum & Wetherall, 2011]. Esses métodos têm como principais características a verificação do meio antes da tentativa de transmissão e um mecanismo de contenção para lidar com colisões. Como em redes sem fio não é possível detectar colisões apenas ouvindo o canal de transmissão, o CSMA/CA é conveniente nesse caso, pois ele tenta prevenir essas colisões. Quando elas ocorrem, são detectadas através do não recebimento da mensagem de confirmação. Esse método é utilizado no protocolo Wi-Fi, por exemplo, que implementa o padrão 802.11 e suas variações.

A verificação do meio é simples, o nó que deseja transmitir deve apenas ouvir o canal para detectar se há alguma transmissão em andamento e a partir disso decidir se tenta transmitir ou não. Este processo diminui a probabilidade de colisões, mas não é o suficiente para que se tenha certeza de que não há uma transmissão em andamento. Algumas outras transmissões podem estar ocorrendo longe do alcance do nó que deseja transmitir, mas no domínio de interferência do receptor. Dessa forma, mesmo que o nó não detecte transmissão ouvindo o meio, quando enviar uma mensagem, ela pode colidir com outra no receptor. Uma forma para o CSMA/CA saber que houve colisão no receptor é a ausência da confirmação da mensagem, quando isso acontece, algumas medidas são tomadas para continuar a comunicação.

O CSMA/CA utiliza como mecanismo de contenção o algoritmo *Backoff binário exponencial* (BEB). Esse mecanismo entra em ação em duas situações, a primeira é quando há colisão, ou seja, uma mensagem foi enviada e o nó transmissor não recebeu confirmação. A segunda acontece quando o nó que deseja transmitir ouve o meio e detecta que existe uma transmissão em andamento. Nesses dois casos, esse nó assume que há muitos nós transmitindo no mesmo momento e então inicia a contenção. A contenção no CSMA/CA consiste em esperar uma quantidade aleatória de *slots* de tempo entre 0 e um valor máximo, este intervalo é chamado *janela de contenção*. Todas as vezes que um dos eventos motivadores da contenção ocorre, o valor da janela é dobrado. Por outro lado, quando ocorre uma comunicação satisfatória, o tamanho da janela volta ao valor inicial. Assim, de forma distribuída, os nós adaptam a frequência com que transmitem à quantidade de nós tentando transmitir, esse mecanismo diminui o número de colisões na rede.

O CSMA/CA pode implementar opcionalmente o mecanismo de *Request to Send/Clear to Send* (RTS/CTS) com a finalidade de minimizar os efeitos de Terminal Escondido e reduzir colisões [Gummalla & Limb, 2000]. Este mecanismo consiste no envio de uma mensagem de solicitação de transmissão (RTS) pelo nó que deseja transmitir e uma resposta do receptor com uma mensagem que informa que o meio está livre para transmitir (CTS). O RTS possui o endereço do receptor e o tempo de duração do envio da mensagem somado à confirmação. Todos os nós que ouvirem essas mensagens saberão que uma transmissão estará em andamento e quanto tempo ela irá durar. O conjunto de nós que ouve o RTS não é necessariamente o mesmo que ouve o CTS, portanto, com as duas mensagens, todos os vizinhos do transmissor e do receptor estarão cientes da comunicação e não irão interferir. Com esse mecanismo, colisões acontecem apenas no envio do RTS, o que diminui bastante o problema. No entanto, o envio dessas mensagens de controle podem ser um *overhead* muito grande na comunicação. Na prática, esse mecanismo só faz sentido para o envio de mensagens grandes, para as quais as colisões representam um problema maior.

Uma das vantagens do CSMA/CA, quando não utiliza o mecanismo RTS/CTS, é a transmissão direta, sem necessidade de trocas de mensagem de controle para esse fim. Com isso, uma parte maior da banda é utilizada para mensagens que estão carregando os dados que se deseja transmitir. Em cenários menos propícios à ocorrência de colisões, esse mecanismo melhora a utilização do tempo. Outra vantagem dessa técnica é que não há necessidade de componentes externos como um coordenador ou um ponto de sincronização para as transmissões. As decisões são tomadas localmente e não há processamento adicional por parte de nenhum dos nós ou qualquer outro dispositivo para que a comunicação aconteça. Além dessas vantagens, quando utiliza o mecanismo de RTS/CTS, o CSMA/CA tem a vantagem de minimizar bastante o problema de terminal escondido. Isso se deve ao fato de os vizinhos dos nós envolvidos na comunicação receberem pelo menos uma das mensagens de controle.

Por outro lado, há também pontos negativos, em ambientes com muitos nós tentando transmitir, o CSMA/CA não tem um bom desempenho [Ziouva & Antonakopoulos, 2002]. Nesse caso, as colisões começam a acontecer com mais frequência e as retransmissões prejudicam a performance da rede. Outro ponto desfavorável dessa técnica está relacionado ao método de contenção. Enquanto os nós estão decrementando o tempo de *backoff* de acordo com o algoritmo BEB, alguns *slots* de tempo ficam inutilizados. Mesmo que haja uma sequência grande de *slots* vazios, todos os nós irão decrementar o tempo de *backoff* até que seja 0 e nenhum deles irá transmitir até que isso aconteça. Há também uma desvantagem em relação à justiça, seus mecanismos de execução não garantem que todos os nós terão igual oportunidade de acesso ao meio.

### 2.1.2 TDMA

A técnica *Time Division Multiple Access* é um exemplo que não utiliza métodos de contenção. Esta técnica tem como principal característica a divisão do tempo em *slots*, que são usados por cada nó separadamente para fazer sua transmissão. Por essa razão não acontecem colisões, não importa quantos nós tenham a intenção de transmitir [Koopman & Upender, 1995]. Dessa forma, não há razões para adotar qualquer mecanismo de contenção.

Por dividir o tempo em *slots* para alocar os nós que desejam transmitir, o TDMA necessita que haja uma sincronização entre os participantes da rede. Essa sincronia pode ser exercida por um elemento externo gerador de um sinal de sincronização ou pode ser gerenciada por um dos nós, nesse caso, esse nó é o coordenador da comunicação. A alocação dos nós em cada *slot* também pode ser feita de forma distribuída ou centralizada. Quando é feita de forma centralizada, o nó coordenador é responsável por iniciar o tempo de comunicação com uma mensagem de sincronização, receber as requisições de comunicação e alocar os nós em seus *slots*. O nó coordenador pode participar da comunicação ou apenas coordenar.

A sequência de passos para haver comunicação na rede começa com o nó coordenador enviando uma mensagem de sincronia e dizendo a ordem em que os nós da rede devem requisitar um *slot* caso queiram se comunicar. Cada nó que deseja transmitir dados responde ao coordenador com uma mensagem de requisição, enviada na ordem estabelecida anteriormente. Então o nó coordenador estabelece a ordem de transmissões alocando cada nó em um *slot* e torna pública essa ordem. A partir daí todos os nós sabem quando deverão transmitir seus dados. Após todos transmitirem, o ciclo se reinicia com o coordenador enviando nova mensagem de sincronização.

Entre os *slots* de comunicação é preciso existir um período de guarda, esse período existe para evitar problemas ocasionados pelas diferenças de sincronização. Assim, quanto mais sincronizados estiverem os nós na rede, menor pode ser esse tempo de guarda e melhor o tempo é aproveitado.

A grande vantagem do TDMA é a ausência de colisões [Demirkol et al., 2006]. Como o tempo é dividido em *slots* e cada nó é alocado exclusivamente para um *slot*, apenas um nó irá transmitir por vez. Isso evita retransmissões e promove um aproveitamento da banda de comunicação. Ao contrário do CSMA/CA, quando a rede possui muitos nós tentando transmitir, no TDMA não ocorrem colisões e a vazão da rede tende a melhorar. Isso acontece porque além de não haver colisões, a quantidade de alguns pacotes de controle como os de sincronização e alocação dos *slots* não aumenta à medida em que mais nós estão tentando transmitir. Além disso, TDMA é uma técnica

que possui justiça, ou seja, devido à forma como o tempo é dividido, todos os nós têm igual oportunidade de se comunicar na rede.

Por outro lado, quando a rede está mais esparsa ou quando poucos nós estão tentando transmitir, a alocação de *slots* torna-se um *overhead* indesejável. Mesmo que apenas um nó queira transmitir é preciso haver a sincronização e a alocação do *slot* de tempo para só então o nó iniciar a transmissão. Outro ponto negativo do TDMA é a dependência de um mecanismo de sincronização, se este mecanismo falhar, toda a comunicação está comprometida. Além disso, se a sincronização não for precisa, é necessário que os períodos de guarda sejam maiores para cobrir as diferenças de sincronia. Isso acrescenta atrasos ao processo que podem diminuir muito a eficiência da comunicação.

É importante notar que apesar de não utilizar mecanismos de contenção para controlar o acesso ao meio, isso não significa que nenhuma contenção ocorra durante a execução do TDMA. É correto afirmar que não há contenção quando a rede está em estado contínuo, no entanto, durante a entrada e saída de nós, por exemplo, pequenos momentos de contenção podem ocorrer. Isso acontece porque ao entrar na rede o nó precisa de alguma maneira informar a sua presença e se adequar ao sistema de sincronização, dessa forma, ele pode esperar o tempo de uma rodada inteira de comunicação para só então ter a oportunidade de começar a transmitir.

### 2.1.3 DFWMAC

O *Distributed Foundation Wireless Medium Access Control* é um protocolo híbrido implementado no padrão 802.11 [Diepstraten & WCND-Utrecht, 1993]. Nesse padrão, é obrigatória a implementação de um método baseado em contenção, o CSMA/CA e são opcionais as implementações de dois mecanismos adicionais, o RTS/CTS e a operação em modo híbrido. O modo híbrido inclui um método com contenção e um método livre de contenção. O DFWMAC constitui a implementação completa, ou seja, incluindo o modo híbrido.

No DFWMAC, todo o período de comunicação é dividido em *Superframes* e cada *Superframe* é dividido em dois períodos chamados *Contention Free Period* (CFP) e *Contention Period* (CP), que ficam dispostos nessa ordem. Durante o CFP é executada a função chamada *Point Coordination Function* (PCF), que corresponde a um protocolo livre de contenção. Já durante o CP é executada a função *Distributed Coordination Function* (DCF), que corresponde ao método com contenção.

No DFWMAC, como DCF, é utilizada a técnica básica CSMA/CA, onde a decisão de transmissão é tomada de forma distribuída e as colisões são tratadas com

mecanismos de contenção exatamente como descrito na Seção 2.1.1. Este é o modo principal de acesso deste protocolo, estando disponível tanto para redes *ad hoc* quanto para redes com infraestrutura.

Como PCF, é utilizada uma forma de gerenciamento onde o ponto de acesso coordena o acesso ao meio. Este ponto de acesso decide qual nó irá transmitir em qual período de tempo. O coordenador irá percorrer uma lista das estações e para cada uma dará permissão para acessar o meio e informará um período de tempo no qual pode transmitir seus dados. Neste mecanismo, a primeira estação é autorizada, após terminado o seu tempo, o coordenador dá a permissão à próxima estação que deseja acessar o meio e assim por diante até que todos os nós tenham a chance de transmitir. Este modo é opcional e só é possível em redes infraestruturadas.

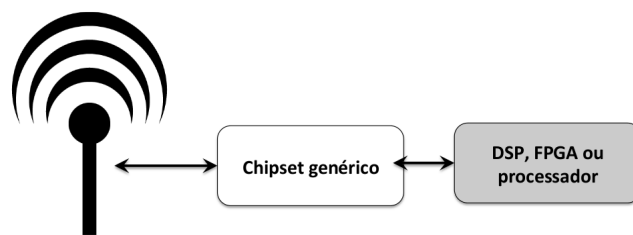
Os mecanismos de temporização do protocolo vão garantir a alternância entre o modo com contenção e o modo sem contenção. Assim, a rede terá comunicação síncrona e comunicação assíncrona, agregando as vantagens e desvantagens dessas duas formas.

Hoje, a implementação de todos os protocolos baseados nas técnicas descritas nas Seções acima é feita em *hardware* ou *firmware*. Por isso, geralmente é muito difícil ou impossível a sua troca, manutenção ou qualquer outro tipo de abordagem que exija um grau razoável de flexibilidade. Uma forma de permitir implementações mais flexíveis é o uso de Rádios Definidos por Software, que serão descritos na próxima seção.

## 2.2 Rádio Definido por Software

Existem várias definições diferentes de SDR, mas todas seguem o mesmo princípio. A definição considerada nessa dissertação foi proposta por Reis et al. [2012]: “SDR é um transceptor sem fio, onde módulos de software implementam as funcionalidades do transceptor. Estes módulos são executados em tempo real em microprocessadores genéricos, processadores de sinais digitais ou circuitos lógicos programáveis.”

Um desenho esquemático da tecnologia é apresentado na Figura 2.2. Neste exemplo, o bloco branco representa um componente baseado em *hardware* e o bloco cinza representa um componente baseado em *software*. A implementação em *hardware* contém o mínimo possível para que sejam realizadas as conversões de sinal entre analógico e digital e também a seleção da frequência de operação, pois não é possível realizar essas operações em *software*. Da esquerda para a direita são mostrados uma antena ligada a um chip genérico TX/RX, que por sua vez está ligado a um chip programável. A parte programável pode ser composta de FPGAs, DSPs e/ou CPUs que realizam operações de processamento de sinais em alta velocidade.



**Figura 2.2.** Exemplo de um SDR reconfigurável. Silva et al. [2015]

### 2.2.1 História

Nas décadas anteriores ao surgimento dos *Software radios*, os sistemas de rádio eram projetados pensando em um tempo de utilização de cerca de 30 anos, possuíam função única e eram otimizados para aplicações específicas. Com o avanços na área de radiocomunicação, o tempo de vida dos rádios diminuiu muito, pois a tecnologia passou a se renovar consideravelmente em períodos de meses. Com isso, em um dado momento, os militares trabalhavam com centenas de sistemas de comunicação que foram desenvolvidos separadamente e em vários casos, sem a intenção de que fossem ser integrados algum dia. Assim, esses sistemas eram muito diferentes quanto à frequência, modulação e outras características, o que impedia a sua interoperabilidade. Além dos militares, os órgãos de segurança, serviços públicos e de emergência também sofriam com a falta de integração de seus equipamentos.

No início da década de 1970 o Departamento de Defesa dos Estados Unidos começou a trabalhar em uma tecnologia para atender as necessidades de integração e comunicação da Força Aérea chamada *Integrated Communications, Navigation, Identification, and Avionics system (ICNIA)* [Camana, 1988]. Esse sistema era baseado em três elementos: um elemento de processamento de baixa latência de 25 MHz, um elemento de processamento de propósito geral e uma unidade de segurança reforçada. Esta tecnologia foi uma das primeiras a utilizar modem e controle programáveis baseados em DSP (*digital signal processor*) e foi uma das bases para a criação de Rádios definidos por *software* nos anos posteriores.

No final da década de 1980, o Departamento de Defesa se viu diante de uma importante questão: Como seria possível assegurar a comunicação com os aliados mantendo uma estrutura de suporte global, evitar a interceptação por parte dos inimigos, tirar vantagem da rápida evolução tecnológica e ao mesmo tempo controlar os gastos com o setor militar? Foi nesse contexto que em 1989 a Força Aérea iniciou um projeto chamado *Tactical Anti-Jam Programmable Signal Processor (TAJPSP)* com o objetivo de produzir um modem reprogramável modular. Esse projeto evoluiu para a criação de



um sistema de rádio programável chamado *Speakeasy*. O *Speakeasy* foi um programa criado para elaborar a nova geração de equipamentos de rádio militares [Lackey & Upmal, 1995].

O programa *Speakeasy* teve duas fases e os resultados foram apresentados na segunda metade da década de 1990. A primeira fase durou de 1992 a 1995 e constituiu uma prova de conceito, sua solução final mostrou a capacidade de desenvolvimento e aplicação de *software radio* no setor militar. No entanto, a solução não era um rádio que poderia ser produzido ainda, pois dentre outros problemas, as mudanças de contexto não aconteciam rápidas o suficiente para que fossem mantidas várias comunicações simultâneas. Já o *Speakeasy II* teve um sucesso maior, a solução foi apresentada em 1997 durante o *Army's Task Force-XXI Advanced Warfighting Experiment*. A nova solução foi tão bem recebida que o desenvolvimento foi interrompido e o equipamento entrou em produção sem que todas as funcionalidades estivessem prontas, operando apenas na faixa entre 4MHz e 400MHz [Cook & Bonser, 1999]. Inicialmente, as únicas formas de onda foram AM e FM para transmissão de voz, mas o programa mostrou ao final os benefícios do uso de uma arquitetura modular, aberta e programável na construção de sistemas de rádio completos [Cook & Bonser, 1999].

Ainda durante a primeira fase do *Speakeasy*, em 1992, Joseph Mitola publica um trabalho onde analisa as tecnologias existentes na época, apresenta a ideia de um *Software Radio* ideal, seu funcionamento, suas possibilidades e o que essa tecnologia idealizada precisaria para operar satisfatoriamente [Mitola, 1992]. Esse trabalho se torna uma referência importante para o desenvolvimento de SDR. No artigo, o autor apresenta também uma perspectiva de como a tecnologia seria no futuro e em alguns trechos descreve rádios que se reconfiguram automaticamente. Essa última ideia é trabalhada por Mitola e anos depois se torna tema da sua tese de doutorado em que propõe o conceito de “Rádios cognitivos” [Mitola, 2000].

Rádio cognitivo é um conceito mais amplo que SDR, pois utiliza todos os recursos disponíveis para fazer análise do contexto e se reconfigurar, se adequando ao ambiente. É um novo paradigma em comunicação sem fio, as redes que utilizarem rádios cognitivos poderão acessar o espectro eletromagnético dinamicamente, mudando de faixa conforme necessidade para melhorar sua utilização. Embora sofisticado, o conceito recebeu pouca atenção inicialmente, pois as demandas de processamento eram demasiadamente altas para a época e as possibilidades de implementação eram mínimas.

O conceito voltou a receber atenção com a publicação do relatório sobre uso do espectro de radio-frequência publicado pela FCC (Federal Communications Commission) em 2002 [Force, 2002]. O relatório mostra que aqueles que detêm a licença do uso de cada faixa de frequência não a utilizam todo o tempo e nem em todo o espaço geográ-

fico para o qual foi licenciada. A Comissão propõe como alternativa para melhorar essa utilização, que usuários que não possuem a licença de uso de uma determinada faixa de frequência possam usá-la quando o detentor da licença não a estivesse utilizando.

O termo *Software Defined Radio* só foi proposto em 1995 por Stephen M. Blust [Blust, 1995] e desde então passou a ser utilizado para designar o que se tornou a evolução dos *Software radios*. O *Ideal Software Radio* discutido por Mitola no artigo de 1992 e posteriormente chamado de *Mitola Radio* continuou sendo considerado como uma abstração útil e uma referência importante para o desenvolvimento da tecnologia SDR [Mitola et al., 2015].

A partir dos anos 2000 a tecnologia ganha popularidade, pois são produzidos equipamentos de baixo custo como os *frameworks* USRP [Ettus, 2016], SORA [Tan et al., 2011], SODA [Lin et al., 2006] e FLAVIA [Bianchi, 2013]. Popularizam-se também as plataformas de *software* como GNU Radio [GNU Radio, 2015], ASGARD [Tavares et al., 2012], IRIS [Tinnirello et al., 2012] e OSSIE [Gonzalez et al., 2009], que facilitam o trabalho de pesquisa e o desenvolvimento de aplicações.

## 2.2.2 Aplicações

Atualmente, o SDR é bastante utilizado tanto em aplicações comerciais como em pesquisas acadêmicas. O amadurecimento da tecnologia nos anos anteriores associado ao surgimento de *frameworks* de baixo custo e plataformas de *software* que apoiam o desenvolvimento facilitaram pesquisas fora dos grandes laboratórios e impulsionaram a produção de aplicações baseadas em SDR.

Em 2011, o *Wireless Innovation Forum* realizou um estudo para verificar o uso de SDR na indústria e naquele ano constatou que mais de 93% da estrutura móvel utilizava SDR de alguma forma [Forum, 2011]. Esse número fica ainda maior quando são analisados mercados como aplicações militares e segurança pública onde a interoperabilidade é um requisito forte. Nesse caso, a pesquisa constatou que quase todos os transceptores e estações base utilizados empregavam a tecnologia SDR.

Em telecomunicação, Rádio Definido por Software está sendo adotado por empresas como a *China Mobile*, através do uso de torres C-RAN [Chen & Duan, 2011]. Essa plataforma executa todo o processamento dos dados em servidores em nuvem e, de forma geral, promove redução de custos, redução de gasto com energia, aumento da eficiência de cobertura de espectro e flexibilidade para manutenção e evolução do serviço.

Existem também, empresas como *FlexRadio Systems* e *Tucson Amateur Packet Radio* que investem no mercado de rádios amadores utilizando SDR. Nesse caso,

os usuários estão interessados em desenvolver suas próprias aplicações com funções diversas como navegação marítima, recepção de sinais de rádio FM, AM, TV e até radiotelescópios.

Uma aplicação importante em SDR que tem sido foco de diversas pesquisas é o Rádio cognitivo. A implementação da alternativa apontada no relatório da FCC exige dos rádios uma flexibilidade e capacidade de reconfiguração que se encaixa perfeitamente com a ideia de rádio cognitivo. É necessário um equipamento que seja capaz de verificar que faixas de frequência estão disponíveis no momento e que se comunique nessas faixas. Também é necessário se comunicar com diversas tecnologias diferentes. Dadas as necessidades, SDR é a tecnologia ideal para a implementação desses rádios, pois possui toda a flexibilidade necessária para atender às necessidades apresentadas.

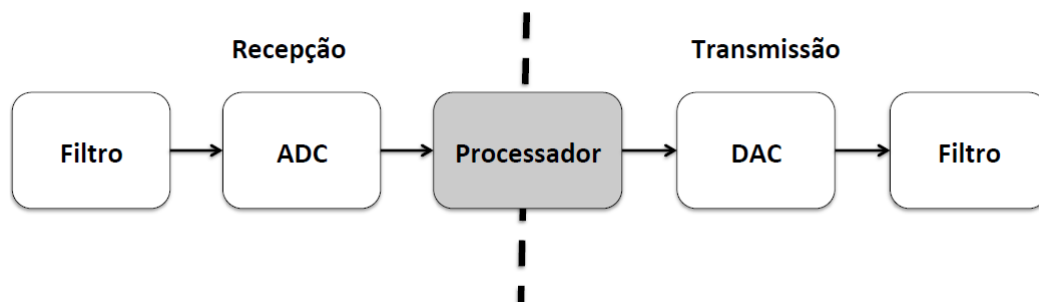
No meio acadêmico, o uso de SDR tem ganhado grande importância. De acordo com Silva et al. [2015], uma das vantagens de *Software Defined Radio* é justamente a redução do custo para pesquisas experimentais da camada física e de enlace, possibilitando a universidades realizar experimentos que antes só eram possíveis a grandes laboratórios. Dessa forma, um infinidade de pesquisas utilizando a tecnologia têm sido desenvolvidas em todo o mundo em temas como *Network coding* [Firooz et al., 2013], [Vieira et al., 2013], separação de canais de controle e dados [Cidon et al., 2012], sincronização em sistemas OFDM [Yildirim, 2015], flexibilização da sub-camada MAC [Hu et al., 2011], integração com SDN em telecomunicação [Cho et al., 2014] e muitos outros.

No nosso trabalho utilizamos a tecnologia SDR devido à flexibilidade que ela nos proporciona. Com SDR é possível alterar as camadas inferiores dos protocolos de comunicação da rede. No nosso caso, alteramos a sub-camada de controle de acesso ao meio. Nos rádios comuns isso não é possível pois neles esses padrões estão geralmente implementados em *hardware* ou *firmware* e são inacessíveis.

### 2.2.3 Funcionamento do SDR

O funcionamento da tecnologia SDR segue o princípio de que parte das funções do transceptor é implementada em software. Não é possível implementar em *software* as funções mais básicas como filtros e conversões de sinais entre modo analógico e digital, por isso, essas são implementadas em hardware. Todas as outras funções do transceptor podem ser realizadas em unidades programáveis como CPUs de propósito geral, processadores de sinais (DSP) e FPGAs [Nagurney, 2009]. A Figura 2.3 mostra a arquitetura geral de um sistema SDR. Aqui, os blocos em cor branca correspondem a elementos implementados em *hardware*, enquanto o bloco em cor cinza representa a

implementação em *software*.



**Figura 2.3.** Arquitetura básica SDR. Silva et al. [2015]

Na implementação de SDR, a arquitetura básica apresentada na Figura 2.3 pode se especificar em duas abordagens diferentes conhecidas como *SDR Modal* e *SDR Reconfigurável*.

No primeiro caso, o equipamento possui vários *chips* com implementações de diferentes rádios. Assim, cada chip é utilizado no momento em que o rádio implementado nele for necessário. Essa abordagem é conveniente quando se sabe de antemão que implementações serão usadas durante toda a vida do equipamento, como por exemplo, celulares que possuem conectividade 3G e 4G. No entanto, percebe-se que *SDR Modal* possui uma flexibilidade limitada, sem muita abertura para upgrades e correção de *bugs*. *SDR Reconfigurável*, por outro lado, possui ilimitadas possibilidades de implementação, já que o componente de processamento pode ser configurado para qualquer aplicação. Essa abordagem, representada na Figura 2.2, se popularizou mais que a anterior, tendo diversas implementações como USRP [Ettus, 2016], SORA [Tan et al., 2011], SODA [Lin et al., 2006] e FLAVIA [Bianchi, 2013].

## 2.2.4 Plataformas e desenvolvimento

A partir dos anos 2000, diversos equipamentos de SDR de custo acessível se popularizaram, possibilitando o desenvolvimento da pesquisa em universidades. Além da implementação física da arquitetura, várias plataformas de software também foram criadas para apoiar o desenvolvimento.

Existem diversas implementações de SDR, algumas delas são:

- **USRP** (*Universal Software Radio Peripheral*): Trata-se de um *framework* que possui uma estrutura completa para o processamento de sinais. O sistema tem como características alta flexibilidade e custo-benefício. Baseado em FPGA, possui vários modelos que se popularizaram como o Ettus B100, Ettus N210 e Ettus

B210 [Ettus Research, 2016]. Este último possui uma abrangência contínua de espectro de 70 MHz a 6 GHz, trabalha com USB 3.0 e permite a integração com o software livre de desenvolvimento *GNU Radio*. É a implementação utilizada neste trabalho, mais detalhes sobre a sua arquitetura e funcionamento podem ser vistos em [Silva et al., 2015].

- **Sora** (*Microsoft Research Software Radio*): Sora [Tan et al., 2011] é a implementação de SDR desenvolvida pela *Microsoft Research (MSR)* Asia em Beijng. Sora combina o desempenho de SDR baseado em hardware com a flexibilidade de um processador GPP (*Generic Purpose Processor*). O equipamento possui apenas um decodificador de banda base, todo restante do processamento é feito por uma CPU x86.
- **SODA** (*Signal-processing On-Demand Architecture*): SODA é uma implementação SDR de baixo consumo [Lin et al., 2006]. Possui um processador de propósito específico, voltado para a realização em *software* das principais funções envolvidas com a camada física (PHY) e controle acesso ao meio (MAC). A plataforma foi utilizada na implementação de dois padrões importantes de redes sem fio, o W-CDMA e 802.11a, apresentando desempenho suficiente para a boa execução dos protocolos.
- **FLAVIA** (*FLexible Architecture for Virtualizable future wireless Internet Access*): FLAVIA [Bianchi, 2013] é um tipo de *soft* MAC, uma plataforma projetada para implementação da camada MAC em *software*. A plataforma executa aplicações SDR em transceptores comerciais de baixo custo [Tinnirello et al., 2012] facilitando a pesquisa e o desenvolvimento de protocolos MAC. Como a camada PHY é fixa, essa implementação tem menos flexibilidade, estando limitada à frequência de 2.4GHz e a uma forma específica de modulação e tratamento do sinal.

As ferramentas de *software* criadas para apoiar o desenvolvimento também foram essenciais para a popularização da tecnologia SDR. Algumas das principais plataformas utilizadas são:

- **GNU Radio**: GNU Radio é um kit de ferramentas livre e de código aberto para desenvolvimento de software que fornece blocos de processamento de sinal para implementar *software radios*. Possui Ambientes de Desenvolvimento Integrado, tais como o GNU Radio Companion (GRC), que permite o desenvolvimento de

aplicações usando uma interface gráfica de usuário. É muito utilizado em ambientes acadêmicos para apoiar a pesquisa em comunicações sem fio. GNU Radio é a plataforma utilizada neste trabalho, mais detalhes sobre ela podem ser vistos em [Silva et al., 2015].

- **IRIS** (*Implementing Radio in Software*): É uma arquitetura de software para rádios cognitivos desenvolvida para processadores de propósito geral. Assim como GNU Radio, esta plataforma é orientada a componentes. Compatível com Windows e Linux, permite a criação de redes altamente reconfiguráveis. IRIS possui um foco grande em flexibilidade, e sua arquitetura apresenta suporte para reconfiguração do rádio em tempo real. Além disso, possibilita o desenvolvimento não só para camada física, mas para todas as camadas da rede.
- **OSSIE** (*Open-Source SCA Implementation - Embedded*): SCA (*Software Communication Architecture*) é uma arquitetura aberta desenvolvida pelo Departamento de Defesa norte americano com o objetivo de dar suporte ao desenvolvimento de SDRs [Gonzalez et al., 2009]. OSSIE [Snyder et al., 2011] é uma implementação dessa arquitetura, um projeto aberto mantido pela Virginia Tech - EUA e patrocinado pela *U.S. National Science Foundation*. Ele foi criado para ser usado em prototipagem rápida e provas de conceito. OSSIE permite a integração com diversos outros ambientes de desenvolvimento, como o Eclipse e o próprio GNU Radio. A plataforma é muito utilizada como ferramenta educacional e de pesquisa.
- **ASGARD** (*Application-oriented Software on General purpose processors for Advanced Radio Development*): É um *framework* desenvolvido pela Aalborg University, da Dinamarca, para a implementação de aplicações SDR [Tavares et al., 2012]. Diferentemente das outras plataformas apresentadas até agora, o ASGARD não dispõe de um fluxo gráfico dos dados para edição ou visualização. A arquitetura do software é baseada em *Application Programmable Interfaces* (APIs), utilizando-se de componentes, módulos, aplicativos e objetos de comunicação.

## 2.3 Rádios cognitivos

O conceito de *Rádio cognitivo* foi originalmente definido por Joseph Mitola. Segundo esse autor, Rádio cognitivo é um novo paradigma em comunicação sem fio no qual os rádios utilizam todas as informações de contexto disponíveis para se

autor-reconfigurar, provendo o melhor desempenho para o usuário [Mitola, 1999, 2000]. A *Federal Communications Commission* define formalmente Rádios cognitivos da seguinte forma:

*“Um ‘Rádio cognitivo’ é um rádio que pode mudar seus parâmetros de transmissão baseando-se na interação com o ambiente em que está operando.”*<sup>1</sup>

A partir dessa definição, duas características principais podem ser extraídas: *Capacidade cognitiva* e *Habilidade de reconfiguração* [Akyildiz et al., 2006]. A Capacidade cognitiva consiste na capacidade do rádio de perceber a informações referentes ao seu ambiente de contexto. A habilidade de reconfiguração consiste em poder se reconfigurar de acordo com as informações do ambiente.

As faixas de frequência do espectro eletromagnético, passíveis de utilização por rádios cognitivos, podem ser divididas em dois grupos: *Faixas licenciadas* e *Faixas não licenciadas*, também chamadas *Faixas ISM (Industrial Scientific and Medical)*. Como requisitos, a tecnologia de rádios cognitivos deve permitir ao utilizador detectar quais faixas de frequência estão disponíveis, selecionar o melhor canal a ser utilizado e desalojar o canal no caso da presença de um usuário licenciado.

Para atender a esses requisitos, uma rede baseada em Rádios cognitivos deve conter algumas funções específicas. De acordo com Akyildiz et al., essas funcionalidades são quatro:

- **Sensoriamento do espectro:** Essa função consiste na detecção das porções do espectro que não estão sendo utilizadas e no compartilhamento dessas faixas sem causar interferência nos outros usuários.
- **Decisão de espectro:** Consiste em apontar a melhor faixa do espectro disponível que atende aos requisitos de comunicação do usuário.
- **Mobilidade do espectro:** Essa função coordena a troca do canal, executando a transição e assegurando que seja mantida uma continuidade de comunicação durante a mudança do sistema para outra faixa do espectro.
- **Compartilhamento do espectro:** Proporciona justiça no escalonamento de uso das faixas livres para os diversos usuários da rede.

O *Sensoriamento do espectro* e a *Decisão de espectro* são processos que podem acontecer centralizadamente ou de forma distribuída [Akyildiz et al., 2006]. No caso

---

<sup>1</sup>FCC [2003] Tradução livre

do Sensoriamento, quando centralizado, apenas uma estação é responsável por colher as informações do ambiente que vão apoiar o processo de decisão pela melhor faixa de frequência. Quando o sensoriamento é distribuído, diversas estações desempenham essa função, isso pode incluir usuários não licenciados. A fase de decisão possui características similares ao caso do sensoriamento, mas está mais concentrada na subcamada MAC e possui uma maior diversidade de possibilidades de implementação. Embora as abordagens colaborativas proporcionem uma maior acurácia nos resultados, acrescentam também um *overhead* de processamento às estações e um tráfego de dados adicional à rede.



# Capítulo 3

## Trabalhos relacionados

Desde a criação de redes sem fio, diversos trabalhos foram publicados com a preocupação de melhorar o controle de acesso ao meio. Com a popularização dessas redes e a diversificação dos protocolos, várias pesquisas foram desenvolvidas no sentido de aumentar a flexibilidade da subcamada MAC. Nesse capítulo revisamos alguns desses trabalhos, mostrando a abordagem utilizada, os avanços e deficiências de cada um.

Com base na abordagem com que cada trabalho procura prover flexibilidade à subcamada MAC, dividimos os trabalhos em três grupos. Primeiro, aqueles que contribuem para a flexibilização do controle de acesso ao meio sem se preocupar com algum protocolo específico, focando em aspectos mais gerais como interação com outras camadas e possibilidade de reprogramação de suas funções. Segundo, aqueles que tentam melhorar de alguma forma protocolos específicos dessa subcamada, adicionando a eles um certo grau de flexibilidade. Terceiro, aqueles que propõem abordagens híbridas, ou seja, que utilizam soluções já existentes combinadas na tentativa de obter os aspectos positivos de cada uma.

### 3.1 Plataformas para MAC flexível

Consideramos aqui *Plataforma para MAC flexível* como um conjunto de funcionalidades que adiciona flexibilidade ao controle de acesso ao meio de uma forma mais geral. Isso significa que essas funcionalidades fornecem recursos para a construção de protocolos MAC ou flexibilizam a atuação de protocolos que já existem, mas sem focar em nenhum deles de forma específica. Além disso, essas plataformas podem atuar diretamente na subcamada MAC ou apenas na sua interface com as outras camadas. Os trabalhos descritos nessa seção possuem essas características.

Em 2005, Rao & Stoica propuseram uma abordagem nova para flexibilizar o controle de acesso ao meio. No seu trabalho, eles apresentam a *Overlay MAC Layer* (OML), que consiste na adição de uma nova camada sobre a subcamada MAC. Esse novo elemento, em software, permite testar ou implementar funcionalidades adicionais à subcamada MAC sem alterar o protocolo original, no caso, o 802.11. Essa abordagem adiciona flexibilidade para adaptar o controle de acesso ao meio a requisitos de roteamento e aplicação. A plataforma poderia receber informações das camadas superiores sobre o tipo de tráfego (ftp, voz, web) e usar essa informação para organizar de forma mais eficiente suas transmissões. Seria possível, por exemplo experimentar novos algoritmos de escalonamento e gerência de largura de banda, avaliar seus benefícios para as aplicações existentes antes de implementá-los diretamente na subcamada MAC. Em outro exemplo, como a implementação da OML fornece diferenciação de serviços baseada na granularidade do fluxo ou dos nós, em *rooftop networks* seria possível modificar a OML para tirar vantagem do aspecto estacionário da qualidade dos links e dos padrões de interferência dessas redes. No entanto, embora haja toda essa flexibilidade, ela está limitada à interface de comunicação entre a camada MAC e a camada de Rede. Não é possível, por exemplo, realizar verificação do meio, pois esses detalhes vindos da camada Física geralmente não são disponíveis para os níveis superiores.

No mesmo ano, surgiu uma plataforma para desenvolvimento de protocolos MAC flexíveis chamada SoftMAC [Neufeld et al., 2005]. Essa plataforma foi desenvolvida sobrepondo a implementação do 802.11 disponível em uma família de placas de redes comerciais que usavam *chipset* Atheros AR5212. O framework possuía uma camada Física fixa, mas incorporava uma base de software que permitia a implementação de protocolos MAC com relativa flexibilidade e baixo custo. A motivação para o seu desenvolvimento foi atender à demanda por implementações reais de protótipos baseados em SDR, que já era objeto de crescente interesse.

FLAVIA é também um projeto desenvolvido para a programação de protocolos MAC [Tinnirello et al., 2012]. Essa plataforma emprega o que os autores chamam de *MAC processor architecture*, um dispositivo programável que fornece um conjunto de comandos para o desenvolvimento de protocolos MAC. O *firmware* implementa uma máquina de estados baseada em eventos comuns nos protocolos MAC como início e fim de intervalos, ocorrência de colisão, recepção de quadro de RTS, CTS, Dados, ACK, etc. Os usuários podem utilizar esses eventos pré-definidos para criar novos protocolos para redes sem fio. Apesar da flexibilidade fornecida pela plataforma, ela está limitada à frequência de 2.4 GHz e não é possível trabalhar com nenhum evento além daqueles estabelecidos previamente.

A plataforma MultiMAC talvez seja, em termos de arquitetura, a que mais se

aproxima do trabalho proposto nessa dissertação [Doerr et al., 2005]. Nessa plataforma, os autores propõem uma arquitetura que pode abrigar diversos protocolos onde cada um pode ser utilizado na situação mais adequada. A troca entre protocolos é baseada em regras, que avaliam quando um dado protocolo deve entrar em operação. Essas regras podem ser alteradas em nível de usuário, mas não está claro como é o mecanismo que utiliza essas regras para selecionar o melhor protocolo. Novos protocolos também podem ser acrescentados, mas os procedimentos para a realização dessa inclusão também não são detalhados. Os experimentos para avaliação da plataforma empregam dois protocolos, sendo um com detecção de erros (CRC) e outro com correção de erros, utilizando apenas duas estações durante os experimentos. Além da avaliação reduzida, o ponto fraco do trabalho é a falta de um mecanismo de sincronização entre as estações, o que é importante para redes de larga escala, pois alguns protocolos podem ser conflitantes entre si, gerando problemas de desempenho se as estações não entram em consenso sobre o melhor protocolo a ser utilizado.

Dhar et al. [2006] apresentam suas experiências com a integração das camadas MAC e PHY utilizando SDR. Eles analisam dois *frameworks*, GNU Radio, que apresentava na época maior suporte para a camada PHY e Click, um *framework* para desenvolvimento de protocolos de comunicação. Os autores descrevem o processo de integração que fizeram entre GNU Radio e Click para gerar uma plataforma capaz de suportar o desenvolvimento de protocolos MAC e PHY. No artigo, eles apresentam importantes conhecimentos adquiridos durante o processo, como características dos protocolos MAC que não podem ser facilmente implementadas em software.

Nesse sentido, Nychis et al. [2009] fornecem contribuição fundamental para a implementação de protocolos MAC flexíveis em SDR. Os autores mostram que o fato de SDR ser flexível não implica necessariamente que a implementação de um protocolo MAC nessa tecnologia será flexível. Identificam um conjunto de funções MAC importantes que devem ser implementadas mais próximas do rádio por razões de eficiência. Definem também uma API que permite ao *host* controlar essas funções, provendo a flexibilidade necessária para a implementação desses protocolos.

## 3.2 Otimização de protocolos específicos

Nesta seção apresentamos alguns trabalhos que buscam melhorar isoladamente protocolos específicos da subcamada MAC como CSMA e TDMA. Os melhoramentos desenvolvidos nesses trabalhos aproveitam a flexibilidade de conceitos como SDR e SDN (*Software Defined Network*) para apresentar soluções que não podem ser desenvolvidas

nas arquiteturas mais rígidas. A simples implementação de protocolos em plataformas programáveis já traz ganhos interessantes como a possibilidade de atualizações e correções de defeitos sem substituição do hardware. Além disso, a flexibilização traz grandes possibilidades de melhoramentos, pois permitem novas abordagens como mudança dinâmica de parâmetros, adaptação do comportamento dos protocolos ao contexto, modularização, reuso de módulos, controle de acesso a nível de fluxo, etc.

Em 2012, Puschmann et al. implementam com sucesso um protocolo baseado em CSMA, bastante similar ao esquema definido no padrão 802.11 DCF. A implementação aproveita a flexibilidade do SDR, bem como a capacidade de modularização provida por Iris, o software utilizado no desenvolvimento. A contribuição deste trabalho está em prover flexibilidade e modularização na implementação deste protocolo específico. As funções mais importantes foram implementadas em módulos separados com interfaces definidas de interação entre si, permitindo a utilização de suas partes em outras aplicações ou pesquisas. Apesar da flexibilidade da implementação, em termos gerais, o protocolo mantém os mesmos problemas relacionados à contenção em redes com grandes volumes de tráfego. Existe ainda o natural *overhead* de atrasos decorrentes da implementação em SDR. Posteriormente, os autores apresentaram novo trabalho onde utilizaram esta mesma implementação do CSMA para propor melhorias no protocolo [Puschmann et al., 2013]. Nesse caso, mostraram estratégias que reduzem o parâmetro de *slot time*, mitigando os efeitos negativos da pouca capacidade de processamento de sinais dos computadores de propósito geral.

O trabalho de Yang et al. tem foco no protocolo TDMA. Eles não modificam o protocolo em si, mas propõem uma plataforma chamada OpenTDMF. Essa plataforma é inspirada na ideia da arquitetura de SDN, onde o plano de controle é separado do plano de dados. Na OpenTDMF, existe um controlador centralizado, implementado em software, que gerencia o acesso ao canal de todos os dispositivos na rede no nível de fluxo. Esse gerenciamento é feito através de uma interface OpenTDMF presente nos pontos de acesso. O controlador basicamente associa um conjunto de *slots* de tempo e um nível de prioridade a cada fluxo. De acordo com o valor de prioridade, cada ponto de acesso determina ordens de transmissão em nível de pacote de seus fluxos locais. A implementação utilizou pontos de acesso TP-Link 4900, modificando o *firmware* e incluindo o protocolo TDMA. A abordagem consegue melhorar a solução de problemas de terminal exposto e escondido, aumentar o nível de justiça e prover maior QoS. Apesar desses ganhos, a abordagem focada em divisão de tempo ainda traz os mesmos problemas quando o nível de contenção da rede é baixo, ou seja, existe o *overhead* das mensagens de controle que deterioram o desempenho da rede quando há poucos dispositivos tentando transmitir.

Di Francesco et al. [2015], baseando-se nas ideias gerais sobre eficiência de protocolos MAC em SDR discutidas em [Nychis et al., 2009], desenvolvem e implementam uma arquitetura para CSMA baseada na separação de funções utilizando o framework USRP com a plataforma Iris. Nessa arquitetura, algumas funções mais sensíveis a atrasos, como gerenciamento de *backoff* e *carrier sensing* são implementadas na FPGA, enquanto as outras continuam implementadas em software. São construídas duas versões do CSMA e através de experimentos reais combinadas a simulações produzidas com OMNeT++, os autores conseguem mostrar o ganho de eficiência decorrente dessa abordagem.

### 3.3 Flexibilização com abordagem híbrida

Nesta seção apresentamos trabalhos que utilizam uma abordagem híbrida na subcamada MAC. Consideramos *abordagem híbrida* aquela que combina soluções já existentes para gerar uma nova solução, aproveitando o melhor de cada uma das originais. Utilizando essa abordagem, há trabalhos que apresentam protocolos híbridos, que são aqueles que utilizam as ideias principais de mais de um protocolo já existente para produzir um terceiro protocolo, e as plataformas híbridas, que são plataformas que utilizam mais de um protocolo, mas cada um operando de forma distinta.

Sharp et al. [1995] apresentam uma ideia que posteriormente seria abordada em diversos trabalhos para tentar flexibilizar o controle de acesso ao meio. Os autores exploram o fato de que protocolos baseados em contenção e protocolos livres de contenção têm comportamento oposto em termos de eficiência em cenários de competição pelo acesso ao meio. Eles propõem então um protocolo híbrido, que alterna entre CSMA e TDMA, tentando aproveitar as vantagens de cada um nos momentos oportunos. A estrutura principal de comunicação do protocolo é baseada no TDMA, ou seja, há uma divisão de tempo em *slots* onde cada *slot* é alocado para uma estação. Essa alocação é feita globalmente a partir de uma sincronização e da informação inicial do número máximo de estações que estarão na rede. A partir de então, quando um *slot* não é utilizado pela estação alocada, ele pode ser ocupado utilizando o CSMA. Isso faz com que quando o canal esteja sendo menos utilizado, o protocolo CSMA se sobressaia, enquanto o TDMA tem maior uso quando o canal está congestionado. O trabalho mostrou as vantagens dessa abordagem e abriu caminho para uma série de protocolos híbridos baseados nessa ideia. Apesar da contribuição, a solução foi implementada sobre um Sistema para demonstração de Rádio Pacotes (PRDS), um equipamento de difícil acesso fora de grandes laboratórios.

Em 2002, Myers et al. apresentam AGENT, um protocolo híbrido para redes *ad hoc* que tem um componente de contenção e um componente de alocação. A abordagem usada aqui utiliza o protocolo baseado em contenção (CSMA) dentro do protocolo livre de contenção (TDMA), ou seja, há um CSMA em operação dentro dos *slots* de tempo do TDMA. Dessa forma, nos cenários onde a carga na rede é baixa, o protocolo tem eficiência similar ao CSMA, enquanto em cenários de maior congestionamento, se comporta similarmente ao TDMA. A solução suporta transmissão de pacotes *unicast*, *multicast* e *broadcast*, e incorpora acesso priorizado a canais, o que garante a cooperação entre o componente de alocação e o de contenção. Apesar da flexibilidade, o protocolo se limita ao uso de uma implementação específica de TDMA e outra de CSMA, não deixando espaço para a inclusão de outros protocolos. A avaliação foi realizada com a simulação de uma rede *ad hoc*, mostrou a efetividade da ideia e as melhorias em relação aos protocolos similares anteriores.

A plataforma MultiMAC também possui uma abordagem híbrida. Ela pode abrigar diversos protocolos, onde cada um é utilizado na situação mais adequada [Doerr et al., 2005]. O *framework* permite não só o uso de CSMA em situações de baixa contenção e TDMA em situações de alta contenção, mas prevê a inclusão de novos protocolos nesse conjunto e a inclusão de novas regras para avaliar qual deles deve entrar em operação. Apesar das ideias propostas, o trabalho não deixa claro como funcionaria a plataforma considerando toda a rede. Ele apresenta apenas o ponto de vista de uma estação, ou seja, como cada estação específica seleciona o melhor protocolo a ser utilizado. Como mencionado anteriormente, além de deixar questões relacionadas a sincronização em aberto, a arquitetura foi avaliada de forma muito restrita.

Z-MAC é também um protocolo híbrido que combina as vantagens de um protocolo baseado em contenção e outro livre de contenção Rhee et al. [2008]. Nesse trabalho, inicialmente é utilizado um algoritmo de escalonamento para a alocação dos *slots*. Uma vez alocados, os nós reutilizam esse *slots* em todos os períodos (*frames*). O ponto principal desse protocolo é que cada *slot* possui um *dono*, que tem prioridade sobre o seu uso, mas quando esse nó não o utiliza, o *slot* fica disponível para ser usado por outros *não donos*. Este esquema de prioridades tem um efeito implícito de alternância entre CSMA e TDMA dependendo do nível de contenção.

Hu et al. [2011] apresentam um protocolo híbrido para a subcamada MAC que chamam de LA-MAC. LA-MAC pode operar em dois modos: CSMA ou HYBRID. Quando no primeiro modo, utiliza uma implementação similar ao padrão 802.11. No modo híbrido utiliza uma implementação do TDMA puro. No modo híbrido, cada nó pode estar no estado de Alta contenção ou Baixa contenção. A métrica utilizada para essa classificação é a notificação de que  $n$  pacotes foram perdidos, e essa noti-

ficção é feita por toda a vizinhança que está a dois saltos de distância. O estado de alta contenção dura um período pré-determinado, depois do qual, sem notificações adicionais, o nó volta para o estado de baixa contenção. Quando em baixa contenção, todos os nós competem igualmente pelos *slots*, ou seja, todos operam em CSMA. Já no estado de alta contenção, os nós competem pelos *slots* com base em suas prioridades. Essas prioridades são baseadas no tamanho da Janela de contenção. Novamente, apesar da flexibilidade provida pelo caráter híbrido da abordagem, assim como no caso do Z-MAC, ela não deixa a possibilidade de uso de outros protocolos, apenas uma implementação específica é possível.

### 3.4 Discussão

Diferentemente dos trabalhos da seção 3.2, FS-MAC não busca a flexibilização da subcamada MAC focando em algum protocolo específico. Pelo contrário, uma das principais ideias da plataforma é permitir que qualquer protocolo possa fazer parte do seu conjunto. O FS-MAC poderia por exemplo, contribuir com a otimização fornecendo informações de contexto para serem utilizadas nesse processo.

A nossa abordagem possui certa semelhança com os trabalhos apresentados na seção 3.3, pois a ideia central desses trabalhos é a utilização de mais de um protocolo aproveitando as vantagens de cada um em determinados cenários. A principal diferença nesse caso é que esses trabalhos tratam as suas soluções desenvolvidas como um protocolo híbrido. Essa abordagem flexibiliza a subcamada MAC quanto ao uso de mais de um protocolo, mas limita esse uso ao conjunto utilizado originalmente. Esses trabalhos, com exceção do MultiMAC, possuem apenas dois protocolos e não fornecem mecanismos para a fácil inclusão de outros e nem facilitam um rearranjo das regras que determinam qual deles utilizar. FS-MAC não é um protocolo, mas uma plataforma que abriga um conjunto deles, podendo inclusive ter como elementos desse conjunto, protocolos híbridos como os apresentados acima.

A nossa proposta possui também uma relação com os trabalhos mostrados na seção 3.1. Assim como esses trabalhos, FS-MAC se preocupa com o controle de acesso ao meio de forma mais geral, fornecendo ao usuário maior poder de configuração e reprogramação de elementos que permitem adaptar o comportamento da subcamada ao contexto da rede. Esse contexto contempla aspectos como nível de contenção, justiça na transmissão, taxa de entrega de pacotes, latência e outros parâmetros que se deseja melhorar.

De todos os trabalhos citados nessa seção, o que guarda maior semelhança com o

nosso é o MultiMAC. Esse trabalho, assim como o FS-MAC, possibilita o uso de mais de um protocolo, permitindo a inclusão de novos elementos nesse conjunto e também a reorganização das regras para priorizar aspectos diferentes da rede. A maior diferença entre as duas plataformas é que o nosso trabalho esclarece pontos fundamentais que não ficaram claros no MultiMAC como a atuação da plataforma quanto à sincronia de uso dos protocolos. Além disso, no nosso caso há um protótipo mais robusto e com testes mais completos que avaliam a solução.

A Tabela 3.1 apresenta uma comparação entre os trabalhos apresentados, mostrando as principais características de cada um quanto à flexibilização da subcamada MAC.

**Tabela 3.1.** Comparação entre os principais trabalhos

<i>Característica</i>	<i>OML</i>	<i>softMAC</i>	<i>FLAVIA</i>	<i>Enabling MAC SDR</i>	<i>Hybrid TDMA CSMA</i>	<i>AGENT</i>	<i>Multi MAC</i>	<i>Z-MAC</i>	<i>LA- MAC</i>	<i>FS-MAC</i>
Permite mais de um protocolo				*	*	*	*	*	*	*
Fácil inclusão de novos protocolos							*			*
Decisão baseada em regras	*				*	*	*	*	*	*
Fácil adaptação de regras	*	*					*			*
Implementada em plataforma flexível	*	*	*		*	*	*	*	*	*
Protótipo completo implementado	*	*	*		*	*		*	*	*
Avaliação de pontos principais	*	*	*	*	*	*		*	*	*
Base para criação de protocolos MAC		*	*	*						



## Capítulo 4

# FS-MAC: *Flexible System for Medium Access Control*

Neste capítulo apresentamos *Flexible System for Medium Access Control* (FS-MAC), uma arquitetura que promove a flexibilização da subcamada MAC de redes sem fio. Esta arquitetura foi desenvolvida para permitir o uso de mais de um protocolo no controle de acesso ao meio, utilizando cada um na situação em que é mais eficiente. FS-MAC permite ainda que novos protocolos sejam adicionados ao conjunto e configurados sem que seja preciso refazer toda a subcamada.

FS-MAC trabalha com um conjunto de protocolos e possui três módulos chamados *Sensoriamento*, *Decisão* e *Troca*, com os quais realiza todos os processos necessários para atingir seu objetivo. A organização e funcionamento desses módulos foram baseados nos conceitos de Rádios cognitivos devido à grande similaridade entre os problemas abordados nos dois casos. Utilizando a tecnologia SDR e a plataforma GNU Radio implementamos um protótipo com o qual avaliamos a arquitetura em uma situação de troca entre um protocolo baseado em contenção e outro livre de contenção.

Na Seção 4.1 apresentamos os aspectos gerais da arquitetura, os conceitos envolvidos na sua construção e o seu funcionamento. Descrevemos a relação entre os problemas abordados por FS-MAC e Rádios cognitivos e como essa similaridade influenciou a construção da solução proposta. Em seguida, detalhamos os módulos que compõem o seu funcionamento e apresentamos a flexibilidade resultante da sua elaboração, com suas vantagens e limitações. Na Seção 4.2 descrevemos os detalhes da implementação de um protótipo construído para avaliar a arquitetura FS-MAC.

## 4.1 Arquitetura

Os requisitos e funções sobre rádios cognitivos apresentados na Seção 2.3 possuem uma grande similaridade com o nosso trabalho. Se pensarmos nas faixas do espectro eletromagnético como um conjunto de recursos, as funções de rádio cognitivo podem ser generalizadas nesse contexto, assumindo as descrições respectivas: *Sensoriamento de recursos*, *Decisão de recursos*, *Mobilidade de recursos* e *Compartilhamento de recursos*. Se considerarmos então, uma camada MAC híbrida conforme proposto pelo nosso trabalho, podemos ver essa camada como um conjunto de recursos no qual cada protocolo seria um elemento desse conjunto. Assim, tanto rádios cognitivos quanto a arquitetura da subcamada MAC deste trabalho abordam um problema de troca de configuração distribuída. No caso dos rádios cognitivos, o elemento a ser trocado é a faixa de frequência e no nosso caso, o protocolo MAC.

Diferente das arquiteturas anteriores para a subcamada MAC, FS-MAC possui alto grau de flexibilidade. Além de promover o aproveitamento do melhor de cada protocolo, ela facilita a adição e configuração de diversos elementos envolvidos nesse processo. A arquitetura possui três módulos e um conjunto de protocolos conforme a representação na Figura 4.1. O módulo de **Sensoriamento** é responsável por reunir dados sobre o contexto da rede e repassá-los ao módulo de **Decisão**. O objetivo do módulo de Decisão é analisar esses dados, determinar qual protocolo se adapta melhor às atuais condições da rede e informar essa decisão ao módulo de **Troca**. Já o módulo de Troca utiliza a informação recebida e executa todos os procedimentos para realizar a substituição do protocolo em operação quando necessário. Nas seções posteriores descreveremos detalhadamente a estrutura e funcionamento de cada um desses módulos.

A arquitetura requer o papel de um Coordenador, que será exercido por um dos nós na rede. A presença de um nó no papel de Coordenador tem importância fundamental para manter a rede operando com o mesmo protocolo, pois é ele quem irá conduzir a troca de protocolos em toda a rede quando for necessário e minimizará o tempo em que estações Comuns estejam atuando com protocolo diferente. A definição de qual nó deve ser o coordenador é feita previamente, ou seja, não há nenhum mecanismo automático que realize esta escolha. No entanto, qualquer nó pode atuar nesse papel, guardando apenas uma única restrição relacionada à sua localização. Como é o Coordenador quem cuida para que o protocolo correto esteja de fato em operação a cada momento, e essa gerência exige comunicação constante com todos os nós da rede, esse nó precisa necessariamente estar ao alcance de todos os outros.

Esses aspectos fazem com que o FS-MAC possua dois modos de operação, um

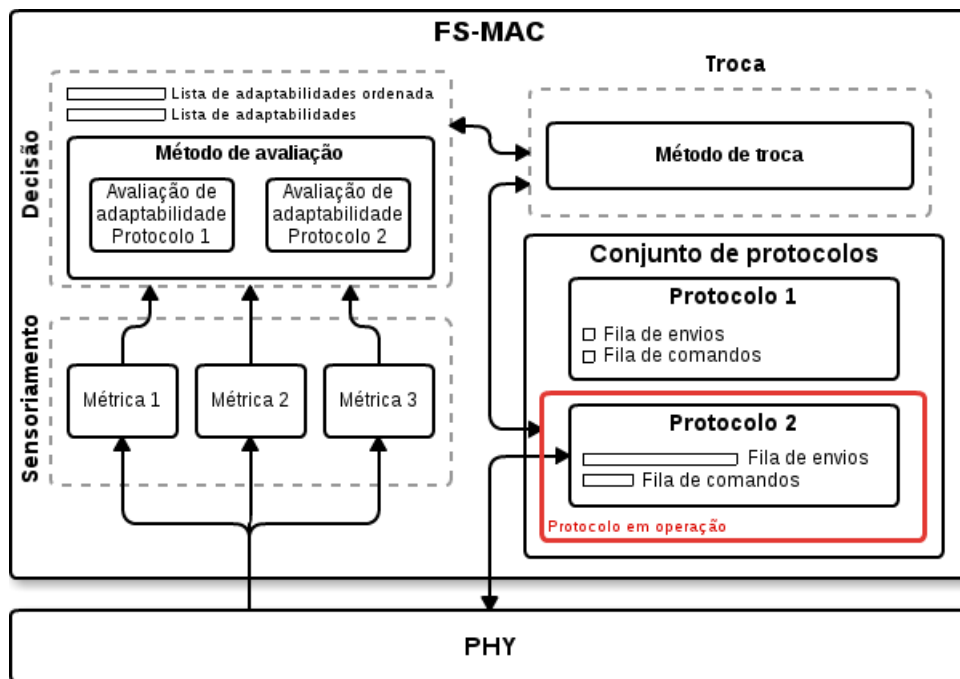


Figura 4.1. Arquitetura FS-MAC

que irá atuar no nó Coordenador e outro que irá funcionar nos nós Comuns. No modo Coordenador, todas as unidades de sensoriamento estarão ativadas, o módulo de Decisão também estará ativo e o módulo de Troca, além de substituir um protocolo por outro quando necessário, é responsável por comunicar a necessidade de troca aos outros nós da rede. Já no modo de operação Comum, as únicas unidades de sensoriamento ativas serão aquelas que reúnem dados de forma distribuída, o módulo de Decisão estará inativo e o módulo de Troca apenas executa a substituição do protocolo quando recebe o comunicado do nó Coordenador.

O FS-MAC suporta dois tipos de sensoriamento, centralizado e distribuído. Quando o sensoriamento é centralizado, o nó Coordenador é responsável por coletar os dados da métrica em questão, como o número de nós se comunicando ou o nível de ruído, por exemplo. Esses dados são comunicados diretamente ao módulo de Decisão do próprio Coordenador. No caso do sensoriamento distribuído, cada nó coleta os dados separadamente e periodicamente os envia para o nó Coordenador.

Além disso, conforme ilustrado na Figura 4.1, FS-MAC dá suporte à existência de mais de uma métrica simultaneamente, portanto, algumas delas podem ser centralizadas e outras distribuídas. A arquitetura prevê também a possibilidade de inclusão de novas unidades de coleta e processamento nesse módulo sem necessidade de alterar as que já existem. O impacto de acrescentar novas métricas é o tratamento delas no

módulo de Decisão, já que a razão de incluí-las é justamente poder utilizar seus valores na seleção de algum protocolo pelo método de avaliação.

A decisão é tomada utilizando os valores das métricas recebidos de forma centralizada e distribuída. O que é informado de fato ao módulo de Troca é uma lista com o grau de adaptabilidade de cada protocolo. O módulo de Decisão decide apenas qual protocolo é mais adequado, a escolha de substituir ou não o protocolo em operação é feita no módulo de Troca e leva em conta outros elementos.

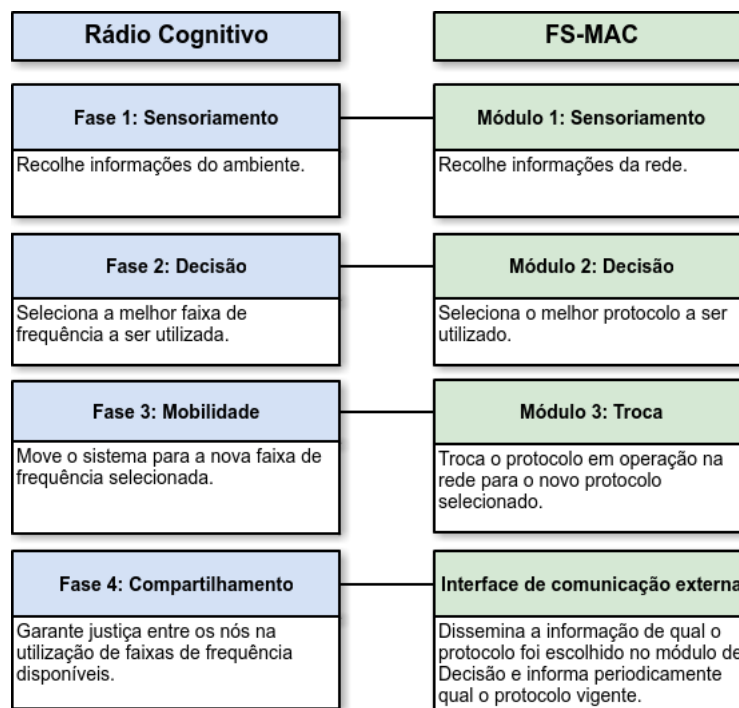
Nas seções a seguir apresentamos com mais detalhes cada módulo da arquitetura.

### 4.1.1 FS-MAC e Rádios cognitivos

A semelhança entre o problema abordado nessa dissertação e aquele tratado em rádios cognitivos (RC) permite que a estrutura da solução apresentada em RC seja aproveitada no caso da troca de protocolos. Em rádios cognitivos existe um conjunto de frequências que podem ser selecionadas, no caso do FS-MAC existe um conjunto de protocolos. Em RC, é preciso recolher informações do ambiente para selecionar a melhor faixa, esta é a fase 1 (*Sensoriamento*). No nosso caso também é necessário colher informações de contexto para escolher o melhor protocolo a ser utilizado, este é o módulo 1 (*Sensoriamento*). Em RC, na fase 2 (*Decisão*) é preciso utilizar as informações recolhidas e selecionar a melhor faixa de frequência. Da mesma forma, no FS-MAC é preciso usar as informações de contexto e selecionar o melhor protocolo a entrar em operação, isso é feito no módulo 2 (*Decisão*). Na fase 3 (*Mobilidade*), em rádios cognitivos, o sistema deixa uma faixa de frequência e passa a utilizar outra. Da mesma forma, no FS-MAC, através do módulo 3 (*Troca*), a subcamada MAC deixa de utilizar um protocolo e passa a utilizar aquele selecionado no módulo 2. Em RC, na fase 4 (*Compartilhamento*), o sistema procura garantir justiça entre os nós quanto à utilização das faixas disponíveis. No FS-MAC, a (*Interface de Comunicação Externa*), implementada em todos os protocolos do Conjunto, procura manter todos os nós operando com o mesmo protocolo ativo. Para isso, ela dissemina a informação de qual protocolo foi escolhido no módulo 2 e informa periodicamente qual o protocolo ativo. A Figura 4.2 mostra a correspondência entre Rádio cognitivo e a arquitetura FS-MAC.

### 4.1.2 Sensoriamento

O módulo de Sensoriamento consiste em colher informações da rede que serão usadas para decidir qual protocolo será utilizado. Este módulo é composto por unidades básicas de coleta e processamento, cada unidade é responsável por coletar dados da



**Figura 4.2.** Correspondência entre Rádios cognitivos e FS-MAC

rede, processá-los e gerar o valor de uma métrica específica. Algumas métricas possíveis são: quantidade de estações se comunicando, volume de pacotes enviados, tipo de fluxo (contínuo ou rajadas) e quantidade de pacotes confirmados. As métricas a serem utilizadas estão relacionadas ao contexto da rede de forma geral, portanto podem vir da própria camada de enlace, da camada de rede ou mesmo de outras camadas superiores como transporte e aplicação. Além disso, o sensoriamento pode ser centralizado ou distribuído, e essa escolha pode variar de acordo com cada métrica.

No caso de sensoriamento centralizado, apenas o nó Coordenador realiza as coletas. Nesse caso, essa estação é responsável por recolher os dados da rede, gerar o valor da métrica e repassá-lo ao módulo de Decisão. Os demais nós da rede mantêm essa função inativa durante toda a comunicação. Um exemplo de métrica que pode ser gerada centralizadamente é o número de estações transmitindo. Quando o sensoriamento é distribuído, todas as estações recolhem dados individualmente e enviam o valor calculado da métrica ao módulo de Decisão do nó Coordenador. Alguns exemplos de métricas que podem ser geradas de forma distribuída são taxa de entrega e latência.

Embora o sensoriamento centralizado seja mais simples, a decisão de manter a possibilidade de um sensoriamento distribuído se apoia no fato de que algumas métricas não podem ser geradas de forma centralizada ou perdem qualidade nesse caso. O valor de uma métrica como taxa de entrega, por exemplo, não pode ser calculado

centralizadamente, pois a um nó não é possível saber isoladamente se outro recebeu confirmação de seus pacotes. Outra métrica como nível de contenção da rede, quando calculada centralizadamente levando em consideração dados locais, pode não corresponder à realidade da rede, principalmente durante a execução de protocolos como CSMA/CA onde não há garantia de justiça.

Considerando que há sensoriamento distribuído, a comunicação é feita diretamente entre o módulo de Sensoriamento de um nó Comum e o módulo de Sensoriamento do nó Coordenador. O valor da métrica é enviado de forma periódica do nó Comum para o Coordenador a fim de manter esse valor atualizado e coerente com o estado da rede. No nó Coordenador, o valor da métrica recebido é encaminhado ao módulo de Decisão onde será usado para avaliar a adaptabilidade dos protocolos.

### 4.1.3 Decisão

O módulo de decisão recebe as informações do Sensoriamento e calcula a adaptabilidade de cada protocolo ao contexto atual da rede. Esse módulo precisa conhecer também as características dos protocolos que fazem parte do conjunto a ser utilizado. Essas informações incluem os cenários onde os protocolos apresentam maior eficiência e as situações onde eles não possuem bom desempenho.

O método de avaliação é o responsável pela compilação de todos os dados e pela decisão de qual protocolo melhor se adapta à rede naquele momento. Este método pode ser implementado utilizando diversos mecanismos como lógica *fuzzy*, aprendizado de máquina ou inteligência artificial por exemplo. Na versão atual decidimos utilizar lógica *fuzzy* para realizar a avaliação. Fizemos essa escolha primeiramente porque o mecanismo é robusto, ou seja, mesmo com entradas não muito precisas e com ruído, fornece saídas satisfatórias para o problema abordado. A segunda razão é a simplicidade, levando em conta o pequeno número de variáveis de entrada que temos, a implementação do sistema de regras de inferência é simples e pode ser alterado com facilidade.

Assim, no processo de implementação do método de avaliação, os protocolos foram caracterizados separadamente e os resultados dessa caracterização foram utilizados para gerar as inferências *fuzzy*. As variáveis de entrada são as métricas vindas do módulo de sensoriamento e as variáveis de saída são as adaptabilidades de cada protocolo.

A avaliação é feita centralizadamente no nó Coordenador e o resultado é uma lista com a adaptabilidade de cada protocolo. Os valores de adaptabilidade variam de zero a cem, onde zero significa a pior adaptabilidade possível e cem significa completamente

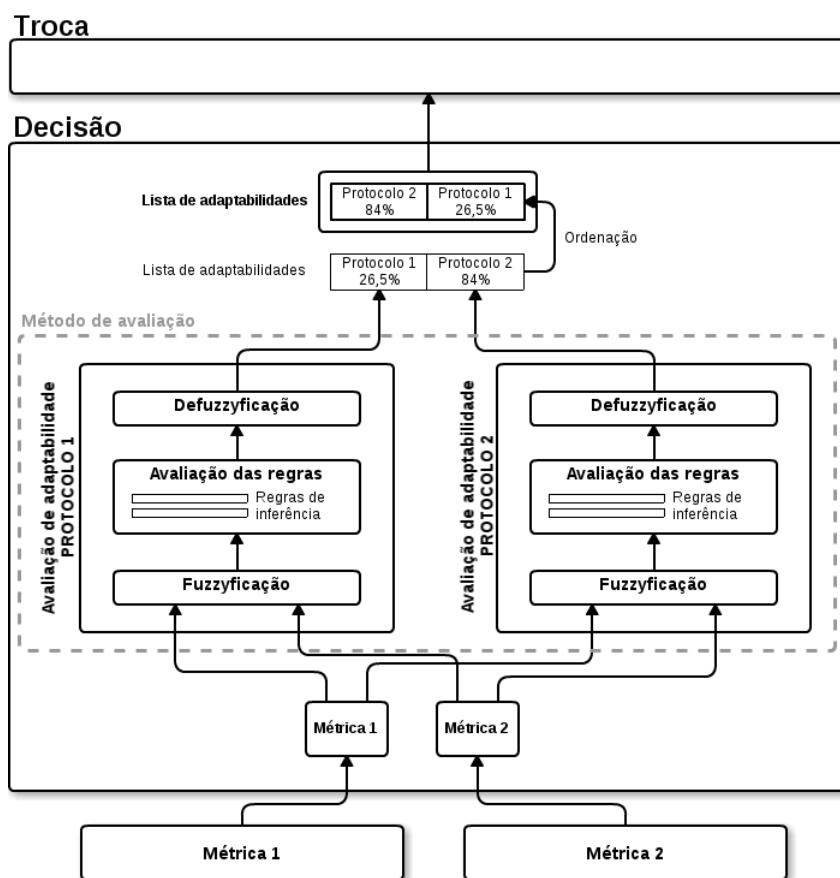
adaptável. Essa lista é ordenada em ordem decrescente, ou seja, o primeiro elemento contém a maior adaptabilidade. Posteriormente a lista é enviada para o módulo de Troca para que o protocolo em operação seja substituído caso necessário. Assim, o módulo de Decisão é responsável por decidir apenas qual protocolo se adapta melhor às condições apresentadas pela rede em cada momento, deixando a avaliação sobre a necessidade de efetiva substituição a cargo do módulo de Troca. Essa decisão abre a possibilidade de um baixo acoplamento entre esses dois módulos, pois caso o valor da métrica utilizada não seja dependente do contexto do protocolo, o módulo de Decisão não precisa saber que protocolo está em operação no momento, e dessa forma, não precisa receber dados a esse respeito, apenas envia a lista de adaptabilidades ao módulo de Troca.

Assim que gera a lista de adaptabilidade dos protocolos, a informação é comunicada ao módulo de Troca. Esse processo deve acontecer periodicamente e a avaliação dessa periodicidade deve estar atenta ao custo/benefício desse contexto. Se o período for muito longo, a rede poderá funcionar muito tempo com um protocolo inadequado. Por outro lado, se o período for muito curto, o processamento adicional pode sobrecarregar desnecessariamente a rede.

O funcionamento do módulo de Decisão é mostrado na Figura 4.3. De todos os módulos, esse é o mais impactado com as modificações que a flexibilidade da arquitetura permite. Quando um novo protocolo é incluído ao conjunto, é necessário que suas características sejam acrescentadas a esse módulo para que o método de avaliação possa calcular a sua adaptabilidade e o protocolo possa ser eventualmente selecionado. Como escolhemos utilizar o mecanismo de lógica *fuzzy*, “acrescentar as características”, nesse caso, significa acrescentar um bloco de avaliação de adaptabilidade com regras de inferência baseadas nas características do novo protocolo, de forma a permitir que o módulo de Decisão o selecione no cenário em que ele é mais eficiente. O módulo de Decisão também sofre impacto quando alguma unidade de coleta e processamento é acrescentada no módulo de Sensoriamento, pois as informações da métrica calculadas precisam ser tratadas no método de avaliação. Por outro lado, as modificações executadas nesse módulo, como alteração do método de avaliação ou modificação no tratamento das métricas, não têm nenhum impacto nos outros módulos.

#### 4.1.4 Troca

O módulo de Troca é o mais importante entre todos, pois ele trata do objetivo central da arquitetura. Nesse módulo, a informação de adaptabilidade de cada protocolo já está consolidada e é comunicada pelo módulo de Decisão. A partir do recebimento da



**Figura 4.3.** Detalhes do módulo de Decisão usando duas métricas e dois protocolos

lista, o processo de troca ocorre em três passos:

1. A lista de adaptabilidades é analisada, se o protocolo em operação é diferente daquele com maior grau de adaptabilidade na lista, os dois protocolos são comparados. Para que a troca seja viável, é preciso que a diferença entre a adaptabilidade dos dois protocolos seja maior que um certo limiar. A definição desse limiar deve garantir que o custo operacional da troca possa ser compensado pela melhora de desempenho na rede. Caso a diferença seja maior, o protocolo deve ser efetivamente trocado e os passos seguintes são executados, caso contrário, mantêm-se o protocolo atual.
2. Depois de concluído que deve haver a troca, o nó Coordenador comunica imediatamente a todos os nós da rede qual deve ser o novo protocolo a entrar em operação.
3. Internamente, cada nó da rede substitui o protocolo em operação pelo novo pro-



protocolo definido.

Durante esse processo de substituição, o módulo de Troca terá que se comunicar internamente com o conjunto de protocolos, mais especificamente com dois deles. Para isso, eles utilizam a interface de comunicação direcionada para essa troca. O protocolo que será desativado receberá uma notificação e deve enviar os dados do seu estado de operação. Já o protocolo que entrará em operação, receberá a notificação e em seguida receberá o estado de operação do protocolo desativado e iniciará o funcionamento a partir desse estado. O Algoritmo 1 mostra o funcionamento da troca como ocorre no nó Coordenador e o Algoritmo 2 mostra o funcionamento da troca nos nós Comuns.

O estado de operação de um protocolo corresponde a uma fila contendo todos os pacotes que ainda não foram enviados e confirmados, na ordem em que aparecem na estrutura. Apenas a notificação de necessidade de troca acarretará envios de pacotes de controle na rede, pois o módulo de troca de cada estação irá intermediar o trânsito interno dos dados de estado de operação.

A efetivação da troca pode ser implementada de diversas maneiras, mas o principal problema relacionado a ela está na sincronização dessa operação. Durante a substituição, os mecanismos utilizados devem tratar a possibilidade de que algumas estações troquem de protocolo enquanto outras permaneçam no protocolo anterior. O resultado de ter dois protocolos diferentes operando na rede deve ser avaliado. Duas abordagens possíveis para esse problema são:

- Conduzir uma troca distribuída atômica, ou seja, enquanto houver pelo menos um nó operando no protocolo antigo, o novo não entra em operação.
- Um mecanismo periódico avisa a todos os nós de tempos em tempos que protocolo está em atividade, minimizando assim o tempo de operação com protocolos diferentes.

Na primeira abordagem, existe a necessidade de disseminação de dados de forma atômica. Como não existe implementação de nenhum mecanismo que possibilite isso em USRPs, escolher essa opção acrescentaria complexidade considerável ao trabalho.

Escolhemos a segunda abordagem porque ela é simples e os protocolos que usamos nessa versão são CSMA/CA e TDMA. Esses protocolos, embora degradem o desempenho da rede quando operando simultaneamente, podem ser executados no mesmo ambiente sem impedir que a rede funcione. Para minimizar o tempo de operação simultâneo, o módulo de troca possui duas atribuições. A primeira é comunicar imediatamente aos nós da rede quando a troca deve ocorrer, a segunda é comunicar periodicamente a esses

nós qual é o protocolo em operação no momento. Assim, se algum nó não conseguir executar a troca, terá a oportunidade periódica de se adequar.

Um aspecto importante é que a interface entre o módulo de Decisão e o módulo de Troca é bem definida e quando as regras de inferência não dependem do contexto dos protocolos, essa interface possui acoplamento mínimo, pois o módulo de Troca apenas utiliza a informação da lista de adaptabilidade. Os módulos podem ser melhorados, atualizados, substituídos de forma completamente independente desde que a interface seja mantida. Entre o módulo de Troca e o Conjunto de protocolos existe uma dependência, mas como a interface de troca é padronizada, o tratamento dessa dependência é simples. Ao acrescentar um novo protocolo ao conjunto, basta adicionar essa informação ao módulo de Troca.

---

**Algoritmo 1:** Procedimento de troca no nó Coordenador
 

---

```

início
  se protocoloEmOperação  $\neq$  protocoloMaisAdaptável então
    se adaptProtocoloMaisAdaptável - adaptProtocoloEmOperação > limiar) então
      Envia aos outros nós da rede a notificaçãoDeTroca
      Envia a protocoloEmOperação a mensagem Parar atuação
      Recebe de protocoloEmOperação a mensagem Atuação parada e a Fila de envios
      Envia a protocoloMaisAdaptável a mensagem Iniciar atuação e a Fila de envios
    fim
  fim
fim
  
```

---



---

**Algoritmo 2:** Procedimento de troca no nó Comum
 

---

```

Entrada: notificaçãoDeTroca, protocoloEmOperação, protocoloMaisAdaptável
início
  Recebe do nó Coordenador a notificaçãoDeTroca
  Envia a protocoloEmOperação a mensagem Parar atuação
  Recebe de protocoloEmOperação a mensagem Atuação parada e a Fila de envios
  Envia a protocoloMaisAdaptável a mensagem Iniciar atuação e a Fila de envios
fim
  
```

---

### 4.1.5 Conjunto de protocolos

Esse conjunto compreende os protocolos implementados e que podem ser escolhidos para utilização de acordo com as características informadas ao método de avaliação do módulo de Decisão. O conjunto deve conter protocolos que sejam eficientes em

diferentes cenários, de acordo com o comportamento desejado da rede. Ao acrescentar um elemento a esse conjunto, suas características devem ser informadas ao módulo de Decisão e sua disponibilidade deve ser adicionada ao módulo de Troca. O desenvolvimento de um protocolo para ser acrescentado ao Conjunto deve seguir o padrão utilizado na rede e conter, além disso, a implementação de duas interfaces: *Interface de troca* e *Interface de comunicação externa*.

A interface de troca possui duas funções básicas:

1. Quando receber uma mensagem de desativação, deve interromper a operação do protocolo e retornar a lista de pacotes não enviados e não confirmados.
2. Quando receber uma mensagem de ativação e uma lista de pacotes, deve colocar os pacotes na fila de envio e iniciar a operação do protocolo.

Mesmo com a interface de troca, continua existindo um problema relativo às mensagens do FS-MAC que precisam transitar pela rede, como comunicado de troca e valores de métricas calculados de forma distribuída. Inicialmente havia a ideia de que cada módulo teria acesso à camada física para envio desses pacotes individualmente. No entanto, com essa abordagem, há a dificuldade de se adequar essas transmissões ao sistema de envios do protocolo em operação. Enviá-las arbitrariamente compromete muito o sucesso do seu recebimento.

Assim, escolhemos criar uma interface de comunicação externa, essa interface é responsável por transmitir as mensagens do FS-MAC pela rede. Seu funcionamento é simples, assim que recebe uma mensagem FS-MAC que deve ser enviada na rede, esta mensagem deve ser a próxima a ser enviada respeitando o esquema de comunicação do próprio protocolo. Dessa forma, qualquer módulo FS-MAC que queira enviar mensagens pela rede, deve utilizar essa interface.

Com a definição dessas interfaces, fica simples adaptar qualquer protocolo já implementado à arquitetura FS-MAC. Independente da implementação, elas são as mesmas para todos os protocolos, o que possibilita aos outros módulos, especialmente o de Troca, ignorar o funcionamento interno de cada protocolo.

## 4.2 Implementação

Baseando na arquitetura apresentada na seção anterior, implementamos um protótipo com a finalidade de avaliar a plataforma FS-MAC. Esse protótipo segue os princípios propostos para a arquitetura e as decisões gerais do projeto, e em alguns casos, contém decisões específicas de implementação. Nessa seção, apresentamos os detalhes de

implementação de cada fase, apontamos as decisões tomadas e sempre que possível, apresentamos as outras possíveis decisões, as vantagens e desvantagens de cada uma e as justificativas de nossa escolha.

Na implementação utilizamos a tecnologia SDR através do *framework* Ettus USRP com o apoio da plataforma de desenvolvimento GNU Radio. Como base para a implementação, utilizando a pilha ZigBee do trabalho [Schmid et al., 2016]. A partir dessa pilha, substituímos o bloco da camada MAC pelo conjunto de blocos que compõem a plataforma FS-MAC, construídos por nós. A implementação utiliza os valores de latência e quantidade de transmissores para decidir que protocolo mais se adapta às condições da rede e no conjunto de protocolos, foram implementados o CSMA e TDMA.

Os blocos não nativos do GNU Radio são chamados *out-of-tree* ou simplesmente OTT, esses blocos podem ser construídos através de duas linguagens, Python ou C++. De acordo com o site oficial do GNR, caso a performance seja um requisito importante no projeto, deve-se utilizar C++, em outro caso, Python é uma boa opção, pois permite um desenvolvimento mais rápido, tendo em vista que os módulos não precisam ser compilados para serem testados. Em todos os módulos onde foi necessário realizar operações de CRC e manipular o conteúdo dos pacotes no padrão 802.15.4, utilizamos C++, nos outros casos, desenvolvemos utilizando Python.

O protótipo possui no total 6 blocos:

1. **Sensor de número de transmissores:** Realiza as operações referentes à coleta de dados e geração do valor da métrica de *quantidade de transmissores* a ser utilizado no módulo de Decisão.
2. **Sensor de latência:** Realiza as operações referentes à coleta de dados e geração do valor da métrica de *latência de entrega dos pacotes* a ser utilizado no módulo de Decisão.
3. **Decisão:** Recebe o valor das métricas consideradas e calcula a adaptabilidade de cada protocolo às condições atuais da rede.
4. **Troca:** Recebe o valor da adaptabilidade de cada protocolo e avalia a necessidade de substituir o protocolo em operação.
5. **CSMA:** Controla o acesso ao meio baseando-se em mecanismos de contenção.
6. **TDMA:** Controla o acesso ao meio baseando-se na divisão e alocação de *slots* de tempo.

Os blocos 1, 2, 4, 5 e 6 foram implementados utilizando a linguagem C++. Essa decisão se deve à necessidade de manipular pacotes no padrão 802.15.4, o que inclui a realização de operações de cálculo de CRC e construção da estrutura dos pacotes em nível mais baixo. Já o bloco 3 foi implementado utilizando a linguagem Python, pois nele não há a necessidade de manipular pacotes mais complexos, apenas valores passados internamente entre os blocos. Como mostra a Figura 4.4, os blocos 1 e 2 fazem parte da fase de Sensoriamento da arquitetura FS-MAC, o bloco 3 representa a fase de Decisão, o bloco 4 representa a fase de Troca e os blocos 5 e 6 compõem o conjunto de protocolos da plataforma FS-MAC.

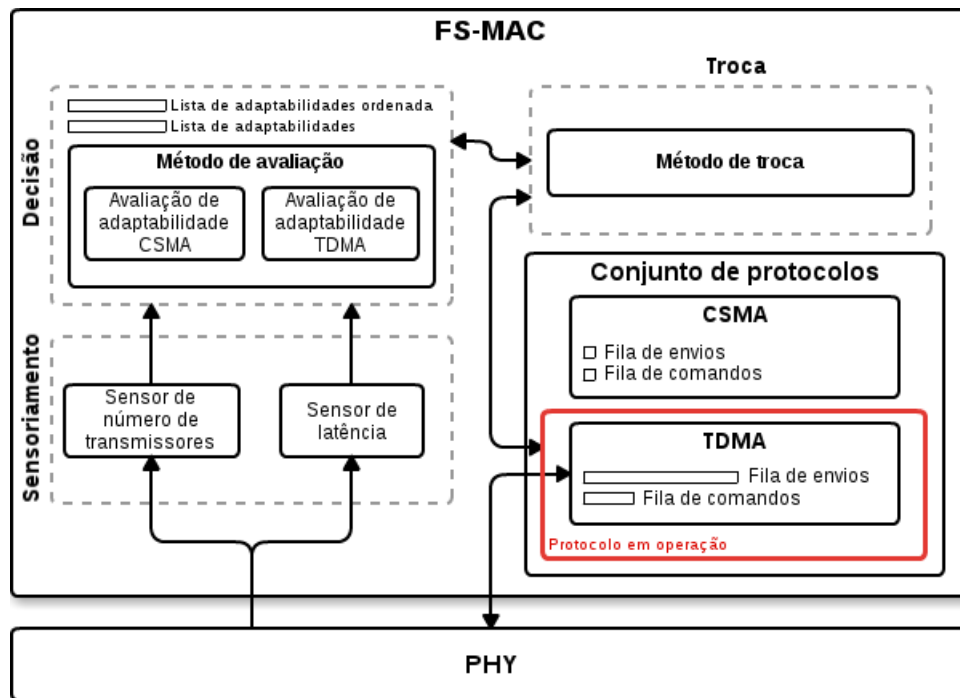


Figura 4.4. Implementação FS-MAC

Durante o funcionamento do protótipo, a comunicação entre os blocos é feita através de mensagens de controle FS-MAC. Foram implementados 6 tipos de mensagem que garantem o completo funcionamento da plataforma. São eles: *Envio de informações*, *Informações enviadas*, *Parar atuação*, *Atuação parada*, *Iniciar atuação* e *Atuação iniciada*. Essas mensagens são tratadas de forma diferente se o nó é do tipo Coordenador ou Comum. Assim, cada nó é configurado inicialmente como Coordenador ou Comum para que internamente possa tratar corretamente as mensagens.

### 4.2.1 Protocolos

A plataforma FS-MAC deve conter um conjunto de protocolos que serão colocados em execução de acordo com as condições da rede. O protótipo que construímos contém nesse conjunto dois protocolos, um baseado em contenção (CSMA) e outro livre de contenção (TDMA). Usamos como apoio para a implementação, a pilha ZigBee apresentada no trabalho [Schmid et al., 2016]. A camada MAC dessa implementação ZigBee é muito limitada, além de intermediar os pacotes entre as camadas de rede e física, a única função existente é a conferência de integridade do pacote através do CRC. Assim, foi preciso desenvolver uma plataforma básica com algumas estruturas e funções que possibilitassem iniciar a construção dos protocolos. Os detalhes da implementação completa dos protocolos estão descritos no Apêndice A.

Conforme descrito na Seção 4.1, para incluir os protocolos no funcionamento da plataforma FS-MAC, eles devem implementar duas interfaces: *Troca* e *Comunicação externa*. Assim, os protocolos implementados e descritos no Apêndice A funcionam separadamente, mas para integrarem o conjunto de protocolos da plataforma FS-MAC foi preciso acrescentar a eles a implementação dessas duas interfaces. A comunicação dos protocolos com o módulo de Troca acontece através de portas de controle. Nos blocos do GNU Radio correspondentes a cada protocolo implementado, foram acrescentadas duas portas, uma de entrada e outra de saída, onde transitam apenas mensagens de controle FS-MAC. Dessa forma, é também através dessas portas que as interfaces implementadas se comunicam com esse módulo.

A *Interface de troca* recebe mensagens de dois tipos: *Parar atuação* e *Iniciar atuação* e retorna também dois tipos de mensagem *Atuação iniciada* e *Atuação parada*. Quando recebe a mensagem do tipo *Parar atuação*, o protocolo interrompe o seu próprio funcionamento e retorna uma mensagem informando que a atuação está interrompida e fornece a fila de envios no estado em que se encontra. Por outro lado, quando recebe a mensagem de *Iniciar atuação* juntamente com a fila de envios, o protocolo estabelece a fila recebida como sua própria fila de envios e inicia a sua atuação. Após o início da atuação, o protocolo envia para o módulo de troca uma mensagem do tipo *Atuação iniciada* para que esse módulo dê prosseguimento ao funcionamento da rede. Os detalhes da implementação dessa interface estão descritos na seção A.4.

A *Interface de comunicação externa* recebe mensagens do tipo *Envio de informações* juntamente com a informação a ser enviada para a rede. Quando recebe essa mensagem, o protocolo gera um pacote no mesmo formato dos seus pacotes internos e como *payload* adiciona as informações a serem enviadas para a rede. O protocolo mantém uma fila apenas com pacotes FS-MAC, que devem ter prioridade sobre os outros.

Assim, antes de enviar qualquer pacote, o protocolo verifica se há algum pacote na fila de mensagens FS-MAC, caso exista, eles são enviados prioritariamente. Essa abordagem permite que mensagens FS-MAC sejam enviadas na rede respeitando o esquema de atuação do protocolo em atividade no momento. Em alguns momentos, como o processo de Troca, o bloco que solicitou o envio das mensagens ao protocolo precisa saber se o envio foi concluído, nesse caso, o protocolo retorna pela porta de controle uma mensagem do tipo *Informações enviadas*.

Devido às possibilidades de perda do pacote FS-MAC enviado, foi adicionada uma redundância para minimizar os problemas causados por essas perdas. Como as mensagens FS-MAC são sempre do tipo *broadcast*, elas não são confirmadas, assim, não é possível saber se a mensagem realmente chegou. Dessa forma, para minimizar perdas, o envio de cada mensagem é repetido duas vezes. Há outros mecanismos na plataforma FS-MAC, especificamente no módulo de Troca, para o caso de a mensagem não chegar ao destino mesmo sendo enviada com redundância. Esse mecanismo é descrito na seção 4.2.4.

## 4.2.2 Sensoriamento

Para a fase de Sensoriamento, foram implementadas duas métricas, *Quantidade de transmissores* e *Latência*. Os dados para gerar o valor da métrica de quantidade de transmissores são colhidos de forma centralizada, ou seja, apenas um nó é responsável por essa métrica. Como o valor dessa métrica é utilizado no módulo de Decisão do nó FS-MAC Coordenador, este nó também é o responsável por essa métrica. Já os dados da métrica de latência são colhidos de forma distribuída, ou seja, cada nó reúne as informações locais referentes à latência de entrega dos pacotes e as envia para o nó Coordenador para que seja gerado o valor da métrica que será enviado ao módulo de Decisão.

Cada bloco do GNU Radio, responsável pelo sensoriamento de cada métrica, possui uma opção de classificação como nó Comum ou Coordenador. Esse valor define internamente o comportamento do nó, de forma que não há blocos diferentes para cada tipo de nó, nem implementações distintas para essa diferenciação. Nessa fase, há grande diferença entre o comportamento dos nós Comuns e do Coordenador, no entanto, o mesmo bloco pode ser utilizado nos dois casos, modificando apenas a configuração através da interface gráfica do GNU Radio.

No caso da métrica centralizada, referente à quantidade de transmissores, os nós Comuns não realizam qualquer ação. Assim que percebem que são nós Comuns e que a métrica é centralizada, mantêm-se inativos durante todo o processo. O nó Coordenador,

quando é ativado, percebe que é Coordenador e que é responsável pela métrica, então, inicia a escuta da rede. Todos os pacotes vindos da camada física passam pelo bloco dessa métrica, então, cada pacote é desencapsulado do pacote PMT (*Polymorphic Types*) e seu endereço de origem é verificado. O bloco mantém internamente um conjunto de endereços ao qual acrescenta cada endereço novo verificado em algum pacote. A cada cinco segundos, o valor da quantidade de endereços desse conjunto é encapsulado em um pacote PMT e enviado ao módulo de Decisão. Dessa forma, o bloco que decidirá que protocolo se adapta melhor às condições da rede terá o valor dessa métrica atualizado a cada 5 segundos. Essa frequência foi escolhida empiricamente, caso a frequência fosse menor, a rede correria o risco de operar com um protocolo inadequado por mais tempo. No entanto, se a frequência fosse maior, a quantidade de pacotes enviados ao módulo de Decisão consumiria processamento e estaria enviando a mesma informação repetidamente.

Em relação à métrica de latência, como se trata de coleta de informações de forma distribuída, os dois tipos de nó têm atuação no processo. O bloco referente a essa métrica no nó Comum, quando ativado, percebe o seu tipo e a natureza da métrica, então, inicia a coleta dos dados de latência para serem enviados ao nó Coordenador. Nesse ponto, duas decisões importantes foram tomadas, primeiramente era necessário decidir se os dados coletados seriam enviados diretamente ao conjunto de protocolos para serem enviados à rede ou se seriam encaminhados ao módulo de Decisão. Caso fossem enviados diretamente, haveria uma economia de processamento na passagem interna de mensagens de controle. Por outro lado, o módulo que faz o sensoriamento deveria saber qual protocolo está em operação para enviar a ele as informações a serem encaminhadas. Outra desvantagem dessa abordagem é que o fluxograma do nó Coordenador seria diferente dos nós Comuns. O envio dos dados para o bloco de Decisão elimina o problema de fluxogramas diferentes e a necessidade de saber qual protocolo está em operação. Assim, a implementação envia a mensagem ao modo de Decisão, que a tratará conforme seu tipo e continuará o processo para enviá-la ao nó Coordenador.

A segunda questão diz respeito ao fluxo da coleta e envio dos valores de latência. Visto que esses valores são gerados dentro do protocolo, o bloco de sensoriamento não manipula de fato esses dados. Esse bloco envia apenas uma mensagem de controle que chegará ao protocolo informando que os dados de latência devem ser enviados à rede. Internamente, o protocolo em operação coleta esses dados e os envia.

A mensagem que inicia esse processo sai do módulo de sensoriamento a cada 10 segundos. Inicialmente testamos uma frequência maior, com intervalos de 5 segundos, no entanto, como a variação dos valores de latência é alta, os valores coletados em



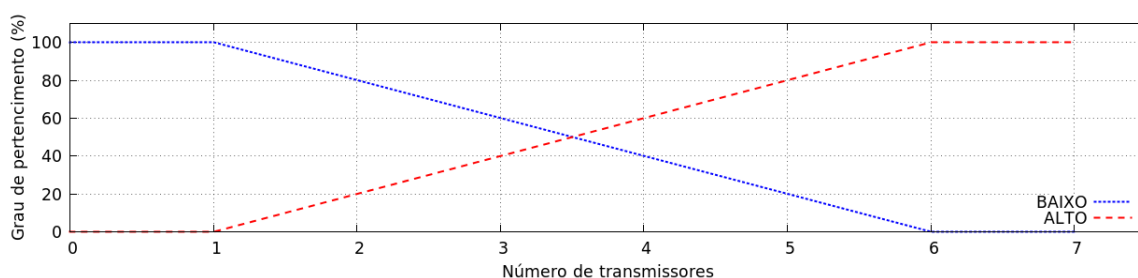
um intervalo de apenas 5 segundos estava gerando valores extremos que confundiam o módulo de Decisão, levando-o a calcular a adaptabilidade dos protocolos erroneamente. Com o intervalo de dez segundos, os resultados se mostraram mais compatíveis com o esperado e essa frequência foi mantida.

### 4.2.3 Decisão

O modelo de decisão construído emprega lógica fuzzy, e é modelado como descrevemos a seguir.

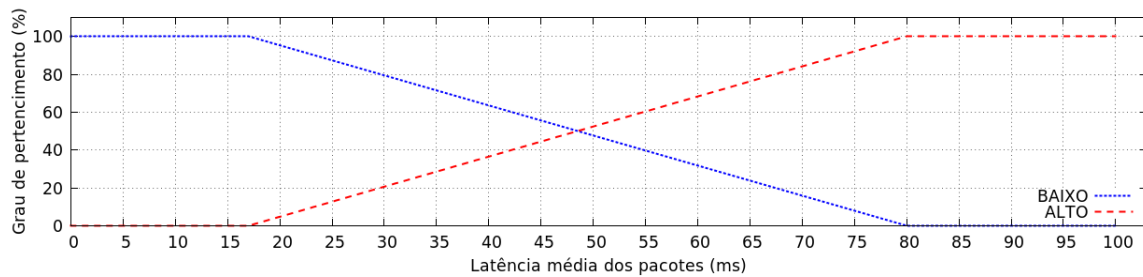
Variáveis linguísticas: O modelo considera três variáveis linguísticas, que são *i*) Latência média dos pacotes (*LM*), *ii*) número de transmissores (*NT*) e *iii*) Adaptabilidade do protocolo (*ADP*). Todas elas aceitam os termos nebulosos *BAIXO* e *ALTO*. Foram empregadas as funções de mapeamento apresentadas nas figuras 4.5, 4.6, 4.7 e 4.8. Essas funções foram geradas a partir da observação do comportamento dos dois protocolos durante os experimentos descritos na seção 5.3 (*Avaliação e caracterização dos protocolos*). A partir desses experimentos observamos em que níveis de contenção cada protocolo se destaca, e assim, classificamos cada faixa de valores relacionados às variáveis linguísticas como *BAIXO* ou *ALTO*.

É importante notar que existem funções distintas para CSMA e TDMA. Isso é necessário porque o pertencimento do valor da latência ao conjunto *BAIXO* ou *ALTO*, usado para inferir a contenção da rede, depende do contexto do protocolo em operação. Como o TDMA tende a ter latências mais baixas e com menor variação, um valor de latência gerado com o TDMA em operação, que indica que há muitas tentativas de transmissão na rede, se for gerado durante a operação com CSMA, pode indicar que há poucas tentativas de transmissão.

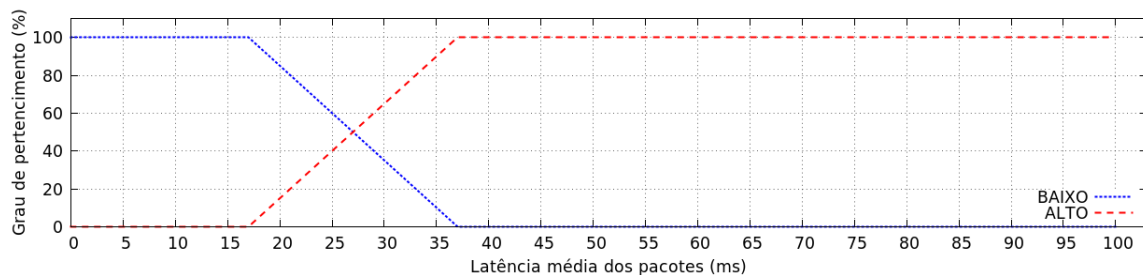


**Figura 4.5.** Função de pertinência para número de transmissores.

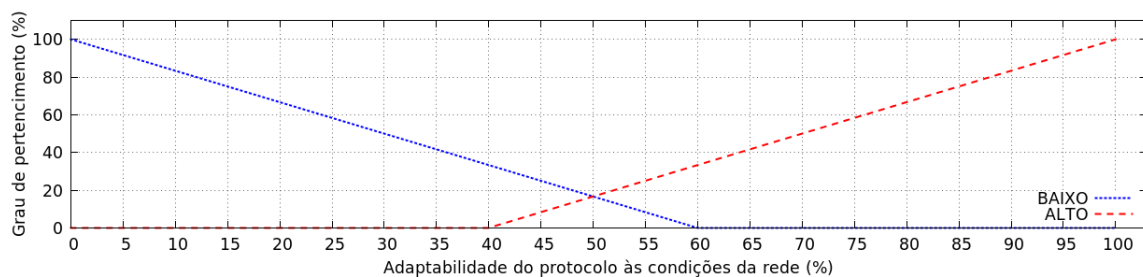
O motor de inferência, por sua vez, calcula o grau de adaptatividade de cada protocolo, conforme modelo descrito na Figura 4.3. As regras de inferência empregadas foram derivadas a partir da constatação de que técnicas de TDMA são adequadas



**Figura 4.6.** Função de pertinência para latência média de entrega dos pacotes (CSMA).



**Figura 4.7.** Função de pertinência para latência média de entrega dos pacotes (TDMA).



**Figura 4.8.** Função de pertinência para adaptabilidade dos protocolos.

quando temos muitos transmissores e quando há um grande grau de contenção. Essas regras consideram também que as abordagens com CSMA se adaptam melhor em cenários com número menor de transmissores ou baixa contenção. A contenção é medida pela latência média, que captura a quantidade de retransmissões no meio. O motor de inferência utiliza as seguintes regras para calcular a adaptabilidade de cada protocolo:

CSMA

Se NT é BAIXO e LM é ALTO então ADP é ALTO

Se NT é BAIXO e LM é BAIXO então ADP é ALTO

Se NT é ALTO e LM ALTO então ADP é BAIXO

Se NT é ALTO e LM é BAIXO então ADP é ALTO

TDMA

Se NT é BAIXO e LM é ALTO então ADP é BAIXO

Se NT é BAIXO e LM é BAIXO então ADP é BAIXO

Se NT é ALTO e LM é ALTO então ADP é ALTO

Se NT é ALTO e LM é BAIXO então ADP é BAIXO

A defuzzificação das regras é feita empregando o método da centróide [Passino et al., 1998]. Esse método retorna um valor entre zero e cem, que na nossa implementação corresponde à adaptabilidade de cada protocolo. A partir desse valor, um dicionário é construído e enviado ao módulo de Troca. Nessa estrutura, as chaves correspondem aos identificadores dos protocolos e os valores correspondem aos graus de adaptabilidade desses protocolos.

#### 4.2.4 Troca

A fase de troca foi implementada inteira em um bloco GNU Radio. Além de executar a troca em si, este módulo é responsável por intermediar a comunicação entre a camada de Rede e o protocolo ativo e também entre o protocolo ativo e a camada Física. Essa função de intermediação é simples, o módulo apenas recebe as informações das outras camadas e encaminha ao protocolo ativo. Da mesma forma, quando o protocolo ativo deseja se comunicar com as outras camadas, o módulo recebe os pacotes do protocolo e os encaminha a essas camadas.

O bloco implementado possui comportamento diferente dependendo do tipo de nó. Quando se trata de nó Comum ele faz as funções de intermediação entre as outras camadas e o protocolo em atividade e também a intermediação entre o módulo Decisão e os protocolos. No caso do nó Coordenador, o bloco possui mais duas funções, a de coordenar a troca na rede e a de informar periodicamente aos outros nós que protocolo está em atividade no momento.

Quando configurado como nó Comum, a intermediação entre o módulo de Decisão e os protocolos ocorre apenas nos casos em que há sensoriamento distribuído. Nesse caso, o módulo de sensoriamento que precisa enviar informações pela rede, repassa essas informações ao módulo de Decisão, que as encaminha ao módulo de Troca. Quando o módulo de Troca recebe essas informações, ele envia ao protocolo ativo uma mensagem do tipo *Envio de informações* juntamente com as informações a serem enviadas. No caso do protótipo implementado, essa situação acontece com o sensoriamento que envolve a métrica de Latência.

Quando configurado como nó Coordenador, a principal atribuição do bloco é realizar a Troca em si. Assim que recebe do módulo de Decisão a lista de adaptabilidades dos protocolos do conjunto, o módulo de Troca verifica se o protocolo ativo é diferente do protocolo com maior adaptabilidade na lista. Caso seja, ele verifica se a diferença entre o valor das adaptabilidades é maior que 5%. Em caso positivo inicia-se a troca. Esse limiar possui duas funções, a primeira é evitar que a troca aconteça quando os protocolos não têm diferença significativa de desempenho e a segunda é evitar que por pequenas variações, aconteçam sucessivas trocas. O valor de 5% foi escolhido empiricamente. Este valor é grande o suficiente para evitar sucessivas trocas, mas não tão grande que impeça essas trocas quando o desempenho tem diferença significativa.

A troca do protocolo em operação na rede está ilustrada no diagrama da Figura 4.9. A troca se inicia com uma mensagem do tipo *Envio de informações* enviada ao protocolo ativo no momento para que ele informe aos outros nós a necessidade de troca. O protocolo ativo envia essa mensagem *broadcast* pela rede e após o envio, responde ao bloco de Troca com uma mensagem do tipo *Informações enviadas*. Ao receber essa mensagem, o módulo de Troca inicia a sua troca local, que funciona da mesma forma tanto nos nós Comuns quanto nos Coordenadores. Essa troca local tem início com o envio de uma mensagem do tipo *Parar atuação* para o protocolo ativo. O módulo de Troca aguarda a parada do protocolo e a resposta com uma mensagem de *Atuação parada (Fila de envios)*. Após receber essa mensagem, o módulo de Troca envia uma mensagem do tipo *Iniciar atuação* ao protocolo que deve ser ativado e aguarda uma mensagem do tipo *Atuação iniciada*, assim a troca é concluída. No caso da implementação desse protótipo, a passagem da fila é feita por referência através do recurso de implementação de uma classe *singleton* conforme descrito na Seção 4.2.1.

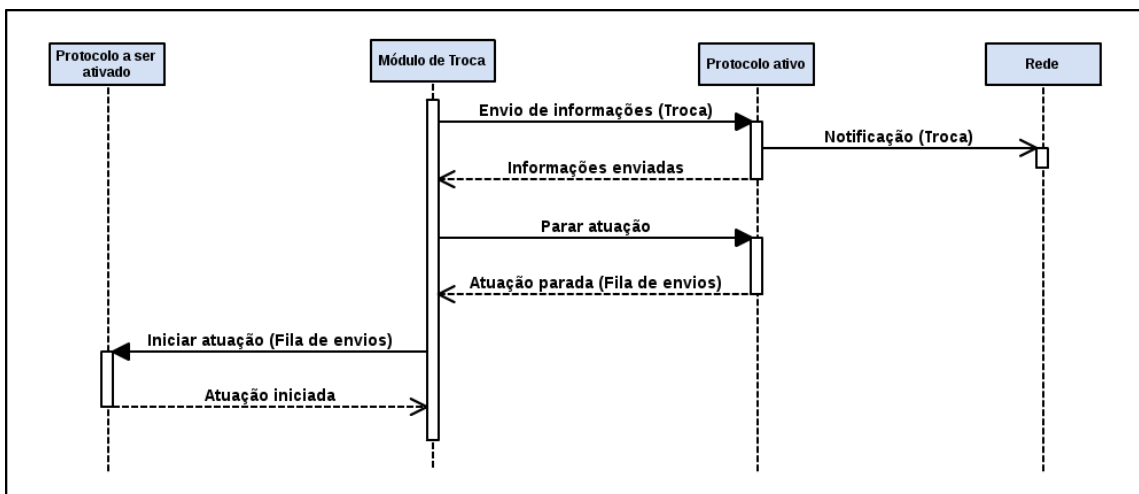


Figura 4.9. Troca de protocolo em operação na rede

Durante esse processo de troca, no intervalo de tempo entre a mensagem de *Parar a atuação* enviada ao protocolo ativo e a mensagem de *Atuação ativada* recebida do novo protocolo, o módulo de Troca filtra as mensagens que chegam das camadas de Rede e Física para que não sejam enviadas aos protocolos, pois estes estão envolvidos no processo de troca.

Após a troca, há a possibilidade de algum dos nós não ter recebido as mensagens de aviso para trocar, então, é responsabilidade do módulo de Troca do nó Coordenador FS-MAC enviar periodicamente uma mensagem informando que protocolo está ativo no momento. Esse mecanismo procura minimizar o tempo em que dois protocolos estejam em atuação ao mesmo tempo, pois esse evento degrada a eficiência da rede. A frequência de envio dessa mensagem também foi escolhida empiricamente, a mensagem é enviada a cada 5 segundos.



# Capítulo 5

## Avaliação experimental

Neste capítulo apresentamos a caracterização dos protocolos CSMA e TDMA separadamente e a avaliação da plataforma FS-MAC. A caracterização foi necessária para a elaboração das regras de inferência da plataforma FS-MAC, por isso, analisamos esses protocolos de forma individual a fim de verificar as condições da rede onde cada um melhor se adapta. Após essa caracterização, com a implementação da plataforma já concluída, avaliamos o seu comportamento quanto à capacidade de se adaptar às condições da rede e também, em termos de desempenho, como ela se sai em relação aos protocolos quando operam de forma isolada.

O restante do capítulo está organizado da seguinte forma: a seção 5.1 define alguns termos que serão utilizados na avaliação; a seção 5.2 apresenta os aspectos gerais de metodologia utilizados em todos os experimentos; a seção 5.3 traz a avaliação dos protocolos CSMA e TDMA separadamente e a seção 5.4 mostra a avaliação da plataforma FS-MAC.

### 5.1 Convenções

Durante a avaliação experimental, utilizamos alguns termos para definir de forma mais precisa e adotar como convenção. Chamamos de *Caso de uso* a definição de uma avaliação a ser realizada. Essa avaliação possui um ou mais experimentos e pode ser usada para analisar cenários semelhantes. Um exemplo de caso de uso é *Avaliação da vazão por número de transmissores*. Esse caso de uso pode ser usado para analisar redes que operam tanto com CSMA, quanto com TDMA ou FS-MAC.

Chamamos de *Experimento* um conjunto de configurações associado a um conjunto de procedimentos. As configurações dizem respeito a equipamentos e sistemas utilizados, à disposição topológica desses equipamentos e os parâmetros adotados para

a realização dos procedimentos. Já os procedimentos correspondem a um conjunto de passos realizados utilizando as configurações definidas.

## 5.2 Metodologia

Os aspectos gerais metodológicos detalhados aqui foram utilizadas em todos os experimentos dos casos de uso descritos nas seções posteriores. As variações específicas de cada caso são mencionadas nas respectivas seções, mas tendo sempre como base esses aspectos gerais.

### 5.2.1 Equipamentos e sistema

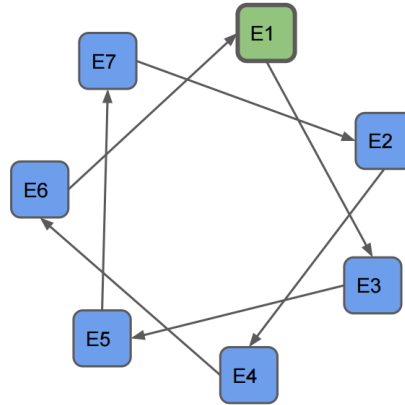
Em termos de equipamento e ambiente de sistema, foram utilizados 7 rádios USRP B210, 3 *notebooks* com processador 4th Generation Intel® Core™ i7 4500U com 8GB de memória RAM, um notebook com processador Intel® Core™ i5-3337U com 6 GB de memória RAM e 3 computadores com processador 6th Generation Intel® Core™ i7 6700T com 16 GB de memória RAM. Todos os computadores utilizam o sistema operacional Ubuntu 14.04 LTS, drive UHD 003.010 e GNU Radio 3.7.8. As configurações de memória e CPU do computadores não são limitadores de desempenho durante os experimentos, isso foi confirmado com o monitoramento do consumo desses recursos durante os testes.

### 5.2.2 Topologia

Quanto à configuração e disposição física dos equipamentos, a Figura 5.1 mostra o esquema de comunicação organizado. Nessa figura, a seta indica a comunicação entre duas estações e a direção da seta corresponde à direção em que os dados estão sendo enviados. A estação E1, em destaque, representa a estação que exerce papel de coordenação. No caso onde está em operação o protocolo TDMA isoladamente, E1 é o nó que sincroniza a comunicação e aloca os *slots* de tempo. Já no caso em que está em operação a plataforma FS-MAC, além da função de coordenador TDMA, E1 também é um Coordenador FS-MAC, ou seja, é a estação responsável por, dentre outras funções, coordenar a troca de protocolos quando necessário. Durante a atuação do CSMA, seja isolado ou na plataforma FS-MAC, E1 se comporta como um nó Comum. A configuração física de cada estação está descrita na Tabela 5.1.



Algumas USRPs foram gentilmente cedidas pelo GRUBI - Grupo de Redes Ubíquas<sup>1</sup>. Algumas USRPs e computadores também foram gentilmente cedidos pelo FUTEVOL - Federated Union of Telecommunications Research Facilities for an EU-Brazil Open Laboratory<sup>2</sup>.



**Figura 5.1.** Diagrama de topologia e comunicação dos experimentos.

**Tabela 5.1.** Equipamento utilizado nos experimentos

<i>Estação</i>	<i>USRP</i>	<i>Computador</i>
E1	B210	6th Generation Intel® Core™ i7 6700T com 16 GB de memória RAM
E2	B210	6th Generation Intel® Core™ i7 6700T com 16 GB de memória RAM
E3	B210	6th Generation Intel® Core™ i7 6700T com 16 GB de memória RAM
E4	B210	4th Generation Intel® Core™ i7 4500U com 8GB de memória RAM
E5	B210	4th Generation Intel® Core™ i7 4500U com 8GB de memória RAM
E6	B210	Intel® Core™ i5-3337U com 6 GB de memória RAM
E7	B210	4th Generation Intel® Core™ i7 4500U com 8GB de memória RAM

Essa topologia foi organizada para maximizar a capacidade de cada enlace e ao mesmo tempo proporcionar uma competição pelo meio de comunicação. Inicialmente configuramos a transmissão para ocorrer entre um nó e seu vizinho imediatamente adjacente. No entanto, notamos que a comunicação melhorava quando as USRPs que se comunicavam estavam dispostas em ângulos mais fechados, então escolhemos estabelecer a comunicação entre as USRPs sempre com o segundo vizinho adjacente, como indica a figura. A disposição circular e com distância aproximada de 50 cm entre as estações comunicantes, como mostra a Figura 5.2, foi escolhida para evitar a ocorrência de terminais expostos e terminais escondidos, problemas esses que não são objetos da nossa análise nesse trabalho.

<sup>1</sup><http://asteroide.dcc.ufla.br/~grubi/site/>

<sup>2</sup><http://www.ict-futebol.org.br/>



**Figura 5.2.** Organização das USRPs durante os experimentos.

### 5.2.3 Carga de dados e modos de operação

A carga de dados em cada estação é sempre máxima. Em cada nó, o gerador de mensagens que alimenta o protocolo, simulando a chegada de dados vindos da camada de Rede, gera uma mensagem a cada 20 ms, esta mensagem tem tamanho 110 bytes e está de acordo com o padrão 802.15.4 [LAN/MAN Standards Committee et al., 2003]. O intervalo de 20 ms garante a geração de 50 mensagens por segundo, uma carga sempre maior do que cada nó consegue enviar no melhor caso. Dessa forma, o fluxo de dados na comunicação é contínuo e nenhum nó se mantém ocioso por falta de dados a serem enviados.

Durante a execução dos experimentos dos casos de uso, em termos de comunicação, cada estação pode estar operando em três diferentes modos: recebimento (R), transmissão e recebimento (TR), inatividade (I). No modo (R), o nó não está transmitindo dados, apenas recebe dados de uma outra estação e envia confirmação de recebimento. No modo (TR), a estação transmite dados a outra, recebe confirmação, mas também recebe dados de outras estações e envia confirmação. Já no estado (I), o nó está completamente inativo. Os nós que exercem função de coordenação estão sempre ativos para essa função, independente dos modos de operação descritos acima.

### 5.2.4 Parâmetros

Os valores dos parâmetros do protocolo CSMA implementado neste trabalho e utilizados nos experimentos, bem como a proporcionalidade entre eles foram baseados no trabalho [Puschmann et al., 2012] e no padrão [IEEE 802.11 Working Group et al., 2010] e estão dispostos na Tabela 5.2. O valor de *ACK timeout*, foi inicialmente utilizado como 80 ms, no entanto, após alguns experimentos ajustamos esse valor para

40 ms. Nas USRPs foi utilizado o valor de potência de 60 dB, a frequência central de 2.48 GHz, com a largura de banda padrão do equipamento para o canal. A modulação escolhida foi a OQPSK. A montagem dos blocos no GNU Radio pode ser vista na Figura 5.3.

**Tabela 5.2.** Parâmetros CSMA

<i>Parâmetro</i>	<i>Valor (ms)</i>
$CW_{min}$	4
$CW_{max}$	64
DIFS	5
SIFS	2
<i>Slot time</i>	2
<i>ACK timeout</i>	40

Os valores dos parâmetros do protocolo TDMA implementado neste trabalho foram adquiridos empiricamente e estão dispostos na Tabela 5.3 .

**Tabela 5.3.** Parâmetros TDMA

<i>Parâmetro</i>	<i>Valor (ms)</i>
<i>Slot de Ack</i>	2
<i>Slot de dados</i>	4
<i>Slot de alocação</i>	2
<i>Tempo de guarda</i>	2

### 5.2.5 Dinâmica dos experimentos

A dinâmica de transmissões descrita aqui é utilizada completa ou em partes em todos os casos de uso. Todos os experimentos dos casos de uso descritos nas seção a seguir possuem muitas semelhanças quanto às configurações, os procedimentos adotados e também quanto à avaliação. Por isso, para referência nessas seções descrevemos aqui esses aspectos.

Chamaremos de *Experimento com inclusão gradativa de transmissores* o experimento que utiliza as configurações gerais definidas nas seções anteriores e que utiliza o seguinte procedimento: na primeira etapa há apenas um transmissor e em cada etapa posterior mais uma estação passa a transmitir dados até que todas elas estejam transmitindo. Tomando como referências as Figuras 5.1 e 5.4, o experimento desse tipo se inicia com a estação E1 em modo de transmissão (TR), a estação E3 em modo de recebimento (R) e todas as outras em modo de inatividade (I), ou seja, na primeira

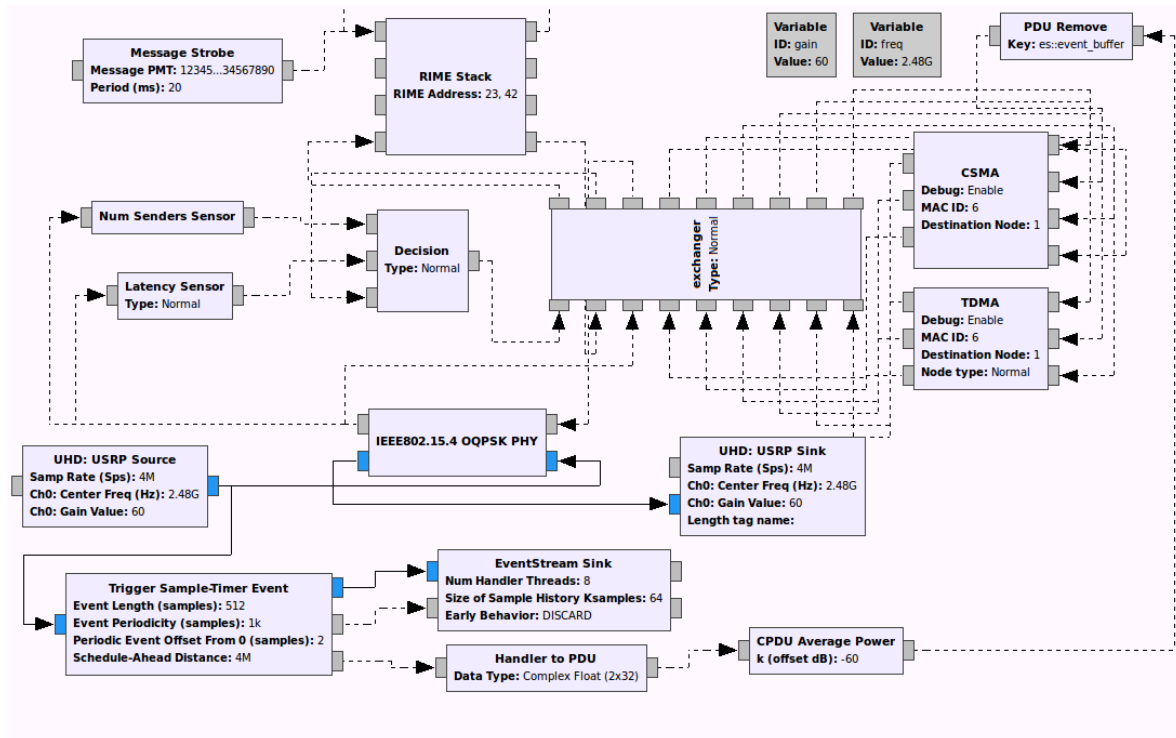


Figura 5.3. Montagem dos blocos no GNU Radio para um nó Comum.

etapa há apenas um transmissor. Na segunda etapa, a estação E3 passa do modo (R) para o modo (TR) e a estação E5 passa do modo (I) para o modo (R), assim, nessa etapa temos dois transmissores. Nas etapas seguintes, as estações E7, E2, E4 e E6, nessa ordem, passam do modo (I) para (R) e em seguida, do modo (R) para (TR) de forma que na última etapa todas as estações operam em modo (TR).

Chamaremos de *Etapa do experimento com inclusão gradativa de transmissores*, o período de transmissão em que o número de transmissores se mantém fixo. Em todos os experimentos, cada etapa dura 180 segundos, dos quais 60 são reservados para a estabilização do nó, instanciação de objetos, ativação dos blocos e formação da fila e durante os outros 120 segundo as métricas de avaliação são coletadas. Conforme definido na seção anterior, a carga de dados preparados para envio é a mesma para todos os nós. No entanto, a efetiva taxa de transmissão de cada estação depende de que protocolo está em operação e da própria dinâmica desse protocolo durante a tarefa de controle de acesso ao meio. Essa taxa varia de uma etapa para outra, pois com o acréscimo dos nós no modo de operação (TR), aumenta-se a quantidade de tentativas de transmissão e conseqüentemente a contenção da rede.

Em todos os gráficos, as barras verticais em cada ponto correspondem ao intervalo de confiança da média, calculado utilizando a distribuição *t de Student* para um nível



**Figura 5.4.** Etapas do Experimento com inclusão gradativa de transmissores

de confiança de 95%. Em todos os gráficos de linha, cada ponto corresponde à média da grandeza apresentada no eixo Y em relação à quantidade de estações transmitindo, apresentada no eixo X.

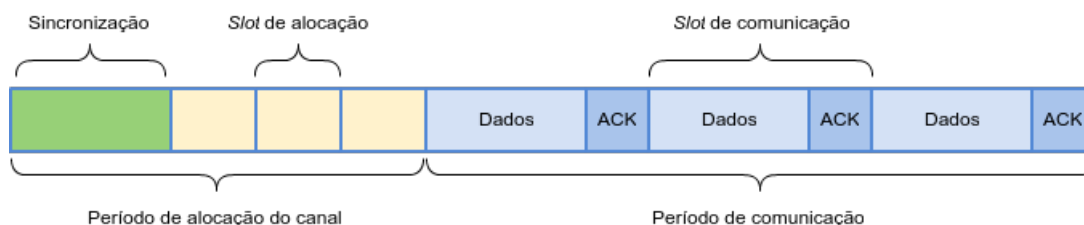
## 5.3 Avaliação e caracterização dos protocolos

Antes de utilizar os protocolos implementados integrados na plataforma FS-MAC foi preciso realizar duas tarefas. 1 - verificar se esses protocolos possuem comportamento compatível com as implementações de CSMA e TDMA descritas na literatura; 2 - Analisar em que situações da rede cada um é mais eficiente. Para proceder essas verificações foi implementado o caso de uso descrito a seguir.

### 5.3.1 Vazão em relação à variação do número de transmissores (CSMA e TDMA)

Esse caso de uso tem como primeiro objetivo avaliar o comportamento dos protocolos CSMA e TDMA e compará-los com outras implementações já descritas na literatura. O segundo objetivo é verificar em que momento um protocolo é mais eficiente que o outro e quais são as condições da rede nesses momentos. Essa verificação serviu de base para definir as regras utilizadas no módulo de decisão da plataforma FS-MAC.

O protocolo CSMA combina uma série de temporizações e verificação do meio para tentar evitar colisões. Espera-se deste protocolo que quando a competição pelo canal seja pequena, poucas colisões ocorram, conseqüentemente, poucos reenvios de pacotes e uma vazão mais favorável. Já o TDMA funciona alocando *slots* de comunicação para cada transmissor, evitando praticamente qualquer colisão. A alocação dos *slots* acontece de acordo com o superquadro mostrado na Figura 5.5. Espera-se do TDMA que quando o canal esteja muito concorrido, a vazão seja mais alta, pois não haverá colisões ou reenvios de mensagens. Por outro lado, quando o canal for pouco concorrido, espera-se que as mensagens de controle utilizadas para alocação dos canais naturalmente prejudiquem a eficiência da rede.

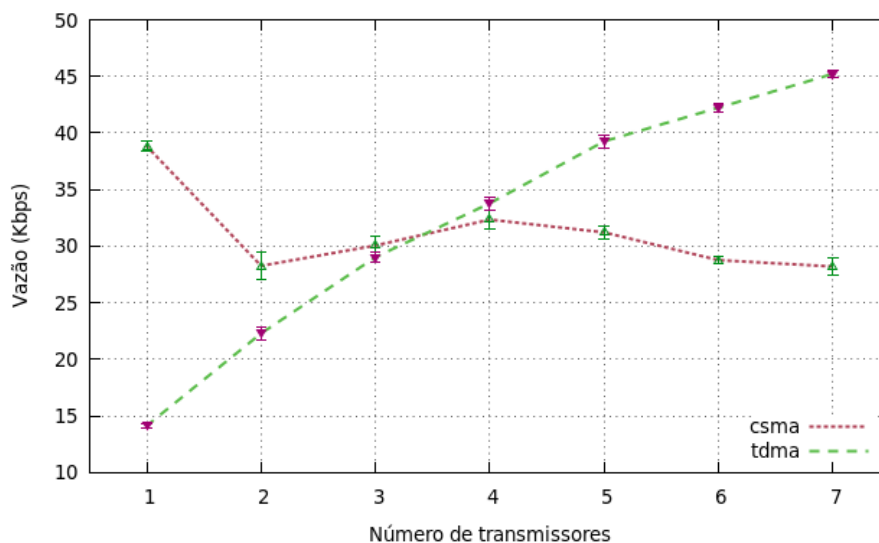


**Figura 5.5.** Superquadro do TDMA

O experimento utilizado nesse caso de uso foi o *Experimento com inclusão gradativa de transmissores* com sete *Etapas* conforme definido na seção 5.2.5. O experimento foi realizado integralmente uma vez com o TDMA e uma vez com o CSMA operando na subcamada MAC.

Cada etapa do experimento foi repetida cinco vezes e foram coletados os valores da quantidade de pacotes enviados, quantidade de pacotes confirmados, quantidade de retransmissões e a latência da entrega de cada pacote. Utilizando os valores de quantidade de pacotes confirmados, o tamanho de cada pacote e o tempo de transmissão, foram calculadas as médias de vazão de cada etapa. Os resultados referentes à vazão são mostrados no gráfico da Figuras 5.6. Através da quantidade de pacotes confirmados e retransmissões foram calculadas as médias de taxa de entrega em cada etapa, que são mostradas no gráfico da Figura 5.7. Com os valores de retransmissões de pacotes foram calculadas as médias em cada etapa e construído o gráfico da Figura 5.8. E finalmente, utilizando os valores de latência, foram calculadas as médias em cada etapa e plotado o gráfico mostrado na Figura 5.9. Em todos os gráficos gerados foram utilizadas as definições estabelecidas na seção 5.2.5.

A Figura 5.6 apresenta a vazão dos protocolos avaliados. Os resultados observados mostram que os protocolos CSMA e TDMA separadamente possuem comportamento compatível com o esperado. Conforme observado em trabalhos como Takagi & Klein-



**Figura 5.6.** Vazão total da rede por número de transmissores

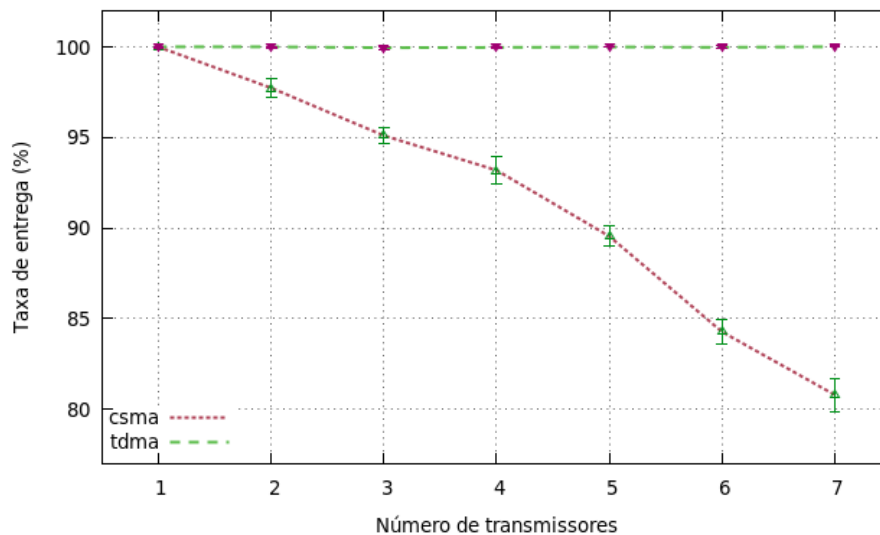
rock [1985], Ziouva & Antonakopoulos [2002] e Hu et al. [2011], o CSMA possui melhor desempenho quando a contenção é baixa, apresenta um ganho durante um intervalo de aumento de contenção, mas tem seu desempenho degradado à medida em que a contenção aumenta muito. Por outro lado, o TDMA possui baixo desempenho em cenários de baixa contenção, mas melhora à medida em que a contenção aumenta. Embora o comportamento dos protocolos em relação à contenção da rede esteja compatível com o esperado, notamos que os valores da vazão para os dois casos estão significativamente abaixo dos valores teóricos previstos para o padrão. Acreditamos que esta diferença se deve ao caráter experimental do equipamento utilizado.

Notamos que durante a avaliação do CSMA existe uma queda de vazão considerável na passagem de um para dois transmissores, e esse é o único comportamento que destoa dos estudos citados. Para encontrar a explicação para esse comportamento, temos em mente que a introdução de um transmissor na rede provoca dois efeitos imediatos, o primeiro é o aumento de dados sendo transmitidos, o que isoladamente contribui para o aumento da vazão, o segundo é a o aumento da probabilidade de colisões, o que contribui para a diminuição da vazão. Assim, combinação desses dois fatores determina se ao final a presença de um novo transmissor aumenta ou diminui a vazão da rede.

A evolução desses dois fatores na dinâmica dos experimentos possui crescimentos que obedecem a funções distintas, dessa forma, em algum momento, a quantidade de transmissores se torna grande o suficiente para que a probabilidade de colisões seja um fator mais relevante do que a quantidade de dados lançados na rede, ou seja, a

partir de uma certa quantidade de transmissores, introduzir mais estações na rede provocará uma queda na vazão. Na nossa implementação do CSMA, acreditamos que esse comportamento acontece duas vezes. A primeira, já esperada, acontece quando a partir de quatro transmissores a curva de vazão adquire uma trajetória descendente, e a segunda, não esperada, quando o sistema passa do cenário de inexistência de colisões à presença de colisões, exatamente na passagem de um para dois transmissores.

Apesar dessa diferença notada em nossos experimentos, o comportamento geral do CSMA quando comparado com o TDMA é compatível com o esperado. O experimento nos mostra também que a partir de quatro transmissores, a rede apresenta um estado em que o TDMA possui vazão maior que o CSMA, portanto é nesse ponto que um deve ser substituído pelo outro.



**Figura 5.7.** Taxa de entrega média dos pacotes por número de transmissores

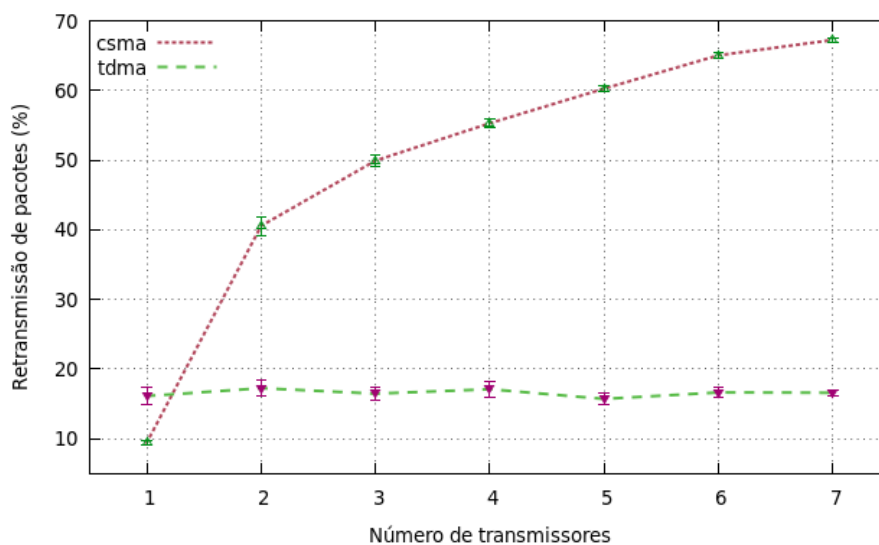
Além da informação de vazão, outras métricas foram aferidas para validar os protocolos, uma delas é a taxa de entrega dos pacotes. Analisando o gráfico da Figura 5.7 percebemos um comportamento também compatível com o esperado. A taxa de entrega do TDMA é sempre alta já que todo pacote de dados é enviado dentro de um *slot* de tempo alocado exclusivamente, assim não há perda por colisões, ou seja, embora em algumas situações a vazão seja baixa por causa do *overhead* da alocação do tempo, praticamente todos os pacotes enviados são entregues, independente da quantidade de transmissores. Por outro lado, o CSMA possui taxas altas de entrega quando não há contenção na rede, mas conforme esperado, à medida em que o número de colisões aumenta, a taxa de entrega tende a cair.

Outra métrica que se relaciona com esse fato é a retransmissão de pacotes. No



TDMA a taxa de retransmissão tende a se manter baixa e constante, enquanto no CSMA essa taxa tende a aumentar à medida em que aumentam as colisões. Os valores de taxa média de retransmissão dos pacotes estão no gráfico da Figura 5.8. Acreditamos que além das colisões, a perda de pacotes no nosso experimento é causada por interferências externas que não controlamos e no caso do TDMA por pequenos problemas de sincronização.

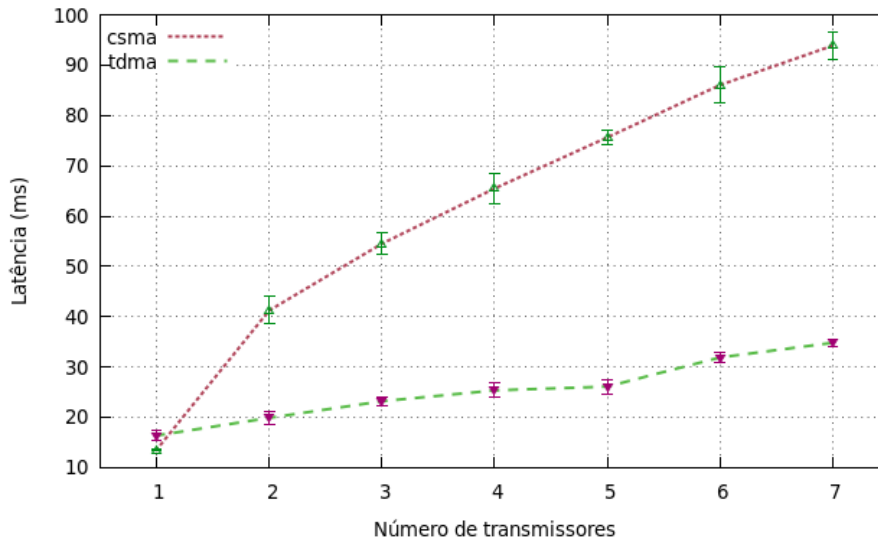
Conforme descrito no Apêndice A, a sincronização no TDMA é feita por meio de envio de pacotes de controle e com a introdução de períodos de guarda para cobrir a variação no tempo de envio e recebimento desses pacotes. Se esse tempo de guarda cobrisse toda essa variação ele seria muito grande e o protocolo seria completamente inviável. Assim, o tempo de guarda cobre a maior parte, mas não toda essa variação. Portanto, em alguns momentos, haverá colisões entre pacotes ack e pacotes de controle, mais precisamente os pacotes de sincronização. Isso pode ser percebido no gráfico da Figura 5.8 se observarmos as retransmissões para a situação onde há apenas um transmissor. Nesse caso, o CSMA possui taxa menor que o TDMA, pois além de estar livre de colisões, tem como empecilho para comunicação apenas as interferências externas.



**Figura 5.8.** Taxa de retransmissão média por número de transmissores

Os valores de latência também são coerentes com o esperado. No TDMA as latências tendem a ser mais baixas e aumentam discretamente, pois há poucos reenvios de pacotes e esses reenvios acontecem em intervalos menores de tempo, condicionados ao tamanho do superquadro apresentado na Figura 5.5 com os valores da Tabela 5.3. Por outro lado, no CSMA as latências aumentam muito à medida em que o aumento

de colisões requerem o reenvio de pacotes. Isso acontece porque cada reenvio acontece depois da espera pelo tempo de *timeout*, que é de 40 ms e o reenvio seguinte espera ainda um tempo de *backoff* que é potencialmente maior que o anterior. As informações de latência do experimento podem ser vistas no gráfico da Figura 5.9



**Figura 5.9.** Latência média de entrega dos pacotes por número de transmissores

## 5.4 Avaliação da plataforma FS-MAC

A partir das informações coletadas na avaliação dos protocolos separadamente, o sistema de decisão da plataforma FS-MAC foi construído, conforme descrito na seção 4.2.3. A partir de então implementamos alguns casos de uso para avaliar o funcionamento da plataforma. Esses casos de uso buscaram responder as seguintes questões:

1. A plataforma FS-MAC é capaz de reconhecer o estado da rede e colocar em operação o protocolo mais eficiente para aquele cenário?
2. A plataforma FS-MAC é capaz de trocar os protocolos automaticamente quando as condições da rede são alteradas?
3. O *overhead* das operações que garantem o funcionamento da plataforma prejudicam significativamente a eficiência da rede?
4. A plataforma apresenta flexibilidade para alteração das regras que definem que protocolo é mais adequado?

5. Qual o impacto no desempenho da rede se a plataforma proceder várias trocas de protocolo sucessivas?

Os três casos de uso estão descritos nas seções a seguir.

#### 5.4.1 Vazão em relação à variação do número de transmissores (FS-MAC)

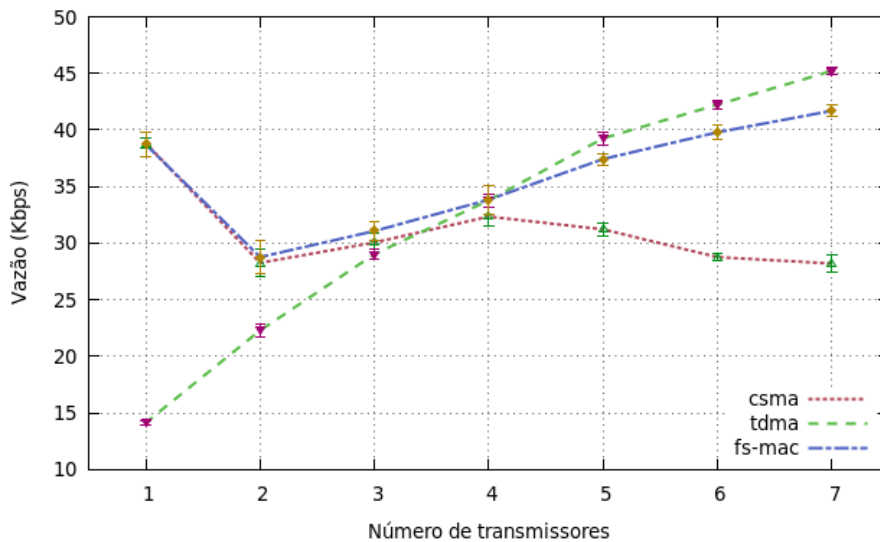
O objetivo deste caso de uso é verificar se o FS-MAC é capaz de identificar os cenários em que os protocolos são mais eficientes e se a troca entre protocolos ocorre automaticamente. O experimento utilizado nesse caso de uso também foi o *Experimento com inclusão gradativa de transmissores* com sete *Etapas* conforme definido na seção 5.2.5. Foram utilizadas as configurações gerais definidas na seção 5.2 e o experimento foi realizado integralmente uma vez com o FS-MAC operando na subcamada MAC.

Como no caso de uso de avaliação dos protocolos, cada etapa do experimento foi repetida cinco vezes e foram coletados os valores da quantidade de pacotes enviados, quantidade de pacotes confirmados, quantidade de retransmissões e a latência da entrega de cada pacote. Todos os gráficos gerados incluem os resultados do FS-MAC juntamente com os resultados relativos ao TDMA e CSMA e em todos eles foram utilizadas as definições e os recursos estatísticos estabelecidas na seção 5.2.5.

Utilizando os valores de quantidade de pacotes confirmados, o tamanho de cada pacote e o tempo de transmissão, foram calculadas as médias de vazão de cada etapa. Os resultados referentes à vazão são mostrados no gráfico da Figuras 5.10. Através da quantidade de pacotes confirmados e retransmissões foram calculadas as médias de taxa de entrega em cada etapa, que são mostradas no gráfico da Figura 5.11. Com os valores de retransmissões de pacotes foram calculadas as médias em cada etapa e construído o gráfico da Figura 5.8. E finalmente, utilizando os valores de latência, foram calculadas as médias em cada etapa e plotado o gráfico mostrado na Figura 5.13.

Analisando a Figura 5.10 podemos notar que o FS-MAC tende a seguir o protocolo que possui melhor desempenho em cada cenário. No início do experimento, onde a contenção é baixa, os desempenhos do CSMA e do FS-MAC são muito próximos. Isso pode ser observado também nos gráficos das outras métricas avaliadas, como veremos a seguir.

Analisando ainda o gráfico da Figura 5.10, até quatro transmissores, considerando os intervalos de confiança da média, o desempenho do FS-MAC é praticamente idêntico ao desempenho do CSMA. Notamos que com 4 transmissores, o desempenho do TDMA já é superior ao TDMA, no entanto, o protocolo em operação continua sendo o CSMA.



**Figura 5.10.** Vazão total da rede por número de transmissores

Isso acontece devido à proximidade do desempenho entre os dois protocolos, nesse caso o FS-MAC precisa que haja uma diferença significativa de desempenho que justifique a troca. Até o cenário com quatro transmissores, o FS-MAC está operando com CSMA, a partir do cenário com cinco nós transmitindo, a contenção da rede aumenta, a diferença de desempenho se torna significativa e assim o FS-MAC troca o protocolo para o TDMA.

**Tabela 5.4.** Média de pacotes confirmados em relação a pacotes de controle

<i>Etapa</i>	<i>Protocolo</i>	<i>Confirmados</i>	<i>Erro</i>	<i>Controle</i>	<i>Erro</i>	<i>Overhead (%)</i>
1	CSMA	5281,20	73,37	49,00	0,64	1,39
2	CSMA	3924,00	100,55	73,00	0,00	2,56
3	CSMA	4237,80	56,28	96,80	0,41	1,33
4	CSMA	4615,40	87,54	121,20	1,50	1,90
5	TDMA	5105,20	35,10	144,00	0,00	0,69
6	TDMA	5427,60	45,70	167,80	0,41	0,84
7	TDMA	5686,60	35,54	192,00	0,00	0,63

Na operação com o TDMA, observamos uma pequena perda de desempenho da plataforma FS-MAC. Inicialmente levantamos a hipótese de que essa perda se devia às mensagens de controle, pois embora não tenham grande relevância quando há poucos nós transmitindo, aumentam linearmente à medida em que os nós são inseridos na rede. No entanto, observando a Tabela 5.4, notamos que o impacto das mensagens de controle é pequeno. Essa tabela mostra na terceira coluna, a média da quantidade de pacotes confirmados durante todo o tempo do experimento. Mostra também na

quinta coluna, a média da quantidade de pacotes de controle FS-MAC enviados na rede durante o mesmo período. Na última coluna temos o percentual representado pelos pacotes de controle em relação aos pacotes confirmados. Em todos os casos, o percentual é menor que 3%, o que inviabiliza a hipótese de que essas mensagens sejam as responsáveis pela queda de desempenho.

Acreditamos que essa queda se deve principalmente às imprecisões relacionadas à sincronização no protocolo TDMA. Como a sincronização é realizada através de envio de mensagens, à medida em que o número de transmissores aumenta, essa sincronização perde qualidade e pode prejudicar o desempenho da rede. Isso tem um impacto maior na plataforma FS-MAC, pois além dos pacotes de controle do próprio TDMA, os pacotes do FS-MAC podem acabar colidindo devido a essas imprecisões. Apesar do *overhead*, o desempenho do FS-MAC mantém uma distância relativamente pequena do melhor protocolo para cada cenário.

Analizamos também o rendimento da rede utilizando cada uma das abordagens. A Tabela 5.5 mostra a quantidade total de pacotes confirmados em cada caso durante todo o período do experimento. Notamos que o rendimento do TDMA é ligeiramente superior ao do CSMA, este comportamento é esperado, pois o TDMA possui melhor desempenho no cenário onde há mais estações transmitindo.

**Tabela 5.5.** Total de pacotes confirmados e rendimento em relação ao FS-MAC

<i>Abordagem</i>	<i>Pacotes confirmados</i>	<i>Rendimento</i>
CSMA	148.427	86,60%
TDMA	154.076	89,89%
FS-MAC	171.389	100,0%

A tabela traz também o percentual de rendimento do CSMA e TDMA em relação ao FS-MAC. No cenário do experimento, onde a rede sofre variação considerável de contenção, a plataforma FS-MAC, colocando cada protocolo em operação nos momentos adequados, proporcionou à rede um rendimento cerca de 10,1% superior ao TDMA e 13,4% superior ao CSMA

Em relação às métricas de taxa de entrega, retransmissão de pacotes e latência, mostradas respectivamente nas nas Figuras 5.11, 5.12 e 5.13, como esperado, o desempenho do FS-MAC segue o desempenho do protocolo mais adaptado para o cenário.

Percebemos com esse experimento que a plataforma foi capaz de identificar e colocar em operação o protocolo mais adequado para a rede, dadas as suas condições. Nos cenários onde é necessário haver uma troca entre os protocolos, essa troca ocorre

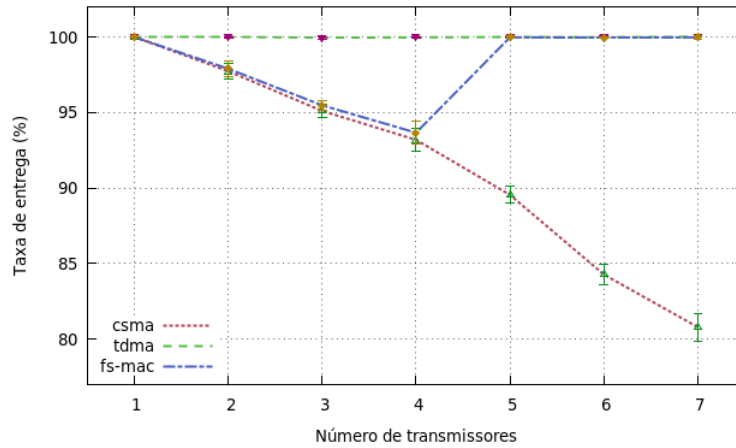


Figura 5.11. Taxa de entrega média dos pacotes por número de transmissores

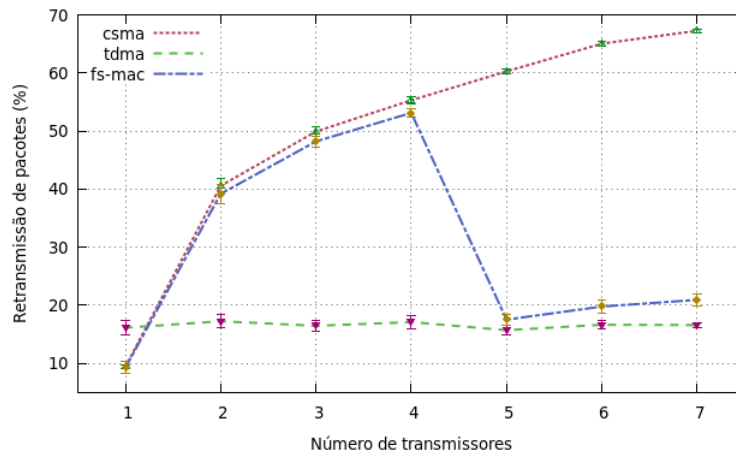


Figura 5.12. Taxa de retransmissão média por número de transmissores

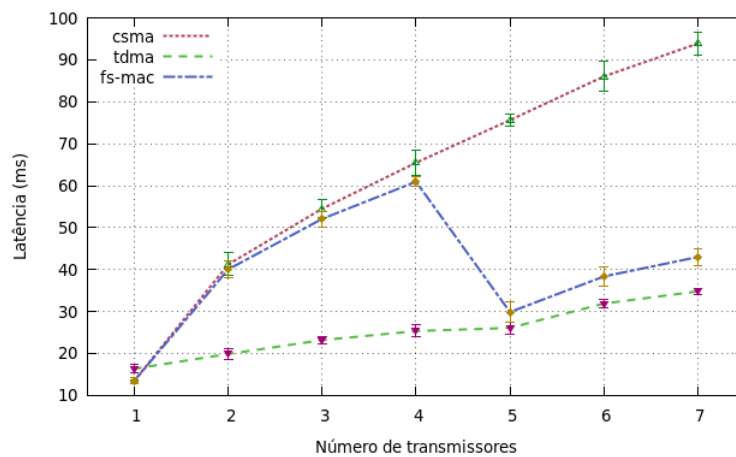


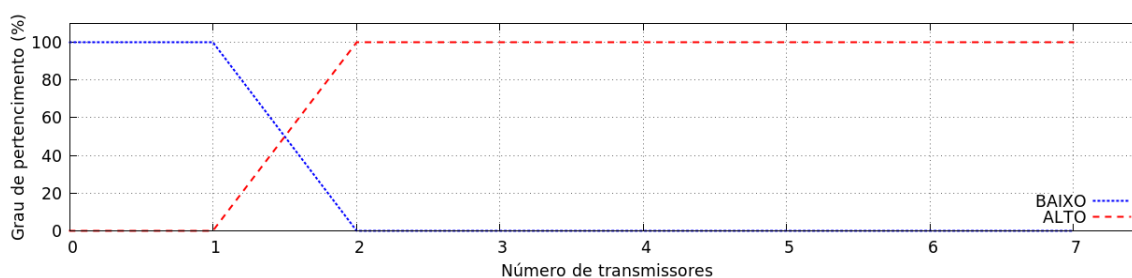
Figura 5.13. Latência média de entrega dos pacotes por número de transmissores

automaticamente. Vimos também que o desempenho dos protocolos operando dentro da plataforma FS-MAC é bastante similar ao seu desempenho operando isoladamente, ou seja, o *overhead* do controle exercido pela plataforma não é suficientemente significativo a ponto de inviabilizar o seu uso.

### 5.4.2 Flexibilização das regras FS-MAC para priorizar a menor taxa de retransmissão de pacotes

A proposta da plataforma é flexibilizar não só a inclusão de protocolos diferentes em operação na camada MAC, mas também as regras que definem quando cada protocolo entrará em operação. Observando a Figura 5.8, notamos que para o cenário de um transmissor, a taxa de retransmissão de pacotes do protocolo CSMA é menor, mas a partir de dois transmissores, essa taxa é menor para o protocolo TDMA. Assim, em contextos onde deseja-se que a retransmissão de pacotes seja mínima, a plataforma FS-MAC poderia ser utilizada para colocar em operação o protocolo mais eficiente em cada cenário tendo em vista essa métrica.

O objetivo desse caso de uso é verificar na plataforma FS-MAC a flexibilidade e efetividade dos resultados quanto à adaptação dessas regras. Para os experimentos, as funções de pertinência do módulo de decisão foram alteradas para colocar em operação o protocolo com menor taxa de retransmissão de pacotes em cada cenário. Essa funções estão ilustradas nos gráficos das Figuras 5.14, 5.15, 5.16 e 5.17.



**Figura 5.14.** Função de pertinência para número de transmissores.

Alteramos também as regras de inferência para se adaptarem ao novo objetivo. As novas regras estão descritas abaixo:

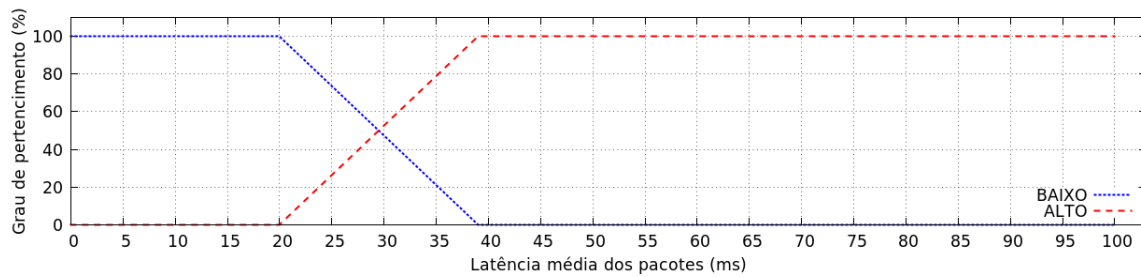
CSMA

Se NT é BAIXO e LM é BAIXO então ADP é ALTO

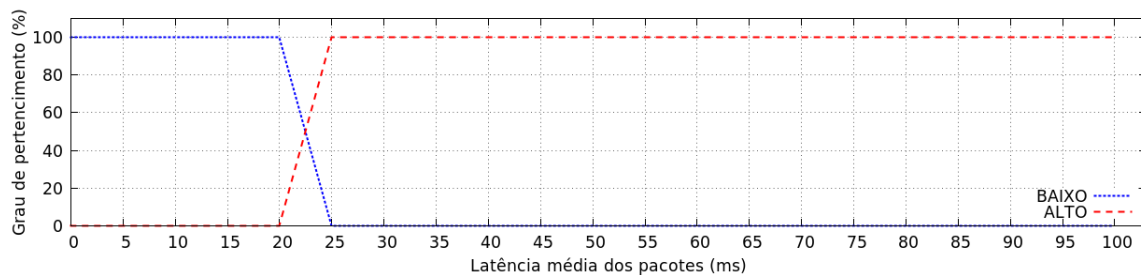
Se NT é BAIXO e LM é ALTO então ADP é BAIXO

Se NT é ALTO e LM ALTO então ADP é BAIXO

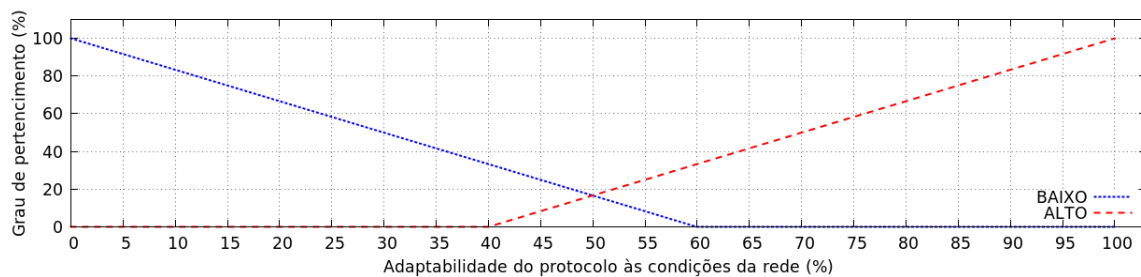
Se NT é ALTO e LM é BAIXO então ADP é BAIXO



**Figura 5.15.** Função de pertinência para latência média (CSMA).



**Figura 5.16.** Função de pertinência para latência média (TDMA).



**Figura 5.17.** Função de pertinência para adaptabilidade dos protocolos.

TDMA

Se NT é BAIXO e LM é ALTO então ADP é ALTO

Se NT é BAIXO e LM é BAIXO então ADP é BAIXO

Se NT é ALTO e LM é ALTO então ADP é ALTO

Se NT é ALTO e LM é BAIXO então ADP é ALTO

Variáveis linguísticas: O modelo mantém as configurações anteriores considerando três variáveis linguísticas, *i*) Latência média dos pacotes (*LM*), *ii*) número de transmissores (*NT*) e *iii*) Adaptabilidade do protocolo (*ADP*). Todas elas aceitam da mesma forma os termos nebulosos *BAIXO* e *ALTO*.

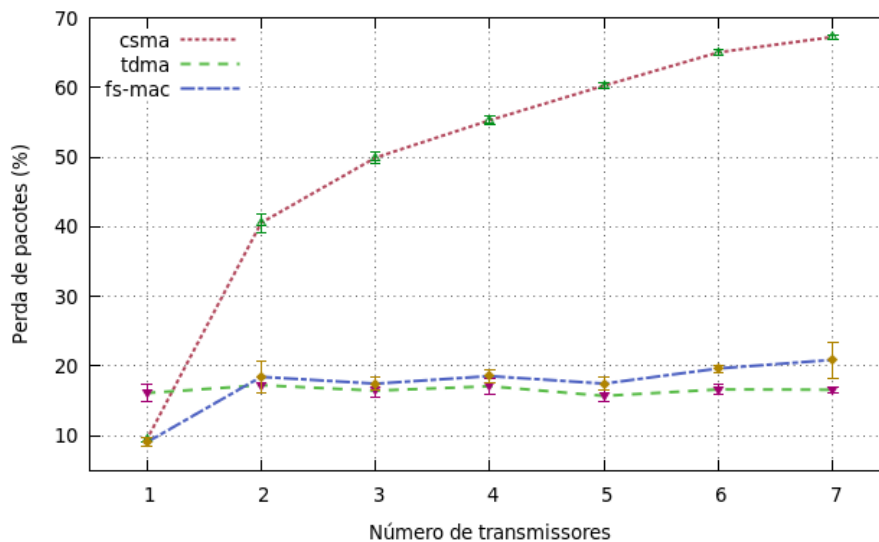
O experimento utilizado nesse caso de uso também foi o *Experimento com inclusão gradativa de transmissores* com sete *Etapas* conforme definido na seção 5.2.5. Foram



utilizadas as configurações gerais definidas na seção 5.2 e o experimento foi realizado integralmente uma vez com o FS-MAC operando na subcamada MAC.

Cada etapa do experimento foi repetida cinco vezes e foram coletados os valores de quantidade de retransmissões. Na construção do gráfico foram utilizadas as definições e os recursos estatísticos estabelecidas na seção 5.2.5. Utilizando os valores de retransmissão de pacotes, coletados durante o experimento, foram geradas as médias por etapa e construído o gráfico da Figura 5.18. Os resultados do FS-MAC foram colocados no gráfico juntamente com os resultados relativos ao TDMA e CSMA.

Como podemos notar no gráfico da Figura 5.18, a plataforma FS-MAC colocou em operação o protocolo CSMA no cenário de um transmissor. Com o acréscimo de estações transmitindo, a plataforma percebeu o aumento na taxa de retransmissão e colocou em operação o protocolo com menor taxa, no caso, o TDMA. Notamos também que os valores de retransmissão da plataforma durante a operação com o TDMA é ligeiramente superior à taxa apresentada com a operação do TDMA isolado da plataforma. Essa observação reforça a nossa ideia de que parte da perda associada ao TDMA decorre das limitações no processo de sincronização descritas nas seções anteriores, pois no caso da plataforma, além dos pacotes de controle do próprio TDMA, existem os pacotes de controle do FS-MAC.



**Figura 5.18.** FS-MAC com regras que priorizam menor taxa de perda

Concluimos com esse experimento que as regras da plataforma podem ser adaptadas para priorizar aspectos diferentes dos protocolos disponíveis. Os resultados apresentados com as novas regras são coerentes e mostram que o FS-MAC possui flexibilidade satisfatória para a alteração de regras sem perder em desempenho.

### 5.4.3 Comportamento da vazão na ocorrência de múltiplas trocas

Este caso de uso tem como objetivo avaliar o comportamento da plataforma FS-MAC no cenário de possível ocorrência de múltiplas trocas consecutivas. Esse cenário pode acontecer em casos onde a caracterização dos protocolos não foi adequada, as funções de pertinência do sistema *fuzzy* não estão coerentes com a caracterização ou quando o valor da diferença de adaptabilidades descrito na seção 4.2.4 for menor que o necessário.

No caso dos experimentos que realizamos, essa possibilidade existe apenas no caso mais crítico, ou seja, quando a troca ocorre. Assim, como nos experimentos anteriores a plataforma se comportou satisfatoriamente, foi preciso alterar artificialmente os valores de latência para simular uma caracterização ruim e forçar a ocorrência automática de várias trocas. Como no caso de uso descrito na seção 5.4 a troca do CSMA para o TDMA ocorreu na etapa com cinco transmissores, realizamos os experimentos apenas nesse caso.

O experimento para esse caso de uso utilizou as configurações gerais definidas na seção 5.2 com uma restrição quanto à topologia. As configurações de topologia se restringiram ao uso de seis estações. Baseado no esquema mostrado na figura 5.1, foram utilizadas as estações E1, E3, E5, E7 e E2 em modo (TR) e a estação E4 em modo (R). Esse arranjo corresponde à etapa com cinco transmissores do *Experimento com inclusão gradativa de transmissores*. Os parâmetros de tempo dos procedimentos são os mesmos definidos para uma *Etapa* na seção 5.2.5 e utilizados em todos os outros experimentos.

A diminuição artificial no valor das latências produz os seguintes efeitos. No cenário com cinco transmissores, mesmo com uma leve diminuição no valor das latências, quando o CSMA está em operação, a média das latências atinge um valor suficiente para que a plataforma FS-MAC troque o protocolo em operação para o TDMA. Porém, quando o TDMA entra em operação, como o valor das latências está artificialmente modificado para parecer menor, em algum momento a plataforma FS-MAC infere erroneamente que o protocolo CSMA deve ser colocado em operação. Então o CSMA é colocado em operação e todo esse processo continua se repetindo.

Durante o tempo de avaliação do experimento ocorreram entre 4 e 6 trocas em cada repetição, sendo que o tempo de operação com o TDMA foi um pouco maior em relação ao CSMA como mostra a Tabela 5.6. Foram feitas cinco repetições e foram coletados os valores de quantidade de pacotes confirmados para serem utilizados no cálculo da vazão. Os valores de vazão foram comparados com os outros cenários conforme os gráficos nas Figuras 5.19, 5.20 e 5.21. Todos os gráficos obedecem às

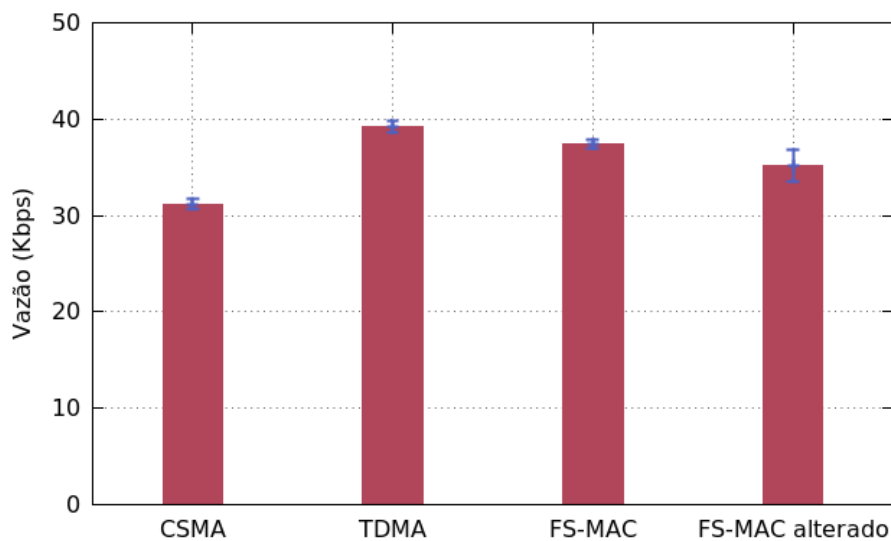
definições expressas na seção 5.2.5, sendo que no gráfico da Figuras 5.19, a altura da barra corresponde à média de vazão observada operando com cinco transmissores.

**Tabela 5.6.** Tempo de operação dos protocolos durante o experimento

<i>Protocolo</i>	<i>Tempo médio (s)</i>	<i>Erro (s)</i>
CSMA	54	8.14
TDMA	66	8.14

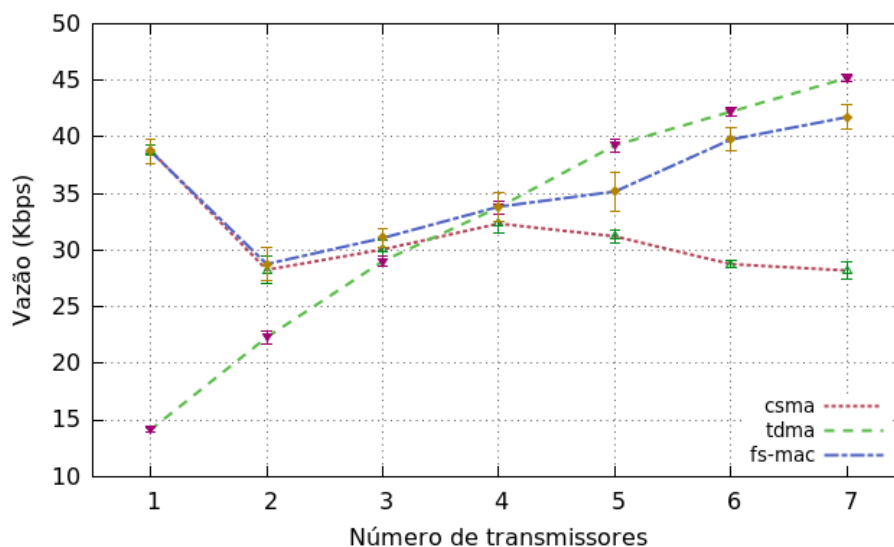
**Tabela 5.7.** *Overhead* do FS-MAC modificado em relação ao FS-MAC operando normalmente para o cenário com cinco transmissores

<i>Versão do FS-MAC</i>	<i>Confirmados</i>	<i>Erro</i>	<i>Controle</i>	<i>Erro</i>	<i>Overhead (%)</i>
FS-MAC normal	5105,20	87,54	144,00	0,00	1,71
FS-MAC modificado	4797,40	115,00	144,60	1,63	2,40



**Figura 5.19.** Impacto na vazão causado por sequência de trocas consecutivas para cinco transmissores

Observando o gráfico da Figura 5.19 notamos que, ao contrário do que esperávamos, o impacto das trocas entre protocolos é pouco significativo. O valor da vazão para o cenário de várias trocas consecutivas ficou entre o valor para TDMA e CSMA isolados e muito próximo do valor do FS-MAC operando normalmente. Percebemos isso de forma contextualizada na Figura 5.20, onde vemos também que mesmo com as trocas, os resultados se mantêm melhores que o protocolo não desejado para o cenário.

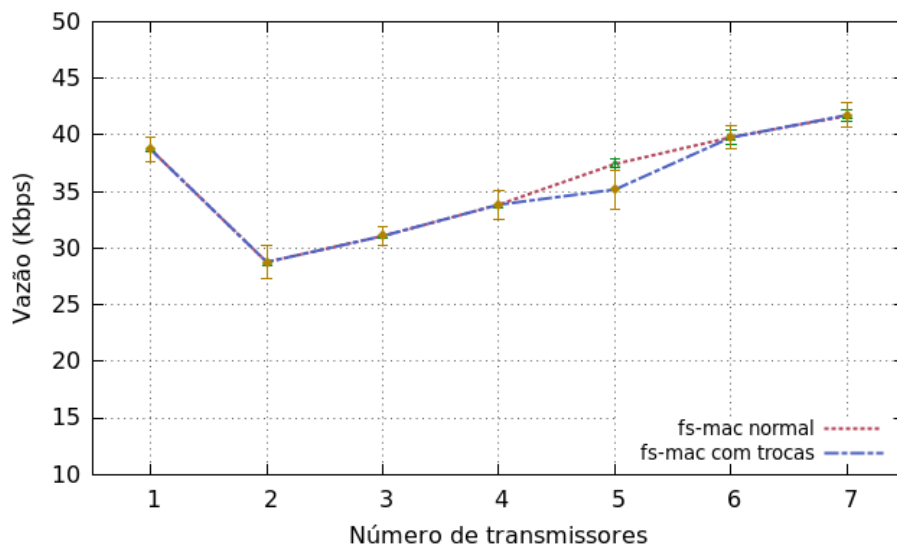


**Figura 5.20.** Desempenho do FS-MAC com múltiplas trocas em relação aos outros protocolos

Quando comparamos os resultados contextualizados com o FS-MAC operando em modo Comum, mostrados na Figura 5.21, reforçamos a ideia de que as trocas têm pouco impacto no funcionamento da plataforma. Além da perda de desempenho ser pequena, é preciso considerar que essa perda advém de dois fatores. O primeiro fator é o *overhead* da troca em si, isso inclui o tempo de sincronização e coordenação da troca de todos os nós rede. O segundo é o fato de a plataforma ficar parte do tempo operando em um protocolo com desempenho ruim para o cenário.

Observando a Tabela 5.7, notamos que conforme o esperado, as mensagens de controle não representam *overhead* nesse caso. A tabela traz as médias de pacotes confirmados e pacotes de controle medidas para o tempo total do experimento em dois cenários, utilizando o FS-MAC normal e utilizando o FS-MAC alterado para forçar as múltiplas trocas. Nos dois cenários a média de pacotes de controle enviados é muito similar. Isso acontece porque a mensagem que comunica à rede a troca do protocolo é a mesma que informaria que protocolo está em operação no momento. Assim, mesmo que haja várias trocas, o número de mensagens de controle FS-MAC irá se alterar muito pouco, não causando *overhead* adicional.

Esse experimento mostra que mesmo em cenários onde a plataforma pode apresentar problemas de sucessivas trocas de protocolo, o desempenho é razoavelmente satisfatório, pois apesar de haver perdas em relação ao melhor protocolo para o cenário, o desempenho está bem acima do protocolo não desejável.



**Figura 5.21.** Desempenho do FS-MAC com múltiplas trocas em relação ao FS-MAC normal



## Capítulo 6

# Conclusão e Trabalhos Futuros

Este trabalho apresentou a plataforma FS-MAC, uma arquitetura que provê flexibilidade para a subcamada MAC de redes sem fio. FS-MAC permite que mais de um protocolo esteja disponível para ser utilizado na subcamada MAC. Além disso, através da coleta de informações da rede e utilizando princípios de rádio cognitivo, a plataforma identifica os cenários em que cada protocolo é mais eficiente e coloca em operação automaticamente aquele que mais se adapta às condições da rede. FS-MAC é flexível não só quanto ao uso e inclusão de protocolos na subcamada, mas também quanto às regras que permitem escolher qual protocolo é mais adequado e quanto às métricas utilizadas para tomar essa decisão.

A partir da arquitetura proposta, utilizando o *framework* SDR USRP e o software de desenvolvimento GNU Radio, construímos um protótipo e avaliamos a vazão da rede com a plataforma em uma situação de troca entre um protocolo baseado em contenção (CSMA) e outro livre de contenção (TDMA). Durante os experimentos, o FS-MAC foi capaz de identificar corretamente que protocolo mais se adapta a cada cenário e colocar automaticamente esse protocolo em operação na rede. Os resultados mostraram que a plataforma possui desempenho ligeiramente inferior ao melhor caso dos protocolos isolados, apesar disso, o desempenho ficou muito próximo do melhor protocolo em cada cenário. Quando analisamos os resultados considerando todo o período do experimento, notamos que utilizando o FS-MAC a rede apresentou um rendimento cerca de 10,1% superior ao TDMA e 13,4% superior ao CSMA. O *overhead* observado, causado pelos pacotes de controle FS-MAC, ficou abaixo de 3% em todos os casos.

Avaliamos também a flexibilização das regras de decisão. Sem fazer qualquer mudança adicional, ajustamos apenas o módulo de decisão para em vez de priorizar a maior vazão da rede, promover a menor taxa de retransmissão de pacotes. Também

neste caso, o desempenho do FS-MAC ficou muito próximo do melhor protocolo em cada cenário.

Analisamos ainda o comportamento da plataforma no caso de acontecerem várias trocas sucessivas de protocolo. Nesse caso, para cinco transmissores, o protocolo que mais se adapta é o TDMA, no entanto, forçadamente, o FS-MAC operou parte do tempo com CSMA e parte com TDMA. O desempenho da plataforma dado esse cenário foi mais baixo do que quando operou normalmente, mas os resultados mostraram que esse desempenho ainda ficou acima do CSMA, o protocolo menos adequado para o cenário.

Assim, os resultados do nosso trabalho sugerem que não só é possível, mas também é viável construir uma plataforma com alta flexibilidade para a subcamada MAC de redes sem fio, que permita utilizar cada protocolo no cenário em que é mais eficiente. Em outras palavras, os experimentos realizados nos mostram que o protótipo construído funciona corretamente e o *overhead* do controle não se mostrou grande o suficiente para inviabilizar o seu uso.

Dentre as várias dificuldades encontradas durante o desenvolvimento desse trabalho duas se destacam e precisam ser mencionadas. A primeira dificuldade está relacionada à disponibilidade de implementações já existentes dos protocolos MAC em SDR. Embora existam contribuições que apresentam o desenvolvimento dos protocolos CSMA e TDMA nessa plataforma, as implementações descritas nos trabalhos não estão disponíveis para uso. Assim, antes de começar a produzir qualquer contribuição foi necessário implementar a partir do início uma versão do CSMA e do TDMA. Essa tarefa consumiu muito tempo e esforço. Outra dificuldade está relacionada à complexidade da configuração dos experimentos utilizando USRPs. Pelo caráter experimental do equipamento e pela impossibilidade de eliminar interferências externas, a correta configuração e realização dos experimentos que garantisse a qualidade dos resultados consumiu também muito tempo e esforço.

A plataforma FS-MAC é um trabalho inicial e acreditamos que as ideias colocadas aqui, bem como o protótipo implementado tenham grande potencial para serem desenvolvidos e aperfeiçoados. Como trabalhos futuros, pretendemos desenvolver as seguintes ideias:

- Analisar como incorporar o conceito de SDN no FS-MAC, de forma que um controlador defina regras para a troca de protocolos.
- Analisar outros mecanismos para a tomada de decisão, como aprendizado de máquina, que permitem que a arquitetura refine as suas escolhas com o tempo.



- Portar o protótipo para outros padrões como 802.11, 802.15.1 e 802.15.3.
- Estudar a utilização de outras métricas como ruído.
- Estudar a priorização de outros aspectos da rede como justiça.



# Referências Bibliográficas

- Akyildiz, I. F.; Lee, W.-Y.; Vuran, M. C. & Mohanty, S. (2006). "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey". *Computer Networks*, 50(13):2127 – 2159. ISSN 1389-1286.
- Bianchi, G. (2013). FLAVIA Project. Disponível em: <http://www.ict-flavia.eu/>. Acesso em: 18/09/2016.
- Blust, S. M. (1995). Software defined radio-industry request for information. *BellSouth Wireless, Inc.*
- Busch, C.; Magdon-Ismail, M.; Sivrikaya, F. & Yener, B. (2004). Contention-free MAC protocols for wireless sensor networks. Em *International Symposium on Distributed Computing*, pp. 245--259. Springer.
- Camana, P. (1988). Integrated communications, navigation, identification avionics (ICNIA)-the next generation. *IEEE Aerospace and Electronic Systems Magazine*, 3(8):23–26. ISSN 0885-8985.
- Chen, K. & Duan, R. (2011). C-RAN –the road towards green RAN. Relatório técnico, China Mobile. [http://labs.chinamobile.com/cran/wp-content/uploads/CRAN\\_white\\_paper\\_v2\\_5\\_EN.pdf](http://labs.chinamobile.com/cran/wp-content/uploads/CRAN_white_paper_v2_5_EN.pdf).
- Cho, H. H.; Lai, C. F.; Shih, T. K. & Chao, H. C. (2014). Integration of sdr and sdn for 5g. *IEEE Access*, 2:1196–1204. ISSN 2169-3536.
- Cidon, A.; Nagaraj, K.; Katti, S. & Viswanath, P. (2012). Flashback: decoupled lightweight wireless control. *SIGCOMM Comput. Commun. Rev.*, 42(4):223–234. ISSN 0146-4833.
- Cook, P. G. & Bonser, W. (1999). Architectural overview of the speakeasy system. *IEEE Journal on Selected Areas in Communications*, 17(4):650–661. ISSN 0733-8716.

- Cordeiro, J. R. S.; Lanza, E.; Macedo, D. F. & Vieira, L. F. M. (2017). FS-MAC: Uma Plataforma para a Flexibilização da Subcamada MAC em Redes Sem Fio. Em *XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 1032--1045. SBC.
- Dao, T.-K.; Nguyen, H.-L.; Pham, T.-T.; Castelli, E.; Nguyen, V.-T. & Nguyen, D.-V. (2014). User localization in complex environments by multimodal combination of GPS, WiFi, RFID, and pedometer technologies. *The Scientific World Journal*, 2014.
- Dawes, B.; Abrahams, D. & Rivera, R. (2015). Welcome to Boost.org! Disponível em: <http://www.boost.org/>. Acesso em: 18/07/2016.
- Demirkol, I.; Ersoy, C.; Alagoz, F. et al. (2006). MAC protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44(4):115--121.
- Dhar, R.; George, G.; Malani, A. & Steenkiste, P. (2006). Supporting Integrated MAC and PHY Software Development for the USRP SDR. Em *Networking Technologies for Software Defined Radio Networks, 2006. SDR '06.1st IEEE Workshop on*, pp. 68--77.
- Di Francesco, P.; McGettrick, S.; Anyanwu, U. K.; O'Sullivan, J. C.; MacKenzie, A. B. & DaSilva, L. A. (2015). A split mac approach for SDR platforms. *Computers, IEEE Transactions on*, 64(4):912--924.
- Diepstraten, W. & WCND-Utrecht, N. (1993). Ieee 802.11 wireless access method and physical specification. *Power*, 5:10.
- Doerr, C.; Neufeld, M.; Fifield, J.; Weingart, T.; Sicker, D. C. & Grunwald, D. (2005). Multimac-an adaptive mac framework for dynamic radio networking. Em *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pp. 548--555. IEEE.
- Ettus (2016). USRP X series. Disponível em: <https://www.ettus.com/product/category/USRP-X-Series>. Acesso em: 08/05/2016.
- Ettus Research (2016). USRP B210 (Board Only). Disponível em: <https://www.ettus.com/product/details/UB210-KIT/>. Acesso em: 18/09/2016.
- FCC, E. (2003). Docket no 03-222 notice of proposed rule making and order.
- Firooz, M. H.; Chen, Z.; Roy, S. & Liu, H. (2013). Wireless network coding via modified 802.11 mac/phy: Design and implementation on sdr. *IEEE Journal on Selected Areas in Communications*, 31(8):1618--1628. ISSN 0733-8716.

- Force, S. P. T. (2002). Spectrum policy task force report et docket no. 02-135. *US Federal Communications Commission*.
- Forum, W. I. (2011). Software Defined Radio - Rate of Adoption. Disponível em: <http://www.wirelessinnovation.org>. Acesso em: 11/08/2016.
- GNU Radio (2015). Welcome to GNU Radio! <http://gnuradio.org/redmine/projects/gnuradio/wiki>.
- Gonzalez, C. R. A.; Dietrich, C. B.; Sayed, S.; Volos, H. I.; Gaeddert, J. D.; Robert, P. M.; Reed, J. H. & Kragh, F. E. (2009). Open-source sca-based core framework and rapid development tools enable software-defined radio education and research. *IEEE Communications Magazine*, 47(10):48–55. ISSN 0163-6804.
- Gummalla, A. C. V. & Limb, J. O. (2000). Wireless medium access control protocols. *Communications Surveys & Tutorials, IEEE*, 3(2):2--15.
- Hu, W.; Yousefi'zadeh, H. & Li, X. (2011). Load Adaptive MAC: A Hybrid MAC Protocol for MIMO SDR MANETs. *Wireless Communications, IEEE Transactions on*, 10(11):3924–3933. ISSN 1536-1276.
- IEEE 802.11 Working Group et al. (2010). IEEE Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 6: Wireless Access in Vehicular Environments. *IEEE Std*, 802:11p.
- Koopman, P. J. & Upender, B. P. (1995). Time division multiple access without a bus master. *United Technologies Research Center, US, Technical Report RR-9500470*.
- Labib, M.; Reed, J. H.; Martone, A. F. & Zaghoul, A. I. (2016). Coexistence between radar and lte-u systems: Survey on the 5 ghz band. Em *2016 United States National Committee of URSI National Radio Science Meeting (USNC-URSI NRSM)*, pp. 1–2.
- Lackey, R. I. & Upmal, D. W. (1995). Speakeasy: the military software radio. *IEEE Communications Magazine*, 33(5):56–61. ISSN 0163-6804.
- LAN/MAN Standards Committee et al. (2003). Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). *IEEE Computer Society*.

- Lin, Y.; Lee, H.; Woh, M.; Harel, Y.; Mahlke, S.; Mudge, T.; Chakrabarti, C. & Flautner, K. (2006). Soda: A low-power architecture for software radio. *ACM SIGARCH Computer Architecture News*, 34(2):89--101.
- Mitola, J. (1992). Software radios-survey, critical evaluation and future directions. Em *Telecommunications Conference, 1992. NTC-92., National*, pp. 13/15--13/23.
- Mitola, J. (1999). Cognitive radio for flexible mobile multimedia communications. Em *Mobile Multimedia Communications, 1999. (MoMuC '99) 1999 IEEE International Workshop on*, pp. 3--10.
- Mitola, J. (2000). *Cognitive Radio — An Integrated Agent Architecture for Software Defined Radio*. Tese de doutorado, Royal Institute of Technology (KTH).
- Mitola, J.; Marshall, P.; Chen, K.-c.; Mueck, M. & Zvonar, Z. (2015). Software defined radio-20 years later: Part 1 [guest editorial]. *IEEE Communications Magazine*, 53(9):22--23.
- Myers, A. D.; Záruba, G. V. & Syrotiuk, V. R. (2002). An adaptive generalized transmission protocol for ad hoc networks. *Mob. Netw. Appl.*, 7(6):493--502. ISSN 1383-469X.
- Nagurney, L. S. (2009). Software defined radio in the electrical and computer engineering curriculum. Em *2009 39th IEEE Frontiers in Education Conference*, pp. 1--6. ISSN 0190-5848.
- Neufeld, M.; Fifield, J.; Doerr, C.; Sheth, A. & Grunwald, D. (2005). Softmac-flexible wireless research platform. Em *Proc. HotNets-IV*, pp. 1--5.
- Nychis, G.; Hottelier, T.; Yang, Z.; Seshan, S. & Steenkiste, P. (2009). Enabling MAC Protocol Implementations on Software-defined Radios. Em *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09*, pp. 91--105, Berkeley, CA, USA. USENIX Association.
- Passino, K. M.; Yurkovich, S. & Reinfrank, M. (1998). *Fuzzy control*, volume 2725. Citeseer.
- Puschmann, A.; Di Francesco, P.; Kalil, M. A.; DaSilva, L. A. & Mitschele-Thiel, A. (2013). Enhancing the performance of random access mac protocols for low-cost sdrs. Em *Proceedings of the 8th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation &#38; Characterization, WiNTECH '13*, pp. 9--16, New York, NY, USA. ACM.

- Puschmann, A.; Kalil, M. A. & Mitschele-Thiel, A. (2012). A Flexible CSMA based MAC Protocol for Software Defined Radios. *Frequenz*, 6:261–268.
- Rao, A. & Stoica, I. (2005). An overlay MAC layer for 802.11 networks. Em *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pp. 135–148. ACM.
- Rawat, P.; Singh, K. D.; Chaouchi, H. & Bonnin, J. M. (2014). Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68(1):1–48. ISSN 1573-0484.
- Reis, A.; Barros, A.; Gusso Lenzi, K.; Pedroso Meloni, L. & Barbin, S. (2012). Introduction to the software-defined radio approach. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 10(1):1156–1161. ISSN 1548-0992.
- Rhee, I.; Warrier, A.; Aia, M.; Min, J. & Sichitiu, M. L. (2008). Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 16(3):511–524.
- Schmid, T.; Bloessl, B. & Wunsch, F. (2016). An IEEE 802.15.4 transceiver for GNU Radio v3.7. Disponível em: <https://github.com/bastibl/gr-ieee802-15-4>. Acesso em: 08/05/2016.
- Sharp, B.; Grindrod, E. & Camm, D. (1995). Hybrid TDMA/CSMA protocol for self managing packet radio networks. Em *Universal Personal Communications. 1995. Record., 1995 Fourth IEEE International Conference on*, pp. 929–933.
- Silva, W. S.; Cordeiro, J. R. S.; Macedo, D. F.; Vieira, M. A. M.; Vieira, L. F. M. & Nogueira, J. M. S. (2015). *Minicursos / XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, capítulo Introdução a Rádios Definidos por Software com aplicações em GNU Radio. SBC, Vitória - ES.
- Snyder, J.; McNair, B.; Edwards, S. & Dietrich, C. (2011). Ossie: An open source software defined radio platform for education and research. Em *International conference on frontiers in education: computer science and computer engineering (FECS'11). World congress in computer science, Computer engineering and applied computing. Las Vegas, NV*.
- Souza, A. D.; Marques, A. F. F.; Macedo, D. F.; Collins, D.; Júnior, G. M.; Cordeiro, J. R. S.; Marquez-Barja, J. M.; Nacif, J. A. M.; Coelho, K. K.; Pinto, L. R. M.; da Silva, L. A.; Luiz F. M. Vieira (UFMG), L. H. A. C.; Vieira, M. A. M.; Alvarez,

- P. & Silva, W. S. (2017). *Livro Texto Minicursos / XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, capítulo Experimental Wireless Networking Research using Software-Defined Radios. SBC, Belém - PA.
- Takagi, H. & Kleinrock, L. (1985). Throughput Analysis for Persistent CSMA Systems. *IEEE Transactions on Communications*, 33(7):627–638. ISSN 0090-6778.
- Tan, K.; Liu, H.; Zhang, J.; Zhang, Y.; Fang, J. & Voelker, G. M. (2011). Sora: high-performance software radio using general-purpose multi-core processors. *Communications of the ACM*, 54(1):99–107.
- Tanenbaum, A. S. & Wetherall, D. (2011). *Redes de Computadores, 5ª edição*. Pearson. ISBN 9788576059240.
- Tavares, F. M. L.; Tonelli, O.; Berardinelli, G.; Cattoni, A. F. & Mogensen, P. (2012). ASGARD: the Aalborg University cognitive radio software platform for DSA experimentation. Em *3rd International Workshop of the COST Action IC0902 "Cognitive Radio and Networking for Cooperative Coexistence of Heterogeneous Wireless Networks"*.
- Tinnirello, I.; Bianchi, G.; Gallo, P.; Garlisi, D.; Giuliano, F. & Gringoli, F. (2012). Wireless mac processors: Programming mac protocols on commodity hardware. Em *INFOCOM, 2012 Proceedings IEEE*, pp. 1269–1277. IEEE.
- Vieira, L. F. M.; Gerla, M. & Misra, A. (2013). Fundamental limits on end-to-end throughput of network coding in multi-rate and multicast wireless networks. *Computer Networks*, 57(17):3267--3275.
- Yang, Z.; Zhang, J.; Tan, K.; Zhang, Q. & Zhang, Y. (2015). Enabling tdma for today's wireless lans. Em *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1436--1444. IEEE.
- Yildirim, M. (2015). Ofdm based communication system using usrp. Em *2015 Twelve International Conference on Electronics Computer and Computation (ICECCO)*, pp. 1--3. IEEE.
- Ziouva, E. & Antonakopoulos, T. (2002). CSMA/CA performance under high traffic conditions: throughput and delay analysis. *Computer Communications*, 25(3):313 – 321. ISSN 0140-3664.



Đurišić, M. P.; Tafa, Z.; Dimić, G. & Milutinović, V. (2012). A survey of military applications of wireless sensor networks. Em *2012 Mediterranean Conference on Embedded Computing (MECO)*, pp. 196–199. ISSN 2377-5475.



# Apêndice A

## Descrição detalhada da implementação

Neste apêndice descrevemos detalhadamente a implementação dos protocolos CSMA e TDMA utilizados no trabalho. Falamos sobre o funcionamento geral desses protocolos e comentamos detalhadamente o fluxo de execução dos seus processos, esclarecendo o máximo possível cada etapa separadamente e a forma como elas se integram. Falamos sobre as linguagens e bibliotecas usadas no desenvolvimento, apresentamos a justificativa para cada decisão expondo as vantagens e desvantagens levadas em consideração. Apresentamos as estruturas de dados utilizadas, o contexto do seu uso e as razões porque foram escolhidas. Comentamos também os recursos e mecanismos envolvidos no desenvolvimento e tentamos mostrar os caminhos que nos levaram à construção da forma como o fizemos.

### A.1 Aspectos comuns

Para construir os protocolos CSMA e TDMA, foi necessário implementar antes uma base com estruturas de dados e operações básicas a serem utilizadas em ambos. Usamos como apoio para a implementação, a pilha ZigBee apresentada no trabalho [Schmid et al., 2016]. Nesse trabalho, os autores deixam claro que a sub-camada MAC é muito básica, de fato, além de intermediar os pacotes entre as camadas de rede e física, a única função existente é a conferência de integridade do pacote através do CRC. Assim, foi preciso desenvolver uma plataforma básica com algumas estruturas e funções que possibilitassem iniciar a construção dos protocolos.

Além dessa estrutura básica, apresentamos aqui, outros aspectos que são comuns aos dois protocolos, como linguagem utilizada, bibliotecas e algumas estruturas já

presentes no GNU Radio.

### A.1.1 Bibliotecas e Estruturas de dados

Descrevemos aqui algumas bibliotecas e estruturas de dados que também são comuns aos dois protocolos desenvolvidos. Essas estruturas não estão intrinsecamente ligadas a nenhum deles, mas fazem parte da base construída para desenvolvê-los, portanto, podem ser utilizadas na construção de qualquer outro protocolo para a sub-camada MAC.

- **Boost:** Boost [Dawes et al., 2015] é um conjunto de bibliotecas que usam recursos de C++ e são amplamente utilizadas por desenvolvedores dessa linguagem. Como *Boost* já é um pré-requisito para instalação do GNU Radio e é utilizada internamente em blocos nativos, visando evitar problemas de incompatibilidade de qualquer natureza, resolvemos utilizar algumas de suas bibliotecas para desenvolver os nossos protocolos. As duas bibliotecas usadas foram “boost/thread.hpp” e “boost/date\_time.hpp” para temporizações tanto no TDMA quanto no CSMA.
- **PMT (*Polymorphic Types*):** É um tipo genérico, projetado para encapsular diversos outros tipos de dados para serem transmitidos com segurança. Este tipo é muito utilizado em GNU Radio para troca de mensagens entre blocos ou *threads*. Com a inclusão da biblioteca “pmt.h” é possível utilizar a estrutura de dados e diversas funções associadas.
- **Pacote `sendPackage`:** É um pacote que encapsula o PMT e contém alguns metadados que auxiliam no processamento das mensagens trocadas. Durante o processo de execução de cada protocolo, operações internas são naturalmente necessárias, alguns exemplos dessas operações são o enfileiramento das mensagens para envio, contagem de reenvios e indicação de confirmação. Essas operações exigem uma série de informações que não estão nas próprias mensagens ou requerem um certo processamento para serem obtidas. Por essas razões, foi criado o pacote *sendPackage*, uma estrutura que contém a mensagem do tipo PMT e guarda algumas informações a seu respeito.

A estrutura do *sendPackage* contém as seguintes informações:

```
{
    pmt::pmt_t package;
    unsigned char idPackage;
```

```
    short resends;  
    bool remove;  
}
```

Nessa estrutura, o atributo *package* contém o pacote PMT já pronto para ser enviado, o atributo *resends* contém o número de vezes que o pacote foi enviado, o atributo *idPackage* contém o id do pacote e o atributo *remove* indica se o pacote pode ser removido ou não.

O *idPackage* contém uma informações redundante, já que o id do pacote também está no cabeçalho do próprio pacote que está encapsulado no PMT. A razão para manter essa informação no atributo *sendPackage* é evitar este procedimento de desencapsular o PMT e verificar o cabeçalho toda vez que for necessário saber o id do pacote nos processamentos internos.

O atributo *remove* é iniciado como *false* e durante a execução do protocolo, ele pode se tornar *true* quando a quantidade de envios for excedida ou quando o recebimento do pacote for confirmado.

Além dos métodos *get* e *set* para todos os atributos, a classe *sendPackage* possui o método *void increaseResends(int limit)*. Este método incrementa o número de reenvios toda vez que o pacote é enviado, quando o número de reenvios excede o limite passado como parâmetro, o método seta o atributo *remove* como *true*.

- **Pacote de dados:** Este pacote está de acordo com o padrão [LAN/MAN Standards Committee et al., 2003] e é a principal estrutura utilizada para a transmissão e recepção de mensagens nos protocolos implementados. A implementação original do pacote já estava na pilha ZigBee utilizada no trabalho, a função que o gera apenas sofreu algumas modificações quanto à modularização e parametrização das informações.

No caso dos protocolos implementados nesse trabalho, fazemos a confirmação de todos os pacotes de dados, por simplificação, não adotamos fragmentações.

Em termos de implementação, os pacotes de dados são vetores do tipo *char* que são construídos byte a byte conforme a especificação apresentada. Primeiramente o cabeçalho é construído, em seguida o *payload* é concatenado ao vetor e por último é calculado o CRC e o resultado é também concatenado ao final. Em seguida o vetor é encapsulado em um *blob* PMT e incluído em um *pair* também PMT. Dessa forma o pacote de dados está pronto para ser transmitido.

- **Pacote Ack:** Este pacote também está de acordo com o padrão [LAN/MAN Standards Committee et al., 2003] e é a estrutura utilizada para a confirmação de recepção de mensagens nos protocolos implementados.
- **Endereços MAC:** De acordo com o padrão 802.15.4, o endereço MAC dos nós em rede pode ser formado por 16 ou 64 bits, decidimos utilizar o endereço de 16 bits. Por razões de simplicidade, escolhemos utilizar a versão menor, dessa forma, além de usar o número MAC como identificação única do nó, utilizamos também como identificador do mesmo na rede internamente, sem grande impacto de processamento ou memória. Caso escolhêssemos o endereço de 64 bits, todos os pacotes teriam 6 bytes a mais em cada campo de endereço e seria necessário criar um outro mecanismo, como outro campo de 1 byte, para identificar o nó na rede internamente.

Os protocolos implementados suportam o envio de mensagens *unicast* e *broadcast*. Em *unicast*, o procedimento é o usual, basta colocar o endereço do destinatário no campo endereço de destino do pacote, assim, os nós aos quais este pacote não é destinado irão ignorá-lo. Já o nó para o qual o pacote é destinado irá recebê-lo e enviar uma confirmação para o endereço de origem que consta no pacote recebido. No caso de *broadcast*, o endereço de destino deve ser *FFFF*, nesse caso, para evitar colisões de pacotes ACK, as mensagens não são confirmadas.

- **Fila de envio:** A fila de envios é uma lista que guarda os pacotes prontos para serem enviados para a rede na ordem em que foram preparados. A implementação da fila utiliza a lista padrão de C++ e os elementos dessa lista são os pacotes *sendPackage*. Assim que os pacotes de dados são encapsulados como PMTs e estão prontos para serem enviados, é gerado um objeto *sendPackage* que contém o pacote PMT, então este objeto é colocado na última posição da lista. Quando o objeto está na primeira posição da lista, o pacote PMT contido nele é enviado e o atributo *resends* do objeto *sendPackage* é incrementado. O objeto permanece na lista para possível reenvio. O objeto só é removido da lista quando é recebido um pacote Ack com o seu número de sequência ou quando o número de reenvios é excedido. Nesses dois casos, o atributo *remove* do objeto *sendPackage* conterà o valor *true*.

### A.1.2 Temporizações

Os dois protocolos desenvolvidos precisam utilizar temporização em situações como *time out* para reenvio de mensagem, alocação de *slot* de comunicação e *backoff*. Os

mecanismos utilizados em todos os casos são similares e são construídos utilizando a biblioteca *boost/thread*. De forma geral, uma *thread* é criada e colocada para dormir durante o tempo da temporização, após esse tempo, ela acorda e uma ação é executada. Há três variações de temporização implementadas, conforme descrevemos a seguir:

- **Espera simples:** Essa temporização é a mais simples e consiste apenas em esperar um determinado tempo. A implementação é feita com uma *thread* que é colocada para dormir durante o tempo que se deseja esperar. É utilizada, por exemplo, no TDMA, para responder uma mensagem de alocação no *slot* correto.
- **Espera com notificação:** Essa espera é diferente da anterior por não ter um limite de tempo previamente definido, neste caso, a *thread* fica esperando uma notificação que é feita por outra *thread* em outro ponto do sistema. Essa espera é utilizada, por exemplo, no CSMA, quando a *thread* que trata a fila de envios aguarda até que algum dado seja enfileirado e possa ser enviado.
- **Espera mista:** Essa espera é um pouco mais complexa e combina as duas soluções anteriores, pois a *thread* espera até que uma das duas situações ocorra, ou seja, a *thread* acorda quando um determinado tempo é decorrido ou quando, antes desse tempo, ocorre uma notificação. Essa temporização é utilizada no CSMA quando um pacote é enviado e o sistema precisa esperar o tempo de *timeout* para reenvio da mensagem, mas pode interromper essa espera caso receba uma confirmação desse pacote.

## A.2 CSMA

O CSMA implementado para esse trabalho tem como principais características: verificação da portadora, confirmação de mensagens, *backoff* binário exponencial e ausência do mecanismo de RTS e CTS. Os fluxogramas do transmissor e receptor são representados respectivamente nas Figuras A.2 e A.3. A implementação possui basicamente 4 *threads* ativas:

1. Trata do *carrier sense*
2. Trata os pacotes recebidos
3. Trata mensagens vindas da camada de rede e as prepara para envio
4. Envia as mensagens

### A.2.1 *Carrier sense*

A principal característica do protocolo CSMA é a verificação do meio para evitar colisões. No protocolo que implementamos, isso é feito medindo a potência do sinal do meio na frequência em que estão sendo feitas as transmissões. A implementação usa os valores de leitura feita por dois componentes OTT auxiliares, são eles o *gr-gpsuhd* e *gr-eventstream*. São usados os blocos *Trigger Sample-Timer Event, Handler to PDU, CPDU Average Power* e *PDU Remove* para verificar uma série de informações sobre meio e informar os valores ao bloco *IEEE 802.15.4 MAC* modificado, através da porta *cs\_in*.

Internamente, foi implementada a função *void cs\_in(pmt::pmt\_t msg)*, esta função recebe a mensagem com os valores de medição mencionados. Os valores estão dispostos em um dicionário, onde a chave é uma *string* contendo o nome da característica e o valor contém o valor medido.

Após extraído o valor da potência, o valor de última leitura do meio é atualizado. O *carrier sense* é feito comparando o último valor de potência do sinal lido para a frequência de transmissão com um limiar definido previamente. Se o sinal está acima desse limiar, considera-se que o meio está ocupado, caso contrário, o canal está livre.

Esse sistema de leitura do ambiente é executado e o último valor lido é atualizado regularmente, independente do funcionamento do Transmissor ou do Receptor. Quando é necessário saber se o meio está ocupado, o Transmissor apenas utiliza a função *bool isChannelBusy(float refValue)*, que faz a comparação do último valor lido com o valor de referência. A frequência de atualização do valor lido e o valor de referência se mantêm fixos durante toda a execução, mas são parâmetros previamente configuráveis.

*av\_protocolos\_azao*

### A.2.2 *Receptor*

A parte dedicada à recepção e tratamento das mensagens recebidas do protocolo CSMA está resumida no fluxograma contido na Figura A.3. Todas as ações tomadas pelo receptor se iniciam com a chegada de alguma mensagem pela porta *pdu\_in* do bloco *IEEE802.15.4 MAC* modificado, estas ações são independentes do funcionamento do transmissor.

Assim que o bloco recebe alguma mensagem, é feito um teste de tamanho do pacote, caso seja menor do que o mínimo esperado, o pacote é descartado. Em caso de sucesso, a mensagem é desencapsulada do PMT e é feita a verificação do CRC, se a verificação retornar falha, o pacote é descartado, caso contrário, o pacote começa a ser tratado de acordo com o tipo.



A verificação inicial de tipo detecta se o pacote é de confirmação. Se o pacote for de ACK, o número de sequência é recuperado e é feita uma busca na lista de envios para proceder a confirmação do pacote. A confirmação consiste em encontrar o objeto *sendPackage* que contenha o pacote PMT que possui o mesmo número de sequência do pacote ACK recebido e então setar o atributo *remove* como *true*. Como mencionado na Seção A.1.1, o número de sequência do pacote PMT é replicado do objeto *sendPackage* para facilitar o processamento. Para evitar problemas de Produtor/Consumidor, o pacote não é removido no momento da confirmação, apenas o Transmissor irá manipular a lista incluindo e removendo elementos. A sinalização de que o pacote pode ser removido é exatamente o atributo *remove* do pacote *sendPackage*.

No caso de ser um pacote de dados é feita uma verificação de endereçamento. O pacote só é tratado caso o endereço de origem seja diferente do endereço local e o endereço de destino seja igual ao endereço local ou igual ao endereço de *broadcast*. Satisfeitas todas essas condições, o *payload* do pacote é encaminhado para a camada de rede através da porta *app\_out*. Após esse envio, caso o pacote não seja de *broadcast*, um ACK com o mesmo número de sequência do pacote de dados recebido é gerado e, após a espera do SIFS, é enviado à camada física através da porta *pdu\_out*.

### A.2.3 Transmissor

A parte dedicada à preparação e transmissão das mensagens do protocolo CSMA está resumida no fluxograma contido na Figura A.2. Existem dois fluxos separados no Transmissor, um deles é responsável por receber e tratar as mensagens que chegam da camada de rede e o outro é responsável por tratar a fila de envios e executar as esperas inerentes ao protocolo.

O fluxo responsável por tratar as mensagens da camada de rede tem suas ações condicionadas ao recebimento de uma mensagem. Assim que recebe a mensagem, o pacote de dados MAC é construído e encapsulado, então, o objeto é colocado na última posição da lista de envios. Como parte da comunicação entre os dois fluxos, existe um atributo global da classe, do tipo *bool*, chamado *data\_ready*. Este atributo é iniciado com o valor *false*, assim que algum pacote é colocado na fila, este atributo recebe o valor *true* e o tratador da fila de envios é acionado. A sincronia entre essas duas partes é feita utilizando um recurso *wait* da biblioteca *Boost*.

A função *runSending()* procede o envio dos pacotes até que a fila esteja vazia, assim, para evitar que pacotes fiquem na fila indefinidamente, a atribuição *data\_ready = false* deve necessariamente estar antes da chamada dessa função.

A função *runSending()* trata o primeiro elemento da fila verificando primeira-

mente se ele deve ser removido. Em caso negativo, é feita a espera do DIFS, utilizando o mecanismo de *Espera simples*, em seguida é feito o procedimento de *backoff*, utilizando o algoritmo BEB (*Backoff* Binário Exponencial), e o pacote é enviado. Então é feita a espera de *time out* e caso o pacote não tenha sido confirmado, é reenviado. A espera do *time out* é interrompida caso o pacote seja confirmado antes, esse é o mecanismo de *Espera mista* descrito na Seção A.1.2. A função repete esses procedimentos até que não haja mais elementos na fila.

### A.3 TDMA

O TDMA implementado para este trabalho tem como principais características: a alocação de *slots* de tempo distintos para a comunicação de cada nó e a confirmação de mensagens. Existem dois tipos diferentes de nós, o nó Comum, que apenas se comunica com outros nós e o nó coordenador, que além de se comunicar normalmente, também gerencia a distribuição de *slots* de tempo para a troca de mensagens. O nó coordenador é definido previamente e permanece assim durante todo o tempo de execução.

O tempo é dividido em intervalos chamados *superquadros*, neste intervalo, todos os nós têm a chance de se comunicar com outros nós transmitindo dados e recebendo confirmações. O *superquadro* e suas subdivisões estão representados na Figura A.1.

O superquadro é formado por dois períodos:

- **Período de alocação canal:** O período de alocação possui uma fase de sincronização e uma fase de alocação dos nós. Na fase de sincronização, o nó coordenador envia uma mensagem *broadcast* de controle informando a ordem em que os nós devem se comunicar para requisitar a alocação de um *slot* de comunicação. Esta mensagem, além de informar esta ordem, serve como marco de início do superquadro, sincronizando o processo entre todos os nós.

Após receber esta mensagem, se inicia a fase de alocação, assim, cada nó Comum irá utilizar a informação de ordem fornecida pela mensagem de sincronização e a *Espera simples* para fazer a requisição de *slot* no momento correto. Quando chega o momento, o nó Comum envia uma mensagem ao coordenador informando se deseja se comunicar neste superquadro ou não. O nó coordenador organiza então a ordem em que os nós devem se comunicar e envia uma mensagem de controle a todos os nós informando essa ordem.

- **Período de comunicação:** Este período é inteiramente formado por *slots* de comunicação, a quantidade de *slots* é igual à quantidade de nós que requisita-

ram o canal. Usando o mesmo mecanismo da fase de alocação, cada nó espera o momento correto para se comunicar. Quando chega esse momento, ele envia a mensagem que deseja e aguarda a confirmação. Os nós tratam as mensagens de Ack apenas no seu *slot* de comunicação, isso evita que um nó confirme as próprias mensagens tratando Acks que ele mesmo enviou, já que o pacote Ack possui identificadores apenas dos pacotes e não dos nós. Quando um nó recebe uma mensagem de outro nó, ele imediatamente responde com um Ack, essa confirmação pode ser feita por qualquer nó durante todo o período de comunicação.

O nó coordenador possui comportamento um pouco diferente para se comunicar, pois na fase de alocação ele não envia pacotes de requisição. Por simplicidade, decidimos alocá-lo sempre no último *slot*. Assim, o nó coordenador é sempre o último a enviar mensagens no período do comunicação.

O tamanho de todos os *slots* e o tamanho do período para confirmação dentro do *slot* de comunicação são parametrizáveis. Entre cada *slot* existe ainda um pequeno período adicional reservado para cobrir imprecisões na sincronização, este período também é parametrizável.

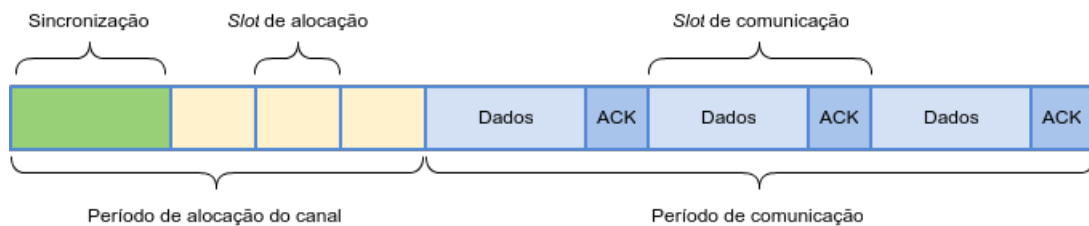


Figura A.1. Superquadro do TDMA

### A.3.1 Mensagens de controle

Além dos pacotes de dados e de confirmação, que são iguais aos usados no CSMA, o TDMA utiliza alguns pacotes de controle para gerenciar a alocação dos *slots*. Existem 3 tipos de pacotes de controle: sincronização, requisição e alocação. A estrutura desses pacotes é a mesma do pacote de dados do padrão 802.15.4, o que os diferencia é apenas o primeiro byte do *payload*, no pacote de sincronização, o primeiro byte do *payload* é o caractere *S*, no pacote de requisição é o caractere *R* e no pacote de alocação é o caractere *A*.

No *payload* do pacote de sincronização, após o caractere identificador, são colocados todos os endereços dos nós da rede, na ordem em que devem responder enviando

as requisições de comunicação. No pacote de requisição, é colocado o caractere  $T$  caso o nó queira se comunicar ou o caractere  $F$  caso não queira. No pacote de alocação, são colocados os endereços de todos os nós que desejam se comunicar, na ordem em que irão se comunicar.

As mensagens de sincronização e alocação são *broadcast* do coordenador para os nós Comuns e a mensagem de requisição é *unicast*, indo sempre dos nós Comuns para o coordenador. Decidimos utilizar a estrutura do pacote de dados para este fim devido à liberdade de endereçamento, ou seja, diferente dos pacotes do tipo *beacon* no padrão 802.15.4, o pacote de dados permite utilizar endereços de origem e destino. Como a mensagem de requisição é *unicast*, necessariamente deveríamos utilizar o pacote de dados, então, decidimos utilizar a mesma estrutura para todas as mensagens de controle.

### A.3.2 Funcionamento dos nós

Os dois tipos de nós são implementados com a mesma estrutura, existe um atributo do tipo *bool* chamado *isCoordinator* que diz se o nó é coordenador ou Comum. Qualquer nó pode ser coordenador e todas as ações durante a execução são guiadas pelo atributo *isCoordinator*. O nó coordenador é definido previamente e permanece o mesmo durante toda a execução.

O nó Comum possui duas *threads* ativas:

1. A primeira *thread* funciona como no CSMA, os pacotes são recebidos da camada de rede, encapsulados e enfileirados para envio.
2. A segunda *thread* é responsável pelo tratamento das mensagens de controle vindas do nó coordenador, faz também o envio da requisição de comunicação para o coordenador, o envio dos pacotes de dados que estão na fila, o recebimento dos pacotes de dados vindos dos outros nós, o envio de Acks e o tratamento de Acks recebidos. Como pode ser visto na Figura A.4, todas as ações são iniciadas a partir do recebimento de alguma mensagem, seja de controle, dados ou ack.

O nó coordenador possui ações mais complexas e possui 3 *threads* ativas.

1. A primeira *thread* funciona exatamente como no nó Comum.
2. A *thread* 2 é responsável pelo envio de Acks, tratamento dos pacotes de dados e Acks recebidos. Essa *thread* trata também os pacotes de requisição de comunicação recebidos, para que posteriormente a *thread* 3 monte o pacote de alocação.

3. A *thread* 3 trata da alocação dos *slots* e gerenciamento da comunicação. Como está representado na Figura A.5, essa *thread* configura e envia o pacote de sincronização, espera o tempo de requisições, em seguida monta e envia o pacote de alocação, espera todo o tempo de comunicação dos outros nós, então envia os próprios dados caso haja e reinicia o ciclo enviando novo pacote de sincronização. Essa *thread* executa parte da comunicação que no nó Comum é feita pela *thread* 2. Ela faz o envio dos pacotes de dados, pois no caso do coordenador, essa não é uma ação reativa, condicionada ao recebimento de um pacote de alocação.

Os dois tipos de nós gerenciam internamente um conjunto de estados, determinados pelo atributo do tipo *int* chamado *state*. Existem 3 estados: (0)sincronização, (1)alocação e (2)comunicação. Para que o protocolo funcione corretamente, os estados devem estar sincronizados entre todos os nós e as ações inerentes a um estado não podem acontecer em outro. Assim, a validação de cada ação em relação ao estado é feita constantemente. Se um nó, por exemplo, estiver no estado de comunicação, aguardando o seu *slot*, e recebe um pacote de sincronização, ele interrompe a fase de comunicação e passa imediatamente para o estado de sincronização. Em outro exemplo, se um nó está no estado de alocação e por algum motivo recebe um pacote de dados, este pacote não será confirmado, pois esta ação só pode ser executada no estado de comunicação. Essas separações garantem a sincronização da execução do protocolo, evitando erros e colisões. Esses estados são representados pelas baias verticais nas Figuras A.4 e A.5.

## A.4 Interface de troca

As seções anteriores descreveram a implementação completa dos dois protocolos separadamente, mas para incluí-los na plataforma FS-MAC é necessário acrescentar algumas funcionalidades. A inclusão dos protocolos na plataforma requer a implementação de duas interfaces, *interface de troca* e *interface de comunicação externa*. A interface de comunicação externa é mais simples e por isso está brevemente descrita na seção 4.2.1. Já a interface de troca é um pouco mais complexa, de forma que julgamos necessário descrever aqui os seus detalhes.

A interface de troca recebe mensagens de dois tipos: *Parar atuação* e *Iniciar atuação* e retorna também dois tipos de mensagem *Atuação iniciada* e *Atuação parada*. Quando recebe a mensagem para parar a atuação, o protocolo interrompe o seu próprio funcionamento e retorna uma mensagem informando que a atuação está interrompida e fornecendo a fila de envios no estado em que se encontra. No caso do CSMA, a parada corresponde a interromper a *thread* de transmissão, descrita na Seção A.2. As outras

*threads* não precisam ser interrompidas, pois elas são ativadas através do recebimento de informações externas e essas informações não serão mais direcionadas para o protocolo quando ele não for o protocolo em atividade. No caso do TDMA, quando se trata de um nó Comum, nada precisa ser feito, pois qualquer atividade desse protocolo, nesse tipo de nó, é uma resposta a algum evento externo e esses eventos não ocorrerão quando esse protocolo não for aquele que estiver em atividade. Quando se trata do nó coordenador do TDMA, a *thread* principal, responsável pela alocação dos *slots* de tempo, é interrompida.

Após interromper seu funcionamento, a fila de envios, contendo todos os pacotes precisa ser passada ao módulo de Troca para que seja encaminhada ao novo protocolo ativo. Esse envio pode ser implementado de diversas formas, algumas delas são: Conversão da fila inteira em um pacote do tipo PMT e envio desse pacote; Envio de todos os pacotes da fila individualmente; Implementação da fila estática e passagem da sua referência; Implementação de uma classe *singleton* que contém a fila. Todas as implementações possuem custos e benefícios, aquelas que passam a fila inteiramente pela porta de controle possuem o custo claro da transmissão. No primeiro caso, qualquer erro requer o reinício de toda a transmissão e no segundo caso, existe o custo adicional de verificar a integridade de cada pacote e um mecanismo para marcar a finalização do envio. O benefício dessa abordagem é a liberdade que os protocolos têm de implementar suas filas internamente de maneiras distintas. Nas abordagens por referência, a fila estática, embora seja muito simples de implementar, pode limitar a forma como a o protocolo é implementado internamente.

Escolhemos então a última abordagem, mantendo uma classe *singleton* que contém a lista e reproduz todos os métodos dessa lista utilizados dentro do protocolo. Dessa forma a passagem da lista é feita por referência e apenas a mensagem de controle do tipo *Atuação parada* é enviada ao bloco de Troca. A desvantagem dessa abordagem é que se a fila implementada internamente a um protocolo que será adicionado à plataforma for diferente da implementação dos demais, o processo de *passagem da fila* executado por esse protocolo incluirá transcrição de sua fila para o formato da classe *singleton* implementada nos demais.

Quando recebe a mensagem de *Iniciar atuação* juntamente com a fila de envios, o protocolo estabelece a fila recebida como sua própria fila de envios e inicia a sua atuação. No caso da abordagem utilizando o padrão *singleton*, é necessário apenas instanciar um objeto do tipo que contém a lista. Nesse caso, como a instância já existe, a referência retornada já conterá a fila de envios do protocolo desativado. O início da atuação corresponde ao oposto das operações realizadas quando o protocolo é interrompido, ou seja, no caso do CSMA, a *thread* de transmissão é reativada e no caso

do TDMA, nó coordenador, a *thread* de gerenciamento dos *slots* de tempo é reativada. Após o início da atuação, o protocolo envia para o módulo de troca uma mensagem do tipo *Atuação iniciada* para que o módulo de troca dê prosseguimento ao funcionamento da rede.





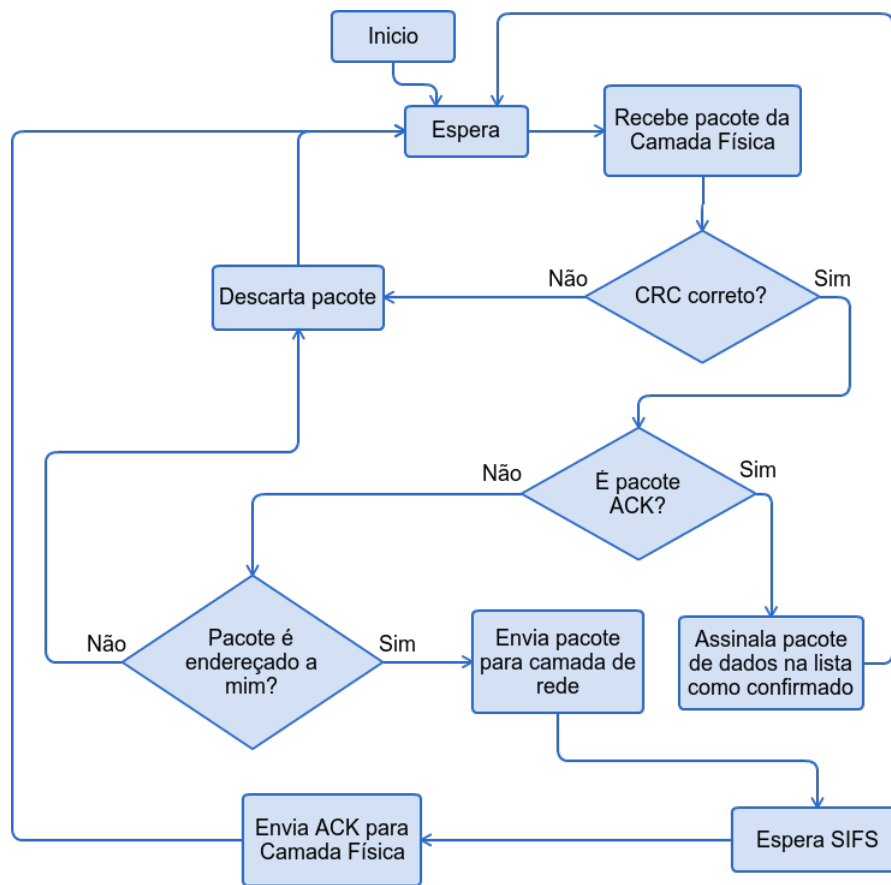
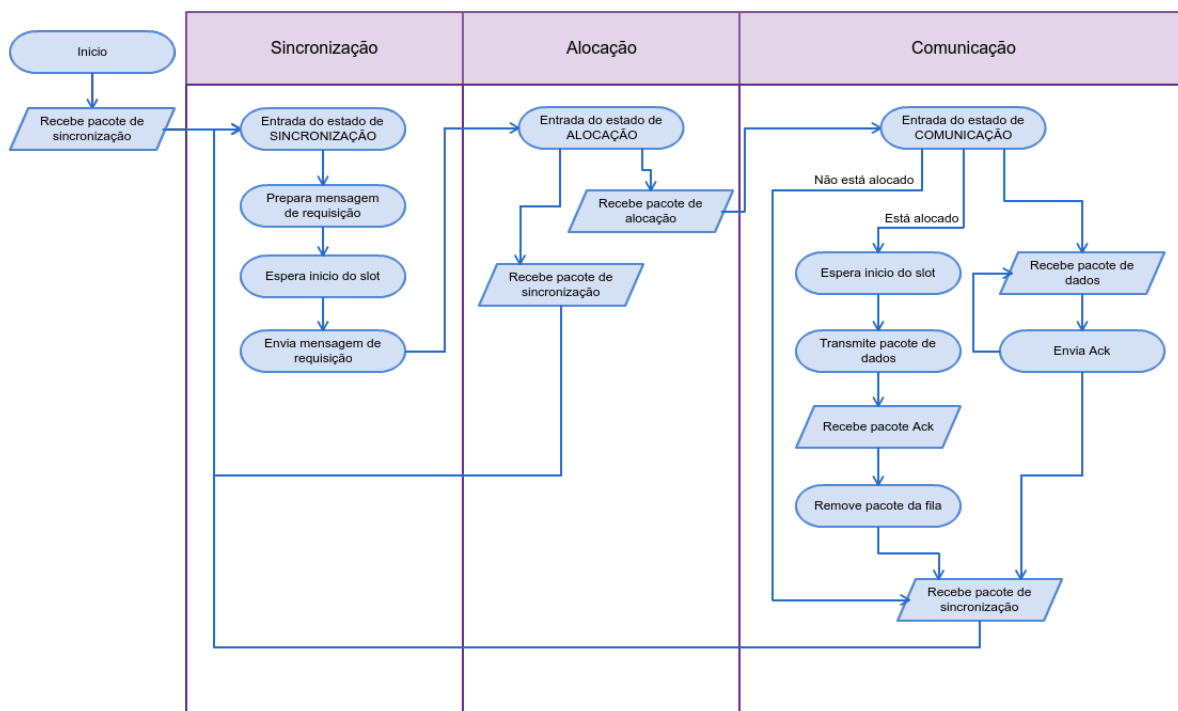


Figura A.3. Fluxograma do CSMA - Receptor



**Figura A.4.** Fluxograma do TDMA - Nó Comum

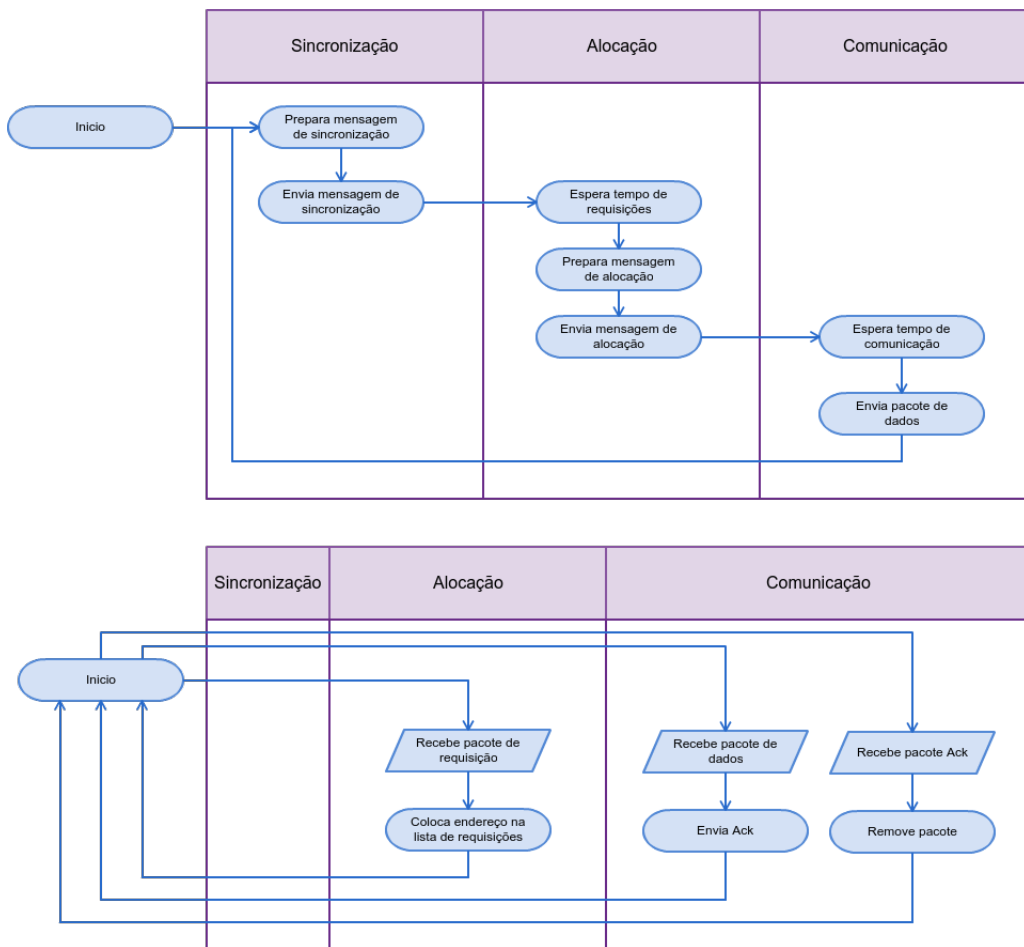


Figura A.5. Fluxograma do TDMA - Nó coordenador