

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

FERNANDO AUGUSTO ALVARES DE CASTRO E SOUSA

**Testes de Robustez em uma Rede
WirelessHART**

Trabalho de Graduação.

Prof. Dr. João Cesar Netto
Orientador

Porto Alegre, junho de 2013.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Vice-Reitor: Prof. Rui Vicente Oppermann

Pró-Reitora de Graduação: Prof. Sérgio Roberto Kieling Franco

Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb

Coordenador do ECP: Prof. Marcelo Götz

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Chegando ao fim desta etapa da minha vida, gostaria de dedicar esse trabalho à minha família que sempre me incentivou e me forneceu ensinamentos valiosos nesses últimos seis anos.

Agradeço aos integrantes do LASCAR, que me forneceram todo o apoio possível e também um ambiente adequado para a realização deste trabalho.

Agradecimentos especiais para meu orientador professor João Cesar Netto, que com sua paciência e sabedoria transmitiu ensinamentos importantes, não só durante a elaboração deste trabalho, mas também em diversos momentos da faculdade. Também ao doutorando Ivan Müller que foi meu colega em discussões e na execução dos experimentos contidos no mesmo.

Por fim, aos meus colegas de trabalho, que tiveram de lidar com a minha ausência em muitos momentos do último semestre.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS	5
LISTA DE FIGURAS	6
LISTA DE TABELAS	7
RESUMO	8
ABSTRACT	9
1 INTRODUÇÃO	10
2 REDE WIRELESSHART	12
2.1 Padrão HART	12
2.1.1 Componentes de uma Rede.....	12
2.1.2 Camada de Enlace.....	17
2.2 Comandos utilizados	19
2.2.1 784 – <i>Read Link List</i>	19
2.2.2 786 – <i>Read Neighbor Property Flag</i>	19
2.2.3 797 – <i>Write Radio Power Output</i>	20
2.2.4 798 – <i>Read Radio Output Power</i>	20
2.2.5 968 – <i>Delete Link</i>	21
2.2.6 971 – <i>Write Neighbor Property Flag</i>	21
3 TESTES DE ROBUSTEZ	22
3.1 Diminuição da Potência de Transmissão	22
3.1.1 Introdução.....	22
3.1.2 Execução do Experimento.....	24
3.1.3 Resultados do Experimento.....	31
3.2 Atraso no Relógio de Referência	37
3.2.1 Introdução.....	37
3.2.2 Execução do Experimento.....	41
3.2.3 Resultados do Experimento.....	44
4 CONCLUSÃO	49
REFERÊNCIAS	51
ANEXO <ARTIGO TRABALHO DE GRADUAÇÃO 1>	53

LISTA DE ABREVIATURAS E SIGLAS

ACK	<i>Acknowledge</i>
ASN	<i>Absolute Slot Number</i>
CRC	<i>Cyclic Redundancy Check</i>
DLPDU	<i>Data-Link Protocol Data Unit</i>
HCF	<i>HART Communication Foundation</i>
ID	Identificador
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
LASCAR	Laboratório de Sistemas de Controle, Automação e Robótica
LNA	<i>Low Noise Amplifiers</i>
MCU	<i>Microcontroller Unit</i>
PLL	<i>Phase-Locked Loop</i>
PPM	Partes por milhão
RF	<i>Radio Frequency</i>
RTC	<i>Real-Time Clock</i>
SF	<i>Superframe</i>
SOC	<i>System-On-Chip</i>
TDMA	<i>Time Division Multiple Access</i>
UFRGS	Universidade Federal do Rio Grande do Sul
WH	<i>WirelessHART</i>

LISTA DE FIGURAS

Figura 2.1: Arquitetura de uma rede <i>WirelessHART</i>	12
Figura 2.2: <i>Gateway</i> utilizado nos experimentos realizados.....	13
Figura 2.3: Exemplo de dispositivo de campo utilizado nos experimentos	14
Figura 2.4: <i>Software</i> de controle da porta de manutenção	15
Figura 2.5: Exemplo da janela de depuração do dispositivo de campo.....	16
Figura 2.6: <i>Sniffer</i> Wi-Analys fornecido pelo LASCAR para a análise de pacotes.....	17
Figura 2.7: Ilustração dos <i>slots</i> integrando um <i>superframe</i>	18
Figura 3.1: Diagrama de blocos do CC2591.....	23
Figura 3.2: Impacto da temperatura e tensão de alimentação na potência de transmissão do CC2591	23
Figura 3.3: Impacto da frequência na potência de transmissão do CC2591	24
Figura 3.4: Formação da rede estável informada pelo <i>Smart Wireless Gateway</i>	25
Figura 3.5: Representação dos <i>links</i> de tipos normais na rede	28
Figura 3.6: Representação de uma rede <i>WirelessHART</i> em linha	29
Figura 3.7: Verificação dos <i>links</i> criados pelo gerenciador de rede.....	30
Figura 3.8: Representação de uma rede em linha sem <i>links</i> redundantes	30
Figura 3.9: Potência lida dos dispositivos na rede funcional.....	31
Figura 3.10: Representação de uma rede em linha com falha	31
Figura 3.11: Saída de depuração do dispositivo TAG1025	35
Figura 3.12: Percepção do gerenciador da rede após o experimento	35
Figura 3.13: Registro do <i>gateway</i> indicando a saída do rádio TAG 1025 da rede.....	36
Figura 3.14: Esquemático dos <i>clocks</i> a serem utilizados pelo MC1322x.....	38
Figura 3.15: Bancos de capacitores ligados ao oscilador de referência.....	39
Figura 3.16: Erro do oscilador de referência em PPM vs XTAL_CTUNE[2:0]	40
Figura 3.17: Erro do oscilador de referência em PPM vs XTAL_FTUNE[4:0]	40
Figura 3.18: Código adicionado chamado quando o dispositivo recebe o comando 127	41
Figura 3.19: Código que efetua o atraso no oscilador de referência	42
Figura 3.20: Código adicionado ao <i>software</i> da porta de manutenção	43
Figura 3.21: Topologia analisada na falha de defasagem do <i>clock</i>	44
Figura 3.22: Correção do relógio (us) X nº da amostra	45
Figura 3.23: Correção do relógio (us) X nº da amostra, com CTUNE = 4 e FTUNE = 21	45
Figura 3.24: Correção do relógio (us) X nº da amostra, com CTUNE = 4 e FTUNE = 26	46
Figura 3.25: Correção do relógio (us) X nº da amostra, com CTUNE = 4 e FTUNE = 31	46
Figura 3.26: Correção do relógio (us) X nº da amostra, com CTUNE = 6 e FTUNE = 22	47
Figura 3.27: Saída da depuração do dispositivo TAG 1025 quando CTUNE = 6 e FTUNE = 22	47

LISTA DE TABELAS

Tabela 2.1: Legenda da janela de depuração do dispositivo de campo	15
Tabela 2.2: Comando 784 – Bytes de dados do pedido	19
Tabela 2.3: Comando 784 - Bytes de dados da resposta.....	19
Tabela 2.4: Comando 786 - Bytes de dados do pedido	20
Tabela 2.5: Comando 786 - Bytes de dados da resposta.....	20
Tabela 2.6: Comando 797 - Bytes de dados do pedido	20
Tabela 2.7: Comando 797 - Bytes de dados da resposta.....	20
Tabela 2.8: Comando 798 - Bytes de dados do pedido	20
Tabela 2.9: Comando 798 - Bytes de dados da resposta.....	20
Tabela 2.10: Comando 968 - Bytes de dados do pedido	21
Tabela 2.11: Comando 968 - Bytes de dados da resposta.....	21
Tabela 2.12: Comando 971 - Bytes de dados do pedido	21
Tabela 2.13: Comando 971 - Bytes de dados da resposta.....	21
Tabela 3.1: Resposta do comando 784 do dispositivo TAG 3333 e ID 2	25
Tabela 3.2: Resposta do comando 784 do dispositivo TAG 1025 e ID 3	26
Tabela 3.3: Resposta do comando 784 do dispositivo TAG 1008 e ID 5	26
Tabela 3.4: Resposta do comando 784 do dispositivo TAG 1000 e ID 6	27
Tabela 3.5: Resposta do comando 784 do dispositivo TAG 1015 e ID 7	27
Tabela 3.6: Saída do <i>sniffer</i> em uma rede em linha sem falhas	32
Tabela 3.7: Saída do <i>sniffer</i> durante o envio do comando 797	32
Tabela 3.8: Saída do <i>sniffer</i> em uma rede em linha com falha	33

RESUMO

A utilização de redes sem fio em ambientes industriais está cada vez maior. Suas vantagens podem ser vistas desde a redução de custo da instalação dos dispositivos, até comunicação com locais de difícil acesso. Em um ambiente industrial, uma rede de monitoração ou controle deve ser robusta tanto contra a influência de efeitos externos como contra ataques à segurança da rede. Esses fatores devem ser analisados cuidadosamente antes da utilização de redes sem fio em ambientes hostis.

Esse projeto visa realizar testes sobre o comportamento de uma rede *WirelessHART* contra erros que podem ocorrer de forma natural nos dispositivos de campo. Serão analisadas duas principais falhas, que em determinadas situações podem ser catastróficas para o sistema todo, para uma parte dele ou apenas a um dispositivo isolado.

Palavras-Chave: WirelessHART, robustez, testes, redes industriais sem fio.

Fault Injection into a WirelessHART Network

ABSTRACT

The use of wireless networks in industrial environments is increasing. Its advantages can be seen from the reduced cost of devices' to ease of installation of devices in environments that are hard to reach. But there are several factors to consider. In an industrial environment, a monitoring or control network must be robust in both security of data transmitted and strenght againt harmful external effects.

This project aims to conduct tests of the behavior of a WirelessHART network against faults that may occur naturally on field devices. Two major flaws will be analyzed, consisting of errors that, in certain situations, can be catastrophic for the entire system, a part of it or for an isolated device.

Keywords: WirelessHART, robustness, test, industrial wireless networks.

1 INTRODUÇÃO

Cada vez é maior a utilização de tecnologias *wireless* em ambientes industriais. Este incremento se deve a alguns fatores principais.

Indústrias muitas vezes utilizam múltiplos sensores para gerenciar seu processo de produção. O custo cada vez menor de componentes como controladores, pontos de acesso e sensores sem fio contribui para o aumento da sua utilização. A maior variedade disponível destes componentes no mercado também é um fator relevante já que mostra o gradual avanço e consolidação de um determinado tipo de tecnologia.

Outro ponto relevante é a robustez do sistema. Por muito tempo, redes cabeadas foram utilizadas devido a sua confiabilidade na transmissão de dados, já que são menos suscetíveis a interferências externas. Contudo, o avanço de protocolos de comunicação permitiu que redes sem fio se tornassem robustas mesmo em ambientes hostis.

Outro fator importante é o de que indústrias utilizam múltiplos sensores para gerenciar seu processo de produção. Redes sem fio permitem a adição, remoção e movimentação de sensores mais facilmente e, por vezes, em locais onde redes com fio não teriam acesso.

Na banda de 2.4Ghz podem ser encontrados três tipos principais de padrões para redes *wireless*: IEEE 802.15.1(Bluetooth), IEEE 802.11 (WiFi) e IEEE 802.15.4 (ZigBee), o qual é utilizado pela norma *WirelessHART*. [HART, 2010]

Cada um destes padrões possui diferentes características, distinguindo-se principalmente na velocidade dos dados transmitidos e na distância máxima de comunicação. Bluetooth possui uma comunicação com velocidade limitada em uma distância relativamente pequena, enquanto WiFi provê uma rede sem fio de mais alta velocidade à uma distância de nível intermediário. ZigBee é um padrão alternativo às duas soluções citadas anteriormente com uma baixa velocidade de transmissão podendo comunicar a médias distâncias. O problema dessas redes é que não possuem a robustez e a segurança desejada em ambientes industriais. [HART Communication Foundation, 2010]

Neste trabalho será realizado um estudo sobre a robustez [AVIZIENIS, 2004] de uma rede *WirelessHART* construída, mais especificamente, com os componentes de rede fornecidos pelo Laboratório de Sistemas de Controle, Automação e Robótica da Universidade Federal do Rio Grande do Sul (LASCAR).

Redes *wireless* possuem uma sensibilidade muito grande às interferências causadas pelo ambiente em que estão situadas, podendo ser afetadas por campos eletromagnéticos, colisão com outras redes sem fio, ou à existência de objetos bloqueando o caminho de comunicação.

O padrão *WirelessHART* é uma solução confiável para utilização em ambientes hostis tais como os do mundo industrial.

Introduzido ao mercado em 2007 e criado por especialistas da indústria, o padrão oferece oportunidades para aplicações desde medições, segurança e produtividade dos trabalhadores até rastreamento de pessoal. A especificação do *WirelessHART* é focada nas funções principais de automação aonde não existiam padrões sem fio apropriados.

Este estudo consistirá inicialmente na análise do protocolo *WirelessHART*, seguido da implementação de dois casos de teste. No primeiro caso, é injetado um atraso no relógio de referência de um dispositivo de campo, enquanto no segundo caso, uma falha na potência de transmissão de pacotes é injetada. Por fim, será realizada uma conclusão indicando os pontos principais, como a fragilidade e a força do sistema, observados durante a execução dos casos de teste.

As falhas contempladas permitirão verificar se o sistema é bem sucedido em identificar um erro, suportá-lo e se recuperar dele. O registro dos resultados obtidos também será importante para auxiliar equipes de diagnóstico e manutenção a recuperar mais rapidamente o sistema de uma situação de falha, permitindo também a identificação de possíveis melhorias em futuras implementações.

É importante lembrar que o conhecimento sobre a robustez de um padrão influi em decisões importantes na implantação da rede como o número de dispositivos de campo que ela deve ter para cumprir com os requisitos desejados. Esse número, porém não pode ser superdimensionado já que afeta diretamente o custo e o tempo de implantação da rede.

Esse trabalho foi realizado utilizando o *Smart Wireless Gateway* da Emerson [EMERSON, 2013], juntamente com dispositivos de campo e uma *suíte* de softwares desenvolvidos pelo LASCAR.

2 REDE WIRELESSHART

Neste capítulo estão contidos dados essenciais para a compreensão dos testes de robustez realizados, que foram retirados das especificações do protocolo [HART, 2011] e dos manuais dos dispositivos utilizados durante os experimentos.

2.1 Padrão HART

2.1.1 Componentes de uma Rede

Uma rede *WirelessHART* pode ser composta de diversos elementos como *gateway*, pontos de acesso, gerenciadores de rede, dispositivos de campo, *handhelds* e adaptadores HART. Os dispositivos que realizam comunicação sem fio na rede podem ser identificados através de identificadores únicos (IDs) que lhes são atribuídos pelo gerenciador da rede ou através de nomes programados nos dispositivos, que serão identificados durante este trabalho como TAGs. Uma distribuição típica da rede pode ser vista na Figura 2.1 [HART, 2010].

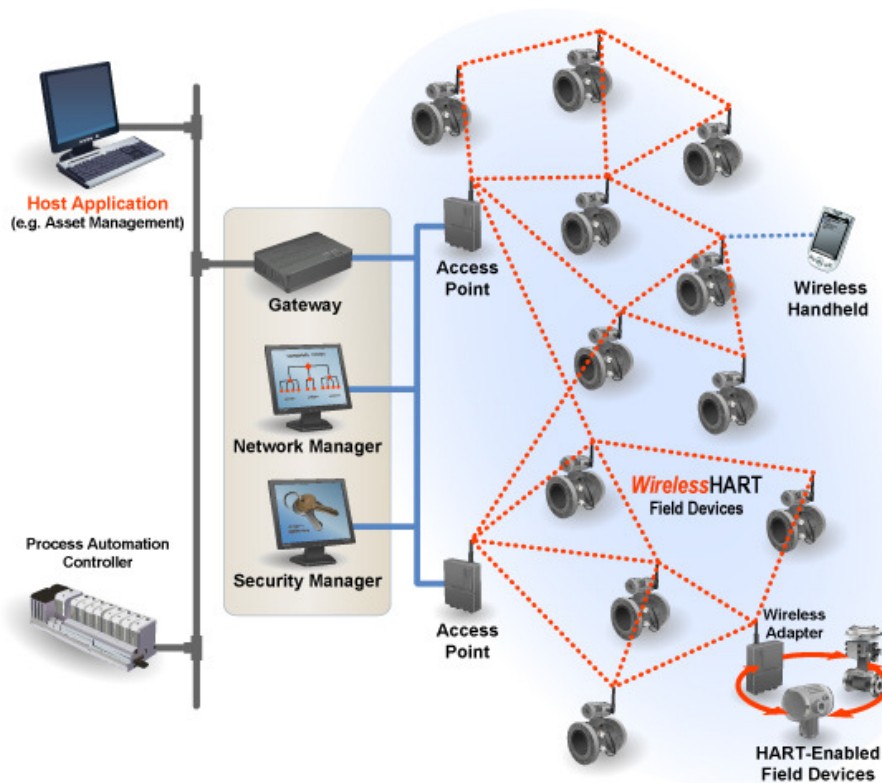


Figura 2.1: Arquitetura de uma rede *WirelessHART*

O nível de tolerância a falhas de uma rede é altamente dependente do seu formato, do número de nodos e da redundância de comunicação entre os dispositivos. O formato da rede varia com o gerenciador de rede utilizado, porém mesmo com um mesmo gerenciador ela pode adquirir organizações lógicas diferentes de acordo com a sua disposição e interferências externas. Para aumentar a disponibilidade da rede, o gerenciador cria caminhos redundantes, lógicos e temporais, entre todos os rádios, que podem ser identificados como os componentes em maior número na Figura 2.1. Uma redundância lógica existe quando existe mais de um caminho entre um rádio e outro. A redundância temporal existe quando múltiplos *time slots* são definidos para um mesmo caminho reduzindo os efeitos de interferências momentâneas.

Existem mecanismos para identificar erros de transmissão que não dependem da organização da rede. Um exemplo é o CRC que é transmitido ao final de cada mensagem e pode identificar rapidamente se houve algum erro de transmissão.

Este trabalho foca três elementos principais: o *gateway*, o gerenciador de rede e dispositivos de campo. Também será descrito um elemento passivo existente, o *sniffer*, que foi utilizado durante os experimentos realizados.

Gateway

Gateway é o elemento central de uma rede *WirelessHART*. Uma rede deve conter ao menos um *gateway*, pois ele viabiliza a comunicação entre os outros elementos da rede. No âmbito deste trabalho, foi utilizado o *Smart Wireless Gateway* da Emerson que consiste em três elementos da rede em conjunto, o *gateway*, o gerenciador de rede e o ponto de acesso. O *gateway* possui um software de gerencia através do qual é possível enviar comandos *HART* definidos na norma para rádios conectados. Essa interface será utilizada para configurar a rede como desejado.



Figura 2.2: Gateway utilizado nos experimentos realizados

Todas as relações entre elementos da rede dependem do *gateway*. No caso da alteração do mesmo, resultados diferentes podem ser obtidos.

Dispositivo de Campo

O dispositivo de campo utilizado foi desenvolvido pelo LASCAR [MULLER, 2010] e é o elemento presente, geralmente, em maior número em uma rede *WirelessHART*. Esse é o componente distribuído pelo ambiente industrial. Dispositivos de campo são responsáveis por realizar a publicação de suas variáveis de processo e a transmissão de pacotes entre dois pontos distintos na rede.

As variáveis de processo podem ser diversas dependendo do dispositivo utilizado, podendo ser a temperatura de um ponto de uma indústria ou a abertura atual de uma válvula de uma tubulação. Os dispositivos de campo são programados para realizar publicações dessas variáveis periodicamente. Para que possam fazer isso, o gerenciador de rede irá configurar *links* de conexão entre os rádios.



Figura 2.3: Exemplo de dispositivo de campo utilizado nos experimentos

O dispositivo, que pode ser visto na Figura 2.3, utiliza o *SoC* MC1322x que está de acordo com o padrão da IEEE 802.15.4. Ele possui um processador ARM7, aceleração de *hardware* para operações de criptografia dos pacotes transmitidos e vários MCU [FREESCALE, 2010].

O *firmware* utilizado é desenvolvido e mantido pelo LASCAR, portanto alterações necessárias para a campanha de testes são possíveis, porém exigirão uma compreensão do código, já que sua documentação não está disponível *online*.

O dispositivo de campo também possui uma porta de manutenção que permite o envio local de comandos *HART* para o rádio, sem que passe pela camada de enlace. Isso permite realizar a configuração inicial do rádio, operações de manutenção e outras operações locais no dispositivo.

Para enviar comandos pela porta de manutenção foi disponibilizado um software de comunicação, desenvolvido em outro trabalho [LIMA, 2011], que possui a implementação de diversos comandos *HART* (Figura 2.4). Esse *software* é flexível e permite a criação de outras funções para controle do rádio durante os testes realizados.

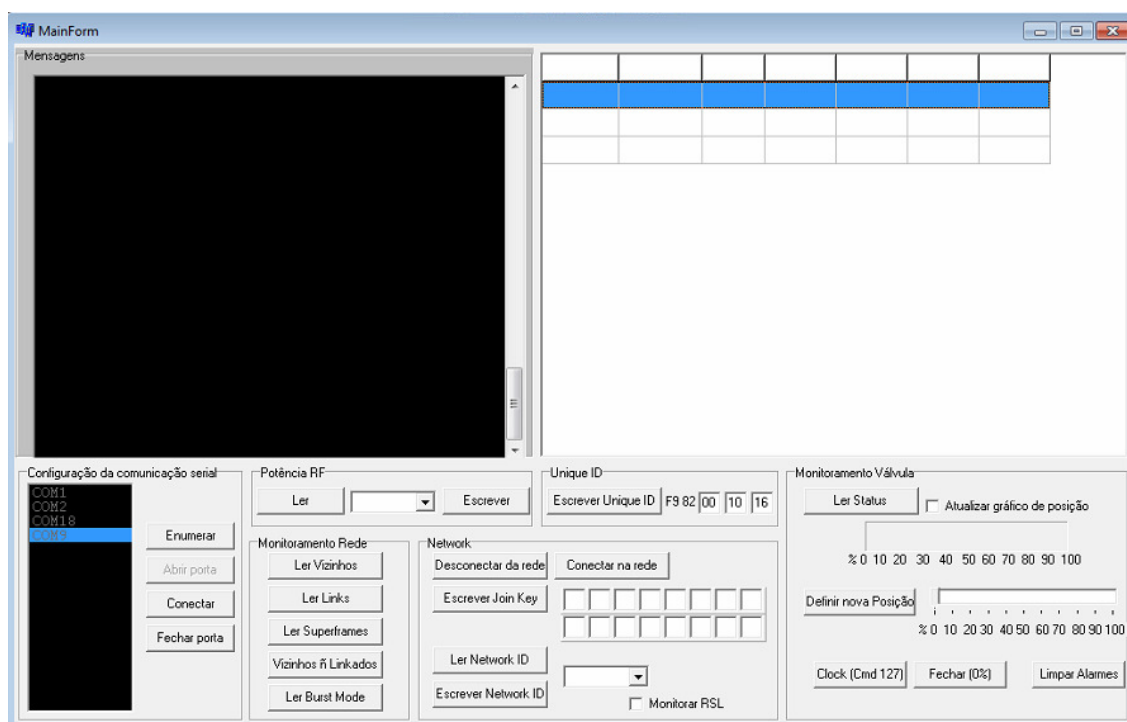


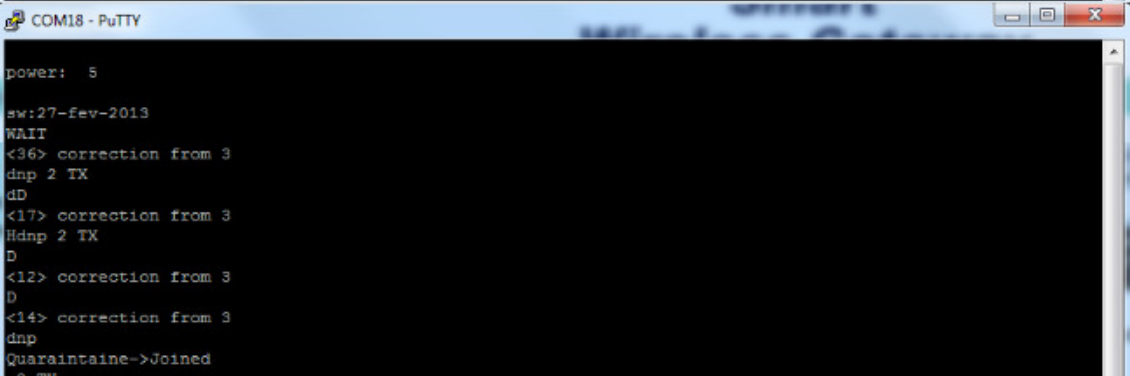
Figura 2.4: Software de controle da porta de manutenção

A depuração do dispositivo de campo pode ser realizada por uma porta serial ligada a um computador comum. No *firmware* podem ser adicionadas linhas que auxiliem o desenvolvedor a visualizar ações que ocorrem no dispositivo. Na Tabela 2.1 podemos ver a legenda da depuração para os elementos relevantes ao trabalho.

Tabela 2.1: Legenda da janela de depuração do dispositivo de campo

Texto de depuração	Descrição	Notas
B	Início de um <i>burst</i> de dados	
D	Transmissão de um pacote de dados no <i>time slot</i> atual.	Somente na transmissão de pacote de dados. Outros pacotes como <i>keepalives</i> e <i>advertisements</i> não são indicados por este texto.
K[003-1]	Indica que uma mensagem de <i>keepalive</i> será transmitida para o dispositivo com ID 3.	
<23> correction from 3	Uma mensagem de confirmação foi recebida e um ajuste de 23µs foi realizado no <i>clock</i> do dispositivo.	Essa mensagem só aparece quando a confirmação é proveniente de um dispositivo <i>time source</i> . Se o valor é positivo o pacote chegou mais tarde do que o esperado.

+	Uma mensagem de confirmação foi recebida, porém nenhum ajuste foi realizado no <i>clock</i> .	
@0	Não foi recebida uma mensagem de confirmação para o pacote enviado em um determinado <i>time slot</i> .	
ASN: 201254	<i>Absolute Slot Number</i> de valor 201254.	



```

COM18 - PuTTY
power: 5
sw:27-fev-2013
WAIT
<36> correction from 3
dnp 2 TX
dD
<17> correction from 3
Hdnp 2 TX
D
<12> correction from 3
D
<14> correction from 3
dnp
Quaraintaine->Joined

```

Figura 2.5: Exemplo da janela de depuração do dispositivo de campo

Na Figura 2.5 vemos um exemplo da saída da porta serial do dispositivo de campo através do *software putty*. Podemos ver informações sobre a inicialização do dispositivo, e dados de depuração. Neste caso temos informações de envio de pacotes de dados juntamente com a correção de tempo realizada após o recebimento da confirmação enviada pelo dispositivo destino.

Sniffer

O *sniffer* Wi-Analys [HAN, 2009], mostrado na Figura 2.6, foi disponibilizado pelo LASCAR para a verificação dos pacotes transmitidos na rede. Ele consiste em um dispositivo de campo que registra os pacotes transmitidos pelos seus vizinhos. Como a comunicação é encriptada, o *sniffer* deve ser ligado antes dos rádios que terão seus pacotes registrados. Isso porque o *sniffer* precisa escutar as chaves privadas da comunicação entre nodos para que consiga decifrar as mensagens transmitidas.



Figura 2.6: *Sniffer* Wi-Analys fornecido pelo LASCAR para a análise de pacotes

Através desse dispositivo é possível identificar se há problemas no envio de um DLPDU, na sua recepção ou no processamento. Também é possível avaliar a carga atual da rede.

2.1.2 Camada de Enlace

O padrão *WirelessHART* utiliza TDMA em conjunto com salto de canais para realizar a comunicação entre dispositivos de campo. Essa técnica é comumente utilizada para realizar a transmissão de pacotes de maneira determinística em uma rede sem fio. A comunicação é feita em espaços de tempos definidos, denominados como *time slots*. Quando agrupamos uma série de *time slots* formamos um *superframe*, começando pelo *superframe* com identificação zero.

A norma define que todos os dispositivos da rede devem suportar múltiplos *superframes* onde, pelo menos um está sempre ativo enquanto outros podem ser habilitados ou desabilitados de acordo com a necessidade. O tamanho dos *time slots*, assim como o dos *superframes*, é constante enquanto estes estão ativos e pode ser modificado enquanto inativo. A transmissão de *superframes* forma um ciclo de transmissão que se repete continuamente. A Figura 2.7 ilustra a relação entre *time slots* e um *superframe*.

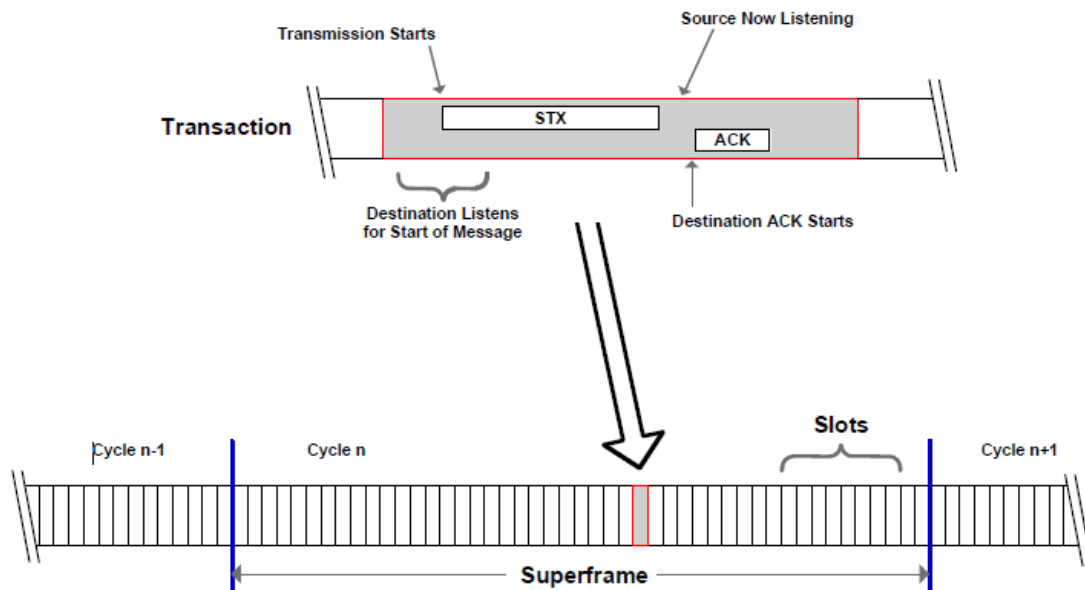


Figura 2.7: Ilustração dos *slots* integrando um *superframe* [HART, 2008]

Normalmente dois dispositivos de campo utilizam um mesmo *time slot* simultaneamente, sendo um a fonte da comunicação e outro o destino, consistindo uma transação. Uma transação suporta a transmissão de um pacote de dados (DLPDU) seguido imediatamente por uma confirmação de recebimento (ACK), que pode conter uma resposta de sucesso, indicando que a transmissão e o processamento pelo dispositivo de destino ocorreram de acordo com o esperado, ou um código de erro indicando algum problema, que pode ser causado por uma falha na rede, por um erro no processamento ou na transmissão do pacote. Um exemplo de erro que pode acontecer é um *overflow* em um *buffer* de recepção, não permitindo que o dispositivo destino guarde a mensagem para posterior processamento.

Existe também a possibilidade que mais de um rádio aguarde dados em um determinado *time slot*. Neste caso o link configurado é do tipo *broadcast* e não recebe uma resposta dos rádios receptores.

Como o TDMA depende que um dispositivo esteja transmitindo em um determinado momento e outro esteja aguardando uma mensagem ao mesmo tempo, ele se torna extremamente dependente da sincronização dos relógios entre os equipamentos. Conseqüentemente a tolerância ao erro nessa sincronização é especificada na norma para garantir o bom funcionamento da rede. Por esta razão, determinados dispositivos de rede são marcados como *time sources*. Esses dispositivos serão responsáveis por realizar a difusão da referência de tempo através da rede.

A transmissão da mensagem começa em um tempo determinado após o início do *time slot*. Este pequeno atraso permite que o rádio fonte e o alvo definam a frequência do canal de transmissão e que o receptor comece a escutar nele. Já que existe uma tolerância nos relógios, o receptor precisa começar a escutar antes do momento ideal do início da transmissão. Quando a mesma estiver completa, sua direção é invertida e o nodo receptor envia uma confirmação do recebimento da DLPDU.

2.2 Comandos utilizados

Abaixo é realizada uma descrição dos comandos, encontrados em [HART, 2008], mais importantes utilizados para o controle da rede durante os testes descritos na sessão 3. As descrições dos comandos servem como referência para as sessões subseqüentes deste trabalho.

2.2.1 784 – Read Link List

Este comando pode ser utilizado pelo gerenciador de rede para recuperar informações sobre a tabela de *links* de um dispositivo. Na Tabela 2.2 e Tabela 2.3 temos os bytes de pedido e resposta desse comando.

Tabela 2.2: Comando 784 – Bytes de dados do pedido

Byte	Formato	Descrição
0-1	<i>Unsigned-16</i>	Índice dos <i>links</i>
2	<i>Unsigned-8</i>	Número de <i>links</i> a serem lidos

Tabela 2.3: Comando 784 - Bytes de dados da resposta

Byte	Formato	Descrição
0-1	<i>Unsigned-16</i>	Índice dos <i>links</i>
2	<i>Unsigned-8</i>	Número de <i>links</i> lidos
3-4	<i>Unsigned-16</i>	Número de <i>links</i> ativos
5	<i>Unsigned-8</i>	ID do <i>superframe</i>
6-7	<i>Unsigned-16</i>	Número do <i>slot</i> deste <i>link</i> no <i>superframe</i>
8	<i>Unsigned-8</i>	<i>Offset</i> do canal para este <i>link</i>
9-10	<i>Unsigned-16</i>	<i>Nickname</i> do vizinho (0xFFFF para <i>broadcast</i>)
11	<i>Bits-8</i>	Opções
12	<i>Enum-8</i>	Tipo de <i>link</i>
13- ...		Os <i>bytes</i> de 5-12 serão repetidos para o número de entradas indicado em 2

2.2.2 786 – Read Neighbor Property Flag

Este comando permite ler as propriedades de um vizinho de um dispositivo. Este comando é utilizado neste trabalho para verificar se um dispositivo foi marcado como *time source* pelo gerenciador de rede ou não. Na Tabela 2.4 e Tabela 2.5 temos os bytes de pedido e resposta desse comando.

Tabela 2.4: Comando 786 - Bytes de dados do pedido

Byte	Formato	Descrição
0-1	<i>Unsigned-16</i>	<i>Nickname</i> do vizinho

Tabela 2.5: Comando 786 - Bytes de dados da resposta

Byte	Formato	Descrição
0-1	<i>Unsigned-16</i>	<i>Nickname</i> do vizinho
2	Bits	Propriedades do vizinho (0x1 quando for <i>time source</i>)

2.2.3 797 – Write Radio Power Output

Este comando é utilizado para configurar a potência de saída de um dispositivo ou do gateway. No caso de o dispositivo não aceitar o valor fornecido como entrada ele irá interpretar como se fosse o valor mais próximo possível ao enviado. Na Tabela 2.6 e Tabela 2.7 temos os bytes de pedido e resposta desse comando.

Tabela 2.6: Comando 797 - Bytes de dados do pedido

Byte	Formato	Descrição
0	<i>Signed-8</i>	Potência de saída desejada em dBm

Tabela 2.7: Comando 797 - Bytes de dados da resposta

Byte	Formato	Descrição
0	<i>Signed-8</i>	Potência de saída efetiva em dBm

2.2.4 798 – Read Radio Output Power

Este comando é utilizado para ler a potência de saída de um rádio. Na Tabela 2.8 e Tabela 2.9 temos os bytes de pedido e resposta desse comando.

Tabela 2.8: Comando 798 - Bytes de dados do pedido

Byte	Formato	Descrição
Nenhum		

Tabela 2.9: Comando 798 - Bytes de dados da resposta

Byte	Formato	Descrição
0	<i>Signed-8</i>	Potência de saída em dBm

2.2.5 968 – Delete Link

Este comando permite apagar *links* em dispositivos da rede. É utilizado neste trabalho através da porta de manutenção para apagar *links* de dispositivos de campo com o *gateway*, simulando uma organização da rede desejada. Na Tabela 2.10 e Tabela 2.11 temos os bytes de pedido e resposta desse comando.

Tabela 2.10: Comando 968 - Bytes de dados do pedido

Byte	Formato	Descrição
0	<i>Unsigned-8</i>	ID do <i>superframe</i>
1-2	<i>Unsigned-16</i>	Número do <i>slot</i> no <i>superframe</i> para esse <i>link</i>
3-4	<i>Unsigned-16</i>	<i>Nickname</i> do vizinho para este <i>link</i> (0xFFFF para <i>broadcast</i>)

Tabela 2.11: Comando 968 - Bytes de dados da resposta

Byte	Formato	Descrição
0	<i>Unsigned-8</i>	ID do <i>superframe</i>
1-2	<i>Unsigned-16</i>	Número do <i>slot</i> no <i>superframe</i> para esse <i>link</i>
3-4	<i>Unsigned-16</i>	<i>Nickname</i> do vizinho para este <i>link</i> (0xFFFF para <i>broadcast</i>)
5-6	<i>Unsigned-16</i>	Número de <i>links</i> restantes

2.2.6 971 – Write Neighbor Property Flag

Este comando permite o gerenciador de rede marcar as propriedades de um vizinho de um dispositivo de rede. No âmbito deste trabalho esse comando é utilizado para marcar um rádio como *time source*. Na Tabela 2.12 e Tabela 2.13 temos os bytes de pedido e resposta desse comando.

Tabela 2.12: Comando 971 - Bytes de dados do pedido

Byte	Formato	Descrição
0-1	<i>Unsigned-16</i>	<i>Nickname</i> do vizinho
2	<i>Bits</i>	Propriedades do vizinho (0x1 quando for <i>time source</i>)

Tabela 2.13: Comando 971 - Bytes de dados da resposta

Byte	Formato	Descrição
0-1	<i>Unsigned-16</i>	<i>Nickname</i> do vizinho
2	<i>Bits</i>	Propriedades do vizinho (0x1 quando for <i>time source</i>)

3 TESTES DE ROBUSTEZ

Neste capítulo serão descritos os experimentos realizados. O formato utilizado será uma pequena introdução sobre o teste, contendo o seu formato, suas possíveis causas, uma descrição da execução do experimento e resultados encontrados.

A escolha dos testes foi realizada devido a sua ocorrência nos dispositivos de campo descritos na sessão 2.1.1 desenvolvidos pelo LASCAR. Em ambos os casos descritos nas sessões 3.1 e 3.2, as falhas foram de difícil diagnóstico devido aos efeitos incomuns que cada uma causava no sistema. Com isso, foi constatada a necessidade de uma análise mais profunda dos casos evidenciados, com o intuito de obter seus modelos de falhas.

3.1 Diminuição da Potência de Transmissão

3.1.1 Introdução

Neste experimento serão explorados problemas na amplificação de potência do sinal de transmissão de um dos dispositivos de campo. O objetivo é ocasionar uma assimetria na transmissão e recepção do sinal em uma rede com as conexões entre dispositivos de campo já estabelecidas. Dessa forma, será possível fazer com que um dispositivo receba DLPDUs e transmita a confirmação de sua recepção. Essa confirmação não chegará ao seu destino devido à potência do sinal estar mais baixa em relação ao momento em que o *link* foi criado. Como não há um segundo passo de confirmação da mensagem, o rádio iniciador da comunicação irá acreditar que o pacote não foi transmitido com sucesso e continuará tentando transmiti-lo. As consequências esperadas são uma sobrecarga de dados na rede e um comportamento anormal dos dispositivos que receberem mais pacotes do que o esperado.

Possíveis causas deste erro são:

- Má programação do dispositivo de campo através do comando 797, diminuindo propositalmente a potência de transmissão com o intuito de realizar a economia de bateria dos rádios;
- Quebra da antena de transmissão de pacote em um dispositivo de campo que possui antenas separadas para transmissão e recepção.
- Degradação do circuito responsável pela transmissão do sinal.
- Utilização do rádio em condições fora do projeto.

O rádio desenvolvido pelo LASCAR [MULLER, 2010] utiliza o CC2591 da *Texas Instruments* como interface de comunicação. O circuito possui um controle digital de

ganho LNA e um amplificador de potência que são a entrada e a saída da comunicação sem fio do rádio.

Na Figura 3.1 [Texas Instruments, 2008] vemos o esquemático do CC2591. Tanto o caminho de transmissão quanto o de recepção utilizam a mesma antena para comunicação. Esse fato impossibilita a falha através da quebra/retirada da antena, pois ocasionaria degradação também na recepção do sinal.

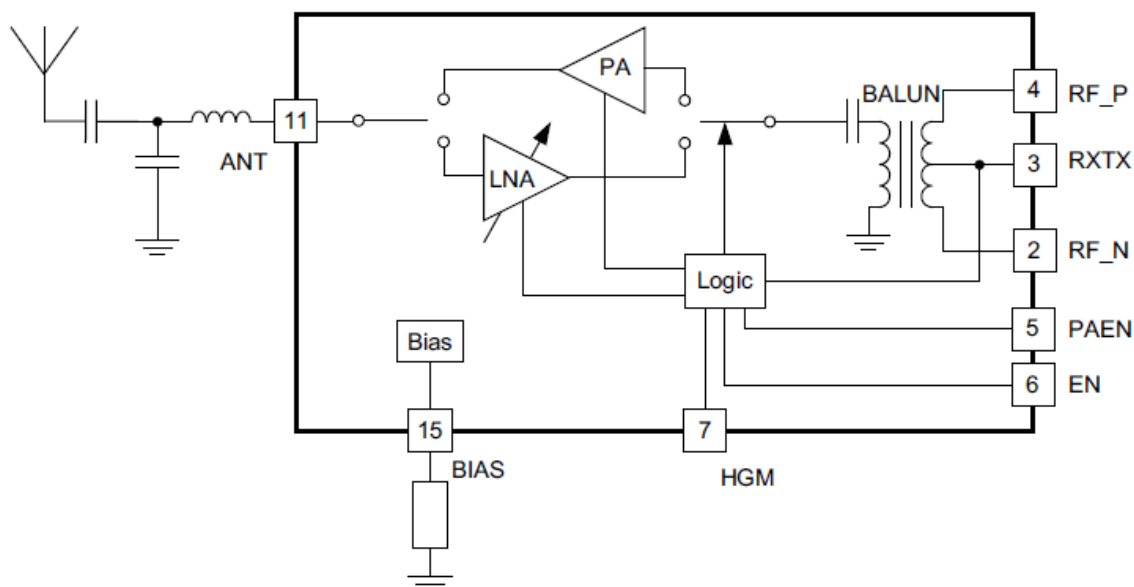


Figura 3.1: Diagrama de blocos do CC2591

Já na Figura 3.2 e Figura 3.3 [Texas Instruments, 2008], podem ser vistos os efeitos que a temperatura, frequência e tensão de alimentação podem causar na potência de saída, podendo variar em até 5 dBm no pior caso, sendo essas algumas das alternativas de causa dessa falha.

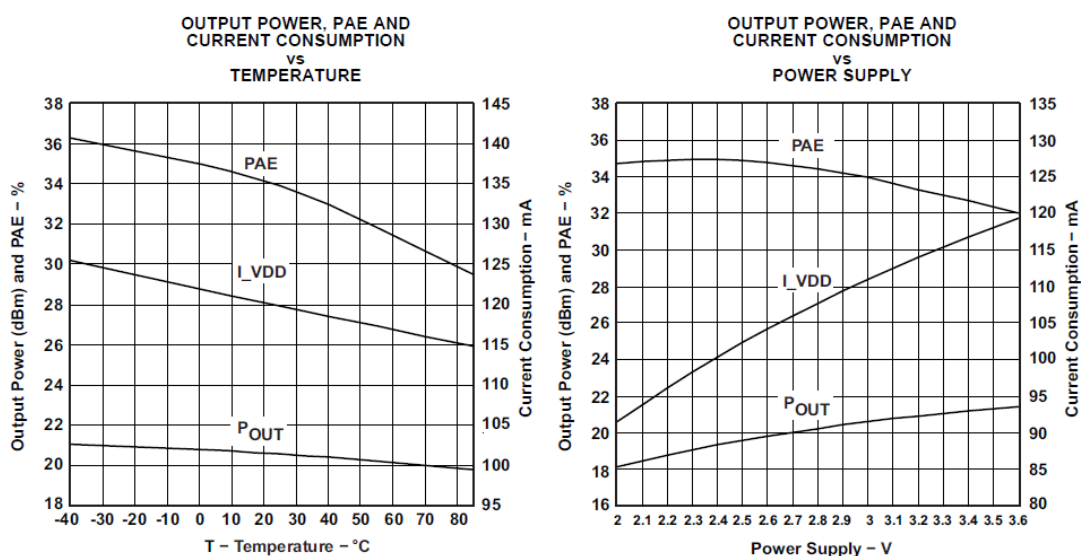


Figura 3.2: Impacto da temperatura e tensão de alimentação na potência de transmissão do CC2591

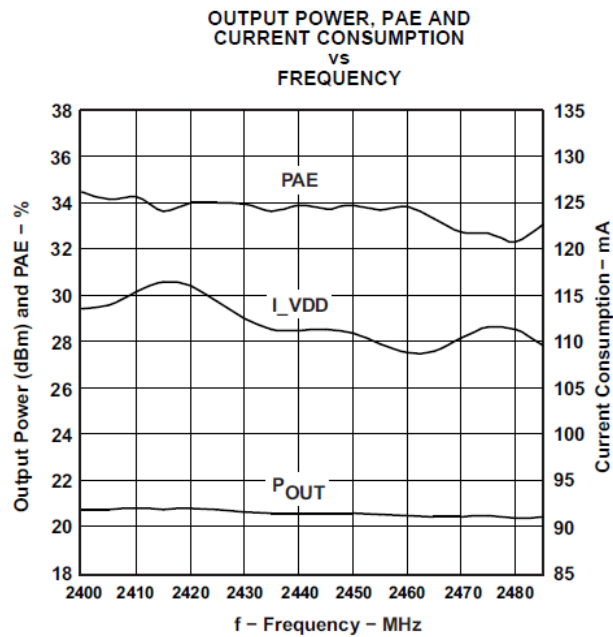


Figura 3.3: Impacto da frequência na potência de transmissão do CC2591

Parâmetros como temperatura, frequência e tensão de alimentação podem ocasionar outras falhas mascarando os objetivos do experimento, então a melhor opção é a utilização do comando `797 Write Radio Power Output` para alterar diretamente a potência de saída do sinal.

3.1.2 Execução do Experimento

Primeiramente uma análise foi realizada sobre uma rede *mesh* já formada e estável, e em um segundo momento uma rede em linha foi criada, propiciando o aparecimento da falha.

A execução do experimento ocorre através dos passos a seguir. Inicialmente a exata topologia da rede é verificada através do comando `784` que identifica os *links* criados pelo gerenciador de rede entre os dispositivos de campo. A seguir, um grafo com as conexões interpretadas anteriormente é desenhado para facilitar a compreensão e análise da rede. Então, se necessário, *links* são apagados com o intuito de que a rede fique em um estado controlado que maximize os possíveis efeitos da falha. Por fim, a potência do dispositivo é reduzida, inserindo efetivamente a falha no sistema.

Os pacotes transmitidos entre os dispositivos de campo são coletados pelo *sniffer* e analisados na seção 3.1.3.

Rede com caminhos redundantes

O primeiro passo é a identificação das ligações existentes entre os dispositivos da rede. A interface de comunicação com o *Smart Wireless Gateway* permite a fácil visualização dos dispositivos de uma rede e seus vizinhos (Figura 3.4).

HART Tag	Node state	Active neighbors	Neighbors
TAG 3333	●	GWEmerson	4
		TAG 1015	
		TAG 1025	
		TAG 1008	
TAG 1025	●	GWEmerson	2
		TAG 3333	
TAG 1008	●	GWEmerson	3
		TAG 3333	
		TAG 1000	
TAG 1000	●	GWEmerson	2
		TAG 1008	
TAG 1015	●	GWEmerson	2
		TAG 3333	

Figura 3.4: Formação da rede estável informada pelo *Smart Wireless Gateway*

Contudo, ainda é necessário verificar os *links* criados pelo gerenciador de rede para verificar o sentido da comunicação entre vizinhos. O comando `784 Read Link List` realiza a leitura dos *links* entre dispositivos de campo. Os dados mais importantes na resposta desse comando são os tipos de cada um dos *links* em conjunto com sua opção de transmissão de pacotes. Sabe-se que a publicação de variáveis de processo é feita somente por *links* de transmissão de dados (TX) do tipo Normal, então esses serão os principais *links* a serem analisados neste teste.

Enviando o comando `784 Read Link List` para cada um dos dispositivos de campo na rede, obtemos os dados indicados nas tabelas Tabela 3.1, Tabela 3.2, Tabela 3.3, Tabela 3.4 e Tabela 3.5 como resposta. Retirando as informações relevantes, pode ser obtido um grafo na Figura 3.5 que transcreve os *links* de publicação dos rádios. Mais detalhes sobre a interpretação dos *bytes* de resposta podem ser encontrados na seção 2.2.1.

Tabela 3.1: Resposta do comando 784 do dispositivo TAG 3333 e ID 2

Link	Tipo	Opções
00 00 01 00 00 01 03 01	TX/RX	Discovery
00 00 83 02 F9 80 02 03	RX	Join
00 00 B1 01 00 01 01 00	TX	Normal
00 01 B1 00 00 01 01 00	TX	Normal
00 03 B1 01 00 01 01 00	TX	Normal
00 02 9E 01 00 03 01 02	TX	Normal
00 03 83 02 00 07 02 00	RX	Normal
00 02 83 01 00 05 02 00	RX	Normal
01 00 05 00 F9 80 01 03	TX	Join
01 00 06 00 FF FF 02 02	RX	Broadcast

01 00 F7 01 FF FF 02 02	RX	Broadcast
01 00 FD 00 FF FF 02 02	RX	Broadcast
04 00 3D 03 FF FF 01 02	TX	Broadcast

Tabela 3.2: Resposta do comando 784 do dispositivo TAG 1025 e ID 3

Link	Tipo	Opções
00 00 01 00 00 01 03 01	TX/RX	<i>Discovery</i>
00 00 9E 00 F9 80 02 03	RX	<i>Join</i>
00 01 6D 01 00 01 01 00	TX	Normal
00 00 6D 02 00 01 01 00	TX	Normal
00 02 6D 02 00 01 01 00	TX	Normal
00 03 6D 00 00 01 01 00	TX	Normal
00 02 9E 01 00 02 02 00	RX	Normal
01 00 FD 00 F9 80 01 03	TX	<i>Join</i>
01 00 33 01 FF FF 02 02	RX	<i>Broadcast</i>
01 00 F7 01 FF FF 02 02	RX	<i>Broadcast</i>
04 00 18 03 FF FF 01 02	TX	<i>Broadcast</i>

Tabela 3.3: Resposta do comando 784 do dispositivo TAG 1008 e ID 5

Link	Tipo	Opções
00 00 01 00 FF FF 03 01	TX/RX	<i>Discovery</i>
00 03 89 01 F9 80 02 03	RX	<i>Join</i>
00 01 8E 00 00 01 01 00	TX	Normal
00 03 9E 02 00 01 01 00	TX	Normal
00 00 96 01 00 01 01 00	TX	Normal
00 02 83 01 00 02 01 00	TX	Normal
00 03 09 00 00 06 02 00	RX	Normal
01 00 FB 00 F9 80 01 03	TX	<i>Join</i>
01 00 05 00 FF FF 02 02	RX	<i>Broadcast</i>
01 00 15 01 FF FF 02 02	RX	<i>Broadcast</i>
01 00 F7 01 FF FF 02 02	RX	<i>Broadcast</i>
04 00 14 03 FF FF 01 02	TX	<i>Broadcast</i>

Tabela 3.4: Resposta do comando 784 do dispositivo TAG 1000 e ID 6

Link	Tipo	Opções
00 00 01 00 FF FF 03 01	TX/RX	<i>Discovery</i>
00 01 B6 01 F9 80 02 03	RX	<i>Join</i>
00 01 67 00 00 01 01 00	TX	Normal
00 02 38 00 00 01 01 00	TX	Normal
00 00 C6 01 00 01 01 00	TX	Normal
00 03 09 00 00 05 01 00	TX	Normal
01 00 07 02 F9 80 01 03	TX	<i>Join</i>
01 00 FB 00 FF FF 02 02	RX	<i>Broadcast</i>
01 00 F7 01 FF FF 02 02	RX	<i>Broadcast</i>
01 00 42 02 FF FF 02 02	RX	<i>Broadcast</i>
04 00 71 03 FF FF 01 02	TX	<i>Broadcast</i>

Tabela 3.5: Resposta do comando 784 do dispositivo TAG 1015 e ID 7

Link	Tipo	Opções
00 00 01 00 00 01 03 01	TX/RX	<i>Discovery</i>
00 00 17 00 F9 80 02 03	RX	<i>Join</i>
00 00 3B 01 00 01 01 00	TX	Normal
00 01 3B 00 00 01 01 00	TX	Normal
00 02 3B 02 00 01 01 00	TX	Normal
00 03 83 02 00 02 01 00	TX	Normal
01 00 04 02 F9 80 01 03	TX	<i>Join</i>
01 00 F7 01 FF FF 02 02	RX	<i>Broadcast</i>
01 00 24 00 FF FF 02 02	RX	<i>Broadcast</i>
01 00 05 00 FF FF 02 02	RX	<i>Broadcast</i>
04 00 6B 03 FF FF 01 02	TX	<i>Broadcast</i>

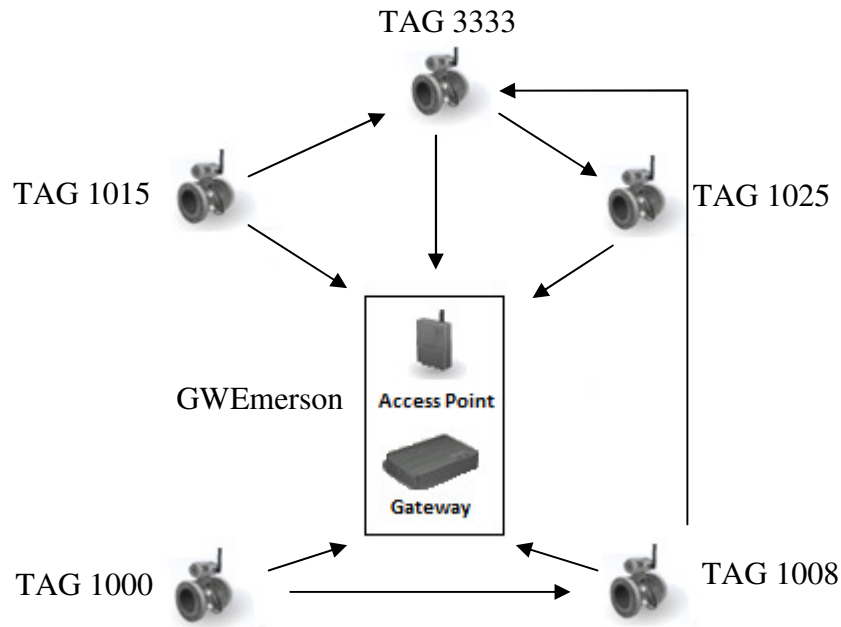


Figura 3.5: Representação dos links de tipos normais na rede

Algumas conclusões podem ser realizadas sobre o gerenciador e como a rede está formada.

Analisando os *bytes* das respostas ao comando 784, pode ser constatado que o *superframe 0* é utilizado somente para comunicação iniciada pelos dispositivos de campo, chamados de *uplinks*. Esse *superframe* possui *links* do tipo normais e indica *time slots* redundantes, já que neste caso existe banda disponível para que essas configurações sejam atendidas. O *superframe* é indicado pelos 2 primeiros *bytes* da descrição de cada *link*.

Em seguida temos os *superframes 1 e 4* que contém somente links de *join* e *broadcast* e, ao contrário do SF 0, é responsável pela comunicação iniciada pelo gerenciador de rede, que pode ser chamado de *downlink*.

A rede formada pelo gerenciador de rede utilizado nos experimentos é altamente centralizada, pois existem *links* de todos os dispositivos para o ponto de acesso central, sendo esse o caminho redundante. Essa característica é intrínseca à forma como a rede é implementada já que uma mensagem não pode ter como fonte e destino dispositivos de campo. Uma forma de minimizar isso seria inserindo mais pontos de acesso em uma rede ou também, descentralizando a rede, fazendo com que dispositivos de campo pudessem se tornar pontos de acesso e assim as mensagens não teriam sempre o mesmo destino.

Outro ponto que pode ser salientado é o fato da rede montada ser determinística somente para a publicação de variáveis, ou *upstream*. Esse fato pode ser visto pelo fato dos *links* do tipo normal estarem sempre nessa direção. O controle realizado pelo gerenciador de rede é sempre feito com *links* do tipo *broadcast*, não sendo determinístico, já que o caminho que um pacote irá realizar até seu destino pode variar de acordo com o momento em que a transmissão é realizada.

Como todos os rádios possuem *links* de publicação redundantes, no caso de uma falha na recepção de confirmação de um DLPDU, o rádio realizaria uma tentativa de envio através do caminho redundante. Por esse motivo, a falha desejada não pode ser inserida nesse tipo de rede.

A consequência desta falha seria apenas uma sobrecarga de mensagens transmitidas pela rede. Como o foco desse experimento não é a sobrecarga de mensagens, mas sim o comportamento do dispositivo no momento em que não consegue receber confirmações quando não possui caminhos redundantes, essa topologia não terá seu estudo continuado.

Rede com caminho em linha

Em um segundo momento, analisa-se uma topologia em linha com o intuito de obter um dispositivo que não possua caminhos redundantes de publicação de variáveis. Para isso inicialmente permite-se que o gerenciador da rede crie uma rede com três nodos possuindo *links* de *upstream* e *downstream* entre si, o que é ilustrado na Figura 3.6.

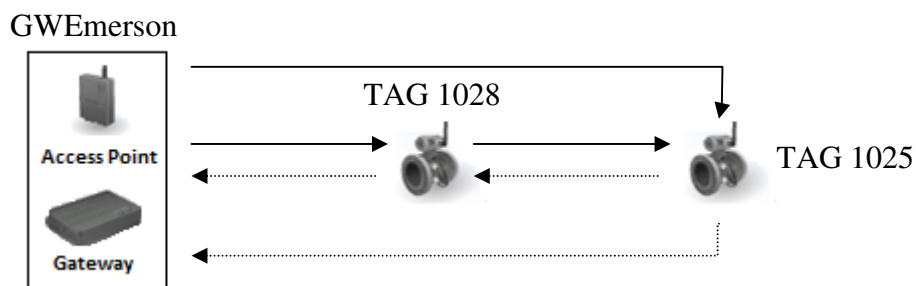


Figura 3.6: Representação de uma rede *WirelessHART* em linha

Os links podem ser verificados novamente através do comando `784 Read Link List` enviado pela interface do gateway, que pode ser vista na Figura 3.7.

Device	TAG 1025
Command code	784
Request data bytes	000099

Send

Response time	4.203 sec
Response code	00
Device status	10
Response data	00 00 0A 00 0A 01 00 C1 02 FF FF 02 02 00 03 B4 02 00 01 01 00 00 00 01 00 00 01 03 01 04 00 28 03 FF FF 01 02 00 00 B4 00 00 01 01 00 01 00 CD 00 F9 80 01 03 01 00 D1 02 FF FF 02 02 00 01 B4 01 00 01 01 00 00 02 89 01 F9 80 02 03 00 02 B4 01 00 01 01 00
Parsed response data	

Figura 3.7: Verificação dos *links* criados pelo gerenciador de rede

Logo após, os *links* entre o *gateway* e o dispositivo TAG 1025 devem ser apagados através comando 968 fazendo com que a rede se encontre na situação ilustrada na Figura 3.8, onde não possui *links* redundantes.

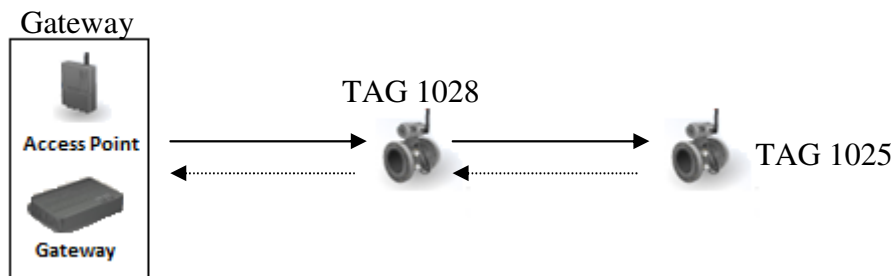


Figura 3.8: Representação de uma rede em linha sem *links* redundantes

Se os três dispositivos estiverem equidistantes, a potência do sinal de transmissão de dados do dispositivo TAG 1028 possui a mesma intensidade nos pontos dos outros dois equipamentos. O dispositivo intermediário deve ser deslocado para que fique o mais próximo possível do ponto de acesso sem prejudicar a transmissão com o rádio identificado pelo TAG 1025.

O funcionamento normal da rede é verificado. Lêem-se os valores de potência de saída dos dispositivos de campo presentes na rede através do comando 798, enviado pela interface do *gateway* (Figura 3.9), para registro do estado atual da rede.

Device	TAG 1025
Command code	798
Request data bytes	

Send

Response time	2.156 sec
Response code	00
Device status	10
Response data	0A
Parsed response data	Response798 [responseCode=0, statusCode=16, power=10]

Device	TAG 1028
Command code	798
Request data bytes	

Send

Response time	2.156 sec
Response code	00
Device status	10
Response data	0A
Parsed response data	Response798 [responseCode=0, statusCode=16, power=10]

Figura 3.9: Potência lida dos dispositivos na rede funcional

A seguir, a potência de saída do dispositivo TAG 1028 é reduzida, utilizando o comando 971, para o mínimo aceitável pelo rádio, neste caso -7 dB, tendo então a situação descrita na seção 3.1.1, onde uma rede foi criada em um momento onde os rádios estariam em funcionamento normal, porém, após certo tempo, ocorreram falhas na transmissão de pacotes.

Com isso pode ser verificado o mau funcionamento da rede no *link* indicado em vermelho na Figura 3.10.

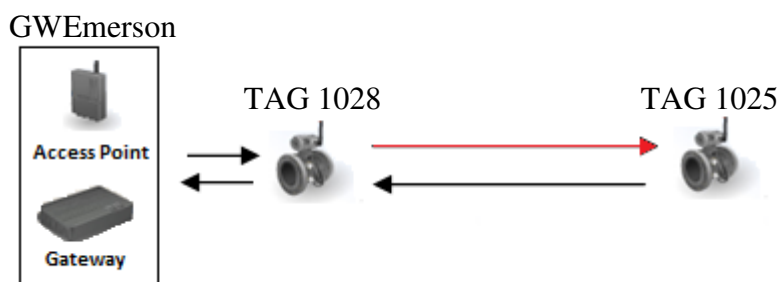


Figura 3.10: Representação de uma rede em linha com falha

Na seção 3.1.3 podem ser vistos os resultados obtidos através desse experimento.

3.1.3 Resultados do Experimento

Nesta primeira análise dos resultados a saída resultante do *sniffer* da rede será avaliada em diferentes momentos do experimento.

Em um primeiro momento existe a comunicação normal entre os dispositivos da rede. Note que, na Tabela 3.6, o tempo entre as transmissões de dados do rádio TAG 1025 é de aproximadamente 1 minuto, que é o tempo de *burst* configurado para ele, o que indica que a transmissão está acontecendo normalmente.

Tabela 3.6: Saída do *sniffer* em uma rede em linha sem falhas

Data/Hora	PDU	ASN Snippet	ASN (0)	Destino	Fonte	Payload
2013-06-17 17:37:32.187	Data	DD2D	D9	TAG1028	TAG1025	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:37:32.828	Data	DD2D	16	1	TAG1028	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:37:32.921	ACK		16	TAG1028	1	
2013-06-17 17:38:07.046	KA		7A	TAG1012	TAG1028	
2013-06-17 17:38:07.046	ACK		7A	TAG1028	TAG1012	
2013-06-17 17:38:13.296	KA		D9	TAG1028	TAG1025	
2013-06-17 17:38:14.640						
2013-06-17 17:38:21.218						
2013-06-17 17:38:23.546	Data		D9	TAG1028	TAG1025	
2013-06-17 17:38:24.203	Data	F442	16	1	TAG1028	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=49.000000]
2013-06-17 17:38:24.281	ACK		16	TAG1028	1	
2013-06-17 17:38:27.734	Data	F5DB	8E	1	TAG1012	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:38:27.890	ACK		8E	TAG1012	1	

Logo após, pode ser visto na Tabela 3.7 o envio do comando 797 *Write Radio Power Output* para o dispositivo TAG 1028, que será o dispositivo com falha, escrevendo uma nova potência de saída no rádio, colocando-a no valor de -7dB, o mínimo aceitável pelo rádio.

Tabela 3.7: Saída do *sniffer* durante o envio do comando 797

Data/Hora	PDU	ASN Snippet	ASN (0)	Destino	Fonte	Payload
2013-06-17 17:38:57.125	Data	013D	EE	TAG1028	1	[WriteRadioPower cmd=797, bc=1,

						powerOutput=-16]
2013-06-17 17:38:57.140	ACK		EE	1	TAG1028	
2013-06-17 17:38:57.609	Data	01F5	16	1	TAG1028	[WriteRadioPower cmd=797, bc=2, rc=8, powerOutput=-7]

E por fim, após a diminuição da potência de transmissão do rádio, a publicação de valores continua sendo feita, porém o TAG 1025 não consegue perceber que a mensagem está sendo enviada corretamente, já que o rádio TAG 1028 não possui potência suficiente para transmitir a confirmação da mensagem, o que pode ser visto na Tabela 3.8. Isso faz com que ele repita a transmissão no *time slot* seguinte de transmissão, tentando cumprir o prazo que lhe foi designado pelo gerenciador de rede. Note que os tempos entre tentativas de publicação da variável de processo do nodo TAG1025 são em torno de 10 segundos, que é muito menor do que o tempo de *burst* configurado no dispositivo.

Tabela 3.8: Saída do *sniffer* em uma rede em linha com falha

Data/Hora	PDU	ASN <i>Snippet</i>	ASN (0)	Destino	Fonte	<i>Payload</i>
2013-06-17 17:40:26.765	Data	237D	D9	TAG1028	TAG1025	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:27.421	Data	237D	16	1	TAG1028	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:27.500	ACK		16	TAG1028	1	
2013-06-17 17:40:28.375	Data	24BB	8E	1	TAG1012	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:28.531	ACK		8E	TAG1012	1	
2013-06-17 17:40:37.031	Data	237D	D9	TAG1028	TAG1025	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:37.828	Data	237D	16	1	TAG1028	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:37.828	ACK		16	TAG1028	1	
2013-06-17 17:40:41.015	KA		7A	TAG1012	TAG1028	
2013-06-17 17:40:41.046	ACK		7A	TAG1028	TAG1012	
2013-06-17 17:40:47.281	Data	237D	D9	TAG1028	TAG1025	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17	Data	237D	16	1	TAG1028	[ReadPv cmd=1, bc=6,

17:40:47.921						rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:48.015	ACK		16	TAG1028	1	
2013-06-17 17:40:57.546	Data	237D	D9	TAG1028	TAG1025	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:58.187	Data	237D	16	1	TAG1028	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:40:58.359	ACK		16	TAG1028	1	
2013-06-17 17:41:07.656	Data	237D	D9	TAG1028	TAG1025	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:41:07.812	ACK		D9	TAG1025	TAG1028	
2013-06-17 17:41:08.453	Data	237D	16	1	TAG1028	[ReadPv cmd=1, bc=6, rc=0, varUnits=Percent, varValue=30.000000]
2013-06-17 17:41:08.531	ACK		16	TAG1028	1	

Em seguida, uma análise é realizada sobre a saída de depuração do dispositivo TAG 1025. Os caracteres '@0#' indicam a falha na recepção do ACK da última mensagem enviada pelo dispositivo em depuração. Sabendo disso, na Figura 3.11, pode ser visto que a partir do ASN 94687449 o rádio não recebe mais nenhuma confirmação de envio de pacotes, o que caracteriza a saída do rádio da rede. Mais detalhes sobre a saída de depuração podem ser encontradas na Tabela 2.1.

```

K[0003-1]
ASN: 94671065
<19> correction from 3
BK[0003-1]D

ASN: 94674137
<23> correction from 3
K[0003-1]

ASN: 94677209
<19> correction from 3
BK[0003-1]D@#D@#-K[0003-1]-D

ASN: 94682329
<13> correction from 3

K[0003-1]@#BD@#-K[0003-1]D
ASN: 94687449

<19> correction from 3

#-K[0003-1]@#BD@#-K[0003-1]D@#-K[0003-1]D@#D@#-K[0003-1]D@#-K[0003-1]D@#BD@#-K[0003-1]D@#-DK[0003-1]]0@#D@#-K[0003-1]D@#-DK[0003-1]]0@#(4171B8)Bdnp 2 TX
D]0@#-K[0003-1]D@#-K[0003-1]D@#D@#-K[0003-1]D@#-K[0003-1]D@#B-#D@#-K[0003-1]D@#f-K[0003-1]D@#D@#-K[0003-1]D@#-D@#K[0003-1]B-(416ED4)(416ED4)NPD@#-K[0003-1]D@#-D@#K[0003-1]D@#-K[0003-1]D@#-D@#K[0003-1]B(417090)D@#-K[0003-1]D@#(4171B8)-K[0003-1]D@#D@#-K[0003-1]D@#-K[0003-1]D@#B-#-D@#-K[0003-1]D@#-K[0003-1]D@#D@#-K[0003-1]D@#-D@#K[0003-1]B-D@#-K[0003-1]D@#-D@#K[0003-1]D@#-K[0003-1]D@#B--K[0003-1]D@#

```

Figura 3.11: Saída de depuração do dispositivo TAG1025

Device name	
Device name	TAG 1025
Burst statistics	
Last burst message latency	8.081 sec
Average burst message latency	9.791 sec
Minimum burst message latency	0.171 sec
Maximum burst message latency	62.331 sec
Burst message 0	
Command	1
Mode	Wireless
Expected burst rate	60.000 sec
Average burst rate	59.249 sec
Updates	33
Missed updates	0
Reliability	100.0 %
Last update	06/17/13 17:57:06

Figura 3.12: Percepção do gerenciador da rede após o experimento

A próxima análise a ser feita é pelo lado do gerenciador de rede, que deveria identificar uma situação de erro e realizar uma ação de correção.

A Figura 3.12 indica que o gerenciador não percebe que a rede está com falha, já que, mesmo com o dispositivo não recebendo confirmações de envio da variável de processo o gerenciador continua recebendo. Com isso, a rede continua operando mesmo que esteja em uma situação de falha.

Time	Description
06/17/13 18:01:10.355	Needs neighbor flag cleared for 00-1B-1E-F9-82-00-10-25.
06/17/13 18:01:10.347	Discovery status for 00-1B-1E-F9-82-00-10-25 is Not Responding.
06/17/13 18:01:10.346	Node status for 00-1B-1E-F9-82-00-10-25 is Unreachable.
06/17/13 18:01:10.341	Mote 00-1B-1E-F9-82-00-10-25 is unreachable.
06/17/13 17:25:10.841	Warning: device TAG 1025 is publishing command 1 which contains
06/17/13 17:25:10.768	Discovery status for 00-1B-1E-F9-82-00-10-25 is Responding.

Figura 3.13: Registro do gateway indicando a saída do rádio TAG 1025 da rede

Após diversas tentativas de envio da variável de processo, pelo nodo TAG1025, sem confirmação, o mesmo para de enviar mensagens, pensando que já está fora da rede. Isso pode ser visualizado na segunda linha do registro do gateway (Figura 3.13), que indica que o dispositivo com o MAC 00-1B-1E-F9-82-00-10-25 não está respondendo.

Nesta situação, o profissional que estivesse realizando manutenção do sistema, sem conhecer a topologia exata da rede, poderia substituir o rádio TAG1025 por outro, acreditando que este estaria estragado mesmo que na verdade o erro se encontrasse em outro dispositivo. Isso ocasionaria no gasto desnecessário de recursos e também em um período indisponível daquele ponto de controle/monitoração da rede.

Pode ser visto que o gerenciador de rede escolhido para o experimento não executa nenhuma operação de manutenção sobre a rede neste tipo de ocorrência. Ele poderia minimizar o impacto desta falha na rede através da identificação e diagnóstico em caso de suspeita da falha.

A identificação pode ser realizada verificando que um dispositivo repete publicações da variável de processo sem necessidade. Durante do diagnóstico, poderiam ser avaliadas, por exemplo, as potências de transmissão dos dispositivos. Dessa forma, seria possível ver que o rádio TAG 1028 estaria com a potência muito baixa, identificando a fonte do problema na rede. Neste caso, a equipe de manutenção poderia realizar os reparos/substituições necessárias na rede com maior facilidade.

Outra possível solução seria adicionar uma mensagem indicando erro quando um dispositivo não recebesse confirmação após sucessivas transmissões de DLPDUs. No caso de existir problemas na transmissão de pacotes, esta alteração não ocasionaria nenhum impacto na rede, porém, se o problema fosse na transmissão da confirmação da DLPDU, como é o caso do experimento, a mensagem chegaria ao gerenciador de rede que então poderia tomar providências sobre o estado da rede. Essa alteração não ocasionaria um grande impacto na norma, já que seria somente uma situação em caso de erro, onde o dispositivo envia um comando extra, e não um comportamento normal dos nodos no sistema.

3.2 Atraso no Relógio de Referência

3.2.1 Introdução

A sincronização dos relógios é um fator essencial em uma rede TDMA. Isso porque dispositivos transmitem e aguardam DLPDUs em tempos determinados.

Independente do tipo de fonte de sincronização dos elementos existe uma diferença entre *clocks* de cada um. Isso ocorre devido a diferenças durante a fabricação, pois não se pode produzir duas referências exatamente iguais, e pela exposição a diferentes fatores externos, que vão ocasionar reações diversas em cada uma dessas referências. Por essa razão, uma determinada diferença de tempo entre as referências dos dispositivos deve ser tolerada.

Em uma rede *wirelessHART*, a referência de tempo principal é fornecida pelo gerenciador de rede. Este irá designar dispositivos de campo como *time sources*, que serão responsáveis por difundir a referência de tempo através da rede.

A sincronização entre dispositivos é feita da seguinte forma: quando um rádio recebe um DLPDU, a marca de tempo do seu recebimento é guardada e utilizada para calcular a diferença entre o tempo em que o rádio acredita ser o ideal para a recepção do pacote em relação ao momento que realmente ocorreu. Essa diferença é informada nas mensagens de confirmação enviadas pelos rádios. Se o nodo, o qual enviou a confirmação, é reconhecido como um *time source*, então o dispositivo receptor irá ajustar o seu relógio de acordo com o dado de correção de tempo enviado. Todas as confirmações enviadas por dispositivos de campo marcados como *time sources* ocasionam em correção no *clock* de outros dispositivos.

No caso de não haver o envio de pacotes durante um longo período de tempo, o equipamento deve enviar uma mensagem de *keep-alive* para garantir a sua sincronia com o resto do sistema. A norma especifica que pacotes de *keep-alive* não devem ser enviados com uma frequência menor do que um a cada 30 segundos em um ambiente onde a temperatura varia em até 2°C por minuto. Os equipamentos também devem tolerar uma segunda tentativa de envio desse tipo de pacotes para o caso de interferência durante a transmissão, isso representa uma margem de 10 segundos e corresponde a aproximadamente uma precisão de relógio compensada de 10ppm ou menos [HART, 2008].

Um *time source* com atraso na sua referência de tempo pode ocasionar um efeito em cascata nos dispositivos relacionados a ele.

O dispositivo de campo desenvolvido pelo LASCAR utiliza o *clock* para obter uma referência para o rádio, processador e periféricos, como uma referência de tempo real (RTC) para acordar do modo de hibernação e também para manter o relógio em sincronia com os limites de tempo impostos pela comunicação TDMA.

Duas referências são utilizadas no dispositivo, um cristal de 24 MHz e um oscilador de 32.768 KHz. O primeiro é um cristal padrão interno ao MC1322x, contudo a fonte primária de relógio pode ser derivada de um *clock* externo ao chip que pode variar entre 13 a 26 MHz.

O oscilador de 32.768 KHz é opcional e pode ser implementado através da utilização de um cristal externo ao MC1322x. A vantagem de sua utilização é seu baixo consumo, enquanto possui uma precisão de longo prazo excelente.

Na Figura 3.14 é mostrado um diagrama contendo o esquema das referências.

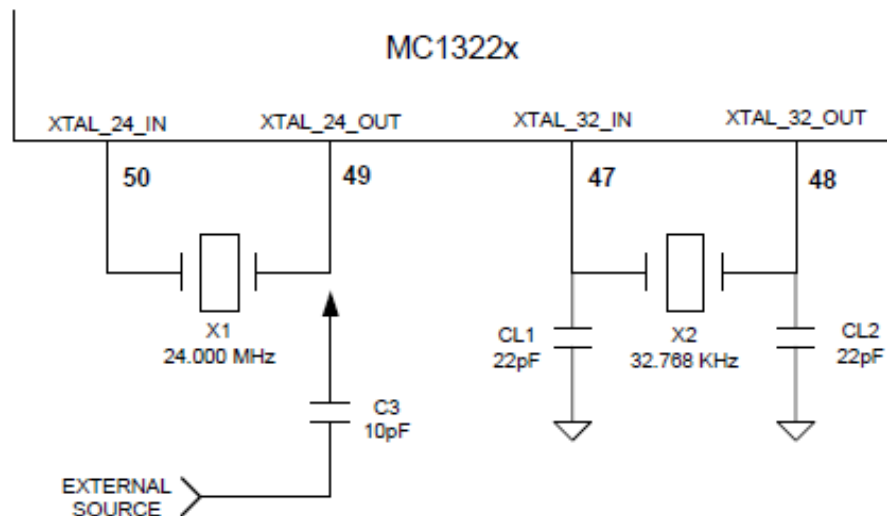


Figura 3.14: Esquemático dos *clocks* a serem utilizados pelo MC1322x [Freescale, 2010]

O oscilador principal de 24 MHz possui um banco de capacitores de carga internos e programáveis. Na Figura 3.15 [Freescale, 2010] pode ser visto o diagrama dos capacitores de carga integrados ao MC1322x. Os bancos possuem um capacitor de ajuste de 4 pf que pode ser habilitado separadamente, uma serie de capacitores de ajuste grosso, com os valores de 1, 2, 4 e 8 pF, e capacitores de ajuste fino, de 5pF em passos de 160 fF.

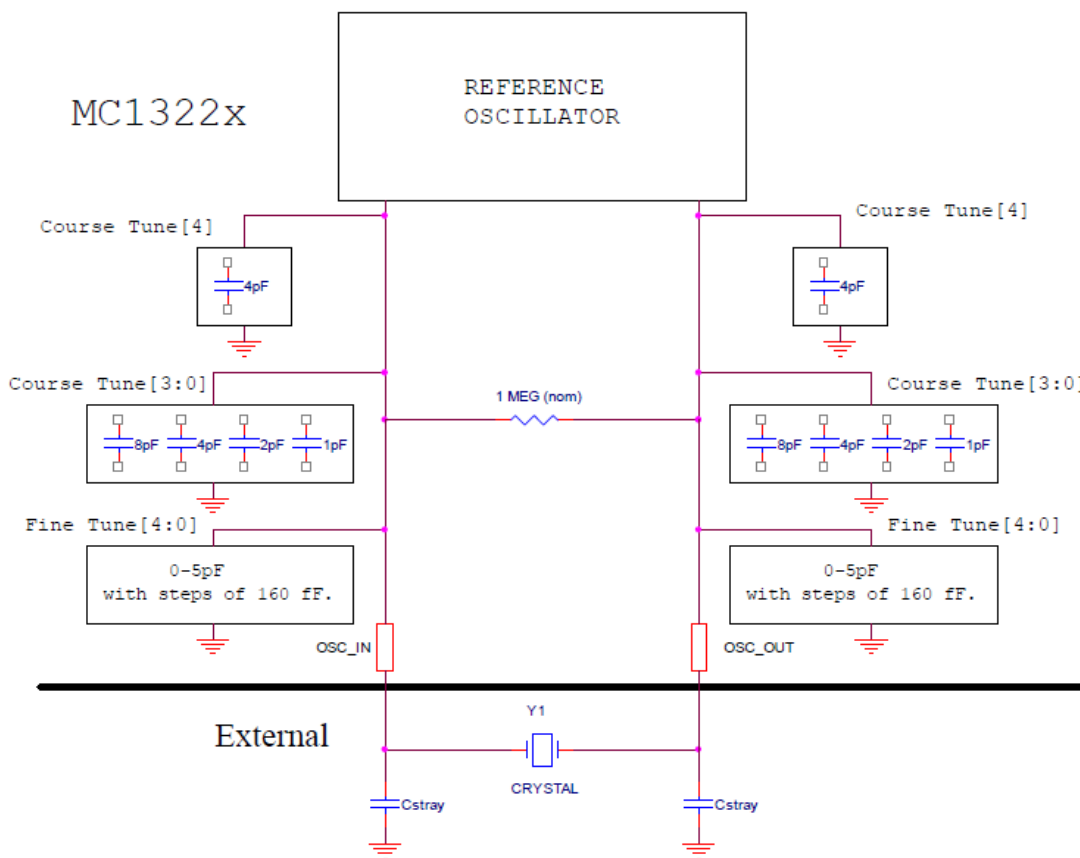


Figura 3.15: Bancos de capacitores ligados ao oscilador de referência.

O modelo mostra que com o banco de capacitores internos, cargas de capacitores externas não são necessárias para o oscilador de referência. Capacitância parasita, proveniente do encapsulamento e outros fatores comuns são normalmente da ordem de 1,5 pF [Freescale, 2010]. A capacitância de carga no oscilador de referência é simétrica e igual à metade da soma das capacitâncias configuradas, em conjunto com a capacitância parasita.

A carga do banco de capacitores pode ser alterada pelo *firmware* através da utilização dos registradores XTAL_CTUNE e XTAL_FTUNE. O registrador XTAL_CTUNE faz um ajuste grosseiro da capacitância do cristal. Os quatro bits menos significativos indicam uma capacitância igual ao seu valor em pF, ou seja, se o valor 0xF está programado nesses bits, a capacitância resultante é de 15 pF. Já o quinto bit do registrador adiciona 4 pF de carga no cristal. A Figura 3.16 mostra o erro em PPM do oscilador para a carga programada.

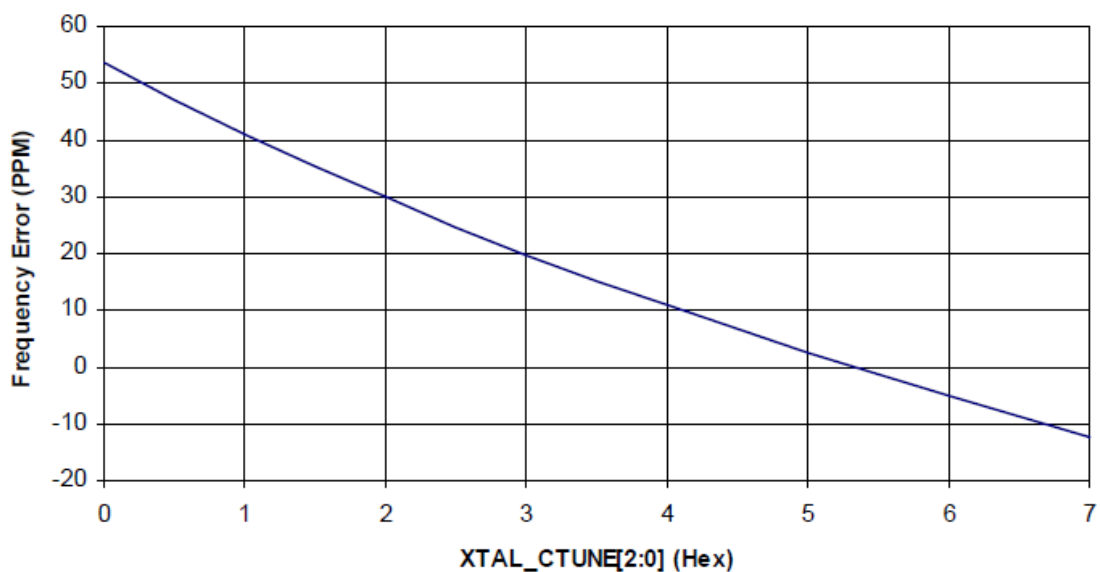


Figura 3.16: Erro do oscilador de referência em PPM vs XTAL_CTUNE[2:0] [Freescale, 2010]

O registrador XTAL_FTUNE faz o ajuste fino do relógio, inserindo uma carga menor no cristal. Podem ser inseridos cinco bits nos registradores que adicionam uma carga máxima de 5 pF em 32 passos de 156 fF. A Figura 3.17 mostra o erro em PPM do oscilador para a carga programada.

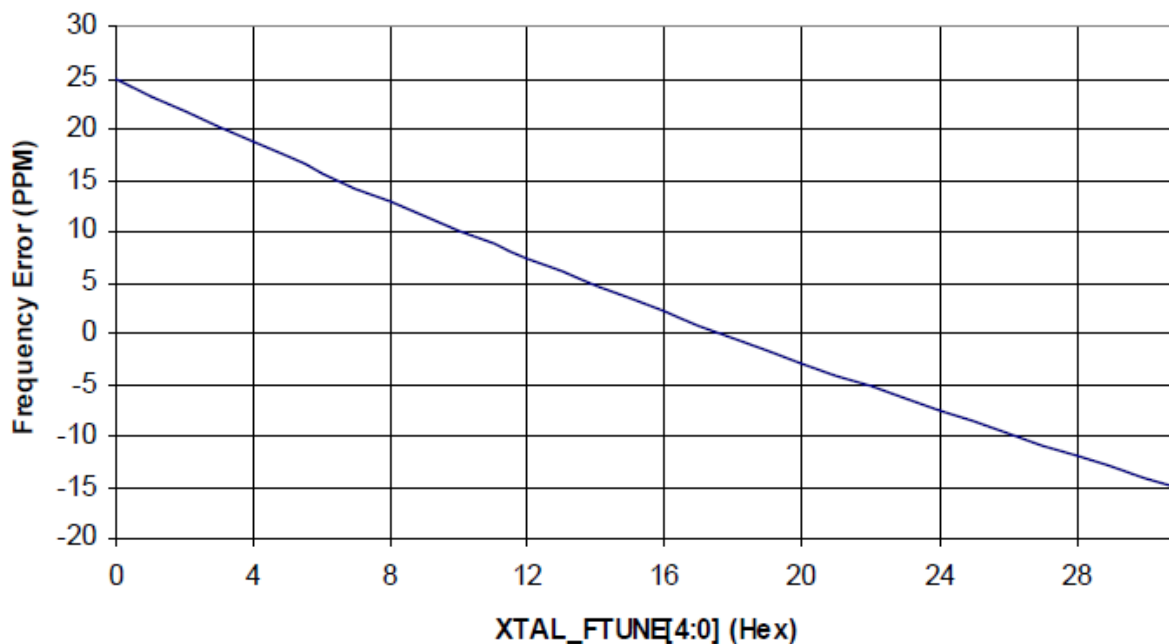


Figura 3.17: Erro do oscilador de referência em PPM vs XTAL_FTUNE[4:0] [Freescale, 2010]

Os valores iniciais são 13, para o registrador XTAL_FTUNE, e 4 para o registrador XTAL_CTUNE, que são aproximadamente os centros dos gráficos de erro da frequência nas figuras anteriores. Considerando que a norma indica que o erro máximo que deve ser suportado na rede é de 10 PPM, temos que a capacitância máxima a ser adicionada é de aproximadamente 1,5pF, o que representa 10 unidades do registrador XTAL_FTUNE.

3.2.2 Execução do Experimento

O objetivo deste experimento é alterar a frequência de *clock* de um dos dispositivos da rede de forma que ele fique defasado e acabe enviando mensagens em slots diferentes do que os configurados nele. Como consequência deste erro, espera-se que todos os dispositivos que dependam do rádio com falha para sincronização com o sistema apresentem falhas também, dificultando a identificação da fonte de erro. Para realizar isso, um comando especial foi criado, que será enviado pela porta de manutenção, que altera a carga de capacitores do oscilador principal do MC1322x.

Comando especial

Visando o controle dos registradores XTAL_CTUNE e XTAL_FTUNE, foi adicionado um comando especial, código 127, ao *firmware*, onde cada chamada adiciona cinco unidades ao valor do registrador XTAL_FTUNE, totalizando uma carga de cinco vezes 156 fF como é descrito na sessão 3.2.1.

É necessária uma compreensão do *firmware* como um todo para a implementação do comando especial, o que dificulta a execução desse passo. Como o *firmware* é proprietário, algumas barreiras ocorreram nesta etapa.

```

/*****
 * insert a clock skew
 * cmd 127
 *
 * FERNANDO SOUSA
 *****/
uint8_t
insert_clock_error(WH_addr_t *src,
                  uint8_t *in_data,
                  uint8_t in_length,
                  uint8_t *out_data,
                  uint8_t *out_length)
{
    #if DEBUG
        printf("\nInsert clock skew\n");
    #endif
    clock_trim_slower_modified();
    SET_OUT_BYTES(0);
    return RSPC_SUCCESS;
}

```

Figura 3.18: Código adicionado chamado quando o dispositivo recebe o comando 127

```

/*****
* Modified version of the above function
* clock_trim_slower
*
*
* This will be used to change the clock correction
* magnitude made by this function
* and it will be called by func 127
*
* FERNANDO SOUSA
*****/
// Bend the 24MHz reference clock slower
void
clock_trim_slower_modified(void)
{

#if DEBUG
    printf("Cfg iniciais ftune: %d - ; ctune: %d",
           CRM_XTAL_CNTL.ftune,
           CRM_XTAL_CNTL.ctune);
#endif

if( CRM_XTAL_CNTL.ftune<31 )
    {
#if DEBUG
        printf("\nIncrementing finetune by one\n");
#endif
        CRM_XTAL_CNTL.ftune++;
    }
    else if( CRM_XTAL_CNTL.ctune<14 )
    {
#if DEBUG
        printf("\nIncrementing coarse tune by one\n");
#endif
        CRM_XTAL_CNTL.ftune -= 13;
        CRM_XTAL_CNTL.ctune += 2;
    }

#if FERNANDO_DEBUG
    printf("ftune: %d - ; ctune: %d",
           CRM_XTAL_CNTL.ftune,
           CRM_XTAL_CNTL.ctune);
#endif
}

```

Figura 3.19: Código que efetua o atraso no oscilador de referência

Para que o comando 127 seja chamado, o *software*, que comunica com a porta de manutenção, foi alterado para contemplá-lo. As funções principais utilizadas, como *buildHeader*, já faziam parte da estrutura do programa antes da criação dessa função. A chamada é realizada através de um botão na interface do software.

```

//-----
#pragma hdrstop
#include "Unit1.h"
#include "cmd127.h"
//-----
#pragma package(smart_init)
//-----
#pragma hdrstop

bool ClockSkew(void)
{
    unsigned short    index_data;
    unsigned char     frame_len;
    unsigned short    response_data_index = RSP_DATA_INDEX_CMD;

    MainForm->Memo_Comm->Lines->Add("Comando 127: Iniciando atraso de clock");
    // Initialize structures
    index_data = MainForm->buildHeader(127);

    // No request data - Index data point do FCS
    frame_len = (unsigned char)index_data + 1; // count FCS

    if (frame_len > FRAME_MAX_SIZE)
        MainForm->Memo_Comm->Lines->Add("ERRO frame_len > FRAME_MAX_SIZE");

    if (!MainForm->buildFCS((unsigned char)index_data))
        MainForm->Memo_Comm->Lines->Add("Error on processing FCS");
    // End of request data

    if(MainForm->execHARTCmd(frame_len))
    {
        MainForm->Memo_Comm->Lines->Add("Comando 127 executado com sucesso");
        return(True);
    } else {
        MainForm->Memo_Comm->Lines->Add(" Comando 127 problemas!");
        return (False);
    }
}
}

```

Figura 3.20: Código adiciona ao *software* da porta de manutenção

Para executar o experimento a chamada da função através da interface do software da porta de manutenção deve ser feita após a entrada do dispositivo de campo na rede. Em seguida, o registro dos valores de correção de relógio realizado pelos rádios envolvidos é realizado através da saída de depuração. Essas etapas são repetidas até que os rádios saiam da rede, indicando que as mensagens estão sendo transmitidas em *time slots* errados, ou o valor máximo de carga do banco de capacitores, representando o valor máximo da defasagem do oscilador de referência, seja alcançado.

Topologia analisada

A topologia analisada nesse experimento é de um dispositivo ligado diretamente a um *time source*, que neste caso é o *gateway*, que por sua vez é o *time source* para outro

elemento da rede. Neste experimento foram utilizados apenas três nodos da rede, porém pode ser extrapolado para diversos dispositivos que dependam do nodo com falha para sincronização.

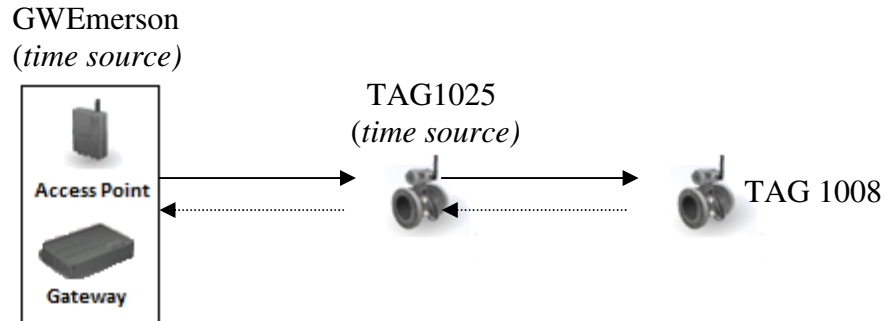


Figura 3.21: Topologia analisada na falha de defasagem do *clock*

3.2.3 Resultados do Experimento

Os testes realizados são em dispositivos com o cristal de 32.768kHz desabilitado do sistema já que é um dispositivo opcional. Isso foi realizado para que as falhas sejam evidentes com uma maior facilidade. O cristal está presente no *hardware* do dispositivo desenvolvido pelo LASCAR, logo, deve ser desabilitado por *firmware*. Isso é feito retirando a definição de `SLEEP_CLOCK_32KHZ` do código..

Utilizando a carga inicial dos capacitores, são coletadas amostras da correção do tempo ocasionada pelas mensagens de confirmação das publicações das variáveis de processo (Figura 3.22). Calculando o erro médio das amostras obtemos uma correção média de aproximadamente 34 μ s.

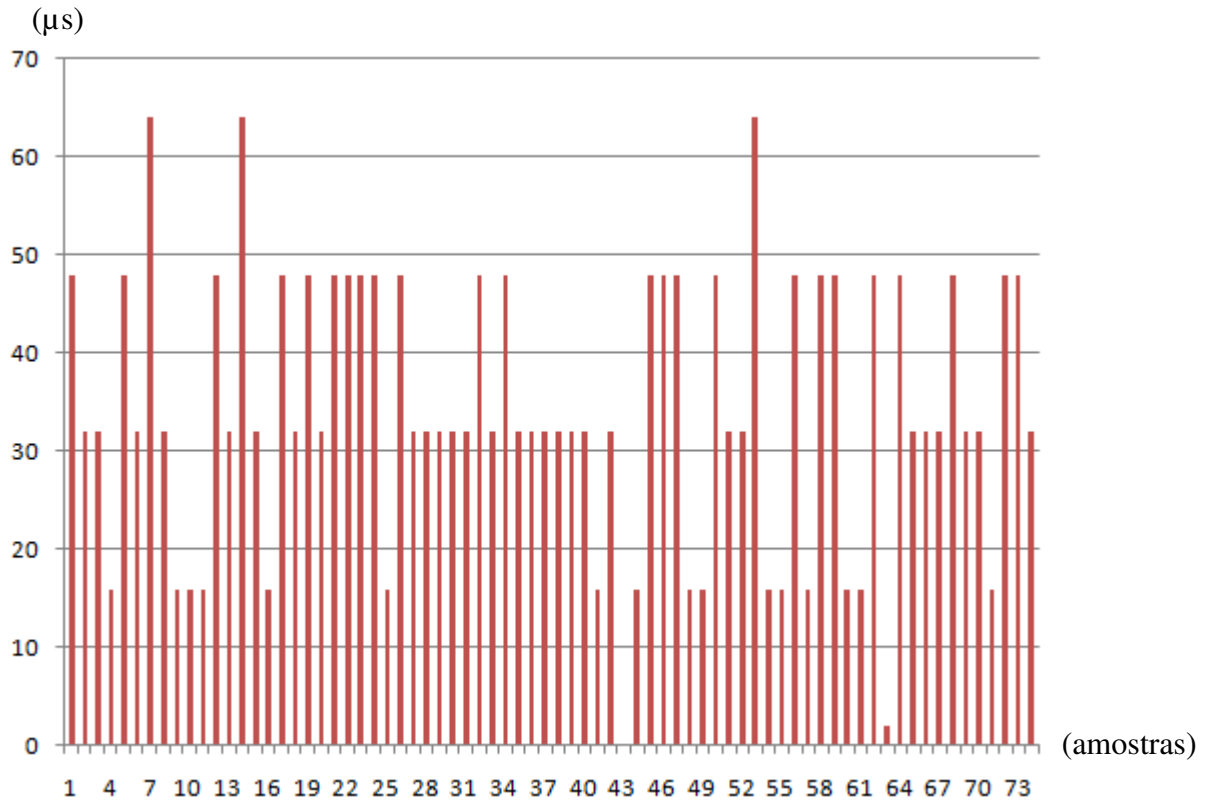


Figura 3.22: Correção do relógio (us) X n° da amostra

A seguir, o valor do registrador de ajuste fino do cristal é incrementado em cinco unidades, deixando o registrador XTAL_CTUNE contendo o valor quatro e XTAL_FTUNE igual a 21. Os valores são coletados novamente e verifica-se que o dispositivo de campo continua na rede sem problemas. O erro médio pode ser calculado como aproximadamente $-108 \mu\text{s}$.

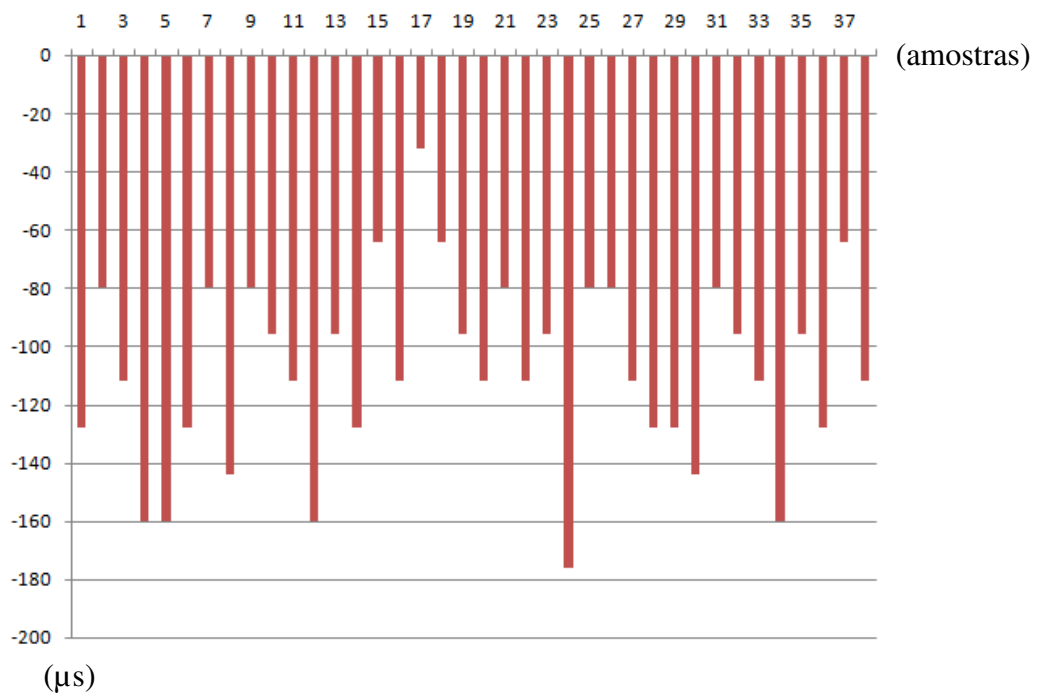


Figura 3.23: Correção do relógio (us) X n° da amostra, com CTUNE = 4 e FTUNE = 21

O processo ocorre de modo similar para os valores dos registradores incrementados de cinco em cinco unidades até que o dispositivo caia da rede. Os resultados obtidos podem ser vistos na Figura 3.24, Figura 3.25 e Figura 3.26.

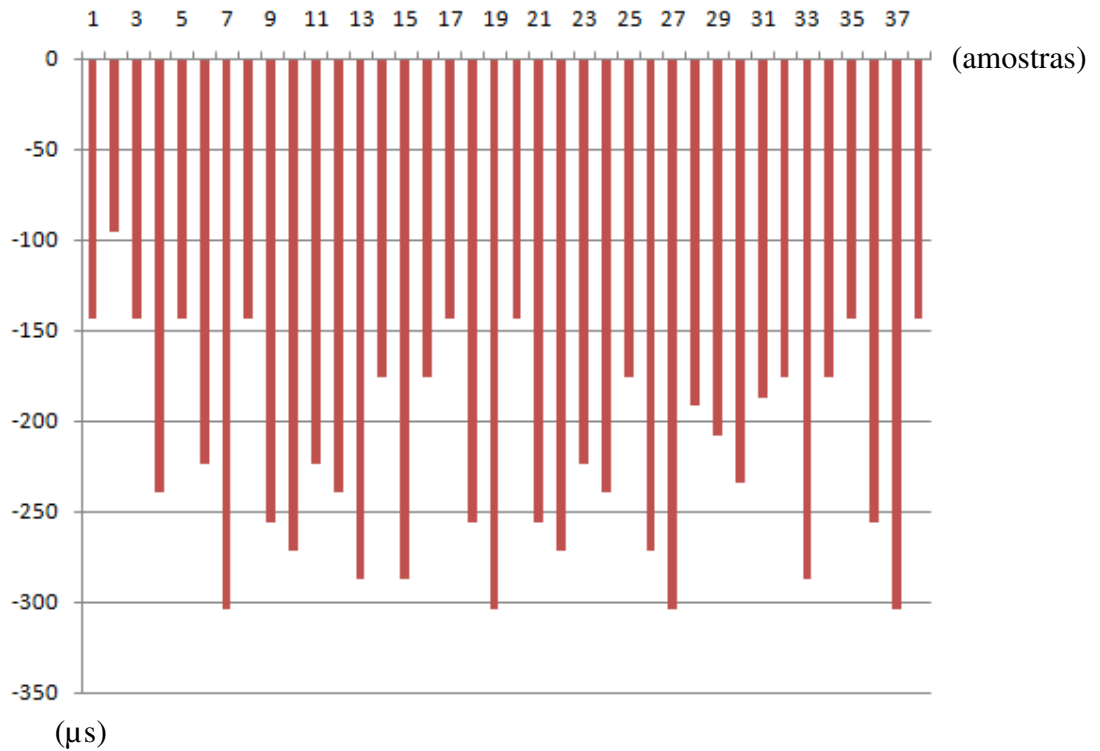


Figura 3.24: Correção do relógio (us) X nº da amostra, com CTUNE = 4 e FTUNE = 26

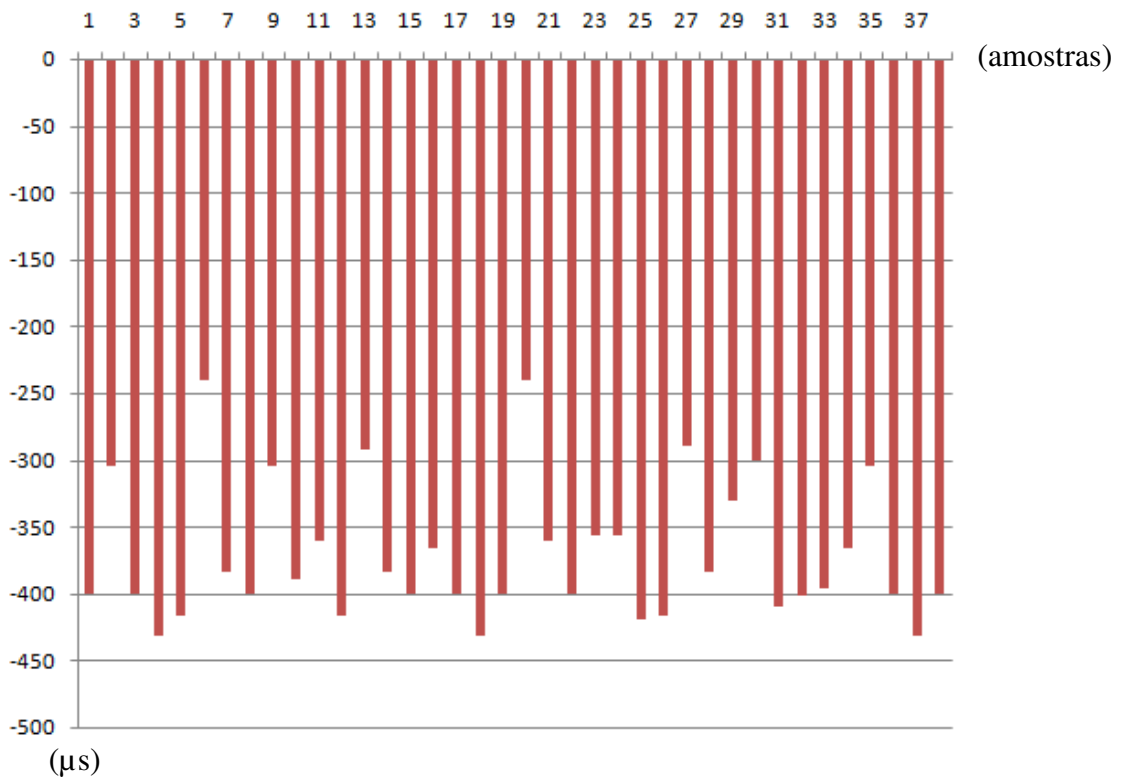


Figura 3.25: Correção do relógio (us) X nº da amostra, com CTUNE = 4 e FTUNE = 31

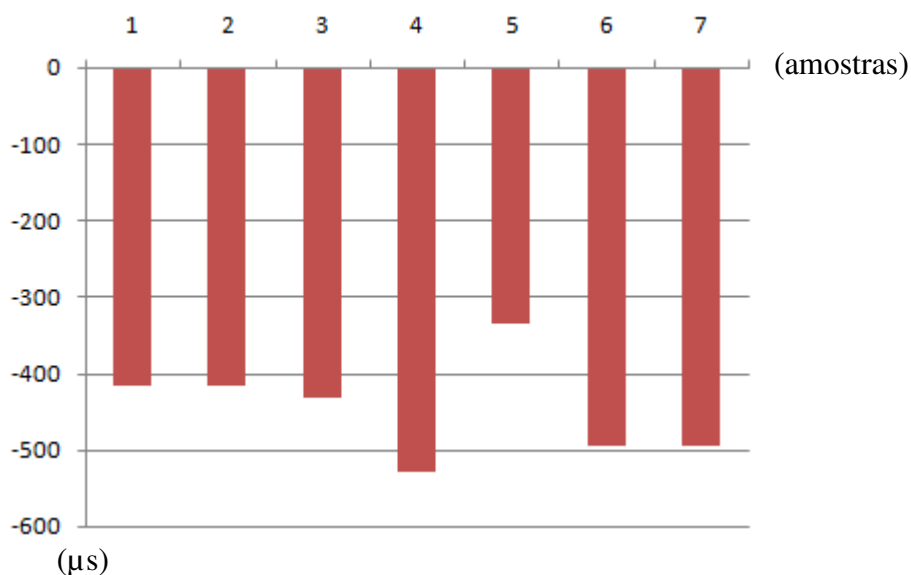


Figura 3.26: Correção do relógio (us) X nº da amostra, com CTUNE = 6 e FTUNE = 22

Os valores médios de erro obtidos nos três últimos resultados mostrados são -217 μ s, -370 μ s e -445 μ s, respectivamente. Veja que quando os registradores estão com os valores seis e 22 (Figura 3.26) só é possível coletar 7 amostras. Isso ocorre, pois nesse momento o rádio não consegue mais enviar mensagens devido a uma defasagem muito grande do seu relógio de referência.

```

Incrementing finetune by one
ftune: 22 - ; ctune: 6 |
K[0001-4]
ASN: 127173461
<-416> correction from 1
BDASN: 127175509
<-416> correction from 1
ASN: 127177729
<-432> correction from 1
K[0001-4]
ASN: 127180885
<-528> correction from 1
BDASN: 127181653
<-336> correction from 1
K[0001-4]
ASN: 127184981
<-496> correction from 1
BDASN: 127187541
<-496> correction from 1
K[0001-4]ASN: 127190613
<0> correction from 1BK[0001-4]D@#D@#D@#D@#D@#D@#-D@#K[0001-4]
D@#D@#D@#D@#D@#D@#D@#-K[0001-4]D@#D@#D@#D@#D@#D@#D@#-K[0001-4]
D@#D@#D@#D@#BD@#D@#-K[0001-4]D@#D@#D@#D@#D@#D@#D@#-K[0001-4]
D@#D@#D@#D@##D@#\-D@#D@#-K[0001-4]D@#D@#D@#D@#D@#D@#D@#-K[0001-4]
D@#D@#D@#D@#B-D@#D@#-K[0001-4]D@#D@#D@#D@#D@#D@##D@#-K[0001-4]
D@#D@#-D@#D@#D@#D@#-K[0001-4]D@#D@#D@#D@#D@#

```

Figura 3.27: Saída da depuração do dispositivo TAG 1025 quando CTUNE = 6 e FTUNE = 22

O sistema suporta uma capacitância de aproximadamente 4 pF antes de fazer com que um dispositivo não consiga mais realizar a transmissão de DLPDUs. Considerando os valores fornecidos na figura Figura 3.16 e na Figura 3.17, os valores de

XTAL_CTUNE em seis e XTAL_FTUNE em 22 representam um erro na frequência de aproximadamente 25 PPM, valor que se encontra muito acima do permitido pela norma.

Como consequência da perda de sincronismo do dispositivo TAG 1025 do restante da rede, o nodo TAG 1008 ajusta sua referência de tempo com base no dispositivo com falha e acaba saindo da rede ao mesmo tempo. Esse efeito é esperado já que os dispositivos estão em sincronia entre eles, porém não conseguem se comunicar no *time slot* correto com o *gateway*. No caso de existirem mais nodos na rede que dependam do dispositivo *time source* com falhas, provavelmente todos terão o mesmo destino.

Uma possível solução para essa falha seria a criação de um comando diferenciado que seria enviado por um dispositivo de campo quando esse estivesse realizando correções muito altas no seu relógio de referência. Se esse dispositivo fosse um *time source*, o gerenciador de rede poderia fazer com que o *time source* fosse alterado para outro dispositivo de campo próximo ao com falha. Dessa forma, o problema estaria contido em somente um nodo da rede, evitando possíveis falhas catastróficas.

4 CONCLUSÃO

O objetivo do trabalho foi alcançado, já que falhas foram injetadas na rede de forma a reproduzir o comportamento anormal obtidos por dispositivos de campo no LASCAR e a partir desses experimentos, modelos de falhas puderam ser evidenciados de uma melhor forma. Através dos modelos obtidos, a robustez da rede pôde ser analisada e algumas conclusões foram realizadas.

Testes em redes *WirelessHART* podem ser difíceis de realizar, já que por vezes não temos controle de determinados elementos da rede. Nos experimentos realizados, os *links* criados pelo gerenciador de rede entre os dispositivos de campo não podiam ser escolhidos, assim como o momento em que os *links* eram criados. Como o objetivo era reproduzir situações de erro que já haviam ocorrido com dispositivos de campo, diversas vezes a rede precisou ser refeita até que uma situação que representasse os casos desejados fosse construída.

Considerando que a formação da rede leva algo em torno de 20-30 minutos, os experimentos podem ser bem longos, especialmente quando as falhas injetadas causam a saída de dispositivos da rede e cada experimento precisa da sua re-instanciação. Esse tempo elevado tem relação com a necessidade dos testes ocorrerem em ambientes com uma rede estável, minimizando eventuais intervenções que o gerenciador de rede possa realizar durante a execução do experimento.

Alterações no *firmware* devem ser tratadas com delicadeza. Sua compreensão é difícil devido ao código ser proprietário e seu tamanho ser grande.

Dois experimentos foram realizados onde, em ambos os casos, o mau funcionamento da rede foi alcançado. Na falha de diminuição da potência do sinal, vemos que a quebra de uma antena de um dos rádios pode manifestar um erro em outro rádio. Sem um diagnóstico da rede ou uma verificação física dos rádios o problema fica mascarado, podendo ocasionar a indisponibilidade de um trecho da rede por um período de tempo maior do que o desejado. Já na falha da defasagem do *clock* de um dispositivo *time source* vemos um ponto frágil do sistema. Algumas sugestões para melhorias na estrutura da rede foram sugeridas como a inserção de informações nos pacotes enviados (como potência do sinal), gatilhos para a inicialização de diagnóstico na rede e comandos diferenciados para indicar situações de erro. Essas soluções podem ser exploradas em trabalhos futuros pensando em melhorias numa rede *WirelessHART*.

Existem diversos pontos que podem ser explorados no estudo do *WirelessHART*. A sobrecarga de pacotes que a rede sofre a cada erro é um ponto interessante, já que quando dispositivos entram e saem da rede diversas vezes, pode provocar uma sobrecarga de processamento no gerenciador de rede, podendo ser pelo recálculo da distribuição de *links*, por exemplo.

De modo geral, *WirelessHART* se mostrou um protocolo robusto em relação aos testes realizados devido a sua redundância lógica e temporal na conexão entre elementos. Contudo, a robustez do sistema poderia ser maior com uma maior inteligência tanto por parte do *gateway* como por parte dos dispositivos de campo, em identificar situações de erro e tentar evitar falhas catastróficas.

REFERÊNCIAS

AVIZIENIS, A. et al. Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 1, NO.1, Janeiro-Março 2004.

Dallas Semiconductor. Extremely Accurate SPI Bus RTC with Integrated Crystal and SRAM, Revisão 2, Outubro 2008. Disponível em <https://www.sparkfun.com/datasheets/BreakoutBoards/DS3234.pdf>

EMERSON Process Management, Gateway Smart Wireless, Folha de dados do produto, Revisão FA, Abril de 2013, Disponível em <http://www2.emersonprocess.com/siteadmincenter/PM%20Rosemount%20Documents/00813-0222-4420.pdf>

Freescale Semiconductor. Reference Oscillator Crystal Requirements for the MC1320x, MC1321x, MC 1322x, nd MC1323x IEEE 802.15.4 Devices, Revisão 1.2, D.N AN3251, 2011.

Freescale Semiconductor. MC 1322x Advanced ZigBee – Compliant SoC Platform for the 2.4 GHz IEEE 802.15.4 Standard Reference Manual, Revisão 1.4, D.N MC1322xRM, Outubro 2010.

MULLER, I. et al. Development of a WirelessHART Compatible Field Device. In: IEEE INTERNATIONAL INSTRUMENTATION AND MEASUREMENT TECHNOLOGY CONFERENCE, 2010, Austin. Proceedings. . . New York: IEEE Press, 2010. p.1430–1434.

HAN, S. et al. Wi-HTest: compliance test suite for diagnosing devices in real-time wirelesshart network. In: IEEE REAL-TIME AND EMBEDDED TECHNOLOGY AND APPLICATIONS SYMPOSIUM, 15., 2009, San Francisco. Proceedings. . . New York: IEEE Press, 2009. p.327–336.

HART Communication Foundation. HART Common Tables Specification, HCF_SPEC-183, Revisão 21.0, Maio 2011.

HART Communication Foundation. HART Communication Protocol Specification, HCF_SPEC-13, Revisão 7.3, Maio 2011.

HART Communication Foundation. TDMA Data Link Layer Specification, HCF_SPEC-075, Revisão 1.1, Maio 2008.

HART Communication Foundation. Wireless Command Specification, HCF_SPEC-155, Revisão 1.1, Maio 2008.

HART Communication Foundation. Wireless Introduction HCF LIT-131. Revisão 1.0. 2010. Disponível em www.hartcomm.org/protocol/training/training_resources_wihart.html

HERESCU, R. R. Automação residencial em redes sem fio. 2009 55f. Dissertação (Graduação em Engenharia de Computação) – Instituto de Informática, UFRGS, Porto Alegre.

HSUEH, M. C. ET AL, Fault Injection Techniques and Tools, Computer 30, no. 4, 75-82, 1997.

LIMA, C. P. Desenvolvimento de um Handheld para rede de dispositivos WirelessHART. 2011 55f. Dissertação (Graduação em Engenharia de Computação) – Instituto de Informática, UFRGS, Porto Alegre.

RECH, J. R. Desenvolvimento de um gerente de rede *WirelessHART*. 2012 55f. Dissertação (Graduação em Engenharia de Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SIQUEIRA, T. F. ET AL. Avaliação Experimental De Estratégias De Tolerância a Falhas Do Protocolo SCTP Por Injeção De Falhas. XXVII Simpósio Brasileiro De Redes De Computadores e Sistemas Distribuídos, 1:915–929, 2009.

Texas Instruments. CC2591 2.4 GHz RF Front End, Revisão A. Março 2008. Disponível em <http://www.ti.com/lit/ds/symlink/cc2591.pdf>

ANEXO <ARTIGO TRABALHO DE GRADUAÇÃO 1>

Este anexo consiste na primeira parte do trabalho de graduação em Engenharia de Computação entregue no segundo semestre do ano de 2012.

Tolerância a falhas em redes wirelessHART

Fernando Augusto Alvares de Castro e Sousa, João Cesar Netto

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{faacsousa, netto}@inf.ufrgs.br

Resumo. *Esse artigo tem como objetivo apresentar o protocolo wirelessHART e evidenciar a necessidade de uma melhor compreensão da robustez do padrão através da injeção de falhas e análise de suas consequências no sistema. Estão descritos os tipos de falhas a serem experimentados, assim como as métricas que serão utilizadas para avaliar o desempenho e a efetividade do padrão em tratar os experimentos.*

Abstract. *This paper presents a wirelessHART standard overview and shows the need of a better comprehension of its strength through fault injection and later analysis of its consequences. A set of faults that will be injected into the network will also be presented as well as the metrics needed to benchmark the effectiveness of the standard in hostile environments.*

1. Introdução

A tecnologia *wireless* tem sido utilizada cada vez mais em aplicações de ambientes residenciais, comerciais e industriais. Isso se deve ao seu custo, que decresce cada vez mais, à sua versatilidade e facilidade de implantação. No âmbito industrial, por muitos anos, redes com fio foram utilizadas pela confiabilidade que é fornecida por esse meio, já que está menos sensível à interferência, porém, com a criação de padrões de redes sem fio mais robustos, a substituição de redes com fio se torna natural.

Indústrias muitas vezes utilizam múltiplos sensores para gerenciar seu processo de produção. Redes sem fio permitem a instalação desses sensores em locais de mais difícil acesso com uma aplicação central que permite controlar variáveis desse processo e também manter a segurança do pessoal presente na indústria. Um alto compromisso com a confiabilidade de um sistema é criado.

Na banda de 2.4Ghz podem ser encontrados três tipos principais de padrões para redes *wireless*: Bluetooth, WiFi e ZigBee. Cada um possui diferentes características, distinguindo principalmente na velocidade dos dados transmitidos e na distância máxima de comunicação. Bluetooth possui uma comunicação com velocidade limitada em uma distância relativamente pequena enquanto WiFi provê uma rede sem fio de mais alta velocidade à uma distância de nível intermediário. ZigBee é um padrão alternativo às duas soluções citadas anteriormente com uma baixa velocidade de transmissão podendo comunicar a médias distâncias. O problema dessas redes é que não possuem confiabilidade e a segurança desejada em ambientes industriais. [HART Communication Foundation, 2010]

Redes wireless possuem uma sensibilidade muito grande às interferências causadas pelo ambiente em que está situada, como campos eletromagnéticos, colisão de outras redes sem fio ou objetos bloqueando o caminho de comunicação.

O padrão wirelessHART é uma solução confiável para ser utilizado em ambientes hostis como os do mundo industrial. Introduzido ao mercado em 2007 e criado por especialistas da indústria, oferece oportunidades para diferentes aplicações desde medições, segurança e produtividade dos trabalhadores até rastreamento de pessoal. A especificação do wirelessHART é focada nas funções principais de automação aonde não existiam padrões sem fio apropriados.

Entretanto para verificar a validade da robustez do padrão, testes de injeção de falhas devem ser realizados para que sejam obtidas métricas de validação significativas. Uma injeção de falhas consiste em selecionar casos específicos de teste, onde erros devem ocorrer, e verificar o estado da rede após a ocorrência dos mesmos. O protocolo pode reagir de três formas distintas: detecção de um estado errôneo, recuperação do mesmo ou mau funcionamento.

Conhecimento sobre a robustez de um padrão influi em decisões importantes na implantação da rede como, por exemplo, no seu dimensionamento, que por sua vez possui um impacto direto no custo e tempo que seria necessário para o funcionamento da rede.

Este artigo contém uma proposta à verificação da confiabilidade do padrão wirelessHART, utilizado na gerência de processos industriais, através da análise de seu comportamento em um ambiente com falhas controladas. Também serão descritas falhas e métricas que serão utilizadas nessa avaliação.

2. Protocolo WirelessHART

2.1. Visão Geral

WirelessHART é um padrão de comunicação introduzido ao mercado no ano de 2007. Seu objetivo é fornecer uma rede segura para controle de processos industriais. Foi criado como um avanço da tecnologia HART com o intuito de estendê-la para a comunicação sem fio. Uma rede wirelessHART contém três elementos principais.

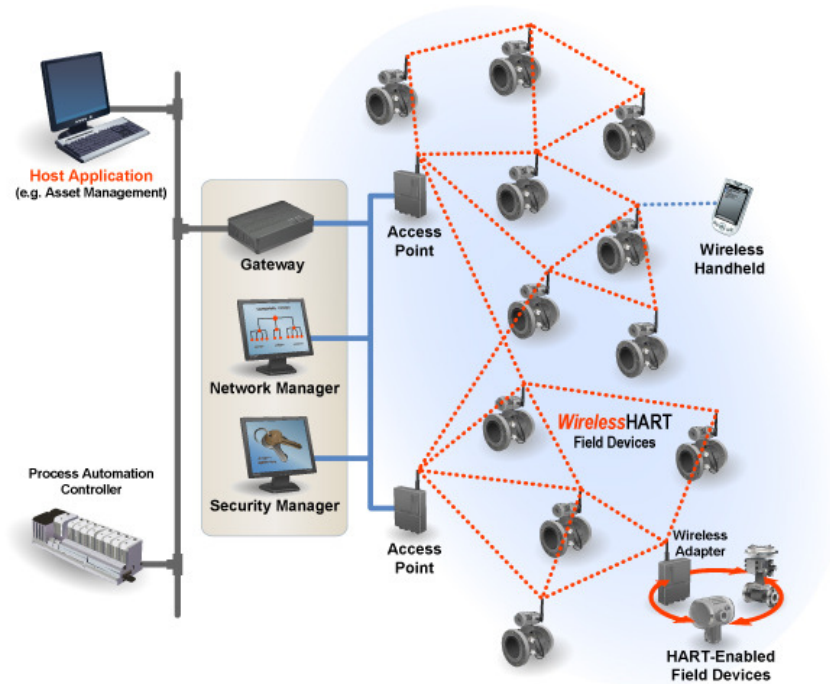
Field Devices: São os dispositivos de campo integrantes da rede. Podem estar conectados a equipamentos ou sensores. Esses dispositivos são capazes de atuar na rede como emissores e receptores de pacotes, assim como roteadores de pacotes enviados por outros integrantes da rede.

Gateway: Componente central de uma rede wirelessHART. Provê a ligação com outras redes permitindo a comunicação e troca de mensagens entre elas

Network Manager: Gerenciador que configura a rede, escalona a comunicação entre os dispositivos, controla as rotas de comunicação e monitora o estado da rede como um todo. Este elemento pode estar integrado no Gateway, na aplicação Host ou no controlador do processo de automação.

Outros: Existem outros elementos pertencentes à rede como adaptadores, que são responsáveis por adaptar dispositivos HART com fio ao padrão sem fio, e *Handhelds*, que são dispositivos portáteis utilizados para monitorar o funcionamento da rede.

A figura abaixo exemplifica como os elementos citados acima são integrados na rede.



1. Arquitetura de uma rede wirelessHART [HCF 2010]

Uma rede wirelessHART opera através de rádios na frequência de 2.4GHz. Os rádios utilizam tecnologia de espalhamento espectral e *channel hopping* para aumentar a segurança e confiabilidade do sistema. *Time Division Multiple Access* (TDMA) também é utilizado para evitar colisões de comunicação e melhorar o desempenho do sistema.

O padrão é utilizado em uma rede *mesh*. Em uma rede *mesh* um dispositivo de campo pode atuar tanto como fonte de pacotes como roteador para mensagens provenientes de outros dispositivos, isso significa que, individualmente, não é necessário que cada um seja capaz de comunicar diretamente com o gateway. Isso implica em uma rede muito mais flexível na adição e remoção de nodos e em um aumento considerável na tolerância a falhas da rede, já que podem existir caminhos de comunicação redundantes entre dois nodos da rede. [HART Communication Foundation 2009]

2.2. Tolerância a falhas

O padrão wirelessHART define várias técnicas que são utilizadas para aumentar a robustez da comunicação, cada uma delas realizando a proteção em níveis diferentes como segurança dos dados transmitidos, tolerância à queda de nodos ou interferência ocasionada pela presença de outras redes sem fio. Abaixo estão descritos alguns dos itens utilizados para alcançar a robustez que o padrão oferece.

AES: A comunicação foi projetada para permitir uma rede sem fio industrial segura e por essa razão é codificada com sessões entre nodos utilizando AES-128 bits. Isso assegura que somente o destino final da mensagem pode interpretar os dados da mensagem.

Channel Hopping: A frequência de comunicação é alterada a cada comunicação para evitar interferência de outras redes sem fio.

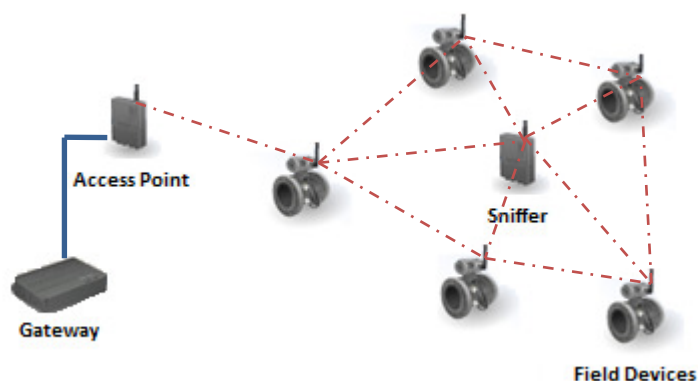
Autoconfiguração da rede: As rotas de comunicação são ajustadas automaticamente para melhorar o desempenho. Instrumentos adicionados à rede são automaticamente ajustados. O monitoramento das rotas de comunicação é realizado constantemente para verificar degradação e, no caso de obstrução de algum, caminhos alternativos são buscados.

3. Experimento Proposto

3.1 Proposta

A proposta deste trabalho é uma análise da tolerância a erros do padrão wirelessHART. Como já foi citado, uma rede wirelessHART deve ser segura e capaz de tolerar diversas interferências externas, as quais podem causar diversos erros de comunicação e mau funcionamento de outros dispositivos da rede. Cada ocorrência dessas falhas faz com que comportamentos fora do padrão sejam acionados para que a rede retorne a um estado normal. Logo, testes devem ser realizados com situações contraladas para avaliar e quantificar a robustez do padrão wirelessHART.

Casos controlados de testes podem ser alcançados através da alteração do *firmware* de um dispositivo de campo que fará com que o nodo da rede se comporte de acordo com o caso de teste desejado e injete falhas no sistema. Durante a injeção de falhas o comportamento da rede deve ser observado e registrado para que possa ser feita uma análise posterior dos dados. Essa coleta pode ser feita através de um *sniffer* de rede que fará a análise do roteamento dos pacotes transmitidos pelos dispositivos integrantes da rede, nos permitindo obter métricas e quantificar a robustez do padrão. Valores registrados pelo *network manager* também podem ser relevantes na análise de resultados de cada experimento. Os tipos de falhas que serão utilizados, assim como as métricas estão detalhados no item quatro.



2. Exemplo da topologia da rede

A figura acima exemplifica a topologia onde será realizada a injeção de falhas. Podem-se identificar três elementos principais distintos: o gateway, nodo central da rede onde o *network manager* é integrado; dispositivos de campo e o *sniffer* de rede, realiza a escuta e análise do roteamento dos pacotes enviados pelos dispositivos de campo.

É importante ressaltar que para que os resultados sejam estatisticamente relevantes cada teste deve ser realizado múltiplas vezes para que a influência de fatores não controlados nos resultados seja diminuída.

3.2 Objetivos

Como foi descrito no item um, o padrão wirelessHART foi desenvolvido para oferecer uma comunicação sem fio robusta em um ambiente industrial, onde existem diferentes tipos de interferências externas como, por exemplo, campos eletromagnéticos. O objetivo deste trabalho é obter métricas significativas sobre o comportamento da rede em condições adversas e, através do conhecimento obtido, facilitar a tomada de decisões na implantação de uma rede.

4. Injeção de falhas

4.1. Sistema para injeção

A infraestrutura para a injeção de falhas será disponibilizada pelo Laboratório de Automação e Implantação de Sistemas (LAIS), pertencente a Universidade Federal do Rio Grande do Sul. O LAIS já possui uma *stack* do padrão wirelessHART desenvolvida e dispositivos de campo disponíveis para a criação de uma rede.

Os testes de injeção de falhas precisarão de um nodo com o *firmware* modificado que permita receber comandos para injetar as falhas descritas na seção 4.2.

4.2. Tipos de falhas

O wirelessHART é composto por uma rede *mesh* onde cada nodo pode se comportar como um roteador. Essa característica permite que qualquer um dos nodos possa ter comportamento indesejado causando falhas na rede. Na literatura [Hsueh, M. C., T. K. Tsai, e R. K. Iyer. 1997] podem ser encontrados erros típicos de protocolos de comunicação que podem ser explorados nos experimentos de verificação do padrão.

Crash de um nodo: Uma das técnicas mais simples onde um nodo com comportamento vazio é inserido na rede. Este dispositivo é programado para não responder as mensagens que lhe são enviadas fazendo com que o sistema tenha que encontrar um caminho alternativo ao seu destino.

Omissão de mensagens: Esta técnica se caracteriza no envio da confirmação de recebimento de um pacote pelo dispositivo, porém sem o repasse da mensagem para o destinatário.

Retardo de envio: Ao receber um pacote o dispositivo deve aguardar um tempo definido pelo caso de teste para repassá-lo. Tempos variados devem ser utilizados para analisar o comportamento da rede na presença de dispositivos com, por exemplo, gargalo no processamento.

Duplicação de pacotes: O dispositivo transmite duas vezes a mesma mensagem.

Corrupção de mensagens: A corrupção de mensagens consiste em alterar bits dos pacotes transmitidos. Essa técnica pode ser pouco efetiva e difícil de ser observada, pois os pacotes alterados podem ser descartados silenciosamente.

4.3. Métodos de análise dos resultados

Após a injeção de falhas, um sniffer será utilizado para coletar os dados transmitidos pela rede nos permitindo estabelecer métricas de comparação com casos típicos sem falhas. Outro recurso que pode ser utilizado é a análise dos dados coletados pelo network manager de desempenho da rede e também o número de pacotes recebidos corretamente em uma transmissão. Com isso podem ser evidenciados alguns pontos [Siqueira, Fiss, Menegotto, Cechin, e Weber 2009] importantes para que os resultados sejam significativos.

Número de falhas recuperadas: Esse parâmetro indica o número de transmissão de pacotes em que foram injetadas falhas e conseguiram chegar ao nodo destino com sucesso.

Número de falhas detectadas: Indicam quantas falhas o protocolo detecta que ocorreu uma anormalidade, mas não consegue retransmitir para o nodo destino.

Sobrecarga da troca de mensagens: Deve ser analisado qual a sobrecarga de pacotes que ocorre no sistema para cada falha injetada. A sobrecarga é um dado importante, pois pode ocasionar no mau funcionamento de outros dispositivos que não estejam sendo controlados pela injeção de falhas devido ao aumento da latência das mensagens por exemplo.

Tempo de recuperação do sistema: Um dos fatores interessantes a serem avaliados é o tempo de recuperação do sistema para falhas em que o sistema consegue se recuperar.

Número de nodos falhos que a rede suporta: O funcionamento da rede depende de uma relação entre nodos funcionais e falhos. Existe um limiar aonde pacotes não chegam ao seu destino devido à sobrecarga na rede ou falta de caminhos possíveis para a entrega. Através dos experimentos deve ser possível determinar uma quantidade para essa variável.

5. Cronograma

Tarefa/Mês	Jan	Fev	Mar	Abril	Mai	Junho
(1)						
(2)						
(3)						
(4)						
(5)						

Tabela 1. Cronograma para a segunda parte do Trabalho de Graduação

- (1) **Análise da topologia da rede:** Através da utilização do *sniffer* deve ser feita uma análise preliminar da topologia e comportamento da rede para que os efeitos da injeção de falhas possam ser maximizados ou minimizados de acordo com o necessário.
- (2) **Alteração do firmware para injeção de falhas:** O código fonte do *firmware* de um dispositivo deve ser alterado para permitir o controle de seu comportamento.
- (3) **Injeção de falhas:** Injeção de falhas em diferentes topologias com um ou mais dispositivos com o *firmware* alterados.
- (4) **Análise de resultados:** Interpretação dos resultados e comparação com os esperados.
- (5) **Escrita da monografia:** Escrita da monografia contemplando os passos necessários para realização dos experimentos, assim como os resultados alcançados.

6. Conclusões

O padrão wirelessHART é definido por uma rede *mesh* onde cada dispositivo dessa rede pode atuar tanto como origem de uma comunicação, como roteador de pacotes sendo transmitidos. O padrão largamente utilizado no ambiente industrial e, por esse motivo, está sujeito às mais diferentes formas de interferência na sua comunicação. O impacto dessa interferência deve ser avaliado para que seja considerada na implantação de redes.

Através do controle de um dos dispositivos presentes na rede podemos realizar experimentos de injeção de falhas explorando diferentes características do padrão que nos permitirá analisar a robustez oferecida pelo mesmo.

Com a definição dos pontos chave citados nesse artigo, pode-se seguir para o desenvolvimento dos experimentos na segunda etapa do trabalho de graduação.

Referências

Siqueira, T. F., B. Fiss, C. C. Menegotto, S. Cechin, and T. S. Weber. "Avaliação Experimental De Estratégias De Tolerância a Falhas Do Protocolo SCTP Por Injeção

De Falhas.” In XXVII Simpósio Brasileiro De Redes De Computadores e Sistemas Distribuidos, 1:915–929, 2009.

Hsueh, M. C., T. K. Tsai, and R. K. Iyer. “Fault Injection Techniques and Tools.” *Computer* 30, no. 4 (1997): 75–82.

HART Communication Foundation; Wireless Introduction HCF LIT-131, Revision 1.0. 2010. Disponível em:
www.hartcomm.org/protocol/training/training_resources_wihart.html

HART Communication Foundation; WirelessHART Technology. Disponível em:
www.hartcomm.org/protocol/wihart/wireless_technology.html

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