UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DIEGO HANSEN HAHN

DESENVOLVIMENTO DE UM PONTO DE ACESSO PARA REDES WIRELESSHART

Porto Alegre
2011

DIEGO HANSEN HAHN

DESENVOLVIMENTO DE UM PONTO DE ACESSO PARA REDES WIRELESSHART

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Engenheiro Eletricista.

ORIENTADOR: Prof. Dr. João César Netto

Porto Alegre
2011

DIEGO HANSEN HAHN

DESENVOLVIMENTO DE UM PONTO DE ACESSO PARA REDES WIRELESSHART

Este Projeto foi julgado adequado para a obtenção dos créditos da Disciplina Projeto de Diplomação do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. João César Netto,
Doutor pela Université Catholique de Louvain - Louvain, Bélgica

Banca Examinadora:

Prof. Dr. João César Netto,
Doutor pela Université Catholique de Louvain - Louvain, Bélgica

Msc. Ivan Müller,
Mestre pela Universidade Federal do Rio Grande do Sul - Porto Alegre, Brasil

Prof. Dr. Marcelo Götz,
Doutor pela Universität Paderborn - Paderborn, Alemanha

Chefe do DELET: _____
Prof. Dr. Altamiro Amadeu Susin

Porto Alegre, dezembro de 2011.

DEDICATÓRIA

À minha família, em especial ao meu avô Flávio, que, durante o tempo em que esteve entre nós, foi fonte inesgotável de suporte e apoio.

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais Dalto e Jane, minha irmã Yasmin, aos meus avós Flávio e Nelcy, e aos meus padrinhos Achylles e Flávia, por todo o suporte e apoio dados durante todos esses anos de faculdade.

Aos colegas e amigos de Engenharia Elétrica Cristina Simm, Laurence Rossetto, Renê Benvenuti e Thiago Both, pelo auxílio e apoio nos momentos de adversidade e de superação pelo quais passamos.

Aos colegas do Laboratório de Automação, Sistemas de Controle e Robótica, em especial ao professor João Netto, Ivan Müller e Jean Winter, que tanto contribuíram para o sucesso deste projeto.

RESUMO

O protocolo WirelessHART é atualmente o mais promissor e confiável protocolo de automação industrial sem fio padrão da indústria. Com a comunicação sem fio, os custos de instalação do cabeamento para interligar diversos dispositivos diminuem, além da possibilidade do controle ou monitoramento de processos em determinado local da planta que antes não podiam ser realizados com a tecnologia com fio, como em equipamentos giratórios. Este trabalho tem como objetivo, o desenvolvimento de um ponto de acesso para redes WirelessHART, sendo a porta de entrada para que demais dispositivos de campo WirelessHART integrem-se à rede.

Palavras-chave: WirelessHART; Protocolos de Comunicação sem Fio; Redes Industriais.

ABSTRACT

The WirelessHART protocol is, currently, the most prominent standard for industrial wireless automation protocol. With wireless communication, the costs from wired structure to connect many devices will decrease. Also, some processes at particular locations of the industrial plant that could not previously be reached with wired technology are now within reach, such as rotating equipments. The main goal of this document is to develop an access point so that others devices can join the WirelessHART network.

Keywords: WirelessHART; Wireless Communication Protocol; Industrial Network.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACK	Acknowledgement Frame
CCA	Clear Channel Assignement
DLPDU	Data Link Packet Data Unit
FCS	Frame Check Sequence
FSK	Frequency Shift Keying
GTW	Gateway
HART	Highway Addressable Remote Transducer
HCF	HART Communication Foundation
IEC	International Eletrotechnical Comission
MAC	Medium Access Control
MIC	Message Integrity Check
NAP	Network Access Point
NL	Network Layer
NPDU	Network Packet Data Unit
OSI	Open Systems Interconnection
PHY	Physical Layer
RCP	Radio Coprocessor
TDMA	Time Division Multiple Access

SUMÁRIO

1 INTRODUÇÃO

Para atender as novas necessidades da indústria, várias tecnologias sem fio foram criadas e promovidas, entre elas pode-se citar o ZigBee e o Bluetooth. Porém, a despeito do sucesso comercial de algumas, seu uso em plantas industriais não teve uma aceitação definitiva, muito por causa da pouca robustez dos protocolos. Ambientes industriais caracterizam-se pelo alto nível de ruídos eletromagnéticos, assim como pelas temperaturas e umidades elevadas. São esses tipos de adversidades que os equipamentos instalados geralmente devem suportar, e os protocolos de comunicação escolhidos devem funcionar de maneira a dar segurança e disponibilidade à rede projetada.

Desenvolvido em meados dos anos 80, o protocolo de comunicação HART^{TM}surgia como uma opção aos campo de automação industrial e instrumentação inteligente. Hoje, passados 30 anos desde sua criação, dispositivos com suporte a HART são produzidos em grandes quantidades, tornando-o um padrão reconhecido mundialmente pela indústria para a comunicação de instrumentos de campo inteligentes.

Em setembro de 2007, é apresentado o novo padrão WirelessHART^{TM}(HART sem fio) pelo HCF (HART Communications Foundation), tendo este o objetivo de expandir as capacidades do protocolo HART original, além de proteger a base global de dispositivos HART já instalados. As facilidades de um protocolo sem fio revelam-se na redução dos custos de instalação de novos dispositivos, assim como a diminuição dos riscos de danos físicos na instalação.

O objetivo deste trabalho é apresentar o desenvolvimento de um ponto de acesso à rede WirelessHART. Esse equipamento é peça fundamental na criação e integração de novos dispositivos à rede.

1.1 HART Communication Foundation

Criado em 1993, o HART Communication Foundation, uma organização internacional sem fins lucrativos, é o proprietário da tecnologia e autoridade central do protocolo HART.

O HCF mantém e controla as normas do protocolo, incluindo melhorias no padrão já existente, e desenvolvimento de novas tecnologias.

O foco principal de todas as atividades do HCF é promover a aplicação da tecnologia HART, fortalecer sua posição no mercado global e ajudar os usuários a maximizar seus investimentos em instrumentação inteligente.

1.1.1 Criação do HCF

Em setembro de 1990, um grupo de profissionais da indústria reuniu-se em Bloomington, Minnesota, para o primeiro encontro oficial daquilo que mais tarde seria chamado de *HART Users Group* (Grupo de Usuários HART). A despeito de ter durado dois dias e requerido apenas cinco páginas de registro, esse encontro marcou o início do HART como uma tecnologia de comunicação aberta.

Representantes de 26 companhias fizeram-se presentes no primeiro encontro, onde discutiu-se sobre a tecnologia HART - suas camadas de aplicação, enlace e física - e a organização de um grupo que manteria as normas do protocolo e proveria suporte mundial à tecnologia.

Entre essas companhias participantes, pode-se citar:

- Exxon
- Hitachi
- Siemens
- SMAR

Em um segundo encontro, em dezembro de 1990, as companhias-membro estabeleceram quatro grupos de trabalho: Definição, Teste de Conformidade, Interface Homem-Máquina e Interoperabilidade. Três encontros foram realizados em 1991 e, em novembro, outros dois grupos de trabalho foram criados: Saída e PID.

Em março de 1993, o Grupo de Usuários HART votou à criação do grupo de apoio que havia sido discutido na primeira reunião: uma organização independente e sem fins lucrativos para gerenciar e dar apoio ao protocolo HART. Em junho, a *HART Communication Foundation* era criada.

1.1.2 Melhorias na Tecnologia

Desde a criação do HCF, as companhias-membro trabalham junto com os usuários da indústria para desenvolver melhorias no protocolo HART que supram as necessidades da indústria por dispositivos de comunicação avançados e para criar ferramentas adicionais para desenvolvimento de novos produtos.

O HCF apresentou, em 1999, o Servidor HART, uma aplicação de fácil uso que fornece uma porta para acessar processos em tempo real e diagnosticar informações disponíveis na instrumentação HART de campo.

Em 2001, o HART 6, a primeira grande revisão do protocolo, foi lançado. Ele incluia suporte a *tags* longas, comandos adicionais e uma comunicação digital mais rápida.

A Linguagem de Descrição de Dispositivo HART (*HART Device Description Language*, ou DDL) tem sido um elemento-chave da tecnologia HART desde 1990. Em 2004, a DDL foi aprovada pelo IEC como um padrão internacional. Hoje em dia ela ainda permanece como a linguagem descritiva de comunicação digital mais importante e utilizada na indústria.

Em 2004, foi lançado o HART DD-IDE (*Device Description Integrated Development Environment*) com o *Smart Device Configurator* (SDC-625), um conjunto integrado de ferramentas que auxilia o desenvolvimento, teste e manutenção de DD para dispositivos HART.

Em 2007, o último passo na evolução do protocolo HART, o HART 7, trouxe a tecnologia ao mundo da comunicação sem fio. O HART 7 inclui o novo padrão WirelessHART e apresenta as necessidades críticas da indústria de processos por uma tecnologia segura, confiável e simples que forneça uma solução de redes industriais que seja de senso comum para a comunicação sem fio.

1.2 Protocolo HART

Com uma base global instalada de mais de 30 milhões de dispositivos, como mostrada na Figura 1 (www.hartcomm.org, 2011), o protocolo de comunicação HART é reconhecido mundialmente como um padrão pela indústria. Uma de suas principais caracterísi-

cas, sendo o que o diferencia dos demais protocolos *fieldbus*, é a sobreposição sem interferência do sinal analógico 4-20mA padrão por um sinal digital. Os dispositivos capazes de trabalhar com esta comunicação híbrida são chamados de *smart*, ou inteligentes.

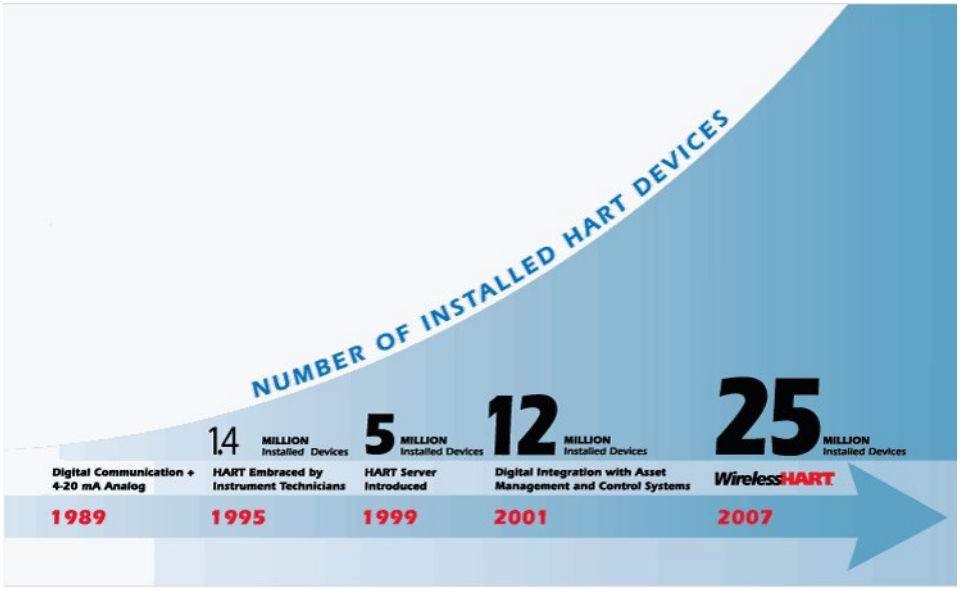

Figura 1: Evolução do protocolo HART.

1.2.1 Modelo OSI

O modelo OSI (Open Systems Interconnection) é uma forma de caracterizar e padronizar as funções de um sistema de comunicações baseado em termos de camadas de abstração. De acordo com a recomendação X.200, existem 7 camadas hierárquicas, cada uma nomeada genericamente de camada N. Dessa maneira, uma camada N+1 usa apenas suas próprias funções, ou da camada anterior N, assim escondendo a complexidade e transparecendo as operações para o usuário, seja ele um programa, ou outra camada.

A Figura 2, como visto em (WINTER, 2010). apresenta uma comparação entre os padrões HART e WirelessHART com o modelo OSI de 7 camadas. Como pode ser visto, tanto o HART quanto o WirelessHART compartilham uma mesma camada de aplicação, contudo, cada um possui sua própria camada física, de enlace e de rede.

Como mostrado na Figura 2, o protocolo HART implementa apenas as camadas 1, 2, 4 e 7. As demais camadas do modelo OSI não são necessárias, sendo bastante reduzidas

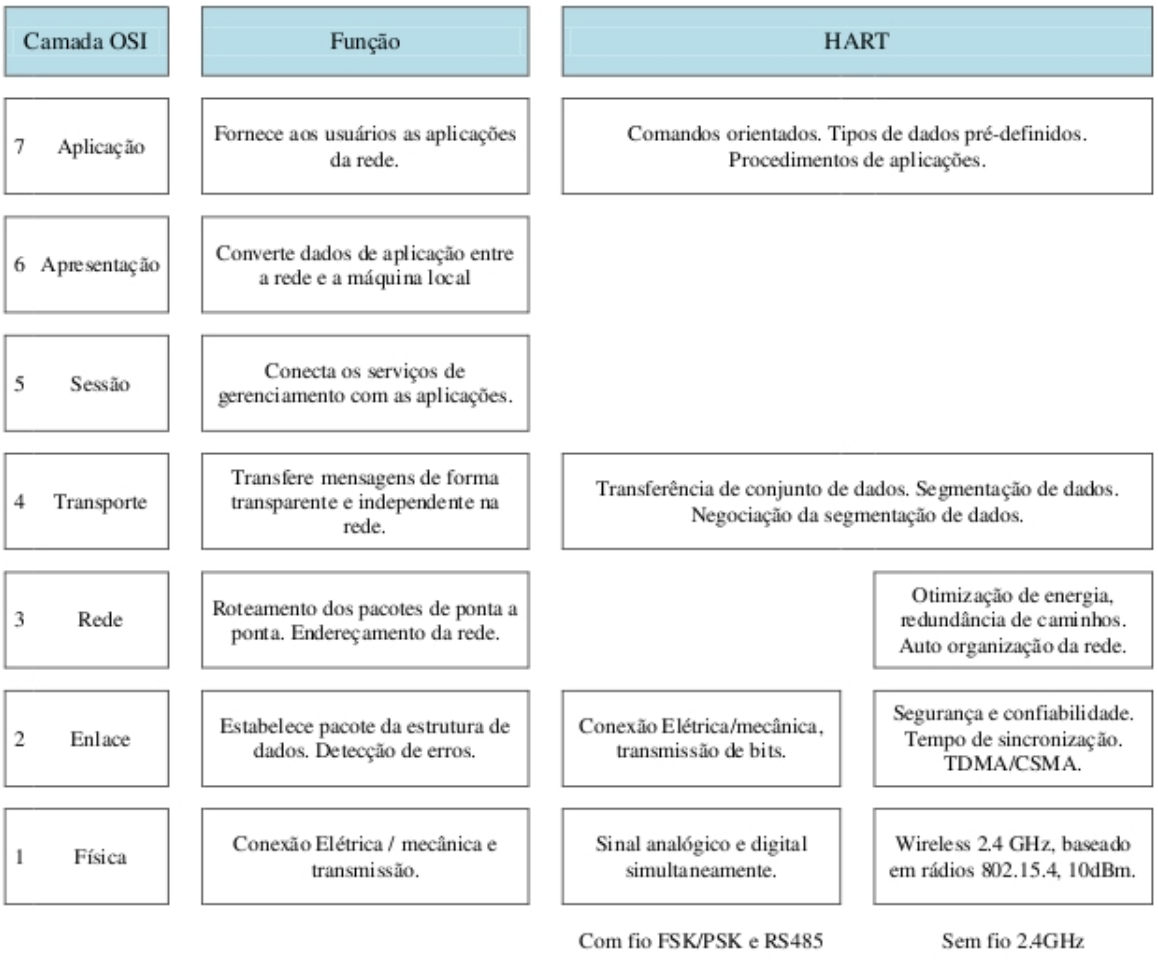

Figura 2: Modelo OSI de 7 camadas.

em um ambiente de rede local com operação entre dispositivos mestre-escravo baseado em transações únicas e sem resposta à mensagens corrompidas ou perdidas.

1.2.2 Funcionamento do Protocolo HART

O protocolo *wired* HART foi baseado no padrão Bell 202. Em um sinal analógico 4-20mA padrão, é sobreposto um sinal digital modulado em FSK (*Frequency Shift Keying*, ou Modulação por Chaveamento de Frequência), como mostrado na Figura 3. Assim, os bits 0 e 1 são transmitidos modulando uma onda sinusoidal de 1mA de valor pico a pico nas frequências de 2400Hz e 1200Hz, respectivamente. Como o valor médio de uma onda sinusoidal é 0, nenhum componente DC é adicionado ao sinal original de 4-20mA, sendo normalmente utilizado um filtro passa-baixas para a recuperação do sinal analógico.

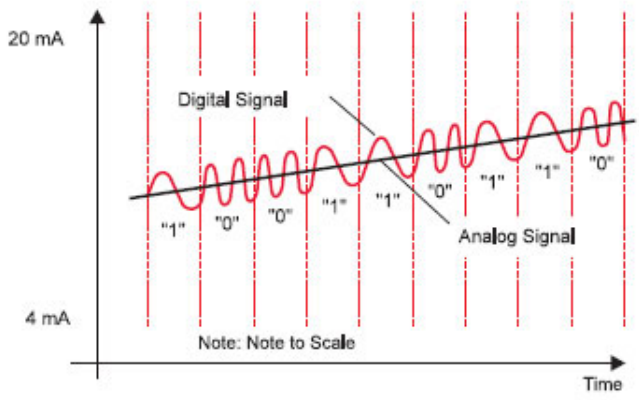

Figura 3: Sinal digital modulado em FSK sobreposto ao analógico 4-20mA.

2 WIRELESSHART

2.1 Visão Geral

O WirelessHART é um avanço da tecnologia HART, sendo o primeiro padrão de comunicação sem fio aprovado pelo IEC (norma IEC 62591). A tecnologia WirelessHART fornece uma rede segura utilizando a banda de rádio de 2,4GHz ISM, de acordo com a norma IEEE 802.15.4 com espalhamento espectral por sequência direta (Direct Sequence Spread Spectrum - DSSS).

Os seguintes dispositivos compõem a rede WirelessHART:

Dispositivos de Campo

São os responsáveis por interagir com o processo. Produzem e consomem pacotes WirelessHART e devem ser capazes de rotear esses pacotes em função dos outros dispositivos da rede.

Adaptadores

São os responsáveis por ligar um dispositivo HART com fio a rede WirelessHART.

Handhelds

São dispositivos portáteis controlados por operadores da planta e usados no monitoramento e diagnóstico da rede WirelessHART.

Gateway

É a entidade responsável por conectar a rede WirelessHART com a rede de automação da planta, permitindo que os dados circulem entre duas redes distintas. O *gateway* também pode ser usado para conversão de dados entre protocolos, já que o padrão WirelessHART não define um protocolo para a planta.

Pontos de Acesso

Conectam o *gateway* a rede WirelessHART. Possuem de um lado uma conexão

WirelessHART, e do outro, uma conexão externa, podendo ser Modbus, TCP-IP ou outra proprietária. O padrão WirelessHART não define essa conexão.

Gerente de Rede

É o responsável por configurar a rede, agendar a comunicação entre os demais dispositivos, gerenciar as rotas e monitorar e reportar a saúde da rede.

A Figura 4, como visto em (www.hartcomm.org, 2011), ilustra uma rede WirelessHART típica.

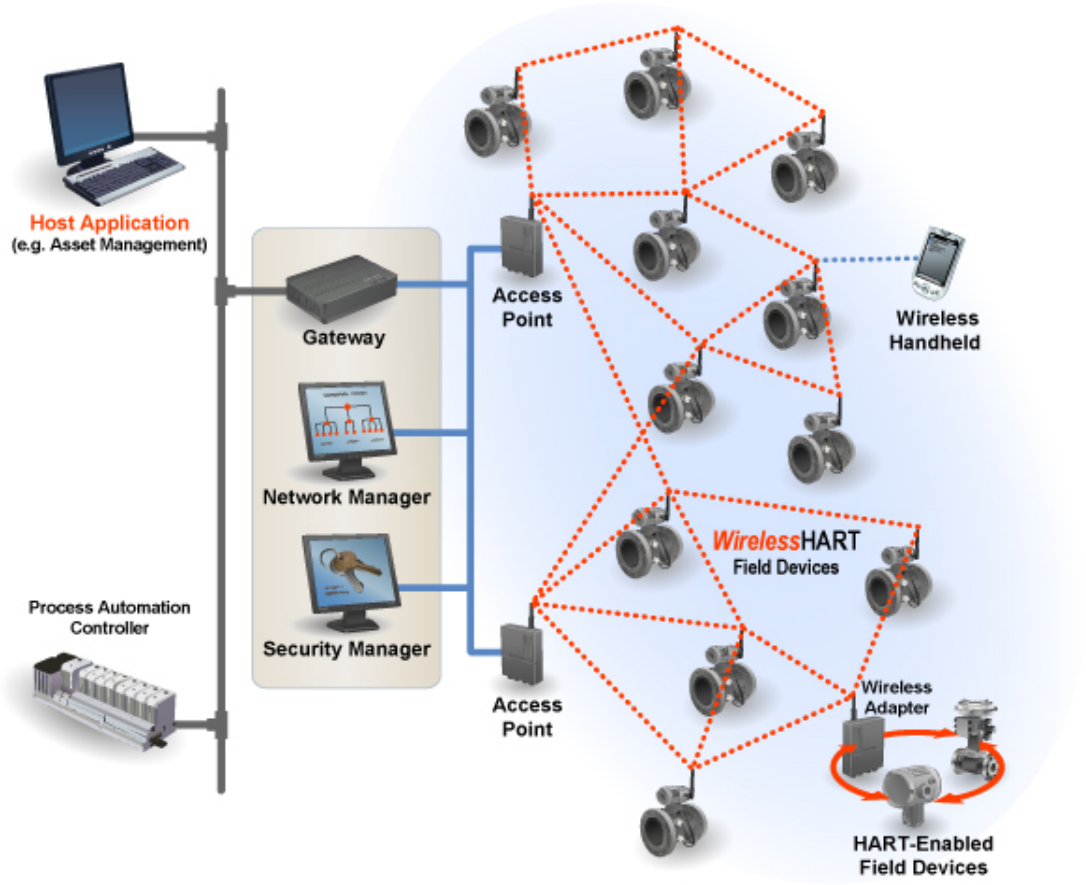

Figura 4: Dispositivos em uma rede WirelessHART.

A comunicação entre esses dispositivos é feita de maneira estritamente sincronizada. São definidos *slots* de tempo precisos de 10ms e é utilizado a tecnologia TDMA (*Time Division Multiple Access*, ou Acesso Múltiplo por Divisão de Tempo) para prover uma comunicação determinística e livre de colisões. Boa parte dessas comunicações são realizadas através de rotas em grafos criadas por um gerenciador de rede central baseado na

topologia da rede. Também é função do gerenciador fazer o agendamento das comunicações, ou seja, definir qual *slot* de tempo será usado em qual momento.

2.2 Camada Física WirelessHART

Esta camada é a responsável por definir as características do rádio, tais como métodos de sinalização, potência do sinal e sensibilidade do dispositivo. É baseada no padrão IEEE 804.15.4-2006 2,4GHz DSSS. Desta maneira, o WirelessHART opera na faixa ISM 2400-2483.5 MHz livre de licença, com uma taxa de dados de até 250 kbit/s. Seus canais são enumerados de 11 a 26, com um intervalo de 5 MHz entre canais adjacentes. Estão disponíveis 15 canais na banda 2450 MHz ISM, uma vez que o canal 26 não é suportado pelo protocolo, visto que seu uso não é permitido em alguns países. O centro de frequência dos canais, em *megahertz*, é definido por:

$$F_c = 2045 + 5(k - 11), \text{com k = 11 .. 26} \quad (1)$$

Onde k é o número do canal.

Índice	Canal 802.15.4	Frequência (MHz)	Índice	Canal 802.15.4	Frequência (MHz)
0	11	2405	8	19	2445
1	12	2410	9	20	2450
2	13	2415	10	21	2455
3	14	2420	11	22	2460
4	15	2425	12	23	2465
5	16	2430	13	24	2470
6	17	2435	14	25	2475
7	18	2440	15	26	Não usado

Tabela 1: Frequência dos canais

Pode-se dizer a camada física WirelessHART limita-se a transmitir e receber mensagens de dados no padrão IEEE 802.15.4. As características notáveis dessa camada são (CHEN *et al*, 2010):

Transmissão de Energia

O padrão IEEE 802.15.4 especifica uma rede de área pessoal (*Personal Area Network, PAN*) para espaços operacionais pessoais (*Personal Operating Space, POS*) de 10 metros. A malha de rede WirelessHART cobre uma área relativamente grande. Todos os dispositivos devem fornecer um EIRP de +10dBm (10mW) $\pm$ 3dB. A potência de transmissão é programável na faixa de -10dBm a +10dBm. O alcance da transmissão deve ser de até 100m.

Salto de Canais

No WirelessHART, o canal físico é alterado a cada transmissão. Dessa maneira, toda a banda de 2450MHz é usada.

Por fim, os dados codificados a serem transmitidos são modulados por deslocamento de fase em quadratura (*Quadrature Phase Shift Keying, QPSK*).

2.3 Camada de Enlace WirelessHART

Como dito anteriormente, uma característica distinta do protocolo WirelessHART é sua camada de enlace sincronizada no tempo, onde é definido um intervalo de tempo (*slot*) rigorosamente preciso de 10ms utilizando tecnologia TDMA. O conceito de *superframe* é apresentado, sendo este uma sequência consecutiva de *slots* de tempo. Um *superframe* é periódico, com período igual a quantidade de *slots* de tempo que o compõe. Todos os *superframes* numa rede WirelessHART iniciam com um ASN (Absolute Slot Number) igual a 0 no momento em que a rede é criada.

No protocolo WirelessHART, uma transação em um *slot* de tempo é descrito por um vetor *frame id, index, type, src addr, dst addr, channel offset*, onde:

Frame ID: identifica um *superframe* específico.

Index: índice do *slot* no *superframe*.

Type: indica o tipo de *slot*. Pode ser transmissão, recepção ou inativo.

Source Address: endereço do dispositivo de origem.

Destination Address: endereço do dispositivo de destino.

Channel Offset: fornece o canal lógico a ser usado na transmissão.

Para ajustar o uso dos canais, o padrão WirelessHART introduz o conceito de "lista negra". Assim o administrador de rede pode desabilitar totalmente o uso de canais afetados por consistentes interferências colocando-os nessa lista. Para suportar o salto de canais, cada dispositivo mantém uma tabela com os canais ativos. Devido à lista negra, cada tabela tem até 16 entradas. Para um dado *slot* e *offset* do canal, o canal atual é calculado por:

$$\text{CanalAtual} = (\text{OffsetCanal} + \text{ASN}) \ \% \ \text{NumCanais} \tag{2}$$

Esse canal atual serve como um índice para a tabela de canais ativos.

A Figura 5, como visto em (CHEN *et al*, 2010), apresenta os seis módulo que, de modo geral, compõem a camada de enlace.

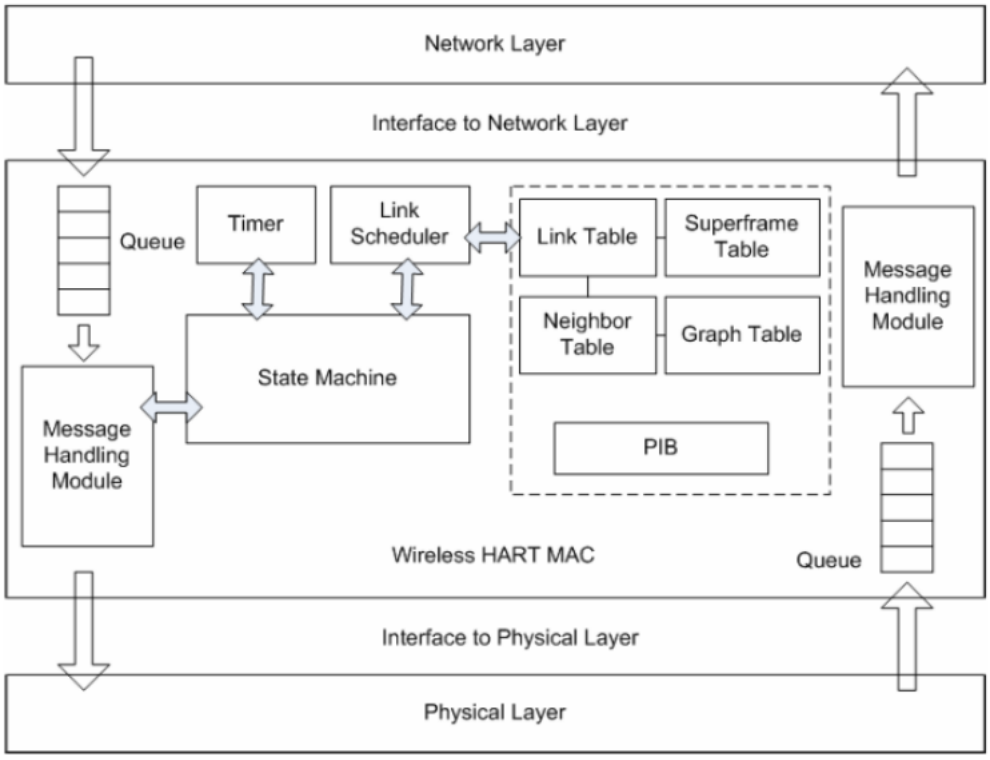

Figura 5: Arquitetura da camada de enlace.

A função da camada de enlace é criar e gerenciar os *frames* utilizados na comunicação, além de ser responsável pela transferência segura entre os nós da rede, detectando e corrigindo os possíveis erros advindos da camada física. Possui duas subcamadas, chamadas LLC (*Logical Link Control*) e MAC (*Media Access Control*).

2.3.1 Logical Link Control

É a camada mais alta entre as duas subcamadas da camada de enlace. O LLC é o responsável pelo controle de erros, controle do fluxo de pacotes, montagem dos *frames* e endereçamento.

O pacote da camada de enlace (DLPDU) é estruturado do seguinte modo:

- Byte único de valor fixo 0x41;
- Um byte com especificador de endereço (*Address Specifier*);
- Um byte com número da sequência (*sequence number*);
- Dois bytes de *Network ID*;
- Endereços de origem e destino (2 ou 8 bytes);
- Um byte de especificação do DLPDU;
- *Payload* da camada de enlace;
- Quatro bytes de MIC;
- Dois bytes de CRC-16;

A Figura 6, como visto em (CHEN *et al*, 2010), ilustra a estrutura do DLPDU.

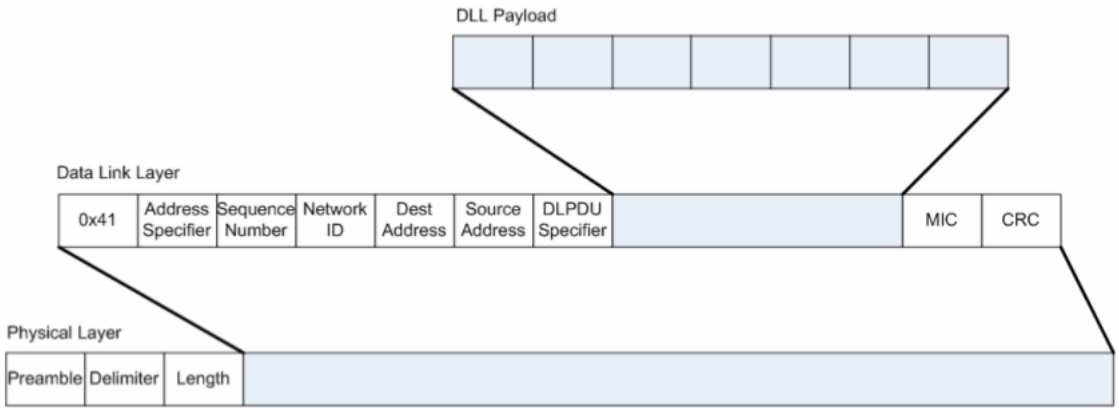

Figura 6: Estrutura do DLPDU.

A descrição de cada campo é mostrada na sequência:

Especificador de Endereço:

Define qual tipo de endereço de origem e destino está gravado no pacote. A Figura 7, como visto em (WINTER, 2010), ilustra o formato desse byte.

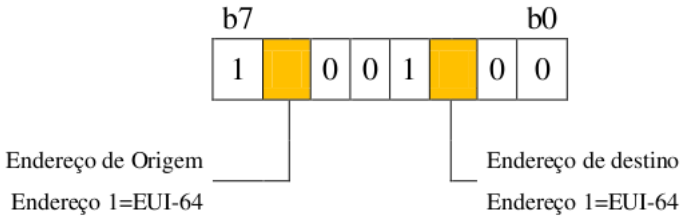

Figura 7: Especificador de endereço.

Sequence Number: é o byte menos significativo do número de *slot* absoluto (ASN).

Network ID: identificador da rede. Se esse identificador não for o mesmo gravado no dispositivo, a mensagem não é para essa rede e é descartada.

Endereço de Origem e Destino: os dispositivos WirelessHART podem ter dois endereços: um curto (ou apelido) de 2 bytes e um longo de 8 bytes seguindo o padrão IEEE EUI-64. O tipo de endereço gravado no pacote é definido pelo especificador de endereço.

O endereço curto é definido pelo gerente de rede e é único apenas dentro da rede a qual o dispositivo pertence. O endereço 0xFFFF é usado para *broadcast*, enquanto os demais endereços são usados para cada dispositivo específico da rede.

O endereço longo consiste de 3 bytes provenientes da OUI (Organizationally Unique Identifier, gerenciado pelo IEEE) e 5 bytes de identificação única controlado pelo protocolo HART.

Especificador do Pacote: especifica o tipo e prioridade da mensagem, além de qual chave (*network key* ou *public key*) será usada para autenticar a mensagem. A Figura 8, como visto em (CHEN *et al*, 2010), define o especificador do pacote.

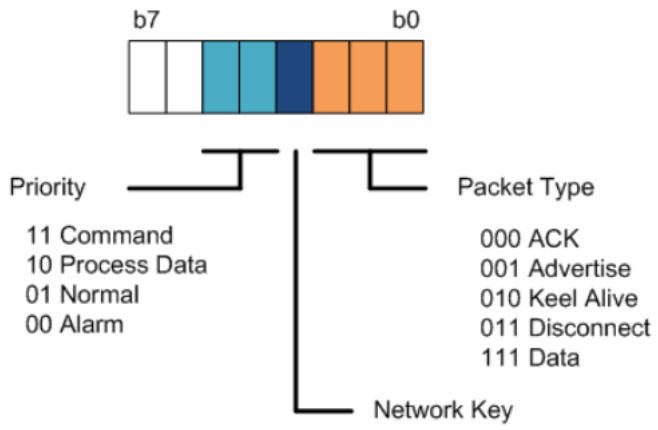

Figura 8: Especificador do pacote.

DLPDU Payload: o conteúdo do pacote depende do especificador do pacote. O *payload* de um pacote de dados é o próprio pacote da camada de rede (NPDU), enquanto que, para pacotes do tipo *ACK*, *Advertise*, *Keep-Alive* e *Disconnect*, o *payload* é vazio.

MIC (*Message Integrity Code*): é utilizado para autenticação do DLPDU.

CRC-16:

Usado para verificar se há erros no pacote. Utiliza o seguinte polinômio:

$$G_{16} = x^{16} + x^{12} + x^5 + 1 \tag{3}$$

2.3.2 Media Access Control

Os principais objetivos da camada MAC são manter o sincronismo dos *slots*, identificar *slots* que precisam de serviços, escutar pacotes que estão sendo enviados pelos vizinhos e propagar os pacotes que vêm da camada de rede. Em suma, o MAC é o mecanismo de arbitração do canal de comunicação. Neste caso, o TDMA.

2.3.2.1 Temporização dos Slots

Todas as transações ocorrem em *slots* que seguem específicos requerimentos de tempo. A Figura 9 apresenta uma visão geral dos tempos de um *slot*, enquanto a Tabela 2 apresenta uma breve descrição de cada tempo.

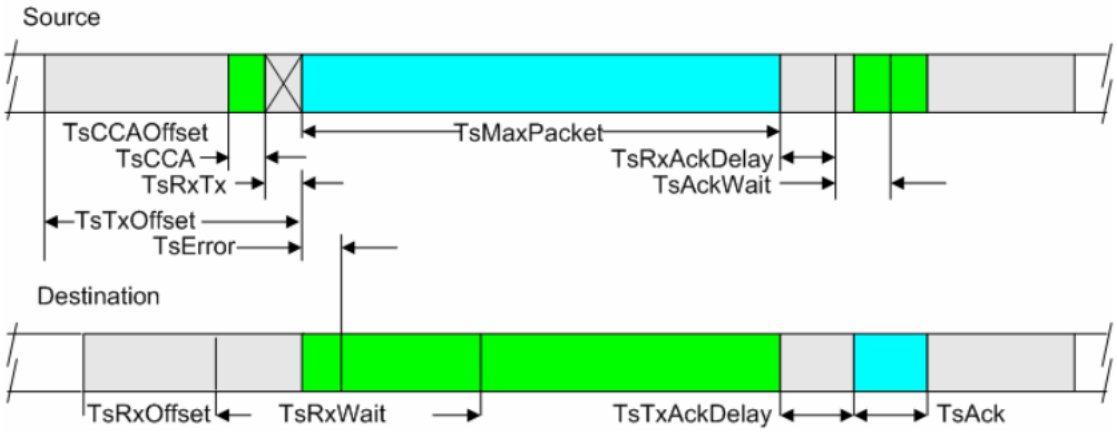

Figura 9: Temporização de um *slot*.

Símbolo	**Descrição**
TsTxOffset	Tempo entre o início do *slot* e o ínicio da transmissão do pré-âmbulo.
TsRxOffset	Tempo entre o início do *slot* e o momento em que o receptor deve começar a escutar.
TsRxWait	Tempo mínimo para esperar pelo início da mensagem.
TsError	É a diferença entre o início de fato da mensagem e o início ideal da mensagem como visto pelo dispositivo receptor.
TsMaxPacket	O tempo total que leva para transmissão da mensagem mais longa possível.
TsTxAckDelay	Tempo entre o fim da mensagem e o início do ACK.
TsRxAckDelay	Tempo entre o fim da mensagem e o momento em que o transmissor deve escutar pelo ACK.
TsAckWait	Tempo mínimo para esperar por um ACK.
TsAck	Tempo para transmitir um ACK.
TsCCAOffset	Tempo entre o início do slot e o começo do CCA.
TsCCA	Tempo para realizar o CCA.
TsRxTx	O tempo mais longo que leva para o dispositivo mudar de receptor para transmissor, e vice-versa.

Tabela 2: Descrição dos tempos em um *slot*.

2.4 Camada de Rede WirelessHART

No modelo OSI de 7 camadas, a camada de rede é a responsável pelas funções de roteamento de rede designando o endereçamento e a entrega dos dados. A camada de transporte controla a confiabilidade e o tempo de transmissão dos dados entre os nós da rede através de controle de fluxo, segmentação e controle de erro. A camada de sessão

controla o diálogo, a sessão, e conexões entre dois nós da rede. No padrão WirelessHART, a camada de rede engloba essas três camadas (CHEN *et al*, 2010). É nessa camada onde as redes tradicionais HART *Token-Passing* e WirelessHART baseada em TDMA convergem, como mostrado na figura 10.

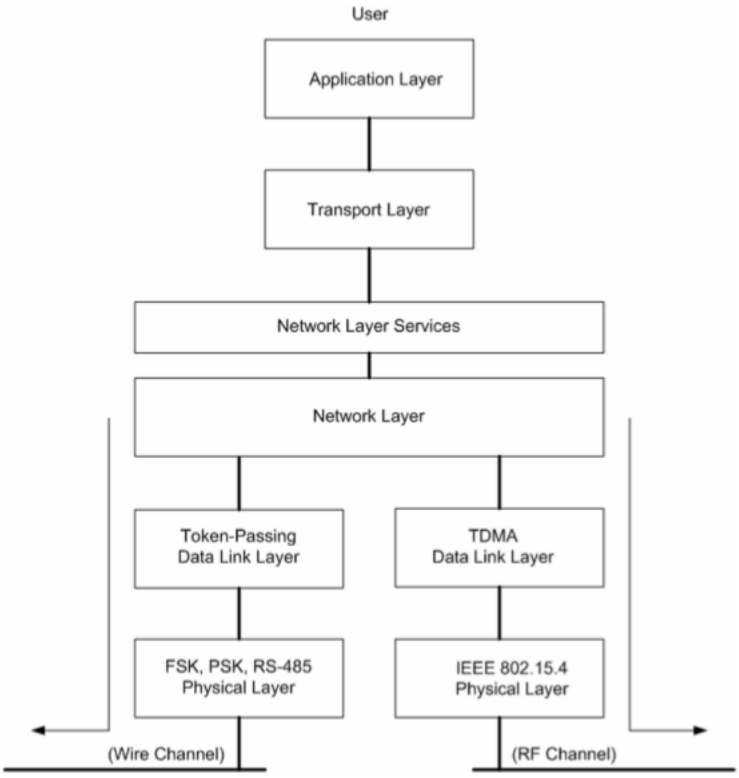

Figura 10: Convergências das tecnologias HART e WirelessHART.

Para dar suporte à tecnologia de comunicação de malha, cada dispositivo WirelessHART deve ser capaz de encaminhar pacotes em nome de outros dispositivos. A camada de rede suporta três protocolos fundamentais de roteamento definidos no padrão WirelessHART: roteamento na fonte (*Source Routing*), roteamento por grafos (*Graph Routing*) e roteamento por *superframes* (*Superframe Routing*).

O pacote de dados da camada de rede (NPDU) consiste de três funções distintas: os primeiros campos definem a rota do pacote até seu destino final; os campos de segurança, para garantir que comunicação privada enter dois pontos não tenha sofrido interferência; o *payload* do pacote, que é encriptado e contém a informação a ser trocada através da rede. Essa estruturação é apresentada na Figura 11, como visto em (CHEN *et al*, 2010).

Os campos da NPDU contêm os seguintes dados:

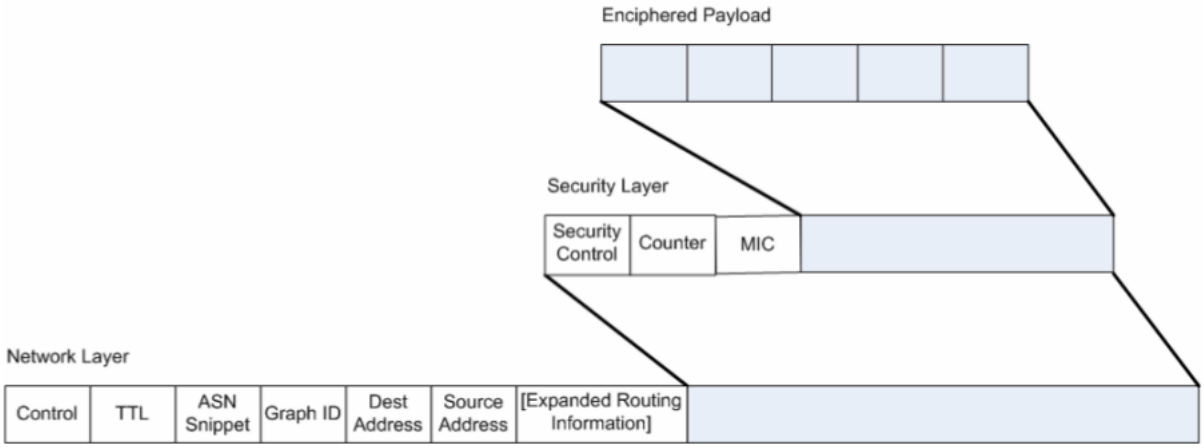

Figura 11: Estrutura do NPDU.

- Um byte de controle;
- Um byte TTL (*Time-to-Live*);
- Os dois byte menos significativos do ASN;
- Dois bytes de *Network ID*;
- Dois bytes para o *Graph ID*;
- O endereço da origem e do destino final;
- Campos de roteamento opcionais;

2.5 Camada de Aplicação WirelessHART

No modelo de camadas OSI, a camada de aplicação é a camada mais próxima ao usuário final. Ela fornece um meio à aplicação para acessar informações na rede. Na camada de aplicação WirelessHART, os comandos entre mestre e escravo são a base da comunicação.

2.5.1 Tabela de Comandos

Os comandos são divididos em sete grupos: Universal, Práticos, Não Público, Família de Dispositivos, Funções Específicas de Dispositivos e Sem Fio. O protocolo suporta comandos de byte único (0 a 255) e de dois bytes (256 a 65536). Para o primeiro tipo, utiliza-se o campo de comando da mensagem para indicar o número do comando. Para o tipo seguinte, o campo de comando apresenta valor 31 (0x1F) e os bytes do comando são

encontrados no campo de dados. Na sequência a descrição de cada família de comandos, como visto em (HCF SPEC-099, 2007). A Tabela 3 apresenta a divisão dos comandos nas famílias.

Comandos Universais: coleção de comandos os quais todos os dispositivos HART devem suportar.

Comandos Práticos: coleção de comandos para aplicação em uma faixa ampla de dispositivos. Fornece funções comuns para a maioria dos dispositivos. Se um dispositivo utiliza comandos práticos, o comando deve ser aplicado exatamente como especificado.

Comandos Não Público: coleção especial de comandos para utilização pelo fabricante apenas durante processo de fabricação do dispositivo de campo. Estes comandos não devem ser utilizados nos dispositivos de campo durante operação normal.

Comandos de Família de Dispositivos: coleção de comandos que permitem a configuração e parametrização dos dispositivos de campo sem necessidade de uso de comandos ou *drivers* específicos, de modo a manter a compatibilidade entre dispositivos de mesma família, mas de fabricantes diferentes.

Comandos Específicos de Dispositivos: coleção de comandos definidos pelo fabricante de acordo com as necessidades dos dispositivos de campo.

Comandos Sem Fio: coleção de comandos destinado a WirelessHART. Todos os dispositivos que suportam WirelessHART devem ter implementados todos esses comandos.

Número do Comando	**Descrição**
0-30, 38, 48	Comandos universais
31	*Flag* de expansão
31-121 (exceto 38 e 48)	Comandos práticos
122-126	Não público
127	Reservado
128-253	Específico do dispositivo
254-511	Reservado
512-767	Comandos práticos adicionais
768-1023	WirelessHART
1024-33791	Família de dispositivos
33792-64511	Reservado
64512-64765	Dispositivo específico WirelessHART
64766-64767	Reservado
64768-65021	Adicional de dispositivo específico
65022-65535	Reservado

Tabela 3: Famílias de comandos HART.

3 DESENVOLVIMENTO DO PONTO DE ACESSO

A partir do estudo sobre as principais características do protocolo WirelessHART, este trabalho propõe o desenvolvimento de um ponto de acesso à redes WirelessHART. A criação de um ponto de acesso permite os mais variados estudos sobre WirelessHART, como testes de montagem de rede, escalonamento e roteamento de pacotes. Este trabalho fez parte do projeto "Controle de Válvulas sem Fio"entre o LASCAR (Laboratório de Automação, Sistemas de Controle e Robótica), UERJ, Petrobrás e Coester, de onde foram fornecidos os *stacks* WirelessHART, documentações e normas, assim como os dispositivos de campo utilizados nos testes.

A Figura 12 apresenta o diagrama de blocos no qual esse trabalho se baseia. Além da criação do ponto de acesso propriamente dito, implementado como duas entidades, criou-se também um *gateway* virtual para fazer o fluxo das mensagens entre o gerente de rede, implementado no formato de *script* apenas para montar a mais simples estrutura de rede para testes. Essas entidades correspondem, respectivamente, aos blocos amarelo, vermelho e verde.

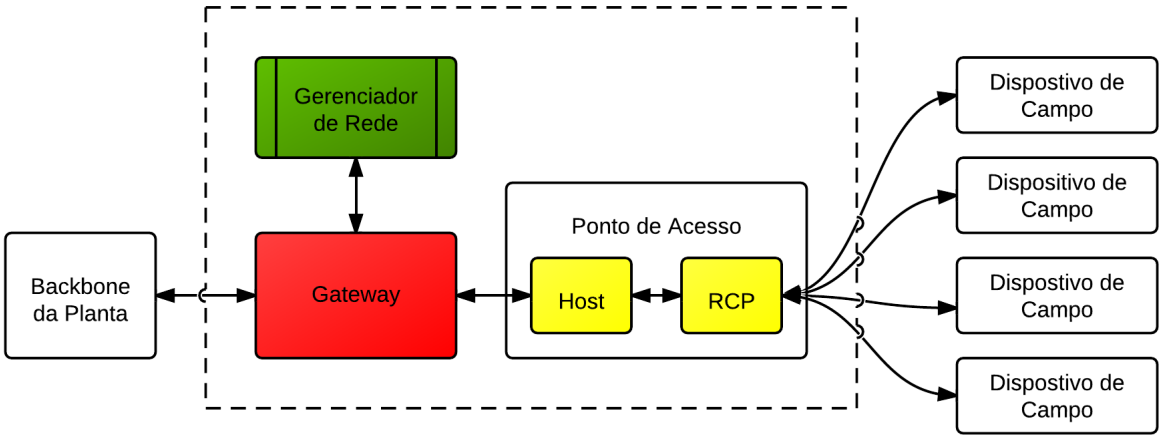

Figura 12: Diagrama de blocos do projeto.

O processo de criação da rede, comandos enviados e fluxo de mensagens, o qual será função das entidades criadas, é baseado na Figura 13, como visto em (HCF SPEC-085, 2009).

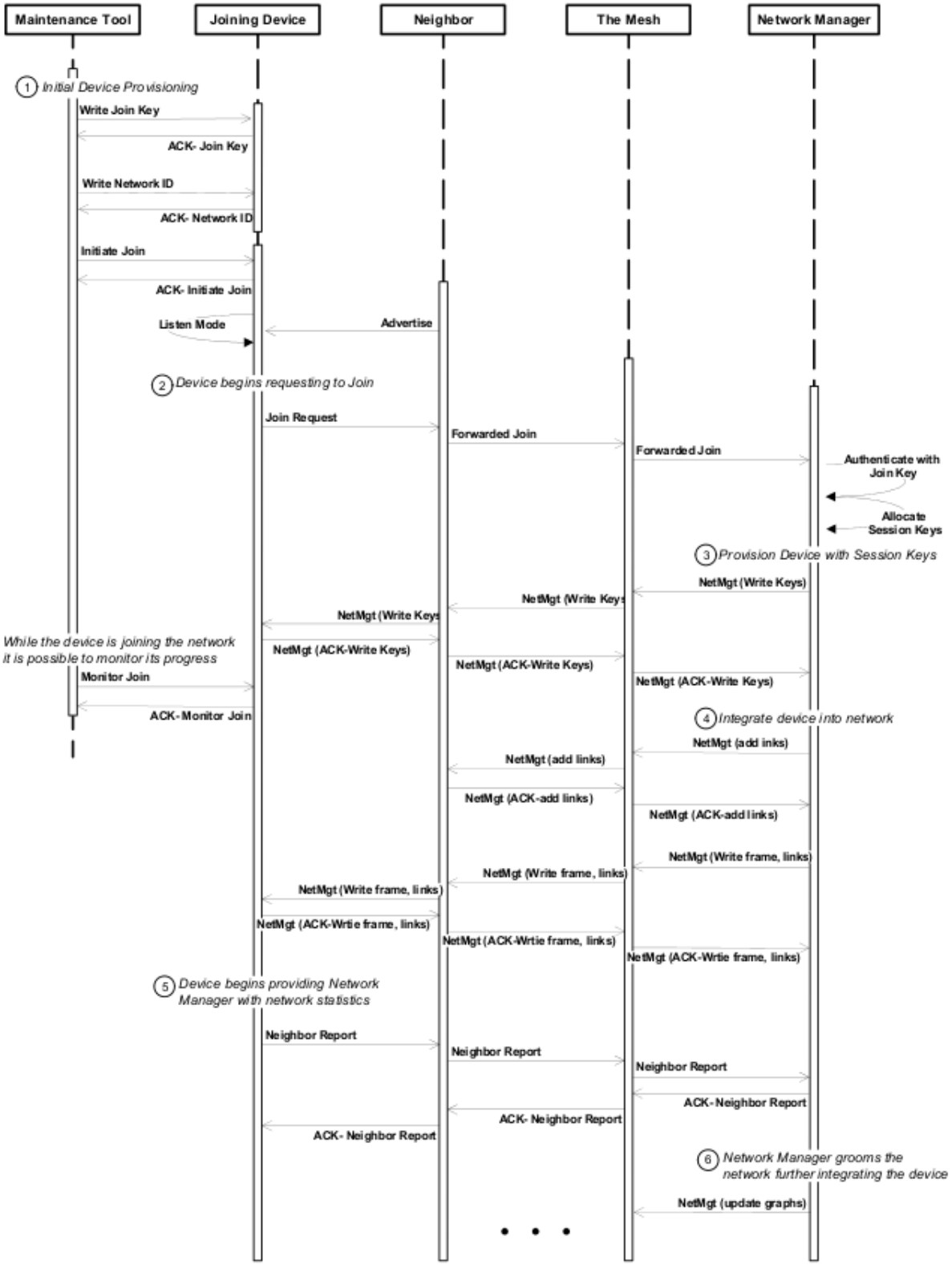

Figura 13: Sequência de passos para estruturação de uma rede.

3.1 Ponto de Acesso

O ponto de acesso é um dispositivo WirelessHART especial que tem a função de descobrir e conectar os demais dispositivos de campo ao gateway. Em resumo, o NAP (Network Access Point) deve ser capaz de fazer o fluxo de mensagens entre o *gateway* e os dispositivos de campo. Para a criação do ponto de acesso, optou-se por dividi-lo em duas entidades:

Software

Desenvolvido para Linux, servirá como ligação entre o *gateway* e o rádio. Será chamado, no decorrer do trabalho, de *host*.

Firmware

Desenvolvido para a plataforma MC1322x, mostrada na Figura 14, fará o papel de rádio, servindo de ligação entre os dispositivos de campo e as entidades virtuais rodando em Linux. Usará os *stacks* WirelessHART fornecido, que funcionarão como camada física, de enlace e de rede.

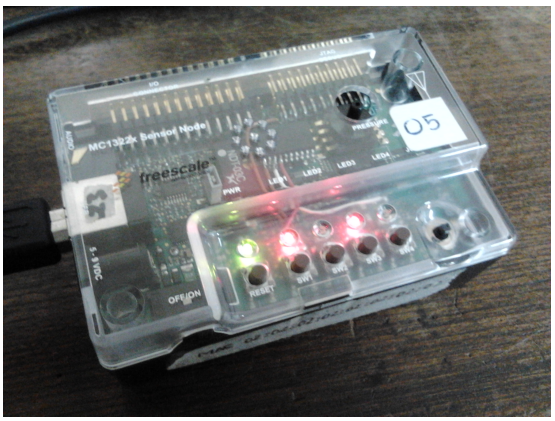

Figura 14: Plataforma Freescale MC1322x.

3.1.1 NAP - Host

O *host* é a entidade que roda em Linux. Ao ser inicializado, o *host* deverá encontrar o *gateway* na rede e estabelecer uma conexão TCP com o mesmo. Para facilitar o desenvolvimento do projeto, não foi estabelecido uma conexão segura entre as duas entidades. É

importante dizer que a norma WirelessHART não define que protocolo usar para conectar o *gateway* ao ponto de acesso. Tendo encontrado o *gateway*, o *host* deve se conectar ao rádio. Para isso, estabelece-se uma conexão serial rápida, com velocidade de 115200 bps, já que tempo é uma variável de grande importância quando lidamos com tecnologia TDMA.

A Figura 15 ilustra o diagrama de blocos implementado no *host*.

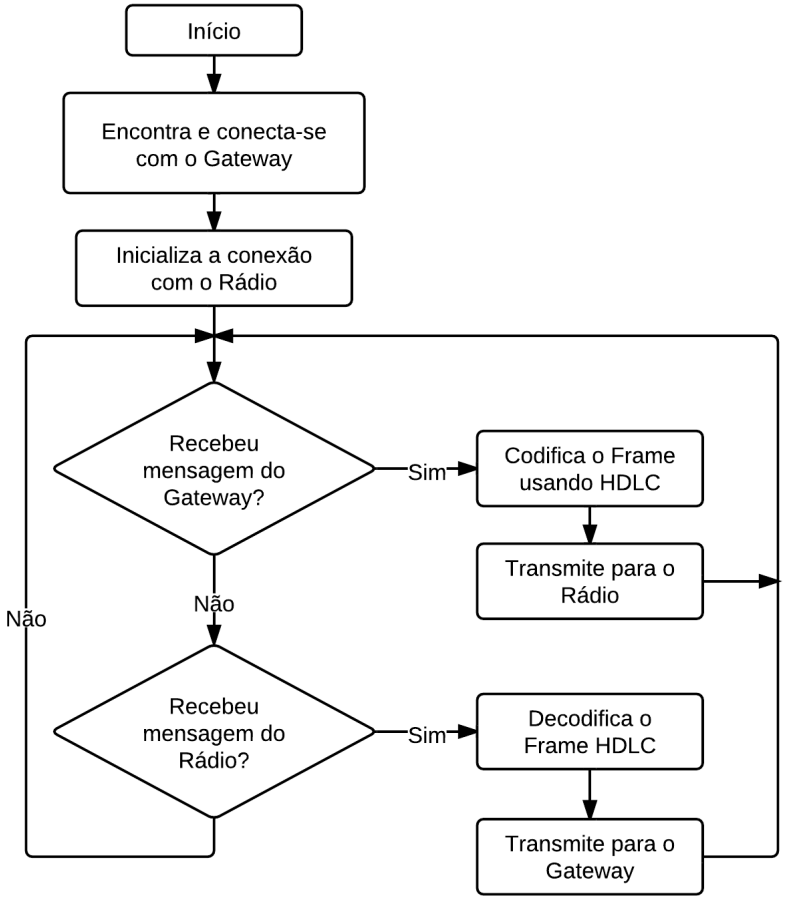

Figura 15: Diagrama de blocos do NAP Host.

Uma vez que o *host* conectou-se ao *gateway* e ao rádio, ele pode começar a fazer o tráfego de mensagens entre essas duas entidades. Assim, o *host* fica em um laço infinito monitorando cada canal de conexão e chamando a respectiva função de processamento para cada tipo de mensagem recebida. Quando uma mensagem enviada pelo *gateway* é recebida, o *host* deve ler o conteúdo do pacote TCP (realizadas através das funções de *socket* fornecidas pelo sistema operacional), codificá-lo em um novo frame "HDLC-like"e

transmiti-lo para o rádio. De outro modo, ele deve fazer o processo inverso, decodificar o frame HDLC e enviá-lo em um pacote TCP para o *gateway*.

Optou-se por codificar os dados transmitidos para o rádio com protocolo HDLC para permitir uma verificação mais eficientes de erros nas mensagens. O protocolo HDLC atua na camada 2 do modelo OSI (camada de enlace) e utiliza a técnica de *byte stuffing* para garantir que as *flags* que delimitam o *frame* HDLC não apareçam no *payload*, além de inserir o campo FCS (Frame Check Sequence) para garantir integridade da mensagem (SIMPSON, 1994).

A Figura 16 apresenta a criação de um frame HDLC a partir de uma mensagem qualquer.

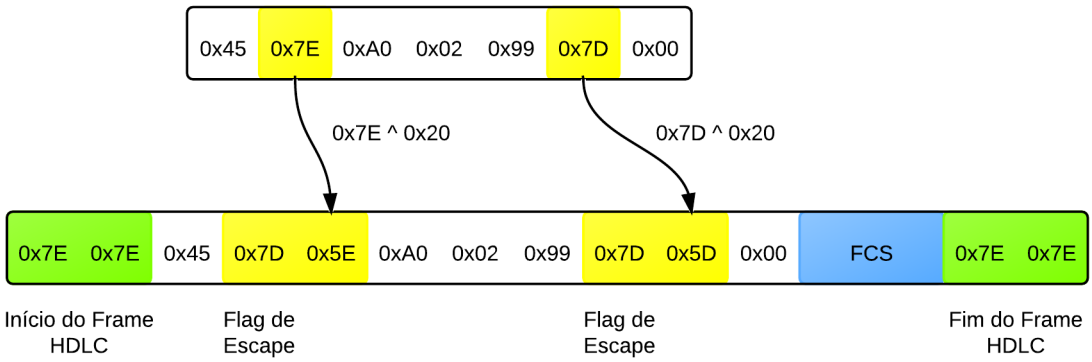

Figura 16: Frame HDLC.

Em resumo, o frame HDLC implementado inicia com duas *flags* 0x7E. Na sequência, vem o conteúdo original da mensagem. Aos bytes 0x7D e 0x7E, que porventura apareçam na mensagem, é aplicada função XOR com o valor 0x20, e inserido o byte de escape 0x7D no frame. Assim, 0x7D aparecerá como 0x7D5D, e 0x7E como 0x7D5E. Por fim é escrito o valor do FCS da mensagem original e duas *flags* 0x7E para finalizar o *frame*. Na implementação original do HDLC, ainda aparecem os campos de endereço e controle, que não foram considerados necessários para o projeto.

3.1.2 NAP - RCP

Para fazer o papel de rádio, optou-se por utilizar a plataforma Freescale MC1322x. A Figura 17 apresenta um diagrama de blocos simplificado dessa plataforma.

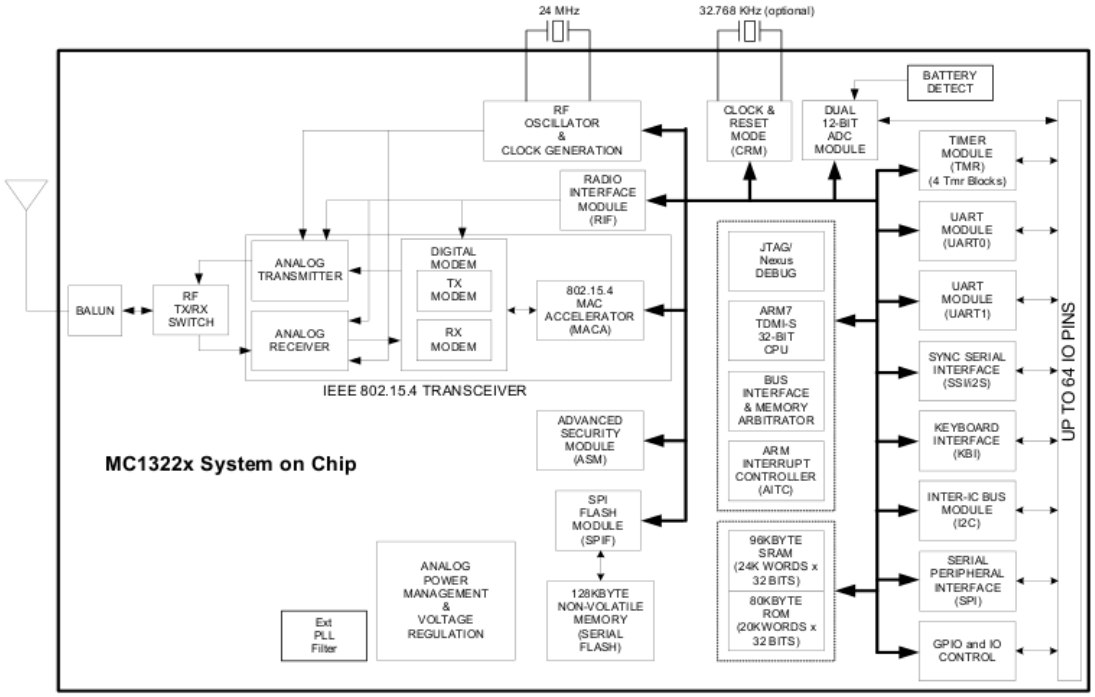

Figura 17: Diagrama de blocos simplificado do MC1322x.

Como pode ser visto, o Freescale MC1322x possui um transceptor para a frequência de rádio ISM de 2,4GHz completo e de baixo consumo, um microcontrolador baseado na arquitetura ARM7 TDMI-S 32-bits, além de aceleração por hardware para a interface MAC e segurança AES da norma IEEE 802.15.4. A plataforma também possui uma porta de depuração, seguindo o padrão JTAG. O desenvolvimento do *firmware* é realizado na IDE IAR Systems.

Neste dispositivo foram implementados os *stacks* WirelessHART fornecidos com apenas algumas mudanças, visto que eles foram desenvolvidos originalmente para projetos de dispositivos de campo. As alterações feitas nos *stacks* serão comentadas no decorrer da seção.

O projeto consistiu em modificar a inicialização das *tasks* pelo sistema operacional para que criasse uma nova *task* onde foram implementadas as funções de recepção e transmissão das mensagens do *host*, informar à camada de enlace relativo a novas mensagens e requisição à camada de enlace para que seja procedido o *join* do NAP à rede WirelessHART. Essa nova *task* não deve ter uma prioridade muito alta, visto que é mais importante processar as mensagens que já estão na camada de enlace (prioridade mais alta

do sistema), do que receber novas mensagens. A Figura 18 ilustra o diagrama hierárquico funcional dessa nova *task*.

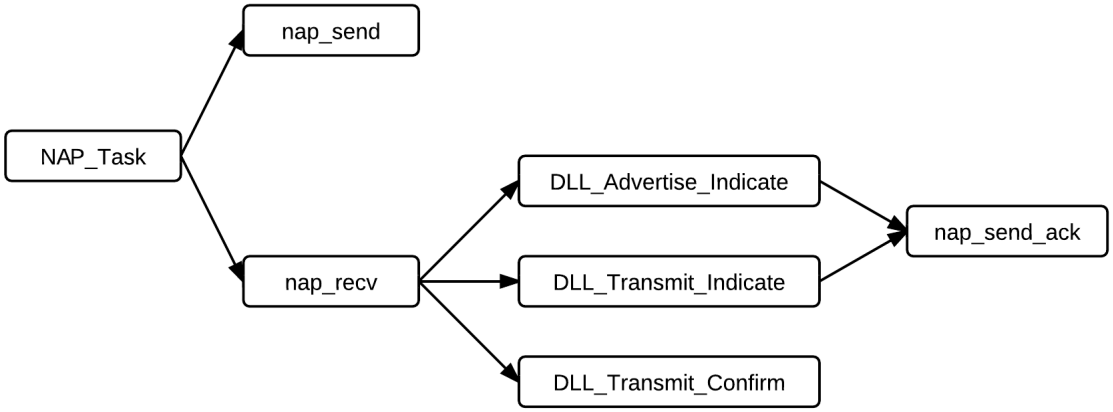

Figura 18: Estruturação da *task* do NAP.

Logo após a criação da *task*, ele deve esperar a inicialização da camada de enlace ser terminada. Desse modo, modificou-se essa parte da *task* da camada de enlace para que, além de avisar as outras *tasks* do sistema, avise também o NAP. Isso foi obtido utilizando-se a função do sistema operacional CMX, OS_Event_Signal, onde os parâmetros são o ID da *task* e o evento a ser informado. Pelo lado do NAP, utilizou-se a função OS_Event_Wait para esperar a atribuição do evento. Junto com a criação da *task*, criou-se um *mailbox* para recebimento de mensagens de outras *tasks*, além de ter-se estabelecido a comunicação serial entre rádio e host.

Com a *task* rodando, ela passa atuar em *polling*. Assim, a cada iteração ela verifica se uma nova mensagem chegou na *mailbox* advinda da camada superior, ou se recebeu alguma mensagem do *host*. Para a primeira opção, a *task* deve chamar imediatamente a função NAP_send, enquanto que, para a segunda opção, ela deve decodificar o frame HDLC antes de chamar a função NAP_recv. Na sequência, uma descrição detalhada das funções pertinentes é apresentada.

NAP_send: a função NAP_send tem o objetivo de transmitir para o *host* o NPDU enviado diretamente pela camada de rede, contendo as respostas dos dispositivos aos comandos enviados. Nessa função, o NPDU será empacotado em um novo frame DLPDU, visto que o NAP tem funcionalidades de camada de enlace, onde será

depois tratado pelo *gateway*. O cabeçalho desse novo frame é diferente do normatizado pelo padrão WirelessHART. A Tabela 4 apresenta os campos presentes no novo cabeçalho.

Campo	Descrição
type	Tipo do frame que será tratado pelo *gateway*. Nesta função, o tipo é sempre NAP_MSG_req, ou seja, uma requisição do dispositivo para o *gateway* (geralmente uma resposta a um comando).
lenght	Largura do NPDU mais o cabeçalho, de tamanho 5.
priority	Prioridade do NPDU. Assim como na norma WirelessHART, uma mensagem de comando tem prioridade mais alta, enquanto alarme mais baixa.
flags	Descreve o tipo de NPDU (ACK, *advertise*, *keepalive*, desconexão ou dados) e qual chave será usada para autenticar as mensagens (pública ou de rede).

Tabela 4: Cabeçalho do frame DLPDU enviado pelo NAP-RCP para o gateway.

O DLPDU gerado pelo NAP é então codificado usando HDLC e transmitido para o *host*.

NAP_recv: quando uma mensagem vinda do *host* é decodificada com HDLC, a *task* deve chamar esta função para lidar com o DLPDU recebido. O primeiro byte do cabeçalho é o tipo de mensagem. A função deve ser capaz de verificar o tipo de mensagem recebida e escolher uma forma de tratá-la.

O primeiro tipo de mensagem que o NAP-RCP deve interpretar é defina pela macro GTW_KEY_req. Essa mensagem é uma requisição do *gateway* para que o NAP-RCP faça o *join* dele na rede WirelessHART. A Tabela 5 apresenta o cabeçalho dessa mensagem. O NAP-RCP deve configurar as camadas de enlace e de rede, respectivamente, com o *network id* e a *join key* lidas do cabeçalho. Ele também deve verificar o ASN e inicializar o gerenciador de tempo. Também deve informar à camada de enlace o mapa de canais a ser utilizado na rede.

Para fazer o processo de *join*, a função também deve criar um pacote interno do tipo *advertise* que será enviado para a camada de rede pela camada de enlace, onde será montado o DLPDU no padrão WirelessHART, através da função DLL_Advertise_indicate.

Nesse pacote serão atribuídos os seguintes parâmetros: prioridade de *join*, identificador do grafo, *nickname* do gerenciador de rede (entidade que está, de fato, requisitando o *join* do NAP) e a potência do sinal (atribuído com valor máximo). Após o *join* do NAP na rede, ele pode começar a fazer o fluxo de mensagens do *gateway* para os dispositivos de campo, e vice-versa.

Campo	**Descrição**
type	Tipo do frame que será tratado pelo *gateway*. O tipo é sempre GTW_KEY_req, ou seja, que o NAP proceda o *join*
lenght	Largura do NPDU mais o cabeçalho.
network id	Identificação da rede a ser estruturada
join key	Chave de 128 bits que será utilizada para fazer o *join* dos dispositivos
slot	Absolute Slot Number
join priority	Prioridade do Join
channel map size	Tamanho do mapa de canais (em bits)
channel map	Mapa de canais. Cada bit diz se o canal está habilitado ou não
graph identifier	Identificador do grafo
superframes	Número de superframes

Tabela 5: Cabeçalho do frame DLPDU do tipo GTW_KEY_req recebido pelo NAP-RCP.

Outro tipo de mensagem que o NAP-RCP deve processar são as de requisições do *gateway*, que devem ser transmitidas para os dispositivos. Esse tipo de mensagem é definida pela macro GTW_MSG_req. O NAP deve apenas verificar o tipo de endereço que deverá constar na mensagem enviada para o dispositivo de campo (*nickname* (ou *short address*), grafos, ou endereços conhecidos (*gateway*, gerente de rede e *broadcast*)). A Tabela 6 descreve o cabeçalho desse DLDPU. Esses parâmetros, junto com o NPDU com os comandos HART enviados pelo gerente de rede, são enviados para a camada de rede através da função da camada de enlace DLL_Transmit_indicate.

A última mensagem que o NAP deve processar é a do tipo definida pela macro GTW_MSG_ack. O cabeçalho dessa mensagem consiste apenas desse campo, e seu NPDU é nulo. A confirmação do recebimento da mensagem por parte do

Campo	Descrição
type	Tipo do frame que será tratado pelo *gateway*. O tipo é sempre GTW_MSG_req, ou seja, que o NAP retransmita a mensagem para o dispositivo
lenght	Largura do NPDU mais o cabeçalho.
priority	Prioridade da mensagem
address type	Tipo do endereço de destino
address	Endereço de destino

Tabela 6: Cabeçalho do frame DLPDU do tipo GTW_MSG_req recebido pelo NAP-RCP.

gateway é informada à camada de rede através da função da camada de enlace DLL_Transmit_confirm.

O NAP também pode responder a requisição do ASN. Esse tipo de requisição não é passada para a camada superior, sendo respondida localmente pela *task* do NAP logo que recebida.

NAP_send_ack: após receber uma requisição do *gateway*, o NAP deve informá-lo que essa requisição foi recebida com sucesso. Para isso, ele deve retornar um ACK. O ACK consiste apenas de um frame contendo o tipo de ACK (GTW_KEY_ack, caso seja em resposta ao processo de *join*, ou GTW_MSG_ack, caso seja em resposta a algum comando) e NPDU nulo. A mensagem de ACK é respondida na própria *task*, não sendo, assim, enviada para a camada superior.

A Figura 19 ilustra o fluxo de mensagens entre as camadas que compõem o NAP, e também justifica o fato do NAP atuar na camada de enlace.

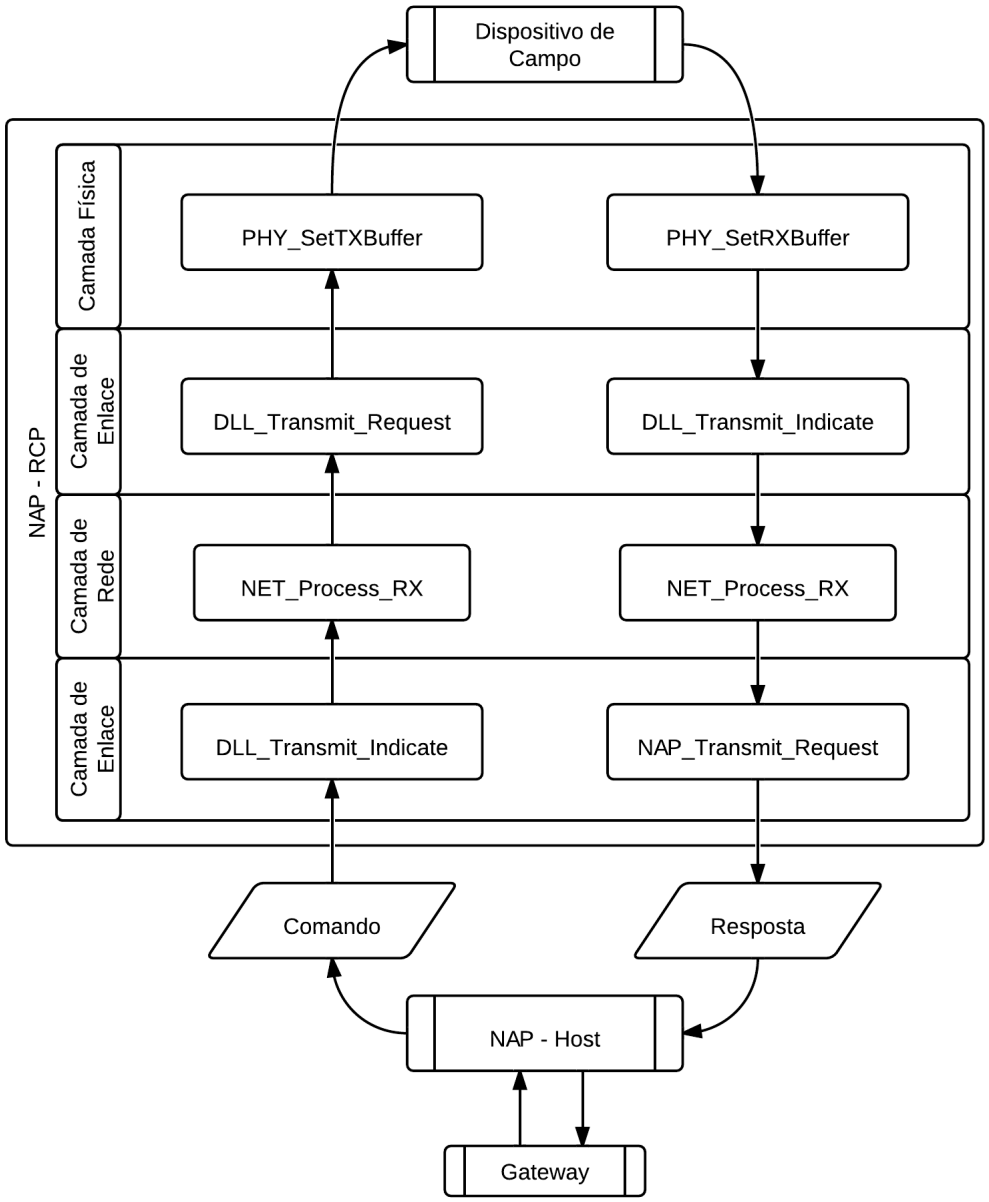

Figura 19: Fluxo de mensagens entre as camadas que compõem o NAP.

Como pode ser visto, a mensagem enviada pelo *gateway/host* chega direto à camada de enlace. Dentro da camada de enlace, a mensagem é retransmitida diretamente para a *mailbox* da camada de rede. A mensagem é então verificada pela camada de rede, onde deve ser decidido se a mensagem é para este dispositivo e, então, enviada para a camada de aplicação, onde será processada, ou se ela deve ser retransmitida. Como o ponto de acesso não possui camada de aplicação, já que os comandos enviados diretamente para ele são processados pelas camadas de enlace e rede, a primeira opção é descartada. Assim, a mensagem deve ser transmitida, ou para um dispositivo de campo, ou de volta para o

host/*gateway*/gerente de rede. Se for para um dispositivo, o fluxo segue normalmente, com a mensagem retornando à camada de enlace, depois é enviada para a camada física, onde será enfim transmitida para o dispositivo. Essa sequência é ilustrada pelo fluxo na parte esquerda da Figura 19. A parte direita da figura apresenta uma modificação realizada no *stack* WirelessHART fornecido. Se a mensagem deve ser enviada para o *host*, então ela deve ser encaminhada para a *task* do NAP, e não para a camada de enlace original. A função NAP_Transmit_request faz esse papel. Basicamente, ele envia o NPDU da camada de rede para a *mailbox* do NAP, permitindo, assim, que a função NAP_send seja processada mais a frente.

3.2 Gateway

Após a criação do ponto de acesso, foi necessário desenvolver um *gateway* virtual para, seguindo o diagrama de blocos apresentado no início do capítulo, fazer o fluxo de mensagens entre gerente de rede e ponto de acesso. O *gateway* também é responsável por fazer a conversão dos dados enviados pela planta para o padrão HART e manter uma tabela *cache* com as mensagens enviadas pelos dispositivos. Como o projeto tem apenas como objetivo montar uma simples rede WirelessHART, não será necessário implementar a troca de mensagens com a planta no *gateway* desenvolvido.

A entidade implementada roda em Linux, assim como o NAP-Host, além de fazer uso dos *stacks* WirelessHART, uma vez que o *gateway* deve responder a comandos do gerente de rede para atribuir alguns parâmetros, e não apenas encaminhar comandos para o ponto de acesso. Uma vez que o *stack* WirelessHART fora programado originalmente para plataformas embarcadas, foi necessário migrar as funções do sistema operacional para a biblioteca *pthreads* (POSIX Threads) do Linux. Também foram necessárias mudanças no que tange a distribuição dos pacotes, responsabilidade da camada de rede (CHEN *et al*, 2010), visto que o *gateway* deve saber se a mensagem é para ele, para o gerente de rede, ou para algum dispositivo de campo através do ponto de acesso.

A Figura 20 apresenta o diagrama hierárquico funcional implementado no *gateway*.

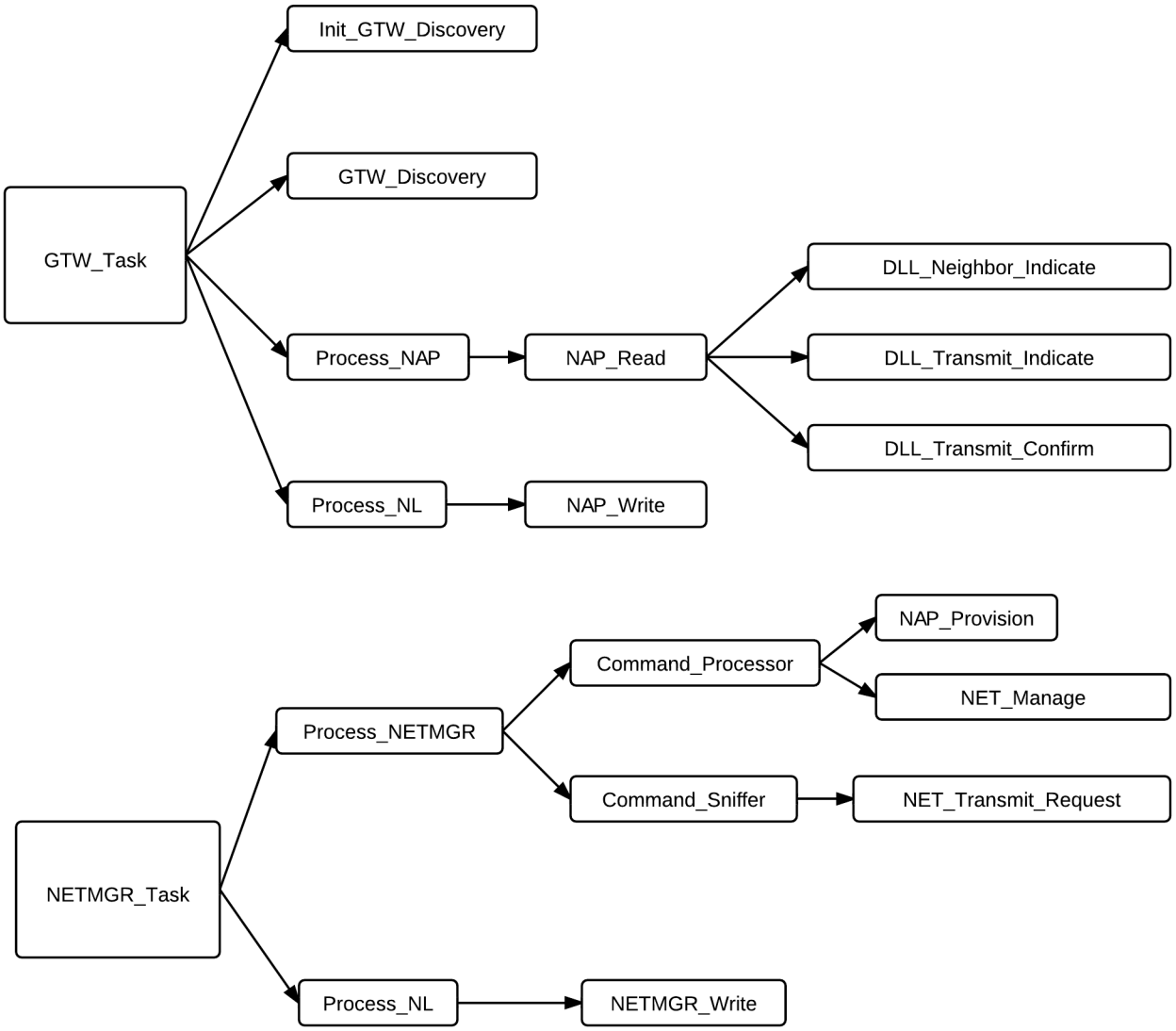

Figura 20: Estruturação das *tasks* do Gateway.

No momento em que o *gateway* é criado, é estabelecido um socket UDP multicast para que as outras entidades, gerente de rede e *host*, encontrem o *gateway*y. Após o recebimento da mensagem de descoberta de uma das entidades, o gateway deve validá-la e estabelecer um canal de conexão único com cada entidade. A Figura 21 apresenta um *log* feito pelo programa Wireshark mostrando a descoberta e estabelecimento da conexão entre o *gateway* e o *host*.

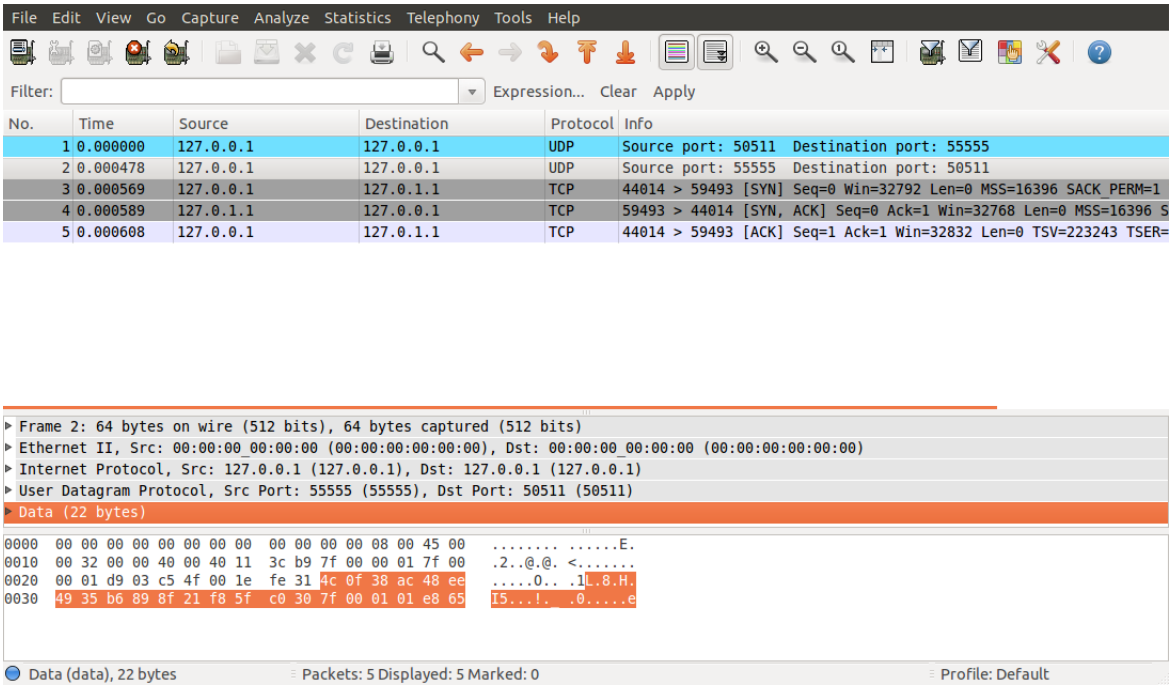

Figura 21: Conexão entre o Gateway e o NAP.

Após o estabelecimento das conexões com o gerente de rede e o NAP, o *gateway* deve processar os comandos 122, 768 e 773 enviados pelo gerente de rede e inicializar o *join* do NAP à rede. A Tabela 7 apresenta uma breve descrição sobre esses comandos.

Comando	**Descrição**
122	Comando não público. De acordo com a norma HART, deve ser utilizado apenas para testes de fábrica. Utilizou-se esse comando para informar ao NAP o mapa de canais.
768	*Write Join Key*. Escreve a chave utilizada para autenticar o processo de *join* dos dispositivos de campo.
773	*Write Network Identifier*. Escreve o identificador de rede a qual todos os dispositivos estão conectados.

Tabela 7: Sequência de comandos de *join* do NAP.

Com a realização bem sucedida do *join* do NAP, o *gateway* pode começar a fazer o fluxo de mensagens entre o gerente de rede e o ponto de acesso.

Na sequência, as *tasks* e funções apresentadas na Figura 20 são detalhadas.

3.2.1 Task do Gateway

Uma das duas principais *tasks* do *gateway*, é a responsável por processar as mensagens de descoberta das outras entidades, e encaminhar as mensagens da camada de rede para o ponto de acesso e vice-versa.

Init_GTW_Discovery: nesta função é criado o socket UDP multicast que será utilizado pelas outras entidades para descobrir o *gateway*.

GTW_Discovery: após o recebimento da mensagem de descoberta, ela é validada de modo que o *gateway* saiba com quem está fazendo a conexão. Isso é realizado verificando-se o campo de identificação que está na mensagem e comparando-o com os valores pré-definidos no código (0x1234 para o ponto de acesso e 0x5678 para o gerente de rede). Tendo a mensagem sido validada com sucesso, o *gateway* estabelece uma conexão unicast com a entidade que fez sua descoberta, como mostrada na Figura 21.

Process_NAP: com o *join* do NAP realizado, o *gateway* deve começar a receber as mensagens enviadas pelo ponto de acesso. O DLPDU recebido nessa função é o mesmo gerado pelo NAP, não sendo aquele utilizado na camada de enlace WirelessHART. Para saber o que fazer com cada mensagem recebida, elas possuem um cabeçalho que as diferenciam entre os tipos:

- **GTW_KEY_ack:** é o ACK do NAP à requisição GTW_KEY_req, ou seja, que seja iniciado o *join*. Ao receber esse tipo de mensagem, o *gateway* deve informar à camada de rede que o ponto de acesso está pronto para fazer o tráfego de informações. Isso é realizado através da função DLL_Neighbor_Indicate.
- **GTW_MSG_ack:** mensagem de ACK enviada pelo NAP quando este recebe uma requisição do tipo GTW_MSG_req. A camada de rede é informada sobre o sucesso do envio da requisição através da função DLL_Transmit_Confirm.
- **NAP_MSG_req:** quando o NAP envia de volta para o *gateway* uma resposta a algum comando. Antes de informar a camada de rede, onde a resposta será

redirecionada para o destino correto, o *gateway* deve gerar as *flags* que serão utilizadas pela camada de enlace para criar um DLPDU adequado para a camada de rede. Após isso, é chamada a função DLL_Transmit_Indicate.

Process_NL: durante a inicialização do *gateway*, são inicializados pares de sockets para comunicação entre as *tasks*. Em relação a camada de enlace, são inicializados dois pares, um para comunicação com a task GTW_task, e outro para comunicação com a *task* NETMGR_task. Esta função é a responsável por receber o DLPDU enviado pela camada de enlace através da camada de rede, interpretar o cabeçalho da mensagem e gerar um novo DLPDU utilizando apenas o NPDU da mensagem original. Esse novo pacote deve ser então, enviado para o ponto de acesso. Todas as mensagens processadas por essa estrutura são requisições aos dispositivos de campo, por isso recebem a identificação GTW_MSG_req. O cabaçalho detalhado desse tipo de mensagem pode ser visto na Tabela 6.

3.2.2 Task do Gerente de Rede

É uma *task* criada logo após o *gateway* ter estabelecido a conexão com o gerente de rede. Deve ser capaz de receber as mensagens vindas do gerente de rede e decidir se deve processá-las ou enviá-las para o ponto de acesso. Também é responsável por receber as mensagens da camada de rede e retorná-las ao gerente de rede.

Process_NETMGR: é a função responsável por receber os comandos do gerente de rede e decidir o destino deles. Esses comandos não são recebidos como DLPDU, ou NPDU, uma vez que o gerente de rede a ser projetado será considerado uma extensão da camada de aplicação. A única diferença nesses pacotes é que serão acrescidos de um cabeçalho dizendo o destino: 1 para que seja processado pelo próprio *gateway* e 2 para que sejam retransmitidos para os dispositivos de campo.

Caso o pacote enviado deva ser processado localmente, a função Command_Processor deve ser chamada. Por outro lado, se o pacote deve ser apenas retransmitido, a função Command_Sniffer será chamada.

Command_Processor: o processador de comandos do *gateway* será o responsável por configurar as camadas de enlace e de rede, além de montar as rotas que serão utilizadas até os dispositivos e criar as sessões para comunicação entre o gerente de rede e o *gateway*, e o *gateway* e os dispositivos de campo. Dessa maneira, projetou-se o *gateway* para que responda a pelos menos os comandos do gerente de rede descritos na Tabela 8.

Comando	Descrição
122	Comando não público. De acordo com a norma HART, deve ser utilizado apenas para testes de fábrica. Utilizou-se esse comando para informar ao NAP o mapa de canais.
768	*Write Join Key*. Escreve a chave utilizada para autenticar o processo de *join* dos dispositivos de campo.
773	*Write Network Identifier*. Escreve o identificador de rede a qual todos os dispositivos estão conectados.
963	*Write Session*. Este comando permite que o gerente de rede crie uma sessão entre o dispositivo o qual a mensagem é destinada e o dispositivo contido na requisição.
974	*Write Route*. Escreve a rota a ser utilizada até o dispositivo.

Tabela 8: Sequência de comandos processados pelo gateway.

Quando o processador de comandos recebe a trinca de comandos 122, 768 e 773, o *gateway* deve configurar o *netid* e a *join key* para iniciar o processo de *join* do NAP, iniciado a partir da função NAP_provision. Os demais comandos são interpretados e, no caso dos comandos que configuram a camada de rede, como o 963 e o 974, chamam a função NET_Manage para configurar a camada de rede.

Command_Sniffer: os comandos que não são direcionados para o *gateway*, são processados nessa função, que basicamente retransmite-os para os dispositivos. Uma vez que os comandos enviados para o *gateway* não possuem os cabeçalhos da camada de rede e de enlace, é nessa função que os parâmetros desses cabeçalhos são definidos, tais como tipo de endereço e tipo de requisição (*unicast* ou *broadcast*, por exemplo). Em seguida, o pacote com os comandos e a estrutura com as definições dos cabeçalhos são enviados para a camada de rede através da função NET_Transmit_Request,

onde será criado um NPDU adequado para a camada de enlace, que é substituída pela função NAP_write. Nessa função o DLPDU é montado e enfim enviado para o *host*.

Process_NL: é responsável por retornar os pacotes que se encontram na camada de rede de volta para o gerente de rede. Para isso, nessa função o NPDU recebido é interpretado e um novo pacote com os comandos HART é criado.

3.3 Gerente de rede

Para testar o ponto de acesso criado, necessitou-se desenvolver um *script* que suprisse as funções de gerente de rede, sendo capaz de montar uma rede WirelessHART pré-estipulada mais simples possível. Dessa maneira, o gerente de rede deverá se conectar ao *gateway*, configurá-lo, adicionar o NAP e um dispositivo à rede, enviar comandos e fechar a conexão (HCF SPEC-085, 2009). A figura 22 ilustra o fluxograma de passos do gerente de rede.

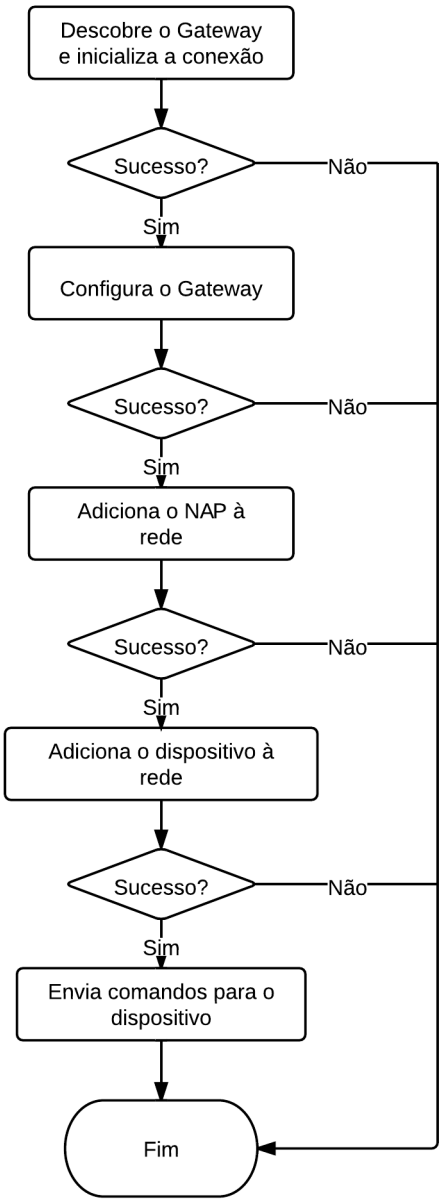

Figura 22: Fluxograma do gerente de rede.

3.3.1 Conexão com o Gateway

A conexão entre o gerente de rede e o *gateway* é realizada do mesmo modo que entre o *host* e o *gateway*. O gerente de rede descobre o *gateway* através das requisições *multicast* UDP e, na sequência, estabelece uma conexão *unicast* TCP.

3.3.2 Configuração do Gateway

Após as conexões entre *gateway* e gerente de rede, e *gateway* e *host* forem completadas, o gerente de rede irá configurar o *gateway* com alguns parâmetros, além de criar uma

sessão *broadcast* que será utilizada para enviar comandos a partir do *gateway* para todos os dispositivos de campo. Essa configuração é gerada através da seguinte sequência de comandos: 122, 768, 773, 963 e 974.

Assim, com os comandos 122, 768 e 773, o *gateway* recebe as informações sobre o mapa de canais que será utilizado, a *join key*, que será utilizada para autenticar os pedidos de *join*, e o *network id*, que identificará a rede. Como visto anteriormente, com esses parâmetros o *gateway* pode iniciar o *join* do ponto de acesso. Por fim, o *gateway* recebe o comando 963, para criar um sessão *broadcast*, e o comando 974, que conectará essa sessão a um grafo.

3.3.3 Adição do NAP à rede

Como visto na seção 3.1, o processo de *join* do NAP é iniciado com a injeção de um *frame* de *advertise* falso na camada de rede. O próximo passo para adicionar o NAP à rede é criar uma sessão de *join* entre o *gateway* e o NAP.

Estabelecida a sessão, o gerente de rede pode verificar o pedido de *join*, ou seja, as respostas aos comandos 0, que fornece o *unique id* do dispositivo, 20, que fornece a *long tag* do dispositivo, e 787, que fornece o número de vizinhos que o dispositivo escuta. No caso do NAP, ele não escuta nenhum vizinho durante o *join*. Desse modo, utiliza-se o *gateway* como proxy. Assim, o NAP poderá usar o *graph id* configurado no *gateway* para encaminhar as mensagens durante o processo de *join* para o gerente de rede.

Em seguida, deve-se estabelecer duas sessões *unicast*, uma para que o NAP receba as mensagens do gerente de rede, e outra para que o *gateway* receba as mensagens do NAP. Além disso, o comando 974 é enviado para estabelecer uma rota até o NAP.

Por fim, o gerente de rede responde ao pedido de *join* (*join request*) com os comandos 961, 962 e 963 (*join reply*). Dessa maneira, o gerente de rede configura a *network key* e o *nickname* do NAP, além de criar uma sessão *unicast* entre si e o NAP.

Com o NAP adicionado à rede com sucesso, o gerente de rede deve criar uma sessão para que o NAP receba as mensagens de *broadcast* do gateway. Na sequência, o gerente de rede deve configurar o NAP para que gere os *frames* de *advertisement*. Para isso, envia-se duas vezes o comando 965 para o NAP, sendo o primeiro para escrever um

superframe exclusivo para as conexões de *join*, enquanto o segundo será usado para as demais conexões. A estrutura do primeiro *superframe* é ilustrada na figura 23. Depois, é enviado três vezes o comando 967, Write Link, para que sejam criados, dentro do primeiro *superframe*, os *links* de recebimento e transmissão (além de um link *shared*). Para atribuir o intervalos de *advertisement*, o gerente de rede envia para o NAP o comando 795, Write Timer Interval, atribuindo um intervalo de meio segundo entre as mensagens de *advertise*. Como a rede montada terá apenas um dispositivo, optou-se por desabilitar o CCA do NAP pelo envio do comando 805, já que o envio de mensagens não sofrerá interferência.

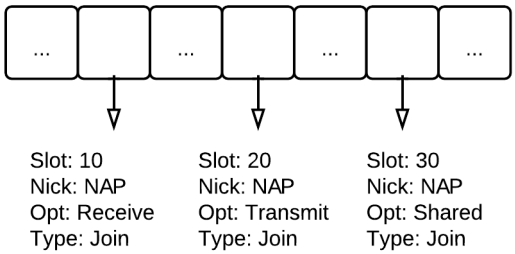

Figura 23: Configuração do superframe de join do NAP.

Após esse processo de configuração do ponto de acesso, o NAP estará pronto para ser descoberto pelos dispositivos de campo.

3.3.4 Adição do dispositivo à rede

O próximo passo é adicionar o dispositivo de campo à rede. Parte do processo é o mesmo realizado para com o NAP. Novamente, o gerente de rede cria no *gateway* uma sessão de *join*. Partindo do fato que o *advertisement* que chega ao dispositivo é verdadeiro, e não uma simulação como ocorre com o NAP, o dispositivo envia um pedido de *join* (comandos 0, 20, 787). O gerente de rede irá interpretar o 787 e verá que o dispositivo se conecta a ele através do ponto de acesso, que será utilizado como *proxy*. Duas novas sessões, mensagem do gerente de rede para o dispositivo, e mensagem do dispositivo para o *gateway*, serão criadas (comando 963) e conectadas a um grafo (comando 967). Após isso, o gerente de rede envia para o dispositivo a resposta ao *join* (comandos 961, 962 e 963). Com o comando 963, é estabelecido uma sessão *unicast* entre o gerente de rede e o

dispositivo. Assim, o processo de *join* estará terminado, restando ao gerente de rede fazer algumas configurações adicionais, tanto no NAP, quanto no dispositivo, para estabelecer uma conexão normal com o novo dispositivo.

No segundo *superframe* criado, logo após a adição do NAP, dois links são criados, um *shared* para enviar as primeiras requisições ao dispositivo, e um de leitura, para receber as respostas do dispositivo, como mostrado na Figura 24. Junto com esses dois comandos, são enviados os comandos 969 e 974. Com o 969, é criado uma aresta no grafo com o *nickname* do dispositivo, enquanto que, com o 974, o dispositivo é conectado ao grafo. Esses comandos são, então, empacotados e enviados para o NAP, de modo que ele possa estabelecer conexão com o dispositivo de campo.

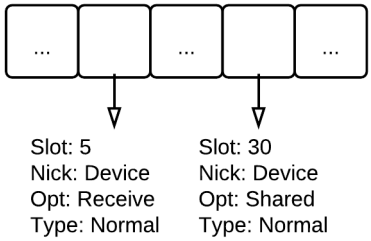

Figura 24: Configuração inicial do superframe do NAP.

Com o NAP configurado, o gerente de rede deve fazer mais algumas configurações adicionais no dispositivo de campo. O primeiro passo é escrever os *superframes* no dispositivo. Assim, uma vez que a rede está utilizando dois *superframes*, o comando 965 é enviado duas vezes, criando um *superframe* para *join* e outro para outros tipos de comunicação. No segundo *superframe*, são criados os *links* de recebimento e transmissão de mensagens vindas do NAP através do comando 967, ilustrado pela Figura 25. O NAP é conectado a um grafo com o comando 969, de modo que as mensagens do dispositivo saibam onde ele está, e o CCA é desabilitado. Esses comandos são, então, empacotados e enviados para o dispositivo.

O último passo é criar as sessões *broadcast* entre os dispositivos com o gerente de rede. Uma sessão de *broadcast* é criada entre o NAP e o gerente de rede, e são criados dois *links* no *superframe*: um para transmitir mensagens para o dispositivo, e outro para transmitir mensagens de *broadcast*. O *link shared* não é mais necessário, por isso é de-

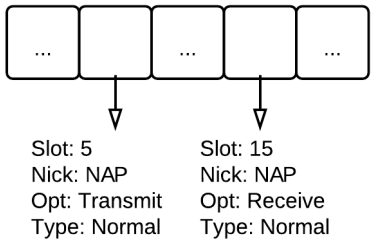

Figura 25: Configuração inicial do superframe do dispositivo.

letado com o comando 968. Então, toda essa sequência de configuração do *superframe* é enviada para o ponto de acesso. Por fim, o gerente de rede envia o comando para o dispositivo criar uma sessão *broadcast* com o gerente de rede, e para criar um *link broadcast* no *superframe* que já está gravado no dispositivo.

As Figuras 26 e 27 ilustram a configuração final dos *superframes* de comunicação normal gravados, respectivamente, no ponto de acesso e no dispositivo.

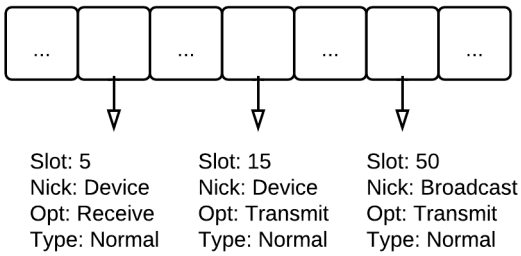

Figura 26: Configuração final do superframe do ponto de acesso.

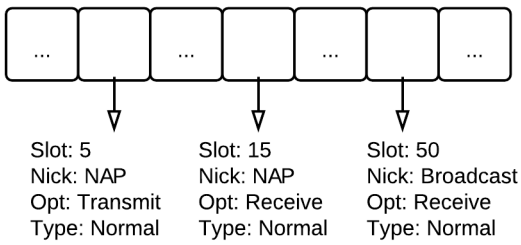

Figura 27: Configuração final do superframe do dispositivo.

3.3.5 Envio de comandos

Após a estruturação da rede ter sido completada, o dispositivo está pronto para responder aos comandos do gerente de rede. Para testes, optou-se por enviar dois comandos simples e sem parâmetros: 779 (*Report Device Health*) e 960 (*Disconnect Device*). A

Tabela 9, como vista em (HCF SPEC-155, 2008), apresenta uma breve descrição sobre esses comandos.

Comando	Descrição
779	*Report Device Health*. Comando que todos os dispositivos devem implementar para fornecer ao gerente de rede e à aplicação informações sobre as estatísticas da comunição dos dispositivos.
960	*Disconnect Device*. Este comando faz com que o dispositivo force sua saída da rede, apague as informações da rede e refaça o processo de *join*.

Tabela 9: Comandos enviados para o dispositivo.

3.4 Resultados

A Figura 28 apresenta, em termos práticos, o arranjo final da rede WirelessHART estabelecida neste trabalho.

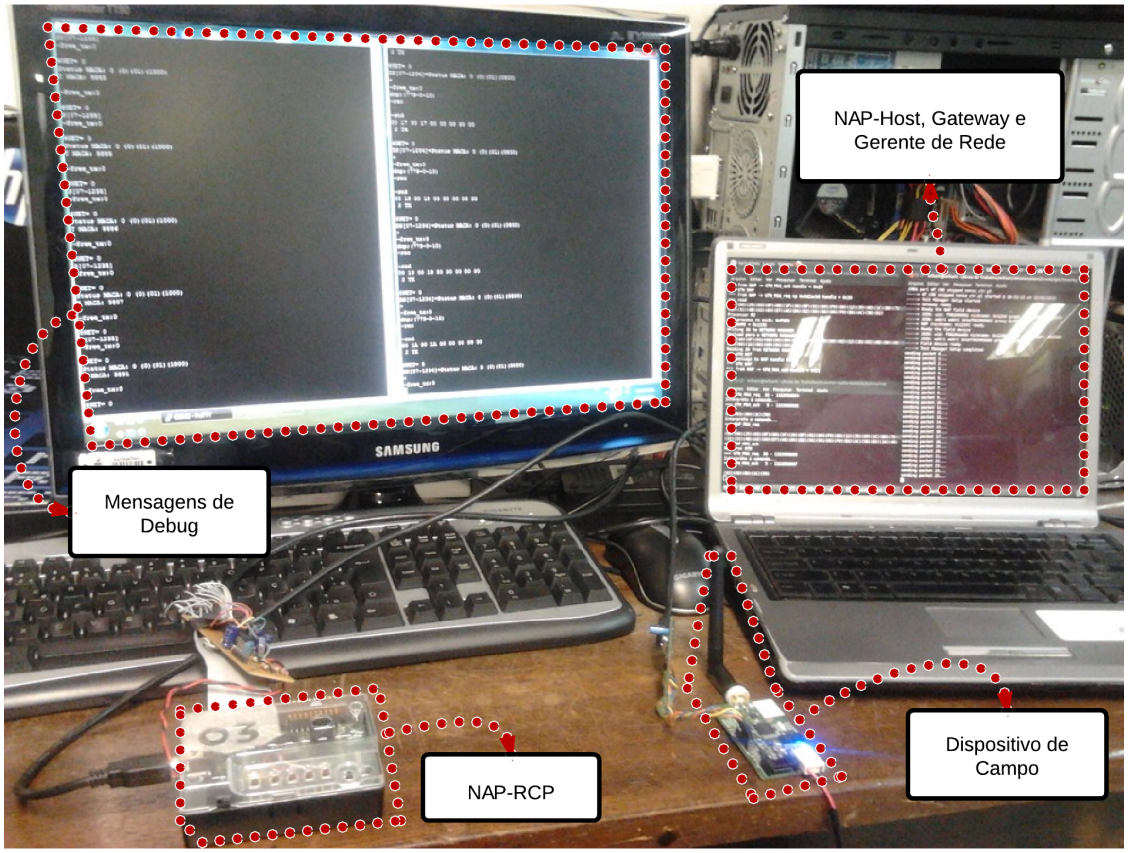

Figura 28: Rede WirelessHART estabelecida.

Após a conclusão das entidades, alguns problemas, apresentados na sequência, foram

observados e corrigidos:

Comunicação entre Host e RCP:

A plataforma MC1322x apresenta, no seu controlador UART, uma FIFO de 32 bytes. Desse modo, as mensagens entre as duas entidades estariam limitadas a tamanhos de 32 bytes, sendo necessário a leitura de um byte da FIFO para que outro espaço fosse liberado. A solução tomada foi, utilizando a interrupção de RX da UART, implementar uma FIFO na RAM. Assim, logo que o controlador UART recebesse um byte, esse byte é lido e gravado na RAM, esvaziando, assim, a FIFO. Essa solução foi baseada na implementada no sistema operacional Contiki OS para a plataforma MC1322x (www.contiki-os.org, 2011).

Dessincronização:

Outro problema visto foi a falta de sincronização entre o ponto de acesso e o dispositivo de campo. Desse modo, algumas mensagens enviadas pelo ponto de acesso para o dispositivo de campo chegavam no *slot* seguinte, ou eram até mesmo abortadas devido ao término do *slot* de 10ms. Resolveu-se isso revisando os tempos de cada operação, como mostrados na Tabela 2, e retirando toda mensagem de *debug* desnecessária, visto que a função *printf* é lenta e consome grande espaço do *stack*.

Com esses problemas resolvidos, a rede WirelessHART foi criada com sucesso. A sequência de comandos para fazer o *join* do dispositivo e inicializar as sessões entre o gerente de rede e o dispositivo mostrou-se correta, permitindo o fluxo de mensagens entre as duas pontas da rede. As respostas aos comandos 779 mostraram que as mensagens recebidas pelo dispositivo de campo não tiveram erros de MIC, nem de CRC, o que demonstra o funcionamento correto da rede.

Na sequência, é apresentado as mensagens geradas pelo gerente de rede, ilustrando, assim, a montagem passo-a-passo da rede.

```
wihart@wihart:~/Área de Trabalho/wihart/$ perl TestMgr.pl
-----> TestMgr.pl started @ 16:24:53 on 12/02/2011
-----> Test Manager Setup started
-----> Gateway ready
```

```
-----> Ready the NAP field device
-----> JOIN: uid: f982000001 nickname: 0x1234 graph
identifier: 0x0300
-----> JOIN: addr1 addr2 1b1ef982000001 proxy 0xf981
-----> NAP (nickname: 0x1234) ready
-----> Ready the field device
-----> JOIN: uid: f98200aabb nickname: 0x1235 graph
identifier: 0x0300
-----> JOIN: addr1 addr2 1b1ef98200aabb proxy 0x1234
-----> Field device ready
-----> Test Manager Setup completed
sending packet 0...
sending packet 1...
sending packet 2...
sending packet 3...
sending packet 4...
sending packet 5...
sending packet 6...
sending packet 7...
sending packet 8...
sending packet 9...
sending packet 10...
sending packet 11...
sending packet 12...
sending packet 13...
sending packet 14...
sending packet 15...
sending packet 16...
sending packet 17...
sending packet 18...
sending packet 19...
sending packet 20...
sending packet 21...
sending packet 22...
sending packet 23...
sending packet 24...
sending packet 25...
sending packet 26...
sending packet 27...
sending packet 28...
sending packet 29...
sending packet 30...
sending packet 31...
Disconnecting Device...
-----> TestMgr.pl PASSED
-----> TestMgr.pl ended   @ 16:27:42 on 12/02/2011
```

4 CONCLUSÃO

Este trabalho versa sobre o desenvolvimento de um ponto de acesso para redes WirelessHART. Para que se atingisse o objetivo principal, foi necessário, além do desenvolvimento do ponto de acesso, o desenvolvimento de um *gateway* e um gerenciador de rede, entidades básicas e fundamentais de uma rede do tipo WirelessHART.

Os dispositivos desenvolvidos foram testados de modo que uma rede WirelessHART básica pudesse ser verificada, através do envio de comandos fundamentais, de formação e manutenção de rede. Os testes executados permitiram observar a criação de uma rede WirelessHART composta pelos seus elementos fundamentais, o ponto de acesso, o *gateway*, o gerenciador e um ou mais dispositivos de campo.

A rede criada é compatível com a norma, pois propicia a comunicação com um dispositivo de campo comercial, desenvolvido para outro projeto sobre redes WirelessHART. A compatibilidade é notória uma vez que não foram realizadas alterações no dispositivo de campo de modo a adequá-lo ao projeto deste trabalho de conclusão.

Verificou-se a funcionalidade da rede através da observação dos dados de depuração, tanto numa extremidade (ponto de acesso) quanto na outra (dispositivo de campo). Nos testes realizados, não houve perda de pacotes, nem pacotes corrompidos e além dos comandos básicos de formação de rede, foram enviados pacotes de manutenção de comunicação.

Utilizando os resultados deste trabalho, pode-se, em trabalhos futuros, desenvolver um gerenciador de redes WirelessHART completo, capaz de escalonar um rede composta por múltiplos nós sensores.

REFERÊNCIAS

HART Communication Foundation; **WirelessHART Technology.** Disponível em: www.hartcomm.org/protocol/wihart/wireless_technology.html Acesso em dezembro de 2011.

HART Communication Foundation; **Command Summary Specification HCF SPEC-099, Revision 9.0**. 2007.

HART Communication Foundation; **Network Management Specification HCF SPEC-085, Revision 1.2**. 2009.

HART Communication Foundation; **Wireless Command Specification HCF SPEC-155, Revision 1.1**. 2008.

International Electrotechnical Comission; **IEC 62951 Ed 1.0: Industrial communication networks - Wireless communication network and communication profile - WirelessHART™**. 2010.

CHEN, D.; NIXON, M.; MOK, A.; **WirelessHART™: Real-Time Mesh Network for Industrial Automation**. Springer, 2010.

WINTER, J. M.; **Software de Análise de Roteamento de Dispositivos WirelessHART**. Porto Alegre, 2010.

SIMPSON, W. A.; **PPP in HDLC-like Framing**. RFC 1662, Daydreamer, 1994.

CMX COMPANY; **User Manual**. 1999

The Contiki Operating System; Disponível em: http://www.sics.se/~adam/contiki/docs/main.html Acesso em dezembro de 2011.