Rodrigo Vieira Steiner

UMA PROPOSTA DE FRAMEWORK PARA A GERAÇÃO DE PROTOCOLOS MAC PARA RSSF

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Ciência da Computação.
Orientador: Prof. Dr. Antônio Augusto Medeiros Fröhlich

Florianópolis

2013

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Steiner, Rodrigo Vieira
Uma proposta de framework para a geração de protocolos MAC para RSSF / Rodrigo Vieira Steiner ; orientador, Antônio Augusto Medeiros Fröhlich - Florianópolis, SC, 2013.
103 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Redes de sensores sem fio. 3. Protocolos de controle de acesso ao meio. I. Fröhlich, Antônio Augusto Medeiros. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Rodrigo Vieira Steiner

UMA PROPOSTA DE FRAMEWORK PARA A GERAÇÃO DE PROTOCOLOS MAC PARA RSSF

Esta Dissertação foi julgada adequada para a obtenção do Título de Mestre em Ciência da Computação, área de concentração Sistemas de Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Florianópolis, 18 de Fevereiro 2013.

Prof. Dr. Ronaldo dos Santos Mello
Coordenador do Curso

Banca Examinadora:

Prof. Dr. Antônio Augusto Medeiros Fröhlich
Orientador

Prof. Dr. Carlos Eduardo Pereira

Prof. Dr. Leandro Buss Becker

Prof. Dr. Mario Antonio Dantas

Dedico esta dissertação aos meus pais, Ana Maria Vieira Steiner e Sérgio Machado Steiner.

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, minha irmã e meus avós, por sempre estarem presentes em minha vida, por seu amor incondicional e apoio constante. Também quero agradecer à minha namorada, Andréa Puerta Pereira Oliveira, por seu amor, carinho, companheirismo e por me fazer uma pessoa mais feliz.

Agradeço ao professor Antônio Augusto Medeiros Fröhlich pelo apoio, dedicação e inúmeros puxões de orelha dados durante a orientação deste trabalho.

Agradeço pela oportunidade de fazer parte do Laboratório de Integração Software e Hardware, local de muito trabalho, discussões técnicas e científicas, alegrias e diversão. Muito obrigado aos meus amigos do laboratório, em especial ao Tiago Rogério Mück que teve participação fundamental no ínico deste trabalho.

Agradeço aos meus amigos de longa data Gillian Luis Andrade Júlio, Leonardo Berns Gorges, Lucas Vieira, Matheus do Valle Gallina e Paulo Leonel Teixeira, pelo companheirismo, por me aturarem todos esses anos e pelos momentos de muitas alegrias.

“It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change."

Charles Darwin

RESUMO

Redes de sensores sem fio são altamente dependentes de protocolos de controle de acesso ao meio para fazer uso efetivo dos poucos recursos disponíveis em seus nós sensores. Entretanto a maioria das otimizações propostas nos protocolos existentes focam em segmentos específicos do espaço de projeto. O que é considerado como uma otimização por uma classe de aplicações pode representar uma forte limitação para outras. Um protocolo com o objetivo de cobrir uma ampla fração do universo de aplicações de redes sensores deve apresentar algum mecânismo de configurabilidade ou adaptabilidade. Nesse sentido surgiu o *Configurable Medium Access Control* (C-MAC) (WANNER; OLIVEIRA; FROHLICH, 2007). O C-MAC funciona como um *framework* de estratégias de controle de acesso ao meio, as quais podem ser combinadas para produzir protocolos específicos de aplicação. Através desse paradigma, programadores de aplicações podem criar novos serviços de comunicação sob demanda e experimentar diferentes parâmetros de comunicação, coletando métricas para identificar e ajustar o protocolo às suas necessidades. Contudo, a arquitetura original do C-MAC fornece um baixo reuso de seus componentes em software, dificultando e reduzindo sua configurabilidade.
Esta dissertação apresenta a proposta, implementação e avaliação de uma nova arquitetura para o *framework* C-MAC. Através de uma análise das diferentes categorias de protocolos MAC para redes de sensores, foram identificadas características comuns e especificidades desses protocolos que foram consideradas no desenvolvimento do novo C-MAC. Além disso, o que era um único grande componente na arquitetura original, foi separado em diferentes microcomponentes e novos microcomponentes foram incluídos de forma a suportar mais funcionalidades. Dessa forma, a arquitetura proposta possibilita um maior reuso de seus componentes em software, fornecendo maior configurabilidade e, consequentemente, suportando mais aplicações.

Palavras-chave: MAC, RSSF, configurabilidade.

ABSTRACT

Wireless sensor networks are highly dependent on medium access control protocols to make effective use of the few resources available on sensor nodes. Nevertheless, most of the optimizations proposed by existing protocols focus on specific segments of the design space. What is considered an optimization by one class of applications can represent a strong limitation for others. A protocol aiming at covering a large fraction of the application universe for sensor networks must feature configuration or adaptation mechanisms. Therefore, the *Configurable Medium Access Control* (C-MAC) was created (WANNER; OLIVEIRA; FROHLICH, 2007). C-MAC is realized as a framework of medium access control strategies that can be combined to produce application-specific protocols. Through this paradigm, application programmers can create new communication services on demand and experiment with different communication parameters, collecting metrics to identify and adjust the protocol to match their applications' requirements. Nonetheless, C-MAC original architecture provides low reuse of software components, hindering its configurability.

This dissertation presents the proposal, implementation, and evaluation of a new architecture for the framework C-MAC. Through the analysis of the different categories of MAC protocols for sensor networks, common characteristics and specificities of these protocols were identified and used in the development of the new C-MAC. Thus, the proposed architecture enables a greater reuse of its software components, providing greater configurability and, consequently, supporting a wider range of applications.

Keywords: MAC, WSN, configurability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS E ABREVIAÇÕES

ACK *Acknowledgment*

ADHOP *Ant-based Dynamic Hop Optimization Protocol*

AES *Advanced Encryption Standard*

B-MAC *Berkeley-MAC*

C-MAC *Configurable Medium Access Control*

CAP *Contention Access Period*

CCA *Clear Channel Assessment*

CFP *Collision Free Period*

CRC *Cyclic Redundancy Check*

CSMA *Carrier Sense Multiple Access*

CSMA-CA *Carrier Sense Multiple Access with Collision Avoidance*

CSMA-CD *Carrier Sense Multiple Access with Collision Detection*

CTS *Clear To Send*

EOA *Energy Optimization Approach*

ELUS *EPOS Live Update System*

EPOS *Embedded Parallel Operating System*

GPIO *General Purpose Input/Output*

GPS *Global Positioning System*

GTS *Guaranteed Time Slots*

HECOPS *Heuristic Environmental Consideration Over Positioning System*

IoT *Internet of Things*

IV *Initizalization Vector*

LPL *Low Power Listening*

LQI *Link Quality Indicator*

MAC *Medium Access Control*

mac *message authenticity code*

MANET *Mobile Ad-hoc Network*

OCB *Offset Codebook*

PER *Packet Error Rate*

RB-MAC *Receiver-Based-MAC*

RSSF Redes de Sensores Sem Fio

RSSI *Received Signal Strength Indicator*

RTS *Request To Send*

RTT *Round-trip Time*

S-MAC *Sensor-MAC*

T-MAC *Timeout-MAC*

TDMA *Time Division Multiple Access*

UML *Unified Modeling Language*

Z-MAC *Zebra-MAC*

SUMÁRIO

1 INTRODUÇÃO

Redes de Sensores Sem Fio (RSSF) são sistemas distribuídos formados por dispositivos, chamados de nós sensores ou nodos sensores, que funcionam de forma autônoma e colaborativa para monitorar algum fenômeno no ambiente em que se encontram. Essas redes podem ser aplicadas em uma grande variedade de cenários, com aplicações que variam entre sistemas de pequeno porte (i.e. monitoramento industrial) e de larga escala (i.e. monitoramento urbano). Tamanha diversidade se traduz em diferentes requerimentos. Desta forma, é importante ter conhecimento sobre uma aplicação para poder responder adequadamente às suas necessidades. A Figura 1, explicada abaixo, apresenta uma taxonomia de aplicações de RSSF e a Tabela 1 classifica algumas aplicações dentro desta taxonomia.

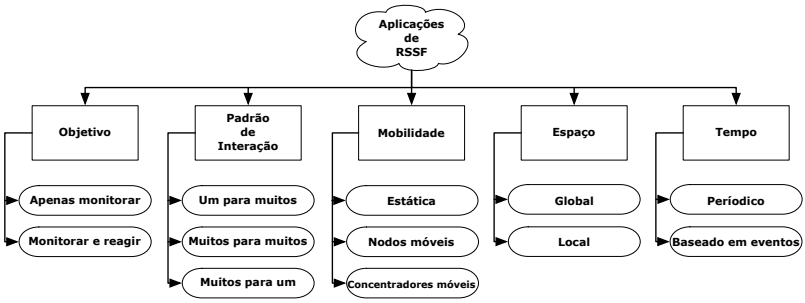


Figura 1: Taxonomia de aplicações de redes de sensores (MOTTOLA; PICCO, 2011).

Objetivo: algumas aplicações *apenas monitoram* o ambiente com o intuito de coletar dados para uma análise *offline*. Outras possuem atuadores, sendo capazes de *monitorar e reagir* de acordo com o dado monitorado.

Padrão de Interação: nodos de uma aplicação podem se comunicar de *muitos para um*, comum em aplicações que apenas monitoram; e de *muitos para muitos*, comum em aplicações que monitoram e reagem. Comunicação de *um para muitos* geralmente é utilizada para enviar comandos aos nodos da rede.

Mobilidade: aplicações *estáticas* não apresentam dispositivos móveis. Outras aplicações podem possuir *nodos móveis* ligados às entidades móveis ou capazes de se mover de forma autônoma; e *concentradores móveis*

caso a coleta de dados deva ser feita quando o concentrador está perto dos nós sensores.

Espaço: em aplicações *globais* os fenômenos de interesse abrangem toda a área geográfica onde a rede está implantada, de forma que o processamento envolve toda a rede, enquanto que em aplicações *locais* o processamento é limitado a uma área de interesse.

Tempo: aplicações *periódicas* são projetadas para de tempos em tempos ler dados dos sensores, processar dados, se comunicar com outros nodos e, se necessário, atuar no meio. Aplicações *baseadas em eventos* ficam monitorando o ambiente até que uma condição seja cumprida, só então a rede começa seu processamento distribuído.

Tabela 1: Algumas aplicações de RSSF aplicadas na taxonomia da Figura 1 (MOTTOLA; PICCO, 2011).

Aplicação	Objetivo	Padrão de Interação	Mobilidade	Espaço	Tempo
Monitoramento de habitat (MAINWARING et al., 2002) (BUONADONNA et al., 2005)	apenas monitorar	muitos para um	estática	global	periódico
Detecção de intrusão (ARORA et al., 2004)	apenas monitorar	muitos para um	estática	local	baseado em eventos
Monitoramento e controle de prédios (DEMIRBAS, 2005)	monitorar e reagir	muitos para um / um para muitos	estática	local	periódico
Navegação de robôs (BATALIN; HATTIG; SUKHATME, 2003)	apenas monitorar	muitos para um	concentradores móveis	local	baseado em eventos
Monitoramento de zebras (JUANG et al., 2002)	apenas monitorar	muitos para um	nodos móveis	global	periódico
Casas inteligentes (NICOLAS et al., 2000)	monitorar e reagir	muitos para muitos	estática	local	periódico

Normalmente, nodos sensores são alimentados por baterias, as quais devem ser substituídas ou recarregadas (por exemplo, usando energia solar) quando esgotadas (DARGIE; POELLABAUER, 2010). Em alguns casos, nenhuma dessas opções é adequada e os nodos são simplesmente descartados uma vez que sua fonte de energia se exaure. O fato de um nodo possuir energia limitada afeta significativamente o projeto de uma RSSF. Nodos sensores devem ser capazes de operar por tempo suficiente para executar as tarefas definidas pela sua aplicação. Como consequência, a eficiência energética é considerada um fator muito importante em RSSF. A Figura 2 mostra o consumo energético de um típico nó sensor. É possível perceber que o rádio é o grande responsável pelo consumo de energia de um nodo, tornando-o um fator crítico para o tempo de vida da rede.

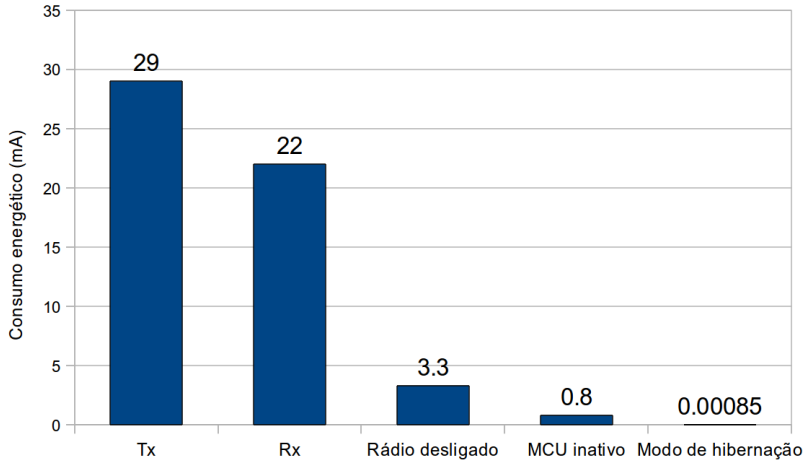


Figura 2: Consumo energético de um típico nó sensor. Dados da plataforma MC1322x (Freescale, 2010).

O uso do rádio em um nodo sensor está intrinsicamente relacionado com os protocolos de Controle de Acesso ao Meio, do inglês *Medium Access Control* (MAC). Esses protocolos são projetados para fornecer acesso eficiente ao meio de comunicação, impedindo que diferentes nodos de uma rede interfiram com as transmissões uns dos outros e lidando com a situação caso isso aconteça. Mais especificamente, protocolos MAC para RSSF são projetados para fazer uso efetivo dos poucos recursos disponíveis em nós sensores tradicionais, energia e vazão em particular, mas também memória e poder de processamento.

Diversos protocolos MAC foram projetados explicitamente para RSSF (BACHIR et al., 2010). Esses protocolos focam em otimizar o consumo de energia da rede diminuindo sua taxa de transferência de dados, enquanto ainda apresentam um desempenho razoável. No entanto, eles não consideram adequadamente todos os requisitos de uma rede de sensores (LIN; WANG; SUN, 2004).

Ainda que existam aplicações típicas de RSSF, diferentes aplicações apresentam peculiaridades sobre o uso da rede. Dessa forma, um protocolo MAC para rede de sensores deve ser flexível o suficiente para suportar uma variedade de padrões de tráfego de rede: comunicação periódica ou esporádica, confiável ou melhor esforço e assim por diante (STANKOVIC et al., 2003). Contudo, flexibilidade também pode se tornar *overhead*. Utilizando duas aplicações da Tabela 1 como exemplos, a aplicação de detecção

de intrusão que envia dados apenas quando um determinado evento ocorre, considerando que esse evento ocorre em intervalos aleatórios, não deveria ter que suportar um mecanismo de agendamento, do inglês *scheduling mechanism*. Por outro lado, a aplicação de monitoramento de habitat que envia dados periodicamente seria muito beneficiada por um mecanismo de agendamento, evitando perder tempo e energia ouvindo o meio fora do período de transmissão. Isso significa que mais do que flexível, um protocolo deve permitir ser configurado de acordo com as necessidades da aplicação pela qual será utilizado.

Dentro deste contexto surgiu o *Configurable Medium Access Control* (C-MAC) (WANNER; OLIVEIRA; FROHLICH, 2007). O C-MAC funciona como um *framework* de estratégias de controle de acesso ao meio, as quais podem ser combinadas para produzir protocolos específicos de aplicação. Ele permite aos programadores de aplicações configurarem diversos parâmetros de comunicação (e.g. sincronização, contenção, detecção de erros, sinais de confirmação, empacotamento) para ajustar o protocolo às suas necessidades. Embora seja altamente configurável, uma implementação anterior do C-MAC para o nodo sensor Mica2 produziu instâncias imitando o protocolo B-MAC apresentando resultados melhores que o original em termos de consumo de memória, vazão e perda de pacotes (WANNER; OLIVEIRA; FROHLICH, 2007). Isto se deve ao uso de técnicas de metaprogramação estática em C++, que permitem ao compilador realizar otimizações no código.

No entanto, a arquitetura original do C-MAC fornece um baixo reuso de seus componentes em software, dificultando e reduzindo sua configurabilidade. Por exemplo, a sincronização foi definida como um único grande componente que deveria ser reimplementado para qualquer novo protocolo, ainda que aspectos como a geração de preâmbulo e sincronização por temporizadores sejam comuns a praticamente qualquer protocolo.

Esta dissertação apresenta a proposta, implementação e avaliação de uma nova arquitetura para o *framework* C-MAC. Através de uma análise das diferentes categorias de protocolos MAC para redes de sensores, foram identificadas características comuns e especificidades desses protocolos que foram consideradas no desenvolvimento do novo C-MAC. Além disso, o que era um único grande componente na arquitetura original, foi separado em diferentes microcomponentes e novos microcomponentes foram incluídos de forma a suportar mais funcionalidades. Dessa forma, a arquitetura proposta possibilita um maior reuso de seus componentes em software, fornecendo maior configurabilidade e, consequentemente, suportando mais aplicações.

1.1 OBJETIVOS

O principal objetivo desta dissertação é o reprojeto do *framework* C-MAC, possibilitando um maior reuso de seus componentes, facilitando e aumentando a sua configurabilidade e, consequentemente, ampliando a gama de aplicações suportadas.

A fim de alcançar esse objetivo principal, foram estabelecidos os seguintes objetivos específicos:

- Estudar os princípios de comunicação em RSSF e os principais protocolos MAC para essas redes, a fim de caracterizar os componentes comuns e as especificidades desses protocolos;
- Criar um modelo formal, utilizando máquinas de estados, para as diferentes categorias de protocolos MAC existentes para RSSF;
- Com o conhecimento adquirido, remodelar e implementar o novo *framework* C-MAC incorporando o mesmo formalismo utilizado para modelar os protocolos MAC para RSSF;
- Gerar diferentes protocolos utilizando o *framework*, verificar o reuso de seus microcomponentes e comparar o desempenho dos protocolos gerados com trabalhos relacionados; e
- Analisar os resultados obtidos.

1.2 ORGANIZAÇÃO DO TEXTO

O restante desta dissertação está organizado da seguinte forma:

Capítulo 2 descreve os princípios de comunicação em RSSF, protocolos MAC, protocolos multicamada e segurança na comunicação.

Capítulo 3 descreve o *framework* C-MAC, a proposta de um novo design para a arquitetura do *framework*, sua implementação, os mecanismos de configuração dessa nova arquitetura e uma estrutura que permite a integração com outras camadas da pilha de comunicação.

Capítulo 4 apresenta a plataforma EPOSMote, utilizada para avaliar a estrutura proposta neste trabalho, e os resultados obtidos.

Capítulo 5 conclui a dissertação e apresenta os trabalhos futuros.

2 COMUNICAÇÃO EM REDES DE SENSORES SEM FIOS

Um canal de rádio não pode ser acessado simultaneamente para transmissão por dois ou mais nodos que estejam dentro de um raio de interferência do rádio, pois isso pode causar conflitos ou interferência nos sinais transmitidos incapacitando a correta recepção dos mesmos. É de responsabilidade do protocolo MAC evitar que isso aconteça e tratar ou sinalizar as camadas superiores da pilha de protocolos de comunicação caso contrário. Em RSSF o MAC é de fundamental importância na determinação da utilização do canal, latência da rede e consumo de energia.

RSSF apresentam uma infraestrutura de comunicação diferente das redes sem fio tradicionais (STANKOVIC et al., 2003). Essas diferenças não se originam apenas de suas características físicas, mas também de suas aplicações, por exemplo: aplicações típicas de RSSF incluem rastreamento de objetos ou detecção de eventos, raramente empregadas em redes sem fio tradicionais.

Enquanto os protocolos das redes tradicionais e das Redes Móveis Adhoc, do inglês *Mobile Ad-hoc Network*s (MANETs), são projetados para maximizar vazão e minimizar latência, protocolos desenvolvidos para RSSF focam em minimizar o consumo de energia. O protocolo IEEE 802.11, por exemplo, consome a mesma quantidade de energia quando está recebendo dados e quando não está recebendo nada (INTANAGONWIWAT; GOVINDAN; ESTRIN, 2000). Isso se deve ao fato que os sistemas das redes tradicionais estão ligados à rede elétrica ou, como os sistemas das redes MANETs, utilizam baterias que podem ser recarregadas ou substituídas. O mesmo já não pode ser dito para RSSF, pois seria no mínimo imprático coletar todos os nodos da rede, substituir suas baterias e devolve-los ao ambiente a ser monitorado.

O projeto de um protocolo MAC para RSSF tem como principal restrição a energia limitada disponível nos nós sensores, mas também leva em conta o baixo poder computacional, pouca memória e a baixa capacidade de sincronização desses dispositivos (BACHIR et al., 2010).

Este capítulo apresenta os princípios de comunicação em RSSF (seção 2.1), protocolos MAC (seção 2.2), protocolos multicamada (seção 2.3) e segurança na comunicação (seção 2.4).

2.1 PRINCÍPIOS DE COMUNICAÇÃO

Para otimizar o consumo de energia dos protocolos MAC é necessário identificar e compreender em que situações há desperdício. As maiores fontes

de desperdício de energia em comunicação baseada em rádio definem diretrizes que são seguidas por virtualmente todos os MAC neste reino (LANGENDOEN; HALKES, 2005):

Escuta ociosa: se um nodo não sabe quando irá receber mensagens de um de seus vizinhos, ele deve manter seu rádio ligado em modo de recepção o tempo todo. Considerando uma aplicação na qual os nós sensores devem medir a temperatura do ambiente de dez em dez segundos, fazer uma média dos dados coletados e enviar o resultado uma vez por minuto. A transmissão de um pacote contendo a média é rápida, não levando mais que 5 ms. Desta forma, um nodo leva em média 5 ms transmitindo, 5 ms recebendo de outro nodo e 59990 ms ouvindo o canal sem que nada aconteça. Como o consumo de energia no modo de recepção é muito maior que no modo de *standby*, esta é a maior fonte de desperdício de energia e, consequentemente, o principal alvo de otimizações.

Colisões: podem ocorrer caso um nodo esteja dentro do raio de transmissão de dois ou mais nodos que estejam transmitindo ao mesmo tempo. Neste caso os dados são corrompidos, as transmissões devem ser repetidas e a energia gasta tanto nas transmissões quanto na recepção é desperdiçada. Além disso, a latência também aumenta.

Escuta desnecessária: como o canal de rádio é um meio compartilhado, um receptor pode ouvir pacotes que não são destinados a si e que são, portanto, descartados.

Pacotes de controle: o envio e recebimento de pacotes de controle, além de diminuir o número de pacotes de dados que podem ser transmitidos, também consome energia.

Existem algumas maneiras de se evitar o problema de *escuta ociosa*. A mais comum delas é utilizar um esquema de ciclo de trabalho, do inglês *duty cycle*, ilustrado na Figura 3, permitindo que os nodos durmam periodicamente. Toda vez que um nodo vai dormir, ele desliga seu rádio para economizar energia. Uma vez escolhido esse esquema o problema passa a ser a maneira de sincronizar os nodos para que eles acordem juntos, de forma a conseguir trocar pacotes.

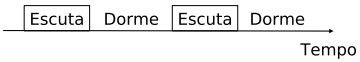


Figura 3: Esquema de *duty cycle* (YE; HEIDEMANN; ESTRIN, 2002).

Outra abordagem para diminuir o desperdício de energia com *escuta ociosa* é utilizar dois rádios em frequências diferentes, um para troca de dados e outro apenas para acordar os nodos (MAGNO et al., 2012). O rádio utilizado apenas para acordar os nós pode ter um design muito mais simples e, portanto, consome muito menos energia (JAIN; BISWAS; AGRAWAL, 2007). Entretanto não é comum que nodos sensores possuam dois rádios, consequentemente essa abordagem não é muito utilizada em RSSF.

Para evitar *colisões* de forma efetiva um protocolo MAC deve ser capaz de determinar precisamente se o canal está ocupado ou livre, Verificação de Canal Livre, do inglês *Clear Channel Assessment* (CCA). Uma maneira de se conseguir isso é utilizando um mecanismo baseado em Acesso Múltiplo com Detecção de Portadora, do inglês *Carrier Sense Multiple Access* (CSMA). Neste caso um nodo que deseja transmitir dados sempre escuta o canal antes de começar a transmissão. Caso o canal esteja ocupado o nodo posterga sua transmissão, evitando interferir com a transmissão em andamento. Caso o canal esteja livre o nó pode realizar sua transmissão. Como a utilização do CSMA não necessita de informações sobre nodos vizinhos, este mecanismo é uma boa opção para redes dinâmicas e com mobilidade.

Existe uma falha no mecanismo CSMA conhecida como problema do terminal oculto, ilustrado na Figura 4. Essa falha ocorre quando dois nodos, A e C, não se escutam, mas ambos são capazes de se comunicar com um terceiro nodo, B. Supondo que o nodo A esteja transmitindo para B e que C deseja iniciar uma transmissão. O nodo C escuta o canal, não detecta atividade, pois está fora do alcance de A, e começa a transmitir. Como A e C estão transmitindo ao mesmo tempo, eles interferem na transmissão um do outro e, consequentemente, o nodo B não é capaz de receber os dados corretamente.

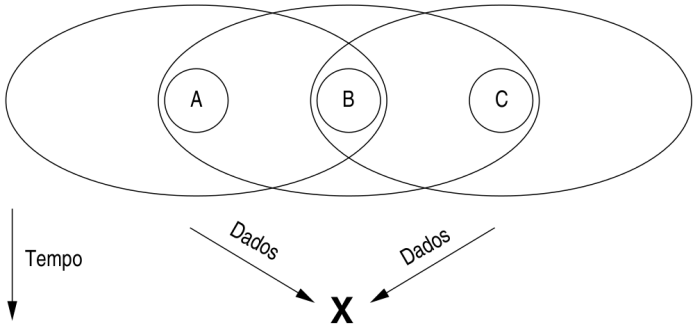


Figura 4: Problema do terminal oculto (WANNER, 2006).

Há duas formas de se solucionar o problema do terminal oculto: CSMA com detecção de colisão, do inglês *Carrier Sense Multiple Access with Collision Detection* (CSMA-CD), e CSMA com prevenção de colisão, do inglês *Carrier Sense Multiple Access with Collision Avoidance* (CSMA-CA).

O mecanismo CSMA-CD exige que os nós escutem a rede enquanto transmitem os dados. Caso uma colisão seja detectada, a transmissão é interrompida, um sinal anunciando a colisão é emitido e o nodo espera um tempo aleatório para tentar transmitir novamente. O mecanismo CSMA-CA utiliza pacotes de controle antes das transmissões, conforme ilustrado na Figura 5. Um transmissor emite uma solicitação para envio, do inglês *Request To Send* (RTS). Ao receber um RTS, um receptor responde que está livre para envio, do inglês *Clear To Send* (CTS). Desta forma, um terceiro nó, que também deseja transmitir, irá ouvir o CTS destinado a outro nodo, ou então não receberá um CTS em resposta à sua requisição. Em ambos os casos o terceiro nodo não inicia sua transmissão.

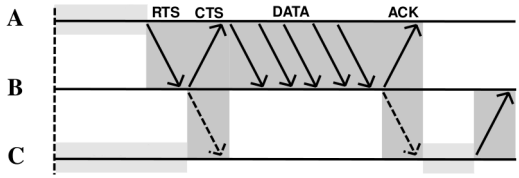


Figura 5: Mecanismo de RTS/CTS (DAM; LANGENDOEN, 2003).

Nodos sensores são desenvolvidos para serem baratos, portanto é comum que possuam dispositivos menos robustos. Desta forma, muitos nós sensores possuem rádios que não são capazes de transmitir dados e escutar o meio ao mesmo tempo. Consequentemente a solução CSMA-CD não serve para RSSF. O mecanismo CSMA-CA é utilizado em RSSF, todavia ele introduz o problema do terminal exposto, apresentado na Figura 6. O problema ocorre quando os sinais de controle silenciam nodos que não iriam interferir com a transmissão em andamento. No cenário ilustrado, as transmissões dos nodos A para B e D para C poderiam ocorrer simultaneamente. Entretanto, o nodo C é silenciado pelo CTS enviado por B, ficando exposto à transmissão de A para B. Consequentemente, D não recebe um CTS de C e não inicia a transmissão.

Outra maneira de evitar *colisões* é utilizar um mecanismo de Acesso Múltiplo com Divisão de Tempo, do inglês *Time Division Multiple Access* (TDMA). O TDMA divide o tempo em quadros, do inglês *frames*, e cada quadro é dividido em fatias, do inglês *slots*. Durante um *frame* cada nó é

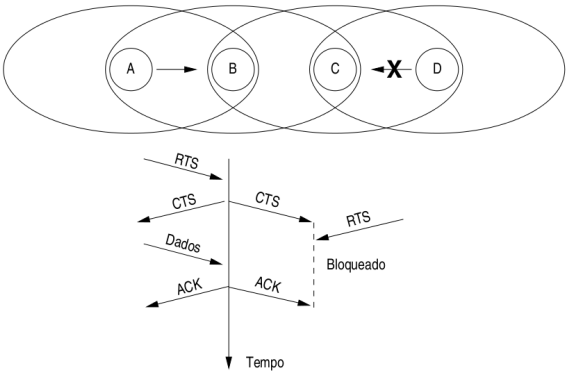


Figura 6: Problema do terminal exposto (WANNER, 2006).

designado a um *slot* no qual só ele tem o direito de transmitir. Como consequência não há colisões. Entretanto, para que este mecanismo funcione, todos os nós devem estar devidamente sincronizados. Além disso, o responsável pela alocação dos *slots* necessita de informações sobre a topologia da rede para fazer um *frame* funcional. Tanto a sincronização quanto a necessidade de conhecer a topologia da rede são características que introduzem *overhead* capazes de tornar essa solução menos atraente.

As abordagens aqui apresentadas e formas de diminuir o desperdício de energia com *escuta desnecessária* e *pacotes de controle* serão discutidas mais detalhadamente na próxima seção.

2.2 PROTOCOLOS DE CONTROLE DE ACESSO AO MEIO

Esta seção apresenta, em ordem cronológica, alguns protocolos MAC projetados especificamente para RSSF e seus mecanismos para otimizar o consumo de energia.

2.2.1 S-MAC

O protocolo *Sensor-MAC* (S-MAC) utiliza um esquema de *duty cycle* no qual os nodos da rede dormem e acordam periodicamente (YE; HEIDEMANN; ESTRIN, 2002). Todo período ativo tem a mesma duração e é dividido em duas fases: uma para sincronização e outra para trocar pacotes de dados que foram bufferizados enquanto o rádio estava desligado. A troca de

informações para sincronização em cada período ativo serve para evitar erros com taxas de desvio do relógio dos microcontroladores, do inglês *clock drift*.

Todos os nós da rede devem seguir uma agenda que determina seus períodos ativos e inativos. Durante a fase de inicialização, um nodo escuta o meio por uma certa quantidade de tempo. Caso não ouça a transmissão de um pacote de sincronização, contendo a agenda de outro nodo, o nó define sua própria agenda e a transmite para seus vizinhos. Caso contrário, o nodo define sua agenda para ficar igual à recebida e depois de esperar um tempo aleatório — para evitar colisões — retransmite aos seus vizinhos. Se ocorrer de um nodo com agenda já definida receber uma diferente, ele passa a seguir ambas.

Para fornecer uma sincronização mais robusta todos os *timestamps* trocados são relativos ao invés de absolutos. Além disso, o tempo de escuta é significativamente maior que a taxa de desvio do relógio. Isto faz com que a sincronização não seja tão crítica quando comparada com o mecanismo de TDMA, que utiliza *slots* muito menores.

Quando um nodo deseja transmitir dados, ele deve disputar o meio durante o período em que o nó destino está ativo. Essa disputa é realizada utilizando o mecanismo CSMA-CA explicado anteriormente. O nodo que mandar um RTS primeiro, ganha o meio. A Figura 7 apresenta o funcionamento do S-MAC em uma situação na qual dois nodos, A e C, querem transmitir e um terceiro nodo, B, não tem transmissões pendentes, mas acorda para ouvir o canal.

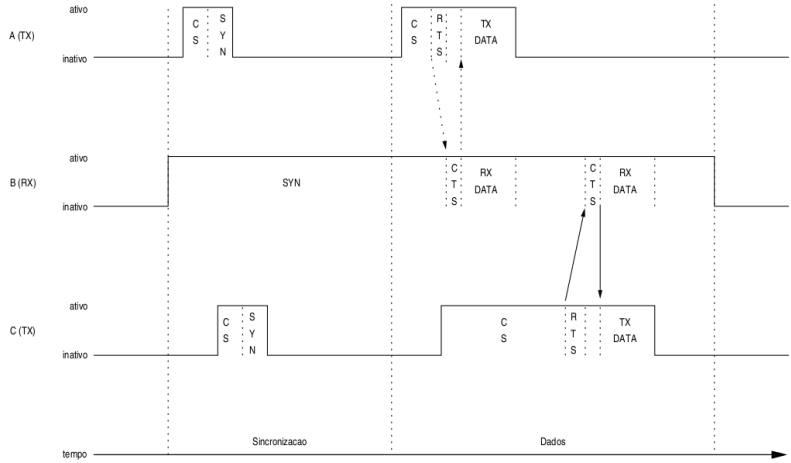


Figura 7: Funcionamento do S-MAC (WANNER, 2006).

2.2.2 T-MAC

O protocolo *Timeout-MAC* (T-MAC) segue a maioria das ideias apresentadas pelo S-MAC com algumas modificações (DAM; LANGENDOEN, 2003). A principal diferença é a utilização de um *duty cycle* adaptativo no qual os períodos ativos não tem uma duração fixa, variando de acordo com o tráfego da rede.

Diminuindo o tamanho do período ativo o T-MAC economiza mais energia quando comparado ao S-MAC. Quanto mais cedo terminar o período ativo, mais energia é economizada. Para isso, o protocolo utiliza um tempo limite, do inglês *timeout*. Enquanto está no período ativo, um nodo permanece ouvindo e possivelmente transmitindo. O período ativo termina quando nenhum evento de ativação ocorrer dentro de um tempo limite TA. Um evento de ativação pode ser, por exemplo, a recepção de qualquer dado pelo rádio, ou o término de uma transmissão do próprio nodo. Desta forma, a duração do TA determina a quantidade máxima de tempo que um nodo pode ter de escuta ociosa.

A Figura 8 apresenta uma comparação entre o funcionamento dos dois protocolos. As flechas indicam transmissões e recepções. Tanto o S-MAC quanto o T-MAC deslocam o tráfego de dados durante um período inativo para o próximo período ativo. O T-MAC finaliza um período ativo mais cedo caso não exista tráfego durante um período de tempo TA.

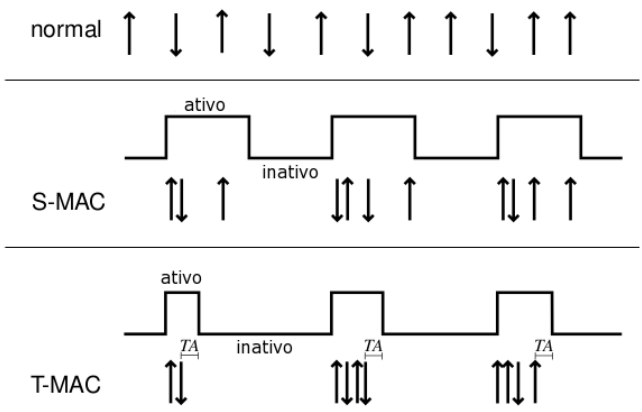


Figura 8: Funcionamento do T-MAC comparado com o S-MAC (DAM; LANGENDOEN, 2003).

2.2.3 B-MAC

O protocolo *Berkeley-MAC* (B-MAC) também utiliza um esquema de *duty cycle* no qual os nodos da rede dormem e acordam periodicamente (POLASTRE; HILL; CULLER, 2004). Entretanto, o B-MAC não segue o mesmo esquema de sincronização utilizado pelo S-MAC e T-MAC.

O protocolo utiliza uma técnica conhecida como escuta de baixa potência, do inglês *Low Power Listening* (LPL). Um nodo dorme a maior parte do tempo e acorda periodicamente para verificar se há alguma transmissão no canal. Essa verificação é de curta duração. Se há atividade, o nodo mantém o rádio ligado pelo tempo necessário para receber o pacote e volta a dormir depois da recepção. Caso nenhuma atividade seja detectada, o rádio é imediatamente desligado. Acontecendo de alguma atividade ser detectada mas nenhum pacote ser recebido (falso positivo), o nodo volta a dormir depois de um tempo limite. Para que as transmissões sejam sempre recebidas o protocolo utiliza um preâmbulo com duração maior ou igual ao período inativo. Por exemplo, se os nós da rede dormem por um segundo antes de acordarem para verificar o canal, o preâmbulo deve ter duração de pelo menos um segundo para que o receptor acorde, detecte atividade no canal e receba os dados.

De forma a evitar colisões, um nodo antes de transmitir verifica se o canal está livre, CCA. Caso o canal esteja ocupado o nodo espera por um tempo, *backoff*, e depois transmite. A Figura 9 apresenta o funcionamento do B-MAC em uma situação na qual dois nodos, A e C, tentam enviar a um terceiro receptor, B.

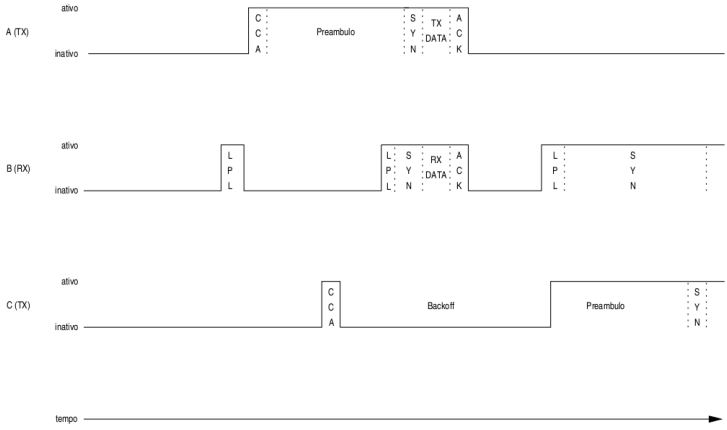


Figura 9: Funcionamento do B-MAC (WANNER, 2006).

2.2.4 X-MAC

Assim como o B-MAC o protocolo X-MAC também utiliza o esquema de LPL e sincronização por preâmbulo (BUETTNER et al., 2006). Entretanto ao invés de enviar um único preâmbulo com uma longa duração, o X-MAC envia uma sequência de preâmbulos curtos, cada um contendo o identificador do nodo destino. Entre cada transmissão de um preâmbulo curto, o nodo transmissor ouve o meio, com o objetivo de diminuir a quantidade de escuta desnecessária e o tempo total da transmissão. Quando um nodo acorda e recebe um preâmbulo curto, ele verifica se é o destino da transmissão. Caso não seja, o nodo volta imediatamente a dormir. Se o nodo for o destino, ele envia uma confirmação, *acknowledgement*, durante o período no qual o emissor está escutando. Quando um emissor recebe um *acknowledgement* de seu destino, ele para de enviar a sequência de preâmbulos e envia os dados. A Figura 10 apresenta uma comparação entre o funcionamento do B-MAC com o X-MAC.

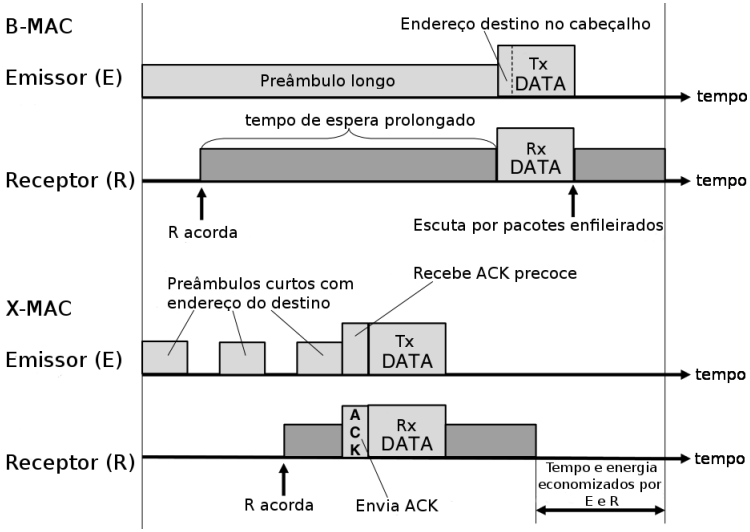


Figura 10: Funcionamento do X-MAC comparado com o B-MAC (BUETTNER et al., 2006).

2.2.5 IEEE 802.15.4 MAC Layer

O protocolo MAC proposto no padrão IEEE 802.15.4 possui dois modos básicos de comunicação: 1) *beacon-enabled*; e 2) *non-beacon* (IEEE Computer Society, 2006). O primeiro modo utiliza um mecanismo de sincronização baseado em *slots*, enquanto o segundo se resume a utilização do mecanismo CSMA-CA já apresentado.

O modo *beacon-enabled* utiliza uma estrutura de *superframe* ilustrada na Figura 11. Essa estrutura é definida por *beacons* — que são enviados por um nodo chamado coordenador — e é dividida em um período ativo e outro inativo (opcional). O período ativo inicia com a transmissão de um *beacon* e é divido em dezesseis *slots* de mesmo tamanho. Esses *slots* são divididos em duas partes: período de acesso disputado, do inglês *Contention Access Period* (CAP), e período livre de colisões, do inglês *Collision Free Period* (CFP). Durante o CAP os nodos competem entre si utilizando CSMA-CA. O CFP é formado utilizando *slots* garantidos, do inglês *Guaranteed Time Slots* (GTS), que são *slots* alocados pelo coordenador a nodos específicos, portanto nenhuma disputa pelo canal é realizada durante este período. O uso de GTSs é opcional e um coordenador pode alocar no máximo sete GTSs, sendo que um GTS pode durar mais que um *slot*.

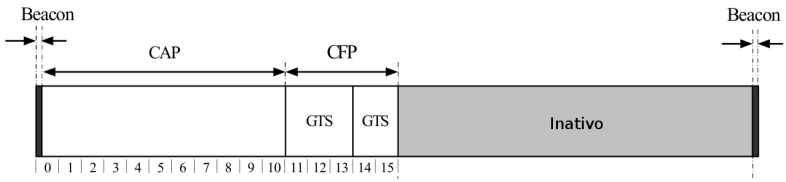


Figura 11: Estrutura do superframe do MAC IEEE 802.15.4 (IEEE Computer Society, 2006).

Um nodo escuta pelo *beacon* para saber o formato do *superframe* e saber se algum GTS foi alocado a ele. Em caso positivo ele espera até o seu *slot* para transmitir. Caso contrário, o nodo disputa pelo canal com os outros nodos durante o CAP.

2.2.6 Z-MAC

Zebra-MAC (Z-MAC) é um protocolo híbrido que combina CSMA e TDMA (RHEE et al., 2008). Ele utiliza um escalonamento TDMA, mas per-

mite que nodos disputem por outros *slots* utilizando CSMA.

Na fase inicial da rede, os nodos executam um algoritmo distribuído para alocação dos *slots*. Uma vez que a agenda TDMA está definida, cada nodo utiliza seu próprio *slot* para transmitir. Se um nodo precisar de mais de um *slot*, ele tenta "roubar" *slots* de seus vizinhos. Para utilizar um *slot* que não o seu, um nodo espera um tempo de *backoff* no início desse *slot*. Se ao término desse tempo de espera o *slot* continua sem ser utilizado, o nodo ganha o *slot* e começa a transmitir. Como os donos de um *slot* possuem a chance de transmitir antes, um nodo só pode "roubar" *slots* que não seriam utilizados.

2.3 PROTOCOLOS MULTICAMADA

Em RSSF os protocolos MAC possuem as informações mais precisas sobre as ligações entre nodos sensores, como Indicador de Força do Sinal Recebido, do inglês *Received Signal Strength Indicator* (RSSI), e Indicador de Qualidade da Conexão, do inglês *Link Quality Indicator* (LQI). Além disso, os protocolos MAC também podem fornecer informações de recursos como energia residual e memória livre dos nodos vizinhos, anexando essas informações em pacotes de dados. No entanto, os protocolos implementados seguindo as pilhas de comunicação tradicionais não aproveitam os benefícios diretos dessas informações que poderiam ser utilizadas para auxiliar decisões nas camadas superiores, como por exemplo descoberta e manutenção de rotas em protocolos de roteamento. Considerando os recursos limitados das RSSF, a otimização e o projeto conjunto das camadas de rede, ou seja, o projeto multicamada, se destaca como uma alternativa promissora à arquitetura tradicional de protocolos em camadas (MELODIA; VURAN; POMPILI, 2005).

Nesse sentido surgiram algumas abordagens multicamada para RSSF, entre elas:

2.3.1 RB-MAC

O protocolo *Receiver-Based-MAC* (RB-MAC) é diferente de todos os protocolos apresentados aqui pois um emissor não escolhe o nodo destino de sua transmissão (AKHAVAN; WATTEYNE; AGHVAMI, 2011). A rede funciona no padrão de interação todos pra um, no qual todos os dados são enviados para a estação base. O protocolo escolhe dinamicamente qual nó dentre os possíveis repassa os dados, baseado nas condições do canal e num *rank* calculado por um algoritmo de roteamento.

A Figura 12 apresenta o funcionamento do protocolo, utilizando como

rank a distância do nodo em relação à estação base. O nodo emissor, S, possui quatro vizinhos: A, B, C e D. Quando S deseja transmitir um pacote, ele transmite uma sequência de preâmbulos curtos diretamente seguidos pelos dados, sem atribuir um endereço destino. Cada preâmbulo curto informa quanto tempo falta para o início da transmissão dos dados e o *rank* de S. Os nodos A, B, C e D em algum momento acordam e escutam um preâmbulo curto. Com base nas informações obtidas do preâmbulo, cada nodo decide se é elegível para repassar os dados. Caso nenhum nodo vizinho de S participe dentro da janela de contenção (o emissor pode verificar por uma confirmação passiva ao ouvir o canal antes do final da janela), a sequência de preâmbulos é retransmitidas para garantir que ao menos um potencial receptor detecte o preâmbulo. O nodo D verifica que possui um *rank* pior que S e volta a dormir. Os nodos A, B e C recebem os dados e iniciam um período de *backoff* proporcional aos seus *ranks* para repassar os dados. Como o nodo A é o melhor candidato, seu período de *backoff* termina antes dos demais e ele inicia a transmissão. Os nodos B e C percebem a transmissão e voltam a dormir.

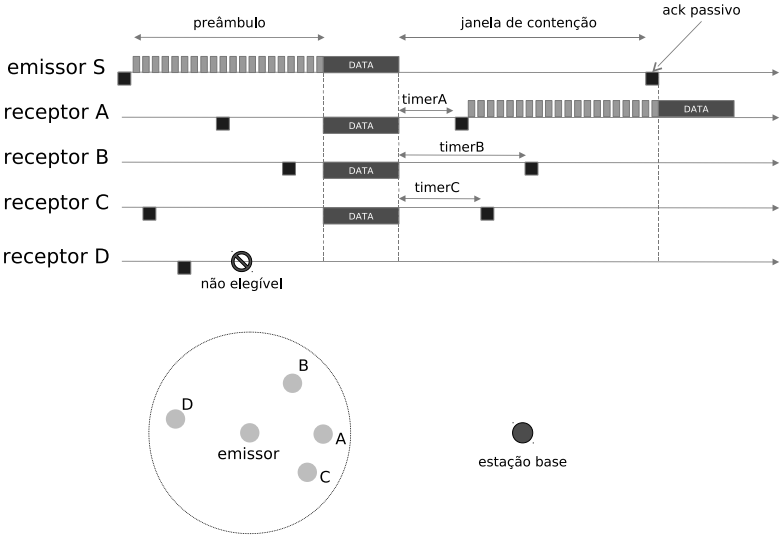


Figura 12: Funcionamento do RB-MAC.

2.3.2 EOA

O *Energy Optimization Approach* (EOA) (BAI et al., 2008) realiza a integração entre as camadas física, MAC e de roteamento. O EOA utiliza um algoritmo que calcula o nível de potência adequado para uma transmissão entre os nós. O protocolo de roteamento utiliza essa informação como métrica, escolhendo rotas com menor consumo de energia para encaminhar pacotes. Por fim, as informações de roteamento são utilizadas pelo MAC para formar uma agenda de períodos ativos e inativos.

2.4 SEGURANÇA NA COMUNICAÇÃO

Um canal de comunicação sem fio está aberto para qualquer um, permitindo a qualquer interface de rádio configurada na mesma frequência monitorar ou participar da comunicação em andamento. Isso fornece um meio muito conveniente para ataques (ZHOU; FANG; ZHANG, 2008).

Um atacante pode *escutar* os pacotes transmitidos e realizar criptoanálise ou análise de tráfego. Os pacotes que foram escutados podem ser *repetidos*, pelo atacante, num momento posterior ou em outra área da rede para gerar inconsistência. Pacotes falsos podem ser *injetados* na rede para confundir os nós sensores. Nós maliciosos também podem *modificar* pacotes recebidos antes de passá-los adiante.

A fim de evitar ataques indesejados, uma infraestrutura segura deve fornecer:

Confidencialidade: previne acesso não autorizado à informação. Pode ser obtida cifrando partes críticas de um pacote antes da sua transmissão. Desta forma somente os receptores autorizados podem acessar os dados, decifrando o pacote. O tipo de informação que deve ser cifrada depende da aplicação. Em alguns casos somente a parte de dados precisa ser cifrada, em outros o cabeçalho também é cifrado para proteger a identidade dos nodos.

Autenticidade: confirma a identidade de origem de uma mensagem. Pode ser obtida ao se assinar a mensagem. Todos os nós devem verificar se a mensagem recebida realmente vem do nodo indicado. Sem autenticação um atacante pode facilmente fraudar a identidade de outros nodos para transmitir informações falsas pela rede.

Integridade: garante que a mensagem recebida não foi alterada por qualquer razão, seja por um atacante ou simplesmente erros não desejados.

Pode ser obtida através de um código de soma de verificação, do inglês *checksum*. Sem a verificação de integridade, nós maliciosos podem modificar o conteúdo de um pacote antes de passá-lo adiante.

O restante desta seção apresenta duas infraestruturas seguras no contexto de RSSF.

2.4.1 TinySec

O TinySec (KARLOF; SASTRY; WAGNER, 2004) define uma arquitetura de segurança para RSSF na camada MAC, fornecendo cifragem e autenticação. Ele suporta dois modos diferentes de segurança: cifragem autenticada (TinySec-AE); e apenas autenticação (TinySec-Auth).

No modo de cifragem autenticada os dados são cifrados utilizando o bloco cifrador Skipjack (National Security Agency, 1998) e os pacotes autenticados com um código de autenticação de mensagem, do inglês *message authenticity code* (mac)[1]. O mac é computado sobre os dados cifrados e o cabeçalho do pacote. No modo de apenas autenticação, o TinySec autentica o pacote inteiro com um mac, mas os dados não são cifrados.

A inclusão de um mac para garantir autenticidade e integridade tem um custo de processamento, no uso do rádio e, consequentemente, no consumo de energia. Isso se deve ao fato de que valores *hash* geralmente representam uma longa sequência de bits — o comprimento de um mac determina o nível de segurança de uma função mac (SUN et al., 2010). O TinySec consegue baixo consumo de energia reduzindo o tamanho do mac, consequentemente diminuindo o nível de segurança fornecida.

2.4.2 MiniSec

O MiniSec (LUK et al., 2007) é um protocolo de segurança na camada de rede que além de cifragem e autenticação também fornece proteção contra ataques de repetição. Ele possui dois modos distintos de operação: um para comunicação *unicast* (MiniSec-U) e outro para *broadcast* (MiniSec-B). Ambos os modos utilizam o método de cifragem *Offset Codebook* (OCB) (ROGAWAY; BELLARE; BLACK, 2003), que prove tanto privacidade quanto autenticidade em apenas uma passagem sobre os dados da mensagem.

Os ataques de repetição são tratados de formas diferentes pelos dois modos de operação. No modo MiniSec-U, cada emissor e receptor mantém

[1] Termo em minúsculo para não causar confusão com a mesma sigla MAC de *Medium Access Control*.

um contador sincronizado que é utilizado como Vetor de Inicialização, do inglês *Initization Vector* (IV) — um bloco de bits utilizado para tornar a cifragem aleatória, produzindo distintos textos cifrados do mesmo texto simples. Desta forma todo pacote cujo contador possui valor igual ou menor que o mantido no receptor é descartado. Para reduzir o consumo de energia do rádio, apenas os últimos bits dos contadores são transmitidos junto com os pacotes. Um problema dessa abordagem é que ela exige a execução de custosas rotinas de ressincronização quando os contadores compartilhados ficam dessincronizados (devido à entrega de pacotes fora de ordem, por exemplo) (JINWALA et al., 2009). No modo MiniSec-B o tempo de vida da rede é dividido em épocas, evitando que pacotes de épocas anteriores sejam retransmitidos. Além disso, cada receptor utiliza um filtro que verifica o histórico dos pacotes dentro da época atual descartando pacotes já transmitidos.

3 FRAMEWORK PARA A GERAÇÃO DE PROTOCOLOS MAC PARA RSSF

Este capítulo apresenta a arquitetura original do C-MAC, a proposta de uma nova arquitetura para o *framework*, sua implementação, uma estrutura elaborada para facilitar a integração com outras camadas da pilha de comunicação e os mecanismos de configuração do novo C-MAC.

3.1 C-MAC - ARQUITETURA ORIGINAL

O Configurable MAC é um *framework* que permite a geração de protocolos de controle de acesso ao meio para RSSF. Sua arquitetura possui um núcleo básico sobre o qual diferentes protocolos podem ser compostos (WANNER, 2006). O *framework* é responsável por fornecer mecanismos que permitem selecionar, configurar e combinar estratégias de controle de acesso ao meio de acordo com os requisitos da aplicação. Cada estratégia é implementada por um componente de software e, uma vez selecionadas, essas estratégias em conjunto com o núcleo básico formam um protocolo MAC completo. Através desse paradigma é possível criar novos serviços de comunicação sob demanda e experimentar diferentes configurações de protocolo, coletando métricas para identificar qual a melhor configuração para determinada aplicação.

Com o intuito de fornecer uma ampla gama de pontos configuráveis que, quando combinados, formem um protocolo MAC completo, foi realizada uma avaliação das características dos principais protocolos MAC para RSSF (WANNER, 2006). É importante ressaltar que o C-MAC não suporta reconfiguração dinâmica. O *overhead* de manter todas as possibilidades de configuração em memória e a necessidade de um segundo protocolo para troca de configurações entre nodos torna o uso em tempo de execução do C-MAC uma opção muito onerosa. Os pontos de configuração da arquitetura original do C-MAC incluem:

- Configuração da camada física;
- Definição do período ativo;
- Mecanismo para evitar colisões;
- Mecanismo para detectar colisões; e
- Mecanismo para tratar colisões.

O C-MAC é implementado através de uma máquina de estados, ilustrada na Figura 13, cujas transições são ativadas por interrupções de um temporizador dedicado ou por interrupções do hardware de comunicação. Transições de um estado para ele mesmo não estão representadas. O nodo permanece no estado inicial (`OFF`) com seu rádio desligado. Após uma interrupção do temporizador o nodo liga o rádio e verifica se o canal está ocupado (`IDLE`). Um nodo com envio pendente, ao detectar o canal livre envia um preâmbulo, seguido por uma sequência de sincronização conhecida pelo receptor e por fim os dados (`SEND`). O nodo receptor quando detecta o preâmbulo passa a procurar pela sequência de sincronização (`SYNC`). Uma vez sincronizado, os dados são recebidos e verificados (`RECV`) e uma confirmação é enviada (`SEND_ACK`). Caso o transmissor não receba uma confirmação (`RECV_ACK`) ele espera um tempo de *backoff* para retransmitir.

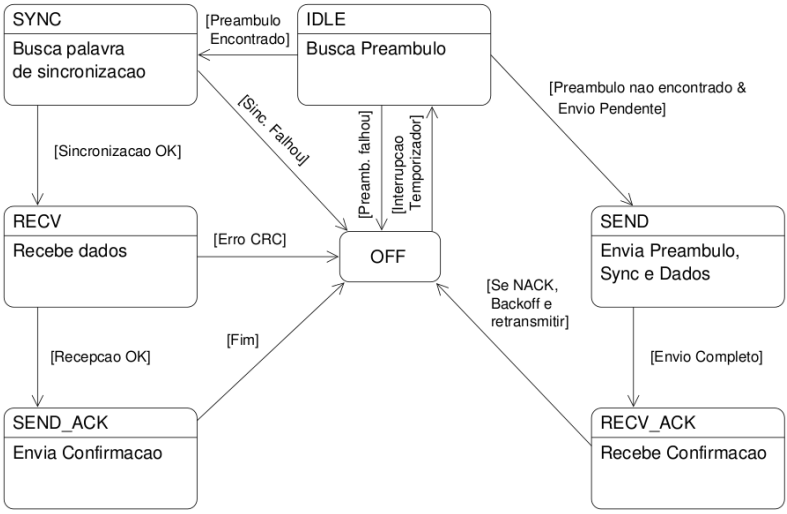


Figura 13: Máquina de estados do C-MAC original (WANNER, 2006).

É possível perceber que a máquina de estados do C-MAC se adequa bem a protocolos que se sincronizam via preâmbulo, como o B-MAC apresentado na Subseção 2.2.3. Contudo isso dificulta a implementação de protocolos que não possuem essa característica, como o S-MAC (Subseção 2.2.1), ou o IEEE 802.15.4 (Subseção 2.2.5), de forma que, para implementar esses protocolos, deve-se adicionar funcionalidades não previstas dentro dos componentes da arquitetura original (e.g. transmissão e recepção de informações de agendamento, transmissão de *beacons*). Além disso, essa arquitetura não

permite um alto reuso de seus componentes. Por exemplo, a transmissão de dados é tratada como um único grande componente (SEND) que, consequentemente, deve ser reimplementado para cada novo protocolo, pois mesmo que as operações de controle do rádio sejam iguais, características como cabeçalhos e mecanismos de tratamento de erro utilizados no empacotamento dos pacotes variam de um protocolo para outro. Outro exemplo é o uso de operações comuns como o envio e recebimento de mensagens RTS/CTS que não foram isoladas e devem ter seu código replicado nas diferentes implementações dos componentes SEND e RECEIVE que as utilizam.

3.2 NOVO DESIGN

O reprojeto aqui apresentado tem como objetivo tornar o C-MAC mais refinado, permitindo o reuso de microcomponentes em uma maior variedade de protocolos específicos de aplicação.

O ponto inicial desse novo design foi a decomposição de protocolos tradicionais com o intuito de se obter uma máquina de estados generalizada para as diferentes categorias de MAC. Existem várias nomenclaturas na literatura que classificam os protocolos MAC para RSSF em categorias diferentes de acordo com suas funcionalidades (STANKOVIC et al., 2003; LIN; WANG; SUN, 2004; BACHIR et al., 2010). Neste trabalho a nomenclatura adotada classifica os protocolos MAC em quatro categorias (KLUES et al., 2007): *channel polling*, *scheduled contention*, *time division multiple access* e híbrido.

Protocolos baseados em *channel polling* seguem uma máquina de estados similar à apresentada na Figura 14. O estado inicial é indicado pela seta apontando para ele a partir de qualquer lugar (SIPSER, 2006). Eles periodicamente ligam o rádio para verificar alguma atividade no canal (LISTEN). Se alguma atividade é detectada, o rádio é mantido ligado para receber um pacote. Caso contrário, o rádio é desligado imediatamente (OFF). Para transmitir um pacote, o nodo emissor primeiramente aguarda o canal ficar livre (BACKOFF, LISTEN) e então começa a transmitir um preâmbulo de sincronização (TX PREAMBLE). Receptores ouvindo o canal detectam atividade e utilizam o preâmbulo (RX PREAMBLE) para sincronizar a si mesmos com o emissor antes de receber o *payload* do pacote (RX DATA). É importante ressaltar que algumas transições só acontecem dependendo do protocolo. Por exemplo, uma vez que um receptor ouviu o preâmbulo, destinado a si, no X-MAC, ele manda uma confirmação para o emissor (TX ACK PREAMBLE) e somente então continua ouvindo para receber os dados. Caso contrário coloca o rádio em modo de baixo consumo. Entretanto, no protocolo B-MAC após receber o preâm-

bulo um nodo não tem outra escolha a não ser continuar ouvindo para receber os dados. Além disso, até mesmo a forma como o protocolo está configurado afeta as transições que podem ser tomadas. Por exemplo o B-MAC pode habilitar ou desabilitar o uso de confirmação, *backoffs* e até CCA.

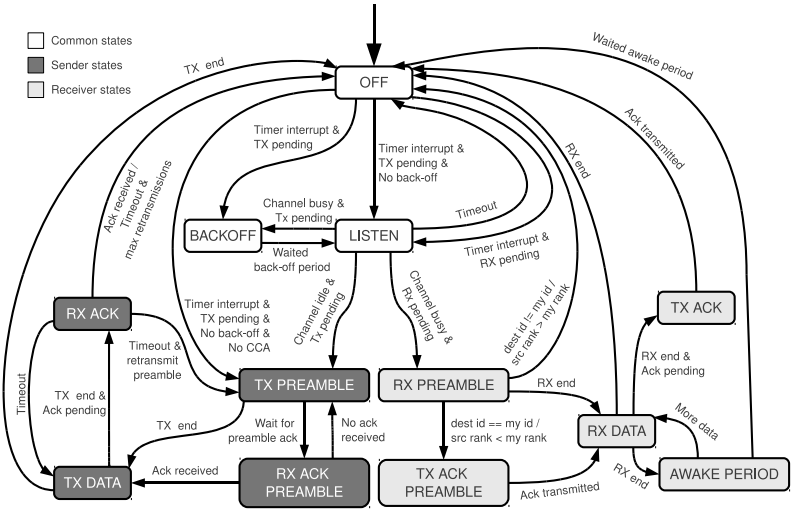


Figura 14: Máquina de estados dos protocolos da categoria *Channel Polling*.

Protocolos na categoria *scheduled contention* programam o tempo no qual nós vizinhos devem acordar para verificar atividade no canal. S-MAC e T-MAC são exemplos desse tipo de protocolo. Eles são generalizados na máquina de estados apresentada na Figura 15. No começo de cada período ativo, nós vizinhos trocam informações adicionais para se manter sincronizados, como o recebimento (`RX SYNC PKT`) e/ou transmissão (`TX SYNC PKT`) de itinerários no S-MAC. Depois disso, os nodos entram no estado `ACTIVE` e estão prontos para transmitir ou receber dados. Nodos dispostos a transmitir dados podem disputar pelo canal utilizando um mecanismo detecção de ocupação de canal (`CCA`) junto com RTS/CTS (`RX RTS, TX RTS, RX CTS, TX CTS`). Depois desses passos os nodos estão prontos para transmitir/receber com a possibilidade de usar pacotes de confirmação (`ACK TX, ACK RX`).

Protocolos baseados em TDMA também programam o tempo no qual os nodos devem acordar para ouvir o canal. A diferença com os protocolos do tipo *scheduled contention* está no fato de que cada nodo, ao invés de um grupo de nodos, possui um *slot* de tempo específico para transmitir. Desta forma, os nodos não precisam disputar pelo canal e não ocorrem colisões. Entretanto,

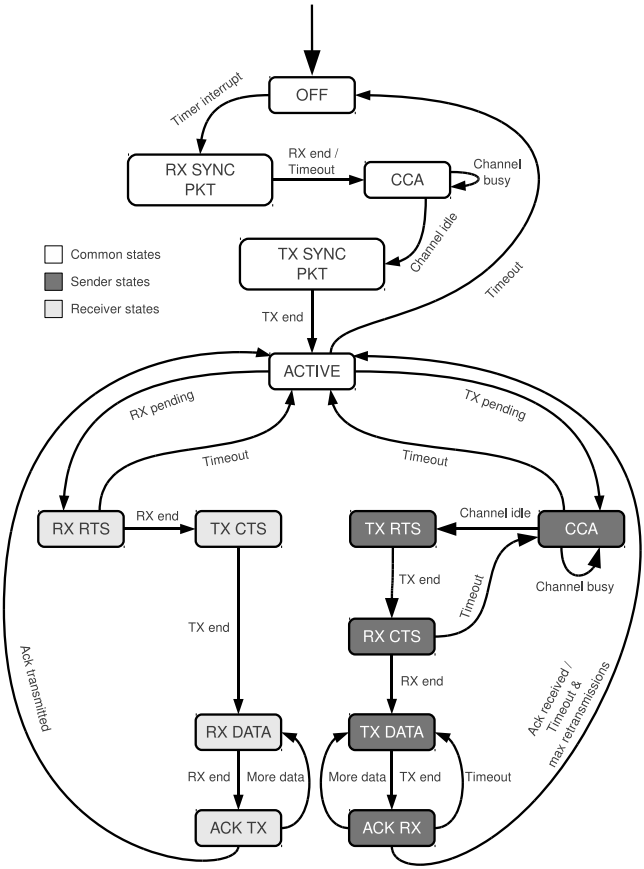


Figura 15: Máquina de estados dos protocolos da categoria *Scheduled Contention*.

se livrar de colisões vem com o preço de menor vazão. Uma vez que um nodo só pode transmitir durante seu próprio *slot*, ele deve ficar em silêncio mesmo se outros nodos não estão transmitindo em seus *slots*. Além disso, esse tipo de protocolo é sensível a mudanças na topologia da rede, necessitando uma realocação dos *slots* sempre que isso acontece. A Figura 16 ilustra a máquina de estados generalizada para protocolos baseados em TDMA. Essa máquina de estados é bastante semelhante à máquina dos protocolos *scheduled contention*, no entanto os passos de sincronização são diferentes caso um nodo seja mestre (i.e. o coordenador de uma rede, responsável por alocar os *slots* de tempo) ou escravo, que é uma configuração comum neste tipo de protocolo.

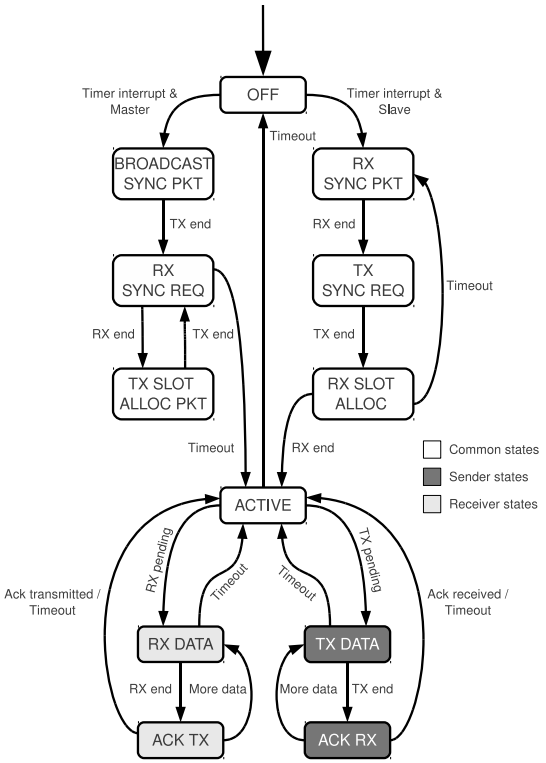


Figura 16: Máquina de estados dos protocolos da categoria TDMA.

Uma análise criteriosa das máquinas de estados apresentadas levou a uma nova máquina de estados do C-MAC, apresentada na Figura 17. Cada estado representa um microcomponente que pode ter diferentes implementações. Esses microcomponentes juntamente com as transições de estados po-

dem ser combinados para produzir protocolos específicos para cada aplicação. Estados tracejados representam estados compostos, conceito presente nos diagramas de estados da *Unified Modeling Language* (UML) (OMG, 2009) — um estado composto encapsula estados e transições que trabalham em conjunto para um objetivo comum fornecendo uma melhor visualização. Utilizando técnicas de metaprogramação estática, microcomponentes representando estados que não fazem sentido para um determinado protocolo podem ser completamente removidos. Quando um estado é removido, suas transições de entrada são encaminhadas para o estado alvo de suas transições de saída, ainda mantendo a semântica original da transição.

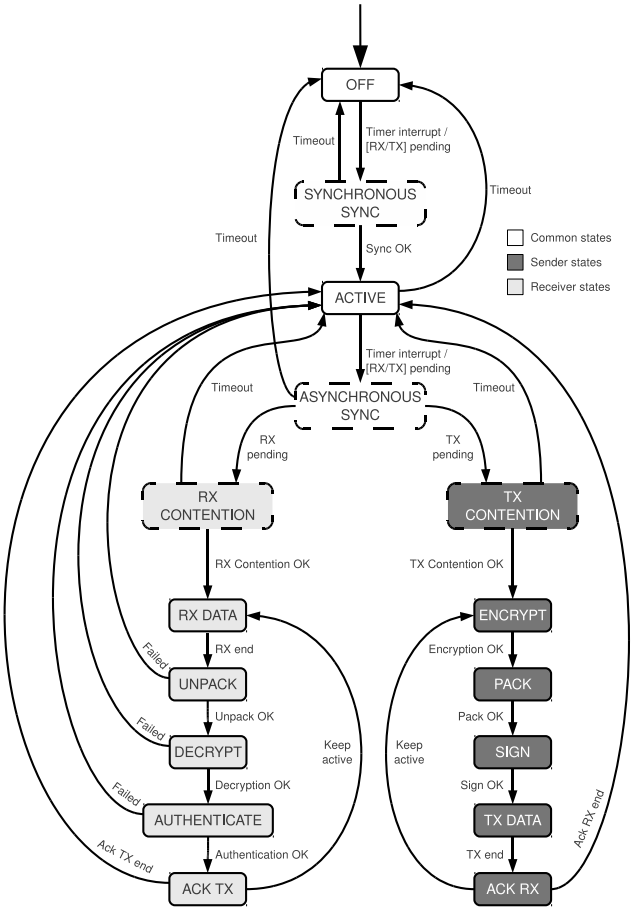


Figura 17: Nova máquina de estados do C-MAC.

A máquina de estados do C-MAC pode ser ativada por eventos de emissão/recepção (i.e. quando o protocolo alvo tem um ciclo de trabalho completo) ou periodicamente por eventos temporais (i.e. quando um ciclo de trabalho dormir/ativo é necessário). O protocolo permanece no estado `OFF`, com o rádio desligado, até que um dos eventos citados anteriormente ative a transição para o estado `SYNCHRONOUS SYNC`.

Os estados que compõem o estado composto `SYNCHRONOUS SYNC` são apresentados na Figura 18. Esses estados são relacionados a operações usadas para sincronizar o ciclo de trabalho e foram obtidos através da fusão dos passos de sincronização apresentados nas máquinas de estados dos protocolos *scheduled contention* e TDMA (Figuras 15 e 16). Um nodo pode começar a sincronização transmitindo (`BROADCAST SYNC PKT`) um pacote contendo informações de sincronização (e.g. seu itinerário em um protocolo *scheduled contention*) que é então seguido pela recepção de informação dos outros nodos (`RX SYNC PKT`) (como pedidos de alocação de slots em um protocolo TDMA). Neste ponto, um nodo pode tanto terminar a sua sincronização (estados em negrito pertencem à visão *top view* da máquina de estados do C-MAC) ou executar trocas de informações adicionais (`TX SYNC PKT, RX SYNC RESP`) com a possibilidade de usar um mecanismo de contenção para evitar colisões nesse processo (`CCA`). Note que no mesmo protocolo cada estado pode ser habilitado, desabilitado, ou executar diferentes operações dependendo da configuração do nodo (e.g. o nodo pode ser mestre, escravo, ou ambos em um protocolo TDMA). A Figura 19 ilustra três diferentes configurações do estado composto `SYNCHRONOUS SYNC`. Após o nodo estar devidamente sincronizado, ele vai para o estado `ACTIVE`. Se existe algum pedido de transmissão/recepção pendente o protocolo vai para os estados correspondentes, caso contrário vai para o estado `OFF` no final do seu ciclo ativo.

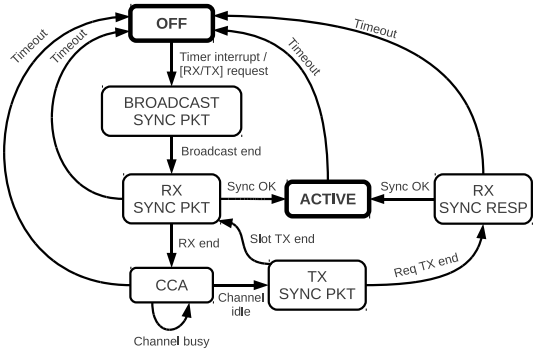


Figura 18: Estado composto `SYNCHRONOUS SYNC`.

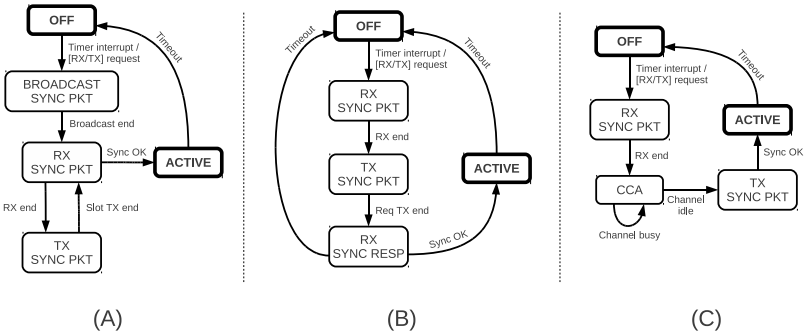


Figura 19: Estado composto `SYNCHRONOUS SYNC` configurado como: (A) *TDMA Master*; (B) *TDMA Slave*; e (C) *Scheduled Contention*.

O estado composto `ASYNCHRONOUS SYNC`, ilustrado na Figura 20, está presente para protocolos que não necessitam possuir um ciclo de trabalho totalmente sincronizado (e.g. B-MAC, X-MAC). A sincronização é feita através da transmissão de um longo preâmbulo, ou uma sequência de curtos preâmbulos. O resultado é um subconjunto dos estados presentes na máquina de estados dos protocolos *channel polling* (Figura 14), já que esse mecanismo é parte desse tipo de protocolo.

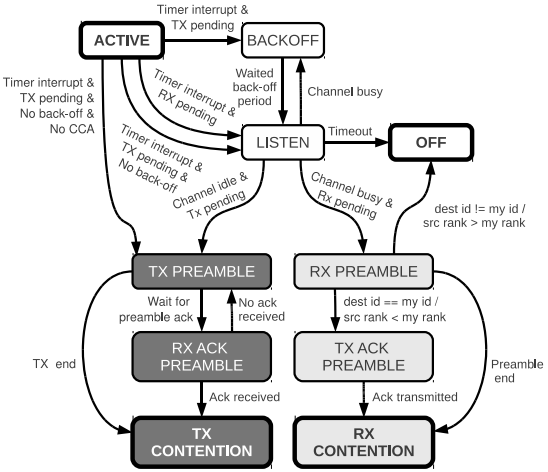


Figura 20: Estado composto `ASYNCHRONOUS SYNC`.

Antes de transmitir ou receber dados os nodos podem passar por mecanismos de contenção de forma a evitar colisões. Esses mecanismos são definidos pelos estados compostos `RX CONTENTION` e `TX CONTENTION` cujas máquinas de estados são ilustradas nas Figuras 21 e 22, respectivamente. Os estados foram projetados para suportar um mecanismo de detecção de ocupação de canal (`CCA`), RTS/CTS (`RX RTS, TX RTS, RX CTS, TX CTS`), ou ambos.

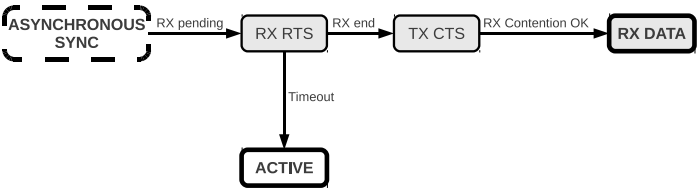


Figura 21: Estado composto `RX CONTENTION`.

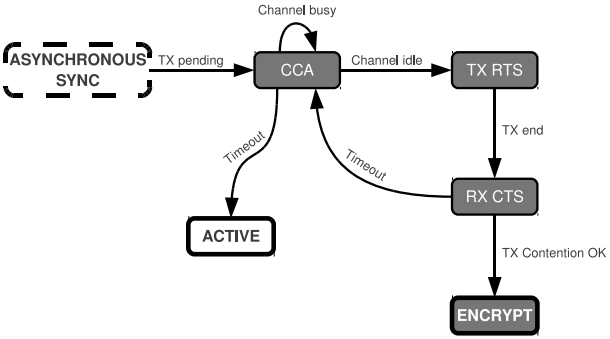


Figura 22: Estado composto `TX CONTENTION`.

Após passar pelos mecanismos de contenção os nodos estão prontos para transmitir ou receber. Quando dados são recebidos (`RX DATA`), eles passam pelos mecanismos de tratamento de erros (`UNPACK`) e segurança (`DECRYPT` e `AUTHENTICATE`) e um pacote de confirmação pode ser transmitido (`ACK TX`). No lado da transmissão, informações para tratamento de erros (`PACK`) e segurança (`ENCRYPT` e `SIGN`) são acrescentadas antes da transmissão (`TX DATA`). O estado `ACK RX` implementa a recepção de pacotes de confirmação. Alguns protocolos permitem a transmissão de pacotes de dados em rajadas (e.g. X-MAC e S-MAC) sem que o nodo tenha que competir pelo canal novamente, o que exigiu as transições `Keep active`, ilustradas na Figura 17.

Através dessas novas máquinas de estados foi possível expandir o C-MAC e fornecer uma gama maior de pontos configuráveis, enquanto alcançando um maior nível de reuso. Os principais pontos de configuração do C-MAC agora incluem:

Configuração da camada física: esses são os pontos de configuração definidos pelo transceptor (e.g. frequência, potência de transmissão, taxa de dados).

Sincronização e organização: fornece mecanismos para enviar ou receber informações de sincronização para organizar a rede e sincronizar os ciclos de trabalho dos nodos.

Mecanismo anti-colisão: define os mecanismos de contenção usados para evitar colisões. Pode ser constituído por um algoritmo de detecção da ocupação do canal (e.g. CSMA-CA), a troca de pacotes de contenção (RTS/CTS), ou uma combinação de ambos.

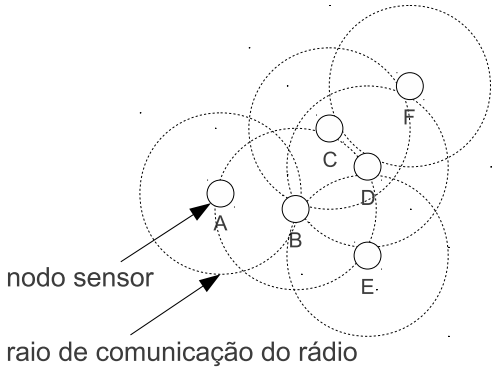
Mecanismo de confirmação: a troca de pacotes de confirmação para determinar se a transmissão foi bem sucedida, incluindo confirmação de preâmbulos.

Tratamento de erro e segurança: determina quais mecanismos serão utilizados para garantir a consistência (e.g. checagem de *Cyclic Redundancy Check* (CRC)) e a segurança dos dados (e.g. cifragem, autenticação).

3.3 INTEGRAÇÃO COM OUTRAS CAMADAS

Para permitir a integração entre o C-MAC e outras camadas da pilha de comunicação foi criada uma estrutura chamada *Neighborhood*. Essa estrutura guarda informações, coletadas em tempo de execução, sobre os nós vizinhos de um determinado nodo da rede — são considerados vizinhos de um nodo os nós dentro do raio de comunicação de seu rádio. Toda vez que um nodo receber ou transmitir dados, as informações sobre o vizinho com o qual está interagindo são atualizadas pelo C-MAC. A Figura 23 apresenta as informações mantidas pelo nodo B, sobre seus vizinhos, no cenário ilustrado.

O C-MAC insere registros sobre os nós vizinhos — informações sobre RSSI, LQI e energia residual — e os protocolos das camadas superiores utilizam esses dados da forma que melhor lhes convêm. Por exemplo, um protocolo de roteamento pode escolher um caminho passando por nodos com maior energia residual, um caminho onde o sinal é mais forte, ou um caminho



address	energy	LQI	RSSI	expire
0xE	10 mAh	220	- 5 dBm	5
0xC	12 mAh	152	- 15 dBm	7
0xD	20 mAh	150	-16 dBm	8
0xA	17 mAh	215	-6 dBm	10

Figura 23: Cenário de uso da estrutura *Neighborhood* para o nodo B.

com melhor qualidade de sinal. Note que o RSSI não caracteriza a qualidade de um sinal; os valores de LQI estão associados a uma taxa de perda de pacotes, do inglês *Packet Error Rate* (PER), que é uma relação entre os pacotes recebidos com erro e o número total de pacotes recebidos. Além disso, um protocolo de roteamento também pode remover o registro de um determinado vizinho caso nenhum dado tenha sido trocado dentro de um período de tempo (expire == 0) — remoção de rota não utilizada.

3.4 IMPLEMENTAÇÃO

O novo C-MAC foi implementado como uma abstração do Sistema Operacional Paralelo e Embarcado, do inglês *Embedded Parallel Operating System* (EPOS) (FROHLICH; SCHRODER-PREIKSCHAT, 1999a, 1999b, 1999c; FROHLICH, 2001), possibilitando o uso do *framework* em diferentes arquiteturas de hardware sem a necessidade de realizar alterações no seu código fonte.

Para cada estado da nova máquina de estados do C-MAC (apresentada nas Figuras 17, 18, 20, 21, e 22) existe um conjunto de microcomponentes

dentre os quais, de acordo com a configuração desejada, um é selecionado em tempo de compilação. Cada microcomponente é implementado através de uma classe. Contudo todos os microcomponentes herdam de uma única classe, de forma que todos obedecem a uma mesma interface.

A Figura 24 apresenta o diagrama UML simplificado da implementação do novo C-MAC. O estado `TX PREAMBLE`, pertencente ao estado composto `ASYNCHRONOUS SYNC`, é utilizado como exemplo para explicar a implementação do *framework*, de modo que classes referentes a outros estados não estão ilustradas. A classe `CMAC_State` define a interface que é herdada e implementada por todos os microcomponentes que compõem o objeto `CMAC`. Esta interface se resume a um método chamado `execute` que recebe uma transição da máquina de estados (`CMAC_STATE_TRANSITION`) como parâmetro e retorna uma transição do mesmo tipo. De acordo com a configuração desejada um dos três microcomponentes (`Tx_Dummy_Preamble`; `Tx_Time_Rank_Preamble`; e `Tx_Id_Preamble`) é escolhido como implementação do estado `TX PREAMBLE` através do metaprograma ilustrado na Figura 25. Repare que existe um `typedef`, de forma que não se utiliza o nome de um microcomponente individual, mas sim o nome do estado que ele representa, no caso `Tx_Preamble`. O mesmo é feito com os outros estados. Como todos os microcomponentes possuem a mesma interface, utilizando `typedef` é possível implementar máquinas de estados genéricas, ilustradas nas Figuras 26 e 27, que são resolvidas em tempo de compilação. Ainda na Figura 27 é possível perceber como funciona a transição de um estado para o outro. Após executar, o microcomponente `Asynchronous_Sync` retorna uma transição da máquina de estados, `result`, que é utilizada para decidir qual o próximo estado.

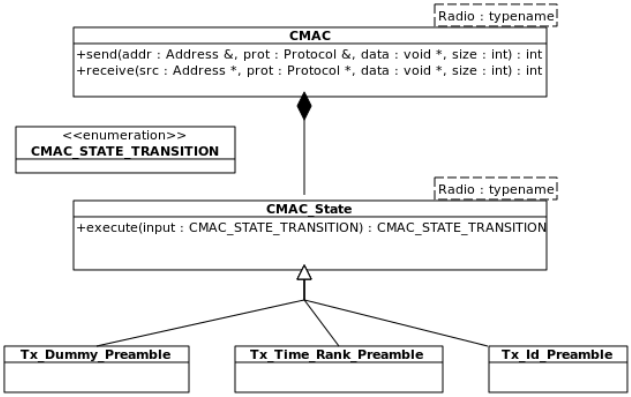


Figura 24: Diagrama UML simplificado do novo C-MAC.

```
// ...
typedef typename
    SWITCH<Traits<CMAC<T> >::PREAMBLE_CONTENT,
        CASE<Traits<CMAC<T> >::DUMMY,
            Tx_Dummy_Preamble<T>,
        CASE<Traits<CMAC<T> >::TIME_TO_DATA_AND_RANK,
            Tx_Time_Rank_Preamble<T>,
        CASE<Traits<CMAC<T> >::DESTINATION_ID,
            Tx_Id_Preamble<T>,
        CASE<DEFAULT,
            Tx_Dummy_Preamble<T>
        > > > >
    >::Result Tx_Preamble;
// ...
```

Figura 25: Metaprograma para decidir qual microcomponente a ser utilizado pelo estado `TX PREAMBLE`.

```
// ...
while ((state != TX_CONTENTION) && (state != RX_CONTENTION)) {
    switch (state) {
        case BACKOFF:
            // ...
            result = Backoff::execute(result);
            // ...

        case LISTEN:
            // ...
            result = Listen::execute(result);
            // ...

        case TX_PREAMBLE:
            // ...
            result = Tx_Preamble::execute(result);
            // ...

        case RX_ACK_PREAMBLE:
            // ...
            result = Rx_Ack_Preamble::execute(result);
            // ...
        // ...
    }
}
// ...
```

Figura 26: Código correspondente a maquina de estados do estado composto `ASYNCHRONOUS SYNC`.

```
// ...
while (_state != OFF) {
    switch (_state) {
        case SYNCHRONOUS_SYNC:
            // ...
            result = Synchronous_Sync::execute(result);
            // ...

        case ACTIVE:
            // ...
            result = Active::execute(result);
            // ...

        case ASYNCHRONOUS_SYNC:
            {
                if (Traits<CMAC<Radio_Wrapper> >::SYNC ==
                    Traits<CMAC<Radio_Wrapper> >::PREAMBLE) {

                    result = Asynchronous_Sync::execute(result);

                    if (result == RX_PENDING)
                        _state = RX_CONTENTION;
                    else if (result == TX_PENDING)
                        _state = TX_CONTENTION;
                    else if (result == TIMEOUT)
                        _state = OFF;
                }
                break;
            }
        case RX_CONTENTION:
            // ...
            result = Rx_Contention::execute(result);
            // ...

        case RX_DATA:
            // ...
            result = Rx_Data::execute(result);
            // ...

        case UNPACK:
            // ...
            result = Unpack::execute(result);
            // ...
        // ...
    }
}
// ...
```

Figura 27: Código correspondente a nova maquina de estados do C-MAC.

Quando um estado não faz sentido para determinada configuração seu código é completamente removido. Por exemplo, protocolos que não se sincronizam via preâmbulo, como o S-MAC e o IEEE 802.15.4, não utilizam o estado composto `ASYNCHRONOUS SYNC`. Repare que na máquina de estados apresentada na Figura 27 existe um `if` dentro do `case` do estado `ASYNCHRONOUS SYNC`. Esse `if` verifica o valor de uma constante e, portanto, pode ser resolvido em tempo de compilação. Assim, caso não se utilize sincronização via preâmbulo esse trecho de código não é gerado e não adiciona *overhead* ao código final. O mesmo é feito com os outros estados.

A Figura 28 apresenta o diagrama UML da implementação da estrutura de integração com outras camadas. A classe *Neighborhood* segue o padrão de projeto *Singleton* (GAMMA et al., 1995), garantindo a existência de apenas uma instância da classe com um ponto global de acesso ao seu objeto. Dessa forma, todas as camadas da pilha de comunicação podem compartilhar o uso do mesmo objeto. Isso evita a duplicação de dados e diminui o número de mensagens trocadas entre camadas, reduzindo consumo de memória e tempo de processamento. Além disso, a pilha de comunicação na qual o C-MAC foi implementado utiliza o padrão de projeto *Observer* (GAMMA et al., 1995), de forma que ao receber um pacote o C-MAC automaticamente notifica a camada superior.

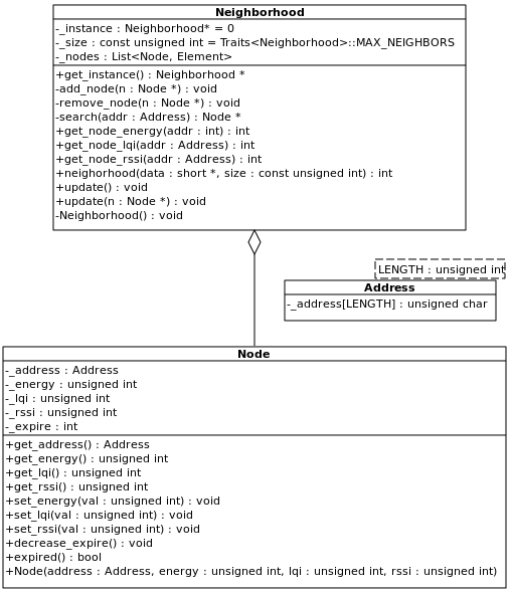


Figura 28: Diagrama UML da estrutura de integração entre camadas.

A utilização da estrutura *Neighborhood* é opcional, podendo ser habilitada ou não. Também é possível definir quanto tempo leva para um registro expirar e o número máximo de vizinhos sobre os quais serão mantidos registros ao mesmo tempo. Registros são mantidos em uma lista. Quando a lista atinge o tamanho máximo e informações sobre um novo nodo são recebidas, o primeiro registro é removido e informações sobre o novo nodo são inseridas no final da lista. Ao se atualizar informações de um nodo já registrado, seu registro vai para o final da lista.

3.5 CONFIGURAÇÃO

A configuração do C-MAC é realizada utilizando o mesmo conceito de *Traits* presente na biblioteca padrão da linguagem C++ (STROUSTRUP, 2000). *Traits* são classes parametrizadas nas quais atributos constantes e estáticos descrevem as propriedades de um determinado tipo (FROHLICH, 2001). Utilizando um arquivo de configuração, programadores podem escolher quais propriedades o protocolo deve possuir. Quando uma propriedade é selecionada, a funcionalidade que ela descreve é incluída no protocolo. Dessa forma é possível definir qual o comportamento do protocolo em tempo de compilação. Devido ao uso de metaprogramação estática e *inlining* de funções, propriedades não selecionadas não adicionam *overhead* ao código gerado.

A Figura 29 ilustra um trecho do arquivo de configuração. O *traits* do C-MAC está dividido em duas partes: primeiro estão listadas as opções de configuração e depois os atributos que definem qual será o comportamento do protocolo. Entre as opções é possível optar por um esquema sem sincronização (`NO_SYNC`), ou com sincronização: via preâmbulo (`PREAMBLE`) ou *beacons* (`BEACON`). O preâmbulo pode ser constituído sem informações úteis (`DUMMY`); conter informações sobre quando inicia a transmissão dos dados e o rank do emissor (`TIME_TO_DATA_AND_RANK`); ou conter o endereço do nodo destinatário (`DESTINATION_ID`). No quesito segurança, os dados podem permanecer em sua forma original (`PLAINTEXT`); ser cifrados (`DATA_ENCRYPTION`); autenticados (`AUTHENTICATION_ONLY`); cifrados e autenticados (`AUTHENTICATION_ENCRYPTION`); ou ainda cifrados, autenticados e validados temporalmente (`TEMPORAL_AUTHENTICATION_ENCRYPTION`)[1]. Também é possível não utilizar nenhum mecanismo para confirmação de recebimento de pacotes (`NO_CONFIRMATION`); utilizar pacotes de confirmação (`ACK`); ou utilizar um mecanismo de janela de contenção com confirmação passiva (`PASSIVE_CONTENTION_WINDOW`). O atributo `time_triggered` de-

[1] Mais informações sobre essa opção são apresentadas no Capítulo 4, Subseção 4.2.4.

finido como `true` faz com que o protocolo seja ativado por interrupções do temporizador, ao invés de ser ativado pelos métodos de envio e recebimento da aplicação. Já o atributo `auto_rx`, que é dependente do atributo `time_triggered`, faz com que o protocolo escute o canal periodicamente de forma automática.

3.5.1 Conflitos

O C-MAC conhece as características particulares de cada um de seus microcomponentes, pois conhece seus atributos de classe. Contudo, o *framework* não tem conhecimento sobre as relações entre microcomponentes e, portanto, não pode garantir a consistência dos protocolos que produz. Por exemplo, caso o usuário opte por utilizar sincronização via preâmbulo (`SYNC = PREAMBLE`) e defina a duração do preâmbulo (`PREAMBLE_LENGTH`) com um valor menor que o período inativo (`SLEEPING_PERIOD`), pode ocorrer de os nodos da rede não conseguirem se sincronizar. Outro exemplo pode ser visto na Figura 29, onde se define a utilização de um algoritmo cifrador (`CIPHER = AES`) ao mesmo tempo em que se define o uso de texto plano (`SECURITY = PLAINTEXT`), de forma que nenhum dado será cifrado.

Para o *framework* produzir um protocolo funcional é necessário que suas opções de configuração estejam definidas coerentemente. Isso pode ser verificado, antes do processo de compilação, por um conjunto de regras que especifiquem dependências e restrições entre microcomponentes, de modo que configurações inválidas possam ser detectadas e rejeitadas. No entanto, a definição e utilização dessas regras está fora do escopo deste trabalho, deixando a responsabilidade de verificação das configurações ao programador da aplicação.

```
// ...
template <> struct Traits<CMAC<Radio_Wrapper> >: public Traits<void>
{
    // Options
    enum Synchronization {
        NO_SYNC  = 0,
        PREAMBLE = 1,
        BEACON   = 2,
    };

    enum Preamble_Content {
        DUMMY                  = 0,
        TIME_TO_DATA_AND_RANK = 1,
        DESTINATION_ID         = 2,
    };
// ...
    enum Security {
        PLAINTEXT                           = 0,
        DATA_ENCRYPTION                     = 1,
        AUTHENTICATION_ONLY                 = 2,
        AUTHENTICATION_ENCRYPTION           = 3,
        TEMPORAL_AUTHENTICATION_ENCRYPTION = 4,
    };
// ...
    enum Confirmation {
        NO_CONFIRMATION           = 0,
        ACK                       = 1,
        PASSIVE_CONTENTION_WINDOW = 2,
    };

    // Settings
    static const bool time_triggered = true;
    static const bool auto_rx        = true;

    static const int SYNC = PREAMBLE;

    static const int PREAMBLE_CONTENT = TIME_TO_DATA_AND_RANK;
    static const int PREAMBLE_LENGTH  = 1000; // ms
    static const int SLEEPING_PERIOD  = 1000; // ms
// ...
    static const int SECURITY = PLAINTEXT;
    static const int CIPHER   = AES;
// ...
};
// ...
```

Figura 29: Configurabilidade do protocolo pelo *Traits*.

4 AMBIENTES E RESULTADOS EXPERIMENTAIS

A avaliação do novo C-MAC foi realizada com o objetivo de corroborar o design apresentado no Capítulo 3. Diferentes cenários de aplicações foram imaginados com o intuito de testar algumas das diversas possibilidades de configurações do *framework*. Foram feitas comparações das diferentes configurações entre si e com trabalhos relacionados, levando em consideração parâmetros como consumo de memória, consumo de energia, taxa de transferência, latência e tempo de processamento. É importante ressaltar que o principal objetivo aqui não é mostrar que determinado protocolo é bom, mas sim mostrar que o novo *framework* possibilita um alto reuso dos seus microcomponentes para gerar diferentes protocolos, alguns dos quais não poderiam ser gerados utilizando a arquitetura original, e todos eles possuem desempenho comparável com trabalhos relacionados. Outro objetivo é mostrar que não existe um protocolo ótimo para RSSF e são fatores como nodos utilizados, ambiente de implantação e características da aplicação que determinam qual a melhor opção para cada caso. Os experimentos realizados neste trabalho utilizaram nodos sensores reais EPOSMote I e II.

4.1 EPOSMOTE I

O EPOSMote I, ilustrado na Figura 30, é uma plataforma de pequeno porte e baixa potência com foco em aplicações industriais (LISHA, 2012). O seu hardware é baseado na combinação do microcontrolador ATmega1281 — arquitetura AVR 8-bit, com 128kB de memória flash e 8kB de RAM — com o transceptor AT86RF230 compatível com o padrão IEEE 802.15.4. Seu sensor SHT11 permite a medição da temperatura e umidade do ambiente no qual está implantado. Devido ao seu tamanho, essa arquitetura não possui interfaces de fácil acesso para programação e deve ser programada em laboratório.

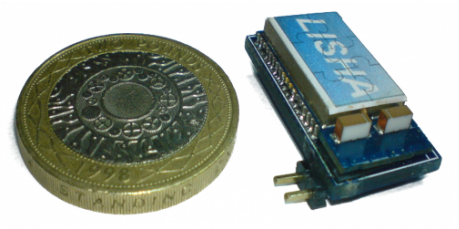


Figura 30: EPOSMote I lado a lado com uma moeda de £2.

4.1.1 IEEE 802.15.4

O C-MAC foi avaliado na plataforma EPOSMote I variando os seguintes parâmetros do MAC IEEE 802.15.4:

- *Beacons* habilitados (sem GTSs);
- *Beacons* desabilitados;
- *Beacons* desabilitados e sem CSMA-CA;
- *Beacons* desabilitados e sem *acknowledgements*; e
- *Beacons* desabilitados, sem CSMA-CA e sem *acknowledgements*.

A Figura 31 mostra quais atributos e como eles devem ser definidos no arquivo de configuração do *framework* para cada uma das opções listadas acima. Não é necessário modificar o código da aplicação, nem mesmo o código do *framework* para gerar os diferentes protocolos. Além dos microcomponentes das operações comuns de empacotar, desempacotar, transmitir e receber, os microcomponentes responsáveis pelo mecanismo de CSMA-CA e pela transmissão e recepção de *acknowledgements* também são reutilizados em todas as configurações nas quais estão presentes.

```
// Beacons habilitados
static const int SYNC         = BEACON;
static const int CONTENTION   = CSMA_CA;
static const int CONFIRMATION = ACK;

// Beacons desabilitados
static const int SYNC         = NO_SYNC;
static const int CONTENTION   = CSMA_CA;
static const int CONFIRMATION = ACK;

// Beacons desabilitados e sem CSMA-CA
static const int SYNC         = NO_SYNC;
static const int CONTENTION   = NO_CONTENTION;
static const int CONFIRMATION = ACK;

// Beacons desabilitados e sem acknowledgements
static const int SYNC         = NO_SYNC;
static const int CONTENTION   = CSMA_CA;
static const int CONFIRMATION = NO_CONFIRMATION;

// Beacons desabilitados, sem CSMA-CA e sem acknowledgements
static const int SYNC         = NO_SYNC;
static const int CONTENTION   = NO_CONTENTION;
static const int CONFIRMATION = NO_CONFIRMATION;
```

Figura 31: Diferentes configurações IEEE 802.15.4 no *Traits*.

Os experimentos com o EPOSMote I foram realizados de forma a simular uma típica aplicação de monitoramento (STEINER; MUCK; FROHLICH, 2010b). Uma estação base recebe os dados transmitidos periodicamente pelos outros nodos da rede responsáveis por coletar informações sobre o ambiente. A Figura 32 ilustra a topologia utilizada. A estação base, que funciona como nodo coordenador para a configuração com *beacons* habilitados, está localizada no centro da rede. Os outros nodos foram colocados de modo que todos os nós da rede ficassem dentro do alcance um dos outros. Dessa forma, todo nodo pode interferir potencialmente na comunicação dos outros. Para todos os experimentos dessa seção foram utilizados: o compilador GNU GCC para AVR, versão 4.0.2 para compilar o EPOS e a aplicação; o *clock* do ATMega1281 foi definido em 1 MHz; pacotes de dados de 64 bytes; e 3 dBm de potência de transmissão. Para a configuração com *beacons* habilitados foi utilizado um ciclo de trabalho no qual o período ativo ocupa 12% do tempo total do *superframe*, de forma que os nodos ficam inativos 88% do tempo.

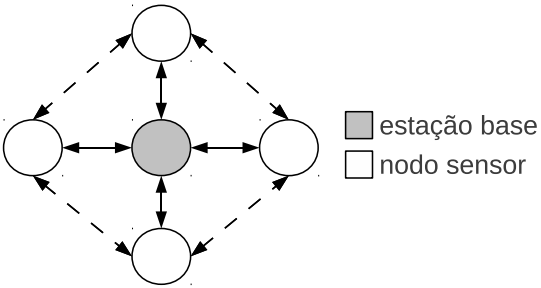


Figura 32: Topologia de rede utilizada na avaliação das diferentes configurações IEEE 802.15.4 do C-MAC no EPOSMote I.

4.1.1.1 Resultados

A fim de analisar o consumo de memória foi utilizada a ferramenta *avr-size*, da *GNU Binutils*, versão 2.19. Os resultados para cada configuração são mostrados na Tabela 2. Como esperado, quanto mais complexa a configuração, maior o consumo de memória. Dessa forma, a configuração com *beacons* desabilitados, sem CSMA-CA e sem *Acknowledgment*s (ACKs) teve o menor consumo, enquanto a configuração com *beacons* habilitados teve o maior. É importante ressaltar que mesmo a configuração que ocupa maior espaço ainda deixa 122kB de memória flash livre para a aplicação.

A Figura 33 mostra as variações na média da taxa de transferência por

Tabela 2: Consumo de memória das diferentes configurações IEEE 802.15.4 do C-MAC no EPOSMote I (bytes).

Configuração	Código	Dados
Beacons desabilitados, sem CSMA-CA e sem ACKs	3248	185
Beacons desabilitados e sem ACKs	3572	185
Beacons desabilitados e sem CSMA-CA	3768	202
Beacons desabilitados	4092	202
Beacons habilitados	5344	215

nodo enquanto o número de nodos da rede aumenta. Em geral, para as configurações com *beacons* desabilitados, o rendimento global da rede melhora à medida que as características dos protocolos são removidas e apresenta pouca variação com o acréscimo no número de nós. Isso se deve ao fato do baixo tráfego da rede e da não coincidência do período de transmissão dos nodos. A exceção é quando o mecanismo de contenção CSMA-CA é desabilitado e o uso de pacotes de confirmação habilitado. É possível perceber uma degradação de desempenho dessa configuração quando a rede possui cincos nodos. Isso ocorre pois não há um mecanismo para evitar colisões. Todavia, as colisões são detectadas e pacotes perdidos são retransmitidos podendo causar ainda mais colisões, consequentemente, reduzindo o desempenho do protocolo. O baixo ciclo de trabalho utilizado na configuração com *beacons* habilitados resultou na pior taxa de transferência. Essa configuração também apresentou deterioração de desempenho com o aumento no número de nodos. Isso ocorre, pois diferentemente das outras configurações, todos os nodos tentam transmitir ao mesmo tempo, dentro de um pequeno intervalo, aumentando a chance de colisões e necessidade de retransmissões.

A eficiência energética foi avaliada medindo-se a energia consumida por byte recebido na estação base. A Figura 34 mostra os resultados obtidos. Como esperado, a configuração com *beacons* habilitados obteve o melhor desempenho, uma vez que essa é a única configuração a tratar o problema de escuta ociosa. Para as configurações com *beacons* desabilitados, a energia consumida por byte diminui à medida que o número de nodos aumenta. Isso ocorre porque a maior fonte de consumo de energia é a escuta ociosa. Com o aumento do tráfego da rede, o tempo que a rede fica ociosa diminui, consequentemente, diminuindo a média de energia consumida por byte.

Para avaliar latência, foi medido o tempo de ida e volta, do inglês *Round-trip Time* (RTT), de um pacote entre dois, três e quatro nodos, como ilustrado na Figura 35 (STEINER; MUCK; FROHLICH, 2010a). Os resul-

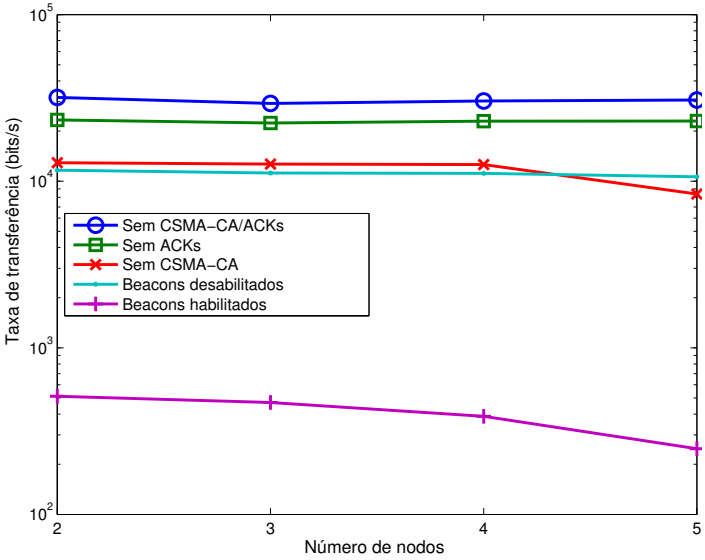


Figura 33: Média da taxa de transferência por nodo das diferentes configurações IEEE 802.15.4 do C-MAC no EPOSMote I (escala logarítmica).

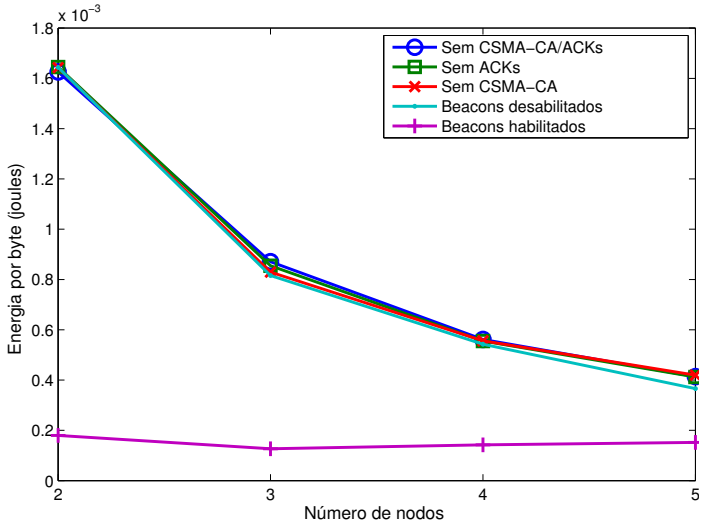


Figura 34: Energia consumida, por byte recebido na estação base, pelas diferentes configurações IEEE 802.15.4 do C-MAC no EPOSMote I.

tados, ilustrados na Figura 36, mostram que a latência aumenta à medida que mais características do protocolo são habilitadas. A configuração com *beacons* habilitados, possui um ciclo de trabalho de 12% resultando em um período de inatividade com cerca de 2 segundos, que é o fator dominante na latência. Entretanto, para essa configuração, o tempo gasto com escuta ociosa é reduzido, assim reduzindo o consumo de energia como mostrado na Figura 34.

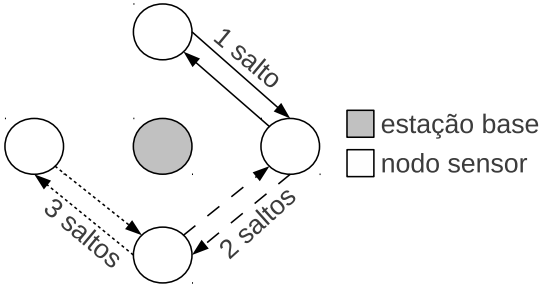


Figura 35: Configurações utilizadas para medir os tempos de ida e volta de um pacote.

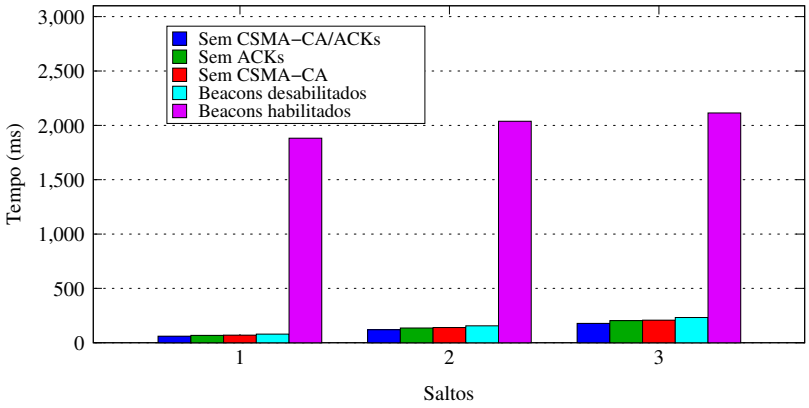


Figura 36: Tempos de ida e volta de um pacote utilizando as diferentes configurações IEEE 802.15.4 do C-MAC no EPOSMote I variando o número de saltos entre 1 e 3.

Analisando os resultados apresentados pode-se concluir que caso a rede possua poucos nodos, necessite de uma alta taxa de transferência e o consumo de energia não seja um problema, a melhor solução é a configuração

com *beacons* desabilitados, sem CSMA-CA e sem *acknowledgements*. De fato essa configuração do C-MAC já foi utilizada em outro trabalho, com essas mesmas características, para transmitir dados multimídia (RUFINO, 2012). Caso a rede possua muitos nodos, o uso de CSMA-CA passa a ser interessante. Se os dados que a aplicação transmite são cruciais e não podem ser perdidos então o uso de *acknowledgements* é essencial. Já se a rede deve funcionar por um longo período de tempo, de forma que o consumo de energia seja um fator crítico, então a configuração com *beacons* habilitados passa a ser a melhor opção.

A Tabela 3 apresenta uma comparação de consumo de memória e latência entre a ZigBeeNet, uma implementação do MAC IEEE 802.15.4 fornecida pela Meshnetics (MeshNetics, 2008) para o mesmo microcontrolador e rádio do EPOSMote I (ATMega1281/AT86RF230), e o C-MAC configurado para fornecer a mesma funcionalidade — *beacons* desabilitados, *CSMA-CA* e *acknowledgements* habilitados. O C-MAC apresenta menor consumo de memória e fornece um desempenho comparável à ZigBeeNet, que possui uma implementação otimizada para a plataforma e não configurável.

Tabela 3: Consumo de memória e RTT: C-MAC IEEE 802.15.4 vs ZigBeeNet IEEE 802.15.4. Ambas as implementações estão com *beacons* desabilitados.

Configuração	Código (bytes)	Dados (bytes)	RTT (ms)
C-MAC IEEE 802.15.4	4092	202	79
ZigBeeNet IEEE 802.15.4	26776	289	62

4.2 EPOSMOTE II

O EPOSMote II surgiu da necessidade de expandir e da dificuldade de programar o EPOSMote I. Essa plataforma foca em pesquisa e apresenta uma arquitetura modular, ilustrada na Figura 37, permitindo fácil programação e expansão. Seu hardware foi projetado como uma arquitetura de camadas composta por um módulo principal, um módulo de sensoriamento e um módulo de alimentação. O módulo principal é responsável pelo processamento, armazenamento de dados e comunicação. O modelo utilizado possui um processador ARM7 32-bit, com 128kB de memória flash, 96kB de RAM e um transceptor compatível com o padrão IEEE 802.15.4. Foi desenvolvido um módulo de sensoriamento básico, que possui um sensor de temperatura e um acelerômetro, LEDs, botões e uma micro USB (que também pode ser uti-

lizada como fonte de alimentação).

O EPOSMote II também possui um acelerador em hardware de *Advanced Encryption Standard* (AES). O AES é um algoritmo criptográfico de chave simétrica que possui um tamanho de bloco fixo em 128 bits, com tamanhos de chave de 128, 192 e 256 bits. Entretanto, o acelerador presente no EPOSMote II só suporta chaves de 128 bits. Como qualquer algoritmo criptográfico computacionalmente seguro, o AES é extremamente dispendioso em termos de tempo de execução, pois necessita executar muitas operações aritméticas e lógicas. O uso de aceleração em hardware de algoritmos criptográficos não só melhora o desempenho dos sistemas de segurança, como também deixa os recursos computacionais disponíveis para a aplicação (CHANG et al., 2010).

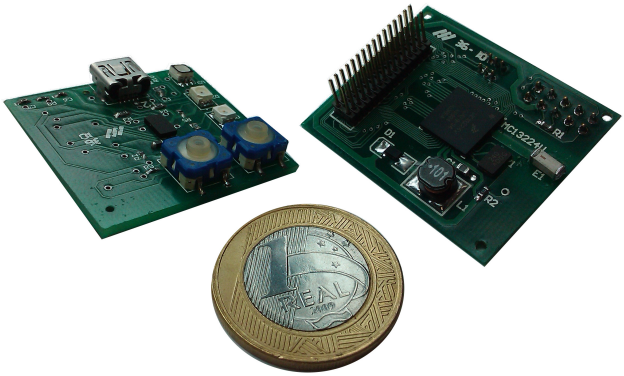


Figura 37: EPOSMote II lado a lado com uma moeda de R$1. No lado esquerdo o módulo de sensoriamento. No lado direito o módulo principal.

4.2.1 IEEE 802.15.4

A fim de permitir uma comparação de desempenho com a plataforma EPOSMote I e com outro trabalho relacionado, a configuração com *beacons* desabilitados, sem CSMA-CA e sem *acknowledgements* foi novamente avaliada, em termos de latência e consumo de memória, agora executando no EPOSMote II. Para os experimentos dessa seção utilizou-se: o compilador GNU GCC para ARM, versão 4.4.4; o *clock* do processador ARM foi definido em 24 MHz; e pacotes de dados de 64 bytes.

4.2.1.1 Resultados

A Tabela 4 apresenta uma comparação de consumo de memória e latência entre a libmc1322x (ALVIRA, 2010), uma biblioteca de software *open source* desenvolvida para a mesma arquitetura do EPOSMote II, e o C-MAC, ambos com a mesma funcionalidade. O consumo de memória foi obtido utilizando a ferramenta *arm-size*, parte do *GNU Binutils*, versão 2.20. O C-MAC apresenta maior consumo de memória, mas fornece um desempenho dez vezes melhor que a libmc1322x. Isso se deve ao fato que a aplicação de teste da libmc1322x possui um *loop* realizando *busy waiting* entre a transmissão e recepção de um pacote. Caso esse *loop* seja removido, a aplicação não funciona corretamente.

Tabela 4: Consumo de memória e RTT: C-MAC vs libmc1322x.

Configuração	Código (bytes)	Dados (bytes)	RTT (ms)
C-MAC	9192	5715	7.2
libmc1322x	6227	4996	72

Observando as Tabelas 2 e 4 e a Figura 36 percebe-se que a mesma configuração do C-MAC (*beacons* desabilitados, sem CSMA-CA e sem ACKs) apresenta resultados diferentes, para consumo de memória e latência, quando executa em diferentes plataformas. Apesar de utilizarem os mesmos micro-componentes, a configuração no EPOSMote I consome menos memória, mas apresenta maior latência. Essas divergências podem ser explicadas por dois fatores. Primeiro, enquanto o EPOSMote I possui uma arquitetura AVR 8-bit, o EPOSMote II possui um ARM7 32-bit, de forma que o tamanho de instruções e algumas variáveis é maior no EPOSMote II. Segundo, as duas plataformas utilizam rádios diferentes e, apesar de ambos serem compatíveis com o padrão IEEE 802.15.4, eles possuem implementações em hardware diferentes, exigindo diferentes mediadores em software. Isso influencia tanto no consumo de memória, quanto na latência. Essa diferença de desempenho entre os EPOSMote I e II serve para confirmar que não existe uma solução ótima e até mesmo a plataforma utilizada influencia na determinação de qual a melhor solução para cada caso.

4.2.2 B-MAC

A fim de obter mais comparações com outros protocolos o C-MAC foi configurado para fornecer as mesmas funcionalidades do B-MAC e os dados obtidos foram comparados com resultados de terceiros. A Figura 38 ilustra as topologias utilizadas para avaliar o C-MAC. Um nodo é definido como receptor e nodos transmissores são adicionados um por vez. Cada transmissor envia 1000 pacotes de dados e o receptor mede o número total de pacotes recebidos. Todo nodo está dentro do alcance dos outros e pode interferir na comunicação dos demais.

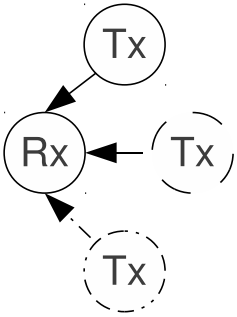


Figura 38: Topologias utilizadas para avaliar o C-MAC configurado como B-MAC.

A configuração foi avaliada em termos de taxa de transferência e perda de pacotes. Do mesmo modo que nos experimentos de terceiros, o período ativo do protocolo foi definido em 100%. Para os experimentos dessa seção utilizou-se: o compilador GNU GCC para ARM, versão 4.4.4; o *clock* do processador ARM foi definido em 24 MHz; e pacotes de dados de 29 bytes (mesmo tamanho utilizado nos experimentos de terceiros).

4.2.2.1 Resultados

A Figura 39 apresenta a taxa de transferência por nodo enquanto o número de transmissores aumenta. Os resultados adquiridos nesse experimento foram comparados com dados de outros trabalhos: a legenda "*B-MAC - Polastre*" representa os dados do artigo original do B-MAC (POLASTRE; HILL; CULLER, 2004); a legenda "*B-MAC - C-MAC original*" representa os dados do artigo original do C-MAC (WANNER; OLIVEIRA; FROHLICH,

2007); e a legenda "*B-MAC - TinyOS*" representa os dados do B-MAC executando utilizando o sistema operacional TinyOS (WANNER; OLIVEIRA; FROHLICH, 2007). A lacuna entre o desempenho do novo C-MAC e das outras versões ocorre pois os dados foram coletados em plataformas diferentes. Enquanto o novo C-MAC executa na plataforma EPOSMote II, ARM7 32-bit com um transceptor integrado, os outros dados foram coletados utilizando o nodo sensor Mica2, arquitetura AVR 8-bit com o transceptor CC1000. A Figura 39 apresenta os dados normalizados da taxa de transferência. Dessa forma, o comportamento do protocolo pode ser analisado sem influência da plataforma na qual está executando. Em geral, todas as versões apresentam o mesmo comportamento: quanto menos transmissores saturando o canal, melhor é a taxa de transferência.

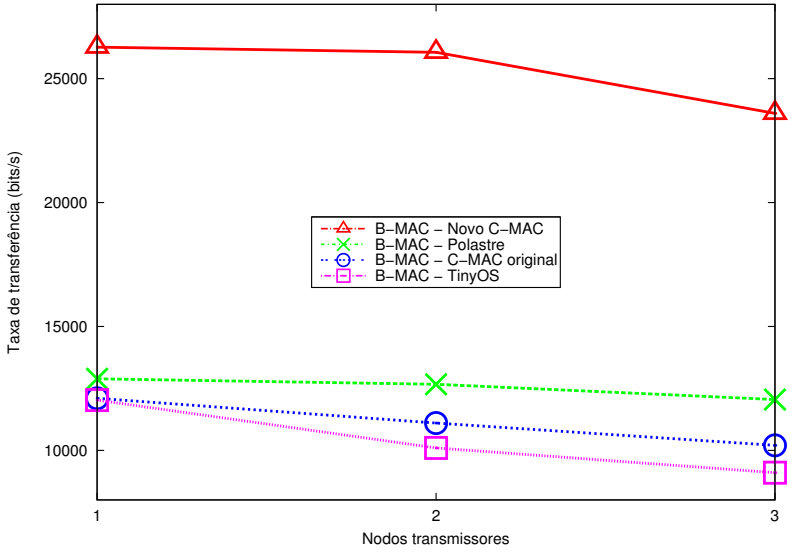


Figura 39: Média da taxa de transferência por nodo das diferentes versões do B-MAC.

A perda de pacotes pode ser observada na Figura 41. Quanto mais transmissores mais colisões e, consequentemente, maior é a perda de pacotes. A proposta original do B-MAC não apresenta esses resultados e por isso não está ilustrada no gráfico para comparações.

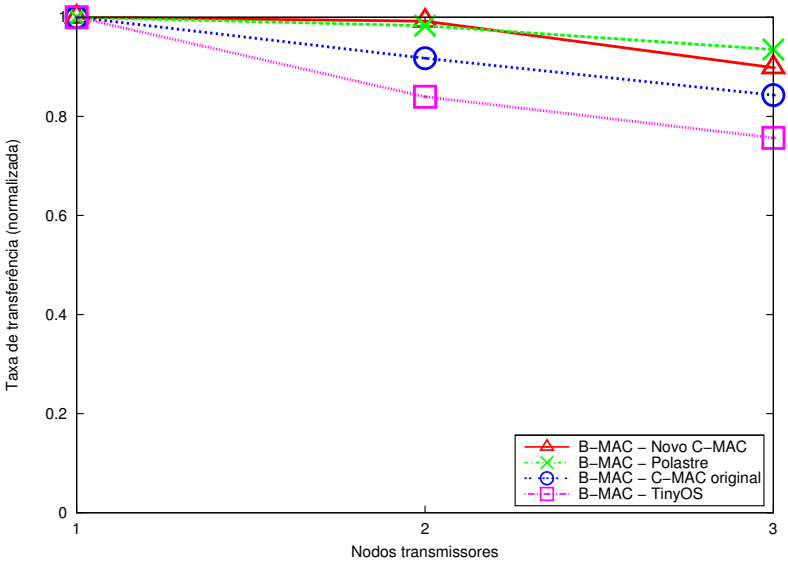


Figura 40: Taxa de transferência do B-MAC (dados normalizados).

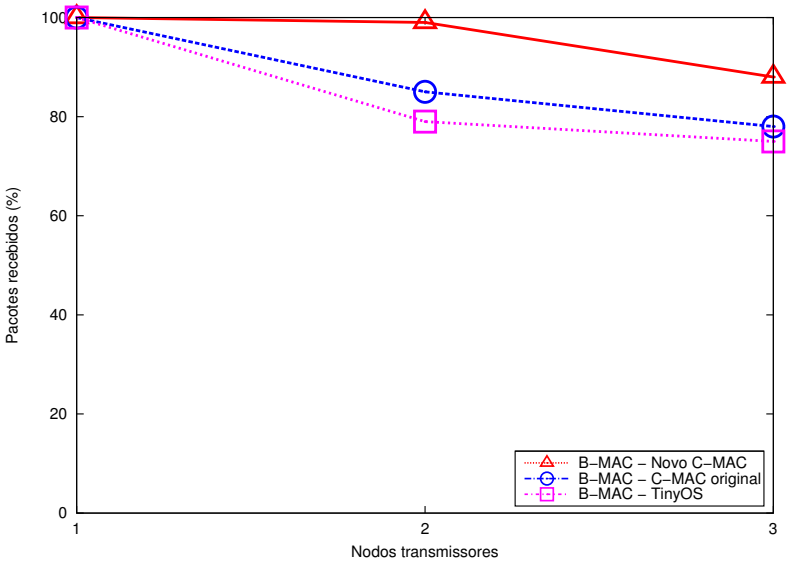


Figura 41: Taxa de pacotes recebidos das diferentes versões do B-MAC.

4.2.3 Variando as condições do canal

As condições do canal de comunicação de uma RSSF são instáveis por natureza e a sua confiabilidade varia com o passar do tempo. Nodos sensores podem ficar sem bateria, seu hardware pode falhar, eles podem se mover para fora do alcance de seus vizinhos originais ou até mesmo serem retirados da rede (DOHLER et al., 2009). Em uma rede onde os nodos são alimentados por energia solar, por exemplo, a topologia pode mudar de acordo com o clima e condições dos painéis solares (e.g. ângulo, poeira e sujeira cobrindo o painel). Levando isso em consideração, foi realizado um experimento para simular variações nas condições do canal (STEINER et al., 2013). O C-MAC foi configurado para funcionar como os protocolos B-MAC e RB-MAC e a probabilidade p de um pacote ser corretamente recebido por um nodo foi alterada entre 1, 0.8, 0.5 e 0.3. Para simular as condições de canal nesse experimento foi utilizado um algoritmo para gerar números pseudo aleatórios fornecido pelo sistema operacional. Cada vez que um nodo recebe um pacote ele gera um número aleatório, dentro de um intervalo, e de acordo com o resultado e com a probabilidade p sendo testada o pacote pode ser descartado.

A Figura 42 mostra quais atributos e como eles devem ser definidos no arquivo de configuração do *framework* para gerar os protocolos B-MAC e RB-MAC. A máquina de estados do C-MAC gerada para os dois protocolos é ilustrada na Figura 43. As duas configurações utilizam os mesmos estados, porém, usam diferentes microcomponentes para realizar a sincronização via preâmbulo e para determinar se uma transmissão foi bem sucedida. Repare que vários estados do C-MAC não são utilizados por esses protocolos e por isso foram completamente removidos. Além disso, a nova arquitetura possibilita o reuso dos microcomponentes de todos os estados em branco.

```
// B-MAC
static const int SYNC             = PREAMBLE;
static const int PREAMBLE_CONTENT = DUMMY;
static const int CONFIRMATION     = ACK;

// RB-MAC
static const int SYNC             = PREAMBLE;
static const int PREAMBLE_CONTENT = TIME_TO_DATA_AND_RANK;
static const int CONFIRMATION     = PASSIVE_CONTENTION_WINDOW;
```

Figura 42: Configurações do B-MAC e RB-MAC no *Traits*.

A Figura 44 ilustra o cenário de avaliação. Primeiramente um nodo é definido como transmissor e outro como receptor. As configurações são avaliadas para cada probabilidade p de recepção. Quando todas as probabilidades são avaliadas um receptor é adicionado e o processo é repetido até que a rede

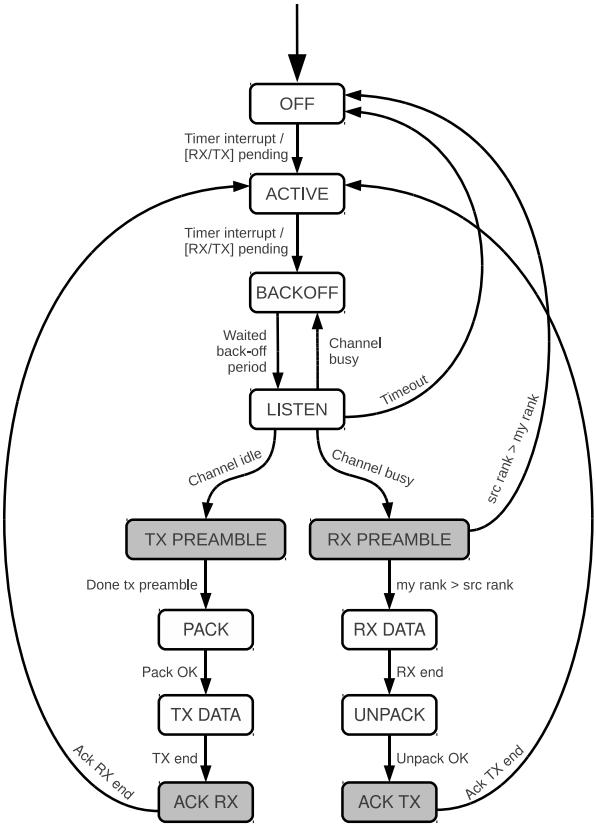


Figura 43: Máquina de estados do C-MAC para as configurações B-MAC e RB-MAC. Estados destacados indicam quais microcomponentes possuem implementações diferentes.

possua um total de quatro receptores ao mesmo tempo.

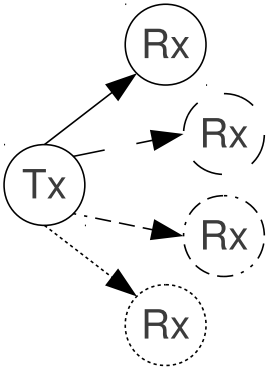


Figura 44: Topologias utilizadas para os experimentos variando as condições do canal.

As configurações foram avaliadas em termos de latência, consumo de memória e energia. Para todos os experimentos dessa seção utilizou-se: o compilador GNU GCC para ARM, versão 4.4.4; o *clock* do processador ARM foi definido em 24 MHz; duração do *CCA* de 1.43 ms; duração do preâmbulo de 143.8 ms; pacotes de dados de 94 bytes; e o rádio foi ajustado para transmitir a 4.5 dBm.

4.2.3.1 Resultados

Foi utilizada a ferramenta `arm-size`, parte do *GNU Binutils* versão 2.20, para obter o consumo de memória de ambas as configurações. Os resultados são apresentados na Tabela 5. Como esperado, o RB-MAC apresenta maior consumo de memória, uma vez que utiliza mecanismos mais complexos. Enquanto o B-MAC utiliza um preâmbulo sem informações úteis, o RB-MAC utiliza informações que permitem aos nodos voltarem a dormir e só acordar para receber os dados ou até mesmo desistir da recepção. Além disso, o B-MAC utiliza simples pacotes de confirmação, ao passo que o RB-MAC utiliza uma janela de contenção para verificar se um pacote transmitido está sendo repassado adiante.

A Figura 45 apresenta os tempos necessários para transmitir um pacote corretamente, incluindo retransmissões, variando as condições do canal para as duas configurações. É possível perceber que o desempenho do B-MAC piora quanto menor a probabilidade de recepção e não apresenta variações

Tabela 5: Consumo de memória do C-MAC configurado como B-MAC e RB-MAC (bytes).

Configuração	Código	Dados
B-MAC	10780	5717
RB-MAC	11236	5717

com o acréscimo de nodos. Isso ocorre pois, no protocolo B-MAC, o nodo emissor envia os dados a um destinatário específico. Dessa forma, quanto menor a probabilidade de recepção, mais retransmissões são necessárias, consequentemente, aumentando o tempo necessário para transmitir um pacote corretamente. Já no protocolo RB-MAC, o emissor não possui um destinatário específico e todos seus nodos vizinhos são potenciais receptores. Quanto mais vizinhos, maiores são as chances de que a mensagem seja recebida corretamente e, portanto, o número de retransmissões é reduzido. Quando a probabilidade de recepção é igual a 1, não há necessidade de retransmissões e as duas configurações apresentam o mesmo comportamento.

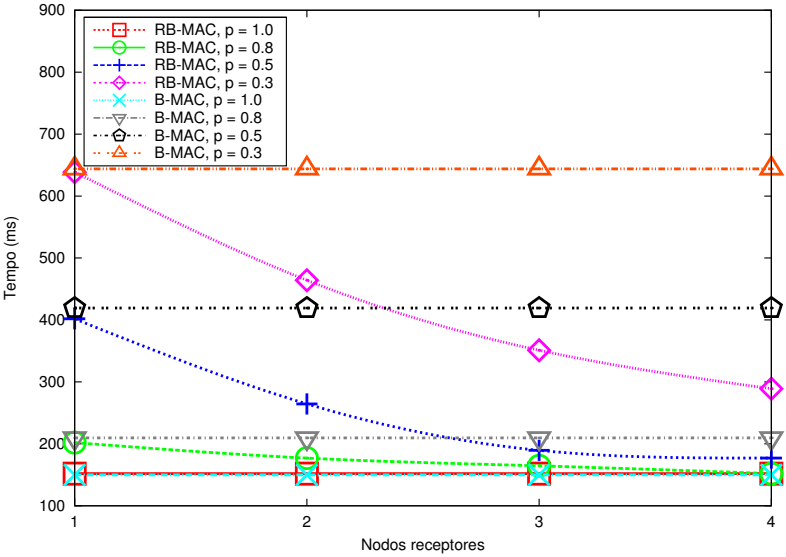


Figura 45: Latência do C-MAC configurado para funcionar como B-MAC e RB-MAC variando as condições do canal no EPOSMote II.

A Figura 46 apresenta os resultados, obtidos por simulação, da latên-

cia do RB-MAC comparado ao 1-hopMAC (WALTEYNE et al., 2006) — outro MAC desenvolvido para RSSF que assim como o B-MAC especifica um nodo destinatário ao transmitir um pacote. Pode-se perceber pelos resultados que os protocolos se comportam da mesma forma tanto na simulação quanto executando nos EPOSMote II.

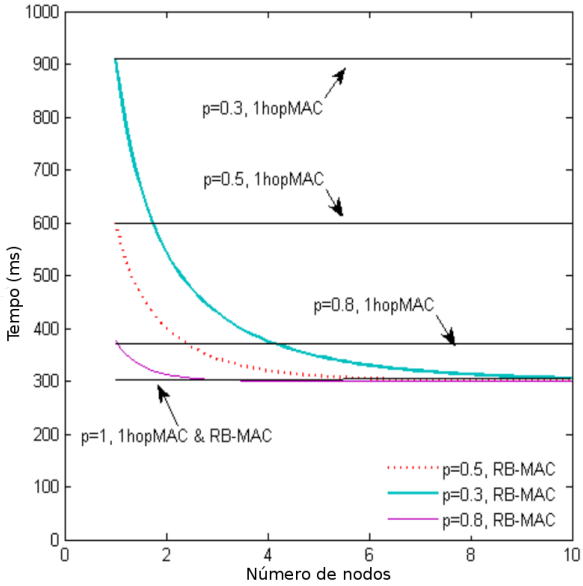


Figura 46: Latência do RB-MAC e do 1-hopMAC (simulação) (AKHAVAN; WATTEYNE; AGHVAMI, 2011).

O consumo de energia das configurações para transmitir um pacote, incluindo retransmissões, variando as condições do canal é ilustrado na Figura 47. Como as transmissões no B-MAC possuem um destinatário específico, cada receptor adicionado à rede passa a ser sinônimo de escuta desnecessária, ou seja, desperdício de energia. Além disso, quanto menor a probabilidade de recepção, mais retransmissões são necessárias e maior é o consumo de energia. Para o protocolo RB-MAC quanto mais receptores, menores são as chances de haver retransmissão. Contudo, isso nem sempre significa menos consumo de energia. Existe uma relação entre a probabilidade de recepção e a energia gasta com escuta desnecessária na qual, a partir de certo ponto, adicionar receptores não reduz o consumo de energia. Isso ocorre porque enquanto um dos receptores recebe os dados, os outros estão gastando energia escutando o meio e essa energia gasta por muitos nodos acaba sendo maior

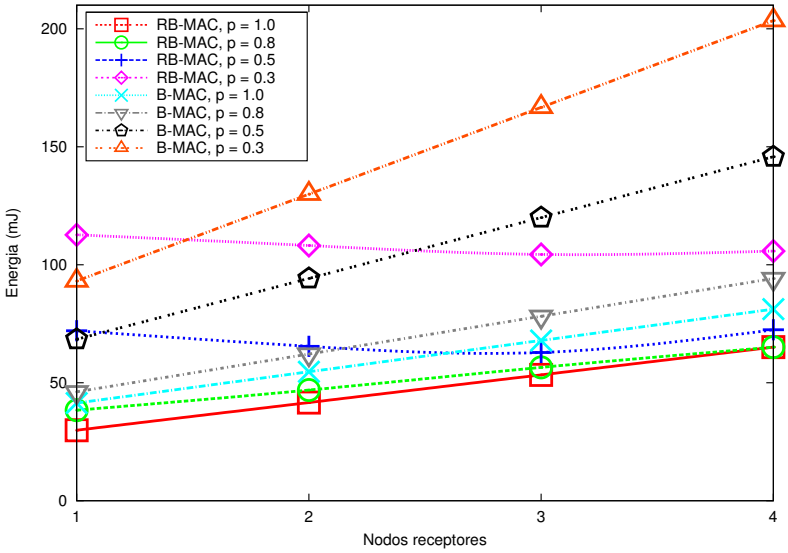


Figura 47: Consumo de energia, para transmitir um pacote, das configurações B-MAC e RB-MAC variando as condições do canal no EPOSMote II.

que a necessária para realizar mais retransmissões com menos nodos. É possível visualizar esse fenômeno na Figura 47. Na configuração RB-MAC com p igual a 0.5 o consumo de energia reduz de um para dois receptores, de dois para três, mas começa a crescer quando há quatro receptores.

Analisando os resultados apresentados pode-se concluir que o RB-MAC é melhor que o B-MAC quando as condições do canal não são boas. Todavia, o RB-MAC só pode ser utilizado por aplicações que apresentam o padrão de interação de muitos para um. Quando os nodos da rede se comunicam de muitos para muitos, ou de um para muitos o RB-MAC não pode ser utilizado. Já o B-MAC funciona para todos os padrões de interação.

4.2.4 Infraestrutura segura para IoT

Os mecanismos de segurança que foram incorporados na nova arquitetura do *C-MAC* foram avaliados dentro de um cenário de Internet das Coisas, do inglês *Internet of Things* (IoT) (FROHLICH; STEINER; RUFINO, 2011). Foi implementada uma infraestrutura utilizando o EPOSMote II com o objetivo de fornecer ao usuário final uma pilha de comunicação segura e com-

patível com TCP/IP. A segurança foi alcançada por uma combinação de mecanismos. A configurabilidade do C-MAC permitiu personalizar a camada MAC de forma a não desperdiçar recursos. Como o TCP garante a entrega confiável e ordenada de pacotes, o C-MAC foi configurado de maneira simplista, sem mecanismos de sincronização e confirmação, utilizando apenas CSMA e períodos de *backoff* de forma a evitar colisões. O *C-MAC* também realiza a cifragem e autenticação dos dados utilizando o acelerador em hardware de AES disponível na plataforma. Por fim, para proteger a rede de ataques de repetição, o mecanismo de autenticação incorpora informações temporais.

A Figura 48 apresenta a arquitetura da rede. Para gerenciar as chaves dos nodos sensores, foi utilizado um esquema de distribuição centralizado. Cada nodo compartilha uma chave única, pré-definida, com a estação base e mais ninguém. Durante o funcionamento da rede, um nodo pode consultar a estação base para verificar se pertence a algum grupo e adquirir as chaves correspondentes. Uma vez que possua uma chave de grupo um nodo pode se comunicar com outros nodos, pertencentes ao mesmo grupo, sem o intermédio da estação base. A qualquer momento a estação base pode inserir ou remover nodos e até mesmo trocar a chave de um grupo.

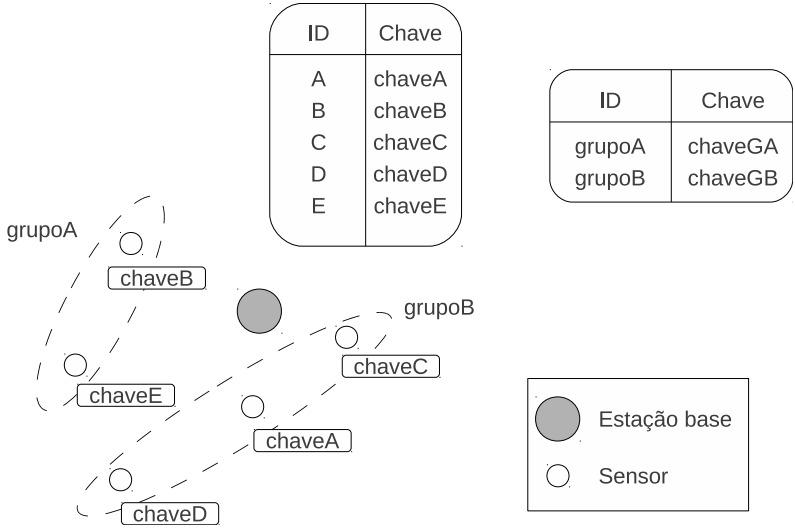


Figura 48: Arquitetura da rede.

Como contramedida aos ataques de repetição foi introduzido um campo nos pacotes transmitidos contendo informação de tempo, conforme ilustrado na Figura 49. Cada pacote inclui os cabeçalhos dos protocolos de comuni-

cação, os dados da aplicação, o tempo atual e o código de autenticação de mensagem. A informação de tempo pode ser obtida por um sistema de posicionamento global, do inglês *Global Positioning System* (GPS), ou utilizando um dos muitos protocolos de sincronização de relógio para RSSF presentes na literatura (FONTANELLI; PETRI, 2009; FONTANELLI; MACII, 2010; SWAIN; HANSDAH, 2010). Utilizando o tempo, mensagens que forem repetidas por um atacante podem ser descartadas.

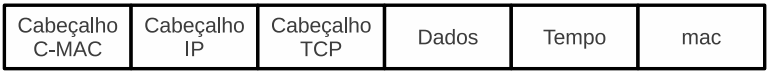


Figura 49: Formato do pacote.

A Figura 50 ilustra o cenário de avaliação. Foram utilizados dois EPOSMote II. Um nodo atua como uma estação base, conectando a IoT à Internet comum, enquanto o outro como um nodo sensor. A estação base envia um pacote cifrado requisitando temperatura a cada 10 segundos. O sensor decifra a requisição, coleta os dados necessários e responde com um pacote assinado e cifrado.

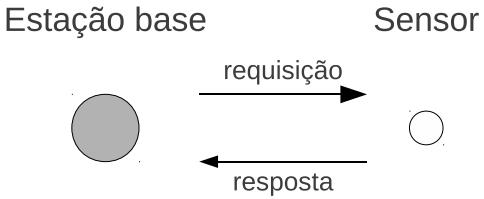


Figura 50: Cenário de avaliação.

A implementação da infraestrutura foi avaliada em três aspectos: consumo de memória, tempos de cifragem/decifragem e consumo de energia. Para todos os experimentos dessa seção utilizou-se o compilador GNU GCC para ARM, versão 4.4.4; o *clock* do processador ARM do EPOSMote II foi definido em 24 MHz; as mensagens foram ajustadas para carregar um *payload* de 16 bytes quando a cifragem está ativa, 7 (requisição) e 6 (resposta) bytes caso contrário; e o rádio foi ajustado para transmitir a 4.5 dBm.

4.2.4.1 Resultados

Para obter o consumo de memória foi utilizada a ferramenta *arm-size*, parte do *GNU Binutils*, versão 2.20. Os resultados são apresentados na Tabela 6. A linha *Mediador AES* representa o código necessário para interagir com o acelerador AES em hardware presente no EPOSMote II, utilizado para cifrar, decifrar e autenticar os dados. A linha *Aplicação com AES* representa o tamanho do código da aplicação utilizando a infraestrutura segura apresentada, enquanto a linha *Aplicação sem AES* representa o tamanho de quando não se utiliza os mecanismos de segurança presentes no C-MAC. É possível perceber que existe uma diferença entre o valor da linha *Aplicação com AES* e a soma das linhas *Aplicação sem AES* com *Mediador AES*. Isso se deve ao fato de que nem todos os métodos do *mediador AES* são utilizados na *Aplicação com AES*. Os métodos do mediador que não são invocados pelo *C-MAC* são eliminados durante a compilação.

Tabela 6: Consumo de memória da pilha de comunicação segura implementada no EPOSMote II (bytes).

	Código	Dados
Mediador AES	1336	10
Aplicação com AES	47184	5485
Aplicação sem AES	45916	5485

Para medir o tempo de cifragem, decifragem e autenticação foi utilizado um osciloscópio. Um pino de *General Purpose Input/Output* (GPIO) no EPOSMote II foi conectado ao osciloscópio e definido para alto antes do procedimento ser executado e para baixo logo após. O experimento rodou por um minuto e as médias calculadas são apresentadas na Tabela 7. Os valores obtidos, além de confirmarem a eficiência da implementação em termos de tempo de execução, também possuem impactos positivos no tempo de vida da bateria de um nodo.

Tabela 7: Tempos para cifrar/decifrar 16 bytes e verificar o mac no EPOSMote II (μs).

	Cifragem	Decifragem	Verificação do mac
Tempo	17	15	12

A Figura 51 mostra o consumo de energia de ambas as aplicações, com e sem AES, sobre o tempo. O pequeno aumento no consumo de energia para a *Aplicação com AES* se deve ao uso do acelerador em hardware presente no EPOSMote II. Após dez minutos executando, a diferença é mínima (53.2 J com AES e 52.6 sem). Após uma hora, as aplicações consumiram 319.5 J e 315.5 J, respectivamente, uma diferença de 1.25%.

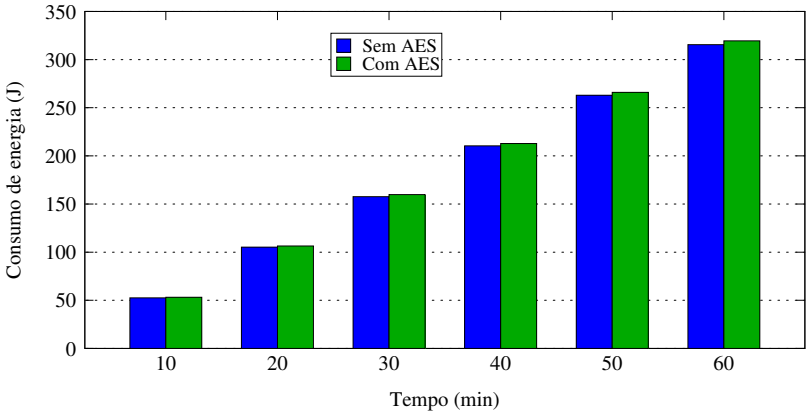


Figura 51: Consumo de energia, no EPOSMote II, da aplicação com e sem os mecanismos de segurança.

A infraestrutura proposta foi comparada com três outras implementações. Primeiro com outra abordagem que também utiliza aceleração em hardware do algoritmo criptográfico. Huai propõe reduzir ciclos de trabalho e consumo de energia da execução do algoritmo AES executando a cifragem e a autenticação em paralelo (HUAI et al., 2009). Sua implementação proporciona melhores resultados quando comparado ao acelerador em hardware de AES sequencial presente no EPOSMote II. O acelerador paralelo, proposto por Huai, leva 71,6 ns para cifrar e autenticar um pacote que possui 17 bits de dados. A segunda comparação foi com uma abordagem sem acelerador em hardware. Lee avalia o desempenho de executar o algoritmo AES, com uma chave de 128 bits, em um microcontrolador AVR de 8-bits (LEE; LEE; SHIN, 2010). Sem um acelerador em hardware, foi possível perceber que o tempo e o número de ciclos de CPU crescem proporcionalmente ao tamanho dos dados a serem cifrados. Para cifrar e decifrar 16 bytes foram necessários 449 e 456 ms, respectivamente. Por fim, a infraestrutura proposta foi comparada com uma implementação estabelecida em RSSF. Contudo, o TinySec não utiliza o algoritmo AES, mas sim o Skipjack com uma chave de 80 bits,

consequentemente, fornecendo menor segurança. Além disso, o TinySec só está disponível para o nodo sensor Mica2 (HILL; CULLER, 2002). Essa plataforma não possui acelerador em hardware, porém o código utilizado é otimizado através do uso de assembly inline para acelerar operações mais comuns. O TinySec leva 1.6 ms para cifrar e 1.0036 ms para decifrar 16 bytes.

5 CONSIDERAÇÕES FINAIS

Esta dissertação apresentou a proposta, implementação e avaliação de uma nova arquitetura para o *framework Configurable Medium Access Control* (C-MAC). O C-MAC funciona como um *framework* de estratégias de controle de acesso ao meio, as quais podem ser combinadas para produzir protocolos específicos de aplicação. Através desse paradigma, programadores de aplicações podem criar novos serviços de comunicação sob demanda e experimentar diferentes parâmetros de comunicação, coletando métricas para identificar e ajustar o protocolo às suas necessidades. Contudo, o C-MAC original fornecia um baixo reuso de seus componentes em software, dificultando e reduzindo sua configurabilidade. Por exemplo, a transmissão de dados era tratada como um único grande componente que, consequentemente, devia ser reimplementado para cada novo protocolo, pois mesmo que as operações de controle do rádio fossem iguais, características como cabeçalhos e mecanismos de tratamento de erro e segurança utilizados no empacotamento dos pacotes variam de um protocolo para outro.

O reprojeto apresentado nesta dissertação partiu de um estudo sobre os princípios de comunicação em RSSF e uma caracterização dos protocolos MAC para essas redes, com o intuito de identificar componentes comuns e as especificidades desses protocolos. Foi adotada uma classificação que divide os protocolos MAC em quatro categorias, *channel polling*; *scheduled contention*; *time division multiple access*; e híbrido, e para cada categoria foi elaborada uma máquina de estados. O novo C-MAC é resultado da união dessas máquinas de estados em uma única máquina generalizada. Cada estado representa um microcomponente, que pode ter diferentes implementações, e estados que não fazem sentido para um determinado protocolo podem ser completamente removidos. O produto final é uma arquitetura que possibilita grande reuso de seus microcomponentes, facilidade de configuração e maior configurabilidade que o C-MAC original e, portanto, suporta uma maior gama de aplicações. O uso de técnicas de metaprogramação estática em C++, na implementação da arquitetura, permitem ao compilador realizar otimizações no código, assegurando assim que a configurabilidade não comprometa o desempenho ou consumo de memória do protocolo gerado. Uma limitação, mantida da implementação original, é que o C-MAC não suporta configuração em tempo de execução. O *overhead* de manter todas as possibilidades de configuração em memória e a necessidade de um segundo protocolo para troca de configurações entre nodos torna o uso em tempo de execução do C-MAC uma opção muito onerosa.

O C-MAC foi implementado como uma abstração do sistema opera-

cional EPOS, o que possibilita seu uso em diferentes arquiteturas de hardware. Dessa forma, foram utilizadas duas plataformas diferentes para a execução dos experimentos: EPOSMote I, arquitetura AVR 8-bit, e EPOSMote II, ARM7 32-bit. Diferentes cenários de aplicações foram imaginados para avaliar algumas das diversas possibilidades de configurações e a capacidade de reuso dos microcomponentes da nova arquitetura do *framework* para gerar diferentes protocolos. Para cada cenário, as diferentes configurações do C-MAC foram comparadas entre si e com trabalhos relacionados, levando em consideração alguns parâmetros como consumo de memória, consumo de energia, taxa de transferência, latência e tempo de processamento.

O primeiro experimento avaliou o desempenho de diferentes configurações do MAC IEEE 802.15.4 executando no EPOSMote I. O *framework* possibilitou o reuso de todos microcomponentes responsáveis pelas operações comuns de empacotar, desempacotar, transmitir e receber, além dos microcomponentes responsáveis pelo mecanismo de CSMA-CA e pela transmissão e recepção de *acknowledgements* em todas as configurações nas quais estes estão presentes. Quanto mais complexa a configuração, maior seu consumo de memória. Analisando os resultados percebe-se que caso a rede possua poucos nodos, necessite de uma alta taxa de transferência e o consumo de energia não seja um problema, a melhor solução é a configuração com *beacons* desabilitados, sem CSMA-CA e sem *acknowledgements*. Caso a rede possua muitos nodos, o uso de CSMA-CA passa a ser interessante. Se os dados que a aplicação transmite são cruciais e não podem ser perdidos então o uso de *acknowledgements* é essencial. Já se a rede deve funcionar por um longo período de tempo, de forma que o consumo de energia seja um fator crítico, então a configuração com *beacons* habilitados passa a ser a melhor opção. Quando comparado à pilha ZigBeeNet, outra solução 802.15.4, o C-MAC configurado para fornecer a mesma funcionalidade apresentou menor consumo de memória, 4294 contra 27065 bytes, e desempenho comparável, RTT de 79 contra 62 ms.

O segundo experimento permitiu uma comparação de desempenho de uma mesma configuração do C-MAC executando nas plataformas EPOSMote I e II. Foi possível perceber que, apesar de utilizar os mesmos microcomponentes, a configuração com *beacons* desabilitados, sem CSMA-CA e sem ACKs consome mais memória, mas apresenta menor latência no EPOSMote II. Como as plataformas possuem arquiteturas diferentes, 8-bit e 32-bit, suas instruções e tamanhos de variáveis são diferentes. Além disso, as duas plataformas utilizam rádios diferentes, o que exige diferentes mediadores em software, afetando tanto o consumo de memória quanto o desempenho. Essa diferença de desempenho entre os EPOSMote I e II mostra que não existe uma solução ótima e até mesmo a plataforma utilizada influencia na determi-

nação de qual a melhor solução para cada caso. A mesma configuração foi avaliada utilizando a biblioteca libmc1322x, executando no EPOSMote II, e os resultados mostraram que o C-MAC consome mais memória, 14907 contra 11223 bytes, mas possui um desempenho dez vezes melhor, RTT de 7.2 contra 72 ms.

O terceiro experimento realizou uma comparação entre o desempenho do C-MAC configurado como B-MAC com resultados retirados da literatura. Assim como no segundo experimento, o fato dos experimentos serem realizados em diferentes plataformas introduz uma diferença de desempenho. A versão do novo C-MAC, executando no EPOSMote II, apresentou melhor taxa de transferência e menor perda de pacotes que a versão original do B-MAC e que a versão original do C-MAC, ambas avaliadas executando em nodos sensores Mica2. Contudo, normalizando os dados foi possível analisar o comportamento do protocolo sem influência da plataforma utilizada. Em geral todas as versões apresentaram o mesmo comportamento: quanto mais transmissores menor a taxa de transferência por nodo e maior a perda de pacotes.

O quarto experimento avaliou o comportamento do C-MAC configurado para fornecer as mesmas funcionalidades dos protocolos B-MAC e RB-MAC variando as condições do canal da rede. Ambas as configurações utilizam os mesmos estados, porém, usam diferentes microcomponentes para realizar a sincronização via preâmbulo e para determinar se uma transmissão foi bem sucedida, de forma que a configuração B-MAC consome menos memória. Todos os outros estados tem seus microcomponentes reutilizados. Como no protocolo B-MAC o nodo emissor envia dados a um destinatário específico, seu desempenho é afetado apenas pelas condições do canal e o acréscimo de receptores na rede só aumenta o consumo de energia. Já no protocolo RB-MAC, todos os nodos vizinhos do emissor são potenciais receptores. Dessa forma, quanto mais receptores, menor é o número de retransmissões, reduzindo assim a energia gasta. Contudo, a partir de certo ponto, adicionar receptores não reduz o consumo de energia. Isso ocorre pois a energia gasta com escuta desnecessária de muitos receptores é maior do que a necessária para realizar mais retransmissões com menos nodos. Analisando os resultados apresentados pode-se concluir que o RB-MAC é melhor que o B-MAC quando as condições do canal não são boas. Todavia, o RB-MAC só pode ser utilizado por aplicações que apresentam o padrão de interação de muitos para um. Já o B-MAC funciona para todos os padrões de interação.

O quinto experimento avaliou os mecanismos de segurança que foram incorporados na nova arquitetura do C-MAC em um cenário de IoT. Uma mesma aplicação foi executada com e sem os mecanismos de segurança e constatou-se que o uso desses mecanismos não introduz grande *overhead*. Em termos de consumo de memória, a aplicação com os mecanismos ocupa

52669 bytes contra 51401 da aplicação não segura. Os tempos para cifrar, decifrar e autenticar uma mensagem são de 17, 15 e 12 μs, respectivamente. Após uma hora de execução, as aplicações consumiram 319.5 J e 315.5 J, uma diferença de 1.25% a mais para a aplicação segura. Quando comparada ao TinySec, a implementação proposta fornece mais segurança, pois utiliza o algoritmo AES com chave de 128 bits contra o algoritmo Skipjack com chave de 80 bits, e melhor desempenho, o TinySec leva 1.6 e 1.0036 ms para cifrar e decifrar a mesma quantidade de dados.

Os resultados obtidos pelas diferentes configurações, nos diferentes cenários de avaliação, corroboram o novo design do C-MAC que apresenta alto reuso de seus microcomponentes e desempenho comparável a outras implementações não configuráveis. Além disso, os experimentos também serviram para mostrar que não existe um protocolo ótimo para RSSF e são fatores como nodos utilizados, ambiente de implantação e características da aplicação que determinam qual a melhor opção para cada caso. Contudo, aplicações utilizando o C-MAC agora podem configurar facilmente um protocolo MAC para corresponder às suas necessidades.

5.1 TRABALHOS FUTUROS

As próximas etapas de desenvolvimento do C-MAC deverão envolver abordagens multicamadas. Um primeiro passo seria utilizar a estrutura *Neighborhood*, apresentada neste trabalho, e integra-la com protocolos de roteamento já existentes e implementados para o EPOS, como o *Ant-based Dynamic Hop Optimization Protocol* (ADHOP) (OKAZAKI; FROHLICH, 2011; OKAZAKI, 2012) e o *Heuristic Environmental Consideration Over Positioning System* (HECOPS) (REGHELIN; FROHLICH, 2006; REGHELIN, 2007). Outra opção seria fazer do próprio C-MAC uma abordagem multicamada, rompendo totalmente com o modelo OSI e possivelmente alterando a máquina de estados atual.

Através do mecanismo de reconfiguração dinâmica de software do EPOS, o *EPOS Live Update System* (ELUS) (GRACIOLI, 2009), é possível tornar o C-MAC um componente reconfigurável do SO. Dessa forma, não é necessário manter todas as possibilidades de configuração em memória e utilizando um protocolo de disseminação (STEINER et al., 2012) é possível reconfigurar o C-MAC em tempo de execução.

O C-MAC pode ainda ser utilizado para descobrir qual configuração e quais parâmetros produzem os resultados mais adequados para uma infinidade de aplicações. Além disso, é possível misturar características de diferentes protocolos, criando assim novos protocolos.

Por fim, pode-se criar um conjuntos de regras que especifiquem dependências e restrições entre os microcomponentes do *framework*, de modo que configurações inválidas possam ser detectadas e rejeitadas antes do processo de compilação.

REFERÊNCIAS BIBLIOGRÁFICAS

AKHAVAN, M.; WATTEYNE, T.; AGHVAMI, A. Enhancing the performance of rpl using a receiver-based mac protocol in lossy wsns. In: *Telecommunications (ICT), 2011 18th International Conference on*. [S.l.: s.n.], 2011. p. 191 –194.

ALVIRA, M. *libmc1322x - library, build system, test code, and utilities for the MC13224V*. 2010. Disponível em: http://mc1322x.devl.org/libmc1322x.html Último acesso em: 13 Novembro 2012.

ARORA, A. et al. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 46, p. 605–634, December 2004. ISSN 1389-1286. <http://dl.acm.org/citation.cfm?id=1045638.1045641>.

BACHIR, A. et al. Mac essentials for wireless sensor networks. *Communications Surveys Tutorials, IEEE*, v. 12, n. 2, p. 222 –248, second 2010. ISSN 1553-877X.

BAI, Y. et al. An energy optimization protocol based on cross-layer for wireless sensor networks. *Journal of Communications*, Academy Publisher, v. 3, n. 6, p. 27–34, nov. 2008. ISSN 1796-2021.

BATALIN, M.; HATTIG, M.; SUKHATME, G. S. Mobile robot navigation using a sensor network. In: *In IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 2003. p. 636–642.

BUETTNER, M. et al. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In: *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2006. p. 307–320. ISBN 1-59593-343-3.

BUONADONNA, P. et al. Task: Sensor network in a box. In: *In Proceedings of European Workshop on Sensor Networks*. [S.l.: s.n.], 2005.

CHANG, J.-T. et al. Hardware-assisted security mechanism: The acceleration of cryptographic operations with low hardware cost. In: *Performance Computing and Communications Conference (IPCCC), 2010 IEEE 29th International*. [S.l.: s.n.], 2010. p. 327 –328. ISSN 1097-2641.

DAM, T. van; LANGENDOEN, K. An adaptive energy-efficient mac protocol for wireless sensor networks. In: *SenSys '03: Proceedings of the*

1st international conference on Embedded networked sensor systems. New York, NY, USA: ACM, 2003. p. 171–180. ISBN 1-58113-707-9.

DARGIE, W.; POELLABAUER, C. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. [S.l.]: John Wiley & Sons, 2010. (Wireless Communications and Mobile Computing). ISBN 9780470997659.

DEMIRBAS, M. Wireless sensor networks for monitoring of large public buildings. *Computer Networks*, v. 46, p. 605–634, 2005.

DOHLER, M. et al. *Routing Requirements for Urban Low-Power and Lossy Networks*. IETF, maio 2009. RFC 5548 (Informational). (Request for Comments, 5548). <http://www.ietf.org/rfc/rfc5548.txt>.

FONTANELLI, D.; MACII, D. Towards master-less wsn clock synchronization with a light communication protocol. In: *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*. [S.l.: s.n.], 2010. p. 105 –110. ISSN 1091-5281.

FONTANELLI, D.; PETRI, D. An algorithm for wsn clock synchronization: Uncertainty and convergence rate trade off. In: *Advanced Methods for Uncertainty Estimation in Measurement, 2009. AMUEM 2009. IEEE International Workshop on*. [S.l.: s.n.], 2009. p. 74 –79.

Freescale. *MC1322x Advanced ZigBeeTM- Compliant SoC Platform for the 2.4 GHz IEEE® 802.15.4 Standard Reference Manual*. [S.l.], 2010.

FROHLICH, A.; STEINER, R.; RUFINO, L. A trustful infrastructure for the internet of things based on eposmote. In: *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*. [S.l.: s.n.], 2011. p. 63 –68.

FROHLICH, A. A. *Application-Oriented Operating Systems*. Tese — Technical University of Berlin, Berlin, 2001.

FROHLICH, A. A.; SCHRODER-PREIKSCHAT, W. EPOS: an Object-Oriented Operating System. In: *2nd ECOOP Workshop on Object-Orientation and Operating Systems*. Lisbon, Portugal: [s.n.], 1999. (Chemnitzer Informatik-Berichte, CSR-99-04), p. 38–43. ISBN ISBN:3-540-66954-X.

FROHLICH, A. A.; SCHRODER-PREIKSCHAT, W. EPOS: Paving the Path for Parallel Applications. In: *Dagstuhl Seminar 238 - High Level Parallel Programming: Applicability, Analysis and Performance*. Dagstuhl Castle, Germany: [s.n.], 1999. p. 18.

FROHLICH, A. A.; SCHRODER-PREIKSCHAT, W. High Performance Application-Oriented Operating Systems – the EPOS Aproach. In: *11th Symposium on Computer Architecture and High Performance Computing*. Natal, Brazil: [s.n.], 1999. p. 3–9.

GAMMA, E. et al. *Design patterns: elements of reusable object-oriented software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-63361-2.

GRACIOLI, G. *ELUS: Projeto e Implementação de um Mecanismo de Reconfiguração Dinâmica de Software para Sistemas Profundamente Embarcados*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2009.

HILL, J.; CULLER, D. Mica: A wireless platform for deeply embedded networks. *IEEE micro*, v. 22, n. 6, p. 12–24, 2002.

HUAI, L. et al. An energy-efficient aes-ccm implementation for ieee802.15.4 wireless sensor networks. *Networks Security, Wireless Communications and Trusted Computing, International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 2, p. 394–397, 2009.

IEEE Computer Society. *IEEE Standard 802 Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. [S.l.], 2006.

INTANAGONWIWAT, C.; GOVINDAN, R.; ESTRIN, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: *MOBICOM*. [S.l.]: ACM, 2000. p. 56–67.

JAIN, V.; BISWAS, R.; AGRAWAL, D. P. Energy-efficient and reliable medium access in sensor networks. In: *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*. [S.l.: s.n.], 2007. p. 1 –8.

JINWALA, D. et al. Replay protection at the link layer security in wireless sensor networks. In: *Computer Science and Information Engineering, 2009 WRI World Congress on*. [S.l.: s.n.], 2009. v. 1, p. 160 –165.

JUANG, P. et al. *Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet*. 2002.

KARLOF, C.; SASTRY, N.; WAGNER, D. Tinysec: a link layer security architecture for wireless sensor networks. In: *Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York,

NY, USA: ACM, 2004. (SenSys '04), p. 162–175. ISBN 1-58113-879-2. <http://doi.acm.org/10.1145/1031495.1031515>.

KLUES, K. et al. A component-based architecture for power-efficient media access control in wireless sensor networks. In: *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007. p. 59–72. ISBN 978-1-59593-763-6.

LANGENDOEN, K.; HALKES, G. Energy-efficient medium access control. In: ZURAWSKI, R. (Ed.). *Embedded Systems Handbook*. CRC press, 2005. cap. 34. <http://www.st.ewi.tudelft.nl/ koen/papers/MAC-chapter.pdf>.

LEE, H.; LEE, K.; SHIN, Y. Implementation and Performance Analysis of AES-128 CBC algorithm in WSNs. In: *The 12th International Conference on Advanced Communication Technology*. [S.l.: s.n.], 2010. p. 243–248. ISBN 978-1-4244-5427-3.

LIN, R.; WANG, Z.; SUN, Y. Energy efficient medium access control protocols for wireless sensor networks and its state-of-art. In: *Industrial Electronics, 2004 IEEE International Symposium on*. [S.l.: s.n.], 2004. v. 1, p. 669 – 674 vol. 1.

LISHA. *EPOS Project*. 2012. Disponível em: http://epos.lisha.ufsc.br Último acesso em: 20 Agosto 2012.

LUK, M. et al. Minisec: A secure sensor network communication architecture. In: *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*. [S.l.: s.n.], 2007. p. 479 –488.

MAGNO, M. et al. Combined methods to extend the lifetime of power hungry wsn with multimodal sensors and nanopower wakeups. In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*. [S.l.: s.n.], 2012. p. 112 –117.

MAINWARING, A. et al. Wireless sensor networks for habitat monitoring. In: *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002. (WSNA '02), p. 88–97. ISBN 1-58113-589-0. <http://doi.acm.org/10.1145/570738.570751>.

MELODIA, T.; VURAN, M. C.; POMPILI, D. The state of the art in cross-layer design for wireless sensor networks. In: *Proceedings of Eurongi Workshops on Wireless and Mobility, Springer Lecture Notes on Computer Science, LNCS 388*. [S.l.: s.n.], 2005.

MeshNetics. *Doc. M-252 08 v.1.0: ZigBeeNet - ZigBee Stack and Software Development Kit*. [S.l.], 2008.

MOTTOLA, L.; PICCO, G. P. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 43, p. 19:1–19:51, abr. 2011. ISSN 0360-0300. <http://doi.acm.org/10.1145/1922649.1922656>.

National Security Agency. *Skipjack and KEA Algorithm Specifications*. May 1998. Versão 2.0 Disponível em: http://cryptome.org/jya/skipjack-spec.htm Último acesso em: 20 Agosto 2012.

NICOLAS, E. P. et al. Sensor-based information appliances. *IEEE Instrumentation and Measurement Mag*, v. 3, p. 31–35, 2000.

OKAZAKI, A. M. *Algoritmo de Roteamento baseado em Colônia de Formigas com Heurísticas Configuráveis para Redes Sensores Sem Fio de Topologia Dinâmica*. 130 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2012.

OKAZAKI, A. M.; FROHLICH, A. A. Ant-based Dynamic Hop Optimization Protocol: a Routing Algorithm for Mobile Wireless Sensor Networks. In: *Joint Workshop of SCPA 2011 and SaCoNAS 2011 - IEEE GLOBECOM 2011*. Huston, Texas, USA: [s.n.], 2011. p. 1179–1183. ISBN 978-1-4673-0038-4.

OMG. *OMG Unified Modeling Language, Superstructure*. Fev 2009. <http://www.omg.org/technology/documents/formal/uml.htm>.

POLASTRE, J.; HILL, J.; CULLER, D. Versatile low power media access for wireless sensor networks. In: *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2004. p. 95–107. ISBN 1-58113-879-2.

REGHELIN, R. *Um Sistema Descentralizado de Localização para Redes de Sensores sem Fios usando Calibragem Cooperativa e Heurísticas*. Dissertação (Mestrado) — Federal University of Santa Catarina, Florianópolis, 2007. M.Sc. Thesis.

REGHELIN, R.; FROHLICH, A. A. A Decentralized Location System for Sensor Networks Using Cooperative Calibration and Heuristics. In: *9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Torremolinos, Malaga, Spain.: [s.n.], 2006. p. 139–146. ISBN 1-59593-477-4.

RHEE, I. et al. Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Trans. Netw.*, IEEE Press, Piscataway, NJ, USA, v. 16, n. 3, p. 511–524, 2008. ISSN 1063-6692.

ROGAWAY, P.; BELLARE, M.; BLACK, J. Ocb: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, ACM, New York, NY, USA, v. 6, n. 3, p. 365–403, ago. 2003. ISSN 1094-9224. <http://doi.acm.org/10.1145/937527.937529>.

RUFINO, L. M. *Integração do Protocolo SIP à Norma IEEE 1451 para Redes de Sensores Sem Fio*. 121 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2012.

SIPSER, M. *Introduction To The Theory Of Computation*. [S.l.]: Course Technology Ptr, 2006. (Computer Science Series). ISBN 9780534950972.

STANKOVIC, J. et al. Real-time communication and coordination in embedded sensor networks. *Proceedings of the IEEE*, v. 91, n. 7, p. 1002 – 1022, july 2003. ISSN 0018-9219.

STEINER, R. et al. Performance Evaluation of Receiver Based MAC Using Configurable Framework in WSNs. In: *Wireless Communications and Networking Conference (WCNC), IEEE*. [S.l.: s.n.], 2013. A aparecer.

STEINER, R. et al. An operating system runtime reprogramming infrastructure for wsn. In: *Computers and Communications (ISCC), 2012 IEEE Symposium on*. Cappadocia, Turkey: [s.n.], 2012. p. 621–624. ISBN 1530-1346.

STEINER, R.; MUCK, T.; FROHLICH, A. A Configurable Medium Access Control Protocol for IEEE 802.15.4 Networks. In: *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2010 International Congress on*. [S.l.: s.n.], 2010. p. 301 –308. ISSN 2157-0221.

STEINER, R.; MUCK, T.; FROHLICH, A. C-mac: A configurable medium access control protocol for sensor networks. In: *Sensors, 2010 IEEE*. [S.l.: s.n.], 2010. p. 845 –848. ISSN 1930-0395.

STROUSTRUP, B. *The C++ Programming Language*. 3rd. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN 0201700735.

SUN, H.-M. et al. An authentication scheme balancing authenticity and transmission for wireless sensor networks. In: *Computer Symposium (ICS), 2010 International*. [S.l.: s.n.], 2010. p. 222–227.

SWAIN, A.; HANSDAH, R. An energy efficient and fault-tolerant clock synchronization protocol for wireless sensor networks. In: *Communication Systems and Networks (COMSNETS), 2010 Second International Conference on*. [S.l.: s.n.], 2010. p. 1 –10.

WALTEYNE, T. et al. 1-hopMAC: An Energy-Efficient MAC Protocol for Avoiding 1-hop Neighborhood Knowledge. In: *IEEE SECON*. [S.l.: s.n.], 2006. p. 639–644.

WANNER, L. *Um Ambiente de Suporte a Execução de Aplicações em Redes de Sensores sem Fios*. 120 p. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2006.

WANNER, L.; OLIVEIRA, A. de; FROHLICH, A. Configurable medium access control for wireless sensor networks. *IFIP International Federation For Information Processing-Publications*, Springer, v. 231, p. 401–410, 2007.

YE, W.; HEIDEMANN, J.; ESTRIN, D. An energy-efficient MAC protocol for wireless sensor networks. In: CITESEER. *IEEE INFOCOM*. [S.l.], 2002. v. 3, p. 1567–1576.

ZHOU, Y.; FANG, Y.; ZHANG, Y. Securing wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials*, IEEE Communications Society, v. 10, p. 6–28, 2008.