PIERCARLO FERMINO SOARES

UM PROTOCOLO CROSS-LAYER HÍBRIDO PARA REDES DE SENSORES SEM FIO BASEADO EM BACKBONE

LAVRAS – MG

2018

PIERCARLO FERMINO SOARES

UM PROTOCOLO CROSS-LAYER HÍBRIDO PARA REDES DE SENSORES SEM FIO BASEADO EM BACKBONE

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

Dr. Tales Heimfarth
Orientador

Dr. João Carlos Giacomin
Coorientador

LAVRAS – MG
2018

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Soares, Piercarlo Fermino.
Um protocolo cross-layer híbrido para Redes de Sensores Sem baseado em Backbone / Piercarlo Fermino Soares. - 2018.
68 p. : il.

Orientador(a): Tales Heimfarth.
Coorientador(a): João Carlos Giacomin.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2018.
Bibliografia.

1. Redes de sensores sem fio. 2. Protocolo cross-layer. 3. Latência. I. Heimfarth, Tales. II. Giacomin, João Carlos. III. Título.

PIERCARLO FERMINO SOARES

UM PROTOCOLO CROSS-LAYER HÍBRIDO PARA REDES DE SENSORES SEM FIO BASEADO EM BACKBONE

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

APROVADA em 16 de Outubro de 2018.

Dr. Luiz Henrique Andrade Correia	DCC UFLA
Dr. Raul Ceretta Nunes	INF UFSM

Dr. Tales Heimfarth
Orientador

Dr. João Carlos Giacomin
Co-Orientador

LAVRAS – MG
2018

Para Ana Maria Theodoro da Silvas

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoa de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Agradeço a Deus, à minha Mãe e aos Professores Tales Heimfarth e João Carlos Giacomin.

640K são mais que suficientes para qualquer um.
William Henry Gates III

RESUMO

Redes de sensores sem fio são redes formadas por nós sensores autônomos alimentados por pequenas baterias. Isto torna a eficiência energética uma questão importante no desenvolvimento de protocolos para esse tipo de rede. Tal problema é abordado de várias formas, sendo que a técnica de ciclos de trabalho como forma de economia de energia é um dos métodos empregados. Nesta estratégia, o rádio transceptor e outros centros de consumo de energia do nó sensor permanecem desligados a maior parte do tempo. Essa estratégia causa um atraso de propagação de mensagens entre nós vizinhos, resultando em um aumento da latência fim-a-fim na rede. Este trabalho apresenta um novo protocolo cross-layer baseado em backbone, cujo objetivo é a redução da latência fim-a-fim com impacto limitado no consumo de energia. O BWise baseia-se no WiseMAC e utiliza ciclos de trabalho e encaminhamento de mensagens através de um backbone. Em simulações onde o backbone é utilizado para o transporte de mensagens, os resultados mostraram que o protocolo implementado obteve uma redução na latência fim-a-fim quando comparado com o WiseMAC e com o X-MAC, protocolos da literatura, mantendo um consumo energético compatível.

Palavras-chave: Protocolo MAC, cross-layer, Redes de Sensores sem Fio, Backbone.

ABSTRACT

Wireless sensor networks are networks formed by autonomous sensor nodes, powered by small batteries. This makes energy efficiency a very important question in developing protocols for this type of network. This problem is addressed in several ways, and the technique of work cycles as a way of saving energy is one of the methods employed. In this strategy, the radio transceiver and other power centers of the sensor node remain off most of the time. This strategy causes a propagation delay of messages between neighbors, resulting in an increase in end-to-end latency in the network. Our work presents a new cross-layer protocol based on backbone, whose goal is the reduction of end-to-end latency with low change in energy consumption. BWise is based on WiseMAC and uses duty cycles and message forwarding through a backbone. In simulations where the backbone can be used for message transport, the results showed that the implemented protocol obtained a reduction in the end-to-end latency when compared to WiseMAC and X-MAC, protocols in literature, maintaining an energy consumption compatible behavior.

Keywords: MAC Protocol, cross-layer Wireless Sensor Networks, Backbone.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SÍMBOLOS

ACK_s Tempo de pacote de confirmação

CS_s Tempo de prospecção de canal

Ct Tempo de ciclo

D_b Cálculo de distância para o *backbone*

d_b Cálculo de distância para os nós do *backbone*

D_g Cálculo de distância para o geo-roteamento

d_n Distância do nó vizinho ao destino

$DATA_s$ Tempo de pacote de dado

R_b Distância do nó mais próximo até destino

RTS_q Quantidade de pacotes de preâmbulo

RTS_s Tempo de pacote de preâmbulo

S_b Distância do emissor até o seu ponto mais próximo do *backbone*

T_a Tempo do nó anterior

T_i Tempo ideal

T_m Novo tempo de acordar para o nó

T_m Tempo de contato do próprio nó

T_n Tempo de contato do nó vizinho

T_s Tempo corrente

X_d Coordenada X do nó destino

X_n Coordenada X do nó *n*

Y_d Coordenada Y do nó destino

Y_n Coordenada Y do nó *n*

SUMÁRIO

1 INTRODUÇÃO

Uma rede de sensores sem fio é um tipo de rede constituída por dispositivos eletrônicos chamados de nós sensores que se comunicam através de um enlace sem fio, através de ondas eletromagnéticas. O propósito desse tipo de rede consiste em monitoramento de fenômenos em um ambiente. Esses fenômenos podem ser relacionados à medicina, à guerra, à produção de alimentos, monitoramento de animais, ou qualquer fenômeno de interesse em que esse tipo de rede possa ser aplicado.

As redes de sensores sem fio são aplicadas, em locais onde uma rede cabeada não possa ser implementada, ou a implementação torna o custo muito alto. Redes sem fio comuns também poderiam ser utilizadas, no entanto esse tipo de rede necessita de fonte constante de energia o que pode impossibilitar seu uso ou criar custos indesejados.

Os protocolos de controle de comunicação das RSSFs são divididos em cinco camadas. Sendo camada de aplicação, transporte, rede, MAC e física (AKYILDIZ et al., 2002; KOCAKULAK; BUTUN, 2017).

Esse foco no consumo energético se deve a limitação que as RSSFs sofrem por contarem com fonte energética limitada. Como pode não existir a possibilidade de manutenção energética no nó é necessário utilizar a fonte energética da forma mais eficiente possível.

Uma das camadas que mais interfere no consumo é a camada de controle de acesso ao meio (MAC). Essa camada controla o momento em que o rádio estará ativo ou não. O rádio é a unidade que consome mais energia no nó sensor por ser o dispositivo de recepção e transmissão (AKYILDIZ et al., 2002; BACHIR et al., 2010; DOUDOU; DJENOURI; BADACHE, 2013).

A camada MAC é muito estudada por esse envolvimento com o consumo energético, existem diversos protocolos MAC que abordam de formas diferentes esse problema.

Uma das abordagens é a utilização de ciclos de trabalho. Os ciclos de trabalho alternam os períodos em que o rádio está ativo para receber ou enviar informações e períodos em que o rádio está desligado. Tal estratégia permite redução no consumo energético. Embora ela reduza o consumo energético, ela sozinha não obtém toda eficiência possível ou desejada. Existem ainda outras estratégias que juntamente com o ciclos de trabalho reduzem o consumo (BACHIR et al., 2010; DOUDOU; DJENOURI; BADACHE, 2013).

Nos protocolos MAC, existem os protocolo que utilizam agendamento, período ativo comum e assíncronos (CANO et al., 2011).

Nas abordagens que utilizam agendamento, para cada nó é atribuído um intervalo de tempo em que ele realiza transmissão ou recepção. Para isso cada nó deve saber o momento em que cada vizinho irá acordar (CANO et al., 2011).

Nos protocolos de período ativo comum, os nós são organizados para dormir e acordar ao mesmo tempo. Assim quando deseja transmitir um nó tem garantia que existirá vizinho para receber o dado (CANO et al., 2011).

Existem ainda os protocolos assíncronos. Esse protocolos não utilizam formas de sincronizar, os nós dormem e acordam independentemente dos outros nós da rede. Neste tipo de protocolo, a mensagem de dados é precedida por um longo sinal de aviso de intenção de transmissão. Ao acordar e ouvir tal sinal, o nós receptor se manterá acordado para receber a mensagem de dados (CANO et al., 2011).

Essa abordagem de ciclo de trabalho melhora o consumo energético, mas cria um problema de latência. Por respeitar os ciclos, os nós recebem um dado e aguardam o momento adequado para envio da mensagem. Ao aguardar o momento da mensagem uma latência é adicionada, pois ela ficará no nó até que o próximo nó no caminho de transmissão esteja com o rádio ligado e pronto para a recepção. Nas RSSFs existem estratégias que são utilizadas junto com os ciclos de trabalhos. Existem estratégias de redução de latência em redes que utilizam ciclos de trabalho, como o uso de protocolos anycast (CMAC, AGA-MAC), (N; CHOU; GHOSH, 2005; HEIMFARTH; GIACOMIN; ARAUJO, 2015), encadeamento de nós sensores (DMAC, EX-MAC), (LU; KRISHNAMACHARI; RAGHAVENDRA, 2004; LI; SHI; ZHANG, 2010), e uso de backbone (GB-MAC, PROC), (HEIMFARTH et al., 2014; MACEDO et al., 2004). Algumas dessas estratégias operam somente na camada MAC ou de Rede, enquanto outros utilizam uma abordagem *cross-layer*.

Uma abordagem que deixe de se localizar somente na camada MAC e tenha funcionalidades em outras camadas para conseguirem melhorar a economia de energia tem sido utilizada e pesquisada. Esse tipo de abordagem é chamada de *cross-layer* a qual camadas não diretamente conectadas conseguem interagir (TOKLU; ERDEM, 2014; RAULT; BOUABDALLAH; CHALLAL, 2014). Com o *cross-layer* é possível fazer com que a camada MAC e a camada de Roteamento trabalhem unidas buscando os melhores caminhos para gerar uma economia de energia. Esse tipo de abordagem cria possibilidades para busca de uma melhor troca entre consumo energético e uma baixa latência nas RSSFs (AKYILDIZ; VURAN; AKAN, 2006; MENDES; RODRIGUES, 2011).

O protocolo proposto pelo presente trabalho é um protocolo *cross-layer* assíncrono com utilização de preâmbulo para o envio de mensagens. Nesse trabalho é utilizada uma estratégia de *backbone* para redução de latência causada pelos ciclos de trabalho, assim reduzindo de forma global na rede a latência para as transmissões mas mantendo bom consumo energético.

O *backbone* é uma parte da rede em que os nós estão sincronizados entre si, criando um fluxo de dados contínuo e veloz para dados trafegarem.

Existem várias abordagens para esse problema de latência, o presente trabalho consiste na utilização de um protocolo de RSSF já conhecido, chamado WiseMAC (EL-HOIYDI; DECOTIGNIE, 2004), e considerado um dos melhores no consumo energético aliado à essa estratégia de *backbone* e escalonamento de horários entre os nós sensores para a diminuição da latência, sem ocorrência de *overhead*.

1.1 Justificativa

Redes que utilizam ciclos de trabalho para economia de energia sofrem aumento de latência. Esse problema tem sido motivo de pesquisa e várias abordagens foram propostas, no entanto existem estratégias ainda não utilizadas a serem investigadas tornando esse um problema em aberto (DOUDOU; DJENOURI; BADACHE, 2013). Abordagens *cross-layer* foram testadas e sua utilização obteve sucesso (VURAN; AKYILDIZ, 2010; MENDES; RODRIGUES, 2011; HEIMFARTH; GIACOMIN; ARAUJO, 2015).

Esse trabalho utiliza a abordagem *cross-layer* envolvendo MAC e Rede. O objetivo é a redução de latência em redes que utilizam ciclos de trabalho através de uma estratégia de semi-estruturação da rede *(backbone)* conectando partes da rede.

Para solucionar esse problema de latência a ideia da utilização de um *backbone* dentro da rede pode criar um caminho rápido de dados entre vários pontos, permitindo uma fluência de dados e ainda manter um consumo energético baixo.

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo deste trabalho é solucionar o problema de *sleep-delay* causado pelos ciclos de trabalho em uma rede que utiliza protocolos assíncronos, sem abrir mão da eficiência energética.

Para isso é necessário o desenvolvimento um protocolo *cross-layer* com utilização de ciclos de trabalho e escalonamento em parte da rede para RSSFs, que apresente baixa latência de transmissão de dados e baixo consumo energético. O protocolo *cross-layer* envolverá funções da MAC e de Roteamento, visando a redução da latência sem aumento do consumo de energia.

Este protocolo não é específico para uma rede convergente, com fluxo de dados preferencialmente para um nó sink. Os nós de origem e de destino serão quaisquer nós da rede a cada transmissão de dados.

1.2.2 metodologia

Para o desenvolvimento completo do trabalho, foram realizadas as seguintes atividades:

- Estudo de outros protocolos que utilizam ciclos de trabalhos.
- Desenvolvimento do protocolo da camada MAC.
- Desenvolvimento do protocolo da camada de Roteamento.
- Desenvolvimento do protocolo *cross-layer* no *GrubiX Wireless Network Simulator*.
- Desenvolvimento do Protocolo WiseMAC para comparação com o deste trabalho.
- Avaliação de desempenho dos protocolos considerando consumo energético total, consumo energético por transmissão e latência.

1.3 Contribuições do trabalho

A contribuição deste trabalho é a utilização de uma abordagem diferente em relação aos protocolos que utilizam ciclos de trabalho para conseguir eficiência energética e baixa latência. Ao invés da utilização de um roteamento com *anycast* para melhor aproveitamento da densidade da rede, o trabalho apresenta uma abordagem *unicast* capaz de reduzir a latência mantendo bom consumo energético utilizando uma semi-estruturação da rede com sincronização localizada. Também realiza comparação entre um protocolo com ciclos de trabalhos com o WiseMAC, XMAC e o protocolo resultante, todos utilizando comunização *unicast*.

1.4 Estrutura do trabalho

O presente trabalho está estruturado utilizando os seguintes itens: capítulo 2, Referencial teórico aonde estão apresentados os conceitos característicos das RSSFs necessários para sua compreensão, apresentação do protocolo WiseMAC, os trabalhos relacionados e uma conclusão referente ao capítulo. No capítulo 3 é apresentada a descrição do protocolo e suas características de funcionamento, os modelos de implementação que foram utilizados para a camada MAC e de Roteamento, o modelo de *clock-drift* utilizado e a definição da estruturação na rede. O capítulo 4 apresenta a validação e comparações feitas entre os protocolos, os parâmetros de simulação utilizados, os cenários de teste e os resultados obtidos. O capítulo 5 apresenta a conclusão e trabalhos futuros que podem se originar desse trabalho.

2 REFERENCIAL TEÓRICO

Neste Capítulo estão apresentados os conceitos que são necessários para compreensão e que foram necessários para a produção do trabalho apresentado.

2.1 REDES DE SENSORES SEM FIO

Uma rede de sensores sem fio é uma rede composta por dispositivos chamados de nós sensores, esses dispositivos realizam suas tarefas e se comunicam através de um enlace sem fio. O objetivo dos nós sensores é realizar monitoramento de ambientes para detecção de eventos de interesse e enviar essas informações através da rede (AKYILDIZ et al., 2002; VIEIRA et al., 2003; BACHIR et al., 2010; CANO et al., 2011; KOCAKULAK; BUTUN, 2017).

Um dos diferenciais das RSSFs é sua natureza distribuída, como esses tipos de redes contam com um número relativamente grande de nós, variando de alguns poucos até milhares de nós. Esses podem operar de forma cooperativa sem necessidade de uma infraestrutura e aproveitando o grande número de nós na disseminação da informação (AKYILDIZ et al., 2002; VIEIRA et al., 2003; KOCAKULAK; BUTUN, 2017; MATEEN et al., 2017).

O uso de um grande número de nós sensores em uma rede, requer que o custo individual de cada nó seja reduzido. A falta de uma infraestrutura de fornecimento de energia elétrica nos locais de monitoramento exige que cada nó sensor tenha sua própria fonte de energia, que normalmente é composta por uma pequena bateria. Isto leva à necessidade do uso de estratégias de economia de energia (LANGENDOEN; MEIER, 2010; BACHIR et al., 2010).

O interesse nesse tipo de rede vem crescendo devido à sua possibilidade de aplicação em locais onde redes comuns sem fio ou cabeadas não são de possível aplicação, ou seu custo é alto. Esse interesse tem levado ao investimento em pesquisa e desenvolvimento para melhora das RSSFs visando melhorar sua confiabilidade e operação (RAWAT et al., 2014).

2.1.1 Nós sensores

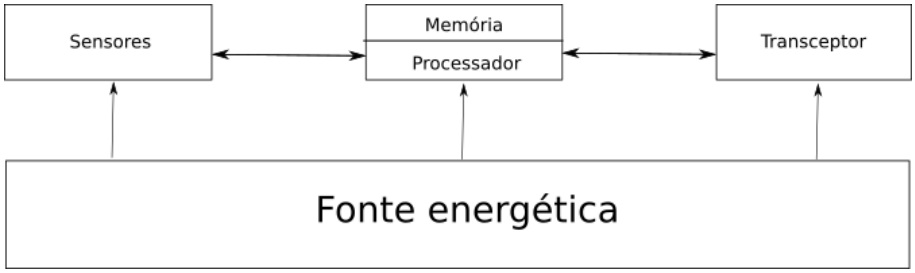
Os nós sensores são os principais componentes das RSSFs, eles são projetados de acordo com a aplicação da RSSF (FERENTINOS; TSILIGIRIDIS, 2007).

De forma geral os nós sensores são compostos por unidade de processamento, unidade de sensoriamento, unidade de comunicação e fonte energética (AKYILDIZ et al., 2002; KOCAKULAK; BUTUN, 2017). Essa composição geral pode ser ampliada de acordo com a ne-

cessidade da aplicação da RSSF em questão. A composição do nó está ilustrada na Figura 2.1, na qual é possível ver os componentes básicos.

Os sensores são responsáveis pelo sensoriamento do ambiente, o processador e memória são responsáveis pelo processamento e armazenamento dos dados e o transceptor realiza a comunicação no nó.

Figura 2.1 – Componentes do nó sensor



Fonte: Elaborada pelo autor

2.1.2 Consumo energético

O consumo energético nas RSSFs é um fator determinante a ser considerado. Esse consumo determina o tempo de operação que a rede terá. Existem vários protocolos da camada MAC que utilizam estratégias para redução de consumo energético obtendo bons resultados. Esses protocolos atingem esse objetivo se utilizando da estratégia de ciclos de trabalho, consistindo na ativação e desativação do rádio em determinados intervalos de tempo. Nessa estratégia de ciclos de trabalho o rádio é desligado por ser o componente do nó sensor que consome a maior quantidade de energia (BACHIR et al., 2010). Além de se desligar o rádio, os sensores também são desligados e o processador é colocado em modo de baixa energia, desativando alguns de seus componentes, como o conversor analógico/digital. Nesta condição, se diz que o nó sensor está no modo inativo (*sleep*). Nos curtos intervalos de atividade (modo *awake*), todos os componentes do nó sensor são ligados para que sejam realizadas a medições de variáveis ambientais e sejam feitas as eventuais comunicações entre os nós da rede. Quanto maior o tempo de *sleep* em relação ao tempo *awake*, maior a economia de energia e maior o tempo de operação da rede.

2.1.3 Arquitetura de comunicação

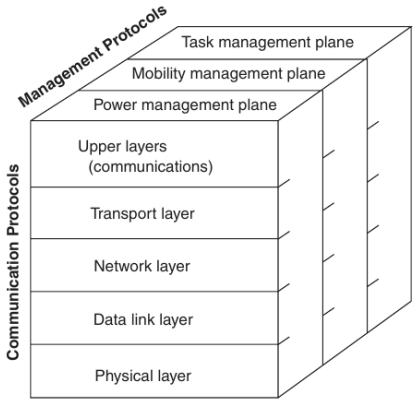
Uma RSSF apesar de suas particularidades é uma rede de computadores, ela segue a padronização da comunicação em camadas dos protocolos de comunicação que oferecem serviços

para as camadas adjacentes. Essa padronização para as redes de computadores é feita através dos modelos de referência OSI (Open Systems Interconnection model).

As RSSF utilizam uma variação dessa arquitetura OSI, sendo reduzida para cinco camadas, conforme Akyildiz et al. (2002), Kocakulak e Butun (2017) .

Na Figura 2.2, estão as camadas e é possível verificar os planos de gerenciamento que podem ser aplicados sobre elas. Esse planos não são essenciais nem obrigatórios. Nessa figura é possível ver as cinco camadas, definidas como Camadas superiores (*Upper Layers* que engloba as camadas de aplicação, sessões e o que for necessário. A camada de transporte (*Transport layer*), camada de Rede (*Network layer*), camada de Ligação de Dados (*Data link layer*) e a camada Física (*Physical layer*).

Figura 2.2 – Camadas na pilha de RSSFs



Fonte: (KOCAKULAK; BUTUN, 2017)

No presente trabalho serão apresentadas características das camadas de controle de acesso ao meio, contida dentro do *Data link layer*, na Figura 2.2, com um protocolo MAC e Rede, com um protocolo de roteamento, que são o enfoque do protocolo criado.

2.2 CAMADA DE REDE

Essa camada é responsável pelo roteamento de pacotes na rede. Ela deve ser planejada de acordo com a aplicação da RSSF, pois suas características tem impacto direto na forma como será o funcionamento da rede.

2.2.1 Protocolos da camada de Rede

Como estratégias para implementação de protocolos da Camada de Rede os mais comuns são o de roteamento plano, roteamento hierárquico e roteamento baseado em localização (AL-KARAKI; KAMAL, 2004).

No roteamento plano os nós da rede operam de forma igual, eles executam sensoriamento e transportam dados na rede até o ponto de entrega, seja para um nó comum ou para um nó *sink*.

No roteamento hierárquico os nós podem executar funções diferentes de outros nós.

No roteamento baseado em localização o posicionamento do nó é importante pois utilizando essa informação é que o nós decidem para qual nó enviar o dado até chegar no nó destinatário.

2.2.2 Roteamento baseado em localização

O roteamento baseado em localização, considera ser possível conhecer as posições dos nós seja através de transmissões ou conhecimento prévio da topologia pelos nós (AL-KARAKI; KAMAL, 2004).

O geo-roteamento consiste no cálculo de distância do nó emissor até o próximo nó, esse tipo de roteamento está dentro do grupo de protocolos baseados em localização. Essa abordagem utiliza a posição do nó corrente, seja o iniciador do envio ou algum nó que recebeu o pacote e precisa retransmiti-lo. O próximo salto é calculado utilizando a posição do nó atual e realizando a distância euclidiana dele e dos seus vizinhos em relação ao nó de destino. Após isso o próximo salto é o vizinho que estiver mais próximo do destino. Esse processo é realizado em cada salto do caminho até atingir o objetivo.

Uma forma comum de implementação do geo-roteamento é a utilização da distância Euclidiana entre dois pontos para calcular a menor distância entre os nós.

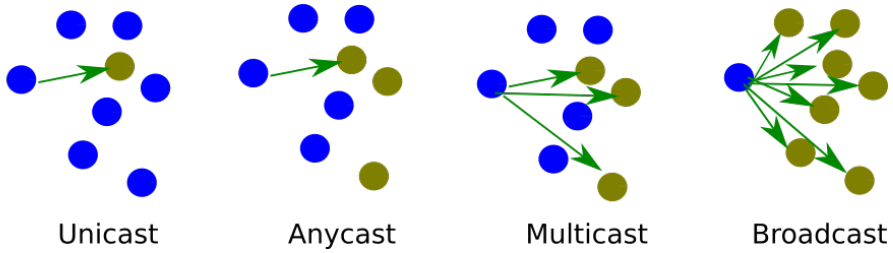
2.2.3 Metodologias de roteamento

A metodologia de roteamento é responsável pela forma como as mensagens são enviadas pela rede.

Existem quatro metodologias. *Unicast* sendo a metodologia que envia a mensagem somente para o nó escolhido. *Anycast* é o envio da mensagem para qualquer nó pertencente a um grupo de nós. *Multicast* faz o envio da mensagem para todos os nós dentro do grupo de

nós. E a metodologia de *Broadcast* envia para todos os nós, consistindo em uma difusão da mensagem pela rede. Essas metodologias estão ilustradas na figura 2.3.

Figura 2.3 – Tipos de roteamento



Fonte: Tik- 110.551 Internetworking Seminar

2.3 CAMADA MAC

A camada MAC tem como uma de suas funcionalidades controle de acesso ao meio, decidindo em que momentos o nó irá estar em modo de transmissão, recepção ou inativo (CANO et al., 2011).

A camada MAC é extremamente importante nas RSSFs pois de acordo com sua operação o nó irá ter consumo energético diferente. O ideal é que o nó opere pelo maior tempo possível com sua fonte energética. Para isso a camada MAC utiliza estratégias para o controle de operação.

Essas estratégias são de agendamento, períodos ativos em comum e assíncronos (CANO et al., 2011).

A estratégia de agendamento consiste em dividir em espaços de tempo entre os nós e cada um dele recebe seu espaço para tentar realizar comunicação. Dessa forma, nessa abordagem os nós somente acordam nos momentos de seus tempos para realizar comunicações. Também é necessário que o nó conheça os tempos de seus vizinhos antes de realizar envio de dados. Esse tipo de protocolo sofre com *overhead*, pois requer que a sincronização seja estritamente mantida em relação aos tempos atribuídos aos nós.

Nos protocolos com período ativo comum, existe a organização dos nós sensores de forma que eles durmam e acordem ao mesmo tempo. Esse tipo de protocolo tem um custo para que seja criado e mantida a ordem no agendamento de acordar e dormir dos nós.

Na estratégia assíncrona, sincronização não é requerida. Cada nó acorda e dorme aleatóriamente sem necessidade de conhecimento de outros nós da rede. Dentro dessa abordagem é necessário algum mecanismo que permita a comunicação entre os nós. Existem duas formas

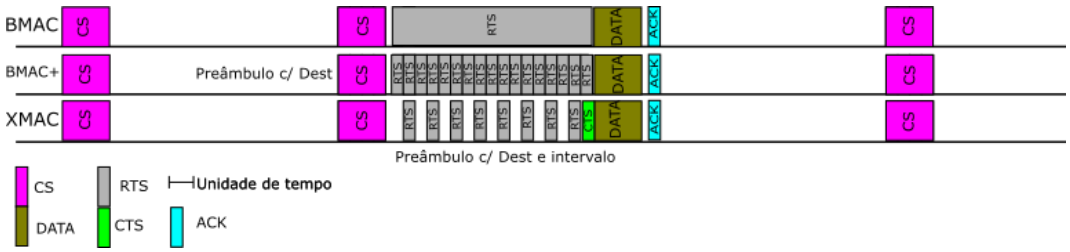
como isso é feito. Com utilização de preâmbulo de envio de mensagem ou transmissão iniciada pelo receptor. No envio de preâmbulos, os nós acordam em momentos para verificarem se o canal de transmissão está ocupado ou não. Quando um nó deseja realizar comunicação, ele envia uma mensagem de aviso, preâmbulo, a qual os nós que estiverem realizando prospecção do meio *carrier sense* detectarem essa mensagem, se mantém acordado para receber a mensagem. Nos protocolos de comunicação iniciada pelo receptor, o nó ao acordar envia uma mensagem alertando os nós que ele acordou, para caso exista desejo de realizar comunicação (CANO et al., 2011).

2.3.1 Protocolos assíncronos

Os protocolo assíncronos juntamente com a abordagem de utilização de preâmbulos, dentro dos protocolos MAC tem características interessantes que os tornam chamativos no contexto das RSSFs. Comparados com os outros tipos de protocolos, eles são simples com baixa complexidade pois não requerem mecanismos de sincronização ou coordenação. A técnica de amostragem de preâmbulos é vantajosa, pois em tráfegos baixos na rede ela obtém um consumo energético inferior, devido ao curto período de tempo de prospecção do canal.

Existem vários protocolos MAC assíncronos, dentre eles os mais comuns são o B-MAC (POLASTRE; HILL; CULLER, 2004), o X-MAC (BUETTNER et al., 2006) e o WiseMAC (EL-HOIYDI et al., 2003). A Figura 2.4 mostra o funcionamento do B-MAC, sua melhora e do X-MAC.

Figura 2.4 – Funcionamento dos protocolos B-MAC,B-MAC+ e X-MAC



Fonte: (KOCAKULAK; BUTUN, 2017)

2.3.2 Primeiros protocolos MAC CSMA

O protocolo pioneiro nas RSSFs, foi o S-MAC. Sua característica era da utilização de ciclos de trabalhos para economizar energia. Esses períodos de atividade são sincronizados em formas de grupos de nós (YE; HEIDEMANN; ESTRIN, 2002). Para isso existe um pacote de

sincronização no protocolo que objetiva manter esses ciclos sincronizados dentro dos grupos. Esse fator acaba por sofrer do problema de atraso por dormência *sleep delay*, por inserir latência entre os nós (BACHIR et al., 2010).

O S-MAC serviu como base para vários protocolos em RSSFs, por ser um protocolo pioneiro e introduzir uma abordagem inteligente que com aprimoramento resultaria em protocolos melhores.

O protocolo T-MAC é baseado no S-MAC. A diferença desse protocolo é o tempo de atividade não ser fixado e ajustável de acordo com o tráfego na rede. Para isso o T-MAC conta com um tempo de *timeout* que reduz o *overhead* criado pela escuta ociosa ajustando dinamicamente o período de atividade. Apesar da estratégia corrigir alguns problemas, ela não resolve o problema de *sleep delay* (DAM; LANGENDOEN, 2003; SINGH; BISWAS, 2012).

Outro protocolo baseado no S-MAC é o DMAC, que procura corrigir o *sleep delay* utilizando períodos de trabalho em grupos de níveis de nós em que eles acordam em sequência, permitindo um fluxo de dado contínuo. Isso funciona bem para taxas de dados baixa e o DMAC se destina a redes em que o fluxo de dados ocorre preferencialmente em uma direção (LU; KRISHNAMACHARI; RAGHAVENDRA, 2004; SINGH; BISWAS, 2012).

2.3.3 Protocolos com amostragem de preâmbulo

Essa abordagem é mais recente do que a abordagem de protocolos MAC com funcionalidade somente baseada em horários de trabalho agendados (BACHIR et al., 2010).

Um dos primeiros protocolos com essa abordagem de amostragem de preâmbulos WiseMAC (EL-HOIYDI; DECOTIGNIE, 2004). No protocolo, os nós respeitam o mesmo ciclo de trabalho, apesar disso os nós não são sincronizados por acordarem em tempos diferentes em relação à outros nós. Existe a prospecção do canal para detectar transmissões. Nesse protocolo após um primeiro contato de transmissão, os nós começam a formar uma tabela populada pelos ciclos de trabalho dos nós vizinhos. Após isso as transmissões passam a utilizar somente a quantidade de preâmbulos necessária para um eventual desvio de *clock-drift*. Esse protocolo sofre com o *sleep delay* (EL-HOIYDI; DECOTIGNIE, 2004; SINGH; BISWAS, 2012).

Outro protocolo que utiliza essa abordagem é o B-MAC (POLASTRE; HILL; CULLER, 2004). Nesse protocolo nenhuma sincronização entre os nós acontece. Todos os nós seguem o mesmo ciclo de trabalho e executam a prospecção do canal em seus ciclos de trabalho. Para realizar transmissão, o nó que deseja realizar um envio de mensagem, precisa enviar um preâm-

bulo longo de reserva do canal (POLASTRE; HILL; CULLER, 2004; SINGH; BISWAS, 2012). Esse preâmbulo longo causa atraso na transmissão dos dados pois ele envia todos os preâmbulos sem interrupção, para após isso enviar o dado. Além disso, o B-MAC causa desperdício de energia devido ao *overhearing*, visto que todos os vizinhos se manterão acordados até o final do envio de DATA.

O B-MAC foi base para outros protocolos, que buscavam melhorar problemas que ele sofria. Um deles o B-MAC+, que consistia na quebra do preâmbulo longo e pequenos preâmbulos buscando melhora no consumo energético. Além disso o preâmbulo curto carrega o endereço do nó para o qual a mensagem se destina. Desta forma os demais vizinhos podem voltar ao estado de baixa energia logo após receber um preâmbulo curto, evitando o *overhearing* (AVVENUTI et al., 2007; SINGH; BISWAS, 2012).

O protocolo X-MAC, baseado também no B-MAC, utiliza o preâmbulo curto com a diferença que existem pequenos intervalos entre os preâmbulos. Esses intervalos permitem que o nó que ouviu o preâmbulo responda com um pacote de liberação de envio, assim o X-MAC finaliza os envios de preâmbulo e envia o dado. Essa técnica reduz o consumo energético e também a latência em relação ao B-MAC (BUETTNER et al., 2006; SINGH; BISWAS, 2012).

2.3.4 Protocolos escalonados

Um protocolo que utiliza de uma forma de sincronização através de escalonamento é o D-MAC, que foi baseado no S-MAC. O escalonamento, consiste em fazer com que os nós da rede acordem em sequência para que a mensagem trafegue. A ideia dele é construir uma árvore à partir do *sink* que consiste nos nós de transmissão das informações até o *sink* criando níveis nessa árvore que determinam os ciclos de trabalho dos nós em cada nível. Criando um escalonamento dos níveis mais altos até os mais baixos no sentido *sink* (LU; KRISHNAMACHARI; RAGHAVENDRA, 2004). Além disso o protocolo conta com agregação de dados para quando o tráfego na rede é baixo.

Esse protocolo considera que a direção de comunicação preferencial é da rede para o *sink*. Em outras situações essa abordagem poderia não funcionar corretamente.

O EX-MAC (LI; SHI; ZHANG, 2010), baseado no X-MAC, utiliza uma forma de reserva de canal em multi-saltos para garantir um escalonamento ao longo da rede para que a latência seja reduzida e manter uma boa eficiência energética. Esse protocolo ainda assume que a rede funciona como nós que se reportam à uma estação base sendo bem aplicado em redes

para detecções de eventos (LI; SHI; ZHANG, 2010). O EX-MAC é adequado para redes aplicadas a detecção de eventos, em que as mensagens ocorrem em rajadas. A primeira mensagem é transmitida lentamente, devido ao *sleep-delay*. Mas esta primeira mensagem carrega uma informação de sincronização dos nós que estão no caminho do nó destino (*sink*). Esta informação é utilizada para escalonar os tempos de atividade dos nós no caminho, reduzindo a latência para as próximas mensagens.

2.4 PROTOCOLO WiseMAC

O protocolo WiseMAC utiliza a técnica de amostragem de preâmbulo para poupar energia em um meio ocioso. O WiseMAC utiliza ciclos de trabalho como forma de economizar energia, alternando curtos períodos de atividade com longos perìodos inativos. Os nós sensores mantêm uma tabela em que são registrados de forma aproximada os horários de atividade de seus vizinhos. Para realizar comunicação, um nó transmissor aguarda a aproximação do horário de atividade de seu vizinho, quando inicia o envio de um sinal de rádio avisando seu interesse em transmitir dados. Este sinal é denominado preâmbulo. Ao entrar em atividade, o nó receptor recebe o preâmbulo e aguarda o envio de dados. O transmissor envia a mensagem de dados logo após o sinal de preâmbulo e ao final do recebimento do dado, o receptor envia uma confirmação de recebimento.

Essa sincronização local nada mais é do que o nó que enviou uma mensagem armazenar em uma estrutura de dados interna o tempo em que ele conseguiu realizar comunicação com o nó. Isso se limitando somente a nós com apenas um salto de distância.

Embora sincronização possa causar *overhead*, isso não acontece no WiseMAC pois a informação de sincronização é carregada pelo pacote de confirmação (ACK).

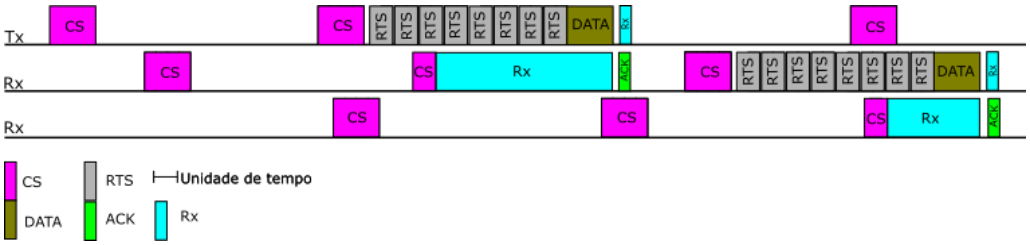
2.4.1 Funcionamento do WiseMAC

No WiseMAC quando não existe informação sobre períodos de operação dos nós vizinhos, um nó que desejar transmitir uma mensagem irá fazer o envio de um preâmbulo chamado de longo, esse pode ser um preâmbulo grande, ou pequenos preâmbulos sendo enviados em sequência. Esse preâmbulo é do tamanho de um ciclo de operação, assim em algum momento esse preâmbulo será interceptado pelo nó que receberá a mensagem.

Diferente de outros protocolos MAC que utilizam CSMA, o WiseMAC não conta com pacote de liberado para envio (CTS), quando um nó escuta um preâmbulo (RTS) ele somente

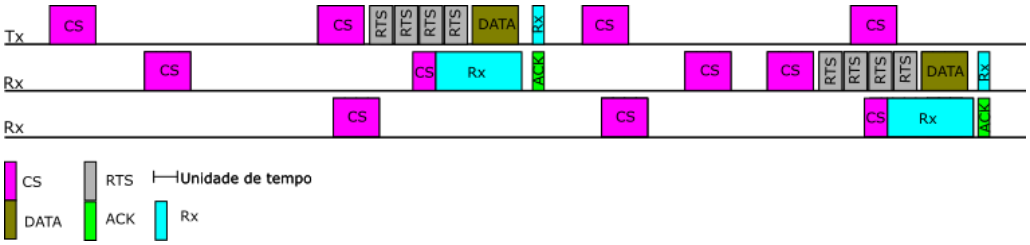
mantém seu rádio em modo de recepção. Ao final da sequência de preâmbulos o nó que está enviando simplesmente fará o envio do dado (DATA). Ao receber o dado, o nó irá fazer o envio de um pacote de confirmação (ACK). Esse pacote irá carregar também a informação de sincronização utilizada pelo protocolo. Todo esse processo está ilustrado na figura 2.5 o comportamento quando não existe informação de sincronização de vizinho. Na figura 2.6 é visível a redução de preâmbulos quando o vizinho já é conhecido.

Figura 2.5 – Funcionamento do WiseMAC sem informação de sincronização



Fonte: Elaborada pelo autor

Figura 2.6 – Funcionamento do WiseMAC com informação de sincronização



Fonte: Elaborada pelo autor

Esse modo de comunicação acontece sempre dessa forma no WiseMAC. No entanto quando já aconteceu uma comunicação e o nó que deseja enviar já possui informação de sincronização, ele se programará para acordar somente quando estiver próximo do tempo de trabalho do nó para qual enviará a mensagem e enviará um preâmbulo curto (EL-HOIYDI et al., 2003; EL-HOIYDI; DECOTIGNIE, 2004). Isso reduz a quantidade de preâmbulos utilizados diminuindo a quantidade energética consumida (LANGENDOEN; MEIER, 2010).

2.4.2 Clock drift

Os nós sensores utilizam um cristal oscilador para marcação de tempo. Esse oscilador pode sofrer de variações de nó para nó, devido a mudanças de temperatura, condições elétricas e outros fatores. O *clock-drift* é a variação que acontece nesse oscilador, criando diferenças de tempo entre nós da mesma rede (BRZOZOWSKI; SALOMON; LANGENDOERFER, 2010).

No WiseMAC para não ocorrerem problemas com *clock-drift*, o preâmbulo curto tem seu tamanho calculado baseado no tempo entre comunicações dos nós que será feita a transmissão. Com isso o protocolo estima um *clock drift* e modifica o tamanho do preâmbulo. Esse preâmbulo começa a ser enviado de forma que metade dele seja enviada antes e depois do tempo de sincronização, para garantir que o nó ouça-o independente de sua direção de *clock drift*.

2.4.3 Desempenho do WiseMAC

O WiseMAC tem um desempenho energeticamente eficiente devido ao número reduzido de preâmbulos e ausência de *overhead* para sincronização. É considerado um dos protocolos com comportamento energético mais consistente comparado a outros protocolos em diversos tipos de cenários (LANGENDOEN; MEIER, 2010).

Apesar disso ele sofre do problema de *sleep delay*. Quando não sincronizado o tempo de espera é aproximadamente um ciclo de trabalho completo. Quando ocorreu sincronização a comunicação entre o transmissor e o receptor é estabelecida em um tempo aproximadamente igual à metade do tempo de ciclo de trabalho $(t/2)$. Apesar de se tratar de uma espera relativamente normal, tal situação causa um aumento de latência se comparado à uma situação em que o nó recebe o dado e já começa seu envio (LANGENDOEN; MEIER, 2010).

2.5 Protocolos Cross-Layer

Os protocolos *cross-layer* são protocolos que utilizam a abordagem de comunicação entre camadas não diretamente relacionadas, buscando melhorar o funcionamento geral unindo comportamentos de camadas diferentes (MENDES; RODRIGUES, 2011). Essa abordagem permite melhoras em mais de uma camada produzindo um resultado final melhor considerando consumo energético e latência.

Esses tipos de protocolos podem ainda unificar funcionalidades ou realizar funcionalidades de outras camadas dentro de uma única. Além disso essa abordagem permite com essa unificação melhorar a performance da rede desenvolvendo protocolos unificados entre camadas (AKYILDIZ; VURAN; AKAN, 2006).

2.6 TRABALHOS RELACIONADOS

2.6.1 WiseMAC-HA

O protocolo WiseMAC-HA, *WiseMAC High Availability* (ROUSSELOT; DECOTIGNIE, 2010) aborda a melhora da latência do WiseMAC normal.

A forma como ele busca isso é utilizando dois modos de operação no protocolo. Um deles é um funcionamento normal do WiseMAC. O outro é um funcionamento intenso, aonde o rádio é mantido ligado para recepção e transmissão rápida. O nó de funcionamento intenso é um nó de maior capacidade que pode estar conectado a um dispositivo de saída da rede. Nele existe uma distinção física dos nós da rede. E são considerados nós que estão somente a um salto do nó de funcionamento intenso. Esse protocolo utiliza uma rede hierarquizada, com dois tipos de nós.

2.6.2 Protocolos com *backbone*

Um protocolo *cross-layer* que une características da camada de Rede e MAC é o GB-MAC (HEIMFARTH et al., 2014). Esse protocolo é baseado no X-MAC e utiliza um *backbone* para criar uma via de transmissão rápida de dados. A abordagem do protocolo é diminuir o tempo de ciclo entre dormência e acordar dos nós que fazem parte do *backbone*, assim aumentando a disponibilidade para transmissão dos nós. Diferentemente do DMAC e do EX-MAC, o GB-MAC acelera a transmissão de dados tanto na direção da rede para o *sink* quanto na direção contrária. No GB-MAC não é utilizada nenhuma forma de sincronização dos nós sensores. O *backbone* é reorganizado periodicamente para que seja feito o balanço no consumo de energia da rede.

Outro protocolo que faz uso de *backbone* é o PROC (MACEDO et al., 2004). Esse é um protocolo de roteamento da camada de Rede para RSSFs. Nele um *backbone* é construído periodicamente entre os nós e o *sink*. A construção do *backbone* é realizada em duas etapas. Aonde são eleitos nós que coordenam o processo de construção e os que não. Com a dinamicidade do *backbone* as rotas são atualizadas e isso permite que sejam adaptadas às mudanças de topologia da rede.

2.7 CONSIDERAÇÕES FINAIS

As redes de sensores sem fio tem suas aplicações e podem ser utilizadas para diversas funções devido às suas características versáteis e custo relativamente baixo. São redes que necessitam de um projeto adequado pois elas são específicas para a situação em que ela será aplicada.

No escopo dessas redes um problema estudado é a disponibilidade energética limitada. Para isso sempre são considerados fatores que permitam um consumo eficiente da fonte energética disponível.

Diretamente relacionado a isso está a camada MAC, responsável pelo controle de atividade do nó sensor. Uma das funções da camada MAC seria acordar o nó somente quando fosse o momento exato de necessidade, realizasse a transmissão e dormisse imediatamente, diminuindo o tempo de trabalho do nó ao mínimo possível.

Com esse objetivo existem diversos protocolos assíncronos que buscam conciliar consumo energético com baixa latência, tentando reduzir ao mínimo os ciclos de trabalho dos nós.

Dentre esses protocolos o WiseMAC se destaca, pois ele funciona energeticamente muito bem (LANGENDOEN; MEIER, 2010). Em contrapartida sua latência não é ótima, sendo essa a desvantagem pelo baixo consumo energético.

3 PROTOCOLO BWISE

3.1 Descrição do protocolo

O BWise (*Backboned WiseMAC*) é um protocolo cross-layer assíncrono para redes de sensores sem fios, que envolve funções das camadas MAC e de Rede. É baseado no protocolo WiseMAC, utiliza ciclos de trabalho como forma de economizar energia. Divide os nós da rede em dois tipos, conforme o modo de operação. Os nós do tipo 1 executam o protocolo WiseMAC tradicional, tendo latência média variando de um salto igual à metade de um tempo de ciclo de trabalho ou um ciclo de trabalho completo, de acordo com existência de contato prévio com o vizinho. Os nós do tipo 2 utilizam o protocolo WiseMAC, com uma forma de encadeamento dos tempos de atividade entre os vizinhos. Esses nós fazem parte de um conjunto que estabelece uma rota de alta velocidade, denominada *backbone*. O tempo de ciclo de trabalho é comum a todos os nós da rede, tanto do tipo 1 quanto do tipo 2. A escolha dos nós do *backbone*, tipo 2, é feita de forma automática no início da operação da rede.

O escalonamento dos nós do tipo 2 consiste em modificar o tempo de acordar de cada nó dentro do *backbone* para que eles se sincronizem, e não mantenham o comportamento do WiseMAC de espera de tempo para realizar a transmissão. O escalonamento dos períodos ativos dos nós do *backbone* é feito de forma semelhante ao do protocolo DMAC (LU; KRISHNAMACHARI; RAGHAVENDRA, 2004).

Para que a mensagem de dados trafegue mais rapidamente pelo *backbone*, é feito um escalonamento nos tempos de atividade dos nós do tipo 2. As mensagens trafegam pelo *backbone* sempre em uma mesma direção. Os nós mais à frente entrarão em atividade poucos instantes após seu vizinho anterior. Desta forma, quando um nó termina de receber uma mensagem de data, o vizinho à frente estará pronto para receber esta mensagem e encaminhar para o próximo. Desta forma, podem ser usados preâmbulos curtos.

Dentro desse *backbone* o dado é recebido e retransmitido sem espera. As mensagens de dados geralmente se originam em nós do tipo 1 e se destinam a outro nó do tipo 1. As transmissões dessas mensagens podem ocorrer somente através de nós do tipo 1 ou podem, eventualmente, se beneficiar do uso do *backbone*, com consequente redução de latência.

Apesar dos nós terem comportamentos diferentes, todos os nós tem a mesma capacidade de se tornar um nó do *backbone*.

O protocolo é do tipo *cross-layer*. A camada MAC tem como função sincronizar os nós tipo 2 e operar como um WiseMAC comum no caso dos nós tipo 1, e a camada de roteamento tem como objetivo determinar por onde o dado deve trafegar na rede. As informações sobre o nó ser tipo 2 ou tipo 1 são comuns para ambas as camadas.

O desenvolvimento do protocolo foi realizado em um simulador de RSSFs baseado em eventos, chamado GrubiX. O Grubix é um simulador de RSSFs baseado no simulador ShoX (LESSMANN; HEIMFARTH; JANACIK, 2008).

Todo o protocolo foi desenvolvido em linguagem Java, utilizando as funcionalidades fornecidas pelo Grubix. Foram desenvolvidos um protocolo de roteamento e um protocolo MAC que compõem o protocolo final.

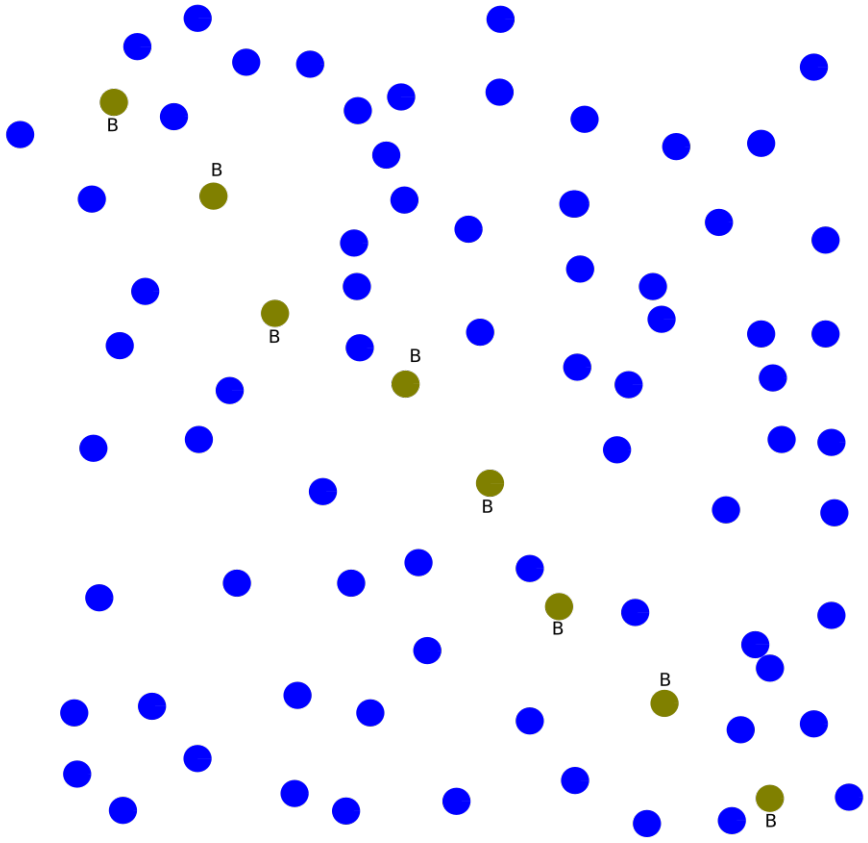
3.2 Definição do Backbone

O *backbone* é uma via de transmissão rápida de mensagens. Esse tipo de estrutura pode ter diversos formatos, como formato de árvore, topologia estrela, circulo. Os componentes desse *backbone* são nós que operam de forma diferente, realizando transmissões rápidas de mensagens, para que o resultado seja essa via de transmissão rápida.

Um *backbone* pode ser definido estaticamente, pode ser definido através de eleição pelos nós, baseado em nível de energia disponível ou outras propriedades dos nós. Pode contar com nós diferentes em nível de hardware ou não.

A Figura 3.1 representa um esboço do *backbone* utilizado. O *backbone* é denotado pelos nós marrons e os nós azuis são os nós comum da rede. Esse *backbone* é linear e conecta a parte superior esquerda da rede até a inferior direita. O fluxo no *backbone* é também da parte superior da esquerda para direita na parte inferior.

Figura 3.1 – Esboço de forma do *backbone*



Fonte: Elaborada pelo autor

3.3 Protocolo da camada de roteamento

O protocolo da camada de roteamento foi implementado de forma híbrida com transmissão de mensagem *Unicast*. Utiliza uma abordagem de geo-roteamento e uma abordagem de busca do *backbone*.

3.3.1 Geo-roteamento

O geo-roteamento foi utilizado para compor parte do método de roteamento final.

A equação utilizada para o geo-roteamento foi 3.1. Essa é a equação do cálculo da distância euclidiana entre dois pontos. Essa equação é executada para todos os vizinhos do nó que deseja enviar a mensagem. O termo d indica para qual nó será o salto. X e Y representam a coordenada X e Y do nó vizinho quando acompanhados pelo termo $_n$ e do destino pelo termo $_d$. Após cada vizinho ter sua distância em relação ao nó destino calculada, o que estiver mais próximo dele será o salto da mensagem. Para todo salto esse processo é realizado.

$$d_n = \sqrt{(X_n - X_d)^2 + (Y_n - Y_d)^2} \tag{3.1}$$

3.3.2 Roteamento híbrido inteligente

Esse roteamento é chamado de inteligente pois é capaz de determinar se é melhor transmitir a mensagem através do *backbone* ou se o ideal é ir diretamente utilizando geo-roteamento. Ele determina o ponto de entrada ideal e saída para o nó destino (D). Sejam esses pontos interior ao *backbone* ou nas extremidades.

Outra característica é, caso a mensagem saia do *backbone* e de forma incorreta tente retornar ele é capaz de avaliar se esse é o comportamento correto.

O roteamento híbrido é capaz de determinar a direção do *backbone*, assim as mensagens só trafegam em um sentido. Quando o sentido da mensagem for oposto a ele, o roteamento trata isso ignorando o *backbone* e enviando a mensagem através do geo-roteamento.

3.3.3 Matemática do roteamento híbrido inteligente

O roteamento híbrido inteligente consiste em um conjunto de cálculos matemáticos para definir a utilização do *backbone* e calcular ponto de entrada e saída da mensagem.

Inicialmente quando um nó tipo 1 deseja transmitir um dado, sua camada de roteamento executa três cálculos através da distância euclidiana. O primeiro cálculo é a distância direta do nó até o nó destino não considerando o *backbone*. Após isso é calculada a distância do nó até o ponto mais próximo do *backbone* e a distância entre o ponto do *backbone* mais próximo ao nó de destino. Devido ao *backbone* ser um fluxo contínuo de dados com latência baixa os saltos dentro dele são considerados tendo zero de custo no cálculo.

Caso o valor da distância do ponto de saída e entrada somados sejam menores que a distância direta do nó ao destino, o protocolo optará pelo uso do *backbone*. Após a decisão tomada, o nó irá buscar atingir o *backbone* no ponto calculado como o melhor ponto referente a esse nó. Quanto a mensagem atingir o ponto ideal de saída, o nó de saída irá realizar um geo-roteamento até atingir o nó destino.

Caso não seja vantajoso utilizar o *backbone*, a mensagem será enviada utilizando geo-roteamento, visto que o uso do *backbone* acabará por custar mais para o trafego da mensagem.

Para o funcionamento a distância entre nós tipo 2 é considerada zero, pois esses nós não contribuem para latência, visto que eles consistem em nós de envio direto sem espera.

As equações utilizadas para o roteamento híbrido inteligente são as equações 3.2, 3.3, 3.4, 3.5 e 3.6.

A equação 3.2 representa a execução para cada vizinho do nó emissor o cálculo da distância com geo-roteamento, utilizando a equação 3.1. Desses resultados, o menor é selecionado representando o nó mais próximo do destino, que no geo-roteamento seria escolhido para envio da mensagem. Esse valor é armazenado para comparação.

Na equação D_g representa o valor da distância utilizando geo-roteamento. O termo d_n representa a equação 3.1. O termo R representa o nó destino da mensagem e o n representa todos os vizinhos do nó corrente em relação ao nó destino, N representa o conjunto de todos os nós vizinhos do nó corrente.

$$D_g = \min_{\forall n \in N} d_n(R,n) \tag{3.2}$$

A equação 3.3 representa o cálculo de distância euclidiana utilizando como nó destino os nós do *backbone*. Essa equação é utilizada para as equações 3.4 e 3.5

$$d_b = \sqrt{(X_n - X_b)^2 + (Y_n - Y_b)^2} \tag{3.3}$$

Na equação 3.4 todos os nós vizinhos do emissor tem sua distância calculada em relação a seu ponto mais próximo do *backbone*. Após isso o menor valor entre o vizinho e o ponto mais próximo do *backbone* é selecionado e esse vizinho será o destino do próximo salto da mensagem.

O termo S_b representa a menor distância entre nó emissor e o ponto mais próximo dele no *backbone*. O termo d_b representa a equação 3.3 com o parâmetro b como o valor do nó do *backbone* dentro do conjunto de nós do *backbone* B e S como o valor do emissor para cálculo da distância. O valor de b que gera a menor distância é o ponto de entrada do *backbone* para mensagem enviada a partir desse S.

$$S_b = \min_{\forall b \in B} d_b(S,b) \tag{3.4}$$

Semelhante à equação 3.4, a equação 3.5 executa o mesmo cálculo, mas agora no papel de emissor está o nó destino. Assim calculando a menor distância de algum ponto do *backbone* até esse nó que será o ponto de saída do *backbone*.

O termo R_b representa a distância do nó de saída do *backbone* até o destino. Na equação 3.5, d_b representa a equação 3.3 com passagem do termo R que é o destino final da mensagem e b como valor do nó do *backbone* dentro do conjunto de nós B que fazem parte do *backbone*.

$$R_b = \min_{\forall b \in B} d_b(R, b) \tag{3.5}$$

Após ambas as distância entrada e saída do *backbone* terem sido calculadas, sua soma é realizada em 3.6.

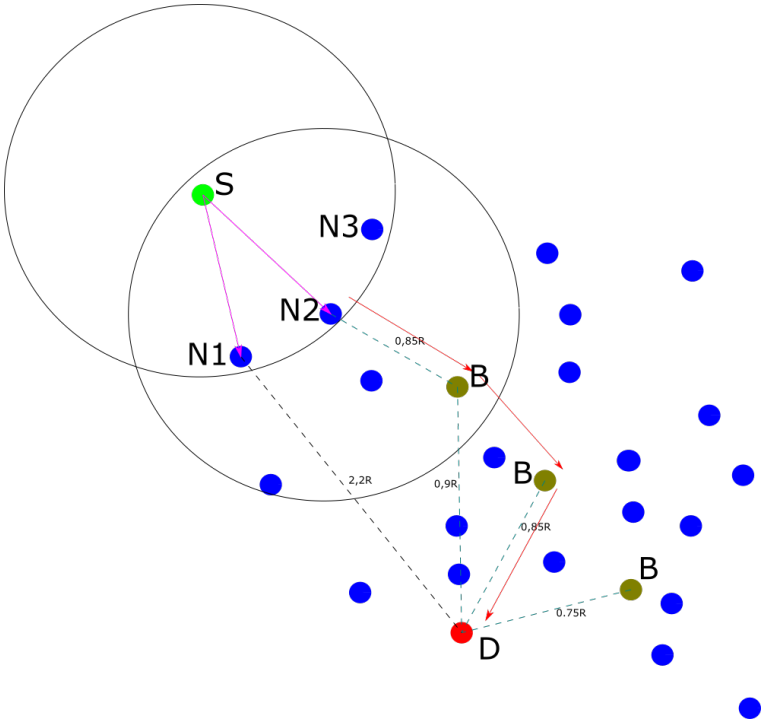
$$D_b = S_b + R_b \tag{3.6}$$

O protocolo após executar todas essas equações verifica qual das distâncias é menor entre D_b e D_g. Essa comparação serve para determinar se é melhor ir pelo *backbone* ou utilizando somente geo-roteamento.

Quando D_b é menor o protocolo verifica que o pacote será enviado via *backbone*. Assim é executado um geo-roteamente pelo nó tipo 1 caso ele esteja a mais de um salto de distância do ponto de entrada determinado para o *backbone*. Caso ele esteja somente a um salto de distância o envio é direto. Para a saída é executado o cálculo considerando o ponto de saída até atingir o nó receptor da mensagem utilizando geo-roteamento ou indo diretamente caso o nó esteja somente à um salto de distância do *backbone*.

A Figura 3.2, representa um envio envio de mensagem utilizando o roteamento inteligente. O nó originador da mensagem S realiza o cálculo direto até o destino D, o cálculo em relação a todos os pontos do *backbone*, representados por B e o cálculo em relação aos pontos B até o destino D. Nela é possível verificar que a distância indo pelo ponto B mais próximo de $N2$ e pelo ponto mais próximo de B até o D a distância é menor que indo diretamente por $N1$ com 3 saltos, que tem valor 2,2R.

Figura 3.2 – Roteamento inteligente



Fonte: Elaborada pelo autor

3.4 Protocolo da camada MAC

O protocolo da camada MAC implementado, consistiu em uma variação do WiseMAC, adaptada para se comportar adequadamente quando um nó é tipo 2 ou quando um nó é tipo 1.

Todos os nós sejam tipo 1 ou tipo 2 na camada MAC contam com duas estruturas de dados. Essas estruturas armazenam os dados de tempo de contato com o vizinho e o tempo do último contato.

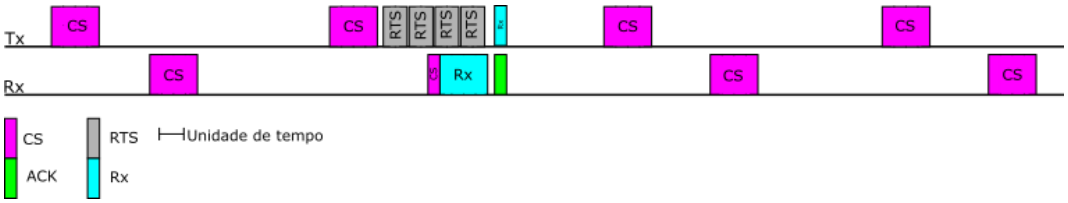
Isso foi interessante, pois por mais que o nó tenha um número n de vizinhos ele pode nunca contatar esses vizinhos, assim uma alocação estática seria desperdício. O acesso na estrutura também é simples. O nó faz uma busca pelo identificador do vizinho que ele deseja realizar o envio e assim obtém seu tempo de contato. E uma vez que já aconteceu o contato, caso necessite ser feita uma mudança no tempo do nó em questão, o acesso é pela chave e a alteração é feita sem problemas.

A estrutura de tempo do último contato armazena os dados que são utilizados para o cálculo da quantidade de preâmbulos que devem ser enviados considerando uma estimativa de *clock-drift*.

No início da transmissão de um dado a camada MAC primeiramente verifica na estrutura de tempo de contato para o caso de já conter informações sobre tempo de contato para o vizinho que a mensagem deve ser transmitida. Caso não contenha informações indicando um primeiro contato com esse vizinho, a camada MAC irá prosseguir na execução de prospecção do canal, se o canal estiver livre ele inicia o envio dos preâmbulos. O primeiro preâmbulo a ser enviado é um preâmbulo longo para cobrir o tempo de ciclo total, pois não existe previsão de quando o nó que receberá a mensagem acorda.

No protocolo implementado, o preâmbulo longo consiste de uma quantidade de preâmbulos curtos que somados resultam no total necessário para cobrir o ciclo de trabalho, formando assim o preâmbulo longo. A Figura 3.3 mostra o funcionamento desse preâmbulo longo, ilustrando a composição do preâmbulo longo através de preâmbulos curtos.

Figura 3.3 – Preâmbulo longo



Fonte: Elaborada pelo autor

Após o a sequência longa de preâmbulos ter sido enviado, o nó envia o dado e espera uma confirmação (ACK). Quando ele recebe essa confirmação ele insere os dados de tempo de contato e de último contato nas duas estruturas para consultar em uma próxima transmissão entre ele e esse mesmo vizinho. Após isso ele dorme e acorda novamente respeitando seu ciclo de trabalho.

Considerando o caso de um novo contato com um vizinho conhecido, a execução consiste em uma consulta nas duas estruturas de dados de tempo de contato e de tempo de último contato. Quando o nó encontra o tempo de contato do seu vizinho ele verifica se já deve começar a fazer o envio de preâmbulos. Caso não seja o tempo correto, ele volta a dormir pelo restante necessário para atingir o tempo de envio ideal. Quando o tempo ideal é atingido, uma consulta é feita no *hash* que armazena o tempo de último contato. Isso acontece pois é necessário adaptar a quantidade de preâmbulos em relação ao quanto o *clock-drift* desviou o relógio do nó. Após isso é iniciado o envio dos preâmbulos para transmitir o dado.

O tempo ideal é calculado de duas formas, de acordo com a função do nó. Para o nó tipo 1 seu cálculo é o mesmo do WiseMAC. O cálculo está representado na equação 3.7. Para o nó

tipo 2 a equação 3.8 representa o cálculo do tempo ideal. Ambas as equações são os modelos sem o *clock-drift*.

Na equação 3.7 Ti representa o valor final do tempo ideal em que deve acontecer a transmissão para o nó tipo 1. O termo $int(\frac{T_s}{Ct})$ da equação representa a utilização de somente a parte inteira da relação entre o tempo corrente T_s e Ct que representa o tempo de ciclo dos nós. Essa parte fracionária foi utilizada para que a transmissão possa verificar se o tempo de transmissão já passou ou irá acontecer ainda. T_n representa o tempo de contato do nó vizinho, CS_s se refere ao tempo que o nó gasta para realizar prospecção de canal. RTS_q significa a quantidade de de preâmbulos estática que é utilizada para a transmissão, e RTS_s representa o tempo gasto por cada preâmbulo enviado. A fórmula ao final determina um número de tempo ideal para transmissão considerando que no caso ideal, o nó receptor deve acordar exatamente no meio da sequência de preâmbulos.

$$T_i = \left\lceil \left(\frac{T_s}{Ct} \right) \right\rceil \times Ct + T_n - CS_s + RTS_q \times RTS_s - RTS_s \tag{3.7}$$

Na equação 3.8 está representado o cálculo do tempo ideal para o nó tipo 2. Na equação a diferença entre ela e a equação 3.7 acontece, para que a sincronização em relação ao nó anterior funcione. T_m representa o ciclo de trabalho do próprio nó, RTS_s, $DATA_s$ e ACK_s representam o tempo de transmissão para esses tipos de pacotes. Esse cálculo para o nó tipo 2 é feito para garantir que esse nó respeite a sincronização do *backbone* e considera que como as transmissões são feitas prontamente ao receber o dado, ele considera os três pacotes, pois no caso ideal, um nó receptor irá ouvir metade dos preâmbulos e um dado e necessitará enviar uma confirmação.

$$T_i = \left\lceil \left(\frac{T_s}{Ct} \right) \right\rceil \times Ct + T_m + RTS_s + DATA_s + ACK_s \tag{3.8}$$

No protocolo um nó identifica sua função de tipo 2 ou tipo 1 consultando a estrutura de armazenamento dos componentes do *backbone*. À partir dessa identificação sua operação é adaptada.

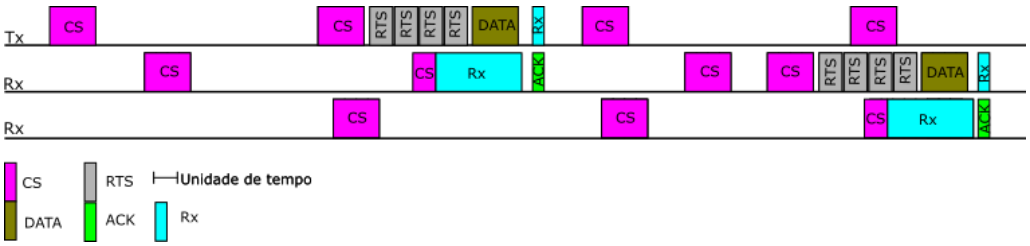
3.4.1 Nó tipo 1

O nó tipo 1 segue esse funcionamento sem alterações. Somente calculando as informações necessárias para calcular quantidade de preâmbulo e tempo ideal para início do envio.

Esses nós mantém seu ciclo de trabalho inicial e não o alteram, independente se realizam contato com outros nós tipo 1 ou com nós tipo 2.

Esse comportamento é visível na Figura 3.4. O nó Tx envia preâmbulos e encontra o o próximo nó, que recebe os dados e volta a dormir. O primeiro Rx acorda normalmente em seu ciclo e volta a dormir para respeitar o tempo de seu vizinho. Ao atingir o ponto de acordar do vizinho ele começa envio de preâmbulos.

Figura 3.4 – Funcionamento do nó tipo 1



Fonte: Elaborada pelo autor

3.4.2 Nó tipo 2

O nó tipo 2 tem funcionamento diferenciado. Ele precisa manter sua sincronização com os outros nós tipo 2.

Para isso, o nó após obter informação de transmissão na primeira passagem de mensagem pelo *backbone*, ele se regula para se adequar ao tempo de trabalho do nó anterior.

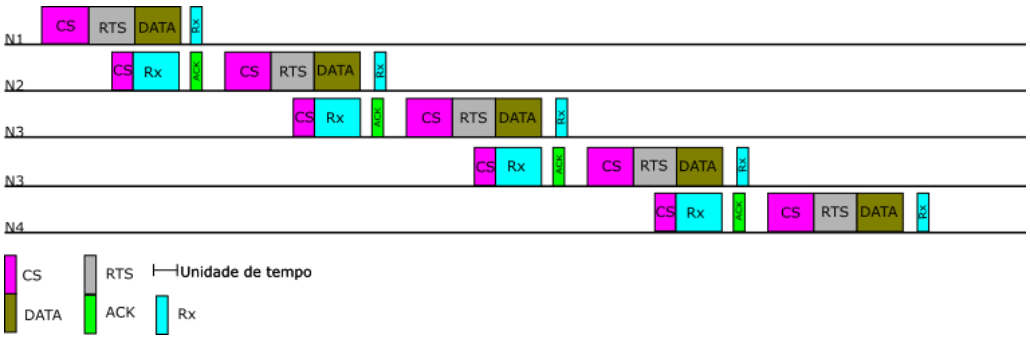
Diferente do WiseMAC, o nó tipo 2 se adapta ao tempo do nó tipo 2 anterior modificando seu horário de operação. Ele mantém a alternância entre acordar e dormir respeitando o ciclo de dormir. Para que os nós tipo 1 que obtiveram sua informação de sincronização sejam capazes de realizar novos contatos.

O novo horário de operação é calculado utilizando o tempo de acordar do nó do ponto de entrada, ele determina o tempo dos nós que estão em sua frente no *backbone*. Após a passagem da primeira mensagem, todos os nós estão sincronizados e realizam a transmissão imediata das mensagens. A fórmula 3.9 apresenta o cálculo interno do nó de seu novo horário de funcionamento quando ele é um nó tipo 2. O termo T_m representa o novo tempo de acordar para o nó. Esse tempo novo é calculado baseado no T_a que é o tempo do nó anterior, adicionado de 2 tempos de RTS_s, um $DATA_s$ e CS_s. Essa equação calcula esse tempo considerando que na transmissão ideal, o próximo nó deve sempre se atrasar seguindo esse padrão, para que o nó acorde sempre na metade da sequência de preâmbulos e também considerando que as transmissões anteriores consistem entre esses preâmbulos, dados, confirmação e prospecção do canal.

$$T_m = T_a + 2 \times RTS_s + DATA_s + ACK_s + CS_s \tag{3.9}$$

A Figura 3.5 mostra o funcionamento dos nós tipo 2. Todos os nós se encadeiam respeitando o novo tempo de sincronização, criando um fluxo contínuo de dados.

Figura 3.5 – Funcionamento do nó tipo 2



Fonte: Elaborada pelo autor

3.4.3 Particularidades entre nós tipo 2

Os nós tipo 2 ainda tem um comportamento diferenciado entre si. Os pontos de entrada e saída na camada de roteamento são dinâmicos e variam de acordo com a origem do dado. A camada MAC utiliza essa informação pois quando um nó tipo 2 é ponto de entrada ele precisa manter sua sincronização baseada na sincronização previamente feita com os outros nós seguintes no *backbone*. Para isso ele recebe o dado e dorme por um um tempo para que comece a transmissão no momento exato para o qual os nós estão sincronizados. Esse nó sabe que ele é um ponto de entrada, pois ao receber o pacote ele verifica o pacote criado pelo roteamento e verifica se quem lhe enviou o pacote é um nó tipo 1 ou tipo 2.

Essa característica foi necessária, pois se o ponto de entrada simplesmente começasse o reenvio para o próximo nó ao acabar de receber a mensagem, ele poderia perder o tempo de sincronização estipulado e não conseguir realizar o envio ou dessincronizar o *backbone*.

Para o ponto de saída também é necessário um comportamento diferenciado. O nó pode conter a informação do nó tipo 1 que deve receber o pacote caso já tenha ocorrido contato. Assim ele se adapta para dormir e acordar no momento ideal para inicio do envio para esse nó. Esses dois comportamentos são modelados conforme a equação 3.7.

Essa segunda característica é necessária semelhantemente à do ponto de entrada.

3.5 Cross-layer

A natureza *cross-layer* do protocolo BWise se deve ao compartilhamento de informações entre as camadas de rede e MAC, através de uma estrutura de armazenamento das informações do *backbone*.

Essa estrutura é uma estrutura de dados em que os identificadores dos nós do *backbone* estão inseridos em ordem de salto no *backbone*. Assim o protocolo é capaz de identificar a direção do *backbone* de forma simples e rápida. Sem ela o protocolo é incapaz de realizar sua função, visto que ele depende das informações para cálculo de ponto de entrada e saída.

Através dessa estrutura o nó quando realizando roteamento identifica seu comportamento, se ele é um nó tipo 1 ou tipo 2. A partir dessa informação ele começa realizar o roteamento adequado para o pacote. Seja pelo *backbone* ou fora dele.

Na camada MAC o *cross-layer* a estrutura de armazenamento é utilizada para identificar a função do nó tipo 2 e assim adaptar seu funcionamento para gerar a sincronização de *wake-up* pelos nós tipo 2 que não seria possível sem conhecimento do ponto de entrada que é calculado na camada de roteamento.

Embora isso não aconteça, caso a camada MAC alterasse os nós pertencentes ao *backbone* ou o roteamento o fizesse, as camadas estariam cientes da mudança.

3.6 Modelo de Clock-drift

Para aproximar a simulação do mundo real foi implementado um modelo de *clock-drift* considerando uma variação de *clock-drift* de até 20 partes por milhão (ppm) atrasando ou adiantando. Assim cada nó pode variar entre -20ppm até 20ppm (BRZOZOWSKI; SALOMON; LANGENDOERFER, 2010). Na equação 3.10 está a representação matemática desse modelo. A função rand(-20,20) gera um valor numérico aleatório nesse intervalo que dividido pela fração determina o *clock-drift* para o cada um dos nós.

$$C_n = \frac{rand(-20,20)}{10^6} \tag{3.10}$$

Esse modelo representa o *clock-drift* variando nesse intervalo. Isso para que exista a variabilidade real entre diferentes nós em relação ao seu *clock-drift*.

O modelo é executado inicialmente quando o nó executa sua primeira configuração. Quando isso acontece o nó determina seu *clock-drift*. Em cada ciclo de dormência que o nó

executa o *clock-drift* do nó é incrementado ou decrementado no seu tempo de acordar de acordo com o valor que foi determinado inicialmente. Criando uma variação similar ao que acontece com os cristais de quartzo dos dispositivos eletrônicos.

No contexto do nó que deseja enviar uma mensagem, ele executa uma estimativa do quanto o relógio do nó que deve receber a mensagem desviou em relação ao tempo de contato informado. Dependendo do resultado do cálculo o número de preâmbulos é modificado para cobrir esse desvio. Para esse cálculo foram considerados os piores casos, que seria quando os nós desviassem ou -20ppm ou +20ppm, assim mesmo que o nó desvie menos, essa estimativa abrange esses casos. Desta forma, o nó transmissor sempre considera a possibilidade de estar 40ppm adiantado em relação ao receptor ou 40ppm atrasado.

No nó emissor o cálculo de estimativa do desvio é feito sempre que existe um dado para ser transmitido. Esse cálculo é definido em uma relação entre o último contato com o vizinho e o tempo corrento. As equações 3.11 e 3.12 representam os cálculos para a mudança na quantidade de preâmbulos.

Na equação 3.11 está representado o cálculo da quantidade de tempo passada desde o último contato com o nó $T_s - T_c$, aonde T_s representa o tempo atual e T_c o tempo de último contato. Essa expressão foi multiplicada pela quantidade de variação que pode ocorrer devido ao *clock-drift* representada por $\frac{40}{10^6}$, considerando sempre o pior caso.

Está sendo utilizado o valor de 40ppm pois esse representa o desvio máximo entre transmissor e receptor.

$$E_n = (T_s - T_c)\frac{40}{10^6} \tag{3.11}$$

Na equação 3.12 o valor da equação 3.11 é utilizado e dividido pelo tamanho do pacote unitário de preâmbulo RTS_s. Isso resulta em um valor numérico, RTS_q, o qual indica a quantidade de preâmbulos à serem utilizados para solucionar o *clock-drift* acumulado.

$$RTS_q = \left\lceil \frac{E_n}{RTS_s} \right\rceil \tag{3.12}$$

É um modelo funcional mas que não considera variações devido à temperatura ou outros eventos externos. Esse modelo foi implementado para demonstrar a utilização do preâmbulo calculado em relação ao *clock-drift* nos nós e aproximar de situações reais que acontecem em RSSFs.

3.7 Protocolo MAC com modelo de Clock-drift

Descritos o Protocolo MAC inicial e o Clock-drift, o Protocolo MAC implementado consiste na união entre ambos.

O funcionamento do protocolo MAC, ocorre da mesma forma como descrito em 3.4, no entanto particularidades nas fórmulas foram adicionadas para que o modelo de *clock-drift* implementado funcionasse com o protocolo.

Para a fórmula dos nós tipo 1, a equação 3.13 representa a alteração necessária para utilização do modelo de *clock-drift*. Na equação 3.13 o novo termo $(2x(C_m xRTS_s)$ representa o *clock-drift* fazendo o cálculo da quantidade de preâmbulos. Esse termo é multiplicado por dois para criar o intervalo completo para mais e para menos da quantidade de preâmbulos.

$$Ti = \left\lceil \left(\frac{T_s}{Ct} \right) \right\rceil \times Ct + T_n - CS_s + (2 \times (C_m \times RTS_s) - RTS_s) \tag{3.13}$$

Somente essa equação precisou ser modificada. Isso se deve ao fato que por mais que o *clock-drift* exista em todos os nós, sejam tipo 1 ou tipo 2, a interferência no tempo ideal só acontece quando o comportamento é semelhante ao do WiseMAC. Nos nós que se comportam diferente, isso não é necessário pois quem controla a sincronização é o nó anterior, o próximo nó acorda em seu tempo estipulado, a diferença é que esse nó irá interceptar mais preâmbulos.

Para calcular os preâmbulos o nó utiliza o cálculo de estimativa e o duplica para criar o intervalo completo de preâmbulos.

O modelo de cálculo para o tempo de acordar também sofre a modificação, para considerar o *clock-drift*. Essa modificação acontece no momento de cálculo do *clock-drift* pela adição de uma janela de 100 unidades de tempo.

A janela permite corrigir problemas que podem ser causados quando o número de preâmbulos devido ao *clock-drift* aumenta em um número significativo, acaba por adição de um atraso entre cada transmissão. Com isso, em algum ponto da rede após um certo número de saltos o atraso causa total dessincronização impossibilitando o funcionamento dos nós tipo 2. Se a janela fosse dinâmica algum tipo de estratégia global coordenada por algum elemento da superior da rede seria necessário. Essa estratégia foi necessária pois é desejado que os nós sejam auto-coordenados, sem dependências externas, somente conhecendo posicionamento geográfico de nós e da posição do *backbone*. Isso sendo feito antes do início do funcionamento da rede.

4 VALIDAÇÃO DO PROTOCOLO

Essa sessão apresenta os cenários de avaliação e comparação entre o BWise e o protocolo o qual ele se baseia (WiseMAC (EL-HOIYDI; DECOTIGNIE, 2004)) e do protocolo X-MAC (BUETTNER et al., 2006). Os protocolos foram simulados no framework GrubiX, que é uma extensão do simulador baseado em eventos de RSSFs ShoX que utiliza Java e XML para modelagem de RSSFs (LESSMANN; HEIMFARTH; JANACIK, 2008). Os protocolos BWise e WiseMAC foram implementados no âmbito dessa dissertação, baseados em um protocolo MAC existente no framework. As simulações feitas com WiseMAC e X-MAC utilizaram um protocolo de geo-roteamento na camada de rede.

Foi usado o modelo de comunicação de raio único, com alcance de 40 metros e capacidade de transmissão de 250kbps. Não foram considerados erros de transmissão.

Na topologia da rede, os nós foram distribuídos aleatoriamente na área de atuação da rede formando uma distribuição de *Poisson* e em cada execução das simulações, a topologia da rede foi gerada novamente. Embora essa modificação da topologia da rede acontecesse, os nós que faziam parte do *backbone* foram mantidos em sua posição original. Não havia movimento de nós nos cenários simulados.

Nesse protocolo, o *backbone* é estático, em formato que se assemelha a uma linha reta direcional. Ele conecta dois pontos distantes da rede em uma forma direta, com os nós tipo 2 sendo os nós que estão no caminho entre os dois pontos distantes, eles formam uma via de alta velocidade direcional para tráfego de mensagens.

Ele conta com vinte nós, considerando a rede que um pacote consegue sair de um lado e atingir seu oposto mais distante com um número médio de vinte saltos. Embora essa estratégia tenha sido adotada, é possível utilizar outras formas de geração do *backbone*.

Todos os cenários foram livres de colisão, trafegando somente uma mensagem por vez na rede. Não foi considerado taxa de erros no canal de transmissão, assim toda mensagem enviada chegou em seu destino. Internamente, todos os nós possuíam as seguintes informações: posição do nó destino, posição de seus vizinhos e posição dos componentes do *backbone*. O *clock-drift* acontecia, no entanto como descrito no protocolo, seu impacto era corrigido pela estratégia de preâmbulos mínimos utilizados e para que o efeito do *clock-drift* causasse um impacto perceptível na rede, era necessário que um grande intervalo de tempo entre as transmissões acontecesse.

4.1 Parâmetros de simulação

Para a avaliação, os parâmetros utilizados foram alcance de transmissão do nó sensor, tamanho de pacotes de dados, tamanho do preâmbulo, tamanho do dado, tamanho do pacote de confirmação, densidade da rede, forma como é a distribuição dos nós na rede, formato e composição do *backbone* e quantidade de variação do *clock-drift*. Todos os parâmetros estão individualmente apontados no Quadro 4.1, que foram baseados nos parâmetros do AGA-MAC (HEIMFARTH; GIACOMIN; ARAUJO, 2015) e (BRZOZOWSKI; SALOMON; LANGENDOERFER, 2010).

Quadro 4.1 – Quadro de parâmetros da simulação

Alcance do Rádio	40m
Potência	60mW
Maior distância origem/destino	≈707m
Densidade da rede	2000 nós em 500x500m
Distribuição da rede	distribuição de Poisson
Tempo de ciclo	0,1s
Duração do pacote de preâmbulo	0,001024s
Duração do pacote de dados	0,002048s
Duração do pacote de confirmação	0,000512s
Tempo de prospecção de canal	0,002048s
Número de Execuções	60
Variação do *clock-drift*	+/-20ppm
Nós no *backbone*	20 estáticos

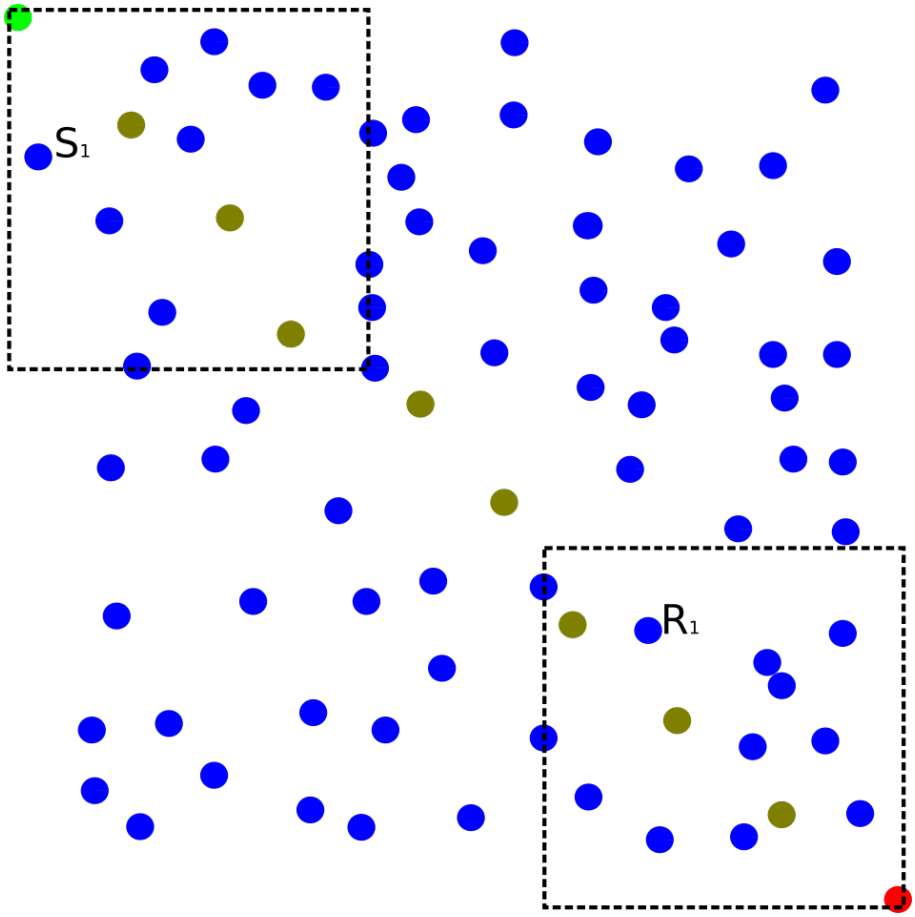
4.2 Cenário um

Neste cenário, a topologia da rede se mantinha aleatória, no entanto a seleção de nós que realizaram as transmissões estava limitada a dois quadrantes da área total da rede. Foi considerada uma área de $250000m^2$. O quadrante para escolha dos emissores estava posicionado nas coordenadas (0,0), (125,0), (125,125), (0,125). Para os nós receptores o quadrante de seleção era determinado por (375,375), (375,500), (500,500), (500,375). Ambos os quadrantes tinham área $15625m^2$. Dentro dessas áreas, existiam em média 125 nós candidatos a serem escolhidos para formar tupla emissor e receptor.

O *backbone* estava posicionado realizando uma conexão entre esses dois quadrantes. Nesse cenário a localização dos nós próximos ao *backbone* faziam com que o melhor caminho fosse ele, visto que sua latência era considerada zero dentro do roteamento proposto no BWise.

A Figura 4.1 representa um exemplo de esboço da topologia. Os nós azuis são nós tipo 1, os nós marrom são do tipo 2 e representam o formato do *backbone* estático utilizado para o BWise. O verde é o emissor da mensagem de escalonamento (mensagem de primeira configuração do *backbone*) e o vermelho o nó receptor, eles representam a maior distância em linha reta possível na rede que é ≈707m. Apesar disso, os nós verde e vermelho não são componentes do *backbone*. Os nós marrons, o nó verde e o vermelho são posicionados sempre no mesmo lugar, em contrapartida, os nós azuis têm sua topologia variada para cada simulação. Para mensagens de dados, a letra *S* representa um exemplo de emissor e *R* um exemplo de receptor, sendo escolhidos somente nos quadrantes demarcados.

Figura 4.1 – Topologia da rede utilizada para os Cenários um e dois



Para ambos os protocolos, foram utilizadas transmissões de 10, 20 e 30 mensagens entre nós podendo ser distintos ou não.

No caso considerado, foi avaliado o consumo energético resultante da transmissão de 10, 20 e 30 dados através da rede, utilizando o protocolo WiseMAC com geo-roteamento e o

BWise. Nesse cenário não foi utilizado o X-MAC pois a natureza do cenário não modifica sua performance em relação à latência e consumo energético, o que não acontece para o WiseMAC.

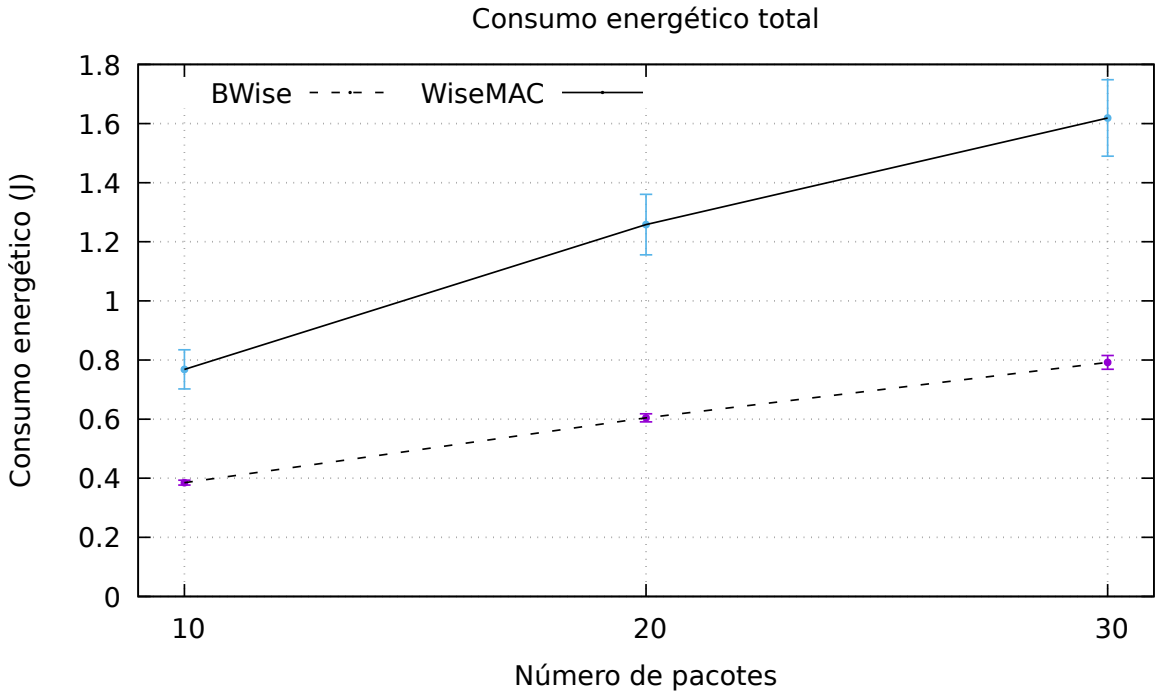
Para calcular o consumo energético, operações de ciclo de trabalho sem transmissão não foram considerados, por serem comuns à todos os nós e fazerem parte do funcionamento normal da rede. Foram considerados transmissões de RTS, DATA, ACK e o CS relativo ao evento de início de transmissão de mensagem.

No caso considerado, foram verificados o consumo total de ambos os protocolos e também o consumo considerando transmissões individuais.

4.2.1 Consumo energético total

Para o consumo total, em todos os números de mensagem enviados, o consumo energético foi crescente. A Figura 4.2 mostra os resultados energéticos de consumo total obtidos para 10, 20 e 30. A linha tracejada representa o consumo do BWise e a linha continua representa o consumo do WiseMAC.

Figura 4.2 – Consumo energético total



Fonte: Elaborada pelo autor

No WiseMAC, para toda transmissão entre vizinhos desconhecidos, o número de RTS enviados é equivalente a um ciclo completo de trabalho. No caso deste trabalho, que o ciclo de trabalho é 0,1 segundo, o WiseMAC utiliza 98 preâmbulos inicialmente entre esses vizinhos.

Caso ocorra reutilização de caminho, a quantidade de preâmbulos cai para um valor entre 2 e 98 dependendo do intervalo entre mensagens. Assim a Figura 4.2 ilustra o fato que quanto mais dados entre nós conhecidos ou caminhos conhecidos, menor o crescimento do consumo energético.

O BWise necessita desses mesmos preâmbulos quando a transmissão é feita entre nós tipo 1. Quando o dado trafega no *backbone*, existe uma tendência de uso de poucos preâmbulos, devido ao maior tráfego nessa região da rede.

Com isso, é possível verificar que o consumo do protocolo deste trabalho foi menor. Isso se deve ao fato do *backbone* estar muitas vezes sincronizado, utilizando a quantidade mínima de preâmbulos entre os nós tipo 2. Os nós tipo 1 seguiam o funcionamento normal do WiseMAC, no entanto a quantidade de saltos fora do *backbone* era menor que dentro.

O impacto da redução da quantidade de preâmbulos é visível na Figura 4.2 no consumo do WiseMAC. Isso se deve ao fato que o WiseMAC necessita de conhecimento de nós vizinhos para diminuir o gasto energético, assim quanto mais transmissões ocorridas pelos mesmo caminhos, menor a quantidade de preâmbulos.

Com isso é possível ainda inferir que o crescimento energético para o WiseMAC não foi o mesmo entre 10 e 20, e depois entre 20 e 30. Isso porque com 20 e 30 nós a repetição de caminhos ocorre com maior frequência, fazendo com que aja diminuição no consumo em relação às transmissões anteriores. Foram realizados testes com 40 transmissões também, que demonstraram esse comportamento não apresentando melhora visível. Na presença de uma quantidade grande de trafego na rede, o protocolo WiseMAC irá manter o comportamento de consumo reduzido.

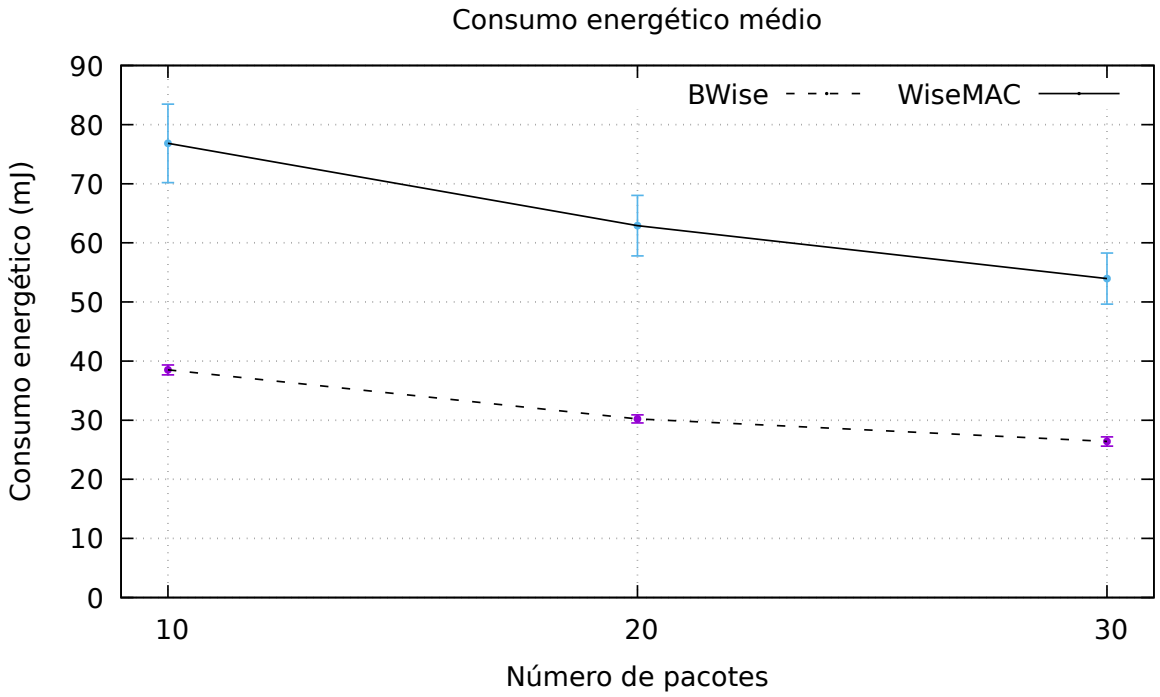
Esse comportamento não acontece no BWise. Nele, todos os nós tipo 2 irão em algum ponto estar utilizando somente comunicação com gasto reduzido e mantendo um consumo energético baixo. No entanto, o consumo no *backbone* irá aumentar sempre conforme o fluxo de dados nele aumenta. Esse consumo localizado é o preço a ser pago pela sincronização e velocidade de transmissão gerada pelo *backbone*, necessitando uma forma de alternância entre os nós componentes dele ao longo do tempo.

4.2.2 Consumo energético médio

O consumo energético médio se refere ao consumo relacionado à quantidade de transmissões e seu custo individual. Em ambos os protocolos esse consumo tende a diminuir.

A Figura 4.3 mostra o comportamento do consumo energético em ambos os protocolos em relação ao número de transmissões executadas. Novamente é possível verificar o teste para 10, 20 e 30 mensagens.

Figura 4.3 – Consumo energético por transmissão



Fonte: Elaborada pelo autor

A partir desses dados, é possível verificar que o custo de transmissões para ambos os protocolos diminui. Esse comportamento se mantém e estabiliza quando um número grande de nós obtém conhecimento de seus vizinhos.

O protocolo desse trabalho tem um consumo energético aproximadamente 4 vezes menor no início e essa relação diminuindo conforme o tempo, devido ao *backbone* sempre consumir o mínimo de energia. Quando os nós tipo 1 começam a reutilizar nós vizinhos e caminhos já conhecidos, o consumo geral também cai, pois nessa situação o nó tipo 1 diminui seu consumo e o *backbone* continua com seu funcionamento.

O WiseMAC tende a ter esse comportamento. Em um momento quando todos os nós da rede conheçam todos seus vizinhos esse consumo tende a ser baixo. Esse conhecimento total normalmente é impraticável. O WiseMAC se beneficia de qualquer conhecimento de vizinhos dos nós de transmissão. Qualquer repetição de transmissão ou reutilização de caminho gera melhora no consumo energético do WiseMAC.

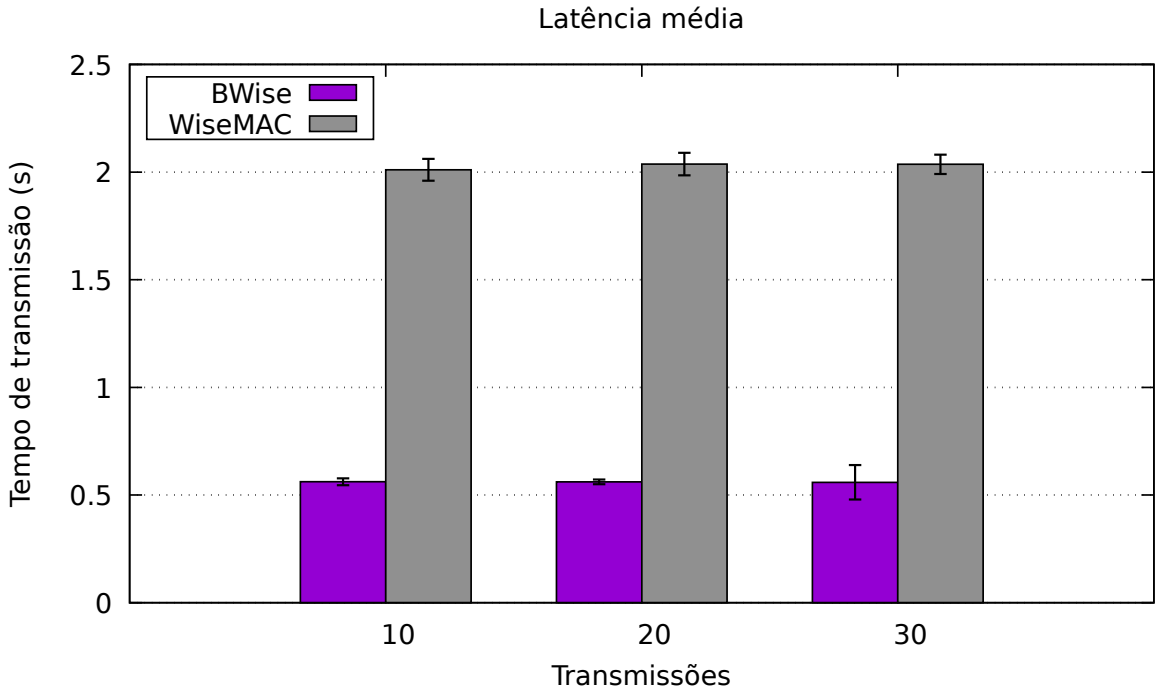
No contexto do BWise, esse comportamento é importante mas não é o fator determinante para melhora do consumo energético por transmissão, pois como dito o fluxo de dados é maior no *backbone* sendo ele o maior responsável pelo consumo.

4.2.3 Latência

Os protocolos que utilizam ciclos de trabalho sofrem comumente do atraso por dormência. O WiseMAC não é uma exceção disso. Ele é mais eficiente energeticamente (LANGENDOEN; MEIER, 2010), mas tende a manter uma latência constante e consideravelmente alta.

A Figura 4.4 demonstra isso para as execuções. Nela é possível observar a diferença de latência entre o WiseMAC e o BWise. Nela é possível verificar que a latência do WiseMAC é quase quatro vezes maior que a do BWise, independente do número de transmissões. Isso ocorre pela concentração de fluxo no *backbone* que está sincronizado para realizar transmissões imediatas ao receber o dado. Nele o dado chega e uma pequena espera é realizada e o dado é transmitido.

Figura 4.4 – Latência média



Fonte: Elaborada pelo autor

Essa diminuição da latência é localizada somente dentro do *backbone*, visto que externamente os nós se comportam como WiseMAC comum. Ele recebe o dado e aguarda dormindo o tempo de transmissão correto para seu vizinho, focado na economia de preâmbulos. Apesar

desses nós adicionarem isso no protocolo proposto, a velocidade de transmissão dentro do *backbone* é maior criando essa diferença clara entre as latências por transmissão. Diferença ainda mais observada pelo fato que em uma transmissão com um número n de saltos, desses n em geral apenas $\frac{n}{3}$ ocorrem fora do *backbone*.

Nesse cenário, a maior distância possível se da para nós que estão a ≈707 metros e a menor ≈350 metros. Assim as transmissões através de geo-roteamento gastaram um número de saltos entre 22 e 9. O roteamento do protocolo proposto em geral gasta mais saltos que o WiseMAC com geo-roteamento. Mesmo assim, a latência é diminuída pelo encadeamento do *backbone*.

4.3 Cenário dois

No cenário número 2 foram escolhidos nós da rede de dentro dos quadrantes utilizados no cenário anterior aleatoriamente. Foi adicionada garantia de repetição, ou seja, ocorria repetição de envio de mensagens entre os pares de nós escolhidos. Essa repetição acontecia com o padrão demonstrado no Quadro 4.2. As tuplas formadas por (a,b,2), indicam o primeiro elemento como emissor (sorteado na área de emissores), o segundo como receptor (sorteado na área de receptores) e o terceiro como a quantidade de mensagens seguidas que foram enviadas. Assim, para o caso com 10 pacotes, entre o emissor e receptor a e b foram enviadas 2 mensagens seguidas, c e d 2 mensagens seguidas, e e f 3 mensagens seguidas. Após isso 1 mensagem é enviada entre cada um dos pares (repetindo os pares $a \rightarrow b$, $c \rightarrow d$ e $e \rightarrow f$). Os cenários com 20 e 30 pacotes mantém esse padrão, variando somente emissores e receptores.

A repetição de nós foi necessária pois o WiseMAC somente tem o consumo energético reduzido quando repetições entre transmissões acontecem, ou ao menos alguns caminhos sejam reutilizados para que os preâmbulo sejam reduzidos, demonstrando sua eficiência energética. O comportamento do X-MAC não é alterado por esses tipos de cenário nem quando a escolha de nós é totalmente aleatória.

Para os três protocolos foram utilizadas transmissões de 10, 20 e 30 pacotes, com repetição de dupla emissor/receptor.

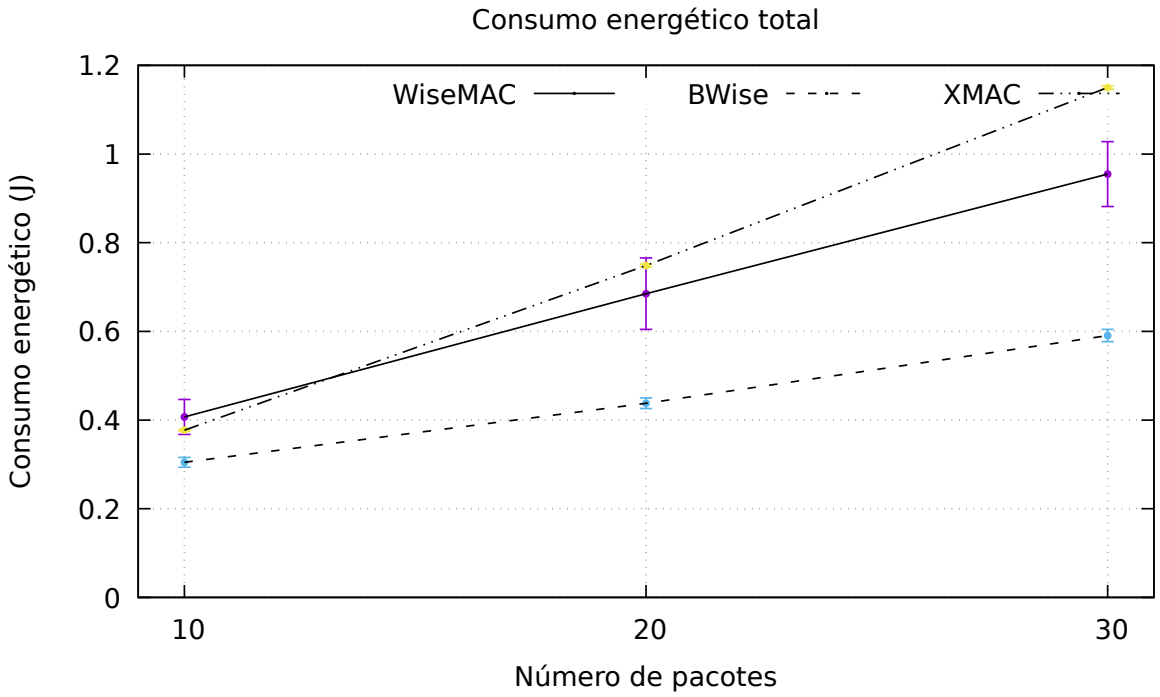
Quadro 4.2 – Padrão de transmissão no cenário dois

Pacotes	Padrão
10	(a,b,2),(c,d,2),(e,f,3),(a,b,1),(c,d,1),(e,f,1)
20	(a,b,2),(c,d,2),(e,f,3),(a,b,1),(c,d,1), (e,f,1) (g,h,2),(i,j,2),(k,l,3),(g,h,1),(i,j,1),(k,l,1)
30	(a,b,2),(c,d,2),(e,f,3),(a,b,1),(c,d,1), (e,f,1) (g,h,2),(i,j,2),(k,l,3),(g,h,1),(i,j,1),(k,l,1) (m,n,2),(o,p,2),(q,r,3),(m,n,1),(o,p,1),(q,r,1)

4.3.1 Consumo energético total

O consumo energético total dos três protocolos pode ser visto na Figura 4.5. O consumo do WiseMAC e do protocolo proposto se mantém semelhantes ao Cenário um. O X-MAC tem uma diferença entre os protocolos, pois ele inicia com poucos preâmbulos em relação ao WiseMAC, assim tendo um consumo energético inicial menor que o WiseMAC, mas maior que o BWise. Isso acontece pois mesmo a diminuição de preâmbulos utilizados pelo X-MAC através da escuta do CTS, esse número em média é a metade da quantidade de preâmbulos da sequência total.

Figura 4.5 – Consumo energético total para o WiseMAC, XMAC e protocolo proposto em Joules



Fonte: Elaborada pelo autor

O X-MAC interrompe o envio da sequência de RTS com recepção de um CTS. No entanto ele não busca se sincronizar. Toda vez que ele realiza uma transmissão, mesmo que para um nó conhecido, ele sempre reinicia sua sequência de preâmbulos até receber um CTS, fazendo com que seu consumo energético não sofra mudanças.

O WiseMAC não interrompe sua sequência de preâmbulos, no entanto quando houver informações sobre vizinhos que já realizaram um contato com o nó, o protocolo utiliza o mínimo de preâmbulos possível. O BWise mantém o comportamento do Cenário um.

4.3.2 Consumo energético médio

O consumo médio está na Figura 4.6. Nela é possível verificar novamente o comportamento anterior do Cenário 1, para os protocolos WiseMAC e o proposto. Suas tendências sem mantém.

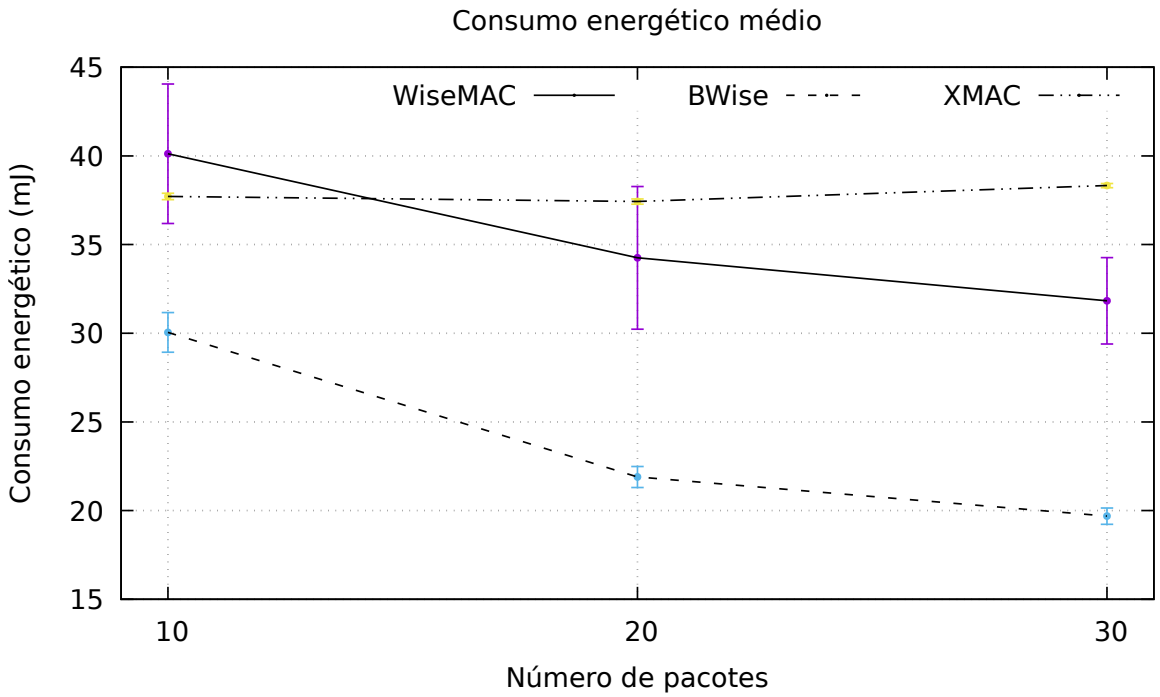
O X-MAC, componente adicional nesse gráfico, tem um comportamento praticamente linear com uma pequena variação dentro do intervalo de confiança. Esse comportamento acontece pois o X-MAC não reduz a quantidade de preâmbulos após contato prévio com algum vizinho ou reutilização de um caminho prévio. Ele somente reduz a emissão de preâmbulos interrompida pela recepção de um CTS.

O BWise inicia com um consumo energético médio de $\approx 30mW$ contra $\approx 38mW$ do X-MAC e essa diferença de $\approx 8mW$ cresce conforme o número de pacotes na rede aumenta.

Em relação ao WiseMAC esse comportamento também acontece, pois conforme já descrito o WiseMAC só utiliza transmissões de preâmbulos longos quando não existe conhecimento da rede. Nesse caso, como existe a certeza de repetição de nós, o WiseMAC se beneficia disso e tem a diminuição da quantidade de preâmbulos e consequentemente um consumo energético por transmissão menor.

Também nessa figura é possível perceber que como existe repetição garantida, o WiseMAC se beneficia dos caminhos conhecidos, seu consumo melhora conforma o aumento de pacotes e tende a se aproximar do BWise.

Figura 4.6 – Consumo energético médio para o WiseMAC, X-MAC e protocolo proposto em millJoules



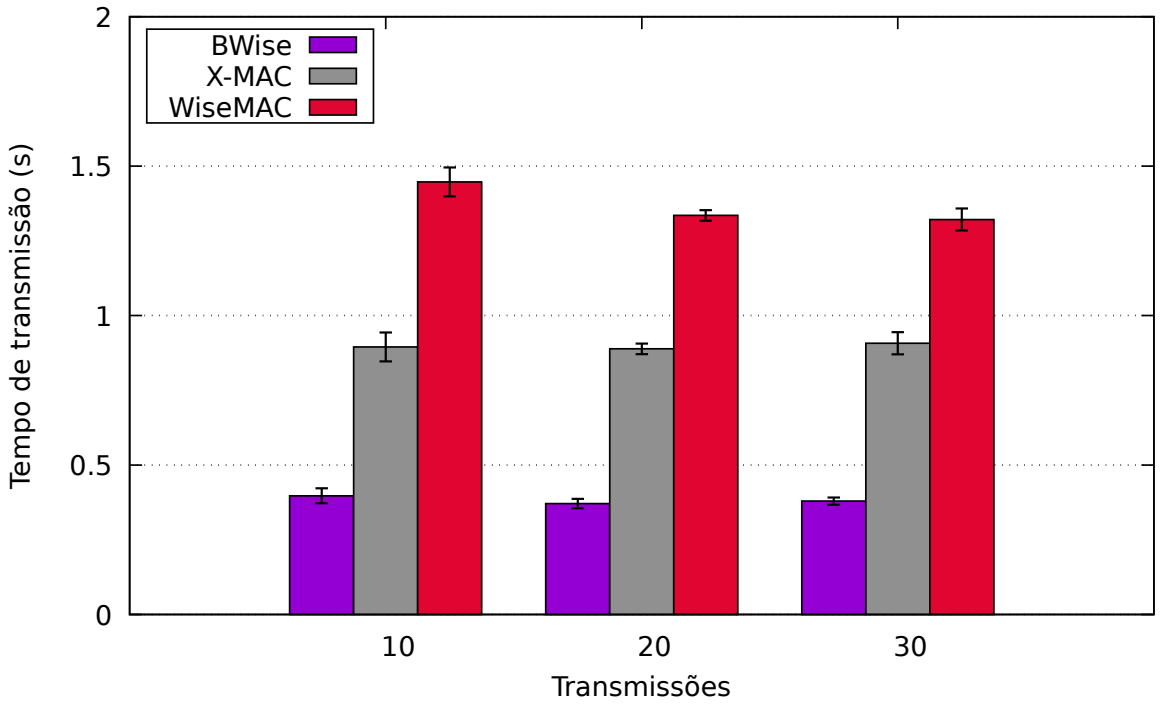
4.3.3 Latência

Para o Cenário dois com repetição garantida, é possível observar na Figura 4.7 que o BWise tem a menor latência, devido ao uso do *backbone* com um trafego de mensagens veloz e apesar de existirem transmissões externas, o impacto delas é compensado pela velocidade proporcionada pelo *backbone*.

O X-MAC gasta em média meio ciclo, devido à interrupção que acontece na sequência de preâmbulos quando o emissor recebe um CTS. Mesmo que os nós interrompam durante momentos diferentes do ciclo, no final das transmissões a média será normalmente próxima a meio ciclo.

O WiseMAC tem o pior comportamento na latência, pois ele pode usar meio ciclo para transmitir dados entre os nós. No entanto diferente do X-MAC, isso acontece caso exista conhecimento do caminho, caso contrário, ele utilizará preâmbulos longos e não é possível ter somente meio ciclo como latência entre os nós na transmissão. A diferença entre os valores do WiseMAC consiste em uma variação estatística.

Figura 4.7 – Latência média para o WiseMAC, X-MAC e BWise em segundos

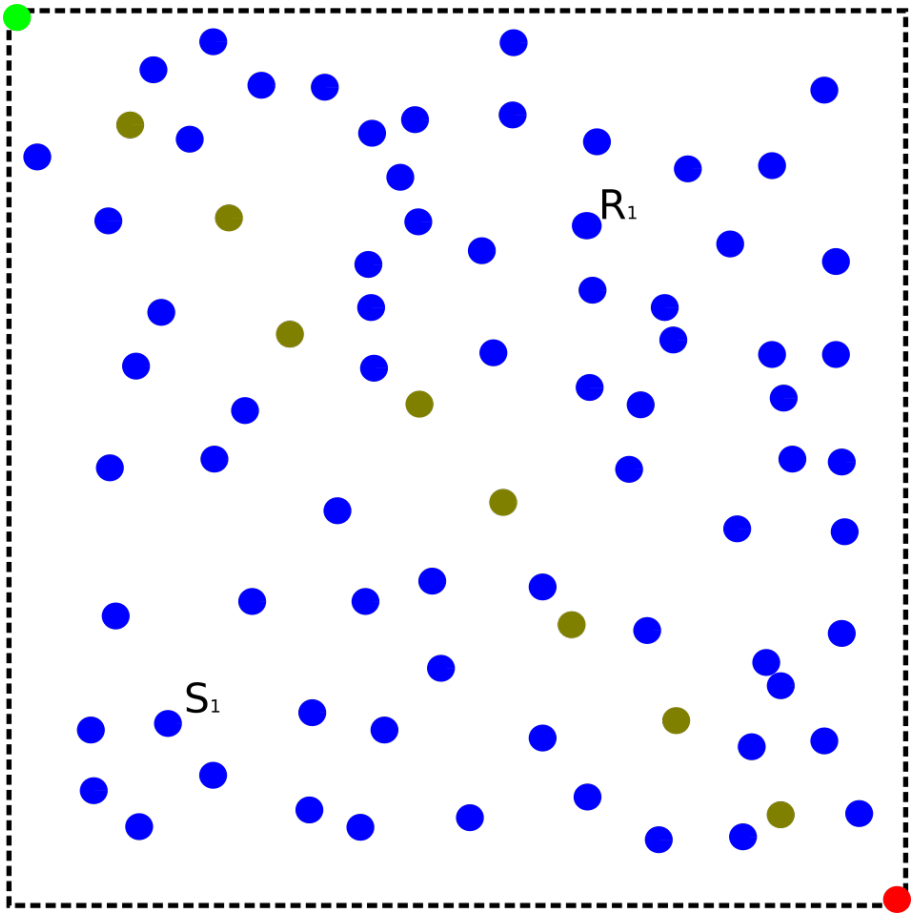


4.4 Cenário três

No cenário número 3, foram escolhidos nós aleatórios em qualquer parte da rede, não existindo distancia mínima para escolha. Os nós que fossem escolhidos realizavam a emissão e recepção de todos os pacotes. Essa repetição permite que o desempenho do WiseMAC seja o ideal para o protocolo, que é quando o reaproveitamento de caminhos é acontece. Assim uma dupla de nós realizava 10, 20 ou 30 transmissões. Foram realizadas 30 simulações para os três números de transmissões.

A Figura 4.8 é um esboço da topologia da rede. O nó verde é o emissor da mensagem de sincronização, o vermelho o receptor. Os nós marrom são os nós tipo 2 dentro do *backbone* e os nós azuis representam os nós tipo 1. Os nós verde, vermelho e marrom estão sempre nas mesmas posições, os nós azuis variam para as topologias geradas para as simulações. Nesse cenário qualquer nó dentro da área tracejada poderia ser sorteado para ser o emissor e outro para ser o receptor. A letra *S* representa um exemplo de emissor e *R* um exemplo de receptor.

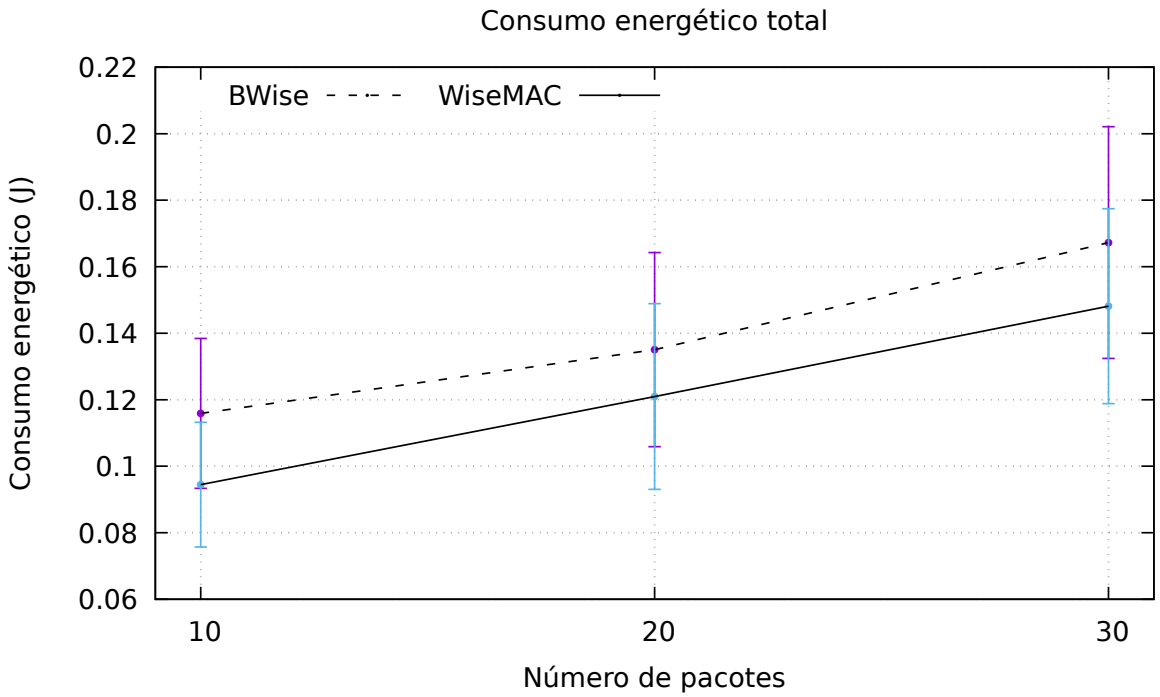
Figura 4.8 – Topologia da rede utilizada para o Cenário três



4.4.1 Consumo energético total

O consumo energético entre o WiseMAC e o BWise pode ser visto na Figura 4.9. Nela é possível verificar que o protocolo WiseMAC, tem consumo energético menor que o BWise. Esse comportamento é esperado pois o BWise quando não utiliza o *backbone* tem comportamento semelhante ao do WiseMAC. Além disso, a diferença em relação ao WiseMAC se deve a saltos em que ele seguiu pelo *backbone* e utilizou um número maior de saltos do que o WiseMAC.

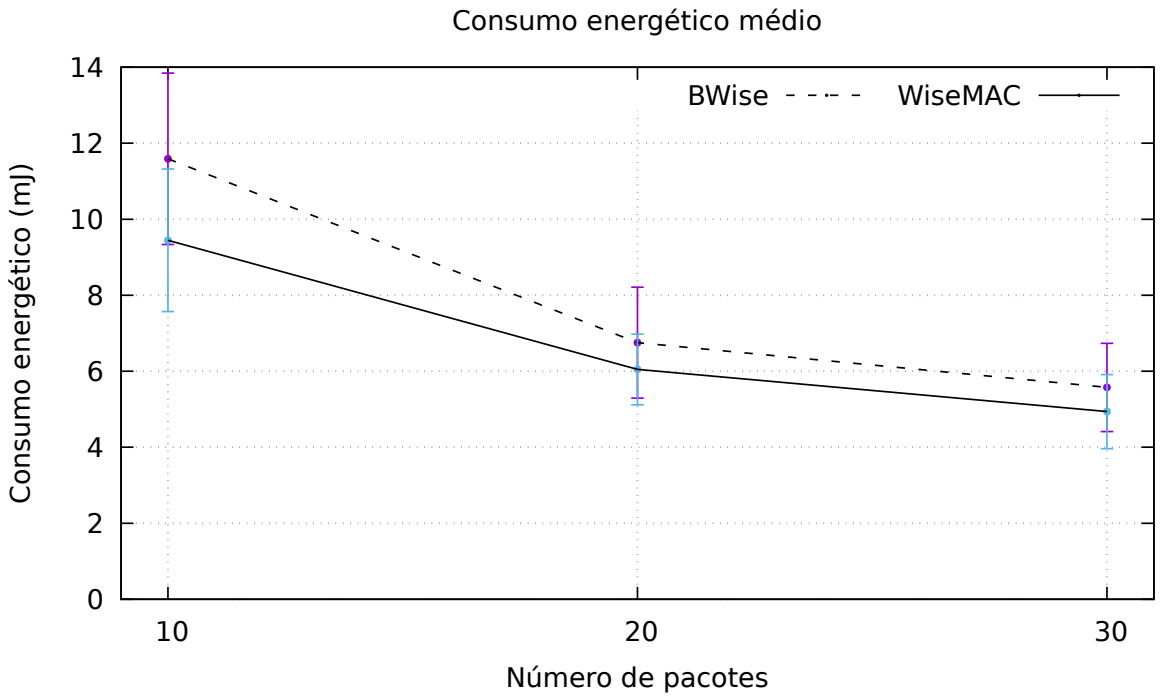
Figura 4.9 – Consumo energético total para o WiseMAC e o BWise



4.4.2 Consumo energético médio

O comportamento das curvas na Figura 4.10 dos protocolos para esse cenário é semelhante aos anteriores, como a diferença que agora quem é melhor é o WiseMAC. Nela é possível verificar a inversão entre os dois protocolos. Nesse cenário são permitidos qualquer distância entre os nós que formam a tupla emissor/receptor. Isso faz com que o *backbone* aumente os saltos, dependendo da localização dos dois nós.

Figura 4.10 – Consumo energético médio para o WiseMAC e o BWise

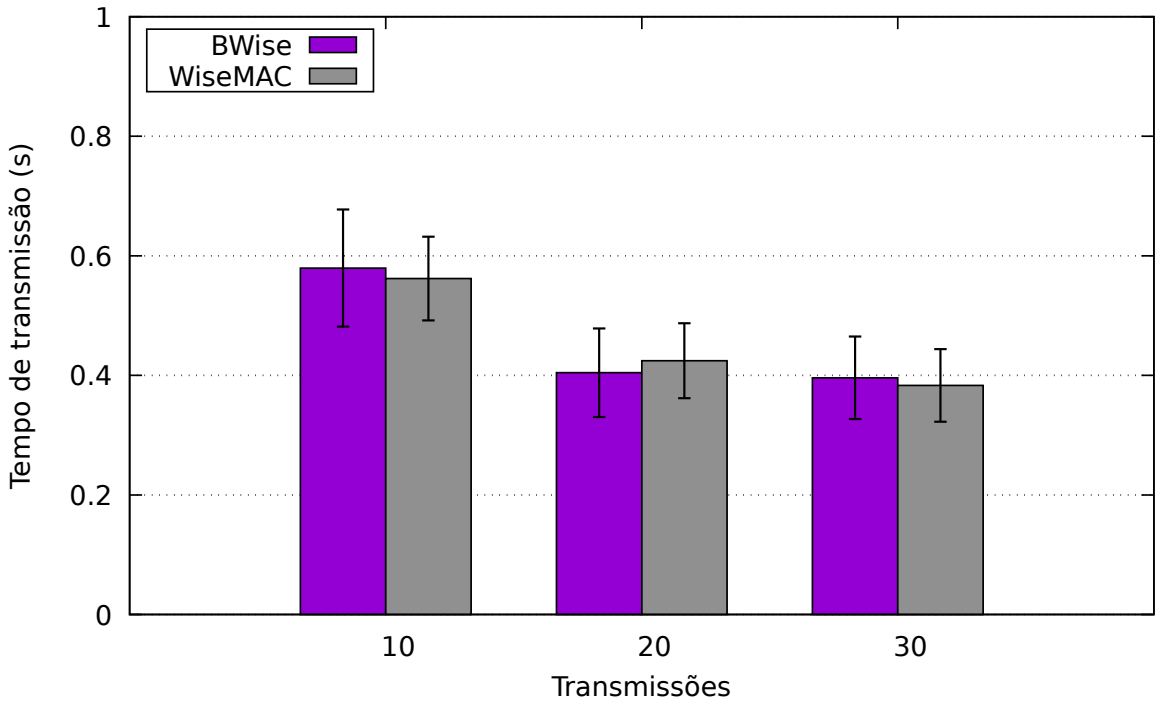


Esse comportamento se deve novamente ao uso do *backbone* do BWise. Quando os nós emissores e receptores são escolhidos pelo fato deles estarem em qualquer local da rede o *backbone* está aumentando o número de saltos para essa transmissão, aumentando o custo energético para ela.

4.4.3 Latência

Para a latência, conforme os cenários anteriores supõe-se que o BWise fosse obter uma latência menor. No entanto os dados mostraram o oposto disso, tanto o BWise quanto o WiseMAC tiveram uma latência próxima, conforme é possível verificar na Figura 4.11. Isso se relaciona novamente ao *backbone* estar utilizando saltos maiores e também com o fato da escolha do *backbone* ter uma possibilidade menor de ocorrer, isso de acordo com a posição dos nós da dupla emissor/receptor.

Figura 4.11 – Latência média para o WiseMAC e o BWise



O motivo para isso é que o *backbone* é limitado. O cenário é totalmente aleatório na questão de escolha da tupla emissor/receptor. A limitação do *backbone* faz com que seu uso acabe sendo diminuído, de forma que mais tuplas que não façam uso do *backbone* sejam sorteadas.

4.4.4 Problema do cenário

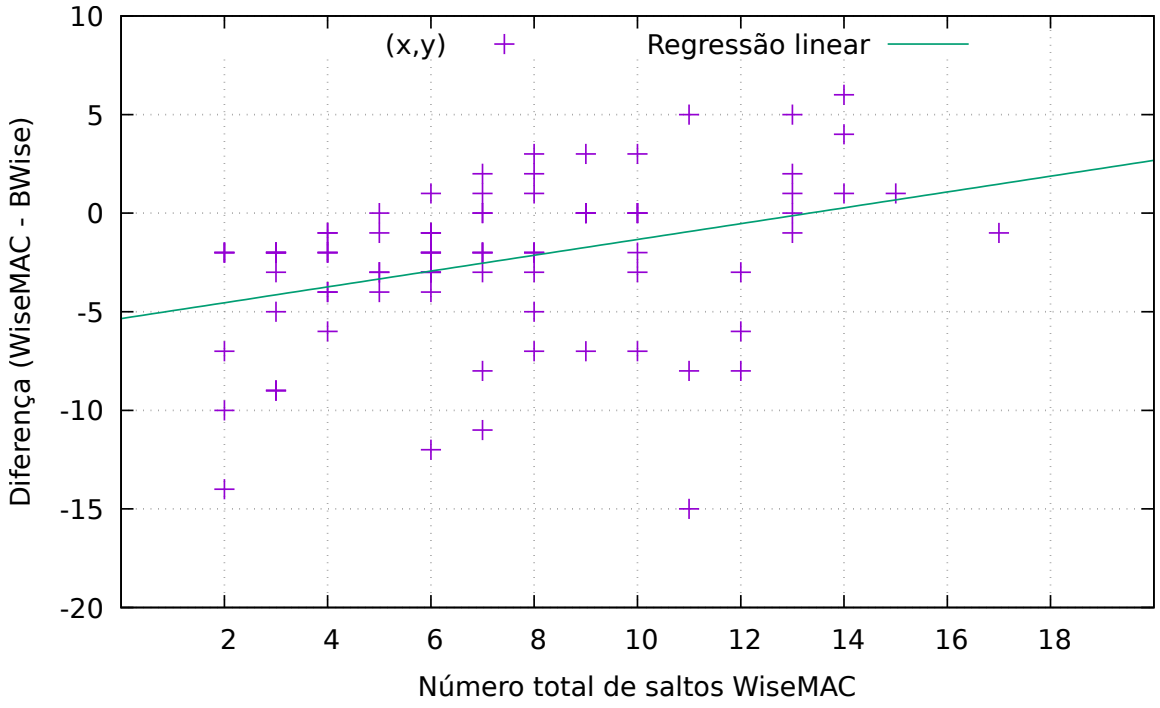
Como descrito, o *backbone* é estático, direcional e linear, por isso para esse cenário onde existe uma aleatoriedade de escolha por toda rede, a chance de escolha de nós que façam utilização adequada do *backbone* é reduzida.

A utilização dele se relaciona também com a quantidade de saltos necessárias para sair do emissor e atingir o receptor.

A Figura 4.12 apresenta uma comparação de latência entre os dois protocolos testados. Nesse caso a latência é medida em número de saltos fora do *backbone*, desde que os saltos dentro do *backbone* tem latência desprezível. Nesse gráfico é apresentada a diferença de saltos utilizando o WiseMAC em relação ao resultado obtido com o BWise. Estes resultados foram obtidos para o número total de saltos variando entre 2 e 17. A linha contínua representa a regressão linear dos dados. Verifica-se que o número de saltos com o WiseMAC se torna maior que o número de saltos obtidos com o BWise à medida que aumenta a distância entre emissor

e receptor da mensagem (representada pelo número total de saltos). De acordo com a regressão linear o uso do *backbone* passa a ser vantajoso a partir de uma distância de 13 saltos entre emissor e receptor.

Figura 4.12 – Diferença no número de saltos para o WiseMAC e o BWise



4.5 Conclusão dos resultados

Os cenários apresentados, comprovam o funcionamento do BWise e sua vantagem em relação aos protocolos comparados. Nos cenários 1 e 2, a vantagem do BWise é comprovada pois ele reduz a latência e consumo energético das transmissões em relações ao o X-MAC e WiseMAC. Isso acontece mesmo com o *backbone* estático e direcional. Esses dois cenários eram limitados em regiões da rede, isso para garantir uso do *backbone*. Essa limitação não interferiu no comportamento dos outros protocolos, pois as distâncias entre emissor e receptor eram semelhantes e o número de saltos também, contando com pequenas variações. Mas em número suficiente para melhora quando utilizado o *backbone*.

Para o cenário 3, o comportamento do BWise é semelhante ao do WiseMAC. A causa disso é à natureza do cenário aliada com a limitação do *backbone*. Essa natureza totalmente aleatória diminui a possibilidade de uso do *backbone* e quando utilizada pode gerar uma desvantagem que não existiria sem ele, devido à sua localização que pode adicionar custos à transmissão. Conforme a Figura 4.12, é possível verificar que o uso do *backbone*, mesmo para

essa escolha de nós totalmente aleatória consegue ser vantajoso quando a distância entre os nós emissor/receptor passa de 13 saltos. Para saltos maiores ele foi na maioria dos testes melhor.

Ainda assim é necessário que o *backbone* tenha capacidade de cobrir uma parte suficiente da rede para obtenção da vantagem de seu uso em qualquer tipo de cenário e uma quantidade de saltos menores, pois a ideia do protocolo é um funcionamento para qualquer transmissão na rede, seja distante ou próxima.

Esse cenário não inválida o protocolo, somente reflete a necessidade de melhora na criação do *backbone* e em relação à sua abrangência. Por se tratar de uma rede densa é possível criar um *backbone* ou vários outros conectando partes diferentes da rede, suprindo esses problemas que existem com o *backbone* utilizado neste trabalho.

5 CONSIDERAÇÕES FINAIS

5.1 Conclusões

No protocolo implementado nesse trabalho, foi possível reduzir a latência da rede significativamente com a utilização do sincronismo entre os nós dentro da estruturação da rede sem precisar manter uma sincronização completa externa ao *backbone*. A sincronização externa local utilizada igualmente ao WiseMAC contribui para a redução do consumo energético. Esse sincronismo foi obtido sem custos extras, pois o pacote responsável por manter os dados de sincronização é o pacote de confirmação (ACK).

Para determinar o sucesso do BWise, foram realizados testes de comparação com o protocolo WiseMAC e o protocolo X-MAC.

O consumo energético médio por transmissões do BWise é semelhante ao do WiseMAC quando todos os nós do caminho percorrido pela mensagem utilizam somente preâmbulos curtos. Em relação à latência, o BWise superou o WiseMAC nos cenários 1 e 2, devido à rápida transmissão proporcionada pelo *backbone*. No cenário 3 o BWise teve um comportamento semelhante ao WiseMAC, pois a possibilidade de uso do *backbone* foi reduzida com a seleção totalmente aleatória de pares de transmissão em qualquer posição da rede. Esse cenário mostra a necessidade de uma ampliação e melhora na abrangência do *backbone*. Com essa abrangência, espera-se obter um comportamento semelhante aos cenários 1 e 2.

Os resultados comprovam que o BWise fez com que essa estruturação da rede funcionasse adequadamente para reduzir a latência e o consumo médio de energia por transmissão nos 2 cenários em que o *backbone* era utilizado corretamente.

Embora isso tenha acontecido, é preciso ressaltar que o consumo total do BWise nos nós utilizados no *backbone* tende a crescer, pois os fluxos são direcionados por ele. Isso pode levar a uma falha no *backbone*, pois os nós podem sofrer esgotamento energético.

Outra consideração é que o *backbone* utilizado para validar a premissa continha um número fixo de nós, sempre posicionados no mesmo local, o que fez com que existisse uma limitação nas possibilidades de entrega de mensagens pelo *backbone*. Outros formatos de *backbone* podem ser definidos e pesquisados para que os nós abrangessem a totalidade da rede, bem como outras estratégias de eleição de eleição *backbone* para evitar esse esgotamento pela constância dos fluxos.

Em relação ao modelo de *clock-drift* implementado, todos os dados estão considerando ele. No entanto, devido ao fluxo contínuo de dados na rede, a variação do *clock-drift* é pequena para causar problemas visíveis, necessitando de muito tempo entre transmissões para que ele consiga interferir a ponto de causa problemas para a transmissão de dados.

5.2 TRABALHOS FUTUROS

Como trabalho futuro existe a necessidade de uma melhora na forma como o *backbone* é escolhido. Assim uma heurística poderia ser pesquisada para determinar um conjunto de nós apropriados que farão parte do *backbone*, de forma distribuída.

Para essa geração de *backbone* é importante considerar também meios de alternância dos componentes do *backbone*. Um *backbone* fixo poderá ser um ponto de falha, pois os nós poderão ter sua energia esgotada mais rapidamente, então uma forma de evitar esse problema ou reduzi-lo é importante.

Uma forma de dispersão dinâmica da informação de roteamento seria também importante para informar para os nós o posicionamento dos nós do *backbone*. Isso é necessário para o *backbone* poder ser gerado dinamicamente, após implantação da rede e também permitir a alternância de nós para evitar problemas de esgotamento energético, caso essa estratégia para o problema energético em potencial seja adotada.

Além disso, é importante melhorar a simulação do sistema de *clock-drift* para funcionar mais próximo do real, levando em conta questões relacionadas a temperatura, umidade entre outras características que interferem no seu comportamento.

É importante a implementação de um modelo de colisões, a adaptação do protocolo para funcionamento com colisões e formas de recuperação de colisões. O protocolo deve apresentar uma robustez em relação a esse aspecto, no que tange a sincronização do *backbone*.

REFERÊNCIAS

AKYILDIZ, I. F. et al. Wireless sensor networks: A survey. **Comput. Netw.**, Elsevier North-Holland, Inc., New York, NY, USA, v. 38, n. 4, p. 393–422, mar. 2002. ISSN 1389-1286. Disponível em: <http://dx.doi.org/10.1016/S1389-1286(01)00302-4>.

AKYILDIZ, I. F.; VURAN, M. C.; AKAN, O. B. A cross-layer protocol for wireless sensor networks. In: **2006 40th Annual Conference on Information Sciences and Systems**. [S.l.: s.n.], 2006. p. 1102–1107.

AL-KARAKI, J. N.; KAMAL, A. E. Routing techniques in wireless sensor networks: a survey. **IEEE Wireless Communications**, v. 11, n. 6, p. 6–28, Dec 2004. ISSN 1536-1284.

AVVENUTI, M. et al. Energy-efficient reception of large preambles in mac protocols for wireless sensor networks. **Electronics Letters**, v. 43, n. 5, p. 59–60, March 2007. ISSN 0013-5194.

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

BRZOZOWSKI, M.; SALOMON, H.; LANGENDOERFER, P. On efficient clock drift prediction means and their applicability to ieee 802.15.4. In: **2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing**. [S.l.: s.n.], 2010. p. 216–223.

BUETTNER, M. et al. X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks. In: **Proceedings of the 4th International Conference on Embedded Networked Sensor Systems**. New York, NY, USA: ACM, 2006. (SenSys '06), p. 307–320. ISBN 1-59593-343-3. Disponível em: <http://doi.acm.org/10.1145/1182807.1182838>.

CANO, C. et al. Low energy operation in wsns: A survey of preamble sampling mac protocols. **Comput. Netw.**, Elsevier North-Holland, Inc., New York, NY, USA, v. 55, n. 15, p. 3351–3363, out. 2011. ISSN 1389-1286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2011.06.022>.

DAM, T. van; LANGENDOEN, K. An adaptive energy-efficient mac protocol for wireless sensor networks. In: **Proceedings of the 1st International Conference on Embedded Networked Sensor Systems**. New York, NY, USA: ACM, 2003. (SenSys '03), p. 171–180. ISBN 1-58113-707-9. Disponível em: <http://doi.acm.org/10.1145/958491.958512>.

DOUDOU, M.; DJENOURI, D.; BADACHE, N. Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks. **IEEE Communications Surveys Tutorials**, v. 15, n. 2, p. 528–550, Second 2013. ISSN 1553-877X.

EL-HOIYDI, A.; DECOTIGNIE, J.-D. Wisemac: An ultra low power mac protocol for multi-hop wireless sensor networks. In: ____. **Algorithmic Aspects of Wireless Sensor Networks: First International Workshop, ALGOSENSORS 2004, Turku, Finland, July 16, 2004. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 18–31. ISBN 978-3-540-27820-7. Disponível em: <http://dx.doi.org/10.1007/978-3-540-27820-7_4>.

EL-HOIYDI, A. et al. Wisemac, an ultra low power mac protocol for the wisenet wireless sensor network. In: ACM. **Proceedings of the 1st international conference on Embedded networked sensor systems**. [S.l.], 2003. p. 302–303.

FERENTINOS, K. P.; TSILIGIRIDIS, T. A. Adaptive design optimization of wireless sensor networks using genetic algorithms. **Computer Networks**, v. 51, n. 4, p. 1031 – 1051, 2007. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128606001678>.

HEIMFARTH, T. et al. Gb-mac: A backbone based low latency protocol for wsns. In: **Proceedings of the 29th Annual ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2014. (SAC '14), p. 692–698. ISBN 978-1-4503-2469-4. Disponível em: <http://doi.acm.org/10.1145/2554850.2555031>.

HEIMFARTH, T.; GIACOMIN, J. C.; ARAUJO, J. P. d. Aga-mac: Adaptive geographic anycast mac protocol for wireless sensor networks. In: **2015 IEEE 29th International Conference on Advanced Information Networking and Applications**. [S.l.: s.n.], 2015. p. 373–381. ISSN 1550-445X.

KOCAKULAK, M.; BUTUN, I. An overview of wireless sensor networks towards internet of things. In: **2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)**. [S.l.: s.n.], 2017. p. 1–6.

LANGENDOEN, K.; MEIER, A. Analyzing mac protocols for low data-rate applications. **ACM Trans. Sen. Netw.**, ACM, New York, NY, USA, v. 7, n. 2, p. 19:1–19:40, set. 2010. ISSN 1550-4859. Disponível em: <http://doi.acm.org/10.1145/1824766.1824775>.

LESSMANN, J.; HEIMFARTH, T.; JANACIK, P. Shox: An easy to use simulation platform for wireless networks. In: **Tenth International Conference on Computer Modeling and Simulation (uksim 2008)**. [S.l.: s.n.], 2008. p. 410–415.

LI, Y.-X.; SHI, H.-S.; ZHANG, S.-P. An energy-efficient mac protocol for wireless sensor network. In: **2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)**. [S.l.: s.n.], 2010. v. 4, p. V4–619–V4–623. ISSN 2154-7491.

LU, G.; KRISHNAMACHARI, B.; RAGHAVENDRA, C. S. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In: **18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.** [S.l.: s.n.], 2004. p. 224–.

MACEDO, D. F. et al. Proc: Um protocolo pro-ativo com coordenação de rotas em redes de sensores sem fio. **22 Simpósio Brasileiro de Redes de Computadores**, p. 571–574, 2004.

MATEEN, A. et al. Comparative analysis of wireless sensor networks with wireless multimedia sensor networks. In: **2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)**. [S.l.: s.n.], 2017. p. 80–83.

MENDES, L. D.; RODRIGUES, J. J. A survey on cross-layer solutions for wireless sensor networks. **Journal of Network and Computer Applications**, v. 34, n. 2, p. 523 – 534, 2011. ISSN 1084-8045. Efficient and Robust Security and Services of Wireless Mesh Networks. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804510002079>.

N, S. S.; CHOU, C.-T.; GHOSH, M. Cooperative communication mac (cmac) - a new mac protocol for next generation wireless lans. In: **2005 International Conference on Wireless Networks, Communications and Mobile Computing**. [S.l.: s.n.], 2005. v. 1, p. 1–6 vol.1.

POLASTRE, J.; HILL, J.; CULLER, D. Versatile low power media access 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. 95–107. ISBN 1-58113-879-2. Disponível em: <http://doi.acm.org/10.1145/1031495.1031508>.

RAULT, T.; BOUABDALLAH, A.; CHALLAL, Y. Energy efficiency in wireless sensor networks: A top-down survey. **Computer Networks**, v. 67, p. 104 – 122, 2014. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128614001418>.

RAWAT, P. et al. Wireless sensor networks: a survey on recent developments and potential synergies. **The Journal of supercomputing**, Springer, v. 68, n. 1, p. 1–48, 2014.

ROUSSELOT, J.; DECOTIGNIE, J.-D. When ultra low power meets high performance: The wisemac high availability protocol. In: **Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems**. New York, NY, USA: ACM, 2010. (SenSys '10), p. 441–442. ISBN 978-1-4503-0344-6. Disponível em: <http://doi.acm.org/10.1145/1869983.1870064>.

SINGH, H.; BISWAS, B. Comparison of csma based mac protocols of wireless sensor networks. **Int J Ad Hoc Netw Syst**, v. 2, n. 2, p. 11–20, 2012.

TOKLU, S.; ERDEM, O. A. Bsc-mac: Energy efficiency in wireless sensor networks with base station control. **Computer Networks**, v. 59, p. 91 – 100, 2014. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128613004155>.

VIEIRA, M. A. M. et al. Survey on wireless sensor network devices. In: **Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference**. [S.l.: s.n.], 2003. v. 1, p. 537–544 vol.1.

VURAN, M. C.; AKYILDIZ, I. F. Xlp: A cross-layer protocol for efficient communication in wireless sensor networks. **IEEE Transactions on Mobile Computing**, v. 9, n. 11, p. 1578–1591, Nov 2010. ISSN 1536-1233.

YE, W.; HEIDEMANN, J.; ESTRIN, D. An energy-efficient mac protocol for wireless sensor networks. In: **Proceedings Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies**. [S.l.: s.n.], 2002. v. 3, p. 1567–1576 vol.3. ISSN 0743-166X.