GILSON MIRANDA JÚNIOR

DESENVOLVIMENTO DE UMA REDE DE SENSORES SEM FIO PARA MONITORAMENTO INDUSTRIAL

LAVRAS – MG

2014

GILSON MIRANDA JÚNIOR

DESENVOLVIMENTO DE UMA REDE DE SENSORES SEM FIO PARA MONITORAMENTO INDUSTRIAL

Monografia apresentada à Universidade Federal de Lavras, como Trabalho de Conclusão de Curso para obtenção do título de Bacharel em Ciência da Computação.

Orientador
Prof. DSc. Luiz Henrique Andrade Correia

Co-Orientador
Prof. DSc. Pedro Castro Neto

LAVRAS – MG

2014

GILSON MIRANDA JÚNIOR

DESENVOLVIMENTO DE UMA REDE DE SENSORES SEM FIO PARA MONITORAMENTO INDUSTRIAL

Trabalho de Conclusão de Curso de Graduação apresentado ao Colegiado do Curso de Bacharelado em Ciência da Computação, para obtenção do título de Bacharel.

APROVADA em 26 de novembro de 2014.

Dr. João Carlos Giacomin

João Paulo de Araujo

Dr. Luiz Henrique Andrade Correia (Orientador)

Pedro Castro Neto (Coorientador)

LAVRAS-MG
Novembro/2014

Dedico esta monografia aos meus familiares e amigos.

AGRADECIMENTOS

Agradeço primeiramente a minha mãe, Andréia Eugênia de Abreu Miranda, pelo apoio incondicional durante toda a minha vida.

A minha irmã, Flávia que mesmo distante sempre me dedicou muita atenção.

A meu pai, Gilson Miranda, que apesar de todas as dificuldades, colaborou substancialmente para meu desenvolvimento pessoal.

A João Paulo de Araujo, um verdadeiro amigo, sempre presente para uma boa conversa, aprender juntos, trocar conhecimentos, ou simplesmente tomar uma cerveja.

A Jaime Daniel Corrêa Mendes, um grande amigo, que tão bem me recebeu em Lavras e sempre esteve disposto a me ajudar.

Ao meu primo Lucas Guimarães, pela amizade e parceria.

Aos meus orientadores, Pedro Castro Neto, Antônio Carlos Fraga e Luiz Henrique Andrade Correia, pela considerável contribuição para meu desenvolvimento pessoal e profissional.

A Lívia Mayumi Saito e toda sua família, pelo apoio e pela paciência.

Aos demais familiares e amigos que de alguma forma contribuíram para o meu desenvolvimento.

Por fim, agradeço a todos os professores que fizeram parte da minha trajetória, desde a formação básica até o curso superior.

RESUMO

Redes de Sensores Sem Fio (RSSF) têm sido empregadas para diversos tipos de aplicações. Seu uso em ambientes industriais, no entanto, ainda é limitado. O custo de equipamentos voltados a esta aplicação é muitas vezes proibitivo, principalmente para indústrias de pequeno porte. Com o barateamento dos equipamentos de transmissão de dados via rádio, a construção de RSSF de baixo custo para monitoramento industrial começa a se mostrar viável. Estas redes, no entanto, devem ser confiáveis e eficientes, tal que sua utilização não acarrete riscos ou prejuízos ao andamento dos processos. Este trabalho apresenta uma solução baseada no rádio nRF24l01+ e na plataforma de prototipagem Arduino, que emprega um protocolo baseado em divisão de tempo, para a construção de uma RSSF que possa ser aplicada ao monitoramento industrial, além de servir de plataforma para o desenvolvimento de novas pesquisas na área de RSSF.

Palavras-Chave: Redes de Sensores Sem Fio; Automação Industrial; Protocolo MAC.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A demanda por melhorias em processos industriais requer um uso mais amplo de sistemas de monitoramento. Estes sistemas fazem a coleta de dados e fornecem a base para o desenvolvimento de equipamentos e processos mais eficientes. Com a grande quantidade de variáveis envolvidas nestes processos, torna-se conveniente o uso de sistemas de baixo custo que permitam sua ampla adoção. Entretanto, estes sistemas devem ser confiáveis e permitir comunicações em tempo reduzido, tal que as mensagens trafeguem pela rede de forma eficiente. Sistemas cabeados têm sido largamente utilizados na indústria, mas com o avanço dos equipamentos de transmissão via rádio, a redução no seu custo e o desenvolvimento de novos protocolos, as Redes de Sensores Sem Fio (RSSF) têm se tornado uma alternativa interessante a ser aplicada neste segmento.

As RSSF podem não só reduzir o custo de implantação de sistemas de automação industrial, mas também permitir novos tipos de monitoramento. Nós sensores podem ser implantados onde seria inviável o uso de sistemas cabeados como, por exemplo, rolamentos, veículos, caixas de embalagem e em ambientes inacessíveis ou perigosos (Low, Win e Er 2005).

Estas redes estão sujeitas a interferências no meio de transmissão. Especialmente em redes industriais, onde aplicações críticas dependem de alta confiabilidade e baixo tempo de resposta, a escolha das tecnologias de transmissão utilizadas é desafiadora. A interrupção da comunicação ou o atraso na transferência de mensagens entre sistemas de controle e equipamentos pode acarretar em desastres em termos financeiros e até mesmo colocar em risco a segurança das pessoas envolvidas (Low, Win e Er 2005).

O desafio na construção de RSSF industriais é garantir que os dados trafeguem pela rede de forma confiável e com um tempo adequado às aplicações. Além disso, o custo de implantação e manutenção destes sistemas deve ser considerado de forma que permita sua adoção em larga escala.

1.1 Motivação

Grande parte dos processos realizados nas indústrias, principalmente as de pequeno porte, são monitorados e controlados por pessoas. Esse tipo de procedimento tem menor precisão e é mais suscetível a falhas. O emprego de sistemas automatizados é muitas vezes proibitivo devido aos altos custos de implantação e manutenção. Além disso, o uso de comunicação cabeada é invável em determinadas situações. O desenvolvimento de uma RSSF com nós de baixo custo, mas que tenha operação confiável, se torna interessante para suprir essas necessidades.

Na academia também existe a demanda por sistemas de monitoramento sem fios e de baixo custo. Nós da RSSF podem ser posicionados dentro de reatores coletando dados de processos enquanto estes estão em operação. Experimentos onde antes se coletava dados com dias de intervalo, podem passar a ser monitorados a cada minuto. Com isso, pode-se obter uma maior rastreabilidade dos processos e permitir o desenvolvimento de novas tecnologias e pesquisas.

Ainda na academia, a falta de plataformas de baixo custo para o estudo de protocolos e técnicas de RSSF demanda a criação de nós acessíveis e de fácil construção, que possam ser adotados por professores e estudantes interessados em fazer pesquisas nesta área.

1.2 Definição do Problema

Segundo Gungor e Hancke (2009), tradicionalmente os sistemas de monitoramento industrial são baseados em soluções cabeadas, que possuem alto custo de instalação e manutenção, e ainda, podem ser inviáveis para o monitoramento de pontos longínquos, locais de difícil acesso e objetos móveis. Assim, sistemas sem fio podem servir de complemento ou até mesmo substituir completamente os sistemas cabeados em determinadas aplicações industriais. Este trabalho visa ao desenvolvimento de nós de baixo custo e seus protocolos de comunicação, para a costrução

de uma RSSF que possa ser empregada no monitoramento de processos industriais e experimentos de pesquisa científica.

1.3 Solução Proposta

A solução proposta é a criação de um nó sensor baseado na plataforma de prototipagem Arduino e no rádio nRF24l01+. Além do nó sensor, serão desenvolvidos os protocolos de comunicação da rede utilizando a abordagem TDMA (*Time Division Multiple Access*) de modo a possibilitar a estimativa de um tempo limite de resposta para a transferência dos dados coletados até o servidor. Os nós da rede fazem a aquisição de dados e então os transmitem para os nós apropriados, que por sua vez encaminham estes pacotes pela rede até chegar ao servidor. O servidor tem a tarefa de agregar, processar e exibir os dados recebidos.

1.4 Objetivos

Esta seção define o objetivo geral do trabalho e o distribui em um conjunto de objetivos específicos a serem seguidos para a elaboração da solução proposta, a obtenção e avaliação dos resultados.

1.4.1 Objetivo Geral

Construir um módulo de comunicação sem fio, que será utilizado para o monitoramento de processos industriais e de pesquisa.

1.4.2 Objetivos Específicos

Com o propósito de atingir o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

1. Avaliação de sistemas de comunicação sem fio disponíveis.

2. Projeto de *hardware* e construção de nós de uma RSSF.

3. Avaliação de protocolos de Controle de Acesso ao Meio (*Medium Access Control*, MAC).

4. Implementação de um protocolo MAC adequado às necessidades de uma RSSF industrial.

5. Realização de testes com os nós construídos.

6. Avaliação dos resultados dos testes e conclusão.

1.5 Organização do Trabalho

Este trabalho está organizado da seguinte forma: no capítulo 2, são apresentados conceitos e trabalhos que serviram de base para o desenvolvimento da RSSF proposta. No capítulo 3, o trabalho desenvolvido é apresentado. Os equipamentos de *hardware* utilizados são descritos, assim como o *software* desenvolvido. O cenário de testes utilizado é apresentado e são definidas as métricas utilizadas para avaliação do trabalho. O capítulo 4 contém os resultados obtidos a partir dos testes realizados, bem como a avaliação destes resultados. No capítulo 5 são apresentadas as conclusões obtidas com o desenvolvimento da RSSF. Por fim, no capítulo 6 são sugeridos trabalhos futuros a serem desenvolvidos, e uma breve descrição de cada um.

2 REFERENCIAL TEÓRICO

Esta seção apresenta o referencial teórico que enfoca os conceitos utilizados para o desenvolvimento deste trabalho.

2.1 Redes de Sensores Sem Fio

Redes de Sensores Sem Fio (RSSF) são redes formadas por diversos nós, que trabalham de forma colaborativa para executar uma aplicação específica. Os nós desta rede são distribuídos de forma que todos possam se comunicar, direta ou indiretamente, e que os dados coletados pelos nós possam ser transmitidos até um ponto onde são coletados e utilizados de alguma forma.

Os nós desta rede podem estar conectados a diversos tipos de sensores utilizados para a coleta de dados de materiais, equipamentos ou ambientes. Da mesma forma, atuadores podem ser acoplados aos nós permitindo que estes interajam com o ambiente mediante comandos de controle vindos de um servidor ou ações pré-programadas nos nós.

Em Akyildiz et al. (2002) é relacionada uma variedade de condições ambientais que podem ser medidas com RSSF, dentre elas:

- Temperatura.
- Umidade.
- Movimento de veículos.
- Condições de iluminação.
- Pressão.
- Níveis de Ruído.
- Presença ou ausência de determinados objetos.

- Informações como direção, sentido e velocidade de objetos.
- Níveis de estresse mecânico.

Com tal variedade de monitoramentos possíveis aliada a mecanismos de auto-organização e tolerância a falhas, as RSSF podem ser empregadas em aplicações militares, ambientais, industriais, dentre outras.

Na indústria as RSSF têm sido aplicadas no monitoramento de redes elétricas inteligentes (*smart grid*) (Gungor, Lu e Hancke 2010), instalações de testes de propulsão (Solano et al. 2004), em equipamentos da indústria alimentícia e na agricultura (Wang, Zhang e Wang 2006), dentre outros campos. Soluções baseadas no padrão ZigBee têm sido avaliadas em ambientes industriais (Zheng 2006), porém estas soluções têm se mostrado menos eficazes em relação a segurança, robustez e consumo de energia quando comparadas a soluções que empregam o padrão industrial *WirelessHART* (Lennvall, Svensson e Hekland 2008). Em outubro de 2007, a ZigBee Alliance lançou a especificação ZigBee PRO, que recebeu recursos para melhoria da segurança assim como o recurso *"frequency agility"*, com intenção de trazer mais confiabilidade para a transmissão de dados. Esse recurso de *"frequency agility"*, no entanto, não é tão flexível quanto a solução de *"frequency hopping"* que já vem sendo empregada em outros padrões e equipamentos (Kim et al. 2008).

2.2 Topologia

Em redes de computadores os nós podem estar dispostos em diferentes formas, ou topologias. Nesta seção são abordadas as topologias infraestruturada, ad hoc e mesh.

Em redes **infraestruturadas**, apenas os nós das bordas se movem, ou seja, existem pontos de acesso fixos e os nós devem se manter ao alcance destes pontos para obter uma conexão. Somente ao obter essa conexão o nó será capaz de enviar

e receber pacotes dos outros nós (Sun 2001). Na Figura 2.1 é ilustrado este tipo de rede.

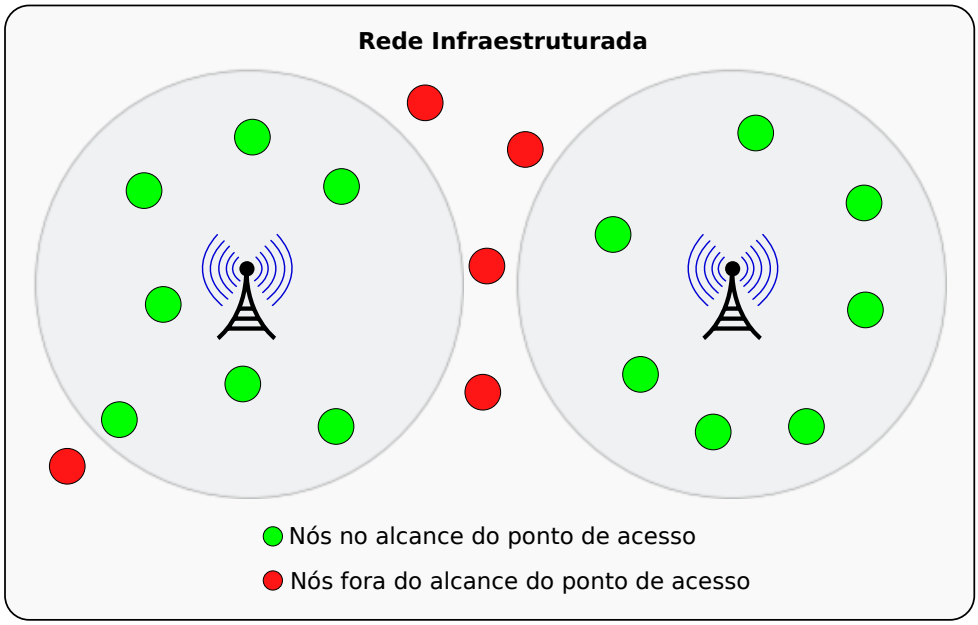

Figura 2.1: Rede infraestruturada.

Redes **ad hoc**, de acordo com Karl e Willig (2005), são redes desenvolvidas para um propósito específico, de modo que os nós de uma destas redes formem rapidamente conexões entre si. Para isso, o aspecto de autoconfiguração é crucial - é esperado que a rede funcione sem a necessidade de configurações ou gerenciamento manual. Baseado nas redes ad hoc, tem-se o conceito de *MANET - Mobile Ad Hoc Networks* (Redes Ad Hoc Móveis), que emprega a comunicação sem fios, tendo como uma das características principais a mobilidade dos nós da rede.

Nestas redes, os nós trabalham de forma colaborativa e fazem encaminhamento e roteamento dos pacotes dos demais nós. Desta forma é possível aumentar o alcance de nós individuais, e permitir a cobertura de maiores áreas do que seria possível com redes infraestruturadas. Um dos desafios básicos desta topologia consiste em reorganizar a rede a medida que os nós se movem, pois as tecnologias de comunicação sem fio possuem alcance limitado, e as conexões e caminhos

estabelecidos entre os nós podem ser alteradas por essa movimentação. A Figura 2.2 ilustra uma rede ad hoc, com os nós conectados sem o uso de pontos de acesso fixo.

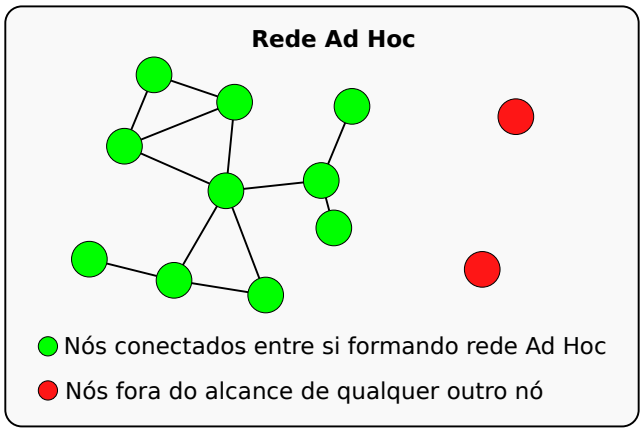

Figura 2.2: Rede ad hoc.

Redes **mesh** são definidas por Bruno, Conti e Gregori (2005) como uma classe de redes ad hoc que, ao invés de serem isoladas e autoconfiguradas, trabalham como extensões flexíveis e baratas para redes infraestruturadas. Redes mesh são composições de nós fixos e móveis, interconectados por redes sem fio que formam redes ad hoc *multi-hop*. A Figura 2.3 ilustra uma rede mesh de acordo com esta definição. Um ponto de acesso fornece conexão para os nós dentro do alcance do seu rádio, estes nós por sua vez fornecem conexão aos demais fora do alcance da infraestrutura. No entanto, nós distantes até mesmo destes nós conectados são incapazes de participar da rede.

A topologia mesh, que combina redes infraestruturadas e redes ad hoc móveis, é particularmente conveniente quando se deseja obter redes tolerantes a falhas de nós ou interferências no meio de transmissão, provendo caminhos alternativos para o roteamento dos pacotes entre os nós e os servidores. Para a implementação desta topologia em RSSF é necessária a escolha dos protocolos de roteamento e controle de acesso ao meio (MAC - *Medium Access Control*) apropriados à aplicação da rede e ao ambiente em que ela será inserida.

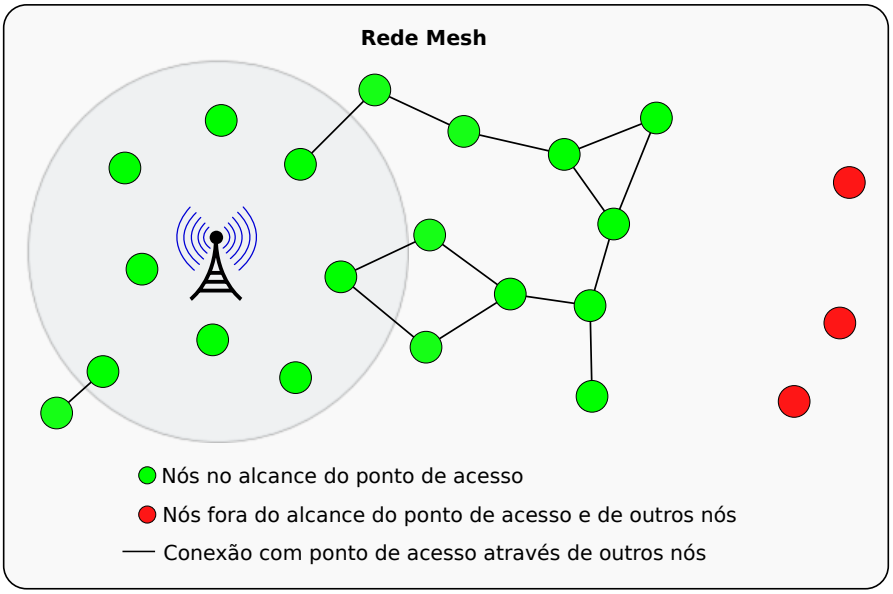

Figura 2.3: Rede mesh.

2.3 Camadas

Visando reduzir a complexidade do projeto, a maioria das redes é organizada em camadas. O nome, o conteúdo e as funções de cada camada diferem de uma rede para outra, no entanto, em todas as redes o objetivo das camadas é prover serviços às camadas superiores e abstraí-las dos detalhes de implementação (Tanenbaum 2002). Na RSSF desenvolvida não existe roteamento de pacotes, apenas encaminhamento de quadros, ou seja, os nós precisam definir apenas para qual de seus vizinhos o quadro deve ser encaminhado. Por esse motivo, nenhum protocolo de camadas acima da camada de Enlace foi utilizado. Nesta seção serão discutidas as camadas Física, Enlace e a subcamada de Controle de Acesso ao Meio (MAC - *Medium Access Control*). A Figura 2.4 ilustra a organização de uma rede com as camadas mencionadas, que proveem serviços a um programa na camada de Aplicação.

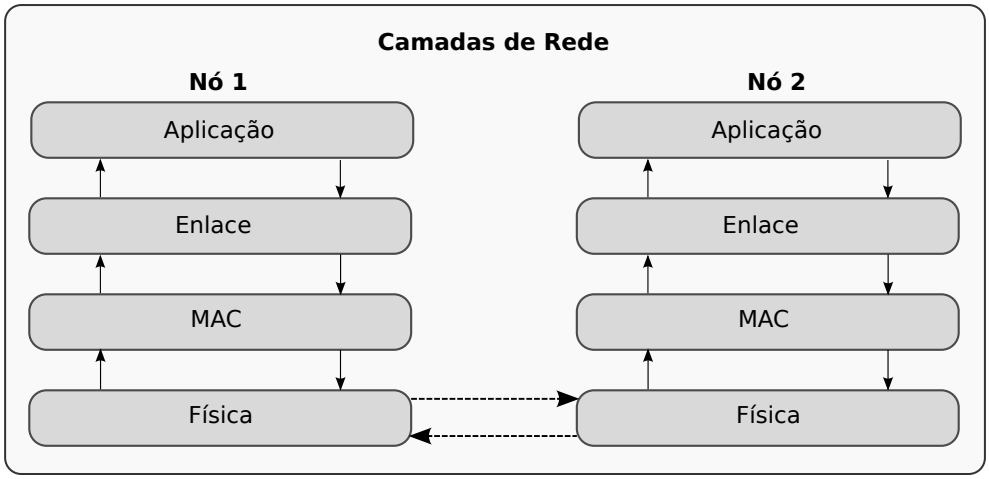

Figura 2.4: Divisão da rede em camadas.

2.3.1 Camada Física

Segundo Karl e Willig (2005), as funções principais da camada física são a modulação e demodulação de dados digitais. Esta tarefa é realizada pelos chamados transceptores. Em RSSF o desafio consiste em encontrar esquemas de modulação e arquiteturas de transceptores que sejam simples, baratas e robustas para os requisitos da rede.

Em RSSF são comumente exploradas três formas de comunicação sem fio: óptica, infravermelho e radiofrequência (RF).

A comunicação óptica consome menos energia por bit transmitido e não requer o uso de antenas para a transmissão, no entanto, necessita de linha de visão para a comunicação, isto é, transmissor e receptor devem estar alinhados (Ruiz, Correia e Vieira 2004). Esta necessidade de linha de visão para a troca de dados inviabiliza seu uso em casos que haja movimentação dos nós ou de objetos próximos a eles, interrompendo a linha de visão.

A comunicação infravermelho, tal qual a comunicação óptica, necessita de linha de visão para sua operação, tornando-a inviável nos mesmos casos que a comunicação óptica (Ruiz, Correia e Vieira 2004).

Por fim, a comunicação em RF é baseada em ondas eletromagnéticas que se propagam no espaço e não necessitam linha de visão para funcionamento. Ondas eletromagnéticas são um meio não guiado, isto é, a propagação dos sinais não é restrita a determinados ambientes. Por este motivo, os nós de uma RSSF baseada em RF devem possuir mecanismos para evitar interferências no meio de transmissão. Essas interferências podem ser causadas pelos demais nós da rede e outros dispositivos que utilizem a mesma frequência (Karl e Willig (2005)).

Para um sistema de comunicação sem fio baseado em RF, uma frequência portadora deve ser cuidadosamente escolhida. A frequência portadora determina características da propagação do sinal - por exemplo, com que facilidade obstáculos são atravessados - e a capacidade de transferência de dados disponível. Uma porção do espectro eletromagnético, chamada banda de frequência é usada para realizar a comunicação. As bandas utilizáveis em RF em geral iniciam em 3kHz e se estendem até 300GHz, porém para uso sem necessidade de licenças de órgãos reguladores existem as bandas ISM (*Industrial, Scientific and Medical*) (Karl e Willig (2005)). Dentre as bandas ISM, a que se estende de 2.400MHz a 2.500MHz é especialmente relevante para este trabalho, pois o transceptor utilizado para implementação da RSSF (nRF24l01+) opera na faixa de 2.400MHz até 2.525MHz (Nordic_Semiconductor, 2013).

A camada física trata da transmissão de bits por um canal de comunicação. Um protocolo de camada física deve definir o que será interpretado como bit 1 ou bit 0 entre transmissor e receptor, além da sequência de bits que indicará o início de uma nova transmissão (preâmbulo) e se haverá ou não uma sequência para indicar o final de uma transmissão. Essa camada pode conter mecanismos para identificação e correção de erros, embora mecanismos mais sofisticados são encontrados nas camadas superiores.

A RF é uma rede de difusão, isto é, a rede tem apenas um meio de comunicação que é compartilhado por todos os nós. Os pacotes enviados por um nó da

rede podem ser recebidos por quaisquer outros nós no alcance do rádio e na mesma frequência de operação. Um campo de endereço no pacote especifica o destinatário pretendido. Ao receber um pacote, um nó verifica se o endereço de destino de pacote é o seu, caso contrário este pacote será descartado (Tanenbaum 2002).

2.3.2 Subcamada MAC

Essa camada pode ser conhecida como uma subcamada, pois tem relação direta com a camada de enlace. Em qualquer rede de difusão, a questão fundamental é controlar qual nó tem direito de usar o meio de transmissão quando ocorre uma disputa. Protocolos na camada MAC (*Medium Access Control* - Controle de Acesso ao Meio) executam a tarefa de controlar quando um nó poderá transmitir, quando deverá aguardar e por quanto tempo. Protocolos para a camada MAC são analisados na Seção 2.4.1.

2.3.3 Camada de Enlace

A camada de enlace trabalha sobre os serviços de transmissão e recepção providos pela subcamada MAC, e oferece serviços para as camadas superiores da rede. Uma das tarefas principais da camada de enlace é criar um canal de comunicação confiável para a tranferência de pacotes entre nós vizinhos, isto é, nós em mútuo alcance de rádio. Segundo Karl e Willig (2005), as tarefas desta camada podem ser divididas da seguinte forma:

- **Enquadramento:** os dados a serem enviados devem ser primeiramente divididos em quadros no tamanho suportado pelo *transceiver*. Esses quadros devem conter os dados (ou partes dos dados) bem como campos de controle para a verificação/correção de erros e remontagem dos dados no receptor.

- **Controle de erros:** variações nos sinais transmitidos podem ser introduzidas pelos meios de transmissão tornando pacotes transmitidos inúteis. Estes

erros devem ser reparados por meio de técnicas de recuperação de pacotes ou retransmissões.

- **Controle de fluxo:** o receptor de uma série de pacotes pode temporariamente estar impossibilitado de aceitar novos pacotes, por exemplo, por falta de espaço em *buffer*. Mecanismos de controle de fluxo devem regular a taxa de transferência de modo a reduzir a perda de pacotes.

- **Gerenciamento de *links*:** este mecanismo envolve a descoberta, criação, manutenção e remoção de *links* para nós vizinhos. Como quesito importante da manutenção de *links* está a estimativa de qualidade do *link*, que pode servir às camadas superiores informações para a alteração de rotas e topologias. Neste contexto ainda podem estar inseridos mecanismos de segurança como criptografia dos dados transmitidos.

2.4 Protocolos

Protocolos definem regras para a comunicação entre camadas de redes. Um protocolo na camada de enlace de um nó deve ser compatível com o protocolo na camada de enlace de outro nó da rede e assim sucessivamente. Nesta seção são analisados protocolos das camadas MAC e Enlace utilizados em RSSF.

2.4.1 Protocolos MAC

Os protocolos da camada MAC podem ser classificados, de acordo com seu modo de operação, como baseados em contenção (*contention-based*) ou baseados em agendamento (*schedule-based*) (Karl e Willig (2005)). Esta classificação se refere a como um nó da rede se comportará quando houver a necessidade de transmitir dados a algum outro nó. Um dos principais objetivos de um protocolo MAC é a autoconfiguração da RSSF, estabelecendo as conexões para transferência de dados (Akyildiz et al. (2002)). Outro objetivo importante é coordenar as comunicações e

evitar a ocorrência de colisões. Colisões ocorrem quando dois ou mais nós tentam transmitir dados simultaneamente, pelo mesmo meio físico e no mesmo canal. Um nó receptor no alcance dos transmissores é incapaz de diferenciar os sinais, e portanto, decodificar as mensagens corretamente.

2.4.1.1 Protocolos baseados em contenção

O funcionamento básico deste tipo de protocolo segue a seguinte forma: quando um dado está disponível o nó tenta transmití-lo imediatamente. Em caso de insucesso o nó aguarda um tempo aleatório e então tenta a transmissão novamente. Este procedimento é repetido até que o dado seja transmitido com sucesso ou que se esgote o número de tentativas definido pelo protocolo.

Com a criação de novos mecanismos para reduzir ou até evitar estas falhas, diferentes protocolos baseados nesta abordagem foram desenvolvidos e puderam oferecer melhor confiabilidade. A Figura 2.5 mostra o fluxograma de funcionamento do protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), que utiliza as mensagens de RTS (*Request to Send* - Solicitação para envio) e CTS (*Clear to Send* - Autorizado para envio).

Quando um nó (**A**) precisa transmitir um dado a outro nó (**B**), ele primeiramente tenta detectar se alguma transmissão já está ocorrendo no canal. Caso nenhuma transmissão seja detectada, uma mensagem RTS é enviada ao nó (**B**). O nó (**A**) aguarda, então, uma mensagem CTS. Se (**B**) não estiver recebendo nenhuma outra comunicação, ele responde com uma mensagem CTS, autorizando que o nó (**A**) transmita seus dados. No entanto, se (**B**) já estiver se comunicando com outro nó, ele não envia a mensagem de CTS, fazendo que o nó (**A**) aguarde um novo período de tempo até tentar transmitir para (**B**) novamente. Esse método visa reduzir as colisões e permitir uma operação mais confiável do protocolo.

Segundo Karl e Willig (2005) protocolos baseados em contenção têm, geralmente, um tempo de resposta menor. No entanto, em redes densas (com mais

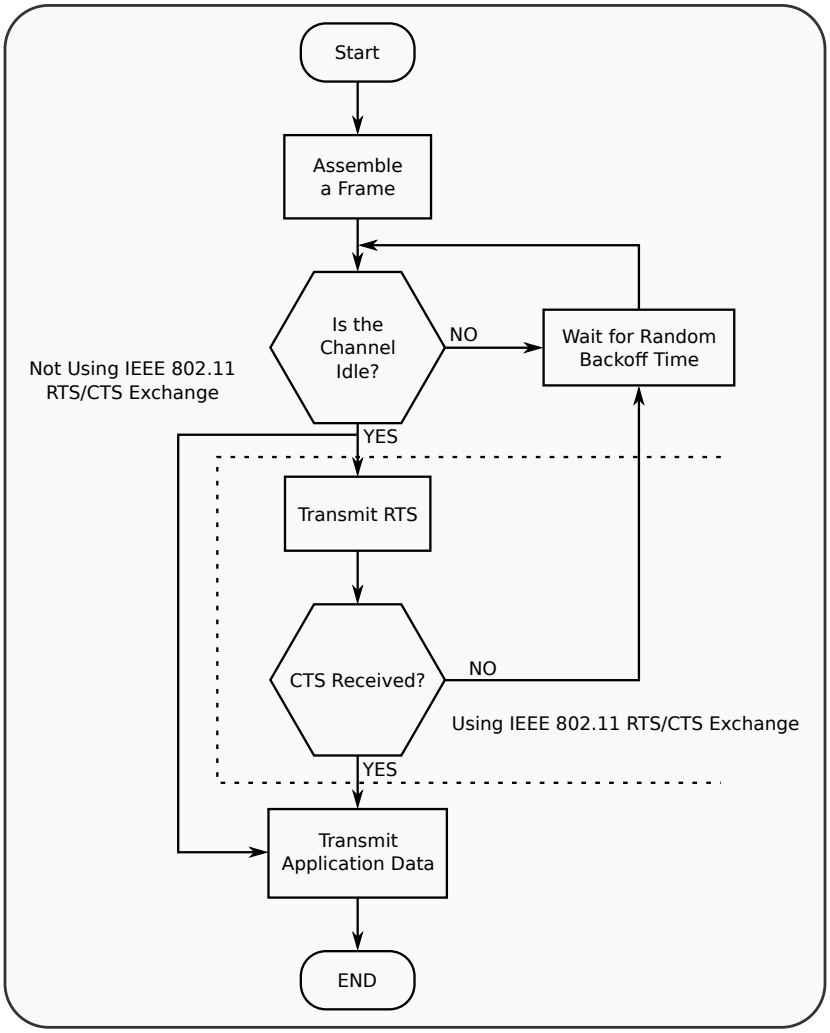

Figura 2.5: Fluxograma do protocolo CSMA/CA (JJGarcia.tsc 2011).

nós por unidade de área), a quantidade de colisões aumenta, elevando os tempos de resposta. Com tal imprevisibilidade, este tipo de protocolo é inadequado em situações críticas, onde se deseja uma estimativa confiável do tempo de resposta da rede.

2.4.1.2 Protocolos baseados em agendamento

Protocolos baseados em agendamento dividem um período de tempo em partes. Cada nó da rede recebe uma fatia do tempo, no qual será permitido enviar ou receber dados. Este modelo é conhecido por TDMA (*Time Division Multiple Access*).

A Figura 2.6 ilustra a divisão de tempo em um esquema TDMA, com cada ciclo contendo 8 *frames* (ou quadros), e cada *frame* contendo 2 *slots*. Neste modelo, cada nó da rede recebe uma fatia do ciclo, recebendo um ou mais *frames*, e uma quantidade de *slots*, onde somente ele poderá fazer suas transmissões. Desta forma, os nós que não participarão de um determinado *frame* podem permanecer desligados até que seja sua vez de operar, economizando energia.

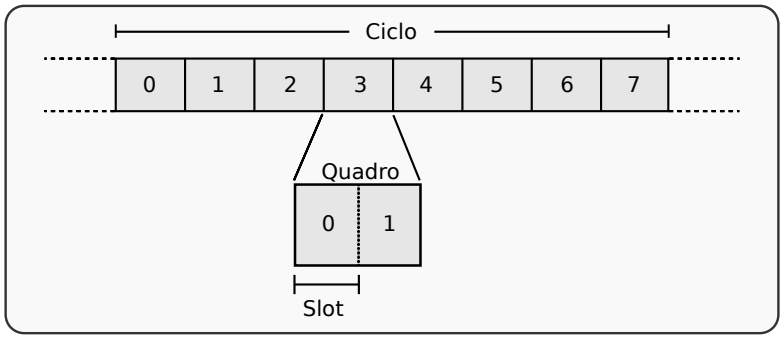

Figura 2.6: Exemplo de divisão do tempo em esquema TDMA.

Diferentes protocolos implementam diferentes variações deste modelo. Alguns possuem períodos reservados para mensagens de controle, configuração ou outras operações necessárias para o funcionamento da rede.

Como cada nó deve conhecer exatamente qual seu momento de operação, algum método de sincronização de tempo deve ser utilizado, de forma que os nós não tentem operar fora de seu período designado. A sincronização de tempo entre os nós de uma RSSF é fundamental para protocolos TDMA e será analisada na Seção 2.5.

2.4.2 Protocolos de Enlace

Os protocolos de enlace determinam os mecanismos para tratamento das atribuições desta camada. Nesta seção são apresentadas algumas técnicas para o controle de erros, enquadramento e gerenciamento de *link*.

As duas principais técnicas usadas para o controle de erros são a correção adiantada de erros (FEC - *Forward Error Correction*) e requisição automática de repetição (ARQ - *Automatic Repeat Request*).

Na técnica **ARQ**, o nó transmissor inicia um contador assim que envia um pacote pela rede. Caso uma resposta para este pacote não seja recebida em um determinado período de tempo, o nó tenta transmitir novamente. Geralmente um número limite de tentativas é estabelecido, e quando este limite é atingido, o protocolo de enlace deve informar à camada superior que a transmissão não pôde ser efetuada.

Na técnica **FEC**, o transmissor adiciona bits de redundância no pacote a ser transmitido. Caso o receptor receba os dados com erros, ele utiliza esses bits de redundância para tentar recuperar a informação ao invés de requisitar uma nova transmissão.

Em conjunto com as técnicas de ARQ e FEC, o controle da potência de transmissão pode ainda colaborar para a redução de erros na rede.

No processo de **enquadramento**, a camada de enlace recebe os dados da camada superior e constrói os quadros que serão posteriormente transmitidos. Um dos mais importantes aspectos neste processo é a definição do tamanho dos quadros. Com quadros maiores, a transmissão é mais suscetível às interferências do meio de transmissão. A definição do tamanho dos quadros deve considerar aspectos como a taxa de transferência de dados, e informações sobre a qualidade dos *links*.

O mecanismo de **gerenciamento de *link*** fornece às camadas superiores, principalmente aos protocolos de roteamento, informações importantes sobre os

vizinhos de um nó e a qualidade das conexões com estes vizinhos. Essas informações podem auxiliar na construção de rotas e colaborar para a redução da taxa de erros na rede (Karl e Willig (2005)).

2.5 Sincronização de tempo

Tempo é um quesito importante para muitas aplicações e protocolos encontrados em RSSF. Os nós da rede marcam o tempo baseado em osciladores internos (geradores de frequência). Um contador é incrementado a cada ciclo do oscilador, e com base no valor deste contador e a frequência de operação do oscilador pode-se criar um contador que incrementa a cada microsegundo, por exemplo. Devido a características físicas destes osciladores, o tempo marcado em cada nó começa a divergir, e a sincronização entre os nós é perdida.

Para se compreender o problema de sincronização entre nós de uma rede, são introduzidos alguns conceitos (Karl e Willig (2005)):

1. **Tempo Físico e Tempo Lógico:** o tempo físico entre nós de uma rede indica que todos os nós possuem a mesma concepção sobre a duração de 1 segundo em seus contadores. Da mesma forma, a duração deste 1 segundo deve ser a mais próxima possível de 1 segundo no tempo **real**, ou seja, do tempo tomado mundialmente como referência. O tempo lógico, permite conhecer a ordem em que eventos ocorrem em um sistema distribuído, mas não necessariamente tem relação direta com o tempo real.

2. **Relógio de *Hardware*:** o relógio de *hardware* se refere ao contador interno do equipamento. Este relógio é um contador, que é incrementado a cada ciclo do oscilador, geralmente a partir do momento que o equipamento é ligado. O contador tem um tamanho (em bytes) limitado. Assim, quando atinge o valor máximo, o contador é zerado e a contagem se inicia a partir de zero novamente.

3. **Relógio de *Software*:** o relógio de *software* marca um tempo calculado pelo programa em execução no nó. Este tempo é uma função de t, onde t é o tempo marcado pelo relógio de *hardware*.

4. **Tempo Local:** tempo local se refere ao tempo marcado por um nó individual da rede a partir de seus relógios de *hardware* e *software*.

5. **Tempo Global:** o tempo global indica o tempo que todos os nós de uma rede devem ter. Para determinadas aplicações e protocolos, este tempo deve ser o mesmo em todos os nós da rede. Os protocolos de sincronização geralmente se encarregam de manter este tempo o mais semelhante possível em todos os nós da rede.

6. ***Clock Offset*:** *clock offset* ou deslocamento de tempo, indica a diferença de tempo entre dois nós que foram ligados em momentos diferentes. Como o contador interno dos nós é iniciado quando o nó é ligado, um nó que foi ligado antes terá em seu contador um valor maior que um nó que acaba de ser ligado.

7. ***Clock Drift*:** *clock drift* ou deriva de tempo, se refere à diferença que os contadores adquirem com o passar do tempo. Devido a características físicas dos osciladores, estes têm pequenas variações nos seus pulsos durante sua operação. Estas variações influenciam nos contadores, de modo que um contador pode acabar ficando "mais rápido"ou "mais lento"que outro, e portando, alterando a proporção que os valores destes contadores crescem.

Os conceitos de *clock offset* e *clock drift* são importantes para a elaboração de algoritmos de sincronização. A relação entre eles e o relógio de *hardware* de um equipamento, para o cálculo do tempo global, é representada pela Equação 2.1, onde para **TempoLocal** é considerado o valor do relógio de *hardware* do nó.

$$TempoGlobal = Offset + Drift \times TempoLocal \tag{2.1}$$

A Figura 2.7 ilustra os conceitos de *clock offset* e *clock drift* entre dois nós de uma rede, onde o *drift* representa o ângulo de inclinação da reta referente a um nó em relação à reta ideal, e o *offset* seu deslocamento inicial.

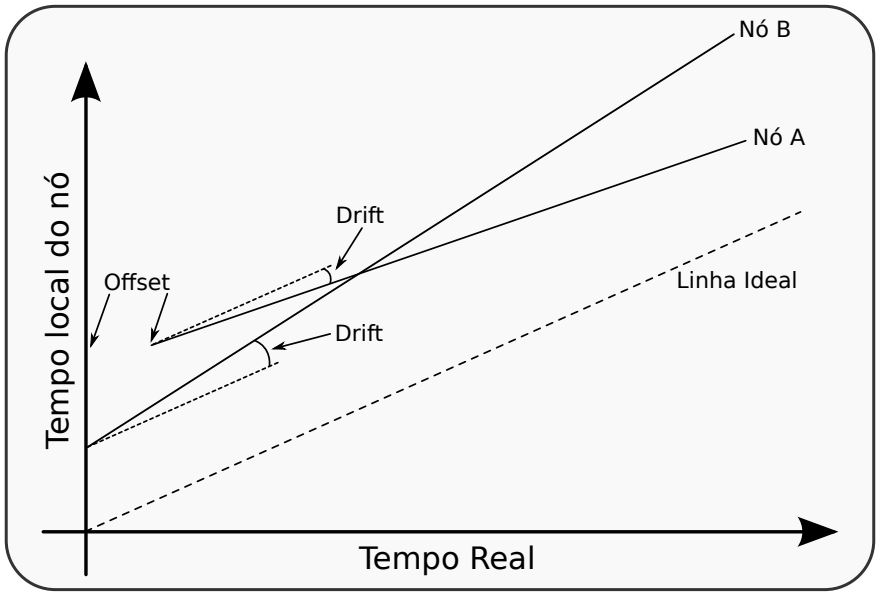

Figura 2.7: Clock Drift e Clock Offset.

Para fazer a sincronização, os nós da rede precisam trocar mensagens contendo o valor de seus relógios. O tempo do nó transmissor é colocado na mensagem logo antes da transmissão ser feita. Ao receber uma mensagem deste tipo, o nó deve marcar o tempo em que ela foi recebida. Estes dados servem de base para os algoritmos que fazem a conversão do tempo.

Algoritmos de sincronização simples se baseiam apenas na estimativa de *offset*. Entretanto, para manter os nós sincronizados, mensagens de sincronização devem ser trocadas com maior frequência, caso contrário, devido ao *clock drift*, a sincronização será perdida com o passar do tempo. O *offset* pode ser calculado pela Equação 2.2.

$$Offset = TempoGlobalRecebido - TempoLocal \tag{2.2}$$

Algoritmos mais sofisticados estimam o *drift* utilizando regressão linear ou outras técnicas, que possibilitam manter a sincronização mesmo com uma troca menos frequente de mensagens (Maróti et al. (2004)). Estes algoritmos geralmente requerem mais processamento e mais amostras para realizar os cálculos, assim a abordagem utilizada para fazer a sincronização de tempo na rede deve levar em conta a capacidade de transmissão de dados, e o poder de processamento dos nós.

Tendo calculados o *clock offset* e o *clock drift*, um nó da rede é capaz de calcular o tempo global pela Equação 2.1. Caso o algoritmo não estime o valor do *clock drift*, este é considerado igual a um.

2.6 WirelessHART

O WirelessHART é um padrão para redes sem fio de monitoramento e controle industriais. De acordo com a *Hart Communication Foundation*, fundação responsável pela especificação do padrão, este padrão é aberto para uso, ou seja, quaisquer empresas que desejarem implementá-lo em seus equipamentos são livres para fazê-lo. A especificação do padrão no entanto é paga, e caso seja desejada a implementação do protocolo seguindo as normas oficiais, os documentos de especificação devem ser adquiridos por meio da fundação (HART, 2014). Os valores dos documentos, segundo o site oficial da fundação estão descritos na tabela 2.1. Os valores são para aquisição de cópias impressas dos documentos, não são considerados impostos e outras tarifas de importação.

Petersen e Carlsen (2011) analisam os fundamentos do padrão *WirelessHART* e descrevem os seguintes elementos como base para esta rede.

- ***Field Devices*:** equipamentos com capacidade de comunicação sem fio com uma rede *WirelessHART*.

Tabela 2.1: Preço dos documentos na *HART Communication Foundation*.

Documento	Preço** (US$)
HART Field Communication Protocol Specifications (Rev 7.5)	975,00
HART Field Communication Protocol Test Specifications (Rev 7.5)	500,00
DDL Specifications	250,00
HART Field Communications Protocol, A Technical Overview	N/D*
The HART Data Link Layer, A Requirement Analysis	50,00
Application Note: Information for HART Modem Designers	50,00
HART Server	500,00
TOTAL	2.325,00

* Valor ainda não determinado para a nova versão do documento.
** Valores obtidos pelo site da *HART Communication Foundation* (http://en.hartcomm.org/) em 28/11/2014.

- **Adaptador:** módulo de comunicação sem fio que pode ser acoplado a equipamentos HART comuns, proporcionando a estes integração com uma rede *WirelessHART*.

- ***Handheld*:** um computador *WirelessHART* portátil, utilizado para configuração, diagnósticos e calibração de *field devices*.

- ***Gateway*:** um ponto de acesso que conecta a rede *WirelessHART* aos sistemas de controle de processos da planta industrial e aos gerenciadores da rede.

- ***Network Manager*:** o *network manager* ou gerenciador de rede é responsável pela configuração e manutenção da rede *WirelessHART*. Este gerenciador solicita informações dos *field devices* e com base nessas informações determina, por exemplo, quais rotas devem ser usadas para comunicação.

- ***Security Manager*:** o gerenciador de segurança é responsável pelos sistemas de segurança que serão empregados na rede, dentre eles, as chaves de criptografia utilizadas na transmissão de dados e os dispositivos autorizados a operar na rede.

Uma rede *WirelessHART* e seus dispositivos é ilustrada na Figura 2.8.

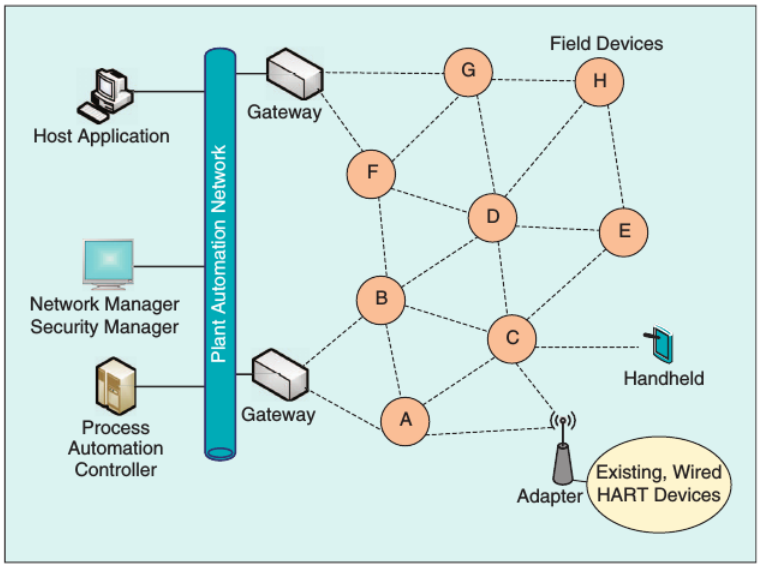

Figura 2.8: Rede *WirelessHART* típica (Petersen e Carlsen (2011)).

No padrão *WirelessHART*, todos os equipamentos possuem a capacidade de roteamento, ou seja, são capazes de encaminhar os pacotes de outros nós. Desta forma, caso a falha de um nó interrompa uma rota, outro nó pode assumir a função e manter o funcionamento da rede (Petersen e Carlsen (2011)).

Na camada física, equipamentos *WirelessHART* utilizam rádios baseados no padrão IEEE 802.15.4 com algumas modificações. Os equipamentos operam na faixa de 2400MHz até 2483,5MHz (ISM), com taxa de transferência máxima de 250kbit/s e 15 canais de operação (Song et al. 2008).

A camada de enlace é subdividida em camada MAC e Controle Lógico de *Link* (LLC - *Logical Link Control*). Esta camada tem como objetivo fornecer uma conexão confiável entre nós vizinhos.

Para a camada MAC, o padrão define um protocolo baseado em TDMA, com o propósito de reduzir colisões e o consumo de energia dos dispositivos. Cada *slot* de tempo tem tamanho fixo de 10 milissegundos. Em cada *slot*, um nó transmissor envia um pacote de dados a um nó receptor, que após receber o pacote, responde com uma confirmação de recepção. O tempo de um *slot* é o sufici-

ente para a transmissão de apenas um pacote de dados e o pacote de confirmação (Kim et al. 2008).

A camada de rede é responsável por manter a tabela de roteamento e por rotear os pacotes pela rede. Todos os dispositivos na rede *WirelessHART* possuem um conjunto de tabelas de roteamento. Estas tabelas são criadas e distribuídas pelo gerenciador de rede (Petersen e Carlsen (2011)).

A camada de transporte é responsável pela comunicação fim-a-fim na rede. O protocolo desta camada permite a troca de mensagens de confirmação de transmissões entre os dois pontos, provendo um caminho confiável de comunicação através de vários nós da rede.

Por fim, a camada de aplicação define comandos que são trocados entre *field devices* e os sistemas de controle. Os dispositivos devem implementar estes comandos para permitirem sua integração com outros dispositivos *HART* ou *WirelessHART*. Estes comandos são definidos na especificação do protocolo.

2.7 TreeMAC

O protocolo da camada MAC TreeMAC (Song et al. 2009) é um protocolo para RSSF baseado em TDMA. O objetivo deste protocolo é garantir uma alta taxa de transferência de dados e oferecer uma distribuição justa de tempo entre os nós. O protocolo organiza a rede na forma de uma árvore.

Para distribuir o tempo de forma justa para os nós, o protocolo leva em consideração a quantidade de filhos que este nó possui. Assim, um nó com N filhos deve receber $N+1$ *frames*, pois além de precisar de um tempo para transmitir seus próprios dados, este nó é responsável por redirecionar dados de seus N filhos. A Figura 2.9 ilustra a quantidade de pacotes que um nó intermediário deve encaminhar, e portanto, estes nós devem receber uma fatia proporcional do tempo. Os nós g, i, j e k necessitam de apenas uma fatia do tempo para transmitir seus dados. O nó h necessita de 4 fatias, pois encaminha dados vindos de seus nós filhos

e mais seus próprios dados. Por fim, o nó f necessita de 6 fatias de tempo, pois encaminha dados de todos os outros nós, mais os gerados por ele.

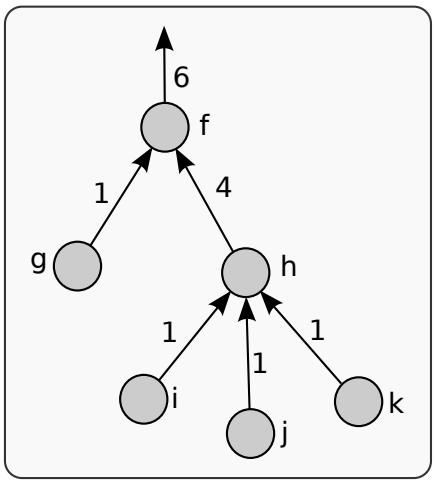

Figura 2.9: Demanda de tempo pelos nós de uma rede organizada em árvore.

Um nó distribui *frames* aos seus filhos de acordo com os *frames* que recebeu. Os *frames* de um filho não podem sobrepor-se aos *frames* de outro filho. Desta forma evita-se interferência horizontal na rede.

Cada *frame* da rede é dividido em 3 *slots*. Os nós calculam em qual *slot* devem transmitir de acordo com a sua distância (em saltos) do *sink*. Desta forma, o *sink* pode transmitir no *slot* 0, seus filhos transmitem no *slot* 1 e assim sucessivamente. O cálculo de *slot* é realizado pela Equação 2.3, utilizando o operador de módulo. Um nó a 3 saltos do *sink*, por sua vez, transmitirá no *slot* 0, evitando-se interferências verticalmente a dois saltos de distância. A Figura 2.10 mostra como fica a distribuição de *frames* e *slots* entre os nós da rede mostrada na Figura 2.9. As setas indicam em qual *slot* o nó pode transmitir de acordo com o cálculo feito baseado em saltos até o *sink* (neste caso o nó f).

$$SlotTransmissao = Profundidade \bmod 3 \tag{2.3}$$

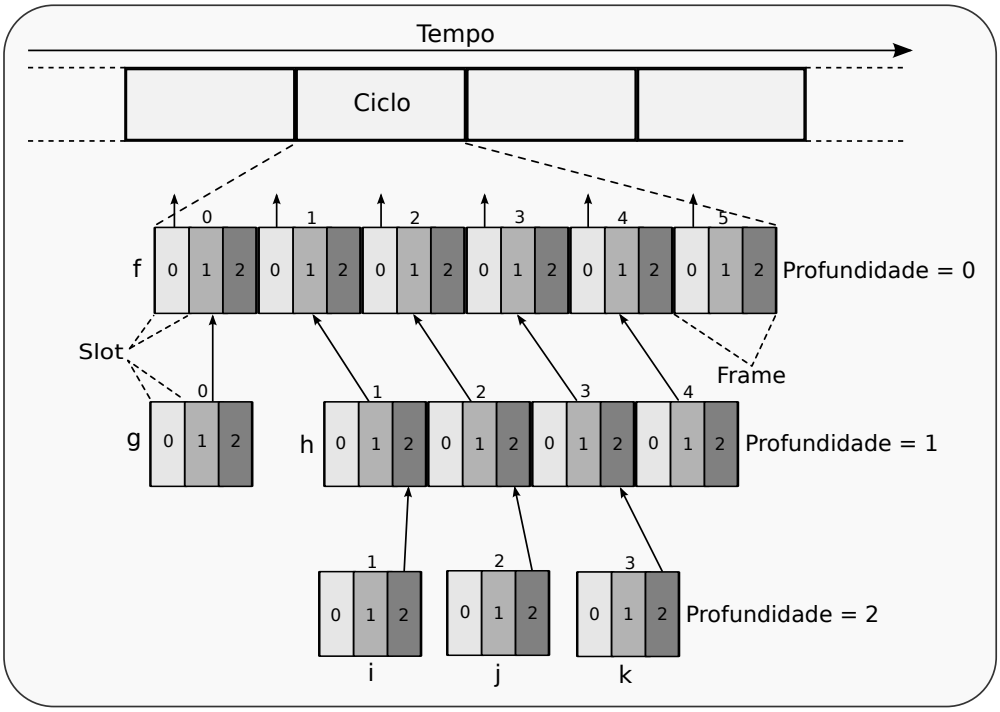

Figura 2.10: Distribuição de tempo entre os nós.

2.8 Plataformas de prototipagem

Para o desenvolvimento de novos produtos, é conveniente o uso de plataformas de prototipagem. Estas plataformas permitem a avaliação de técnicas, protocolos e outros equipamentos para a posterior construção dos produtos finais.

Dentre os principais fabricantes de plataformas de prototipagem com especificação aberta (especificação de *hardware* disponibilizada gratuitamente), estão Arduino (Arduino, 2014), BeagleBoard (BeagleBoard, 2014) e Gumstix (Gumstix, 2014).

A plataforma Arduino, mais especificamente a placa Arduino Uno, possui um microcontrolador Atmel ATmega328P-PU. Ao contrário dos processadores baseados na arquitetura ARM utilizados nas placas desenvolvidas pela BeagleBoard e Gumstix, estes microcontroladores podem ser encontrados no mercado em encapsulamento DIP, que facilita o processo de criação de produtos posteriormente por não necessitar de equipamentos de solda avançados. A Figura 2.11 mostra o microcontrolador ATmega328P-PU, e a Figura 2.12 o processador OMAP3530, semelhante aos utilizados nas placas desenvolvidas por BeagleBoard e Gumstix. Outras características desta placa são analisadas na Seção 3.2.1.

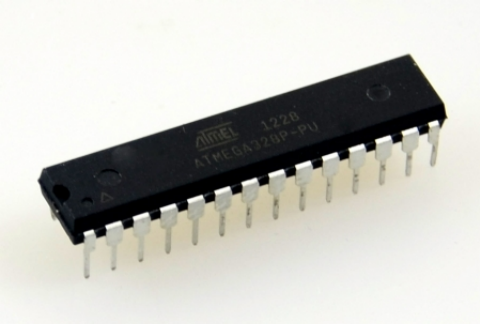

Figura 2.11: Microcontrolador Atmel ATmega328P-PU.

As plataformas desenvolvidas por BeagleBoard e Gumstix possuem mais recursos de *hardware*, sendo ideais para o desenvolvimento de produtos que re-

Figura 2.12: Processador OMAP3530.

querem mais recursos. Estas plataformas, no entanto, têm um custo mais elevado, e devido a características de sua arquitetura, a reprodução das placas de prototipagem requerem equipamentos mais sofisticados.

Assim, a facilidade de construção de equipamentos baseados na plataforma Arduino Uno, e seu preço mais acessível foram determinantes para a escolha desta plataforma para o desenvolvimento da RSSF.

2.9 Custo dos equipamentos

Para construir nós de baixo custo, foram avaliados os valores dos componentes necessários para a construção, bem como suas características de operação, que são fundamentais para o projeto da RSSF. Na Tabela 2.2 são apresentados alguns transceptores de rádio que podem ser utilizados para a construção de nós de RSSF, algumas características importantes destes equipamentos, e o preço de cada um.

Tabela 2.2: Comparativo entre transceptores de rádio.

Características	nRF24l01+	TR24A	CC2520	Zigbee SZ05-PRO
Preço* (R$)	20	30	274	100
Taxa de transmissão (máxima)	2Mbps	1Mbps	250Kbps	115,2Kbps
Canais de operação	126	81	16	16
Tensão de alimentação (V)	1,9 a 3,6	2,5 a 3,7	1,8 a 3,8	5
Corrente (máx) em modo TX	11,3mA	26mA	33,6mA	N/E**
Corrente (máx) em modo RX	12,3mA	25mA	18,5mA	N/E**
Frequência de operação (MHz)	2.400 a 2.525	2.400 a 2.482	2.394 a 2.507	2.405 a 2.480

* Valores estimados por meio de sites de busca.
** Valores não especificados no manual.

O transceptor nRF24l01+ é superior aos demais equipamentos nos quesitos apresentados, e possui preço inferior, sendo uma boa opção para a construção de nós baratos e robustos.

Como base para o desenvolvimento da RSSF, foram avaliadas diferentes plataformas de prototipagem. A Tabela 2.3 apresenta algumas características e os preços das plataformas de prototipagem discutidas na Seção 2.8 (Plataformas de prototipagem). O principal quesito avaliado foi o valor dos equipamentos, pois com plataformas de custo reduzido, os nós da rede são mais baratos de serem construídos, e sua utilização pode se tornar mais abrangente.

Tabela 2.3: Comparativo entre plataformas de prototipagem.

Características	Arduino UNO R3	Gumstix Overo® EarthSTORM COM	BeagleBone Black Rev C
Preço* (R$)	80	274,96*	284,05
CPU/MCU	Atmel ATMega 328P-PU	TI AM3703	TI AM3359
Memória RAM	2KB	512MB	512MB
Memória para programas	32KB	Superior a 4GB	Superior a 4GB

* Valor obtido no site oficial em US$ convertido para R$ em 28/11/2014.

Apesar de ter características de *hardware* inferiores às das plataformas concorrentes, a placa Arduino Uno possui um custo mais acessível. Além disso, não é necessário o uso de sistemas operacionais para simplificar sua operação, os programas desenvolvidos são gravados diretamente no microcontrolador e executados quando o equipamento é ligado. Assim, para o desenvolvimento dos protocolos não é necessário o desenvolvimento de módulos para nenhum sistema operacional específico.

3 METODOLOGIA

Esta seção apresenta a metodologia utilizada para o desenvolvimento do trabalho, o tipo de pesquisa desenvolvida, os materiais e métodos utilizados, o cenário de testes e, por fim, as métricas definidas para a avaliação do trabalho desenvolvido.

3.1 Tipo de Pesquisa

De acordo com Jung (2004), a pesquisa realizada neste trabalho é de natureza aplicada. Esta pesquisa tem como objetivo a obtenção de novos produtos ou processos, combinando diferentes métodos já conhecidos a fim de gerar uma nova abordagem para a resolução de um problema. A pesquisa aplicada apresenta resultados que podem ser medidos pela sua capacidade de resolução de problemas concretos.

3.2 Materiais

Nesta seção são descritos os equipamentos de *hardware* utilizados para a construção dos nós da RSSF e suas características.

3.2.1 Arduino

A placa de prototipagem Arduino Uno (Figura 3.1) possui um microcontrolador Atmel ATmega328P-PU. Este microcontrolador possui uma memória de programa de 32KB, onde os programas desenvolvidos ficam armazenados para execução, uma memória RAM de 2KB, utilizada durante a execução do programa, e uma memória EEPROM de 1KB onde os dados ficam armazenados mesmo após o equipamento ser desligado. O microcontrolador possui ainda um conversor analógico/digital com 6 entradas multiplexadas, que são utilizados para a aquisição de dados por meio de sensores, barramento SPI para comunicação com periféricos

com taxas de transferência de até 8Mbit/s, 3 temporizadores internos para o uso de interrupções, além de outras características.

Além do microcontrolador, a placa possui conexão USB, por onde é feita a programação e que serve de interface de comunicação entre um computador e o microcontrolador, conector de alimentação e reguladores de tensão, fornecendo tensões de 5 volts e 3,3 volts utilizadas para alimentação de módulos adicionais.

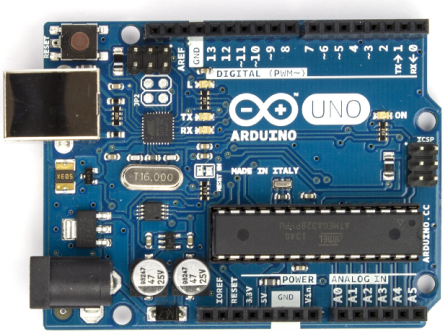

Figura 3.1: Placa de prototipagem Arduino Uno.

3.2.2 Rádio nRF24l01+

O módulo de rádio nRF24l01+ da *Nordic Semiconductor* é um *transceiver* de rádio, ou seja, é capaz de fazer a transmissão e a recepção de dados. O módulo é capaz de operar como transmissor ou receptor, de acordo com a configuração utilizada, mas não nos dois modos ao mesmo tempo. A conexão com o Arduino é feita pelo barramento SPI, desta forma tem-se acesso aos registradores de configuração, comandos e *buffers* do rádio.

De acordo com o manual do transceptor (Nordic_Semiconductor, 2013), destacam-se as seguintes características do nRF24l01+:

- **Frequências de operação:** opera na faixa de frequência entre 2.400MHz e 2.525MHz, com 126 canais possíveis de 1MHz.

- **Potência de transmissão programável:** possui quatro níveis de potência de transmissão. Com potência mais baixa, o consumo de energia é menor, porém o alcance da transmissão também fica reduzido. A relação de potência e consumo é 11,3mA a 0dBm, 9mA a -6dBm, 7,5mA a -12dBm e 7mA a -18dBm.

- **Taxa de transmissão:** taxa de transmissão configurável em 1Mbps, 2Mbps ou em 250Kbps. Dois módulos devem estar configurados na mesma velocidade para se comunicarem.

Além destas características, este módulo possui a tecnologia *Enhanced ShockBurst*™, que funciona como a técnica ARQ (*Automatic Repeat Request*, ou Requisição Automática de Repetição), mencionada em 2.4.2, implementada diretamente no módulo.

Quando o rádio está configurado como transmissor e recebe um dado em seu *buffer*, um pacote da camada de enlace (ou quadro) é montado e então transmitido. Este pacote é composto por:

- **Preâmbulo (1 byte):** o preâmbulo pode ser 01010101, caso o primeiro bit do endereço seja 0, ou 10101010 caso o primeiro bit do endereço seja 1.

- **Endereço (3 a 5 bytes):** endereço do nó de destino do pacote. Este endereço pode ser configurado para 3, 4 ou 5 bytes.

- **Campo de Controle (9 bits):** neste campo são armazenados dados de controle sobre o pacote, como tamanho do campo de dados, identificador do pacote e controle de mensagem de confirmação.

- **Dados (0 a 32 bytes):** o campo de dados contém os dados que se deseja transmitir de um nó a outro. Este campo tem tamanho variável e pode conter no máximo 32 bytes.

- **CRC (1 ou 2 bytes):** o CRC é um valor calculado a partir de todos os outros campos, exceto o preâmbulo. Este valor é utilizado para verificar a integridade dos dados recebidos.

A Figura 3.2 ilustra o pacote gerado pelo rádio nRF24l01+.

Figura 3.2: Pacote gerado pelo módulo nRF24l01 com *Enhanced ShockBurst*™.

Ao receber um pacote, o módulo de rádio recalcula o valor do CRC e compara com o recebido no pacote. Caso os valores sejam iguais o campo de dados é extraído e colocado em um *buffer* para que o microcontrolador possa fazer a leitura. Caso os valores de CRC sejam diferentes, o comportamento do rádio fica definido por duas diferentes configurações.

A primeira, é o bit **NO_ACK**, no campo de controle do pacote. Caso este bit esteja em 1, o pacote será descartado imediatamente. Caso esteja em 0, o comportamento é definido pela configuração do registrador **04 - SETUP_RETR**. Neste registrador, são configuradas quantas tentativas de retransmissão serão feitas pelo transmissor, e quanto tempo deve-se esperar entre cada tentativa. O tempo mínimo de espera entre cada tentativa é de 250μs, o tempo máximo de 4.000μs. A quantidade mínima de tentativas de retransmissão é 0 e a quantidade máxima 15. Após atingir o número máximo de tentativas, o módulo descarta o pacote e retorna um sinal de erro, o qual deve ser tratado nas outras camadas.

Desta forma, o nRF24l01+ implementa em *hardware* um sistema para prover confiabilidade na comunicação entre dois nós para as camadas superiores.

3.3 Métodos

Nesta seção são apresentados os conceitos e métodos utilizados para a organização da RSSF desenvolvida.

3.3.1 Topologia

Com o protocolo MAC baseado no TreeMAC, discutido não seção 2.7, a RSSF desenvolvida fica organizada logicamente como uma árvore. A raiz da árvore é o *sink* da RSSF. O *sink* é o nó que fica diretamente conectado ao servidor, recebendo dados pelo rádio e transmitindo ao servidor pela porta USB. O *sink* pode, ainda, receber comandos do servidor e transmiti-los aos nós da rede. Por padrão, os dados coletados pelos nós são transmitidos para o *sink*. No servidor, uma aplicação fica responsável por receber e gerenciar os dados, bem como transmitir comandos.

3.3.2 Tipos de Dispositivos

Os dispositivos da rede são classificados como ***Field Devices*** ou ***Network Devices***.

Os ***Field Devices*** são responsáveis apenas pela coleta de dados e sua transmissão. Nestes dispositivos podem estar acoplados sensores e atuadores que fazem o monitoramento e controle de processos. Além disso, os *Field Devices* podem servir como adaptadores a outros dispositivos já existentes que realizam monitoramento e controle. Desta forma, os equipamentos transferem os dados ao *Field Device*, que por sua vez os transmitem pela rede.

Os ***Network Devices*** possuem as mesmas funcionalidades dos *Field Devices*, mas são responsáveis por coordenar uma subseção da rede (ou sub-rede). Estes nós são responsáveis por encaminhar os pacotes vindos da sub-rede até o *sink*, e do *sink* para a sub-rede, além coordenar a sincronização de tempo dos nós filhos, o canal de operação, e solicitar a transmissão de dados.

3.3.3 Endereçamento

O rádio nRF24l01+ suporta endereços de entre 3 e 5 bytes. Para o desenvolvimento da RSSF, optou-se por usar endereços de 4 bytes. Na linguagem de programação utilizada na plataforma Arduino não existem tipos primitivos de tamanho igual a 5 bytes. Desta forma, seria necessário o uso de variáveis de 8 bytes para comportar

os endereços, gerando desperdício de memória. Da mesma forma, não existem tipos de 3 bytes, então endereços deste tamanho limitariam a quantidade de nós e ainda haveria desperdício de memória. Com endereços de 4 bytes, os valores podem ser armazenados em variáveis do tipo primitivo **uint32_t**, permitindo uma melhor relação entre quantidade de endereços possíveis e uso de memória RAM.

O endereço do *sink* ocupa apenas os quatro bits mais significativos do endereço (os quatro mais à esquerda). A partir deste endereço, são gerados os endereços do restante da árvore. Se o endereço do *sink* é (em hexadecimal) 0xA0.00.00.00, seu primeiro nó filho terá o endereço 0xA1.00.00.00, o segundo 0xA2.00.00.00 e assim por diante até 0xAF.00.00.00. O primeiro filho do nó 0xA1.00.00.00 terá o endereço 0xA1.10.00.00, o segundo 0xA1.20.00.00 e assim por diante até 0xA1.F0.00.00. Com esta forma de endereçamento, cada nó pode ter no máximo 15 filhos, e a profundidade máxima da rede é de 7 saltos. Esta forma de endereçamento permite 268.435.455 endereços quando há somente um *sink*.

A Figura 3.3 ilustra a organização lógica da rede com os diferentes tipos de nós, as sub-redes formadas e o endereçamento de cada nó.

3.3.4 Pacotes

Os campos dos pacotes transmitidos pelo nRF24l01+ são montados pelo próprio módulo. O programador pode apenas configurar algumas características destes campos, como descrito na Seção 3.2.2. Desta forma, o programador deve colocar no campo de dados, outras informações que sejam necessárias em suas transmissões, como endereços de origem e destino, números de sequência, e outros dados necessários.

Para o desenvolvimento da RSSF, foram definidos quatro tipos de mensagem. Estas mensagens são transferidas para o módulo, que as coloca no campo de dados de seu pacote e as transmite. No receptor, o dado é extraído do pacote e tratado de acordo com um cabeçalho. As mensagens definidas, que são tratadas

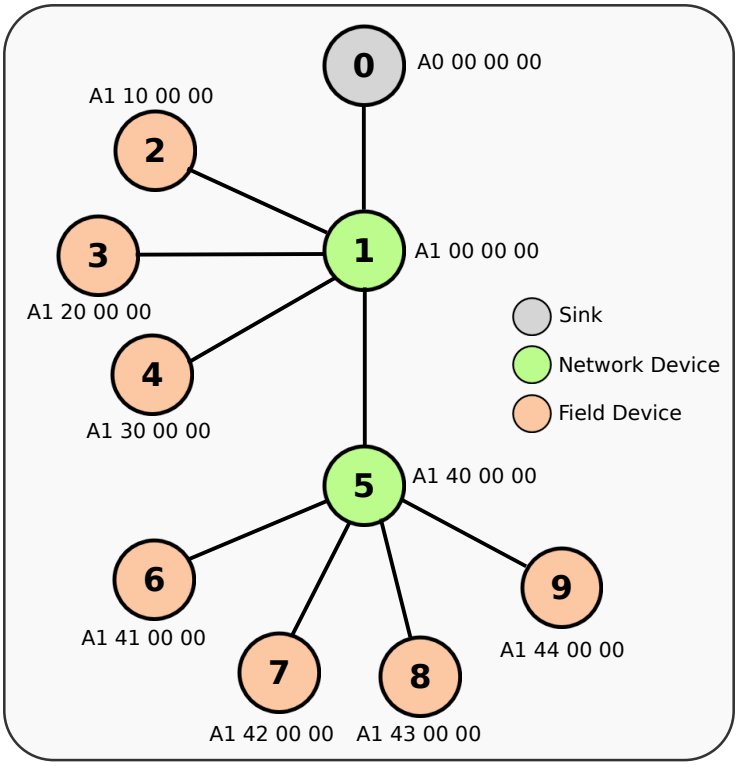

Figura 3.3: Organização da RSSF utilizada para os testes.

como pacotes da camada de enlace, são: **pacote de sincronização**, **pacote de dados** e **pacote de controle**. Cada tipo de mensagem possui campos específicos de acordo com sua aplicação, são eles:

O **pacote de sincronização** é utilizado para troca de mensagens para a sincronização de tempo entre um par de nós da rede. Os campos deste pacote são definidos na Tabela 3.1.

O **pacote de controle** é utilizado para troca de mensagens de configuração da rede, controle de transmissão e envio de comandos para os nós. Os campos deste pacote são definidos na Tabela 3.2.

O **pacote de dados** é utilizado para o envio de dados de monitoramento coletados pelos nós até o *sink*. Os campos deste pacote são definidos na Tabela 3.4.

Tabela 3.1: Pacote de Sincronização.

Campo	Descrição
header **(1 byte)**	Cabeçalho do pacote. Utilizado para que o programa reconheça que é uma mensagem de sincronização e trate-a como tal. Para este tipo de mensagem o cabeçalho foi definido como o valor inteiro 1.
id **(2 bytes)**	Número de identificação do pacote. Este valor é incrementado a cada novo pacote enviado. Utilizado com o propósito de avaliação e testes do funcionamento da rede desenvolvida.
globalTime **(4 bytes)**	Tempo global da rede, de acordo com o nó de origem. Este valor é o tempo global da rede estimado pelo nó de origem, e é atribuído logo antes do pacote ser enviado.
localTime **(4 bytes)**	Ao receber um pacote de sincronização, este campo é preenchido com o tempo local do nó. Este pacote é então passado para a função de conversão, que irá calcular o valor de *offset* para o tempo da rede.
frame **(1 byte)**	*Frame* do nó de origem. Ao receber o pacote, o nó atribui este valor à sua variável de *frame* local.
slot **(1 byte)**	*Slot* do nó de origem. Ao receber o pacote, o nó atribui este valor à sua variável de *slot* local.
source **(4 bytes)**	Endereço de origem do pacote de sincronização. Este campo é utilizado para garantir que os pacotes de sincronização utilizados sejam apenas aqueles vindos de nós mais próximos do *sink*.
data **(15 bytes)**	Campo reservado para que dados adicionais sejam colocados no pacote, permitindo que dados de monitoramento sejam transmitidos caso necessário.

Tabela 3.2: Pacote de Controle.

Campo	Descrição
header **(1 byte)**	Cabeçalho do pacote. Utilizado para identificação da mensagem como um pacote de controle. Para pacotes de controle o cabeçalho foi definido como o valor inteiro 2.
id **(2 bytes)**	Número de identificação do pacote. Este valor é incrementado a cada novo pacote enviado. Utilizado com o propósito de avaliação e testes do funcionamento da rede.
source **(4 bytes)**	Endereço do nó de origem do pacote.
destination **(4 bytes)**	Endereço de destino do pacote.
network frame count **(1 byte)**	Quantidade total de *frames* na rede.
frame count **(1 byte)**	Quantidade de *frames* designados ao nó. De acordo com o campo ***command***, este valor pode ser uma quantidade de *frames* designadas ao nó de destino, ou então a informação de quantos *frames* foram designados ao nó em uma configuração anterior.
lower frame **(1 byte)**	*Frame* inferior recebido pelo nó. Como os *frames* de um nó são sempre contíguos, basta conhecer o *frame* inferior e a quantidade de *frames* para saber seu período de operação.
hops **(1 byte)**	Quantidade de saltos para o *sink*. Cada nó calcula sua distância do *sink*. Este valor é necessário para o determinar o *slot* e o canal de operação dos nós. Em uma mensagem de controle do *sink* para seus filhos, este valor é 0. Ao receber o pacote, um filho do *sink* incrementa este valor em 1 e atribui ao seu contador de saltos local. Ao gerar um pacote de controle, este filho envia o valor de seu contador local. Desta forma, os nós conhecem a sua distância, em saltos, para o *sink*.
channel **(1 byte)**	Campo utilizado para a configuração do canal em que a sub-rede deve trabalhar. A cada 2 saltos a sub-rede trabalha em um canal secundário, permitindo que transmissões ocorram simultaneamente em diferentes níveis de profundidade na rede. Esta configuração será analisada com mais detalhes na Seção 3.3.6.
command **(1 byte)**	Comando enviado ao nó. Um valor numérico representa um comando enviado ao nó. Com este tamanho, são possíveis 255 comandos diferentes. Na RSSF desenvolvida, foram definidos 4 comandos. A Tabela 3.3 especifica estes comandos.
data **(15 bytes)**	Campo reservado para que dados adicionais sejam colocados no pacote, permitindo que dados de monitoramento sejam transmitidos caso necessário.

Tabela 3.3: Comandos definidos para pacotes de controle.

Valor	Comando
1	Configuração básica do nó. Com base nos dados recebidos, o nó calcula sua distância do *sink*, *slot* de transmissão e canal de operação.
2	Requisição de configuração. Com este comando, um *network device* solicita dos nós em sua sub-rede (a 1 salto), o *frame* inferior e a quantidade de *frames*.
3	Resposta a requisição de configuração. Este comando indica que o pacote é uma resposta a uma requisição de configuração recebida. O nó preenche o pacote com as informações e envia ao nó que requisitou.
4	Aguardando dados. Com este comando, o nó indica que já transmitiu seus dados e está pronto para receber dados do nó de destino.

Tabela 3.4: Pacote de Dados.

Campo	Descrição
header **(1 byte)**	Cabeçalho do pacote. Utilizado para identificação da mensagem como um pacote de dados. Para este tipo de mensagem o cabeçalho foi definido como o valor inteiro 3.
id **(2 bytes)**	Número de identificação do pacote. Este valor é incrementado a cada novo pacote enviado. Utilizado com o propósito de avaliação e testes do funcionamento da rede.
source **(4 bytes)**	Endereço do nó que originou o pacote de dados.
destination **(4 bytes)**	Endereço de destino do pacote. Ao gerar um novo pacote de dados, o valor padrão deste campo será o endereço do *sink*.
data **(21 bytes)**	Campo com os dados coletados pelo sistema de monitoramento. O formato dos dados contidos neste campo são definidos pela aplicação da RSSF.

3.3.5 Sincronização de tempo

A sincronização de tempo entre os nós da rede é fundamental para o correto funcionamento de um protocolo TDMA. Na RSSF desenvolvida, um nó somente inicia sua operação após estar sincronizado. Um nó é considerado sincronizado se ele já recebeu pelo menos um pacote de sincronização.

O protocolo de sincronização é responsável por estimar o tempo global da rede. Na RSSF desenvolvida, este protocolo estima este tempo baseado no deslocamento (ou *offset*) do tempo marcado pelo nó em relação ao tempo global. As operações do protocolo MAC em um nó somente se iniciam após ele se considerar sincronizado. Isto ocorre após o recebimento de um pacote de sincronização e o cálculo de *offset* a partir das informações deste pacote. O *sink*, porém, é considerado sincronizado mediante comando vindo do servidor de aplicação. Ao se considerar sincronizado, o *sink* inicia a transmissão de pacotes de sincronização a seus nós filhos.

Os *network devices* por sua vez, ao estarem sincronizados, estimam o tempo global baseados nas informações que receberam e transmitem este tempo aos seus nós filhos. Quando as mensagens de sincronização atingem todos os nós da rede, todos os nós devem ser capazes de estimar o tempo global com diferença inferior ao tempo de guarda definido pelo protocolo MAC (definido como 60 milissegundos, conforme a seção 3.3.6.2).

Ao receber um pacote de sincronização, o nó marca o pacote com seu tempo local e subtrai deste valor uma constante referente ao tempo de processamento do pacote. Em uma transmissão ideal, onde a marcação do pacote recebido fosse instantânea, esta constante seria referente apenas ao tempo de propagação do pacote pelo meio de transmissão, e portanto, desprezível para as distâncias de transmissão possíveis com o rádio utilizado. Contudo, antes do pacote ser retirado do *buffer* do rádio e marcado com o tempo local do nó, há um período de processamento, onde ocorre a verificação de integridade do pacote e sua transmissão do

buffer do rádio para o microcontrolador. Este tempo foi estimado experimentalmente e seu valor fixado em 20 milissegundos. A Figura 3.4 ilustra o funcionamento de um protocolo de sincronização como o desenvolvido, em um caso onde não há atraso de processamento do pacote. Na Figura 3.5 é ilustrado o funcionamento da marcação de pacotes para sincronização em uma situação real, onde o tempo de processamento deve ser considerado.

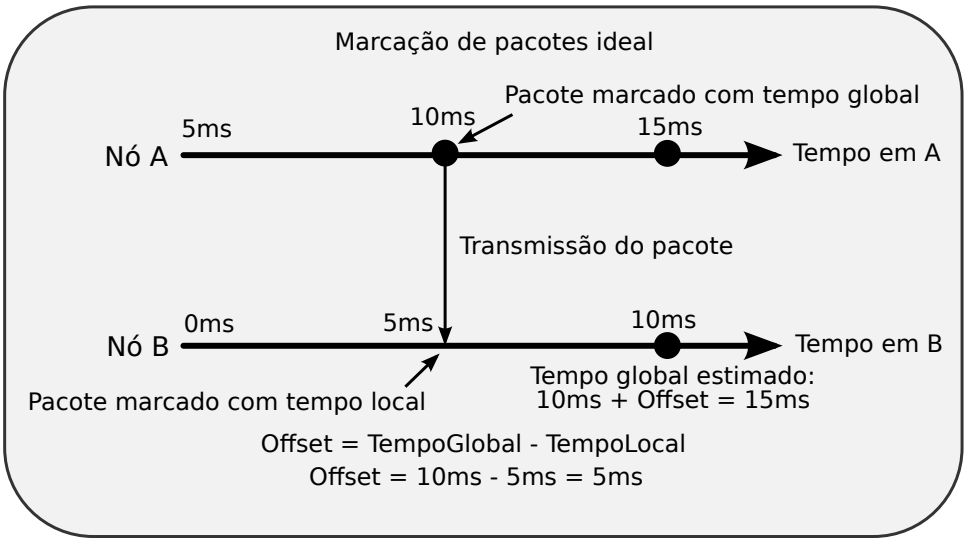

Figura 3.4: Marcação de pacotes ideal.

O pacote de sincronização é então passado a uma função que calcula o *offset* subtraindo do tempo global (recebido no pacote), o tempo local (marcado no pacote assim que recebido), como descrito pela Equação 2.2. Este *offset* é então utilizado para estimar o tempo global da rede de acordo com a Equação 3.1.

$$TempoGlobal = Offset + 1 \times TempoLocal \tag{3.1}$$

Para a implementação do protocolo de sincronização, optou-se por deixá-lo o mais independente possível do protocolo MAC, permitindo que diferentes métodos para cálculo de *drift* e *offset* sejam testados e avaliados, sem a necessidade de interferir no funcionamento do protocolo MAC.

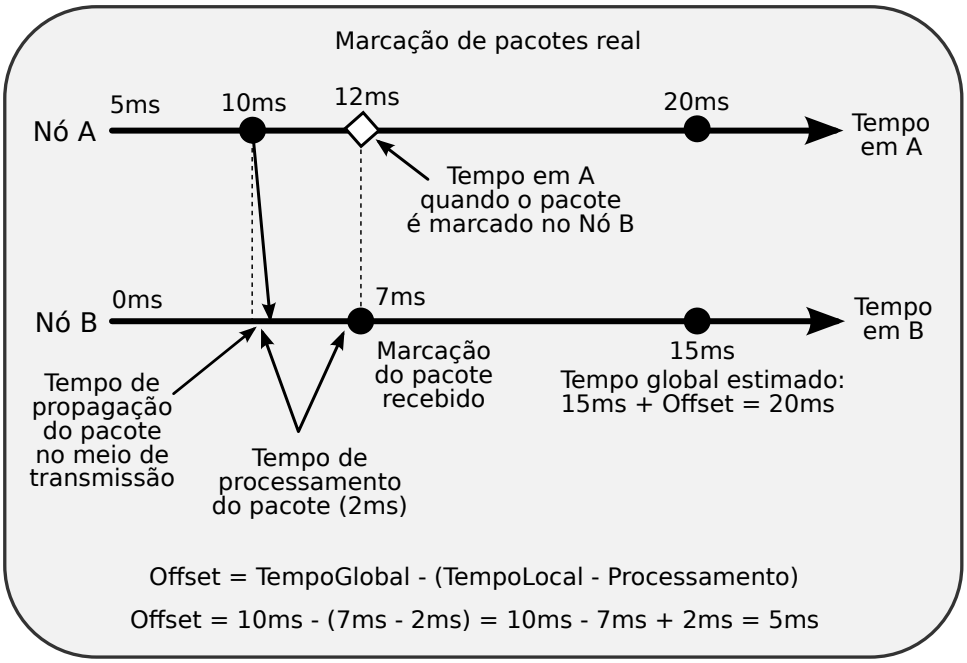

Figura 3.5: Marcação de pacotes real.

3.3.6 Protocolo MAC

O protocolo MAC desenvolvido na RSSF é baseado no protocolo TreeMAC (seção 2.7) com algumas modificações. Nesta seção são descritas as operações realizadas pelo protocolo MAC e os mecanismos utilizados para tal.

3.3.6.1 Ciclos

A cada ciclo de tempo, diferentes operações são feitas pelo protocolo MAC. Cada nó possui um contador que é incrementado a cada ciclo de tempo da rede. No final de cada ciclo, um tempo é reservado para o envio de mensagens de controle ou sincronização pelos *network devices*. O tipo da mensagem é definido pelo valor do contador de ciclos. Quando o valor deste contador é múltiplo de 2, são enviadas mensagens de sincronização. Quando é múltiplo de 5, são enviadas mensagens de controle, com o comando de requisição de configuração. Contudo a prioridade é para mensagens de sincronização, logo, quando o valor é múltiplo de 2 e de 5, são enviadas mensagens de sincronização. Se o valor não for um múltiplo de 2 nem de 5, são enviadas mensagens de controle comuns. A Figura 3.6 ilustra este comportamento entre três nós, ligados em tempos diferentes.

O *sink* é ligado, e quando seu contador de ciclos é igual a 1, uma mensagem de configuração é enviada ao seu nó filho (nó 1). Neste momento, o nó 1 acabou de ser ligado e seu contador de ciclos está em 0. No ciclo 2 do *sink*, uma mensagem de sincronização é enviada ao nó 1. Até esse momento, o início e o fim dos ciclos dos nós não estavam sincronizados. O nó 1 irá, então, fazer o ajuste do seu temporizador no próximo ciclo (ciclo 2), aguardando até que o tempo global seja um múltiplo do tamanho do ciclo. Após o ajuste, os temporizadores dos nós estão também sincronizados, e os ciclos estão sendo iniciados e terminados ao mesmo tempo. A mesma operação ocorre quando o nó 1 envia um pacote de sincronização ao nó 2.

No ciclo 5 do *sink*, uma mensagem de requisição de configuração é enviada ao nó 1, que responde com uma mensagem contendo suas informações. Esses dados são utilizados para que o *sink* saiba em quais fatias de tempo ele pode transmitir dados e comandos ao nó 1. Da mesma forma, o nó 1 no seu quinto ciclo envia uma mensagem de requisição de configuração ao nó 2.

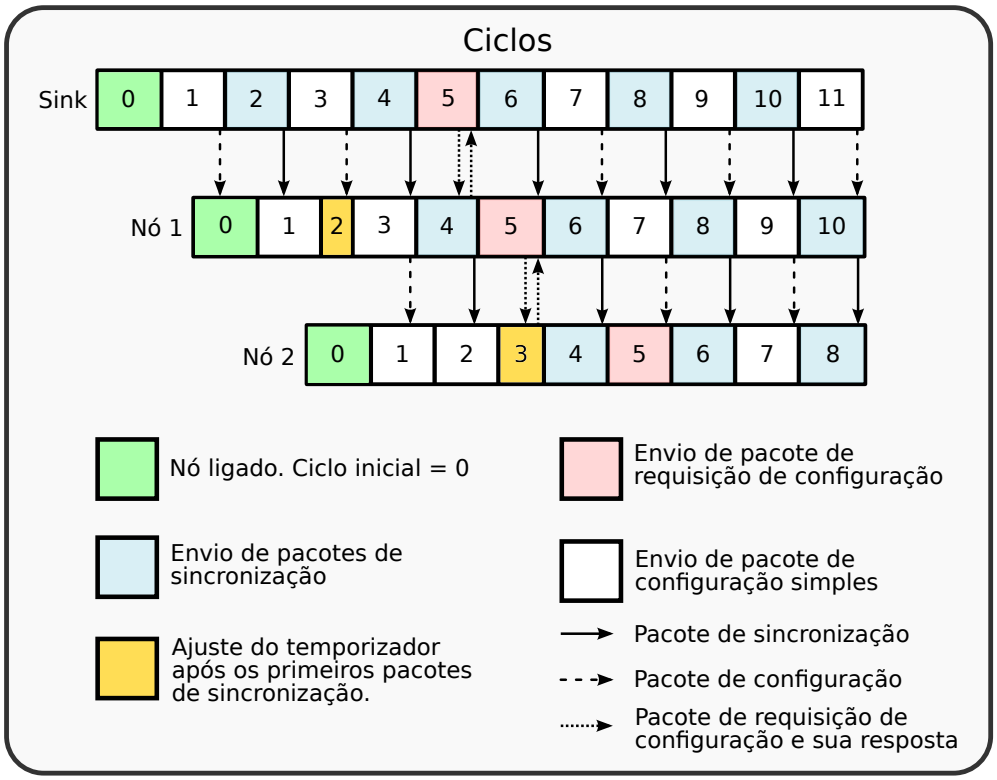

Figura 3.6: Operações na rede a cada ciclo.

3.3.6.2 *Frames* e *Slots*

A distribuição de *frames* é feita do mesmo modo que no protocolo TreeMAC. Cada nó recebe uma quantidade de *frames* proporcional às suas demandas. Cada *frame* é dividido em dois *slots*. Por padrão, a quantidade total de *frames* na rede será igual à quantidade de nós mais um, que é utilizado para ajuste de temporizadores. Os nós calculam seu *slot* de transmissão de acordo com sua distância em saltos

para o *sink*, utilizando o operador de módulo. Com a Equação 3.2 o nó obterá um valor 0 ou 1, que indicará em qual *slot* o nó está autorizado a iniciar transmissões. A Figura 3.7 ilustra como este esquema de definição de *slot* de transmissão reduz colisões em 2 saltos.

$$SlotTransmissao = DistanciaSink \bmod 2 \tag{3.2}$$

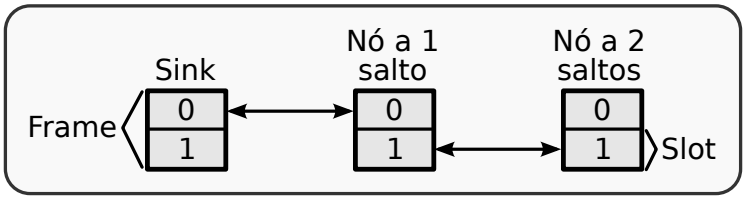

Figura 3.7: Redução de colisões em 2 saltos.

Os dois últimos *frames* do ciclo são reservados para tarefas de controle da rede. Em uma rede com N *frames*, indo de 0 a $N-1$, o *frame* $N-2$ é reservado para o envio de mensagens de controle ou sincronização, de acordo com o contador de ciclos. Durante este período, os *network devices* enviam a seus filhos as mensagens de configuração ou sincronização. Neste tempo o controle de acesso ao meio fica a cargo dos recursos do módulo de rádio, que realiza retransmissões automaticamente em caso de colisões.

Ao ser ligado, um nó inicia um temporizador que dispara um sinal de interrupção a cada 250 milissegundos. Uma função trata esta interrupção, apontando para o próximo conjunto *frame/slot*. Por exemplo, se o conjunto atual é *frame* 5 e *slot* 0, o protocolo passará a apontar para o *frame* 5 e *slot* 1. Na próxima interrupção passará a apontar para *frame* 6 e *slot* 0, e assim por diante, até chegar no último conjunto *frame/slot* e voltar a apontar para *frame* 0 e *slot* 0.

O *frame* $N-1$ é reservado para que os nós ajustem seus temporizadores. Neste momento, os nós desativam as interrupções por temporizador e aguardam até que o seu tempo global estimado seja um valor múltiplo do tamanho do ciclo.

Quando este tempo é atingido, todos iniciam novamente a operação. Este ajuste tem por objetivo aproximar o tempo em que as interrupções são disparadas em todos os nós, fazendo com que as mudanças de *frame* e *slot* em cada nó sejam o mais próximas possível. O ajuste só é realizado após o nó receber pelo menos uma mensagem de sincronização e calcular o *offset* pra estimar o tempo global da rede. A Figura 3.6 ilustra a operação deste ajuste após os nós receberem os primeiros pacotes de sincronização.

Mesmo após estar sincronizado, o tempo global estimado pelos nós não é exato. Por isso, um tempo de espera deve ser aguardado dentro do *slot*, de modo a garantir que todos os nós estejam no *slot* e com as configurações corretas carregadas. Este tempo deve ser grande o suficiente para superar os erros de estimativa, mas não tão grandes que tomem parte expressiva do tempo de transmissão.

Na RSSF desenvolvida foi utilizado um tempo de espera conservador, tal que se permita a avaliação do funcionamento e estabilidade dos protocolos. O tempo de espera foi fixado em 60 milissegundos. Uma janela de transmissão de aproximadamente 130 milissegundos é o tempo designado dentro do *slot* para a transmissão de pacotes entre os nós. A Figura 3.8 mostra a divisão do tempo dentro do *slot*.

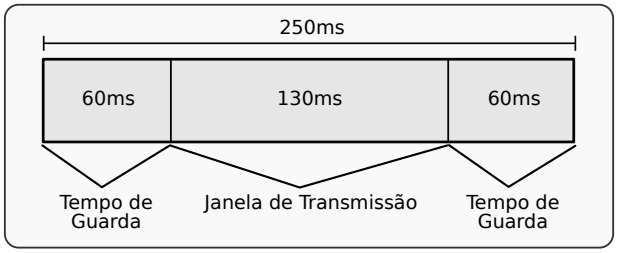

Figura 3.8: Divisão de tempo dentro do *slot*.

3.3.6.3 Canais

Aliado ao esquema de dois *slots*, e com o objetivo de melhorar a capacidade de transferência de dados, o protocolo MAC emprega o uso de múltiplos canais de

rádio, permitindo a ocorrência de comunicações simultâneas sem interferências de sinal. Com o uso de dois canais, é possível obter duas comunicações ao mesmo tempo, uma em cada canal. Desta forma, a cada *frame* podem ocorrer quatro comunicações, sendo duas simultâneas a cada *slot* e em profundidades diferentes da rede. A Figura 3.9 ilustra o funcionamento do esquema com dois *slots* e dois canais.

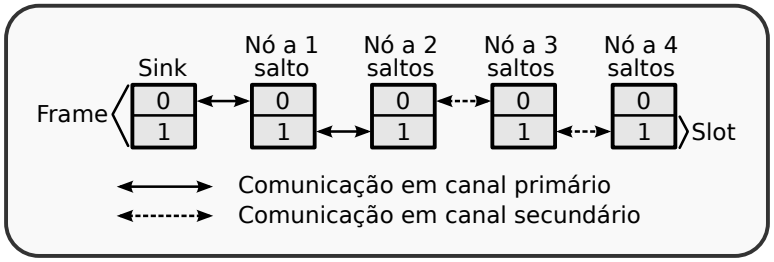

Figura 3.9: Operação da rede com dois *slots* e dois canais diferentes.

A configuração dos canais de operação das sub-redes é feita por pacotes de controle comuns. Para permitir que nós recentemente integrados à rede sejam configurados, no *frame* $N-2$ todos os nós operam no canal primário. Com isso, mesmo os nós recentemente iniciados podem receber pacotes de controle e se configurarem no canal correto.

3.3.6.4 *Buffer*

O protocolo MAC opera com um *buffer* capaz de armazenar até 20 pacotes. Este *buffer* é fundamental para a operação do protocolo, e tem a seguinte estrutura:

- **Pacote:** este pacote pode ter no máximo 32 bytes (que é o tamanho máximo permitido pelo rádio), e deve ser um pacote de controle ou de dados.
- **Endereço de Destino:** este endereço é calculado assim que o pacote é colocado no *buffer*. Ele representa o endereço de destino no próximo salto. Quando o valor do endereço do destinatário no pacote é menor que o valor do endereço do nó, o pacote é enviado ao nó pai. Se o endereço de destino do

pacote é maior que o do nó, então este pacote deve ir para a sub-rede. Contudo, o nó deve saber para qual dos filhos este pacote deve ser encaminhado. Este cálculo é feito pela operação descrita na Figura 3.10, tomando-se como base a topologia apresentada na Figura 3.3. Primeiramente a máscara de rede do nó atual (no caso 0xF0.00.00.00) é deslocada 4 bits à direita. Para preencher os 4 bits à esquerda que ficaram com valor 0, é feita a operação de **OU** lógico com o valor 0xF0.00.00.00, obtendo a máscara 0xFF.00.00.00. Com a máscara resultante, é feito o **E** lógico com o endereço de destino do pacote, obtendo o endereço para o qual o pacote deve ser encaminhado.

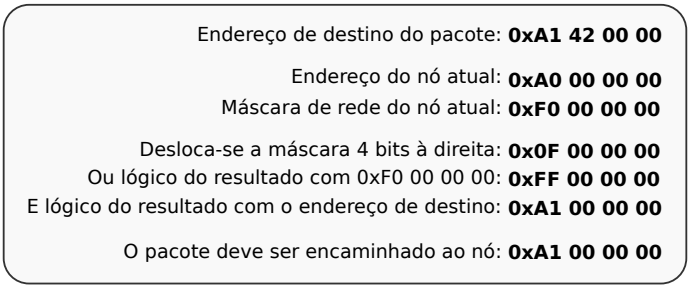

Figura 3.10: Cálculo do endereço para o qual o pacote deve ser encaminhado.

Sempre que um nó tem a oportunidade de transmitir para outro, é feita uma busca no *buffer* por pacotes destinados àquele nó. Ter o endereço já calculado e armazenado na estrutura do *buffer* evita que o cálculo seja feito a cada novo pacote buscado, reduzindo o tempo de processamento. Com o uso do *buffer*, as aplicações ficam encarregadas apenas de colocar os dados no *buffer*, evitando qualquer necessidade de interferência na operação do protocolo MAC.

O tamanho do *buffer* é limitado pela memória RAM do microcontrolador. Quanto maior o *buffer*, maior o consumo de memória RAM, e portanto, menos memória para a aplicação em execução. Caso a aplicação requeira uma quantidade maior de memória RAM, pode ser necessária a redução do tamanho do *buffer*.

3.3.6.5 Configuração dos nós

Algumas configurações dos nós são armazenadas na EEPROM e carregadas durante a inicialização. Nesta memória, os dados ficam armazenados mesmo após o equipamento ser desligado. Assim, caso o nó seja reiniciado devido a algum tipo de falha, sua integração à rede é simplificada, pois o nó já possui sua configuração básica armazenada localmente.

A Tabela 3.5 especifica as configurações armazenadas na memória EEPROM. Na primeira coluna, são especificados os endereços da memória onde estão armazenados os dados. Em cada endereço é possível armazenar um valor de 8 bits (por exemplo, números inteiros de 0 a 255). A segunda coluna especifica as variáveis que devem receber tais valores, e a terceira coluna, observações sobre cada configuração.

Tabela 3.5: Configurações na EEPROM dos nós.

Endereço	**Configuração**	Observação
0 a 3	*node_address*	Endereço do nó. Byte mais significativo no endereço 3, byte menos significativo no endereço 0.
4	*device_type*	Tipo de dispositivo. *FIELD_DEVICE = 1*, *NETWORK_DEVICE = 2*.
5	*frame_count*	Quantidade de *frames* na rede.
6	*lower_frame*	*Frame* inferior designado ao nó.
7	*upper_frame*	*Frame* superior designado ao nó.
8	*child_number*	Número de filhos do nó (filhos a 1 salto).

3.3.6.6 Operação do protocolo

O protocolo opera baseado em interrupções. Interrupções são sinais capazes de interromper a execução do programa em andamento, executar funções reponsáveis pelo tratamento destes sinais, e continuar a execução novamente.

No microcontrolador utilizado existem três interrupções internas, baseadas em temporizadores, e duas interrupções externas, onde o sinal é obtido em um dos pinos. A interrupção interna 0 é utilizada pela biblioteca da plataforma Arduino

para a contagem de tempo desde que o equipamento foi ligado. A cada período, um contador é incrementado, com intervalos que se aproximam do tempo real. Com base nestes contadores operam funções essenciais como por exemplo:

- ***delay(x)*:** faz com que o programa espere "x"milissegundos sem fazer nenhuma operação.
- ***delayMicroseconds(x)*:** semelhante à anterior, mas em microsegundos.
- ***millis()*:** retorna o tempo decorrido, em milissegundos, desde que o microcontrolador foi ligado (ou resetado).
- ***micros()*:** semelhante à millis() mas retorna o tempo em microsegundos.

A interrupção interna 0, portanto, não deve ser utilizada para outros fins, pois pode afetar a operação de outras funções das bibliotecas utilizadas. Com isso, ficam livres para uso as interrupções internas 1 e 2, que têm objetivos bem definidos para uso no protocolo.

A interrupção interna 1 é utilizada para programar as mudanças de *slot*. Na RSSF desenvolvida, um sinal é disparado a cada 250 milissegundos, indicando que o *slot* deve ser trocado. Uma função trata este sinal e atualiza as variáveis de *slot* e *frame* para os próximos valores.

Por fim, a interrupção interna 2 controla o tamanho da janela de transmissão. Após o tempo de espera dentro do *slot*, uma variável é inicializada com o valor inteiro 10, e a cada 13 milissegundos este valor é decrementado. Quando este valor chega a 0 indica que está encerrado o período de transmissão, e o nó aguarda um sinal da interrupção 1 para mudar de *slot*.

A interrupção externa é utilizada pelo módulo de rádio. Um sinal é disparado pelo pino **IRQ** do módulo quando chegam novos pacotes, quando há falha de transmissão e quando há uma transmissão bem sucedida. No protocolo desenvolvido, é utilizada a interrupção do rádio somente quando um pacote é recebido.

Quando isso ocorre, uma função é responsável por retirar os dados do *buffer* e processá-los.

A Figura 3.11 mostra o diagrama de estados da operação do protocolo. Ao ser ligado, o nó carrega as configurações da EEPROM, inicia o temporizador e aguarda interrupções. Cada tipo de interrupção é tratado por uma função específica. Quando um pacote é recebido, uma interrupção é causada pelo transceptor, e uma função para o processamento da mensagem é executada. O diagrama de estados da Figura 3.12 descreve a operação do protocolo para o processamento dessas mensagens. As interrupções disparadas por temporizador são processadas de acordo com o *frame*, o tipo de dispositivo e o estado de sincronização do nó. O diagrama de estados da Figura 3.13 descreve a operação do protocolo quando um *network device* entra no estado de envio de dados para algum nó filho. Por fim, o diagrama da Figura 3.14 descreve a operação do protocolo para o envio de pacotes de controle, que ocorre no *frame* $N-2$.

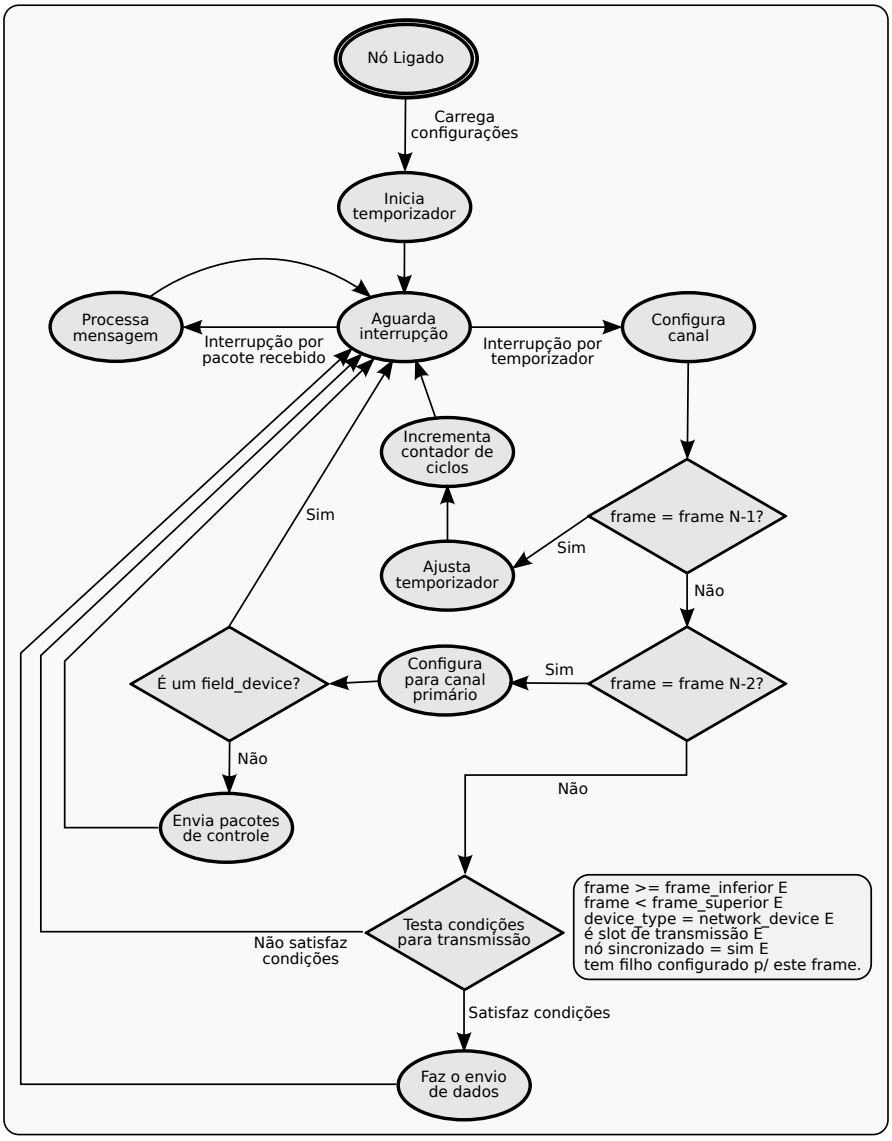

Figura 3.11: Diagrama de estados da operação do protocolo MAC.

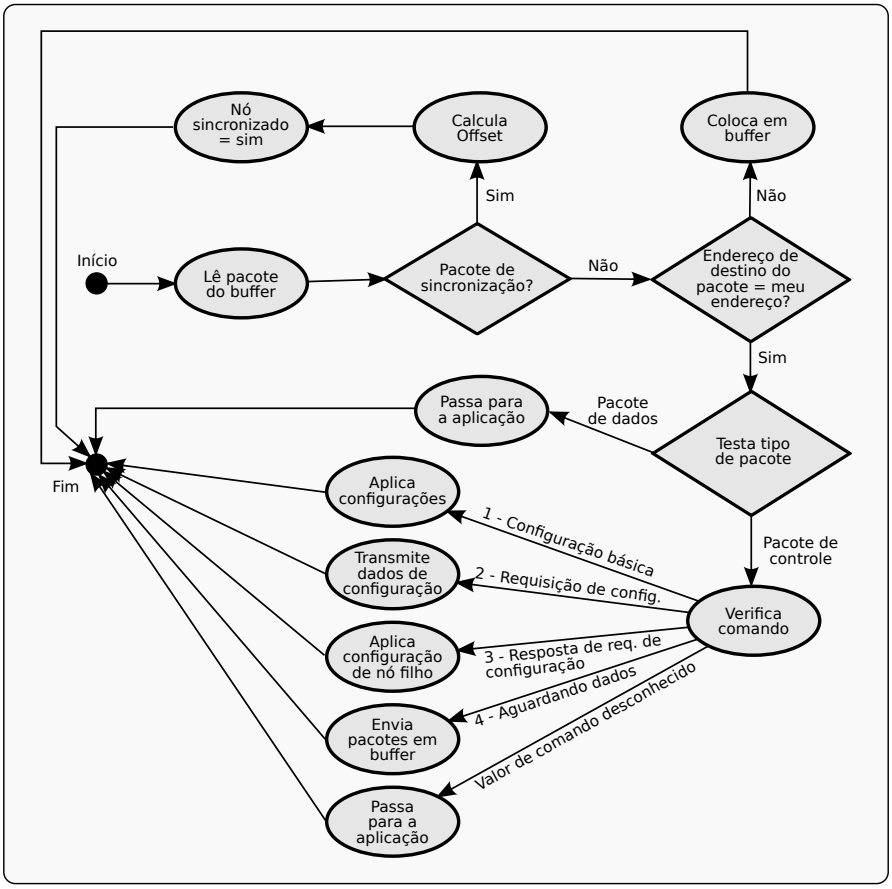

Figura 3.12: Diagrama de estados de processamento de mensagens.

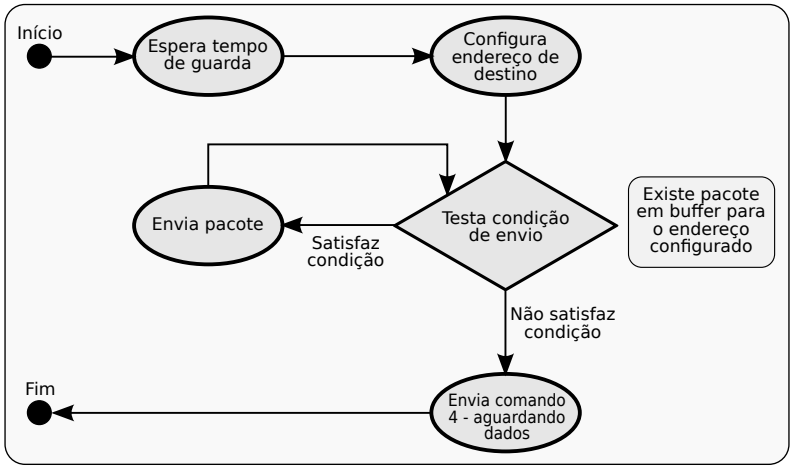

Figura 3.13: Diagrama de estados do envio de dados.

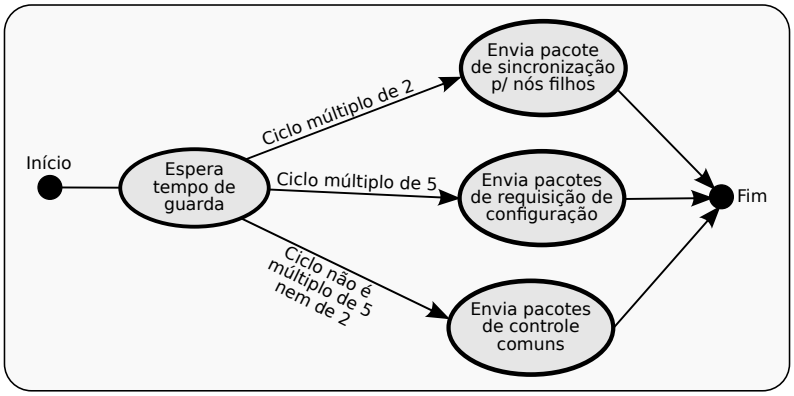

Figura 3.14: Diagrama de estados do envio de pacotes de controle.

3.3.7 Aplicação

A aplicação desenvolvida tem por objetivo avaliar a operação da rede e apresentar os dados recebidos pelos nós. Por meio da porta USB da placa Arduino Uno, os dados são transmitidos de forma serial. Uma *thread* é responsável por ler estes dados, armazená-los em um arquivo de *log* e apresentá-los em tela. Outra *thread* fica responsável por receber comandos via terminal e transmití-los ao nó *sink*.

3.4 Cenário

Os testes da RSSF desenvolvida foram realizados utilizando-se 10 nós. A topologia e o endereçamento utilizado são os mesmos descritos na Figura 3.3. A Figura 3.15 ilustra a distribuição de *frames* entre os nós, bem como os *slots* em que ocorrem as transmissões.

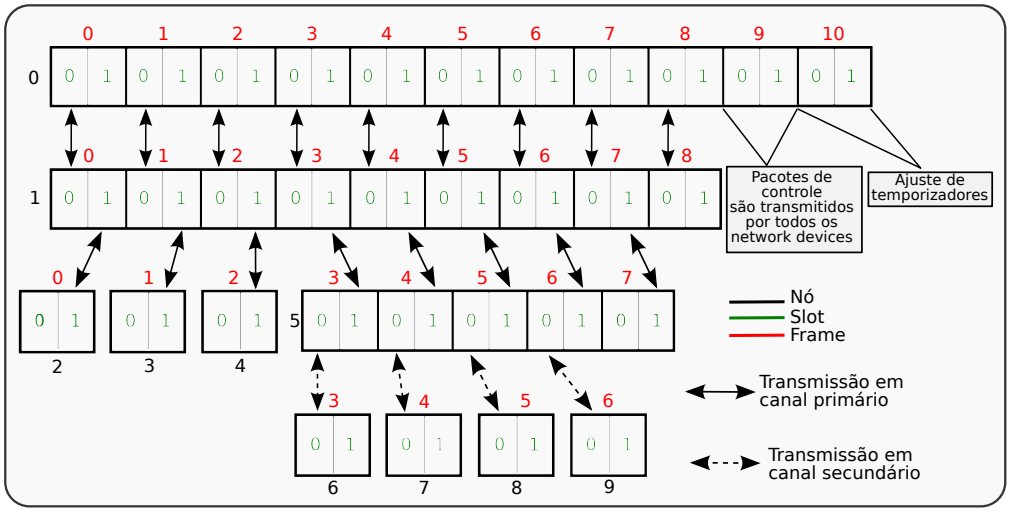

Figura 3.15: Distribuição de *frames* para a topologia testada.

3.5 Métricas

Com o objetivo de avaliar o desempenho dos equipamentos utilizados e os protocolos desenvolvidos, foram utilizadas as seguintes métricas:

- **Capacidade de transmissão de dados dentro de um *slot*:** quantidade de dados que podem ser trocados entre nós durante a janela de transmissão de um *slot*. Com isso, pode-se avaliar o desempenho do módulo de rádio e o *overhead* dos pacotes de dados.

- **Precisão do protocolo de sincronização de tempo:** analisou-se o erro de estimativa no cálculo do tempo global em diferentes profundidades na rede. Além disso, a quantidade de pacotes utilizados para o cálculo da média do *offset* foi avaliada, levando-se em consideração a precisão da estimativa, o uso de memória RAM, processamento e a quantidade necessária de pacotes de sincronização enviados.

- **Tempo de transmissão de pacotes da origem até o destino:** tempo que um pacote leva para ser transmitido de um *field device* até o *sink*, representando a transmissão de dados coletados, e o tempo que um pacote de controle leva para ir do *sink* até um *field device*, representando o envio de comandos de controle por um servidor.

4 RESULTADOS E DISCUSSÃO

Esta seção apresenta os testes realizados na RSSF desenvolvida, os resultados obtidos e avaliação destes resultados. A avaliação é feita com base nas métricas definidas na Seção 3.5

4.1 Teste de taxa de transmissão de dados

O teste de taxa de transmissão tem por objetivo avaliar a capacidade de transferência de dados entre dois nós da rede. Com isso, é possível avaliar o desempenho do transceptor, bem como a eficiência do *software* desenvolvido. Além disso, é analisado o *overhead* gerado pelos cabeçalhos das mensagens. Este *overhead* é a sobrecarga na comunicação gerada por dados de controle dos pacotes. Com um tamanho de pacote limitado, quanto menor o espaço ocupado por cabeçalhos, mais espaço sobra para dados úteis (dados efetivamente da aplicação), e portanto, maior a capacidade de transmissão efetiva. A Tabela 4.1 detalha a distribuição de espaço em um quadro gerado pelo nRF24l01+, a partir de um pacote de dados do protocolo MAC (descrito na Tabela 3.4). Na primeira coluna (Quadro (nRF24l01+)), é especificada a distribuição de espaço no pacote gerado pelo rádio, com um campo de dados de 256 bits. Neste campo é colocado o pacote do protocolo, descrito na coluna seguinte. Por fim, a última coluna especifica o quadro resultante, com o pacote do protocolo posicionado dentro do quadro do rádio. Como o protocolo necessita de cabeçalhos adicionais, o quadro resultante possui um total de 153 bits de cabeçalho. Os 168 bits restantes carregam os dados da aplicação.

Tabela 4.1: Distribuição de espaço em um quadro.

	Quadro (nRF24l01+)		Pacote do protocolo		Quadro Resultante	
Cabeçalho	65 bits	20,25%	88 bits	34,37%	153 bits	47,66%
Dados	256 bits	79,75%	168 bits	65,63%	168 bits	52,33%
Total	321 bits	100%	256 bits	100%	321 bits	100%

Os testes foram realizados colocando-se pacotes de dados no *buffer* de um nó filho, que por sua vez transmite o mesmo pacote repetidamente durante a janela de transmissão, para seu nó pai. A comunicação inversa também é avaliada.

Para todos os testes foram feitas 100 repetições, e para a análise foi utilizada a média aritmética de pacotes transmitidos, com nível de confiança para intervalo de confiança de 95%. Em todos os casos, o erro foi inferior a 0,01% do valor obtido.

A Tabela 4.2 apresenta os resultados obtidos. As transmissões no sentido de um *network device* para um *field device* são sutilmente superiores devido ao modo como o protocolo prioriza as transmissões. Um nó sempre aguarda um pacote de controle de seu nó pai informando que pode transmitir seus dados. Um nó pai, no entanto, não precisa aguardar mensagens de controle e envia seus dados em *buffer* imediatamente ao nó filho. Este comportamento tem o intuito de priorizar a transmissão de possíveis mensagens de controle vindas do servidor, e com destino a nós instalados em equipamentos industriais. Estas mensagens podem ser críticas para o controle de processos em andamento ou mesmo em casos de emergência.

Tabela 4.2: Taxa de transmissão máxima obtida entre dois nós.

	Nó filho → Nó pai	Nó pai → Nó filho
Pacotes enviados/frame (média)	$138,2\ (\pm 0,14)$	$138,6\ (\pm 0,09)$
Tempo médio de transmissão	$163,90ms\ (\pm 0,161ms)$	$163,84ms\ (\pm 0,108ms)$
Velocidade média	$270.680bps\ (\pm 170bps)$	$271.634bps\ (\pm 11bps)$
Velocidade de transmissão de dados da aplicação	$141.664bps\ (\pm 89bps)$	$142.163bps\ (\pm 5bps)$
Total transmitido/frame	$5,41KB\ (\pm 0,005KB)$	$5,43KB\ (\pm 0,003KB)$
Total de dados/frame	$2,83KB\ (\pm 0,002KB)\ (52,31\%)$	$2,84KB\ (\pm 0,001KB)\ (52,31\%)$
Overhead	$2,58KB\ (\pm 0,002KB)\ (47,69\%)$	$2,59KB\ (\pm 0,001KB)\ (47,69\%)$

Como durante um *slot* de tempo apenas um par de nós se comunica, a quantidade de pacotes que um nó efetivamente pode receber fica limitada ao tamanho do *buffer* de dados (funcionamento do *buffer* descrito na Seção 3.3.6.4). Isto se deve ao fato de que, no melhor caso, somente no *slot* seguinte os pacotes recebi-

dos poderão ser encaminhados, e assim liberar espaço para receber mais pacotes. Com o intuito de avaliar a capacidade de vazão real da rede, considerando nós com *buffer* para 20 pacotes, a quantidade de pacotes transmitidas por *frame* foi limitada a 20. Por essa razão, alguns dados obtidos não obtiveram variações durante os testes, como a quantidade de pacotes enviados, total transmitido por *frame*, total de dados por *frame* e o *overhead*.

Tabela 4.3: Taxa de transmissão entre dois nós com limitação de *buffer*.

	Nó filho → Nó pai	Nó pai → Nó filho
Pacotes enviados/frame	20	20
Tempo médio de transmissão	$23,72ms$ $(\pm\ 0,043ms)$	$23,65ms$ $(\pm\ 0,005ms)$
Velocidade	$270.608bps$ $(\pm\ 490bps)$	$271.411bps$ $(\pm\ 59bps)$
Velocidade de transmissão de dados da aplicação	$141.544bps$ $(\pm\ 256bps)$	$142.047bps$ $(\pm\ 31bps)$
Total transmitido/frame	$802,5B$	$802,5B$
Total de dados/frame	$420B$ $(52,33\%)$	$420B$ $(52,33\%)$
Overhead	$382,5B$ $(47,66\%)$	$382,5B$ $(47,66\%)$

Com base nos resultados apresentados na Tabela 4.3, a Tabela 4.4 mostra a quantidade de leituras que podem ser transmitidas por *frame*, de acordo com o tamanho das variáveis utilizadas para armazenar estas leituras. São considerados os principais tipos de variáveis utilizados para a coleta de dados de monitoramento.

Tabela 4.4: Quantidade de leituras transmitidas por tipo de variável.

Tipo de variável	Tamanho do tipo em Bytes	Quantidade de leituras transmitidas por frame
char uint8_t int8_t bool	1	420
int32_t uint32_t float void	4	105
long long int int64_t uint64_t double	8	52

Estes resultados mostram que para o cenário de testes, em uma rede com 10 nós, a capacidade máxima de transmissão de dados de monitoramento por ciclo é de 468 leituras de dados de 8 Bytes, 945 leituras de dados de 4 Bytes ou 3.780 leituras de dados de 1 Byte.

4.2 Precisão do protocolo de sincronização

A precisão do protocolo de sincronização utilizado é crucial para a definição do tamanho do tempo de espera dentro de um *slot*. Com protocolos mais precisos, o tempo de espera pode ser reduzido, permitindo ainda, a redução do tamanho do *slot* e melhorando o tempo de resposta da rede.

Para a execução dos testes do protocolo de sincronização foram utilizados dois nós. O tempo do nó pai é considerado o tempo global, e o nó filho deve estimar este tempo basedo nas mensagens de sincronização. Após as mensagens iniciais de configuração e de sincronização, foram transmitidas mensagens de controle de um nó a outro. Ao transmitir uma mensagem, um nó imprime seu tempo global estimado. O outro nó por sua vez, também imprime o tempo global estimado assim que recebe a mensagem. Desta forma, foi possível avaliar a diferença do tempo global no nó pai para a estimativa realizada pelo nó filho. Para cada caso foram coletadas 400 amostras, os tempos descritos utilizam a média dos valores obtidos, e a análise foi realizada com nível de confiança para intervalo de confiança de 95%.

Os testes foram feitos avaliando duas diferentes abordagens para o cálculo de *offset*. A primeira, utiliza apenas o *offset* do pacote de sincronização recebido mais recentemente. A segunda, utiliza a média dos N últimos pacotes de sincronização recebidos. Esta abordagem, com a média de N pacotes é utilizada por Maróti et al. (2004) para a estimativa de *offset* no protocolo *Flooding Time Synchronization Protocol*.

No primeiro teste foi avaliado o erro de estimativa do tempo global utilizando diferentes quantidades de pacotes para cálculo de *offset*. Com o uso de apenas um pacote, o *offset* estimado é igual ao do pacote recebido mais recentemente. A partir de dois pacotes, o *offset* é estimado pela média dos *offsets* calculados a partir dos *N* últimos pacotes recebidos. Os resultados mostrados no gráfico da Figura 4.1 indicam que o uso de apenas um pacote para o cálculo de *offset* gerou o menor erro de estimativa, e o aumento de pacotes não resultou em redução da estimativa de erro.

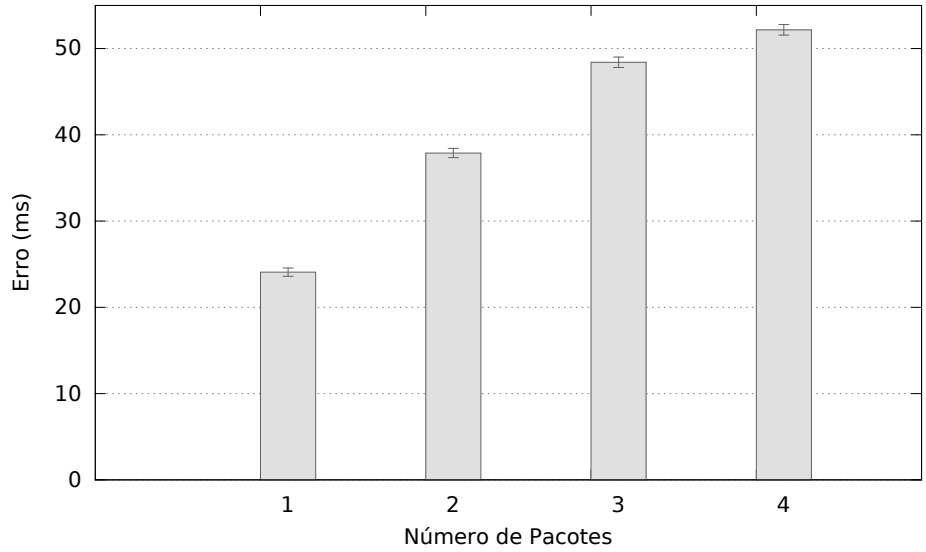

Figura 4.1: Erro na estimativa de tempo global por quantidade de pacotes.

Após definir que seria utilizado apenas o *offset* do último pacote de sincronização recebido, foram realizados testes para avaliar o valor da constante de processamento a ser subtraída do tempo local marcado no pacote. Foram realizados testes com diferentes valores, sendo observado que uma constante de 20 milissegundos apresentou o menor erro médio de estimativa. O gráfico da Figura 4.2 mostra a variação no erro médio de estimativa em relação ao tamanho da constante de processamento utilizada.

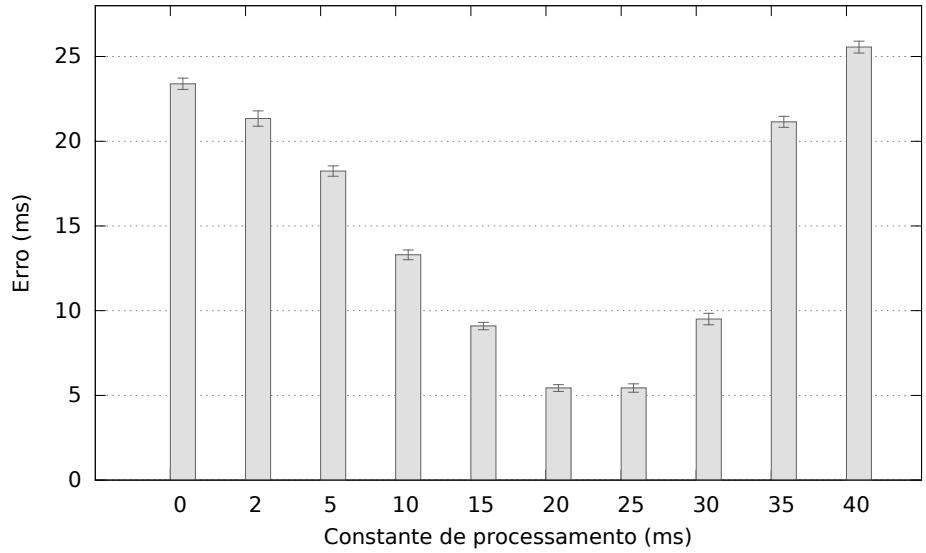

Figura 4.2: Erro na estimativa de tempo global por constante de processamento.

Com base nestes resultados, o valor da constante de processamento utilizada pelo protocolo de sincronização foi definido como 20 milissegundos, e o *offset* utilizado é aquele calculado a partir do último pacote recebido. Além do erro de estimativa mostrado no gráfico da Figura 4.2, a eficácia do protocolo de sincronização foi atestada ao se observar os *frames* e *slots* que cada pacote foi transmitido, e comparar com os *frames* e *slot* em que foi recebido pelo nó vizinho. Esta análise mostrou que nenhum pacote transmitido (após a sincronização da rede), em um determinado par *frame*/*slot* chegou ao outro nó em um par *frame*/*slot* diferente.

4.3 Tempo de transmissão de um pacote através da rede

Para avaliar a viabilidade do protocolo desenvolvido nas aplicações propostas, o tempo levado para a transmissão de dados entre dois nós da rede, a diferentes distâncias (em saltos) foi analisada. A organização da rede para os testes é a mesma apresentada na Figura 3.3.

Este teste foi realizado colocando-se, em um pacote de dados, um campo de contador. Ao ser colocado em *buffer*, o tempo local do nó era marcado. Quando o pacote era retirado e transmitido, o tempo inicial marcado era subtraído do tempo local atual, e seu valor resultante somado ao contador do pacote. Os pacotes foram colocados em *buffer* em tempos aleatórios. Desta forma, foi estimado o tempo médio levado desde que um pacote é colocado no *buffer*, até sua chegada ao nó de destino.

A transmissão foi testada a partir do *sink* com destino aos nós inferiores da rede, representando mensagens de controle partidas do servidor. Foi testada também a transmissão partindo dos demais nós da rede com direção ao *sink*, representando mensagens de monitoramento coletadas pelos nós. Para cada teste foram enviados 100 pacotes, e seu tempo de transferência da origem até o destino foi analisado. As análises foram realizadas com nível de confiança para intervalo de confiança de 95%. A Tabela 4.5 descreve os pares de nós que foram testados, bem como o tempo mínimo, máximo e médio obtido em cada caso.

Tabela 4.5: Taxa de transmissão entre dois nós com limitação de *buffer*.

Origem → Destino	Saltos	Tempo mínimo	Tempo máximo	Tempo médio
Nó 0 → Nó 1	1	63,48 ms	1.491,71 ms	466,91 ms
Nó 0 → Nó 2	2	368,12 ms	6.212,91 ms	3.237,47 ms
Nó 0 → Nó 5	2	415,01 ms	4.726,51 ms	1.934,10 ms
Nó 0 → Nó 6	3	4.424,15 ms	14.675,80 ms	8.672,47 ms
Nó 1 → Nó 0	1	5,98 ms	1.448,78 ms	406,55 ms
Nó 2 → Nó 0	2	307,70 ms	11.160,71 ms	4.279,85 ms
Nó 5 → Nó 0	2	267,62 ms	7.209,32 ms	2.307,90 ms
Nó 6 → Nó 0	3	822,69 ms	77.849,66 ms	11.992,34 ms

Analisando-se os resultados na Tabela 4.5, conclui-se que o tempo médio é maior para nós que possuem menos *frames* para transmissão. Este comportamento é o previsto de acordo com a distribuição de *frames* feita pelo protocolo, que prioriza nós que possuem mais filhos. Estes nós, possuem mais oportunidades de comunicação, e portanto, precisam ficar com os dados armazenados em *buffer*

por menos tempo. Os nós que possuem apenas um *frame*, no pior caso, devem aguardar todo um ciclo para ter uma nova oportunidade de comunicação. Além deste tempo, ciclos adicionais podem ser necessários para que os pacotes transmitidos por estes nós cheguem até o destino, pois estão limitados às oportunidades de transmissão de seus nós superiores.

Além disso, vale notar que os nós 2 e 5 estão a dois saltos do *sink*, no entanto, possuem características diferentes. O nó 2 é um *field device*, que não encaminha pacotes de nenhum outro dispositivo, e tem apenas um *frame* de comunicação. O nó 5 por sua vez, é um *network device*, responsável por encaminhar pacotes de outros quatro nós, e tem cinco *frames* de comunicação. Por esta característica, o tempo de transmissão dos pacotes provenientes do nó 5 foi consideravelmente inferior ao dos pacotes provenientes do nó 2.

O gráfico da Figura 4.3 apresenta o tempo médio de transmissão de pacotes entre os nós. As comunicações em *upstream* são aquelas partindo dos nós da rede, em direção ao *sink*. As comunicações em *downstream* são aquelas partindo do *sink* e com direção aos outros nós da rede. O eixo x do gráfico indica qual nó se comunica com o *sink*. O resultado indica um tempo de transmissão médio em *downstream* inferior, indicando que o protocolo satisfaz o objetivo de priorizar as mensagens de controle.

O gráfico da Figura 4.4 apresenta a diferença média de tempo de transmissão de pacotes dos nós 2 e 5, que estão ambos a dois saltos de distância do *sink*, mas possuem configurações diferentes. O nó 2 tem apenas um *frame* de transmissão, enquanto o nó 5 possui cinco *frames*.

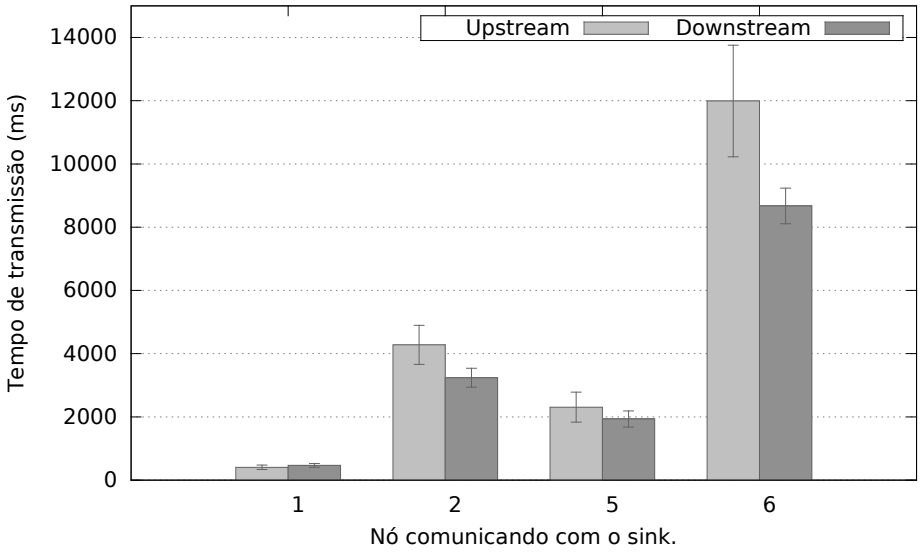

Figura 4.3: Tempo de transmissão entre o sink e os nós.

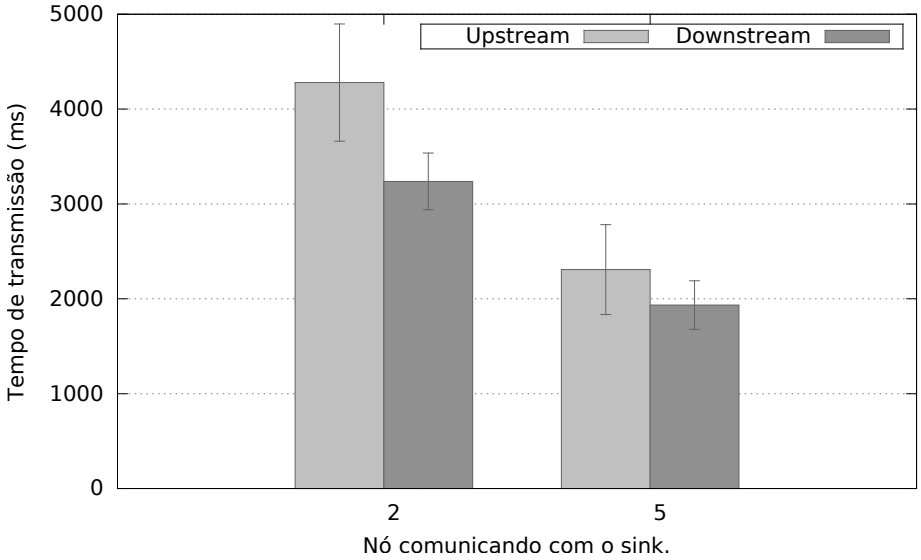

Figura 4.4: Tempo de transmissão entre o sink, e os nós 2 e 5.

5 CONCLUSÃO

A RSSF desenvolvida e apresentada neste trabalho oferece uma alternativa para integração de sistemas já existentes e o desenvolvimento de novas soluções. Além disso, o *hardware* de baixo custo e de montagem simples permite a sua utilização em futuras pesquisas na área de RSSF.

A escolha de um protocolo baseado em TDMA para o controle de acesso ao meio, que oferece uma melhor previsibilidade da operação da rede, se mostrou uma solução particularmente desafiadora. A demanda por sincronização precisa de tempo para este tipo de protocolo, aliada às limitações do *hardware* utilizado, como o tamanho da memória de programa (de 32KB) e a quantidade de memória RAM (de 2KB), exigiu um desenvolvimento cauteloso dos algoritmos, assim como a definição de limites de tempo mais moderados para sua operação confiável.

Os resultados dos testes mostram que a capacidade de vazão de dados da rede, bem como as taxas de transmissão obtidas com o rádio utilizado, permitem a transmissão de uma quantidade considerável de dados de monitoramento, mesmo com o *overhead* gerado pelos cabeçalhos dos pacotes. Os tempos de propagação dos pacotes, no entanto, se mostraram altos em algumas situações de operação. Isto se deve à definição de *slots* de tempo mais longos, com o intuito de avaliar as características funcionais do protocolo, as quais servem de base para novos testes visando a melhoria do desempenho.

6 TRABALHOS FUTUROS

Com o objetivo de aplicar a RSSF desenvolvida no monitoramento de processos industriais, de forma que a rede seja auto-organizável e robusta, são sugeridas as seguintes atividades para desenvolvimento futuro:

- **Desenvolvimento da aplicação do servidor:** esta aplicação é responsável por receber os dados da rede, armazená-los em um banco de dados e exibí-los ao operador do sistema. A aplicação deve ainda oferecer funções de controle de processos, realizado por meio de comandos enviados do servidor até os equipamentos em operação.

- **Implementação e testes de um protocolo de configuração:** o protocolo atual é capaz de configurar automaticamente diversos parâmetros da rede e dos nós. Com o intuito de tornar esta configuração totalmente automática, mecanismos adicionais devem ser desenvolvidos.

- **Desenvolvimento do protocolo de comunicação entre nós da rede e outros equipamentos de monitoramento:** para permitir o uso de nós da rede desenvolvida como adaptador para equipamentos já empregados em monitoramento industrial, um protocolo de comunicação entre estes dispositivos deve ser elaborado e implementado. Com isso, um padrão de comunicação é definido e novos sistemas de monitoramento desenvolvidos devem implementá-lo para permitir a integração com os nós da RSSF.

- **Avaliação do uso de algoritmos de criptografia e sua implementação:** ataques a redes de computadores são comuns em todas as áreas. Em RSSF industriais, o acesso não autorizado a dados de monitoramento pode oferecer dados importantes sobre o funcionamento dos equipamentos. Além disso, pessoas mal intencionadas podem interferir no controle de processos e causar danos catastróficos. Por isso, mecanismos de segurança devem

ser empregados. Com as limitações do *hardware* utilizado, os algoritmos de criptografia devem ser avaliados com cautela, de modo que interfiram o mínimo possível nas operações de monitoramento e controle, enquanto oferecem a segurança necessária.

- **Testes com diferentes protocolos de sincronização:** o alto desacoplamento entre o protocolo MAC e o protocolo de sincronização utilizado permite que diferentes formas sincronização sejam testadas e avaliadas. Com protocolos que oferecem menores erros de estimativa, pode-se reduzir o tempo de espera dentro dos *slots*, possibilitando uma redução global no tempo de resposta da rede, e assim, viabilizando seu uso em aplicações críticas.

REFERÊNCIAS BIBLIOGRÁFICAS

Akyildiz et al. (2002) AKYILDIZ, I.F.; W, SU; Y, SANKARASUBRAMANIAM; E, CAYIRCI. Wireless sensor networks: a survey. *Computer Networks*, v. 38, n. 4, p. 393 – 422, 2002. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128601003024>.

Arduino, 2014 ARDUINO. *Arduino*. 2014. <http://arduino.cc/>. Acessado em 5-11-2014.

BeagleBoard, 2014 BEAGLEBOARD. *BeagleBoard*. 2014. <http://beagleboard.org/>. Acessado em 5-11-2014.

Bruno, Conti e Gregori (2005) BRUNO, R.; CONTI, M.; GREGORI, E. Mesh networks: commodity multihop ad hoc networks. *Communications Magazine, IEEE*, v. 43, n. 3, p. 123–131, March 2005. ISSN 0163-6804.

Gumstix, 2014 GUMSTIX, I. *Gumstix*. 2014. <https://www.gumstix.com/>. Acessado em 5-11-2014.

Gungor e Hancke (2009) GUNGOR, V.; HANCKE, G. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on*, v. 56, n. 10, p. 4258–4265, 2009. ISSN 0278-0046.

Gungor, Lu e Hancke 2010 GUNGOR, V.; LU, B.; HANCKE, G. Opportunities and challenges of wireless sensor networks in smart grid. *Industrial Electronics, IEEE Transactions on*, v. 57, n. 10, p. 3557–3564, 2010. ISSN 0278-0046.

HART, 2014 HART. *Hart Communication Foundation*. 2014. <http://www.hartcomm.org/>. Acessado em 5-11-2014.

JJGarcia.tsc 2011 JJGARCIA.TSC. *Diagram of CSMA-CA algorithm with RTS/CTS exchange as provided by 802.11 standard*. 2011. <https://commons.wikimedia.org/wiki/File:Csma_ca.svg>. Acessado em 5-11-2014.

Jung (2004) JUNG, C. F. *Metodologia para pesquisa e desenvolvimento: aplicada a novas tecnologias, produtos e processos*. [S.l.]: Axcel Books, 2004.

Karl e Willig (2005) KARL, H.; WILLIG, A. *Protocols and Architectures for Wireless Sensor Networks*. [S.l.]: John Wiley & Sons, 2005. ISBN 0470095105.

Kim et al. 2008 KIM, A.N.; HEKLAND, F.; PETERSEN, S.; DOYLE, P. When hart goes wireless: Understanding and implementing the wirelesshart standard. In: *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*. [S.l.: s.n.], 2008. p. 899–907.

Lennvall, Svensson e Hekland 2008 LENNVALL, T.; SVENSSON, S.; HEKLAND, F. A comparison of wirelesshart and zigbee for industrial applications. In: *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on*. [S.l.: s.n.], 2008. p. 85–88.

Low, Win e Er 2005 LOW, K.-S.; WIN, W.; ER, M.-J. Wireless sensor networks for industrial environments. In: *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*. [S.l.: s.n.], 2005. v. 2, p. 271–276.

Maróti et al. (2004) MARÓTI, MIKLÓS; KUSY, BRANISLAV; SIMON, GYULA; LÉDECZI, ÁKOS The flooding time synchronization protocol. In: *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2004. (SenSys '04), p. 39–49. ISBN 1-58113-879-2. Disponível em: <http://doi.acm.org/10.1145/1031495.1031501>.

Nordic_Semiconductor, 2013 NORDIC_SEMICONDUCTOR. *nRF24l01 Product Specification v2.0*. [S.l.], 2013.

Petersen e Carlsen (2011) PETERSEN, S.; CARLSEN, S. Wirelesshart versus isa100. 11a: The format war hits the factory floor. *Industrial Electronics Magazine, IEEE*, IEEE, v. 5, n. 4, p. 23–34, 2011.

Ruiz, Correia e Vieira 2004 RUIZ, L. B.; CORREIA, L. H. A.; VIEIRA, L. F. M. *Arquiteturas para Redes de Sensores Sem Fio*. 2004.

Solano et al. 2004 SOLANO, W.M.; JUNELL, J.; SCHMALZEL, J.L.; SHUMARD, K.C. Implementation of wireless and intelligent sensor technologies in the propulsion test environment. In: *Sensors for Industry Conference, 2004. Proceedings the ISA/IEEE*. [S.l.: s.n.], 2004. p. 135–138.

Song et al. 2008 SONG, J.; SONG, H.; MOK, A.K.; DEJI CHEN; LUCAS, M.; NIXON, M. Wirelesshart: Applying wireless technology in real-time industrial process control. In: *Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08. IEEE*. [S.l.: s.n.], 2008. p. 377–386. ISSN 1545-3421.

Song et al. 2009 SONG, W.-Z.; RENJIE HUANG; BEHROOZ SHIRAZI; RICHARD LAHUSEN Treemac: Localized {TDMA} {MAC} protocol for real-time high-data-rate sensor networks. *Pervasive and Mobile Computing*, v. 5, n. 6, p. 750 – 765, 2009. ISSN 1574-1192. PerCom 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1574119209000649>.

Sun 2001 SUN, J.-Z. Mobile ad hoc networking: an essential technology for pervasive computing. In: *Info-tech and Info-net, 2001. Proceedings. ICII 2001 -*

Beijing. 2001 International Conferences on. [S.l.: s.n.], 2001. v. 3, p. 316–321 vol.3.

Tanenbaum 2002 TANENBAUM, A. *Computer Networks*. 4th. ed. [S.l.]: Prentice Hall Professional Technical Reference, 2002. ISBN 0130661023.

Wang, Zhang e Wang 2006 WANG, N.; ZHANG, N.; WANG, M. Wireless sensors in agriculture and food industry - recent development and future perspective. *Computers and Electronics in Agriculture*, v. 50, n. 1, p. 1 – 14, 2006. ISSN 0168-1699. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168169905001572>.

Zheng 2006 ZHENG, L. Zigbee wireless sensor network in industrial applications. In: *SICE-ICASE, 2006. International Joint Conference*. [S.l.: s.n.], 2006. p. 1067–1070.