Universidade de São Paulo

Escola de Engenharia de São Carlos

RENATO DA VEIGA TORRES

Simulador de Redes Profibus

São Carlos

2013

Renato da Veiga Torres

Simulador de Redes Profibus

Dissertação apresentada como parte dos requisitos do título de mestre em Ciências, Programa de Engenharia Elétrica da Escola de Engenharia de São Carlos - Universidade de São Paulo.

Área de Concentração: Sistemas Dinâmicos.

Orientador: Prof. Dr. Dennis Brandão

São Carlos

2013

Trata-se da versão corrigida da dissertação. A versão original encontra-se disponível na EESC/SP que aloja o Programa de Pós-Graduação em Engenharia Elétrica.

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

T691s

Torres, Renato da Veiga
Simulador de redes Profibus / Renato da Veiga Torres ; orientador Dennis Brandão. São Carlos, 2013.

Dissertação (Mestrado - Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Sistemas Dinâmicos)-- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2013.

1. Redes industriais. 2. Simulador de redes. 3. Redes Profibus DP. 4. Diagnóstico de redes. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **RENATO DA VEIGA TORRES.**

Título da dissertação: "Simulador de redes profibus DP".

Data da defesa: 19/09/2013

Comissão Julgadora:	**Resultado:**
Prof. Associado **Dennis Brandão (Orientador)** (Escola de Engenharia de São Carlos/EESC)	aprovado
Prof. Dr. **Orides Morandin Junior** (Universidade Federal de São Carlos/UFSCar)	APROVADO
Prof. Dr. **Danilo Hernane Spatti** (Escola de Engenharia de São Carlos/EESC)	Aprovada

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica e Presidente da Comissão de Pós-Graduação:
Prof. Titular **Denis Vinicius Coury**

Agradecimentos

Agradeço a Deus e a minha família. Forças presentes em todas as horas que me motivam sempre.

Agradeço ao professor Dr. Dennis Brandão, pela orientação, oportunidade e paciência.

Ao amigo Eduardo Mossin, que me escalou para ajudá-lo mas que ao fim me ajudou muito mais.

À Universidade de São Paulo, por colocar à disposição sua estrutura.

À Divisão de Desenvolvimento da Empresa SMAR Equipamentos Industriais que financia e valoriza este investimento.

Resumo

TORRES, R.V. (2013). **Simulador de redes Profibus**. 106p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

Este trabalho propõe o desenvolvimento de um simulador de redes Profibus DP que reproduz o comportamento de uma rede real. A simulação de redes em condições de falha permite aos usuários experimentar situações que podem ocorrer em instalações reais, facilitando o treinamento de manutenção e antecipando ações corretivas. O simulador permite o treinamento de usuários que utilizam ferramentas de diagnóstico. Auxilia no estudo do protocolo Profibus DP e também contribui com o desenvolvimento da ferramenta Profidoctor que é um sistema especialista com o objetivo de fazer a análise de problemas de rede.

Palavras chave: redes industriais, simulador de redes, redes Profibus DP.

Abstract

TORRES, R.V. (2013). **Profibus Network Simulator**. 106p. Thesis (Master) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2013.

This work aims development of Profibus DP network simulator that reproduces real network behavior. The network simulation under fault conditions allows experiment situations that may occur in real network installations, helping in maintenance training and corrective actions. This simulator tool helps in diagnostic tools users training, Profibus DP protocol learning and also assists development of new tool Profidoctor: An expert system that will report complete analysis of network problems.

Keywords: industrial network, network simulator, Profibus DP.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

DP *Descentralized Periphery* - Periferia Descentralizada

FDL *Fieldbus Data Link* - Camada de Enlace

GSD *General Slave Data* - Arquivo de Descrição do Escravo

IP *Internet Protocol* - Protocolo de Internet

OPEX *OPerating EXpense* - Custo Operacional

PLC *Programmable Logic Controller* - Controlador Lógico Programável

PA *Process Automation* - Automação de Processo

SAP *Service Access Point* - Código de Serviço

TCP *Transmission Control Protocol* - Protocolo de Controle de Transmissão

Sumário

Capítulo 1

Introdução

A atual queda nos custos dos microcontroladores, o contínuo aumento de desempenho dos processadores, a popularização dos computadores em ambientes industriais e o desejo constante da melhoria da eficiência produtiva impulsionam o aumento da aplicação de instrumentos e a expansão do uso de redes digitais dedicadas no ambiente industrial.

Tais fatores influenciam diretamente na queda de custos de aquisição e implantação de sistemas de automação, assim como o aumento da quantidade de pontos de medição e análise em processos industriais.

Redes digitais permitem reduzir drasticamente a quantidade de fios destinados à transmissão de dados e elevam a confiabilidade da qualidade dos sinais medidos. Isso gera uma redução de OPEX, no que diz respeito à manutenção de infraestrutura física de automação. A evolução da "inteligência"dos equipamentos facilita a comunicação vertical dentro da infraestrutura de comunicação industrial (veja Figura 1.1): permitindo a implantação de planos de manutenção preditiva, viabilização de estratégias de negócio e impactando diretamente no aumento da disponibilidade de máquinas e processos.

A confiabilidade das redes industriais está fortemente relacionada à qualidade da instalação e a correta configuração dos equipamentos. As regras e normas são definidas e controladas de acordo com a instituição ou associação que representa cada padrão - em geral formada por um conjunto de fabricantes que buscam interoperabilidade entre si. Estes fabricantes objetivam garantir fatias de mercado formado por usuários que selecionam o uso destes padrões buscando benefícios como: acesso a uma gama maior de fornecedores de instrumentos sem precisar trocar o sistema de automação, um critério para filtrar melhores fornecedores e outros benefícios em

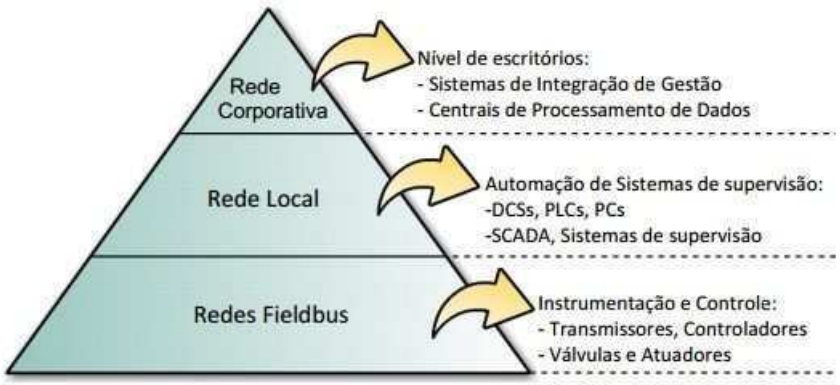

Figura 1.1: Representação da infraestrutura de comunicação industrial - (Brandão, 2005)

ganhos de processo que foram anteriormente citados.

Estes requisitos deram origem a um mercado de automação formado por diversos fornecedores de equipamentos, softwares e serviços especializados no treinamento de pessoal, certificação de instalações e monitoração continua da qualidade das redes em operação.

Como a "saúde"das redes digitais pode influenciar diretamente o desempenho produtivo, faz sentido o investimento em ferramentas destinadas a esta monitoração e certificação das instalações.

A evolução destas ferramentas tem fornecido uma quantidade de medições e dados das redes onde estas são conectadas, porém, a análise destes dados exige sempre um especialista no protocolo em questão. A Figura 1.2 apresenta a arquitetura utilizada pelas ferramentas de captura de frames de rede.

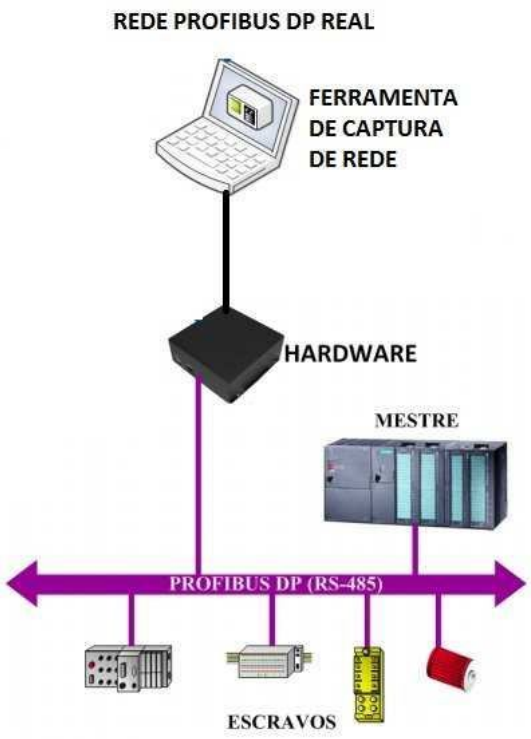

Figura 1.2: Interface de captura de rede Profibus DP e rede real

Dentro deste contexto, visando facilitar o treinamento de especialistas no diagnóstico de problemas de rede e conhecimento do protocolo Profibus este trabalho tem por objetivo o desenvolvimento de uma ferramenta que permite simular o comportamento de redes Profibus DP, onde se é possível observar as iterações dos equipamentos em uma rede com problemas.

Outros trabalhos vêm sendo desenvolvidos na área de diagnósticos de redes industriais, visando fornecer diretivas para melhorar a qualidade da instalação, prover uma análise do sinal de comunicação e criar indicadores que permitam avaliar se as normas que visam o funcionamento de uma rede industrial estão sendo cumpridas.

Essa dissertação propõe um ambiente que permite a simulação das redes Profibus DP (Figura 1.3), tornando possível avaliar os efeitos das falhas de comunicação na operação da rede e verificar como o protocolo Profibus DP opera em diferentes casos de falha.

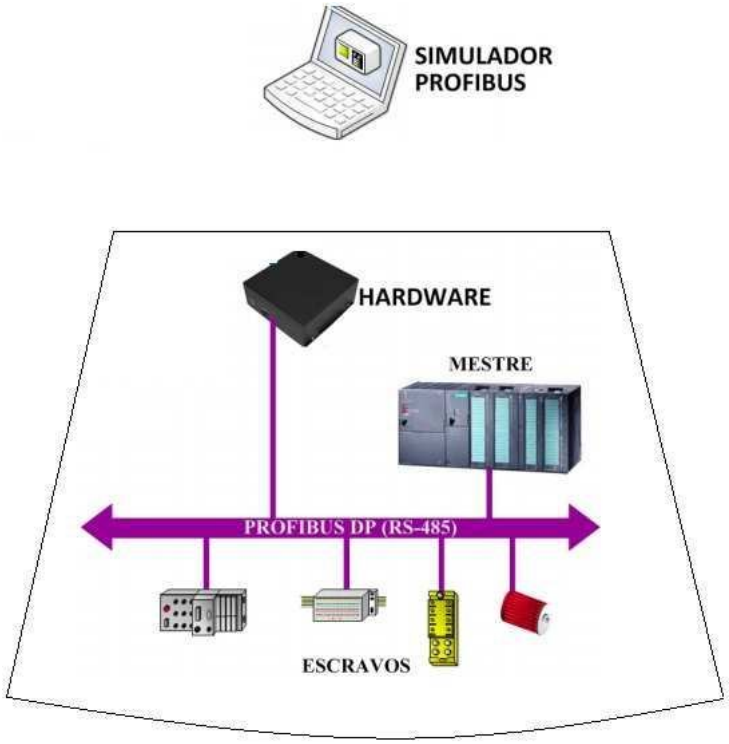

Figura 1.3: Simulador com mesmo aspecto da ferramenta de captura e a rede simulada

1.1 Objetivos do Trabalho

O primeiro objetivo deste trabalho é o desenvolvimento de um simulador de redes Profibus DP cuja função principal é examinar diferentes comportamentos de redes industriais frente a problemas e configurações que possam prejudicar a comunicação e avaliar como o protocolo se comporta nestas situações. Busca-se isso através do desenvolvimento de uma interface intuitiva que permita ao usuário compreender e visualizar o funcionamento da camada de enlace de uma rede Profibus DP através da captura de frames.

Este trabalho visa o estudo especifico do Profibus DP em RS-485, que é o meio físico original e mais popular.

O conteúdo e abordagem sobre redes Profibus DP podem envolver assuntos que variam desde linhas de transmissão para modelamento da camada física até engenharia de software com foco no desenvolvimento de aplicações relacionadas. A ênfase deste trabalho é o protocolo - camada de enlace (FDL) e camada de aplicação. Estes compõe o foco do simulador, pois são os elementos que quando afetados devido a uma distorção do canal de comunicação, resultam na interrupção e tomada de decisões nas estações da rede.

O critério utilizado para verificar se este objetivo principal foi atingido é avaliar o conteúdo produzido pelo simulador em relação a uma captura real de pacotes de uma rede, verificando quão próximo o simulador consegue representar o comportamento de uma rede existente.

Como entrada, o simulador utiliza (faz a leitura) de arquivos de configuração do configurador Hilscher Sycon (arquivos formato *.pb). O mesmo software industrial utilizado para configurar mestre Profibus DP em redes reais.

1.2 Organização do Trabalho

No Capítulo 2 é apresentada uma breve introdução sobre redes Profibus DP. É definido o escopo neste assunto que aborda aspectos da norma Profibus DP interessantes para o desenvolvimento e compreensão deste trabalho. No Capítulo 3 é realizado um levantamento dos simuladores e trabalhos dedicados à simulação de redes Profibus DP.

O capítulo 4 apresenta as características e o desenvolvimento da ferramenta de simulação. O Apêndice apresenta as telas numa melhor resolução para facilitar o entendimento.

O Capítulo 5 apresenta a metodologia usada para avaliar o funcionamento do simulador e resultados dos testes realizados frente a capturas de frames de redes reais em operação.

O Capítulo 6 faz as considerações finais, apresentando propostas para trabalhos futuros.

Capítulo 2

Redes Profibus

Nos últimos anos, o nível de informação na indústria sofreu mudanças propagando adição de características nos equipamentos de campo e softwares dedicados. A principal mudança está ligada a necessidade de integração entre os equipamentos dispostos no processo e os softwares utilizados na configuração e supervisão. Tal integração garante à indústria uma maior confiabilidade em seus processos.

As redes Profibus foram difundidas de maneira global desde 1989 com o principal objetivo de descentralizar as conexões de sensores e atuadores para estações remotas, onde estas seriam posteriormente levadas a um controlador através desta rede. Profibus DP significa *Descentralized Perifery* fazendo alusão à possibilidade de descentralizar as conexões de sensores e atuadores, evitando cabeamento de sinais analógicos muito longos que são prejudiciais à integridade das medições. Segundo a Organização Profibus (Profibus Nutzorganisation E. V.), que regula e especifica as normas relacionadas ao padrão em questão, ao fim de 2009, foram contabilizados cerca de 30 milhões de equipamentos com este protocolo e presentes nos mais diferentes ramos industriais (PROFIBUS, 2010).

A Figura 2.1 apresenta uma comparação entre a instalação requerida em um sistema convencional e em um sistema com Profibus DP. Nota-se que em um sistema convencional centralizado deve-se prolongar o cabo de cada um dos transmissores do campo até o painel onde se encontra o controlador. Enquanto que no uso das redes Profibus DP é necessário apenas um cabo de dados passando por cada uma das estações remotas. Reduzindo o comprimento dos cabos analógicos (contribuindo a integridade do sinal medido) e levando as informações dos sensores através deste cabo ao controlador.

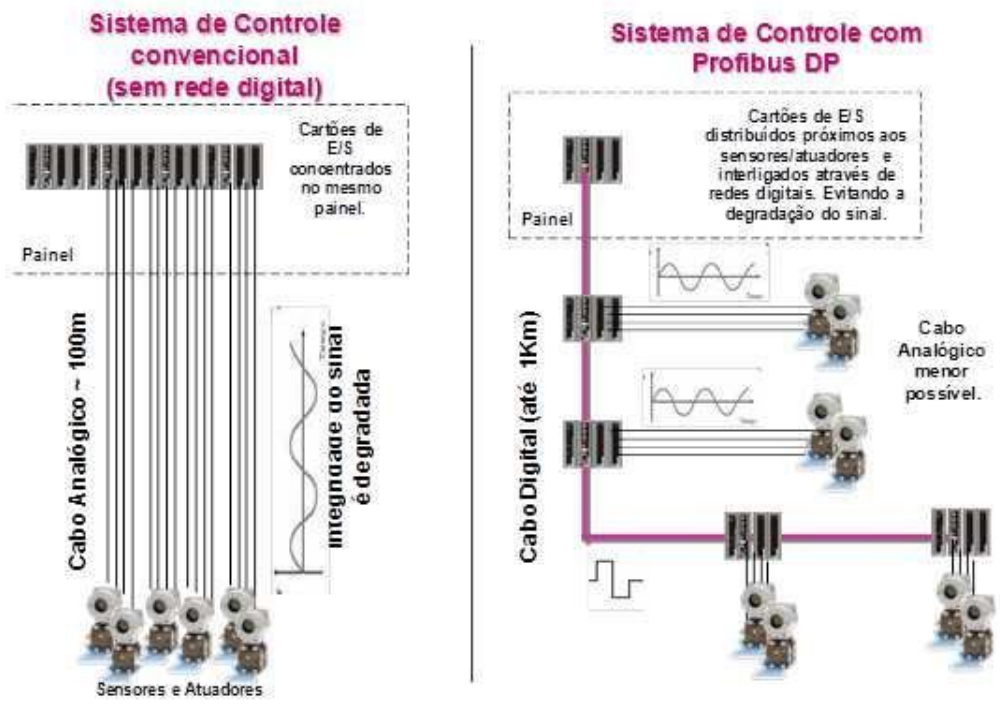

Figura 2.1: Vantagens na instalação utilizando Profibus DP.

Com o passar dos anos e evolução dos equipamentos, a inteligência de comunicação passou a ser integrada diretamente nos próprios instrumentos, permitindo a monitoração remota de diagnósticos e o protocolo que fora criado somente para integrar estações remotas que concentravam equipamentos convencionais passou a integrar diretamente os dispositivos finais como: relés, inversores, medidores e transmissores, conforme a Figura 2.2.

Equipamentos que antes apresentavam somente interfaces de comunicação analógica e proprietária, exigindo conversores de tensão e corrente para permitir seu controle, ficaram acessíveis de maneira padronizada através das redes digitais em geral.

Utilizando ainda os recursos do protocolo Profibus DP, os fabricantes de dispositivos tornaram acessíveis remotamente parâmetros que eram acessíveis apenas no próprio equipamento, que podem ser configurados previamente ou durante a operação, permitindo também uma parametrização automática em novos equipamentos quando houver necessidade de substituição devido à falha ou manutenção.

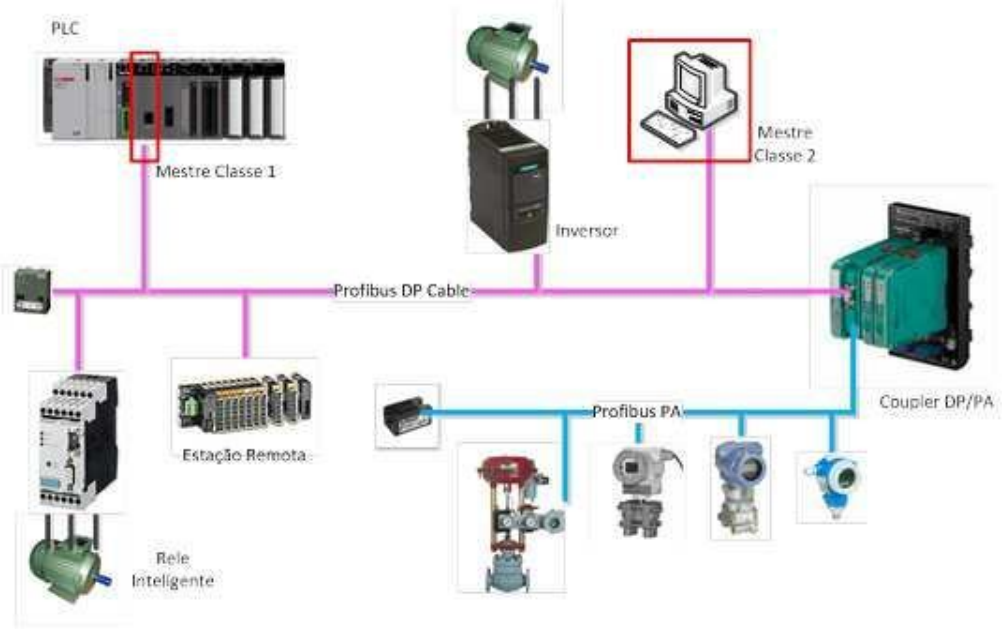

Figura 2.2: Conjunto de equipamentos comumente encontrados em redes Profibus DP.

Tentando prover solução para todas as áreas da indústria e proteger o investimento dos fabricantes de equipamentos e clientes finais, a Organização Profibus fundamentou sua interoperabilidade em uma camada de enlace simples e bem especificada através de uma norma geral que incorpora desde a especificação da camada física e sugestão de arquiteturas de hardware.

A norma foi baseada no modelo ISO/OSI com um número reduzido de camadas visando simplificar as implementações de mestres e escravos levando em conta o que as aplicações exigiam. Esta estrutura pode ser vista na Figura 2.3.

Diferente de outros protocolos existentes no mercado observa-se, na prática, que problemas de interoperabilidade são inexistentes ou são contornáveis através de modificações em configuração, sem a necessidade de substituição ou atualização de equipamentos ou infraestrutura de rede.

Os serviços de parametrização e configuração foram recebendo extensões ao longo do tempo para contemplar diferentes novas aplicações e prover evolução ao protocolo mantendo sua compatibilidade.

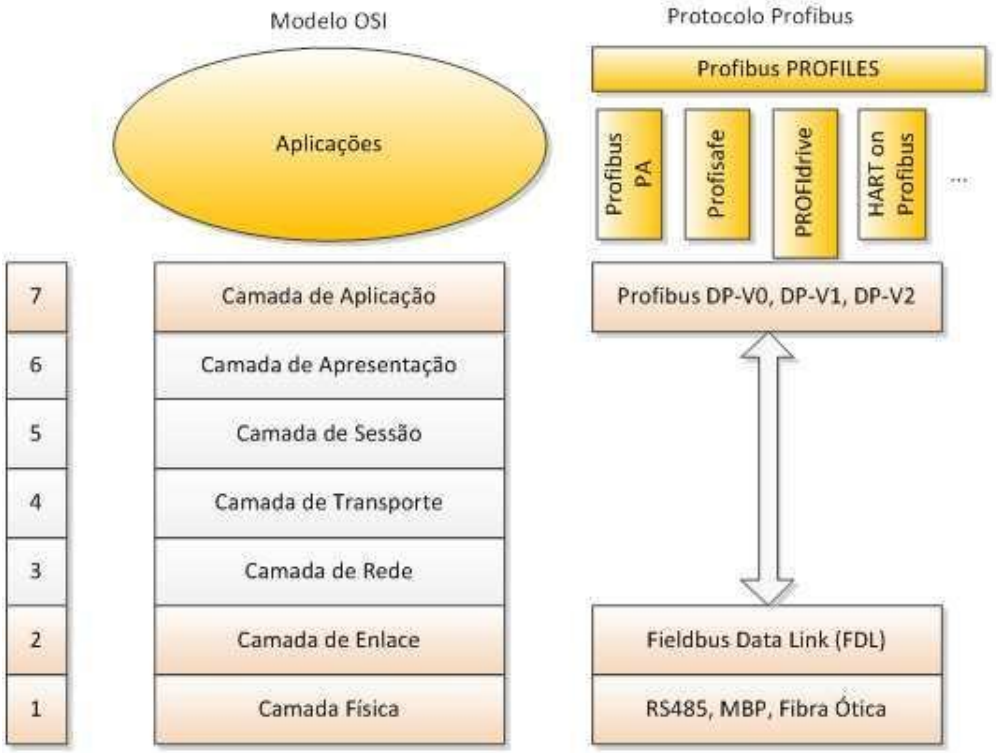

Figura 2.3: Modelo do Protocolo Profibus segundo OSI/ISO.

Com objetivo de abranger diversos mercados, abrangendo as necessidades e requisitos, foram desenvolvidas novas versões da camada de aplicação (DP-V1 e DP-V2) que adicionaram diferentes características (de forma incremental), permitindo aos fabricantes de equipamentos e clientes utilizar a versão DP-V0 (inicial) ou focar a atenção nas características presentes nas novas versões, caso necessário.

O aumento de inteligência dos dispositivos levou a Organização Profibus, em conjunto com os fornecedores de equipamentos, desenvolver normas que sugerem o funcionamento da camada de aplicação, como um guia de estilo dedicado às aplicações mais frequentes onde o protocolo é aplicado - Profiles. Como por exemplo: Aplicações de medição de nível comportam diferentes tipos de medidores de nível (barométrico, radar, sonar, chave de nível...). O profile para medição de nível sugere nomes e tipos de variáveis para cada tipo de medidor de nível.

Os Profiles possibilitam aos fabricantes a padronização da camada de aplicação de seus equipamentos (Figura 2.3) permitindo aos clientes finais o desenvolvimento de configurações de controle sem a necessidade de saber previamente exatamente qual o fornecedor de um equipamento. Permite ainda a substituição de um fornecedor por outro durante a operação sem

impacto em projeto ou configuração.

A comunicação adequada entre as estações depende, dentre outros requisitos, do funcionamento correto da camada física normatizada para o protocolo Profibus. O hardware que compõe a conexão de cada um dos equipamentos dispõe de uma unidade de acesso ao meio (MAU) que é responsável pela adequação dos sinais digitais (Figura 2.4). Isso permite a comunicação entre estações distantes de maneira robusta.

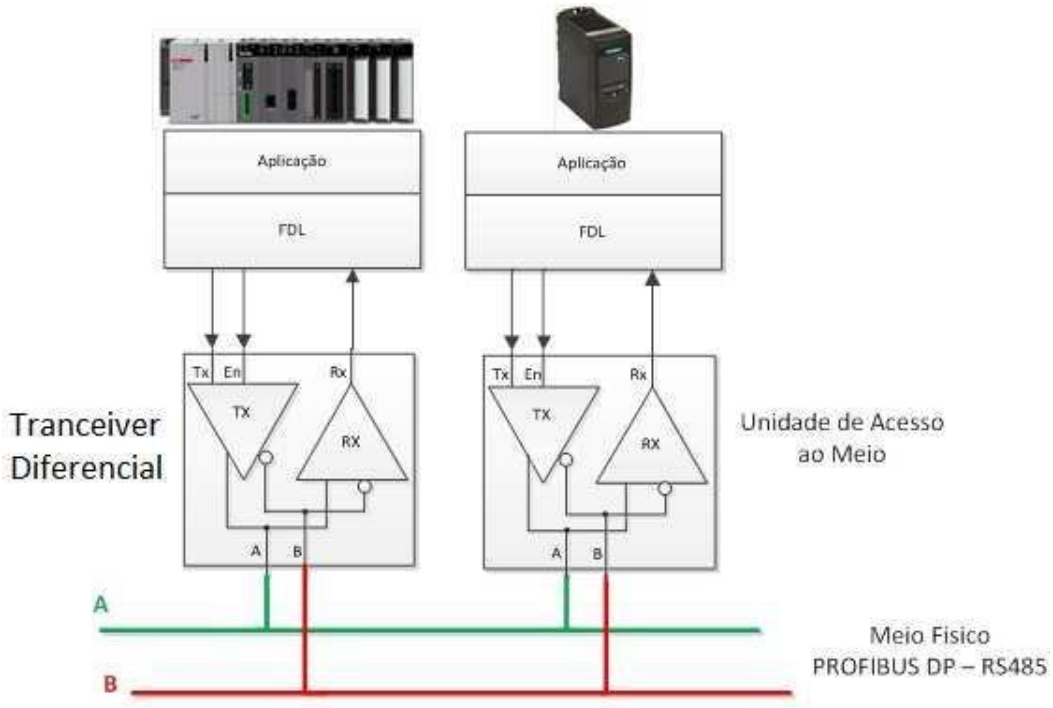

Figura 2.4: Detalhe do Transceiver que permite o acesso ao meio físico.

Na Seção 2.1 deste trabalho há uma descrição detalhada da camada de enlace e do funcionamento da mesma do ponto de vista do mestre e do escravo. Nesta seção também está explicado o funcionamento da maquina de estados da camada de enlace. Desta maneira, este documento pode ser usado como uma introdução a respeito de como desenvolver um escravo e um mestre Profibus DP, pois do ponto de vista de software (algoritmo), foi necessário fazê-lo para implementar o simulador.

Como há a necessidade de uma temporização rígida para evitar conflitos na comunicação e permitir o gerenciamento da rede por parte dos mestres, na Seção 2.2 são definidos todos os *timers* e temporizações das estações que são exigidas pela norma para garantir o funcionamento do protocolo. Tais temporizações foram incluídas no simulador com o objetivo de reproduzir a

temporização de uma rede real.

Na Seção 2.3 apresenta-se os requisitos de uma configuração Profibus necessária para efetuar operação da rede. E na Seção 2.4 é mostrado o aspecto do sinal digital em ambientes com problemas de instalação.

2.1 Camada de Enlace - Estrutura dos Frames e Protocolo

Neste capítulo é detalhada a camada de enlace e os tempos que devem ser calculados para garantir a exatidão do *timestamp* segundo as especificações da norma. Estes conceitos foram aplicados neste trabalho para garantir que a simulação da camada de enlace fosse o mais próximo possível da realidade.

A rede Profibus DP é uma rede do tipo *multidrop*, assíncrona, *half duplex* e utiliza a comunicação do tipo passagem de *token (Token Passing)* e mestre-escravo. O mecanismo de passagem de *token* permite a aplicação de múltiplos mestres em uma mesma rede compartilhando o acesso (domínio da rede).

Somente o mestre pode iniciar a comunicação na rede. Os escravos comunicam somente para responder requisições do mestre. A rede Profibus DP permite a operação permanente com mais de um mestre, desde que configurados individualmente e de maneira adequada nas restrições da norma (PROFIBUS, 1998). Quando não há diferenciação de serviços ou características, mestres e escravos são tratados com o nome de estações.

O endereço de uma estação é atribuído fisicamente diretamente no equipamento, através de uma IHM de configuração ou chaves seletoras numéricas. O protocolo prevê um serviço (comando) opcional que permite realizar a troca de endereço remotamente, através da rede.

O endereço configurado deve ser único na rede (não é permitido 2 ou mais estações com mesmo endereço), conforme a Figura 2.5, isso é um requisito de projeto e instalação.

Assim, a faixa de endereços disponível para uso vai de 0 a 126, permitindo que a rede comporte um número máximo de 127 estações. O endereço 127 é reservado para comandos de *broadcast* e não pode ser atribuído a uma estação.

O endereço 126 é utilizado como endereço temporário, geralmente utilizado para fazer manutenção em um equipamento na rede. Este deve ser trocado depois de efetuada a manutenção. Alguns equipamentos utilizam o endereço 126 como padrão após uma reinicialização de fábrica.

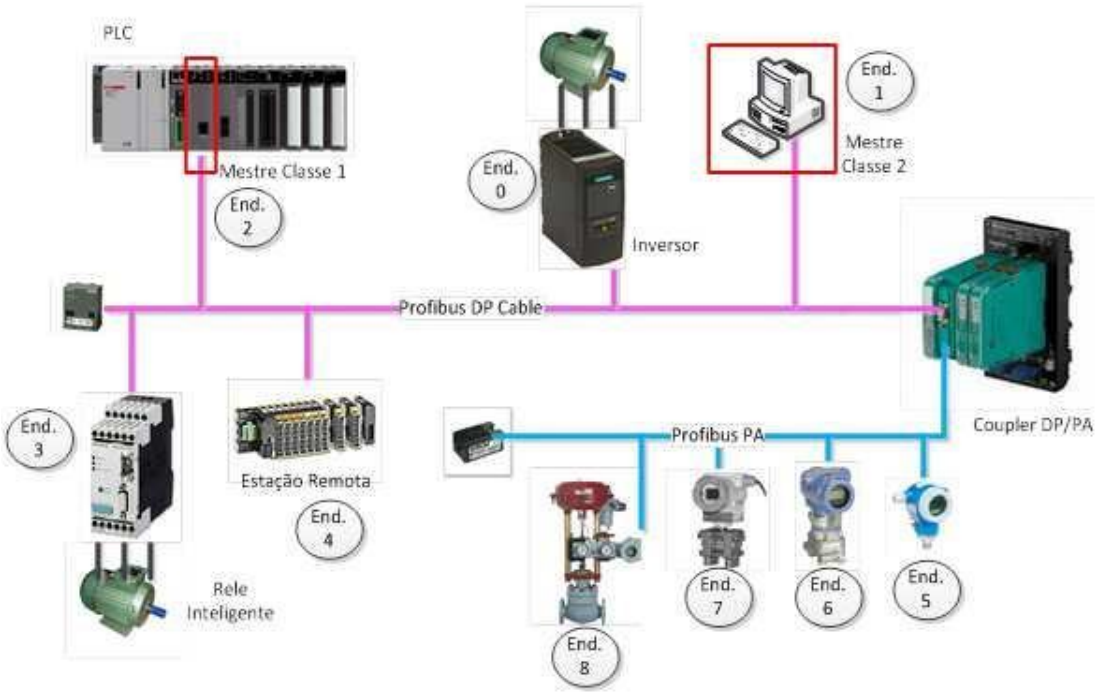

Figura 2.5: Mestres e escravos compartilham mesmo conjunto de endereços.

2.1.1 Estrutura dos frames

Neste trabalho, utiliza-se a palavra mensagem, telegrama, datagrama e frame como sinônimos. Embora estes termos tenham diferentes significados em comunicação TCP/IP, a especificação Profibus utiliza o nome frame para as transferências de dados de comunicação entre estações.

Esta seção apresenta inicialmente a estrutura dos frames, antes de explicar a dinâmica do protocolo. Embora alguns termos tratados na estrutura dos frames sejam desconhecidos, isso já permite ilustrar como ocorre a comunicação entre as estações e a codificação dos dados produzidos pela camada de enlace.

2.1.2 Estrutura de cada caractere do frame

Cada frame é formado por um conjunto de caracteres. Porém cada caractere não é formado somente por 8 bits. No caso da UART, cada caractere é formado por 11 bits. Os 3 bits a mais são utilizados para fornecer uma sinalização de inicio e fim de transmissão de cada caractere (2 bits) e um bit de paridade par usado para checagem da integridade da comunicação no receptor. A Figura 2.6 apresenta a estrutura do caractere de comunicação que compõe a unidade dos

frames.

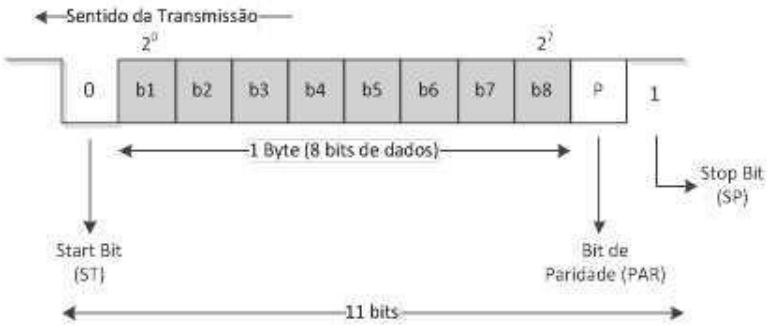

Figura 2.6: Estrutura do caractere da comunicação UART. Cada retângulo representa 1 bit.

O estado de IDLE da comunicação (linha desocupada) é representado pelo nível lógico 1, assim para que o hardware do receptor detecte o inicio de chegada do sinal é necessária a existência de um Start Bit (ST) que é a transição do nível 1 para 0. Um receptor UART ideal deve amostrar o valor no centro do bit, conforme a 2.7 e apresentar baixa variação do período de amostragem.

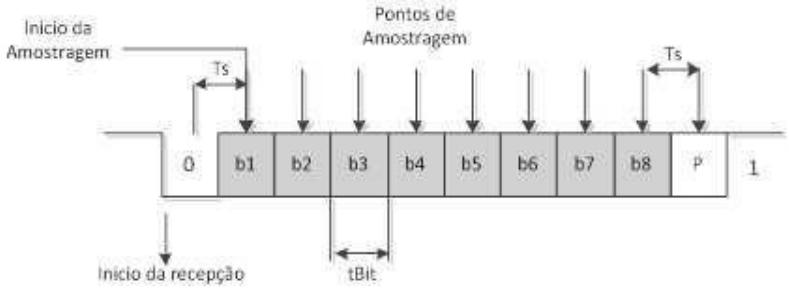

Figura 2.7: Amostragem dos bits realizados pelo hardware da UART.

A amostragem, por parte do receptor, deve começar após 1/2 tBIT a partir do inicio da recepção e o período de amostragem (Ts) deve ser igual ao tBIT para permitir a amostragem no centro do bit, onde a influência de problemas que possam prejudicar o sinal é minimizada.

O tempo consumido pela transmissão de cada bit (tBIT) é função da taxa de transmissão, conforme observado na equação 2.1.

$$Ts = tBit = 1/(TaxaTransmissao) \tag{2.1}$$

Após a amostragem a paridade deve ser avaliada, no caso do Profibus DP, a paridade par

é usada. Portanto, no caso de uma quantidade par de bits de valor lógico 1, o bit de paridade deve vir com o valor lógico 1. Caso contrário, o bit de paridade deve vir com o valor lógico 0.

O receptor avalia a paridade a cada byte recebido. Caso a paridade avaliada não tenha o mesmo valor que o bit de paridade, o frame inteiro será descartado (não somente o caractere em questão).

Um frame é constituído por um ou mais caracteres. Para frames com 2 ou mais caracteres, não é permitido períodos de linha desocupada entre caracteres (IDLE). Um frame com mais de um caractere é exemplificado na Figura 2.8.

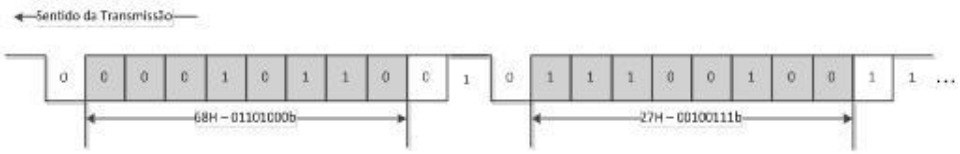

Figura 2.8: Frame contendo os bytes 68H e 27H em sequência.

2.1.3 Tipos e formato dos frames

Um frame é composto por 1 a 255 caracteres. Porém há 5 tipos de frames definidos em norma. A diferenciação entre os tipos é realizada pelo cabeçalho do frame (primeiro caractere), onde cada um dos tipos possui um valor diferente.

Os frames são especificados de acordo com a natureza do campo de dados, os formatos são os seguintes:

1. Frames de tamanho fixo sem campo de dados.
2. Frame de resposta curta ou reconhecimento.
3. Frames com campo de dados de tamanho variável.
4. Frame de *token*.

A Figura 2.9 a seguir apresenta um resumo dos tipos de frames. Conforme descrito anteriormente, cada tipo de frame é identificado pelo primeiro caractere. As funções dos tipos de frame tem importância primordial na camada de enlace.

Os frames do tipo 1 são usados pelo mestre para pesquisar (*polling*) novas estações que possam estar pressentes na rede (mestres ou escravos).

O frame do tipo 2 (SC) é usado quando uma resposta não pode ser fornecida imediatamente ou quando não há resposta para um determinado comando. É uma indicação que a estação de destino recebeu o frame.

Para comunicações entre as camadas de aplicação, o frame utilizado é sempre tipo 3 (SD2). Os serviços e dados são encapsulados no DATA_UNIT e analisados de maneira específica na própria camada de aplicação. O uso deste tipo para a aplicação faz dele o mais utilizado e foco em descrições posteriores.

Os frames do tipo 4 são produzidos apenas por estações do tipo mestre e informam a passagem de *token*, isto é, a transmissão de uso do barramento de um mestre a outro.

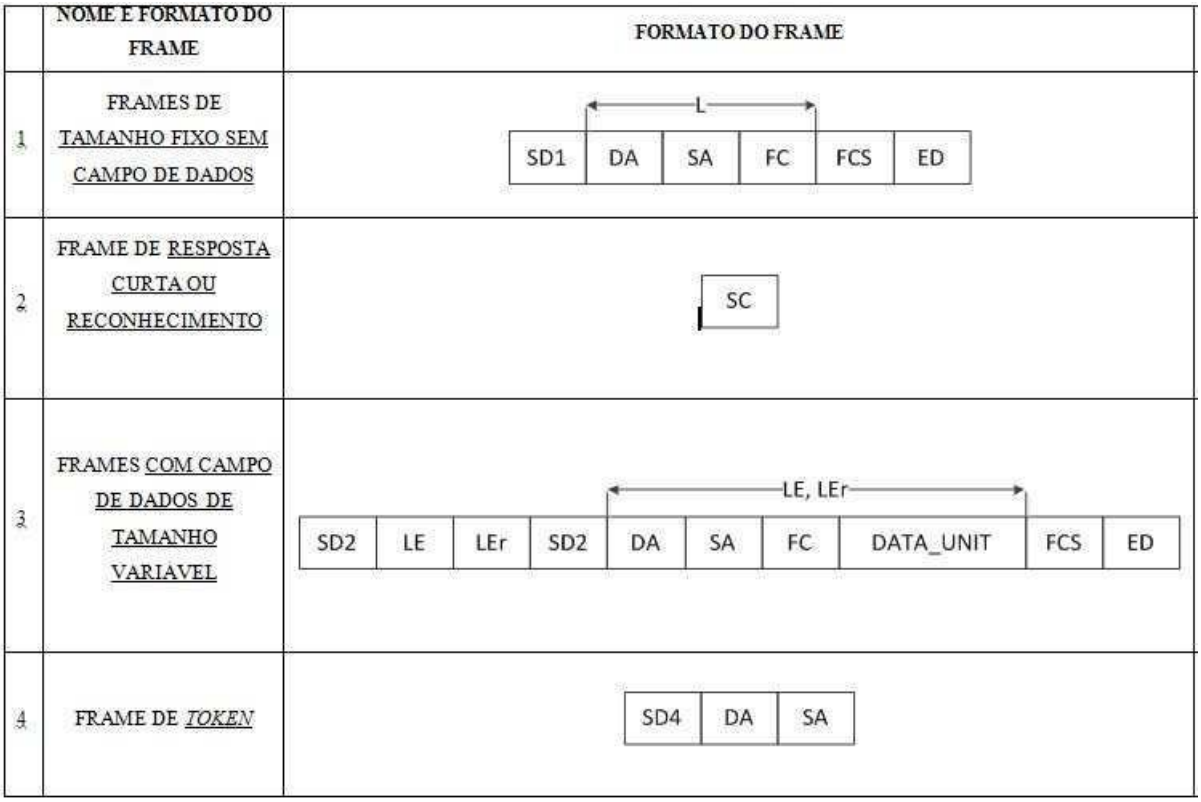

Figura 2.9: Tipos de Frames PROFIBUS DP.

2.1.4 Tamanho do frame e Campo de dados

O tamanho máximo de um frame são 255 caracteres. O valor de LE (ou LEr) varia de 5 a 249. O LE compreende a quantidade de bytes do campo DATA_UNIT além do DA, SA e o FC.

Portanto o tamanho máximo do campo DATA_UNIT é de 246 bytes.

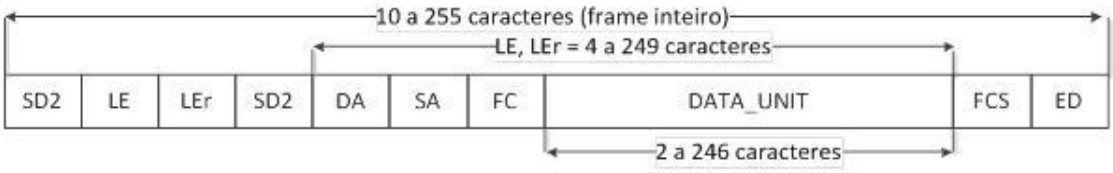

Figura 2.10: Faixas possíveis de tamanho de campos e frame tipo SD2.

O aproveitamento do frame transmitido é mensurado através da porcentagem de utilização do DATA_UNIT (porção do frame destinada à carga útil de dados) e o tamanho total do frame.

Assim, para o caso onde o DATA_UNIT tem tamanho unitário o aproveitamento é de 10 por cento e cresce proporcionalmente ao DATA_UNIT sendo no máximo 96,5 por cento conforme a Figura 2.11 quando são utilizados todos os 246 bytes.

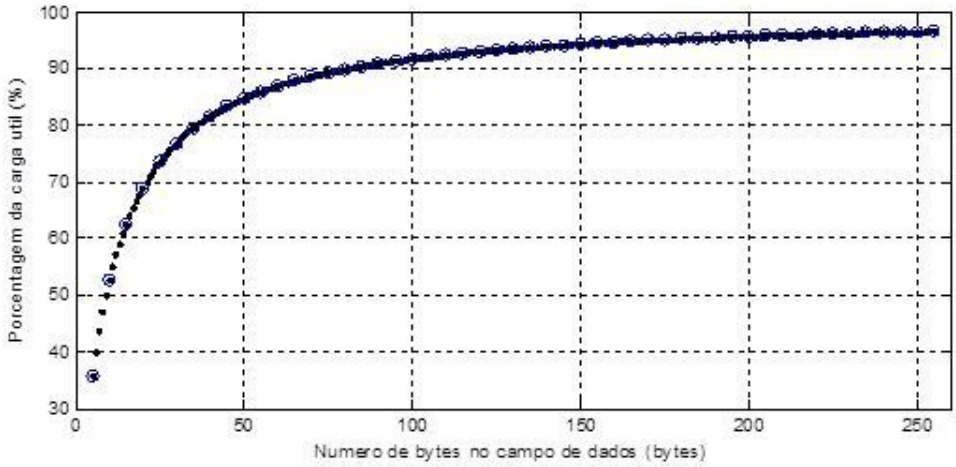

Figura 2.11: Aproveitamento do frame segundo a utilização do DATA_UNIT.

2.1.5 Campos de Endereço

Embora os campos de endereço suportem valores entre 0 e 255 (1 byte), somente os 7 bits menos significativos são utilizados efetivamente para o endereçamento das estações, conforme destacado na Figura 2.12, permitindo o endereçamento de até 127 estações (0 a 126), o endereço 127 conforme explicado anteriormente é reservado ao broadcast.

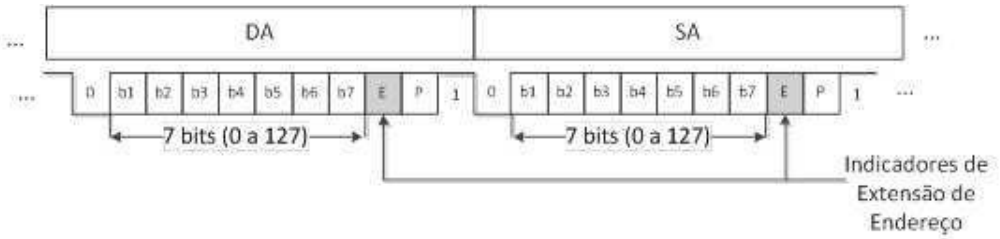

Figura 2.12: Campos de endereço e uso do bit mais significativo como extensão.

Em um frame de resposta os endereços DA (*Destination Address*) e SA (*Source Address*) são simplesmente invertidos em relação ao frame de requisição.

Quando o tipo de frame SD2 é utilizado, o bit mais significativo do byte de endereço é utilizado para sinalizar que o frame que está sendo enviado possui uma extensão de endereço. Isso indica que no campo DATA_UNIT os dois primeiros bytes estarão ainda relacionados com o protocolo, conforme descrito na Figura 2.13.

Essa extensão de endereço é usada pela camada de aplicação para implementar diferentes serviços de parametrização e configuração. Os códigos dos serviços são chamados SAPs (*Service Access Points*) apresentados na Figura 2.13. DAE (*Destination Address Extension*) é a extensão do endereço de destino ou SAP de destino e SAE (*Source Address Extension*) é a extensão do endereço de origem ou SAP de origem.

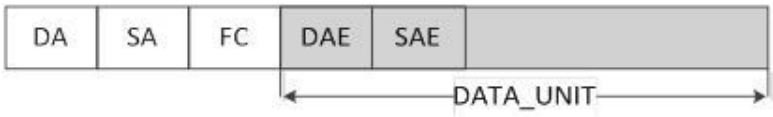

Figura 2.13: Uso dos primeiros bytes do DATA_UNIT para indicação do SAP.

Quando uma estação entra no estado de troca de dados, que é o objetivo final do processo de configuração, não há o uso das extensões de endereço, destinando a totalidade do DATA_UNIT para os dados específicos dos equipamentos (dados de aplicação). As extensões aplicadas ao Profibus DP (DPV0) são apresentadas na Tabela 2.1 a seguir:

2.1.6 Campo de controle do Frame

Este campo é usado para indicar o tipo de frame (requisição ou resposta), controle de sequencia que evita a perda de mensagens e contém o tipo da estação e o status da FDL desta

Tabela 2.1: Serviços utilizados pela camada de aplicação.

Service (-)	Master SAP hex (dec)	Slave SAP hex (dec)
Data Exchange	não	não
Set Slave Addr	3E (62)	37 (55)
Read Inputs	3E (62)	38 (56)
Read Outputs	3E (62)	39 (57)
Global Control	3E (62)	3A (58)
Get Configuration	3E (62)	3B (59)
Slave Diagnostics	3E (62)	3C (60)
Set Parameters	3E (62)	3D (61)
Check Configuration	3E (62)	3E (62)

estação (escravo ou mestre). O significado dos bits do campo FC é apresentado a seguir na Figura 2.14:

Figura 2.14: Descrição dos bits do campo FC (Frame Control)

Caso o b7 seja 1 então o frame é uma requisição. Neste caso os bits b6 e b5 são chamados FCB e FCV e são usados para controle de sequenciamento (veja o item a seguir). Caso o b7 seja 0 então se trata de uma resposta. O b6 e b5 identificam o tipo (mestre ou escravo) e estado da estação que está respondendo. O código da função indica o tipo de requisição ou resposta (como requisição com confirmação ou sem confirmação e resposta positiva ou negativa).

2.1.7 Controle de sequencia

O bit FCB evita recepção de mensagens duplicadas e reenvio da última mensagem quando for detectado que o escravo não respondeu a última requisição.

Após um escravo de endereço específico se tornar operacional, na primeira mensagem o mestre envia o FCB=1 e FCV=0. O escravo guarda o valor do FCB=1 e marca essa mensagem como a primeira recebida do mestre.

Após receber a resposta deste escravo, o mestre sempre enviará FCV=1 para este escravo (sinalizando que o mestre iniciará a indicação do controle de sequência) e ficará alternando o valor do FCB entre 0 e 1 (começando com FCB=0). Deste modo, caso o mestre não receba alguma resposta desta estação, ele enviará novamente a última requisição sem alternar o valor do FCB, indicando ao escravo que foi um pedido de *retry*, isto é, o mestre não recebeu a última resposta deste equipamento.

O mestre deve implementar este controle para cada um dos endereços dos escravos que estiverem em sua configuração. Este controle está presente apenas nos frames que estão relacionados com a camada de aplicação. No frame *Request FDL Status* esse mecanismo não é usado.

2.1.8 Cálculo do FCS

O FCS (Frame Check Sequence) é usado para verificar a integridade da comunicação e permitir ao receptor descartar frames inválidos ou corrompidos evitando a chegada destas informações até a camada de aplicação.

O cálculo do FCS é a soma aritmética do valor dos bytes contidos no DATA_UNIT e dos valores dos bytes do DA, SA e FC (equação 2.2). No caso dos frames sem campo de dados, somente o DA, SA e FC são somados. Caso o valor ultrapasse 255 (0xFF), ignora-se os bytes mais significativos e utiliza-se apenas o byte menos significativo da soma.

$$FCS = DA + SA + FC + DATA\_UNIT \tag{2.2}$$

No caso do Profibus PA, não há o end delimiter (veja Tabela 1) e o FCS é substituído por um CRC16 aplicado a qualquer frame. Para as situações onde a segurança e integridade proporcionada pelo FCS forem insuficientes, mesmo no Profibus DP, existe um *Profile* (*Profisafe*), que faz uso de CRC32 encapsulado dentro da camada de dados.

2.1.9 Procedimentos de transmissão e o controlador da camada de enlace

Um ciclo de comunicação de um mestre com um conjunto de escravos é chamado ciclo de mensagens. Este ciclo é interrompido somente para a transmissão de *token* e mensagens de *broadcast*. Todas as estações devem monitorar todas as requisições. Uma estação deve responder somente quando endereçada. A resposta deve ocorrer dentro de um tempo pré-definido (*SlotTime*). Limite de tempo que o mestre aguarda a resposta, antes de tentar novamente a requisição para a mesma estação.

Uma estação que deixa de responder a uma requisição, mesmo após sucessivas tentativas do mestre é marcada como não operacional. O controlador da FDL, que faz a manutenção do status dos endereços presentes na rede opera de acordo com uma maquina de estado definida na norma.

Há quatro tipos de modo operação que ocorrem, estas definem o comportamento em relação a temporização e prioridade dentro de um ciclo de mensagens. As operações são as seguintes:

1. Recepção e envio do Token.

2. Comunicação acíclica.

3. Comunicação cíclica ou polling.

4. Registro das estações

2.1.10 Recepção e envio do Token

O token é um tipo de mensagem que é transmitida entre os mestres, e é usado com o objetivo de um mestre transferir o acesso ao meio para outro mestre. Na Figura 2.15 observa-se uma ilustração do processo de operação das estações em uma rede Profibus.

Cada mestre acessa o canal de comunicação no modo mestre-escravo. Ao fim do acesso, transmite uma mensagem de *token* ao próximo mestre. Então, o mestre comanda o canal e posteriormente passa o *token* ao próximo mestre. Quando o ultimo mestre termina de realizar os comandos necessários aos escravos e aos outros mestres, ele passa o *token* novamente ao primeiro mestre e o processo reinicia. A esse anel lógico dá-se o nome de *logical token ring*.

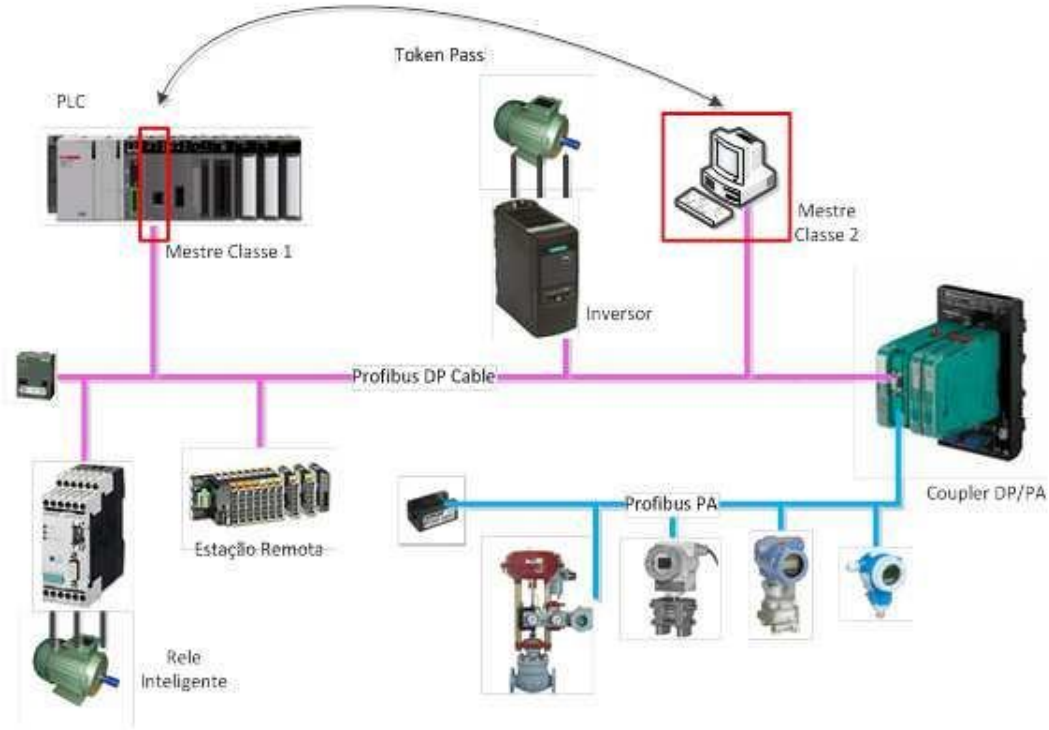

Figura 2.15: Transferência do controle de acesso ao meio entre os mestres.

Uma rede que tem somente um único mestre resume sua operação na comunicação tipo mestre-escravo e ao fim de cada ciclo o mestre passa o *token* para ele mesmo.

A manutenção da rede é de responsabilidade de cada mestre. Assim, cada mestre deve ter o conhecimento de todos os equipamentos (endereços) que estão presentes na rede e qual sua natureza (mestre ou escravo). Após um conjunto de ciclos de comunicação, o mestre consulta um novo endereço através de um comando dedicado a manutenção da rede. O mestre aguarda o retorno da resposta do endereço consultado: se houve resposta então o mestre adiciona este endereço a uma lista de estações presentes. Se não houve resposta, considera-se que esse endereço é vago. Um equipamento presente na rede, que responde a comandos de algum mestre, é denominado operacional.

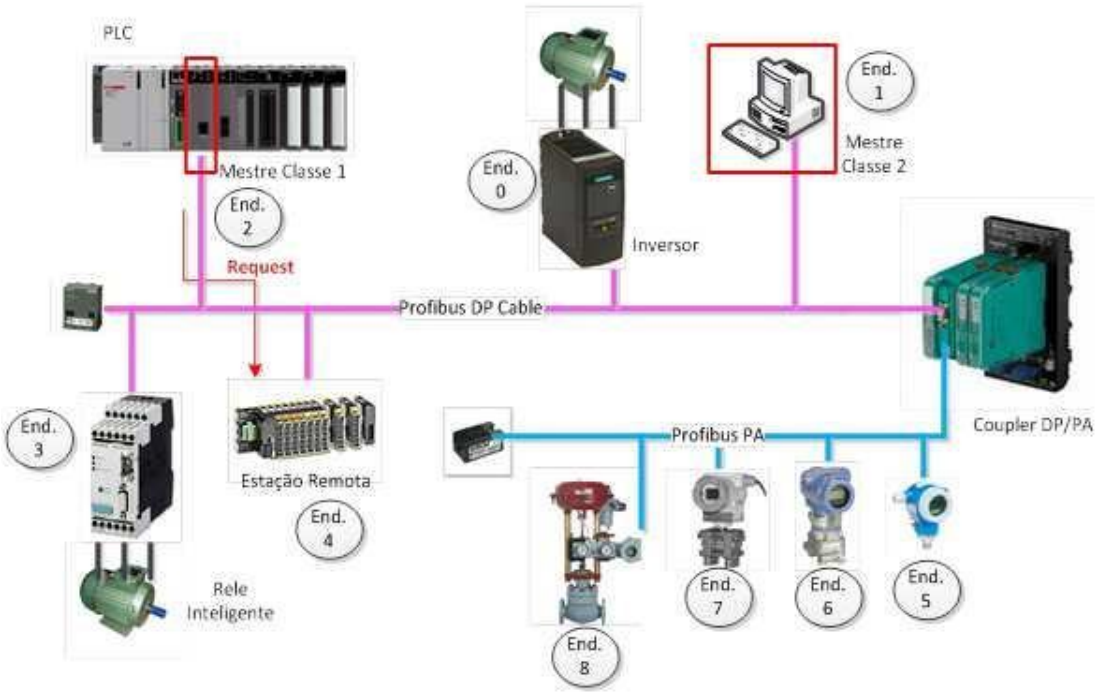

Figura 2.16: Cada mestre possui sua lista de endereços ativos e passivos (operacionais).

2.1.11 Inicialização da rede

Durante o processo de energização da rede e estações, não há nenhuma mensagem presente na linha, assim há a necessidade de um protocolo de partida.

Ainda que todos os escravos estejam energizados e prontos para comunicação, caso não exista nenhum mestre energizado e conectado a rede, o tráfego será nulo - pois somente os mestres podem iniciar a comunicação.

Após a energização e inicialização da camada de comunicação, todos os mestres aguardam um tempo (TTO) proporcional ao seu endereço por atividade na linha (presença de outro mestre). Até que o mestre de menor endereço envie 2 comandos de *token* seguidos para ele mesmo e inicia a comunicação normalmente. Isso sinaliza aos outros mestres que há operação na rede e estes aguardarão a recepção do *token* para que também possam operar seus ciclos de comunicação. O cálculo deste período de tempo, assim como a máquina de estados que regula estes comportamentos acima descritos serão apresentados posteriormente.

Este mecanismo que diferencia o tempo de partida baseado no endereço evita que haja uma colisão contínua que possa impedir a inicialização da rede.

2.1.12 Comunicação acíclica e cíclica

A lista de equipamentos da configuração, isto é, que pertencem ao modo de comunicação cíclica é passada ao controlador FDL pela camada de aplicação. As estações que não responderem durante a comunicação cíclica são classificadas como não operacionais.

Cada mestre mantém sua própria lista (configuração) de equipamentos pertencentes a comunicação cíclica. Ao fim da comunicação cíclica, prioritária, é realizada a comunicação acíclica e pelo menos um endereço não operacional pode ser consultado, veja o diagrama na Figura 2.17.

A comunicação cíclica é baseada em respostas imediatas dos escravos e mestres. Ela é prioritária em relação às mensagens acíclicas. São usadas para transmissão de dados de controle do processo: *setpoint*, pressão, temperatura, fins de curso - variáveis que entrarão na malha de controle e cuja taxa de amostragem interfere no desempenho do sistema.

As mensagens acíclicas são baseadas em consulta, isto é, o mestre inicia a requisição e no próximo ciclo pergunta novamente para verificar se a estação já tem a resposta. Em geral, são usadas para supervisão ou parametrização de dados internos das estações, cuja variação do valor é limitada, por exemplo: escalas de entrada e saída, configuração de alarmes, tipos de sensores para equipamentos configuráveis, configuração de display - variáveis relacionadas à configuração dos equipamentos e instrumentos.

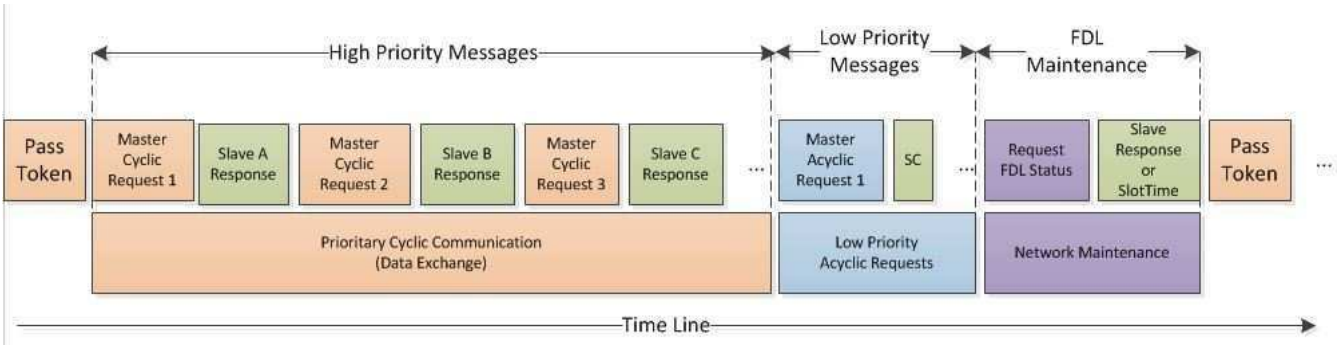

Figura 2.17: Diagrama mostrando distribuição de mensagens cíclicas e acíclicas

Mensagens cíclicas e acíclicas, além da manutenção da rede, devem ocorrer dentro de um período configurado pela aplicação (TTR).

2.1.13 Registro das estações

Como explicado anteriormente, o registro das estações é de responsabilidade de cada mestre da rede. O registro é realizado através do comando *Request FDL Status* que é um pacote do tipo SD1, destinado a manutenção da rede.

Dependendo da resposta, a estação se identifica como escravo ou mestre, verificando o status desta estação a fim de incluí-la no *logical token ring*.

2.1.14 Maquina de estado do controlador da FDL

A Figura 2.18 mostra em qual camada a maquina de estados do controlador da FDL é implementada. A maquina de estados é composta por 10 estados, sendo que estações mestre possuem os estados 0 a 9 e estações do tipo escravo apenas os estados 0 e 10.

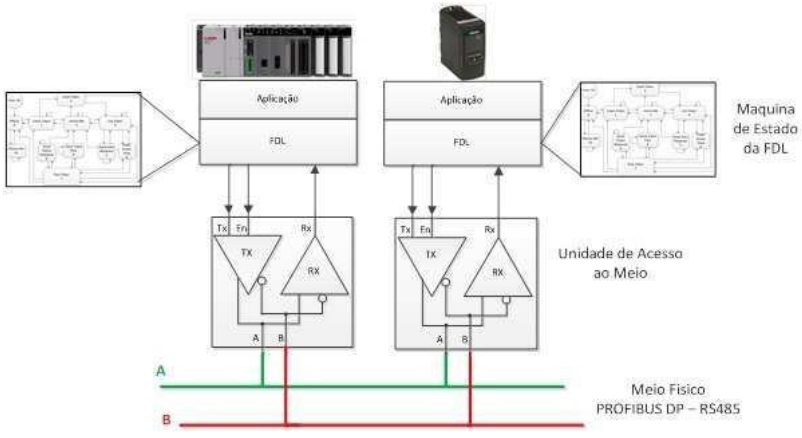

Figura 2.18: Camada de software que implementa a máquina de estado da FDL.

O diagrama da maquina de estados é apresentado na Figura 2.19.

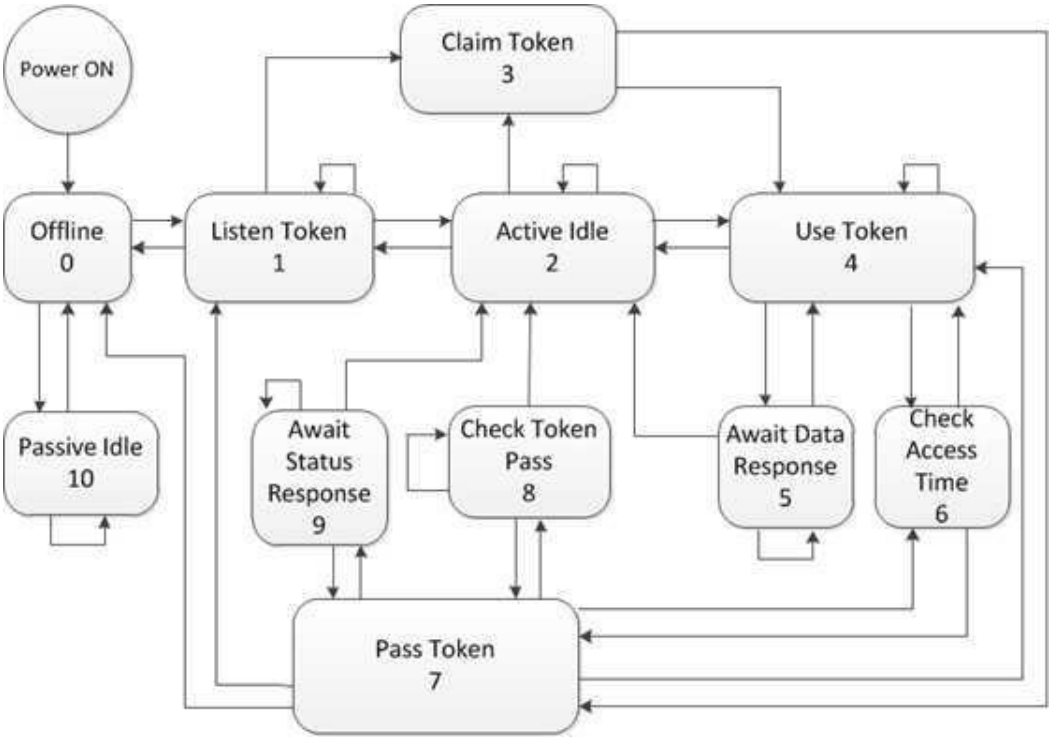

Figura 2.19: Maquina de estados da FDL.

- Offline (0) - É o estado que a estação se encontra logo após a partida, reset do controlador da FDL (comandado pela camada de aplicação) ou no caso de alguma condição de erro detectada. Após a energização, cada estação deve realizar um auto teste que varia de acordo com a natureza específica do equipamento. Caso apresente algum problema, o equipamento permanecerá neste estado. Após todos os parâmetros de operação carregados, o controlador da FDL vai para o estado *Passive_Idle* (caso a estação seja um escravo) ou para o estado *Listen_Token* (caso seja um mestre).

- Passive_Idle (10) - Após inicialização dos parâmetros uma estação escrava deve entrar neste estado e aguardar comandos para seu endereço. Ela deve responder quaisquer comandos para seu endereço, menos comandos de broadcast e frames do tipo *token*. No caso de detecção de alguma falha ou condição de erro ou ainda um comando de reset do FDL o controlador FDL deve retornar para o estado *Offline*.

- Listen_Token (1) - No caso do mestre, após a inicialização dos parâmetros internos, a estação deve entrar neste estado caso ela esteja pronta para entrar no *logical token ring*, isto é, quando ela estiver apta a receber *token*, transmitir, comandar a rede e enviar o *token* a próxima estação mestre conhecida.

É neste estado que o controlador da FDL da estação mestre deve monitorar a linha com o objetivo de identificar quais estações mestre já estão no *logical token ring*. Para que isso ocorra a estação neste estado deve analisar os frames de *token* e construir uma lista de estações ativas (*LAS - List of Active Stations*). Durante a construção da LAS a estação não deve responder quaisquer frames endereçados a ela.

Após verificar duas rotações completas de token, isto é, um mesmo comando de passagem de *token* (mesmo endereço de origem e destino), a FDL se mantém neste estado até que receba um comando *Request FDL Status* do mestre com endereço inferior mais próximo. Então esta estação deve responder dizendo que está pronta para entrar no *Logical token ring* (*ready to enter logical token ring*). Assim, ela receberá o próximo frame de *token* e deverá entrar no estado *Active_Idle*.

Caso a estação detecte dois frames de *token* que tenham endereço de origem igual ao seu, ela deve assumir que há outro mestre com o mesmo endereço e ir para o estado de *Offline*. Se a FDL não observar atividade na linha pelo período de TTO, ela deve assumir que a inicialização ou restauração do *Logical token ring* é necessária. Então a FDL deve ir para o estado *Claim_Token* e reinicializar o *token ring*.

- Active_Idle (2) - Neste estado a estação deve receber os frames da linha sem se tornar uma estação ativa. Ela deve responder aos frames endereçados a ela. Somente quando receber o *token*, a FDL deve ir para o estado *Use_Token*. Caso ela receba dois frames seguidos de token, pode significar que o mestre anterior não detectou a presença desta estação na linha, neste caso ela deve retornar ao estado de *Listen_Token*. Caso a FDL verifique que a estação foi retirada do *logical token ring*, ela também deve ir para o estado *Listen_Token*.

  Se a FDL não observar atividade na linha pelo período de TTO, ela deve assumir que a inicialização ou restauração do *Logical token ring* é necessária. Então a FDL deve ir para o estado *Claim_Token* e reinicializar o *token ring*.

- Claim_Token (3) - Quando a estação estiver nos estados *Active_Idle* ou *Listen_Token* e o *time-out* TTO ocorrer, então considera-se que a linha está sem atividade e a FDL entra neste estado. Ela deve entrar no estado *Use_Token* imediatamente. Após inicialização, o primeiro *token* deve ser enviado com endereço de destino igual ao endereço de origem. Isso tem o objetivo de incluir esta estação na LAS de outros mestres.

- Use_Token (4) - A FDL entra neste estado após receber um token ou após a inicialização.

Neste estado que a estação mestre em questão acessa o canal e estabelece a comunicação cíclica e acíclica com os escravos.

Ao entrar neste estado, o TRR (*Real Rotation Time*) deve ser subtraído do TTR (*Token Rotation Timer*) e o timer deve ser reinicializado. Pelo menos um ciclo de mensagens cíclicas deve ser realizado. Envio de mensagens acíclicas somente devem ser realizados caso o TRR < TTR durante este processamento.

Após cada comando de requisição do mestre, a FDL deve entrar no estado *Await_Data_Response* e inicializar o *Slot Timer*.

O controlador da FDL deve entrar no estado *Check_Access_Time*, caso no inicio do estado *Use_Token* não seja necessária a realização de um conjunto de mensagens cíclicas ou após completar um conjunto de mensagens cíclicas ou acíclicas.

- Await_Data_Response (5) - A cada requisição realizada pelo mestre, a FDL deve entrar neste estado onde a ela aguarda a resposta por um tempo configurado pelo usuário (*Slot Time*). No caso de um serviço sem resposta, então deve retornar ao estado *Use_Token* para processar demais requisições. Caso o serviço exija resposta, o que é mais comum, deve-se esperar por um dos seguintes eventos: a) Uma resposta válida endereçada ao mestre que originou a requisição. b) Qualquer outro frame válido (token ou requisição). c) Um frame inválido ou sem resposta (Slot Time expirou). Após recepção e processamento da resposta, é necessário retornar ao estado de Use_Token para processar demais requisições. A recepção de outro frame válido (caso b) indica que ocorreu um erro de consistência grave da rede. A FDL deve entrar no estado *Active_Idle* e descartar o frame recebido. Se um frame inválido for recebido (caso c) ou se expirar o *Slot Time*, deve tentar novamente a última transmissão. Caso persista a falha a FDL deve retornar ao estado de *Use_Token* e não repetir novamente a requisição para a ultima estação até que um ciclo de mensagens completo sem falhas seja realizado.

- Check_Access_Time (6) - Neste estado o THT (*Token Holding Time*) deve ser computado pela diferença entre o TTR e TRR (THT = |TTR - TRR|). Apenas se houver THT disponível a FDL deve retornar ao estado *Use_Token*. Caso contrário, deve entrar no estado *Pass_Token*.

- Pass_Token (7) - No estado de *Pass_Token* a estação deve tentar passar o *token* para a próxima estação no *logical token ring*. Quando transmitir o frame de *token*, o FDL deve

checar sua própria transmissão (para verificar se o transceiver está operando corretamente). Caso não receba seu próprio comando, considera-se que houve um erro e a FDL deve ir para o estado *Offline*.

Caso ela receba o próprio frame, porém corrompido, não deve parar a operação e deve ir para o estado *Check_Token_Pass*. Caso retorne para este estado, o frame de *token* deve ser retransmitido e a FDL ir novamente ao estado *Check_Token_Pass*. Caso retorne novamente para este estado, a FDL deve parar a atividade na linha e ir para o estado de *Listen_Token*.

Quando o *GAP Update Time* expirar, mas ainda houver TTH disponível, antes de passar o *token*, a FDL deve tentar testar um novo endereço com o objetivo de incluí-lo no *logical token ring*, se necessário. Com este propósito, a FDL transmite um *Request FDL Status* e entra no estado *Await_Status_Response*. Se uma nova estação mestre responde que deseja ser incluída no *logical token ring*, a FDL deve passar o *token* para esta estação. Caso não seja detectada nenhuma outra estação mestre, então o FDL deve passar o *token* para si mesmo e ir para o estado *Use_Token*.

- Check_Token_Pass (8) - Neste estado a FDL aguarda um *Slot Time* pela reação da estação que estiver recebendo o *token* (endereço de destino do frame de *token*). Este tempo permite que o próximo mestre receba o *token* e inicie uma transmissão.

  Se a FDL detectar um início de frame válido dentro de um *Slot Time*, ele assume que a passagem do token ocorreu com sucesso, então a FDL vai para o estado de *Active_Idle*. Se um frame inválido for detectado, a FDL assume que outra estação está ativa e também vai para o estado de *Active_Idle*. Caso nenhum frame seja detectado dentro deste período, FDL retorna ao estado *Pass_Token*.

- Await_Status_Response (9) - Neste estado a FDL deve aguardar por um *Slot Time* por um frame de resposta relacionado a identificar a natureza (mestre ou escravo) e o estado da estação em pesquisa. Caso receba uma resposta válida ou não receba nenhuma resposta, então ele deve retornar ao estado Pass_Token. Caso receba uma resposta inválida (pode indicar a existência de múltiplos *tokens*), ele deve entrar no estado *Active_Idle*.

2.2 Taxa de Comunicação e Temporização entre estações

Esta seção trata da temporização normatizada que deve ser respeitada por cada estação entre um novo envio e resposta entre frames. O estudo realizado nesta seção permitirá simular a

temporização, reproduzindo a sequencia de frames de uma rede real com as estampas de tempo (*timestamp*) reais. No simulador cada um destes tempos é calculado com o objetivo de calcular o *timestamp* de cada mensagem gerada. Assim todas as formulas desta seção farão parte do simulador.

A taxa de comunicação de uma rede Profibus DP é geralmente atribuída pelo mestre. Em equipamentos mais antigos há um conjunto de chaves ou configuração através da interface do equipamento para seleção da taxa de comunicação em cada estação, atualmente os escravos são dotados de circuitos integrados dedicados que detectam automaticamente a taxa de comunicação. Em uma rede com múltiplos mestres, cada mestre deve ser configurado separadamente, mas todos com a mesma taxa de comunicação.

A taxa de comunicação é selecionada pelo usuário de acordo com os requisitos de projeto e limitações físicas da instalação. A Tabela 2.2 apresenta os máximos comprimentos de cabo permitidos pela norma em relação às taxas de comunicação (*Baud Rate*).

Tabela 2.2: Taxa de Comunicação e Distancia Máxima Permitida

Taxa de Comunicação (bps)	Distancia Máxima Permitida (m)
9600	1000
19200	1000
45450	1000
93750	1000
187500	1000
500000	400
1500000	200
3000000	100
6000000	100
12000000	100

O mestre utiliza temporizadores (*timers*) para verificar e garantir que a resposta dos escravos está sendo realizada dentro do previsto, assim como evitar que ocorra colisão ou transmissão simultânea de mais de uma estação. Estes timers são apresentados a seguir na forma de TBit que permite manter os mesmos valores em função da taxa de comunicação.

2.2.1 Bit Time (TBit)

É o tempo decorrido (s) durante a transmissão de 1 bit. O valor em segundos do TBIT é inversamente proporcional a taxa de comunicação (ou taxa de transmissão) em bits por segundo

(bps).

$$TBit = 1/BaudRate \tag{2.3}$$

2.2.2 Tempo de Sincronismo (TSYN)

É o intervalo mínimo de tempo que cada estação deve verificar o estado de linha desocupada antes de receber um comando de requisição ou *token*. A definição de linha desocupada é o valor binário 1. O valor do tempo de sincronismo (TSYN) é constante e igual a 33TBIT. Este tempo é usado pelo receptor para configurar internamente seu sistema de comunicação de modo que fique pronto para receber um novo frame.

$$TSyn = 33 * TBit \tag{2.4}$$

2.2.3 Tempo de atraso da estação (TSDx)

É o período de tempo que compreende a transmissão ou recepção do último bit até a transmissão ou recepção do primeiro bit. TSDI é o tempo entre a última recepção e o início de uma nova requisição ou *token*. O TSDI é aplicado somente a estações mestre. Min TSDR e Max TSDR são respectivamente os atrasos mínimo e máximo entre todas as estações. É o tempo mínimo e máximo que uma estação leva para responder a uma requisição. O TSDR varia de acordo com o fabricante do escravo e depende da tecnologia utilizada para implementar o sistema de comunicação. Este valor é incluído no arquivo que descreve os escravos (GSD) de modo que isso possa ser levado em conta no momento do projeto da rede de comunicação.

$$MinTSDR = Min(TSDR1, TSDR2..., TSDRn) \tag{2.5}$$

$$MaxTSDR = Max(TSDR1, TSDR2..., TSDRn) \tag{2.6}$$

2.2.4 Tempo de vazio (TQUI)

Este tempo de linha inativa (*Quiet Time*) deve ser considerado em todas as vezes que for utilizado um dispositivo que possa inserir um atraso adicional após transmissão ou recepção dos

frames.

Por exemplo, quando se inclui um repetidor RS-485, um conversor fibra-óptica ou um conversor wireless com o objetivo de estender a rede DP, este equipamento recebe um frame inteiro ou o cabeçalho inteiro do frame antes de iniciar a retransmissão.

Neste período, do ponto de vista do mestre que está realizando a requisição, o canal de comunicação recebe um atraso adicional. Para evitar que o mestre interprete isso como uma queda de comunicação e tome as providências, este tempo é incluído na configuração do mestre para ser considerado.

2.2.5 Margem de segurança (TSM)

Após o tempo de sincronismo, a margem de segurança deve ser respeitada antes de uma nova requisição.

$$TSM = 2 * TBit + 2 * TSET + TQUI \tag{2.7}$$

2.2.6 Idle Time (TID)

É o tempo de espera que o mestre deve respeitar entre a recepção do último bit da última resposta e o início do primeiro bit da próxima requisição na rede. Para este tempo dá-se o nome de TID1. Caso o mestre envie um comando que não admita resposta (um frame de broadcast ou token), então a próxima requisição deve também respeitar um tempo de linha a este se dá o nome de TID2.

$$TID1 = Max(TSYN + TSM, MinTSDR, TSDI) \tag{2.8}$$

$$TID2 = Max(TSYN + TSM, MaxTSDR) \tag{2.9}$$

2.2.7 Atraso da Linha (TTD)

É o tempo máximo de transmissão do meio entre o transmissor e o receptor quando um frame é transmitido. Exemplificando: para uma linha de 200m e taxa em 500Kbps, sem repetidor, o

atraso é de aproximadamente 1us, portanto, TTD = 0,5.TBIT.

2.2.8 Slot Time (TSL)

É o máximo tempo que um equipamento que realiza uma requisição deve esperar pelo primeiro caractere de resposta após o envio do último bit do frame de requisição. O Slot Time, também é o tempo que um mestre aguarda pelo envio do primeiro caractere de um outro mestre após realizar a passagem do *token*. Este tempo é configurado no mestre, e deve ser adequado de maneira a não ser curto demais prejudicando a comunicação por não esperar a resposta dos escravos mais lentos, assim como não inserir um atraso muito grande na rede quando algum escravo parar de responder.

$$TSL1 = 2 * TTD + MaxTSDR + 11 * TBit + TSM \tag{2.10}$$

$$TSL2 = 2 * TTD + MaxTID1 + 11 * TBit + TSM \tag{2.11}$$

Com o objetivo de simplificar a implementação, utiliza-se o TSL de maior valor.

$$TSL = Max(TSL1, TSL2) \tag{2.12}$$

2.2.9 Time-out (TTO)

É o máximo tempo que um mestre aguarda em uma rede em inatividade antes de ir para o estado *Claim_Token* e enviar um *token* na linha. O valor do TTO depende e é proporcional ao valor do endereço do mestre (A).

$$TTO = 6 * TSL + 2 * A * TSL \tag{2.13}$$

O primeiro termo da equação garante que este timer não irá expirar dentro do período de tempo entre a requisição e uma resposta e o segundo termo dificulta a ocorrência do envio de múltiplos *tokens* simultâneos (originados de vários mestres).

2.2.10 Resumo dos tempos

Um resumo dos tempos já apresentados pode ser observado na Figura 2.20, para auxiliar na compreensão:

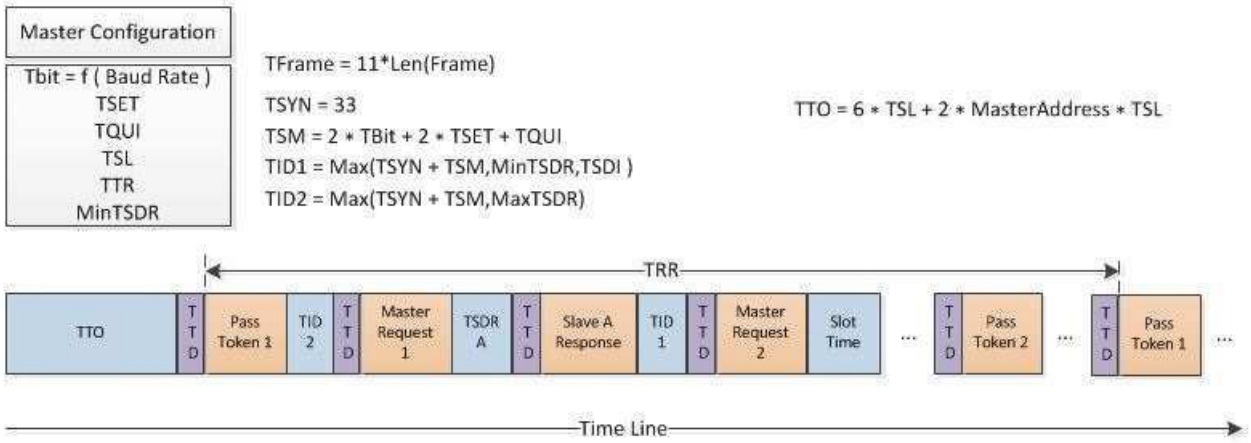

Figura 2.20: Ilustração dos tempos utilizados na comunicação Profibus

2.2.11 Período de atualização do GAP (TGUD)

É o período de tempo que o mestre aguarda para realizar uma nova atualização na live-list, isto é, após a primeira pesquisa do estado de todos os endereços, o mestre realizará uma nova pesquisa a cada período TGUD.

$$TGUD = G * TTR \tag{2.14}$$

O valor de G deve estar entre 1 e 100. O termo G é também chamado de fator GAP (*GAP factor*) e influencia no tempo de percepção do mestre de novas estações que possam ser adicionadas na rede. Somente após este período, o mestre iniciará novamente a consulta de novas estações através de comandos *Request FDL Status* cada vez que estiver com o *token*.

2.2.12 Tempo de ciclo de Mensagem (TMC)

É o tempo compreendido entre o início do envio de uma requisição, o recebimento da resposta e o início de outra requisição. Onde LReq e LRes representam respectivamente o comprimento de todos os bits de comandos de requisicao e dos comandos de resposta.

$$TR = LREQ * 11 * TBit \tag{2.15}$$

$$TAR = LRES * 11 * TBit \tag{2.16}$$

$$TMC = TR + TSDR + TAR + TID + 2 * TTD \tag{2.17}$$

2.2.13 Tempo de Reação do Sistema (TSR)

O Tempo de reação do sistema com um mestre e n estações escravas é obtido através do Tempo de Ciclo de Mensagem e do número de estações escravas. Prevendo a possibilidade de m *retries* por ciclo, o TSR é computado da seguinte maneira:

$$TSR = n * TMC + m * TMC \tag{2.18}$$

Para um sistema onde há várias estações mestre, além de escravos, o TSR é igual ao TTR (*Target Rotation Time*). O tempo de reação do sistema permite redimensionar o projeto de uma rede segundo um requisito de tempo de resposta de controle. Pode-se modificar a taxa de comunicação ou número de escravos por rede para permitir atingir um determinado tempo de reação.

2.2.14 Target Rotation Time (TTR)

O *Target Rotation Time* (TTR) é um valor de tempo configurado pelo usuário com o objetivo de garantir o tempo de reação do sistema desejado e uma sobra de tempo que pode abrigar possíveis *retries* e comunicações acíclicas. O período de tempo que o usuário quer garantir entre a estação receber o *token*, realizar a comunicação cíclica (prioritária), fazer a manutenção da rede (caso o TGUD tenha expirado), permitir tratar *retries*, comunicação acíclica (não prioritária), circular o *token* entre todas as estações ativas e receber o *token* novamente.

2.2.15 Real Rotation Time (TRR)

O período de tempo entre a estação receber duas vezes o token é chamado *Real Rotation Time* (TRR). Quando um mestre recebe o token ele guarda o valor de tempo e após a operação do *Logical token ring*, o mestre recebe novamente o *token* e calcula a diferença de tempo. Este tempo é o TRR. Cada vez que o mestre recebe o *token*, ele realiza pelo menos um ciclo de mensagens prioritárias, após este ciclo, conforme visto anteriormente, ele vai para o estado de *Check_Access_Time* e verifica se TRR > TTR.

2.2.16 Watchdog Time (TWdg)

Após configurados, os escravos que recebem saídas (atuadores, motores com controle de velocidade, válvulas on/off) dependem das atualizações do mestre para seu controle. Na falta (ou falha) do mestre ou queda da rede, estes equipamentos têm que atribuir, por sí próprios, em suas saídas uma posição de segurança, escolhida de acordo com a aplicação.

Prevendo este tipo de situação, o protocolo incorporou o *Watchdog Time*. Este valor de tempo é configurado pelo usuário (em ms), que o escravo deve aguardar por uma atualização do mestre. Este valor é enviado aos escravos durante a configuração dos mesmos.

Caso o escravo não receba uma atualização para o seu endereço, em um período menor que o TWdg, o escravo escreve um valor de segurança na saída e pára a troca de dados. Sendo necessário um novo ciclo de configuração deste escravo.

O TWdg é dimensionado de acordo com o pior caso previsto para o tempo de ciclo (ou tempo de reação do sistema). Levando em conta possíveis instabilidades na rede.

2.2.17 Exemplo de interface para configuração dos parâmetros de tempo

Em geral, os configuradores de mestres Profibus DP existentes no mercado disponibilizam uma interface de edição para cada um dos parâmetros de tempo que podem ser alterados. Nestes aplicativos já ocorre uma validação entre os parâmetros de tempo que podem ser correlacionados, porém a configuração é individual para cada mestre. Na Figura 2.21, a tela de configuração dos parâmetros de tempo do *Sycon*, da empresa *Hilscher* é apresentada.

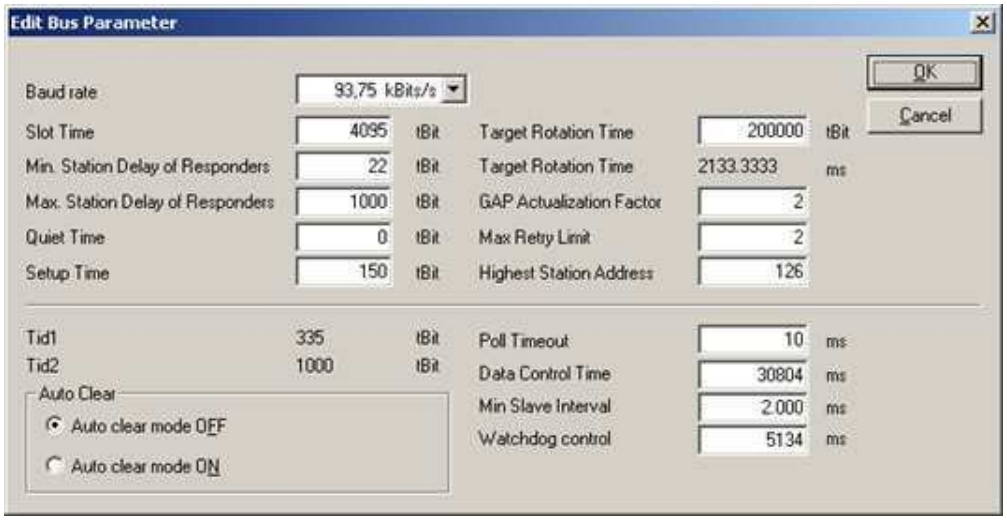

Figura 2.21: Tela de modificação dos parâmetros de rede do Sycon (Hilscher).

A Figura 2.22 mostra como o aplicativo SIMATIC (Siemens) apresenta os parâmetros de tempo para configuração do usuário.

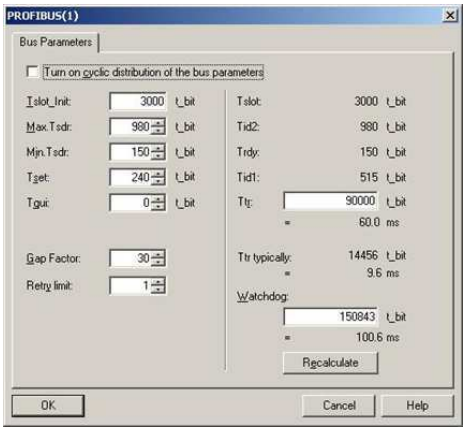

Figura 2.22: Tela de configuração de tempos do SIMATIC (Siemens).

2.3 Configuração do usuário

O projeto do sistema de automação pode ser realizado de diversas formas, levando em conta as dimensões (quantidade de instrumentos, mestres, cabeamento, etc) e também levando em consideração normas de projeto requeridas pelos usuários.

O projeto lógico da rede Profibus é realizado através de um software configurador do mestre a ser usado, em geral, desenvolvido pelo próprio fabricante do mestre. Neste trabalho é usado o

Hilshcer Sycon, a Figura 2.23 mostra a tela principal do configurador:

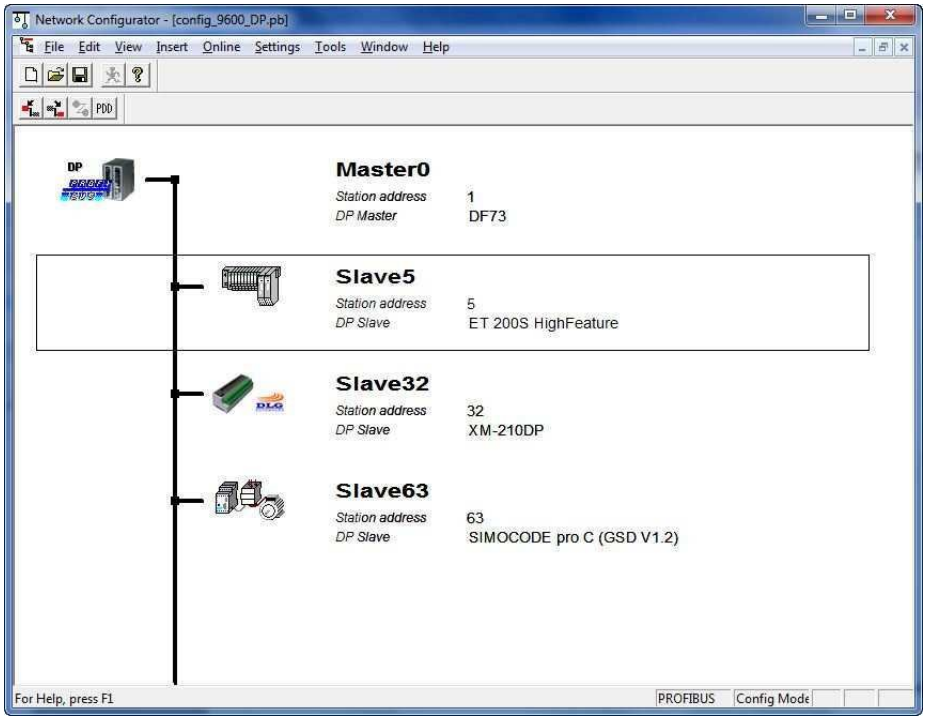

Figura 2.23: Configurador Hilscher Sycon

Para a configuração do mestre, são mandatórios os seguintes ítens:

1. Temporização: Taxa de comunicação, *Slot Time*, *Watchdog Time*, *TTR*, e demais parâmetros de tempo anteriormente citados que podem, em alguns casos, serem determinados automaticamente pelo configurador.

2. Endereçamento: a faixa de endereços que varia entre 0-125 está disponível para configuração dos equipamentos. A maioria dos equipamentos possui chaves físicas (*dip-switches*) para este endereçamento. A configuração deve refletir este endereçamento físico para cada modelo de equipamento correspondente.

3. Modelo dos equipamentos: Traz consigo o *Ident Number* que é único por modelo de equipamento e definido (registro) junto à Organização Profibus. Este código está presente dentro do arquivo GSD correspondente - arquivo que descreve o equipamento para o configurador.

4. Quantidade e tipos de módulos: O arquivo GSD apresenta os possíveis módulos que o escravo suporta (quantidades de bytes de entrada e saída). O usuário pode escolher a quantidade e combinação dos módulos e esta deve corresponder à instalação física dos

mesmos. Por exemplo: uma estação remota com 1 cartão de entrada e 1 cartão de saída deve ser adicionada no configurador com 1 módulo de entrada e 1 módulo de saída considerando a ordem e modelo dos cartões.

A Figura 2.24 apresenta a Tela de escolha dos módulos a serem usados por um equipamento, conforme abordado, a seleção de módulos deve levar em conta o projeto detalhado, que apresetará a quantidade e modelo de cartões utilizados.

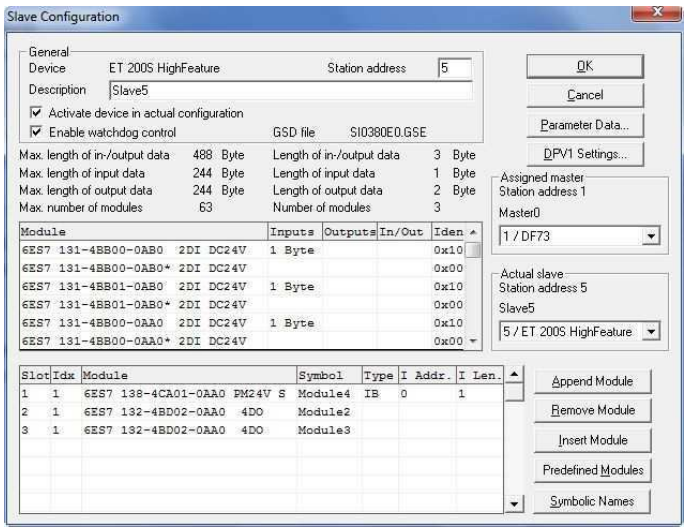

Figura 2.24: Janela do configurador Hilscher Sycon para configuração de módulos.

2.4 Problemas de Comunicação em Redes Profibus DP

2.4.1 Problemas de Instalação

Instalação inadequada (cabos de comunicação com impedância inadequada, falta de aterramento, cabeamento muito próximo a campos magnéticos variáveis, etc) podem causar modificação no sinal de comunicação causando instabilidades no funcionamento da rede.

Em (SOUZA, 2012) propõe-se uma metodologia para classificar e detectar problemas de instalação conhecidos fazendo análise do sinal de transmissão.

Embora não seja escopo deste trabalho detalhar causas e apontar problemas físicos de instalações de redes Profibus, não se pode deixar de citar que tais inadequações são causadoras da maioria dos problemas de partidas de plantas que optam pelo uso de redes digitais. Posteriormente, ao longo do funcionamento da planta, problemas de instalação podem se manifestar de

maneira ocasional sendo de difícil detecção e solução.

Alterações físicas não normatizadas em cabeamento e instalações podem resultar em problemas de comunicação, que são evidenciados através de deformações no sinal de comunicação observados em osciloscópio. O sinal apresenta características particulares de acordo com o problema físico em questão (SOUZA, 2012).

Na Figura 2.25, modificada de (MOSSIN, 2012) observa-se medições de sinal relacionadas a problemas de instalação introduzidos em laboratório. Algumas figuras foram sintetizadas em relação ao sinal adequado, porém aplicando-se o mesmo efeito sobre o sinal amostrado em laboratório. Os trabalhos mencionados documentam os critérios de avaliação do sinal de comunicação.

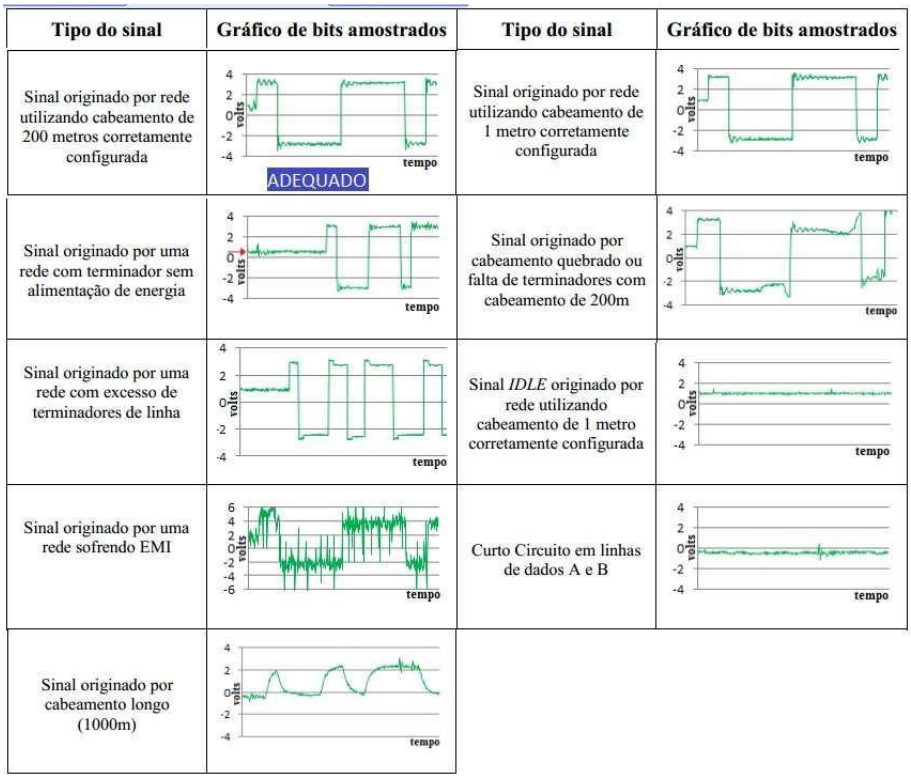

Figura 2.25: Alterações de sinal decorrentes de problemas de instalação - modificado de (Mossin, 2012)

2.4.2 Problemas de Configuração

Conforme explicado na seção sobre configuração (Seção 2.3), o usuário é responsável pela definição e configuração dos seguintes itens:

1. Configuração dos endereços na configuração do mestre.
2. Ajuste local (ou remoto) de endereço nos instrumentos.
3. Configuração do modelo correto de equipamento (GSD Correto).
4. Quantidades e tipos de módulos por escravo (que vão totalizar numero de entradas e saídas por para cada instrumento).
5. Configuração dos tempos (que pode ser auxiliada pelo software de configuração).

Assim, como a quantidade de itens não é pequena, a possibilidade de ocorrerem erros de configuração é grande. Principalmente quando executada por usuários sem treinamento adequado. Desse modo para cada um dos ítens acima citados podem ocorrer os seguintes erros a seguir:

1. Endereçamento errado na configuração: Atribuição de endereços na configuração do mestre que não foram previstos na arquitetura da instrumentação. Um sintoma deste tipo de problema é a detecção na rede dos endereços estavam previstos no projeto, porém o mestre não os configura. Em contrapartida o mestre pode estar procurando por um outro endereço (endereço errado na configuração) que também não responde. Ou mesmo, tal endereço de projeto não foi contemplado na configuração.

2. Endereçamento duplicado ou diferente do projeto: O instrumentista, no momento da instalação, ou em bancada, troca o endereço do instrumento para um incorreto em relação a arquitetura de instalação. Vários sintomas contemplam este caso: resposta corrompida sempre que um determinado endereço é consultado, inexistência do endereço na captura de rede e problemas de parametrização ou configuração, em casos que o endereço foi acidentalmente trocado pelo instrumentista com de outro instrumento da mesma rede.

3. Falha de Parametrização: (*Parameter Fault*) Ocorre quando o endereço de um instrumento foi trocado com o de outro (no ajuste de instalação) e ambos são de modelos diferentes. Ou caso o modelo instalado do instrumento em questão seja uma variação do fabricante (*firmware* diferente, características diferentes). Em ambos casos, o Ident Number enviado pelo mestre durante o processo de parametrização é diferente do Ident Number do escravo. O escravo compara o Ident Number recebido com o seu, rejeita a parametrização e sinaliza

através do bit de diagnóstico *Parameter Fault.*

4. Falha de Configuração: (*Configuration Fault*) Ocorre quando há discrepância entre os módulos que foram configurados e os módulos que estão presentes no escravo. Módulo é o nome dado a uma variável ou conjunto de variáveis que pode ser selecionado para o equipamento, através do GSD para integrar a configuração. Em estações do tipo entradas e saídas remotas, o módulo em geral é equivalente aos cartões de entrada e saída correspondentes. Caso o usuário configure módulos errados, ou na ordem errada em relação aos cartões da remota instalados em campo esse problema ocorrerá. Ou ainda, durante a operação, um dos módulos entra em falha - significa uma difereça de configuração, que irá causar este problema.

5. Erro na configuração do TTR ou Watchdog Time: O TTR e o Watchdog Time são automaticamente calculados pelos softwares configuradores atuais. Porém, este cálculo é baseado na quantidade de entradas e saídas dos escravos e os atrasos máximos que cada um pode incluir na rede. Porém, em alguns casos, deseja-se adicionar outro mestre (para controle de ativos por exemplo) e pode haver instabilidades na rede (retries). Estes eventos dilatam o TRR (Real Rotation Time) e geram um atraso no ciclo de atualizações dos escravos. Caso o TWdg esteja muito 'justo' então podem acontecer quedas aleatórias de instrumentos devido ao Watchdog Time ser ultrapassado. Ferramentas de captura modernas fazem a leitura do período de atualização do escravo e podem indicar se o TWdg foi incorretamente dimensionado.

No capítulo seguinte serão apresentadas algumas ferramentas de simulação relacionadas ao protocolo Profibus que embora estas não representem fonte de inspiração para o desenvolvimento da ferramenta deste trabalho devido sua finalidade, representam o estado da arte de simuladores de mestre e escravos Profibus DP.

Capítulo 3

Ferramentas de Simulação de Redes Profibus

As ferramentas de simulação têm sido cada vez mais usadas para reduzir custos em projetos industriais. Elas têm auxiliado reduções no tempo de partida e comissionamento e em alguns casos viabilizando projetos que seriam rejeitados devido ao alto custo do período de testes após a instalação da rede real.

A importância da simulação, principalmente, de sistemas complexos, já é uma questão consolidada. Segundo (HOSSEINPOUR e HAJIHOSSEINI, 2009), sistemas simulados podem auxiliar na seleção de um melhor projeto, assim como na solução de problemas em projetos já implantados.

Embora os conceitos de simulação e emulação sejam usados como sinônimos em muitos casos, inclusive em algumas partes deste trabalho, deve-se lembrar de que são conceitos diferentes. Conforme (MCGREGOR, 2002), a simulação é usada para desenvolver diferentes soluções com o objetivo de se encontrar a melhor, baseando-se em conjunto de dados predefinidos. Já a emulação reflete mais precisamente o sistema que será implementado e pode ser usado para realizar verificações que visam garantir o desempenho ou tempo de reação de um sistema.

Neste trabalho o aplicativo desenvolvido possui aspectos de emulador e de simulador. Ele pode ser considerado um simulador de uma rede Profibus DP, pois utiliza um conjunto de informações e comportamentos extraídos do protocolo para efetuar a execução de um comportamento que tenta se aproximar do comportamento real. Este pode ter sua configuração alterada de modo a produzir diferentes saídas de acordo com as configurações de rede. O aplicativo também se

encaixa como emulador do futuro hardware de aquisição, auxiliando o desenvolvimento da ferramenta Profidoctor e permitindo a demonstração do funcionamento da mesma sem a necessidade de uma rede real.

A seguir serão apresentadas duas ferramentas dedicadas à simulação de equipamentos Profibus DP. A primeira, mais precisamente, destinada a emulação do mestre em um PC para comunicação com escravos reais. A segunda tem a função de emular os escravos para permitir a simulação de uma planta. Embora nenhuma delas possa ser comparada diretamente a este trabalho devido a suas finalidades distintas, elas representam aplicações dentro do universo de simuladores, cuja função é simular ou emular estações Profibus DP.

3.1 Simulador de Mestre Profibus DP/DP-V1 B+W

O Simulador de Mestre Profibus é dotado de um software e um conversor serial RS232/RS485 ou USB/RS485 com a finalidade de fazer interface com um computador pessoal. A pilha de protocolos é executada no computador e a porta serial (ou USB) é usada para fazer interface com a rede Profibus DP real. A montagem do sistema é simples como mostra a Figura 3.1. A execução do mestre ocorre sob o sistema operacional Windows. A velocidade da rede é limitada a 19.200 bps pela ferramenta. Isso torna possível a implementação e execução da camada de enlace do mestre Profibus DP através do Windows sem prejudicar a operação da rede (HUANG, 2005).

Embora denominado PROFIBUS Master Simulator, o software permite ser utilizado como um mestre real através da porta serial.

A empresa *Bihl+Wiedemann* (B+W) é a fabricante e distribuidora desta ferramenta, embora outros fabricantes também forneçam com suas marcas através de OEM.

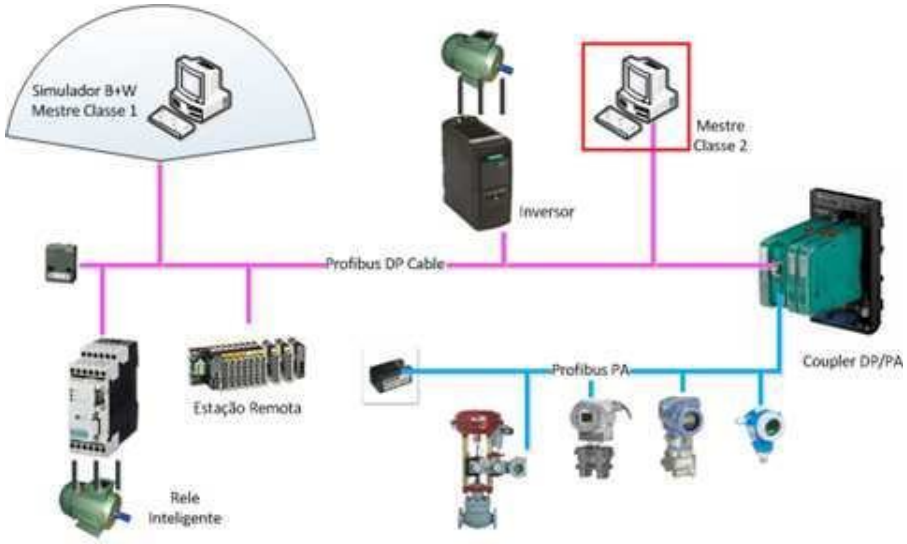

Figura 3.1: Arquitetura para uso do Simulador de Mestre - PROFIBUS Master Simulator.

O aplicativo permite a seleção da porta de comunicação serial do computador e seleção do endereço do mestre, mostrado na janela retratada na 3.2.

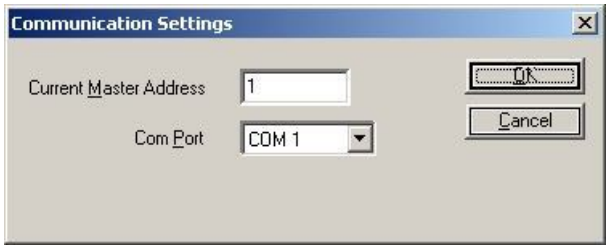

Figura 3.2: Seleção da porta serial do PC no simulador do mestre.

É possível construir uma configuração com escravos através de uma janela de adição de escravos conforme a Figura 3.3, permitindo também modificar um *Ident Number* de um escravo para efetuar testes.

Figura 3.3: Tela de adição de escravos do simulador de mestre.

Ele apresenta janelas para monitoração dos escravos e modificação dos dados de aplicação que são parte do DATA_UNIT. A Figura 3.4, apresenta a janela em questão, que permite ainda a visualização dos dados de diagnostico dos escravos.

Há janelas para envio de pacotes destinados a comunicação acíclica que podem ser controlados pelo usuário ou acionados de maneira continua conforme a Figura 3.5.

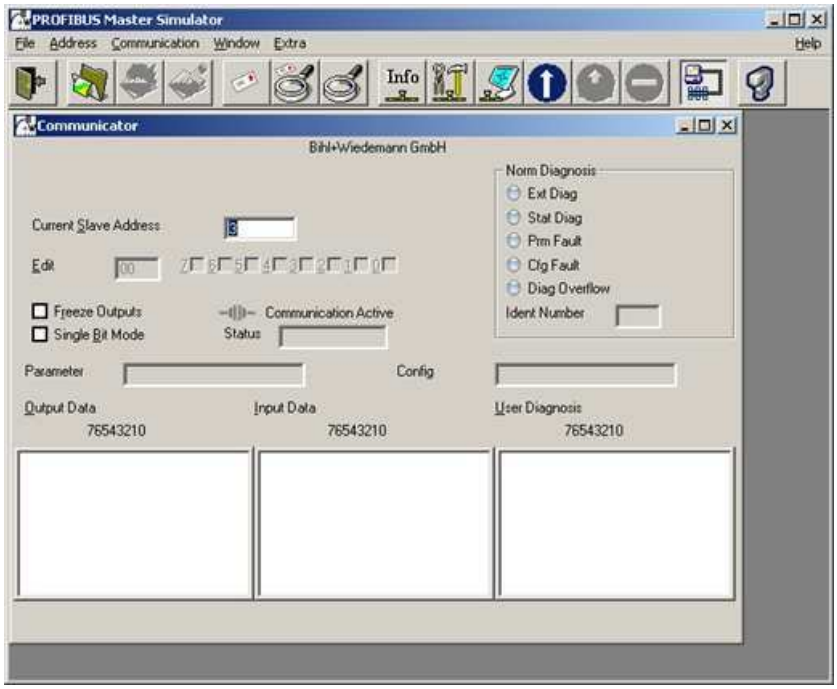

Figura 3.4: Tela Principal do PROFIBUS DP Master Simulator.

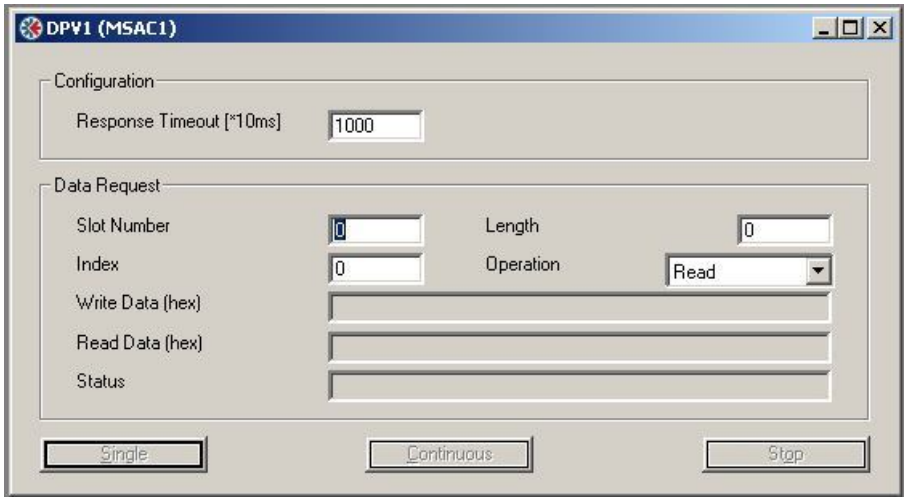

Figura 3.5: Entrada de dados para comunicação acíclica.

Em (NARDUCI et al., 2010) apresenta-se um simulador semelhante em desenvolvimento com o objetivo de integrar a ferramenta Profidoctor.

3.2 Sistema de Simulação de escravos - SIMIT

O ambiente SIMIT é um pacote completo para simulação de plantas e sistemas de automação. Fornecido pela Siemens, ele é composto por uma aplicação e equipamento (SIMBA - Figura 3.6) cujo objetivo é ser conectado ao PLC através de uma rede Profibus DP. Este equipamento tem a função de emular os escravos Profibus DP. Seus dados de aplicação podem ser modificados através de uma estação de operação dedicada a este fim.

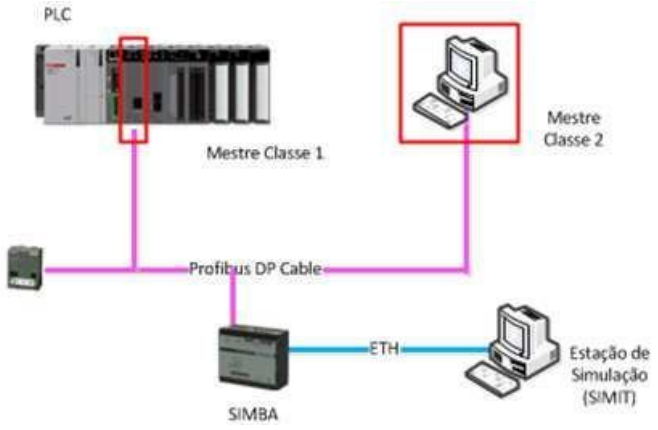

Figura 3.6: Arquitetura do SIMBA usado para simulação de escravos.

A Figura 3.7, a seguir, mostra a arquitetura do sistema SIMIT:

Através de um ambiente integrado (3.8), é possível construir uma interface para o usuário e simular o comportamento da planta ou processo usando blocos de controle.

Após o desenvolvimento da interface e comportamento do sistema, o usuário transfere esta configuração para o SIMBA (IM-PBDP), através de um canal ethernet, que passa a se comportar como os equipamentos Profibus DP reais em relação ao PLC conforme a Figura 3.9. O SIMBA pode operar simulando até 8 redes com até 125 equipamentos em cada uma das redes. O custo do hardware de simulação varia de acordo com a quantidade de canais desejados.

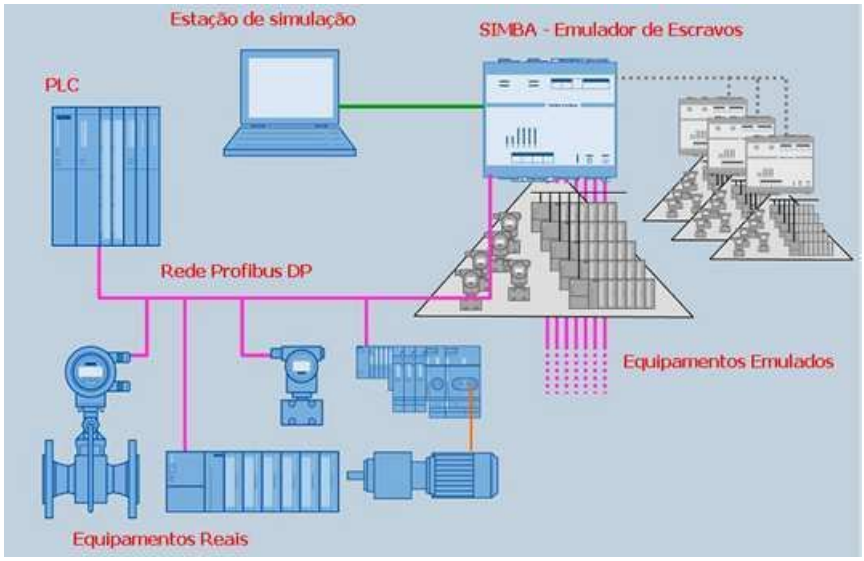

Figura 3.7: Arquitetura do sistema SIMIT (Siemens).

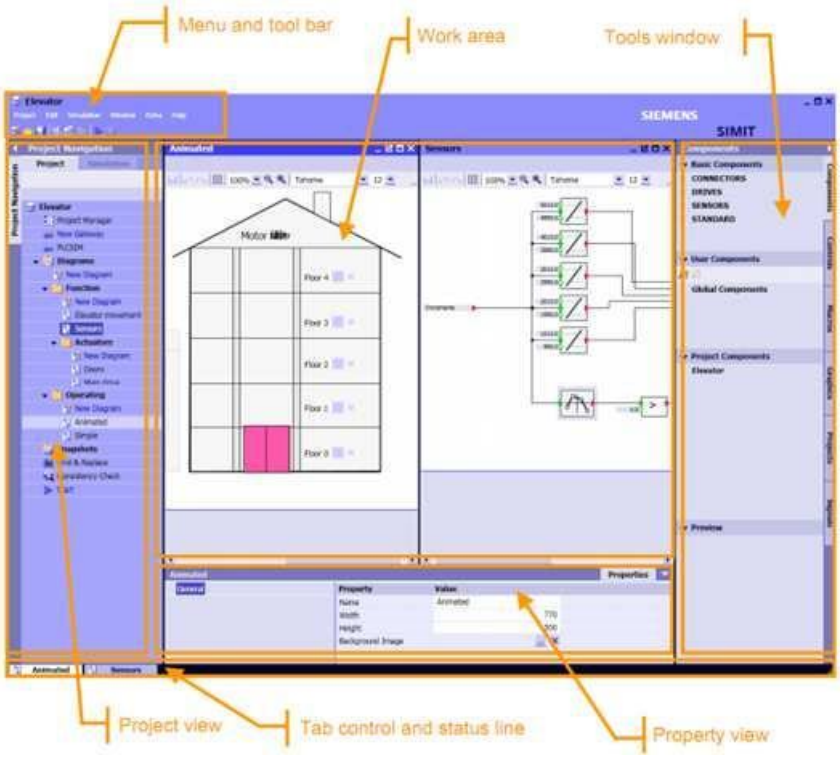

Figura 3.8: Ambiente de desenvolvimento da simulação e interface (Siemens).

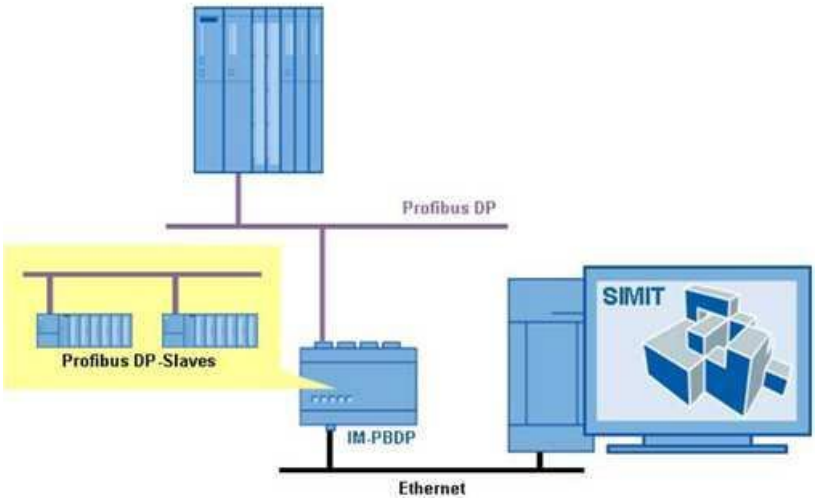

Figura 3.9: Estação de simulação SIMIT (Siemens).

Este simulador tem sido usado também como base para produção de alguns trabalhos acadêmicos como em (Klang e Lindholm, 2005). Pode-se usar o sistema de simulação para programar um modelo e simular o comportamento de um sistema ou equipamento fornecendo variáveis através da rede Profibus DP.

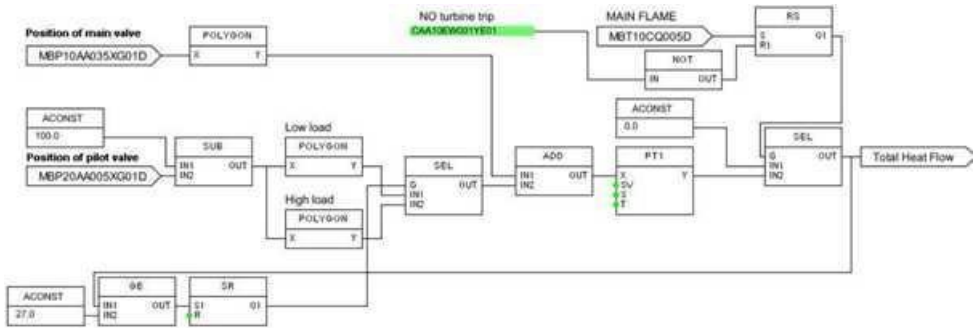

Figura 3.10: Modelo de uma turbina a gás programado no SIMIT (Klang e Lindholm, 2005).

3.3 Outras Ferramentas e Trabalhos Relacionados

Neste item busca-se resumir outros trabalhos relacionados a simulação de redes Profibus DP, embora estas tenham um menor destaque ou sejam implementações dedicadas para obtenção de resultados específicos.

A empresa SprintPROSI desenvolveu uma placa comercial (DPC31 USB MultiSlave CARD) utilizando circuitos integrados dedicados a controladores profibus (DPC31) permitindo simular até 16 escravos. Esta tem a mesma função do SIMIT (3.2).

Figura 3.11: Placa Simulação 16 Slaves Profibus DP.

Em (ANTONOPOULOS et al., 2004) há o desenvolvimento de uma aplicação em Visual Basic onde o usuário pode configurar os parâmetros de rede em conjunto com parâmetros experimentais (B.E.R, *workload*, número de estações, número de mestres, etc) com o objetivo de

determinar o atraso médio de resposta dos escravos e o *throughput* permitindo criar um modelo de simulação para o Profibus DP.

Em (TORRES et al., 2010) um simulador simples é aplicado para verificar o período de aquisição em redes Profibus DP para um determinado conjunto de escravos particulares.

Em outro trabalho, configuram-se vários mestres e programa-se falhas aleatórias na forma de erros em bits, o experimento é realizado na pratica através do uso de vários mestres modificados onde se pode inserir os erros e verificar os diversos motivos de interrupção da comunicação em uma rede Profibus DP (CARVALHO et al., 2005).

Alguns trabalhos procuraram desenvolver simuladores dos atrasos da rede Profibus DP, baseando-se nos parâmetros de rede e usando a plataforma *Simulink* do MATLAB. (CASANOVA et al., 2006) e (HE e GUO, 2008) desenvolveram o mesmo simulador, modelando, conforme a Figura 3.12, os atrasos existentes em redes Profibus DP, para verificar o efeito em conversões AD/DA que realizadas nos escravos (Figura 3.13).

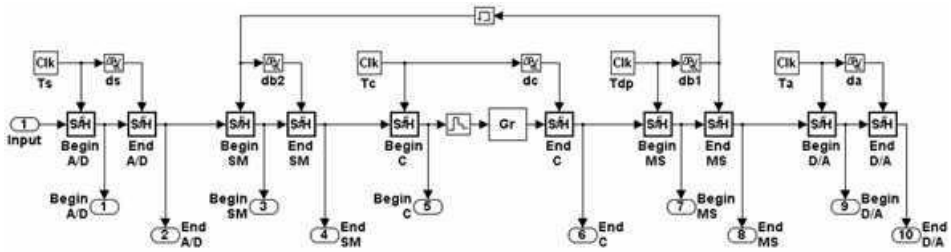

Figura 3.12: Modelo transmissão de uma rede Profibus DP no Simulink (CASANOVA, 2006).

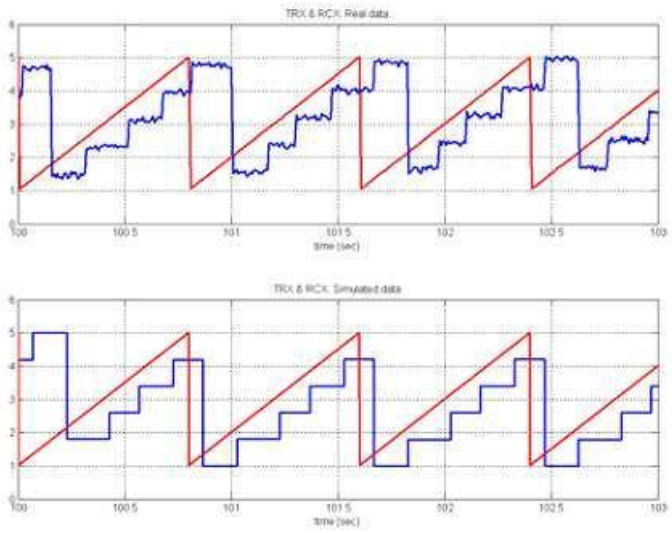

Figura 3.13: Transmissão e Recepção real e simulada de um dado analógico (CASANOVA, 2006).

Neste trabalho os objetivos do simulador diferem dos simuladores apresentados, pois os trabalhos citados são orientados em verificar os diversos efeitos das redes Profibus DP em sistemas de controle e no desempenho da rede, enquanto este trabalho busca mostrar os efeitos de falhas na camada de enlace e apresentar aos usuários efeitos decorrentes no protocolo que causam a interrupção da comunicação.

No próximo capítulo, serão mostradas as características do Profibus Simulator, a metodologia utilizada para seu desenvolvimento e ferramentas envolvidas.

Capítulo 4

Desenvolvimento da Ferramenta de Simulação

Após a apresentação da camada de enlace e dos diferentes simuladores mais próximos a este trabalho, dedica-se esta seção a retratar a estrutura e desenvolvimento do simulador.

O desenvolvimento iniciou-se com a extração dos requisitos desejados. A lista de requisitos apresentada na Seção 4.2 influenciou a definição da plataforma de desenvolvimento conforme mostrado na Seção 4.3, que foi aprovada visando também o cumprimento do prazo para defesa deste trabalho.

A arquitetura da ferramenta presente na Seção 4.4 é apresentada segundo módulos funcionais sem exibir em detalhes todas as funções, classes e métodos utilizados, pois fugiria do escopo deste trabalho. A interface gráfica do simulador é apresentada já evidenciando as funcionalidades do simulador e possível efeito esperado na ferramenta de análise 4.5.

4.1 Comparativo entre as ferramentas apresentadas

A tabela 4.1 apresenta um comparativo entre as ferramentas comerciais apresentadas e o desejado para a ferramenta proposta.

Embora as ferramentas apresentadas não possam ser comparadas diretamente com o simulador proposto neste trabalho, ainda é possível listar as características desejadas sob o ponto de vista de simulação de redes.

Tabela 4.1: Comparativo entre as ferramentas e requisitos para o Simulador Profibus

Característica	Simulador Mestre B+W	Simulador de Escravos SIMIT	Simulador Profibus
Simulação de Mestres	SIM	NÃO	SIM
Simulação de Escravos	NÃO	SIM	SIM
Necessita de hardware externo para operação	SIM	SIM	NÃO
Necessita de uma rede real para simulação	SIM	SIM	NÃO
Período para aprendizado do simulador	Curto	Grande	Curto
Permite interaçao com rede real	SIM	SIM	SIM

Para permitir a comunicação com o aplicativo Profidoctor, um protocolo simples de aplicação foi definido entre as ferramentas dando origem a uma especificação de pacote. Este terá a função de carregar os dados de cada comando da rede gerado por cada iteração do simulador.

4.2 Requisitos e Arquitetura da Aplicação

1. Permitir a construção de uma configuração de rede, simular a mesma e modificar o estado dos equipamentos durante a execução da simulação.

2. Permitir a simulação de erros que ocorrem em rede.

3. Manter uma interface amigável semelhante à interface de ferramentas de mercado (como Profitrace) para permitir treinamento e reduzir o tempo de aprendizado (aprendizado compartilhado)

4. Permitir dividir o simulador em módulos de simulação de mestre e escravos para validar ambos separadamente em testes de certificação, se desejado.

Historicamente, este trabalho faz parte de um projeto de pesquisa que contempla: uma ferramenta de análise (Profidoctor)(MOSSIN, 2012), um Hardware para aquisição dos frames de uma rede Profibus real e por fim o simulador, que tem como uma de suas funções emular

também o hardware de aquisição. Assim sendo, logo ao inicio do projeto do simulador buscou-se definir um protocolo de comunicação padronizado para que a ferramenta de análise, o simulador e o hardware de aquisição se comunicassem usando o mesmo mecanismo. A priorização desta tarefa permitiria antecipar o teste da ferramenta de análise sem a necessidade do hardware.

Devido ao requisito da comunicação utilizando meio ethernet procurou-se utilizar um protocolo baseado em TCP, para evitar preocupações como ordenação de pacotes e confiabilidade da comunicação. Embora o TCP tenha um mecanismo mais complexo que a comunicação via UDP, atualmente a comunicação via TCP está disponível para a maioria das plataformas de hardware e software, então isso não foi considerado um problema. Outro ponto importante na escolha do meio ethernet e do TCP é que estes abrem um grande leque de aplicações, como a monitoração remota e o uso de múltiplas estações executando a ferramenta de análise.

Como o hardware ou o simulador seriam os fornecedores de dados e a ferramenta Profidoctor seria a consumidora, então atribuiu-se a função de servidor TCP ao simulador e a função de cliente TCP ao Profidoctor. A Figura 4.1 ilustra a arquitetura de comunicação entre o software Profidoctor, o Profidoctor Hardware e uma rede Profibus DP:

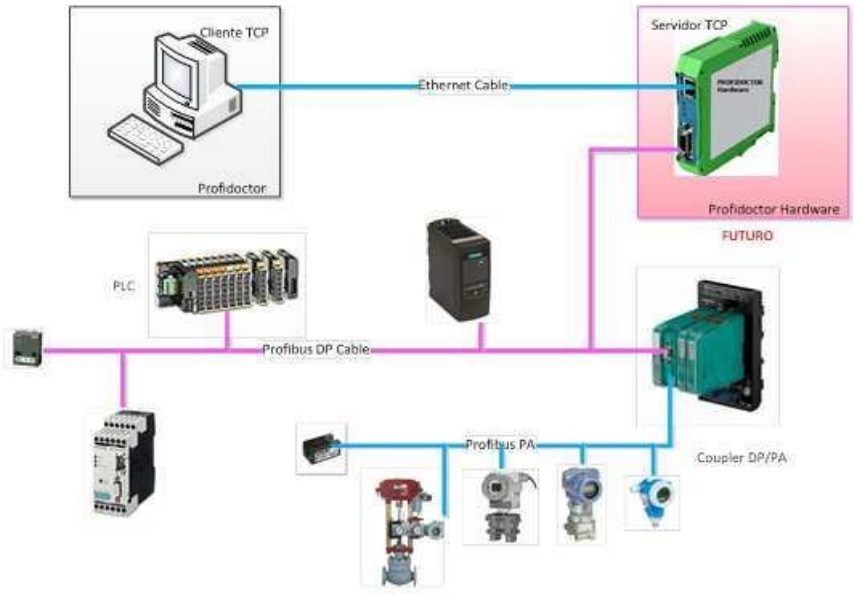

Figura 4.1: Arquitetura do Profidoctor aplicado a uma rede Profibus DP.

Buscando uma possível integração ao software Profidoctor no futuro, optou-se por desenvolver o simulador em uma plataforma do tipo PC, o que facilitaria a integração e permitiria selecionar o ambiente de desenvolvimento da aplicação do simulador num espectro bem mais amplo, devido a diversidade existente para PC.

O simulador deve ser inserido de modo a refletir o comportamento do equipamento de aquisição (Profidoctor Hardware) e da rede Profibus DP onde o mesmo está inscrido. A Figura 4.2 mostra a abrangência do simulador dentro da arquitetura:

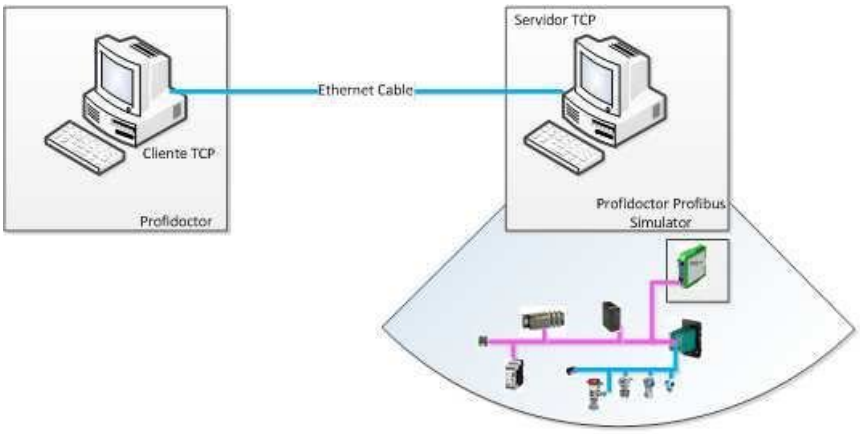

Figura 4.2: Abrangência do Simulador.

4.3 Plataforma de Desenvolvimento

Uma integração futura com a própria ferramenta de análise é um requisito, o que limitaria o uso de ferramentas comerciais de simulação, uma vez que poderia limitar a integração e distribuição junto à ferramenta Profidoctor final. O desenvolvimento da ferramenta de análise fora iniciado usando-se a linguagem JAVA, buscando-se uma independência entre os sistemas operacionais mais populares (Windows e Linux), além de ser uma linguagem que apresenta uma biblioteca rica em componentes e com ambientes de desenvolvimento e teste gratuitos.

Como ambos estariam sendo desenvolvidos na mesma linguagem, além do aproveitamento de código, evitam-se possíveis problemas de interoperabilidade: como formatos numéricos, datas, etc. Estas características podem diferir entre linguagens de programação.

Pensou-se no desenvolvimento de classes e pacotes que fossem comuns a ambos os aplicativos, acelerando o desenvolvimento de ambos e evitando duplicação de trabalho. Para auxilio deste trabalho em grupo, a ferramenta SVN (sistema de controle de revisões) foi usada facilitando assim o controle de versões desses componentes comuns. Um servidor de uso gratuito foi escolhido para permitir o compartilhamento e controle de versão do código.

Como ambiente de desenvolvimento, optou-se pelo NetBeans IDE que é uma ferramenta de código aberto, gratuita e patrocinada pela companhia Oracle. A Tabela 4.2 resume a plataforma

de desenvolvimento utilizada:

Tabela 4.2: Itens da plataforma de Desenvolvimento do Simulador.

Plataforma de Hardware	Laptop Toshiba Satellite modelo A665-S5179
Sistema Operacional	Windows 7 Running on x64
Plataforma de Software de Desenvolvimento	NetBeans IDE 6.9 (Build 201011082200) Java: 1.6.0_21 Java HotSpot(TM) Client VM 17.0-b17
Sistema de Controle de Versões	SVN ProjectLocker

4.4 Estrutura da Aplicação

A aplicação foi dividida em módulos funcionais conforme ilustrado na Figura 4.3. Um resumo de cada um dos módulos é apresentado posteriormente. A ferramenta pode ser usada para simular uma rede com qualquer número de equipamentos dentro da especificação.

A maneira como o simulador foi estruturado evita que o uso do PC como plataforma para simulação cause problemas de atraso ou imprecisões devido a características do sistema operacional (HUANG, 2005) que não incluem a operação das aplicações em tempo real. Nenhum módulo ou biblioteca de tempo real foram utilizados.

A operação do simulador ocorre como uma aplicação qualquer e o *timestamp* dos frames são calculados de acordo com as regras de tempo exigidas pelo protocolo, permitindo o uso do simulador sem sobrecarregar o computador e também possibilitando a pausa e o reinício da simulação a qualquer momento sem comprometer os resultados.

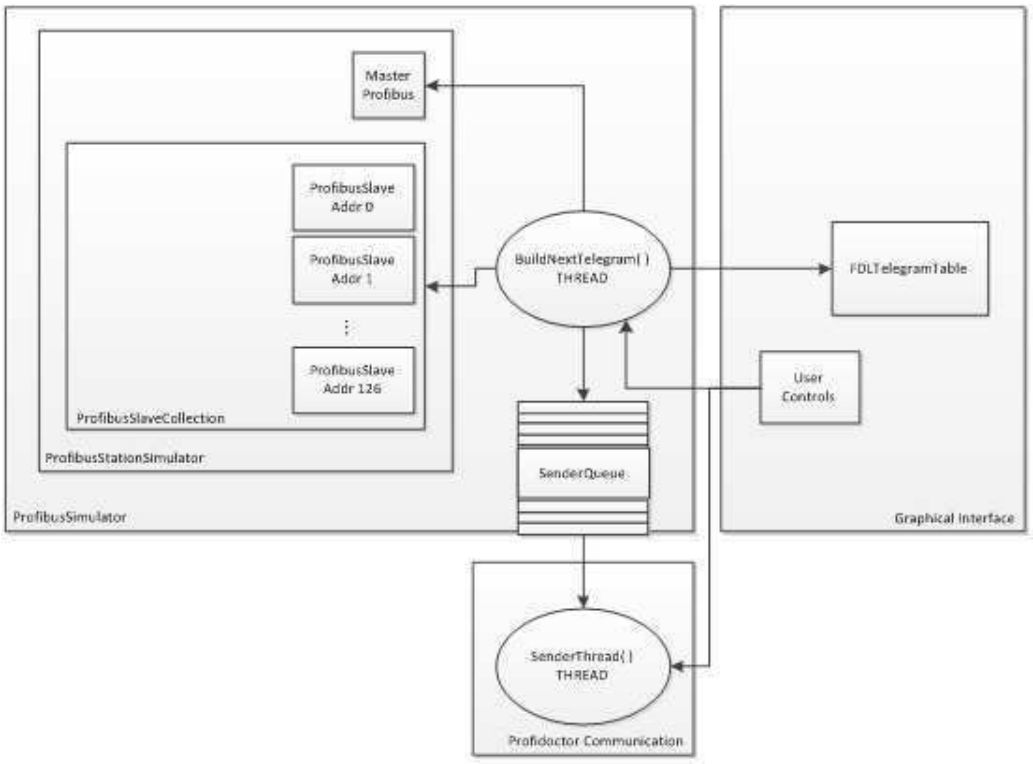

Figura 4.3: Estrutura modular da aplicação.

Tabela 4.3: Lista e descrição dos Módulos do Simulador.

Modulo	Descrição
Profibus Master	O módulo Profibus Master incorpora parte da pilha de protocolos necessária para que se possa estabelecer a comunicação do tipo mestre-escravo.
Profibus Slave	Este módulo representa parte da pilha de protocolos de um escravo para permitir a resposta de comandos oriundos do mestre
ProfibusSlaveCollection	Representa o conjunto de todos os escravos da simulação. Este módulo é usado como um ponto de acesso aos atributos de algum escravo. Pode ser usado futuramente para facilitar a persistência do estado de todos os escravos.
ProfibusStationSimulator	Contém o estado de todas as estações, estado do mestre e permite acesso a qualquer atributo de qualquer estação por um módulo externo.
ProfibusSimulator	Núcleo do software de simulação e atribui-se a esse módulo a tarefa de fazer a interação entre o Modulo de Mestre e escravos. Neste módulo é onde esta a thread *BuildNextTelegram( )* que processa a iteração do mestre com os escravos, adiciona os pacotes gerados a uma fila de saída destinada a coleta por uma ferramenta externa e atualiza a interface gráfica. É responsável ainda pela geração do *time-stamp* simulado. Informação de tempo utilizada para aproximação em relação a uma rede real.
Graphical Interface	Formada pelo conjunto dos controles (*User Controls*) e pela tabela *FDL-TelegramTable* que apresenta o resultado da simulação a cada iteração.
Profidoctor Communication	Contém o servidor TCP que pode receber a conexão de uma aplicação Profidoctor. Responsável por transmitir os frames simulados no formato padronizado entre a ferramenta de análise e o simulador

4.5 Interface do simulador com o usuário

Inicialmente, nas versões preliminares do simulador, não houve preocupação com a interface gráfica, pois se acreditava que ele não apresentaria interface e seria executado como um componente da ferramenta de análise. Porém, o escopo do uso da ferramenta como aplicativo para treinamento, assim como a necessidade de permitir alterações online, destacou que havia uma necessidade de iteração com o usuário maior que a inicialmente planejada.

Isso motivou a construção de uma interface gráfica semelhante à usada na ferramenta de análise (Profidoctor), permitindo que seu uso seja intuitivo e o aproveitamento de componentes gráficos comuns tanto ao simulador quanto a ferramenta de análise. A Figura 4.4 apresenta a interface mais recente do simulador, que será detalhada a seguir:

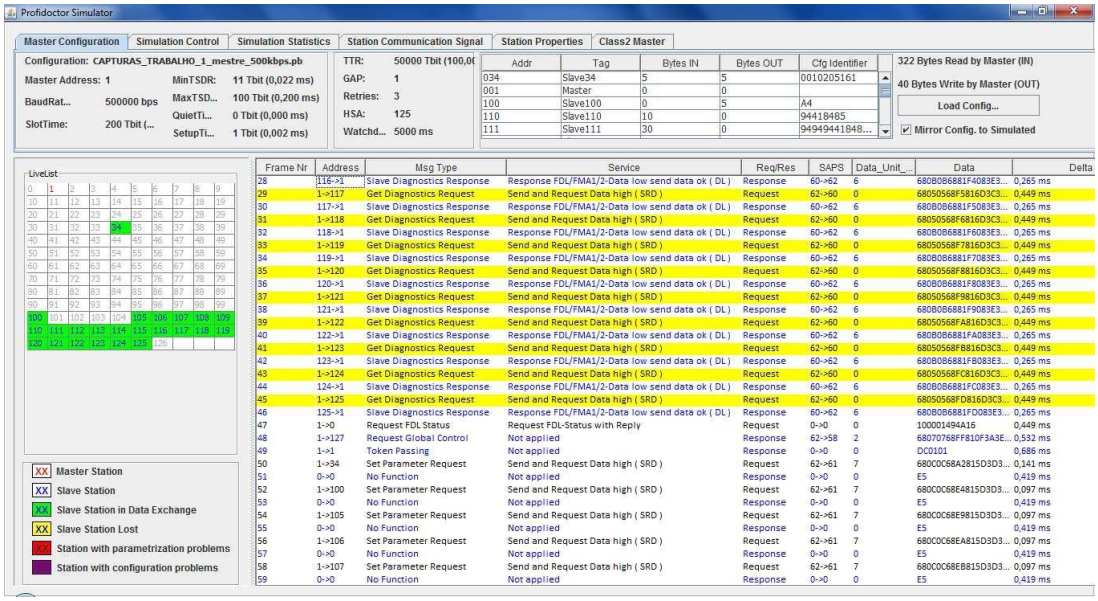

Figura 4.4: Interface gráfica do simulador.

A interface foi dividida em painéis que permitem as seguintes ações:

1. Master Configuration: Permite carregar arquivos de configuração do tipo (PB), padrão utilizado pelo configurador Hilscher Sycon. Permite também, através de caixa de checagem, copiar as configurações dos escravos (oriundas do arquivo de configuração do mestre) para os escravos simulados, evitando utilizar 2 arquivos (um para configuração do mestre e outro para configuração da rede real).

2. Simulation Control: Permite iniciar e parar a simulação de rede em qualquer momento, modificar a tela de captura e estabelecer comunicação com o Profidoctor.

3. Simulation Statistics: Apresenta informações relativas à análise da captura simulada, como por exemplo, o tempo de ciclo e o TRR.

4. Live List: Representa a lista de estações simuladas, do ponto de vista do mestre. Caso use o botão direito do mouse, pode ativar ou desativar qualquer estação durante a execução da simulação.

5. Station Properties: Ao se clicar em um endereço na Live List, este é selecionado e esta aba representa as características desta estação. Pode-se alterar os bits de diagnósticos da estação e modificar seu Ident Number ou configuração.

6. Station Communication Signal: Esta aba é funcional apenas para o aplicativo Profidoctor, cuja função é gerar os sinais de comunicação analógicos amostrados correspondentes a cada frame e enviar ao aplicativo de análise. Apenas a caixa de checagem Readable é funcional para o simulador.

7. Class 2 Master: Utilizada para permitir que o Mestre simulado possa interagir com escravos de uma rede real. Embora prevista nesta versão do simulador, esta característica só estará presente numa versão futura, pois foge do escopo de simulação deste trabalho.

Uma tabela contém todos os frames enviados pelo mestre e respondidos pelos escravos operacionais. A Figura 4.5 apresenta essa divisão:

4.5.1 Controle seletivo da simulação

Na aba Simulation Control foram adicionados os controles da execução do simulador. Os botões Step ou Start disparam a *thread BuildNextTelegram( )*, responsável pela interação entre os módulos de simulação de mestre e escravos. Uma vez iniciada a simulação ela se comporta como uma ferramenta de captura de pacotes recebendo os frames do hardware de aquisição.

O botão Step permite executar uma iteração da simulação por vez. Esta opção é muito interessante para poder observar o comportamento da rede exatamente após inserir uma falha ou um comando de parametrização e configuração.

O botão Clear All Frames limpa a tabela de telegramas. A Figura 4.6 mostra estes controles em detalhes:

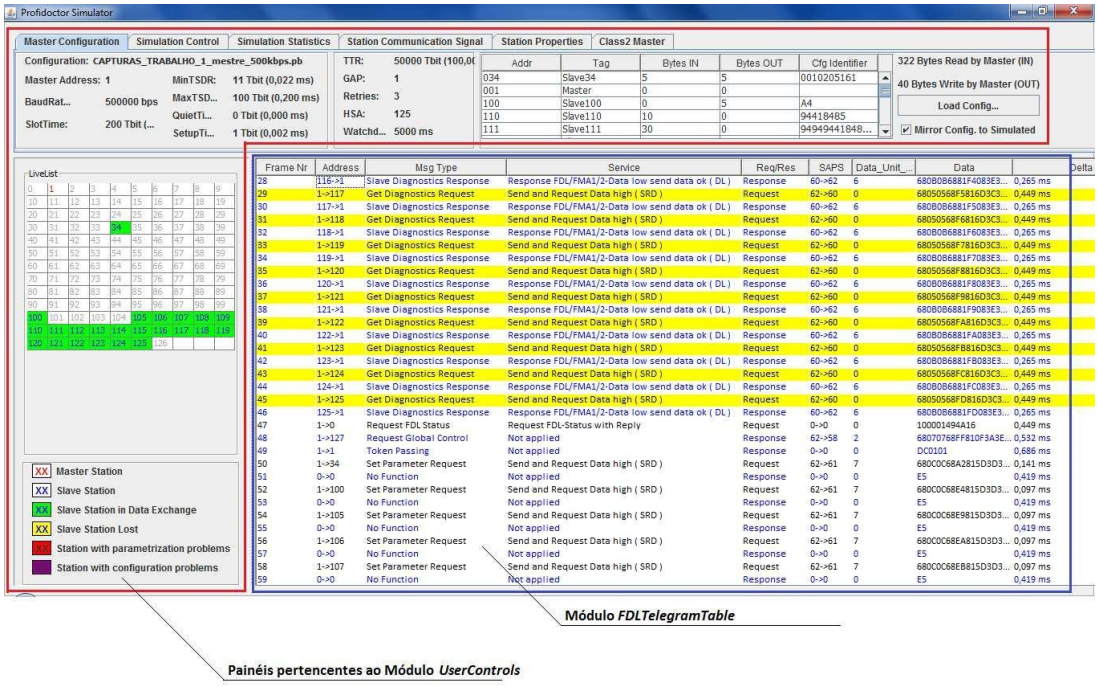

Figura 4.5: Detalhe dos módulos pertencentes à interface gráfica.

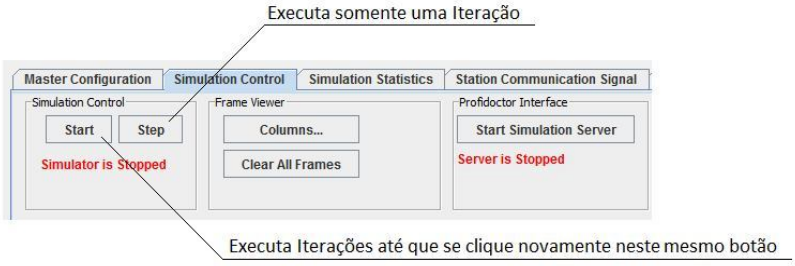

Figura 4.6: Detalhe do painel de controle da execução do simulador.

4.5.2 Live-List

O painel de live-list tem a função de permitir a seleção dos escravos para efetuar a modificação do seu comportamento durante a simulação. Ele também reflete o estado corrente dos equipamentos na rede simulada. A Figura 4.7 aponta a relação entre os painéis de live-list e atributos dos escravos.

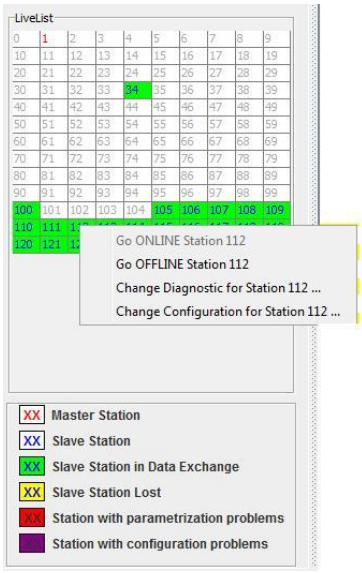

Figura 4.7: Detalhe da seleção de um escravo para modificação.

A modificação das propriedades de um escravo pode ocorrer durante a execução do simulador, permitindo criar situações para a ferramenta de analise de maneira dinâmica, por exemplo: simular a retirada ou adição de uma nova estação na rede.

4.5.3 Diagnóstico do Escravo

A Figura 4.8 mostra a aba que permite visualização e alteração do diagnóstico do escravo.

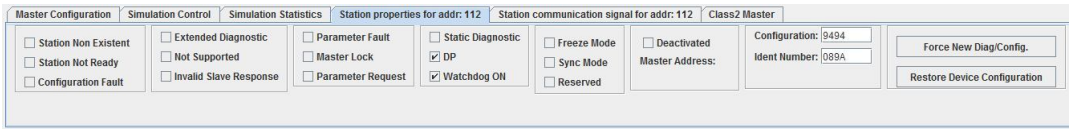

Figura 4.8: Detalhe do controle do estado operacional de um escravo.

Ela traz também a configuração de módulos e Ident Number do escravo em questão. Com estes itens, esta tela se torna a mais utilizada para testes de falhas de rede devido problemas de configuração. Permitindo alterar a configuração e Ident Number dos escravos em relação à configuração do mestre.

4.5.4 Sinal de comunicação

Como um dos resultados de um trabalho anterior (SOUZA, 2012), vários ensaios foram realizados para observar o formato do sinal de comunicação quando um problema de instalação

era inserido em uma rede. O catálogo de tais problemas e sinais resultantes permite notificar a ferramenta de analise sobre uma inadequação no sinal de comunicação.

Este catálogo fora inserido no simulador e permite inserir um problema físico no resultado de uma captura realizada por uma ferramenta de aquisição. Tais características podem ser alteradas através do painel de propriedades analógicas de sinal, veja na Figura 4.9:

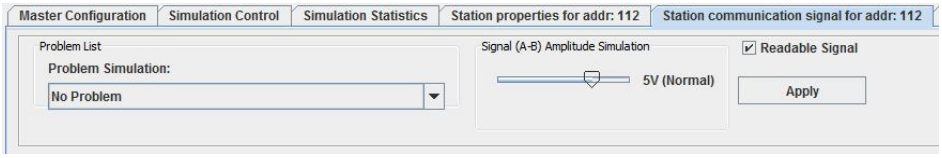

Figura 4.9: Controle para modificações do sinal de comunicação.

Os problemas catalogados podem ser selecionados através de uma caixa de combo, conforme a Figura 4.10 e a amplitude do sinal também pode ser modificada segundo o que a norma classifica como normal e anormal.

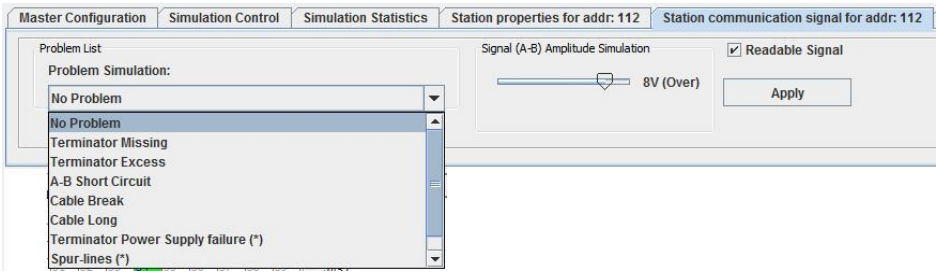

Figura 4.10: Controle para modificações do sinal de comunicação.

Este item foi incluído no simulador para uso em teste da ferramenta de análise em (MOSSIN, 2012), porém neste trabalho não aborda-se o teste desta funcionalidade, contemplada no trabalho citado.

4.5.5 Tabela de captura

Esta parte do simulador é a que mais se assemelha a uma ferramenta de captura de frames de rede. O resultado ou saída principal do simulador é acrescentado à tabela de frames, que é parte do módulo *FDLTelegramTable*. Esta tabela apresenta o frame decodificado segundo a especificação do protocolo e apresenta também o *Delta Time* simulado, que em uma rede real é o tempo decorrido desde o início do frame anterior até o início do frame atual.

É possível redimensionar, ocultar e reorganizar as colunas para facilitar a visualização de determinado serviço ou característica de algum frame. As colunas da tabela, de maneira resumida,

Frame Nr	Address	Msg Type	Service	Req/Res	SAPS	Data_Unit_Len	Data	Delta Tim
36	120->1	Slave Diagnostics Response	Response FDL/FMA1/2-Data low send data ok (DL)	Response	60->62	6	680B0B6881F8083E3...	0,265 ms
37	1->121	Get Diagnostics Request	Send and Request Data high (SRD)	Request	62->60	0	68050568F9816D3C3...	0,449 ms
38	121->1	Slave Diagnostics Response	Response FDL/FMA1/2-Data low send data ok (DL)	Response	60->62	6	680B0B6881F9083E3...	0,265 ms
39	1->122	Get Diagnostics Request	Send and Request Data high (SRD)	Request	62->60	0	68050568FA816D3C3...	0,449 ms
40	122->1	Slave Diagnostics Response	Response FDL/FMA1/2-Data low send data ok (DL)	Response	60->62	6	680B0B6881FA083E3...	0,265 ms
41	1->123	Get Diagnostics Request	Send and Request Data high (SRD)	Request	62->60	0	68050568FB816D3C3...	0,449 ms
42	123->1	Slave Diagnostics Response	Response FDL/FMA1/2-Data low send data ok (DL)	Response	60->62	6	680B0B6881FB083E3...	0,265 ms
43	1->124	Get Diagnostics Request	Send and Request Data high (SRD)	Request	62->60	0	68050568FC816D3C3...	0,449 ms
44	124->1	Slave Diagnostics Response	Response FDL/FMA1/2-Data low send data ok (DL)	Response	60->62	6	680B0B6881FC083E3...	0,265 ms
45	1->125	Get Diagnostics Request	Send and Request Data high (SRD)	Request	62->60	0	68050568FD816D3C3...	0,449 ms
46	125->1	Slave Diagnostics Response	Response FDL/FMA1/2-Data low send data ok (DL)	Response	60->62	6	680B0B6881FD083E3...	0,265 ms
47	1->0	Request FDL Status	Request FDL-Status with Reply	Request	0->0	0	100001494A16	0,449 ms
48	1->127	Request Global Control	Not applied	Response	62->58	2	68070768FF810F3A3E...	0,532 ms
49	1->1	Token Passing	Not applied	Response	0->0	0	DC0101	0,686 ms
50	1->34	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68A2815D3D3...	0,141 ms
51	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
52	1->100	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68E4815D3D3...	0,097 ms
53	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
54	1->105	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68E9815D3D3...	0,097 ms
55	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
56	1->106	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68EA815D3D3...	0,097 ms
57	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
58	1->107	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68EB815D3D3...	0,097 ms
59	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
60	1->108	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68EC815D3D3...	0,097 ms
61	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
62	1->109	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68ED815D3D3...	0,097 ms
63	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
64	1->110	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68EE815D3D3...	0,097 ms
65	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms
66	1->111	Set Parameter Request	Send and Request Data high (SRD)	Request	62->61	7	680C0C68EF815D3D3...	0,097 ms
67	0->0	No Function	Not applied	Response	0->0	0	E5	0,419 ms

Figura 4.11: Resultado da simulação apresentado em tabela de frames.

respresentam os dados a seguir:

1. Frame Nr: Ordem sequencial do frame gerado na simulação

2. Address: Apresenta, através da decodificação do frame, o frame de origem e destino representados pelo símbolo ('->').

3. Msg Type: Serviço decodificado através do SAP de origem e SAP de destino. Veja Tabela 2.1 que mostra os Serviços e SAPs relacionados.

4. Service: Decodificação do FC (Frame Control) detalhando se o frame é de requisição ou resposta. Além dos bits de prioridade.

5. Req_Res: Coluna para consulta rápida se o frame é uma requisição ou resposta.

6. SAPS: Apresenta os códigos dos serviços contidos no frame, caso aplicável.

7. Data_Unit_Len: Tamanho do campo de dados úteis do frame, excluindo os cabeçalhos e trailers.

8. Data: *Frame raw data* - apresenta o frame completo na forma de bytes, em hexadecimal, sem decodificação.

9. Delta Time: Tempo decorrido entre o início do último frame e o início do atual.

No Capítulo 5 será apresentada a metodologia e resultados dos testes realizados com objetivo de testar as características apresentadas neste capítulo.

Capítulo 5

Resultados

Buscando a validação do simulador em relação a uma rede real, foram montadas algumas configurações de rede na prática, com equipamentos reais, visando à comparação com casos práticos. Os mesmos ensaios realizados na rede real foram repetidos para a rede no simulador.

Primeiramente apresenta-se a arquitetura da rede real e as configurações de tempos atribuídas para obtenção de dados reais. Em seguida, executa-se o simulador e os resultados para cada um dos critérios de teste são apresentados e comentados.

Como critério de análise, os itens a seguir foram usados como parâmetros de avaliação dos resultados produzidos pelo simulador:

1. Exatidão dos Frames: Verificação se os frames simulados obedecem ao formato normatizado do protocolo.

2. Funcionamento conforme o Protocolo: Mestre e escravos devem seguir as regras do protocolo e sequencias de mensagens corretas devem ser observadas segundo as maquinas de estado das estações definidas na norma.

3. Exatidão Temporal: Os intervalos de tempo entre os frames devem se aproximar dos intervalos entre frames reais.

4. Simulação de Problemas de Rede: Selecionar e introduzir problemas de rede disponíveis na ferramenta e verificar o resultado.

5.1 Arquitetura da rede real e configurações de tempo utilizadas

A Figura 5.1 mostra como foi realizada a montagem da rede real. A quantidade de estações utilizadas foi de 24 equipamentos, sendo 1 mestre e 23 escravos. O conjunto de endereços utilizado foi intencionalmente preservado.

O controlador (Mestre Profibus DP) utilizado foi o DF73, desenvolvido pela SMAR Equipamentos Industriais. O *stack* de comunicação deste mestre é da empresa *Hilscher Gesellschaft für Systemautomation mbH*. O configurador utilizado é de origem da mesma empresa.

As taxas de comunicação selecionas foram do intervalo de 9.600bps à 1.500Kbps. Estas taxas são as mais utilizadas devido à distância proporcionada (veja Tabela 2.2) e para evitar problemas com o acoplador DP/PA acima destas taxas.

Cada ensaio com a rede real foi realizado da seguinte maneira:

1. A ferramenta de captura e o computador foram disparados para iniciar a captura em uma dada taxa de comunicação referente à configuração.
2. O download da configuração é realizado no mestre.
3. Após a rede entrar em regime (todos os escravos em troca de dados), o endereço 34 foi desligado e ligado novamente. Simulando uma falha momentânea para este endereço.
4. Após o retorno do endereço 34, a captura ferramenta de captura foi finalizada e o trafego ocorrido foi salvo.
5. Repetiu-se este procedimento para diferentes configurações de tempo (casos).

O endereço 34 se trata de uma estação remota WAGO, que é um escravo com estradas e saídas sendo digitais e analógicas. Os demais equipamentos são transmissores de temperatura e pressão (TT303 e LD303), posicionadores de válvulas (FY303), conversores de corrente (IF303) e outros cuja função é adicionar carga (tráfego) na rede para análise.

Os demais escravos foram adicionados sob o gateway DP/PA cuja função é fazer a conexão da rede Profibus DP com a rede Profibus PA. O modelo utilizado P+F permite que o acesso aos equipamentos Profibus PA seja realizado de maneira transparente para o mestre.

A Tabela 5.1 mostra a quantidade de dados que será trocada entre o mestre e cada um dos equipamentos durante a comunicação cíclica. Para obter o mesmo efeito no simulador, os

identificadores de configuração foram incluídos no arquivo de configuração do mestre referente a cada um dos endereços.

O ponto de vista relacionado a entrada e saída quantificado na tabela é em relação ao mestre. Por exemplo, o endereço 100 (FY303) possui 5 bytes de saída - isso indica que este escravo deve receber 5 bytes do mestre durante a troca de dados e estes serão atribuídos a uma posição de válvula ou outra atuação. No caso do endereço 107 (TT303) deve fornecer 10 bytes de dados para o mestre (10 bytes de entrada), relacionados a medição de temperatura em 2 canais.

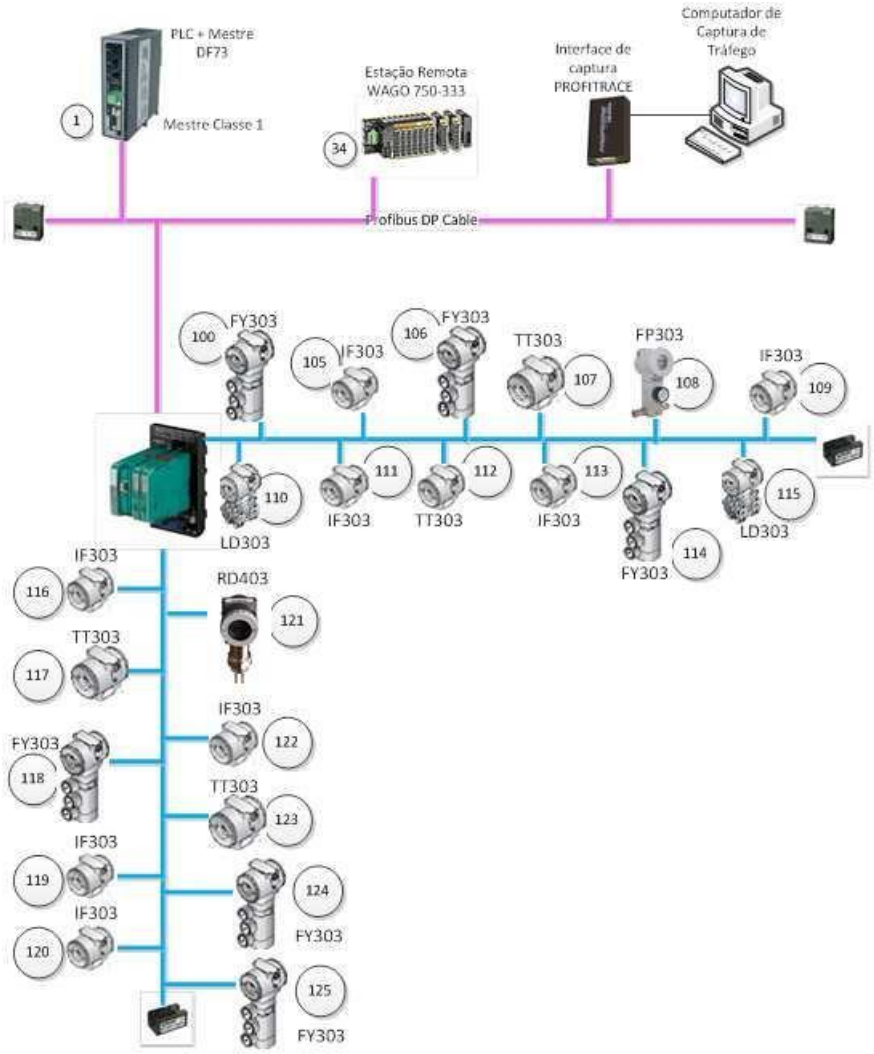

Figura 5.1: Representação do sistema real.

Tabcla 5.1: Configuração do mcstrc rclativo ao sistcma rcal.

Address	TAG	Ident Number	Bytes In	Bytes Out	Identifier
001	Master	-	-	-	<não aplicado>
034	Slave34	0xB754	5	5	0x0010205161
100	Slave100	0x0897	0	5	0xA4
105	Slave105	0x0896	30	0	0x9494944184854184854184854
106	Slave106	0x0897	0	5	0xA4
107	Slave107	0x089A	10	0	0x9494
108	Slave108	0x0898	0	5	0xA4
109	Slave109	0x0896	30	0	0x9494944184854184854184854
110	Slave110	0x0895	10	0	0x94418485
111	Slave111	0x0896	30	0	0x9494944184854184854184854
112	Slave112	0x089A	10	0	0x9494
113	Slave113	0x0896	30	0	0x9494944184854184854184854
114	Slave114	0x0897	0	5	0xA4
115	Slave115	0x0895	10	0	0x94418485
116	Slave116	0x0896	30	0	0x9494944184854184854184854
117	Slave117	0x089A	10	0	0x9494
118	Slave118	0x0897	0	5	0xA4
119	Slave119	0x0896	30	0	0x9494944184854184854184854
120	Slave120	0x0896	30	0	0x9494944184854184854184854
121	Slave121	0x0904	10	0	0x94418485
122	Slave122	0x0896	30	0	0x9494944184854184854184854
123	Slave123	0x089A	10	0	0x9494
124	Slave124	0x0897	0	5	0xA4
125	Slave125	0x0897	7	5	0xC68486080508050505

5.2 Metodologia para obtenção dos dados da rede real

Conforme detalhado anteriormente, os ensaios compreendem a configuração do mestre, captura dos frames da rede real, simulação de uma falha e captura dos frames durante a falha.

A captura dos frames foi realizada utilizando-se a ferramenta Profitrace. Desenvolvida pelo fabricante PROCENTEC - Figura 5.2. Esta ferramenta conta com uma interface externa dotada de uma memória de armazenamento, garantindo a amostragem de todos os frames em transito na rede, além da atribuição de uma estampa de tempo em até microssegundos, de maneira independente em relação ao computador.

É considerada uma ferramenta consolidada no mercado e também utilizada para esta finalidade: captura e avaliação temporal de frames Profibus. As capturas podem ser persistidas (salvas em disco), para análise posterior - incluindo *timestamps* de cada frame capturado.

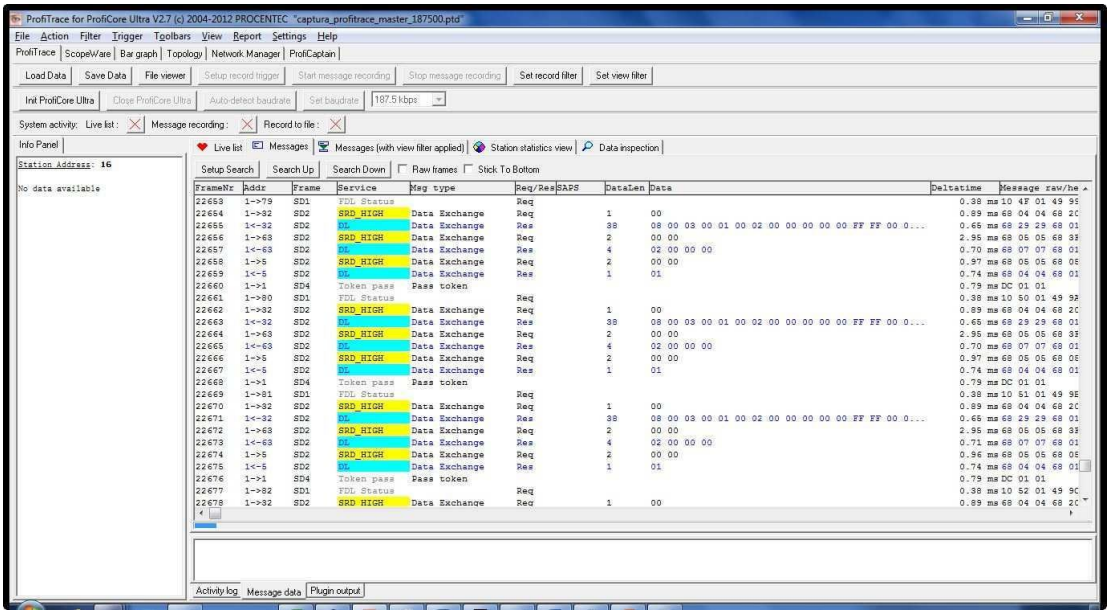

Figura 5.2: Ferramenta Profitrace - PROCENTEC.

5.3 Exatidão dos Frames

Foram selecionados alguns frames, visando verificar se foram montados adequadamente. O frame simulado é comparado com um frame de mesmo tipo gerado na captura real.

Os frames comparados foram os seguintes:

- Slave Diagnostics.
- Request FDL Status.
- Set Parameter.
- Check Configuration.
- Data Exchange.
- Set Global Control.
- Token.
- Short Ack.

5.3.1 Frame Slave Diagnostics

Na Figura 5.3 observa-se a tela da ferramenta de captura realçando o frame Get Diagnostics para o endereço 34 e a resposta do mesmo durante o período de configuração. Logo em seguida há a comparação com o simulador.

```
1->34 Get Diagnostics  Req  62  60  0  68 05 05 68 A2 81 5D 3C 3E FA 16
1<-34 Get Diagnostics  Res  60  62  6  68 0B 0B 68 81 A2 08 3E 3C 00 0C 00 01 B7 54 BD 16
```

Figura 5.3: Realce do frame Get Diagnostics na ferramenta de captura.

Na Figura 5.4 e na Tabela 8 obtidos com o simulador, observa-se a mesma saída encontrada na ferramenta de captura

21745	1->34	Get Diagnostics Request	68050568A2817D3C3E1A16
21746	34->1	Slave Diagnostics Response	680B0B6881A2083E3C000C0001B754BD16

Figura 5.4: Realce do frame Get Diagnostics no simulador.

Ainda no caso deste frame, como se trata de um serviço de aplicação podemos obter sua decodificação na ferramenta de captura e no simulador, comparando ambas. A Figura 5.5 mostra a decodificação na ferramenta de captura e a Figura 5.6 mostra a decodificação realizada no simulador.

Figura 5.5: Decodificação do frame Get Diagnostics na ferramenta de captura.

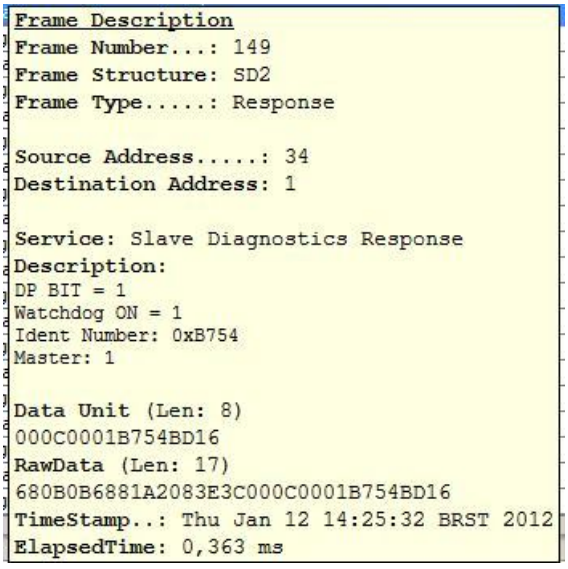

Figura 5.6: Decodificação do frame Get Diagnostics no simulador.

Para os demais frames, o resultado também foi coincidente entre o simulador e a ferramenta de captura. Era esperado, pois a metodologia do teste é igual a utilizada para o frame de diagnóstico.

5.3.2 Análise dos Resultados

Observamos pelos resultados que o Simulador reproduz exatamente os frames reais, calcula os valores de *CheckSum* corretamente e inclui ele e o *End Delimiter* (16H) somente para os frames onde isso se aplica (não se aplica ao token e short ack.).

Este é um requisito mínimo que foi atingido ainda durante a qualificação para possibilitar o progresso em demais funções no simulador.

5.4 Funcionamento conforme o protocolo

Para o mesmo endereço 5, na Figura 5.7 destaca-se o processo de configuração conforme exigido pelo protocolo, que segue conforme o protocolo definido na camada de aplicação.

Frame Nr	Address	Msg Type	Data	
1	1->5	Get Diagnostics Request	6805056885816D3C3EED16	1
2	5->1	Slave Diagnostics Response	680B0B688185083E3C020500FF80E0EE16	
3	1->0	Request FDL Status	100001494A16	
4	1->127	Request Global Control	68070768FF810F3A3E00000716	
5	1->1	Token Passing	DC0101	
6	1->5	Set Parameter Request	680C0C6885815D3D3E88FF010B80E000D116	2
7	0->0	No Function	E5	
8	1->2	Request FDL Status	100201494C16	
9	1->127	Request Global Control	68070768FF810F3A3E00000716	
10	1->1	Token Passing	DC0101	
11	1->5	Check Configuration Request	6808086885817D3E3E1020204F16	3
12	0->0	No Function	E5	
13	1->3	Request FDL Status	100301494D16	
14	1->127	Request Global Control	68070768FF810F3A3E00000716	
15	1->1	Token Passing	DC0101	
16	1->5	Get Diagnostics Request	6805056885815D3C3EDD16	4
17	5->1	Slave Diagnostics Response	680B0B688185083E3C000C000180E0F516	
18	1->4	Request FDL Status	100401494E16	
19	1->127	Request Global Control	68070768FF810F3A3E00000716	
20	1->1	Token Passing	DC0101	
21	1->5	Data Exchange	6805056805017D00008316	5
22	5->1	Data Exchange	6804046801050800 0E16	

Figura 5.7: Processo de Parametrização e Troca de dados com o endereço 5.

A estação 5 foi incluída na configuração. Para os equipamentos em configuração o mestre realiza necessariamente o comando *Get Diagnostics*, independentemente se a FDL já marcou a estação como operacional ou não. Este é um mecanismo de otimização de partida.

No ciclo (1), o mestre enviou um comando de *Get Diagnostics* para verificar a condição da estação 5. Esta respondeu dizendo que estava aguardando parametrização e configuração.

Em seguida nos ciclos (2) e (3) o mestre enviou os comandos de *Set Parameter e Check Configuration* para a estação 5. Esta apenas respondeu com um E5 (*Short Acknowledgement*). Somente no ciclo (4) o mestre enviou um novo comando de *Get Diagnostics* verificando na resposta que a estação estava pronta para a troca de dados.

No ciclo (5), após o mestre verificar que a estação estava pronta desde o ciclo anterior, ele começa o processo de troca de dados enviando um comando *Data Exchange* ao endereço 5. Este estado deverá perdurar até que o escravo pare de responder o comando de *Data Exchange* ou o mestre retire o escravo deste processo.

5.4.1 Análise dos Resultados

O simulador reproduz o ciclo completo de configuração de um escravo Profibus, reproduzindo a máquina de estados do escravo. Este teste também mostra que a máquina de estados do mestre também está funcional.

Nesta Seção destacou-se a configuração para um escravo, porém o teste realizado com a configuração real também foi concluído com sucesso, cumprindo com o ciclo completo de configuração de todos os escravos.

5.5 Simulação de Problemas de Rede

5.5.1 Equipamento Trocado - Ident Number errado

O processo mostrado anteriormente, apresenta a sequência de comandos esperada quando o equipamento a ser configurado é o mesmo previsto na configuração.

Este é o caso normal e o usuário deve buscar adequar as configurações dos instrumentos do campo a fim de que sejam compatíveis com o projeto. Ou vice-versa caso desejado, realizando uma revisão de projeto.

A seguir, o Ident Number do equipamento será trocado para simular uma falha de troca de modelo de equipamento, isto é, o usuário deseja o equipamento de modelo X para um determinado endereço, porém, o equipamento real é do modelo Y. Neste caso, o protocolo possui uma proteção que não permite que o instrumento entre em troca de dados, por questões de segurança da planta.

Conforme visto no Capítulo 2 - Seção 2.4.2, podem haver diferentes causas para este problema de Ident Number trocado ou errado.

Antes da simulação ser iniciada, trocou-se o Ident Number do endereço 34 (valor em 0xB754) para o valor (0x0896) - Figura ??. Após essa troca, clica-se no botão Force New Diag./Config.

Figura 5.8: Detalhe aba Slave Properties - troca de Ident Number.

Após iniciar a simulação, faz-se 2 observações:

1. Endereço 34 pisca com cor vermelha na Live-List. Marca-se com esta cor, quando o diagnóstico do escravo vêm com o bit Parameter Fault em 1. Veja a Figura ??.

2. Ao se parar a simulação observa-se que o ciclo de configuração para o endereço 34 não termina, e observa-se que ele repete-se indefinidamente, pois após o escravo receber o comando Set Parameter com o Ident Number errado, ele rejeita a parametrização e marca o bit Parameter Fault no frame de Diagnóstico.

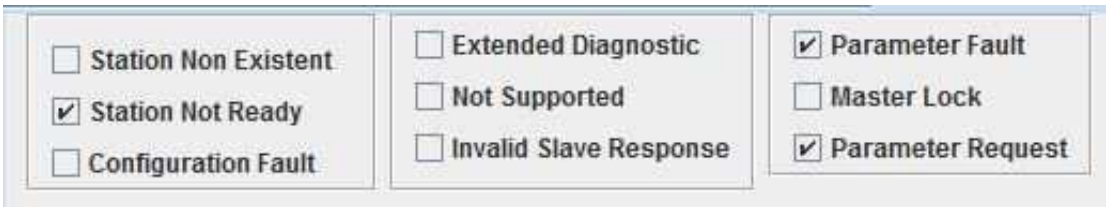

Figura 5.9: Detalhe aba Slave Properties - Parameter Fault.

Ao retornar-se o valor correto e clicar-se novamente no botão Force New Diag./Config., como o equipamento está em parametrização, no próximo frame de parametrização o Ident Number correto é recebido pelo escravo, e ele continua o processo de configuração até entrar em troca de dados.

5.5.2 Erro de Configuração dos Módulos

Neste caso, mudamos o conjunto de módulos do escravo. Tornando a quantidade ou tipo de módulos existentes no escravo diferentes dos existentes na configuração do mestre.

Como visto no Capítulo 2 - Seção 2.4.2, neste caso pode ser uma falha de um módulo do equipamento durante a operação ou a configuração errada pelo usuário.

Mesmo após a configuração estar rodando e o endereço 34 estar em troca de dados, alterou-se o conjunto de modulos (Configuration), modificando o final de (...61) para (...62) - Figura **??**. Após essa troca, clica-se no botão Force New Diag./Config.

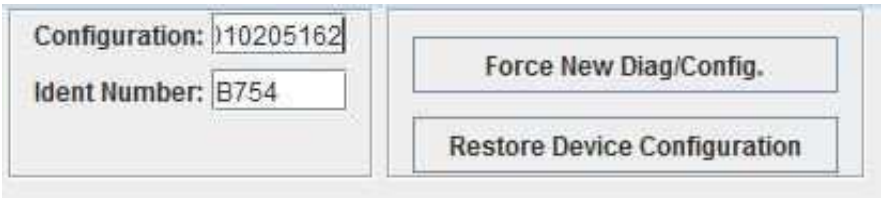

Figura 5.10: Detalhe aba Slave Properties - troca de configuração.

Após iniciar a simulação, faz-se 2 observações:

1. Endereço 34 pisca com cor roxa na Live-List. Marca-se com esta cor, quando o diagnóstico

do escravo vêm com o bit Configuration Fault em 1. Veja a Figura **??**.

2. Ao se parar a simulação observa-se que o ciclo de configuração para o endereço 34 não termina, e observa-se que ele repete-se indefinidamente, pois após o escravo receber o comando Check Configuration com os módulos errados, ele rejeita a parametrização e marca o bit Configuration Fault no frame de Diagnóstico.

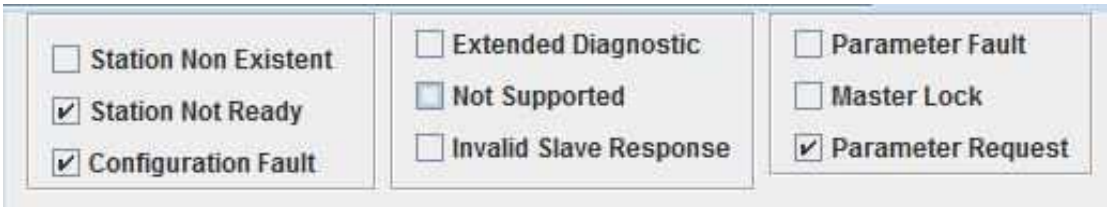

Figura 5.11: Detalhe aba Slave Properties - troca de configuração de módulos.

Ao retornar-se o valor correto e clicar-se novamente no botão Force New Diag./Config., como o equipamento está em parametrização, no próximo frame de Check Configuration o conjunto de módulos correto é recebido pelo escravo, e ele continua o processo de configuração até entrar em troca de dados.

5.5.3 Análise dos Resultados

A Seção de testes de erros de rede também faz parte da parte de funcionamento de acordo com o protocolo. Ela foi colocada a parte para destacar as possiblidades de uso do simulador.

Os testes mostram que os escravos estão realizando a validação da parametrização e configuração. O funcionamento está ocorrendo de acordo com a máquina de estados do escravo. A decodificação dos *Identifiers* também é realizada tanto para módulos comuns e módulos especiais.

5.6 Exatidão Temporal

Para cada frame gerado pelo simulador, é também gerado uma estampa de tempo que indica o momento em que o frame foi recebido pelo escravo ou pelo mestre.

No simulador é apresentado, para cada frame, a diferença entre a estampa de tempo do frame atual e do frame anterior *(Delta Time)* em milissegundos (ms). Para este intervalo atribui-se o nome de delta, para facilitar a nomenclatura.

Este intervalo de tempo obedece aos parâmetros de protocolo (definidos no Capítulo 2), dependem da configuração do mestre e da quantidade de bytes dos frames.

A ferramenta de captura, utilizada para obter os frames de uma rede real, também permite apresentar o delta, permitindo armazenar diretamente o intervalo entre frames, de maneira independente em relação ao início da captura ou início da simulação.

Para verificar se os tempos gerados no simulador correspondem com os tempos obtidos na prática, compararam-se os seguintes índices ou parâmetros de tempo para cada um dos casos para cada uma das taxas de comunicação entre 9.600bps e 1.500Kbps:

1. SlotTime: Representa o tempo que o mestre aguarda após uma requisição para enviar um novo frame quando não há resposta. É um *Time-out* em relação ao recebimento da resposta de um escravo.

2. TRR (Tempo de ciclo): representa o tempo entre 2 passagens de *token*, como estamos utilizando um único mestre, este tempo é equivalente ao tempo de reação do sistema ou tempo de atualização.

5.6.1 Slot Time

O *Slot Time* é medido em uma captura real através da leitura da coluna *Deltatime* na ferramenta *Profitrace*. Ele pode ser medido quando há 2 requisições seguidas do mestre, veja na Figura 5.12. Por exemplo, quando não há resposta de alguma estação que estava em troca de dados, o mestre aguarda o tempo do *Slot Time* para efetuar cada *retry* - exemplo 1.

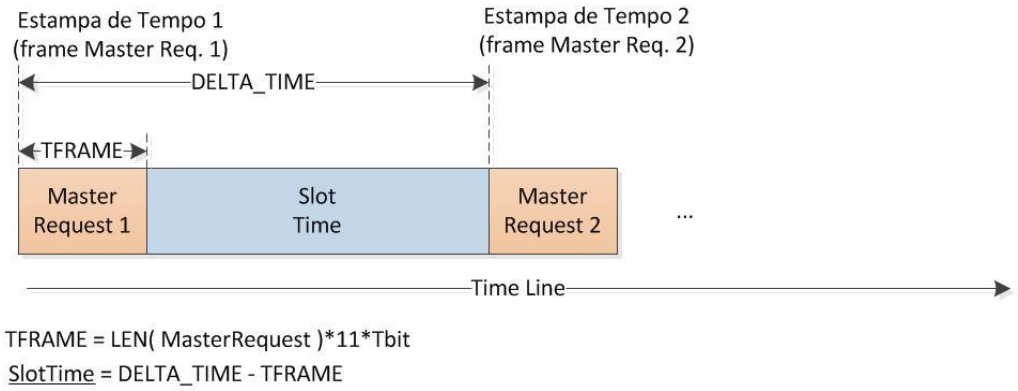

Figura 5.12: Obtenção do SlotTime através do DELTA.

No simulador, pode-se reproduzir o mesmo cenário e então comparar o valor obtido. As

figuras 5.13, 5.14, 5.15 e 5.16 mostram como foi realizada a obtenção dos dados para comparação dos valores de *Slot Time*.

Figura 5.13: Ex. 1 - Medição do DELTA real na ferramenta Profitrace (retry) - em 1500Kbps.

Um outro exemplo é o caso onde o mestre envia um comando *Request FDL Status* no processo de manutenção da rede (verificação das estações operacionais) e aguarda o período do *Slot Time* para enviar uma nova requisição - exemplo 2.

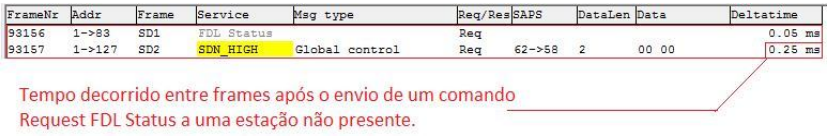

Figura 5.14: Ex. 2 - Medição do DELTA real na ferramenta Profitrace (Req. FDL Status) - em 1500Kbps.

A tabela 5.4 apresenta os resultados comparativos entre o *Slot Time* configurado, real e simulado, respectivamente medidos e obtidos para cada uma das taxas de comunicação mantendo-se a mesma configuração do mestre tanto para a rede real quanto para a rede simulada. Para composição da tabela foram utilizados os valores de delta obtidos através do *Request FDL Status* para uma estação não presente ou não operacional. Os valores de delta encontrados são mostrados na tabela 5.2. Para os valores medidos foi realizada uma amostragem aleatória. Os valores configurados e simulados não têm variação.

Figura 5.15: Ex. 1 - Obtenção do DELTA simulado na ferramenta (retry) - em 1500Kbps.

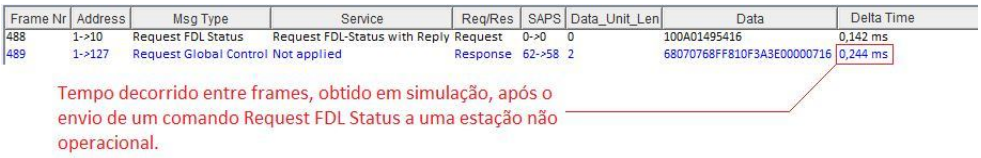

Figura 5.16: Ex. 2 - Obtenção do DELTA simulado na ferramenta (Req. FDL Status) - em 1500Kbps.

Tabela 5.2: Valores medidos (Delta REAL) e obtidos (Delta SIMULADO)

Baud Rate (bps)	Delta REAL (ms)	Delta SIMULADO (ms)
9600	17,29	17,292
19200	8,65	8,646
45450	3,63	3,652
93750	1,76	1,771
187500	0,89	0,885
500000	0,54	0,532
1500000	0,25	0,244

Com os valores de *DeltaTime* (DELTA) reais e simulados para cada uma das taxas de comunicação pode-se encontrar o valor do *Slot Time* subtraindo-se o tempo ocupado pelo frame do valor de delta, obtendo-se somente o tempo restante, que compreende o *Slot Time*.

Como o frame é o *Request FDL Status* (6 bytes de extensão) e sabe-se que cada byte ocupa 11 bits devido ao *start e stop bit* e o bit de paridade então temos que o total de bits utilizados pelo frame de requisição é de 66 bits. Para cada taxa de comunicação temos um tempo de ocupação (diferentes valores de Tbit). A tabela 5.3 mostra o tempo de ocupação referente a cada taxa de comunicação.

Tabela 5.3: Valores de tempo ocupado pelo frame Request FDL Status

Baud Rate (bps)	Ocupação (TFRAME) (ms)
9600	6,88
19200	3,44
45450	1,45
93750	0,70
187500	0,35
500000	0,13
1500000	0,04

As colunas SlotTime Real e SlotTime Sim. são resultado dos valores obtidos na tabela

5.2 subtraindo-se de ambos o valor da ocupação do frame de requisição referente a cada taxa, encontrado na tabela 5.3.

Tabela 5.4: Tabela comparativa de SlotTime real e simulado.

Baud Rate (bps)	SlotTime conf. (Tbit)	SlotTime conf. (ms)	SlotTime Real (ms)	SlotTime Sim. (ms)	|Dif.| (ms)	|Dif.|/Real (%)
9600	100	10,42	10,42	10,417	0,002	0
19200	100	5,21	5,21	5,209	0,004	0
45450	100	2,20	2,18	2,200	0,022	1
93750	100	1,07	1,06	1,067	0,011	1
187500	100	0,53	0,54	0,533	0,005	1
500000	200	0,40	0,41	0,400	0,008	2
1500000	300	0,20	0,21	0,200	0,006	3

As Figuras 5.22 e 5.23 a seguir representam graficamente a Tabela 5.3.

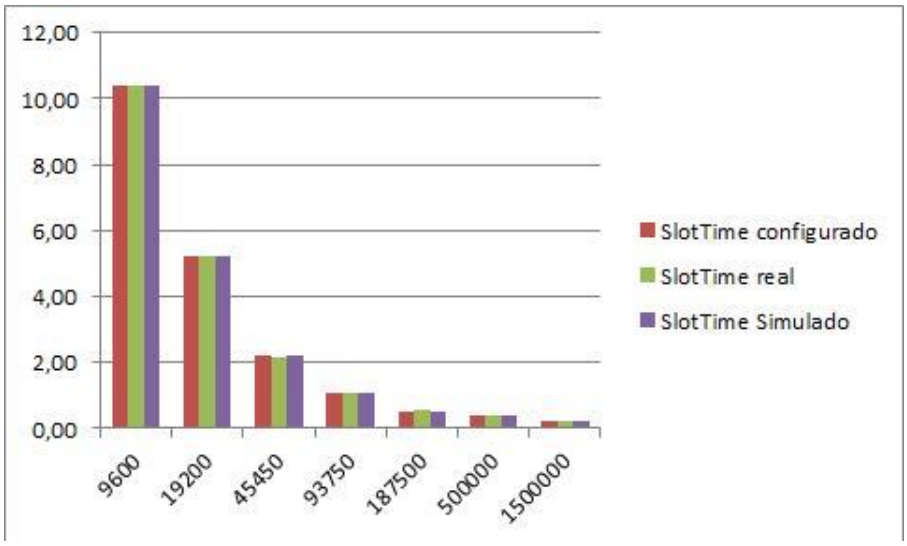

Figura 5.17: Gráfico do Slot Time (ms) x Baud Rate (bps)

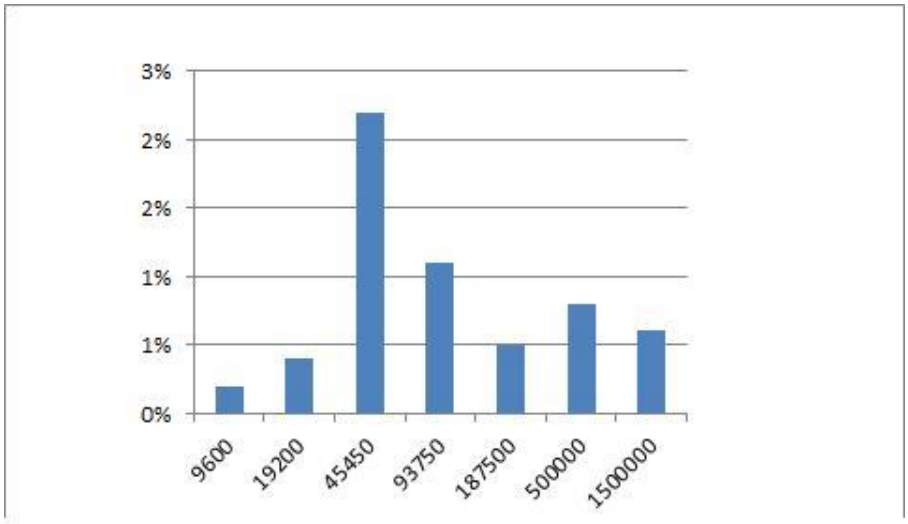

Figura 5.18: Erro (%) para o Slot Time Real x Baud Rate (bps)

5.6.2 Tempo de Ciclo

O tempo de ciclo pode ser medido através do intervalo entre 2 trocas de dados a um determinado instrumento, ou caso exista somente um mestre, ele pode ser obtido através da medição da diferença das estampas de tempo entre 2 passagens de *token*, veja a Figura 5.19. O tempo de ciclo, para uma rede configurada e instalada corretamente tende a se manter em um valor fixo.

O tempo de ciclo depende da configuração de tempos e taxa de comunicação utilizados pelo mestre, da lista de instrumentos, da quantidade de bytes que é trocada entre o mestre e os escravos. Este tempo pode sofrer variações caso a rede sofra perturbações, como entrada e saída de estações e modificação da configuração de estações durante a troca de dados.

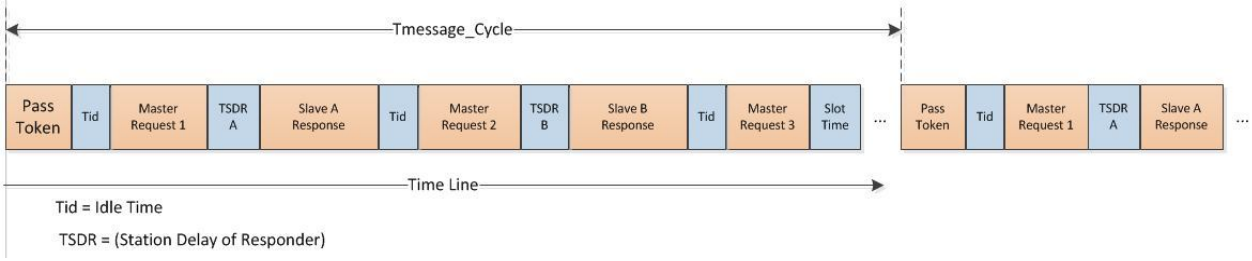

Figura 5.19: Medição e obtenção do tempo de ciclo.

Neste trabalho o tempo de ciclo real foi medido para as 2 configurações de rede. Ele é obtido através da ferramenta de captura, na aba *Station statistics view*, veja na Figura 5.20. Para medição, foi tomada uma amostra aleatória de um dos escravos. Para simulação, o valor é constante.

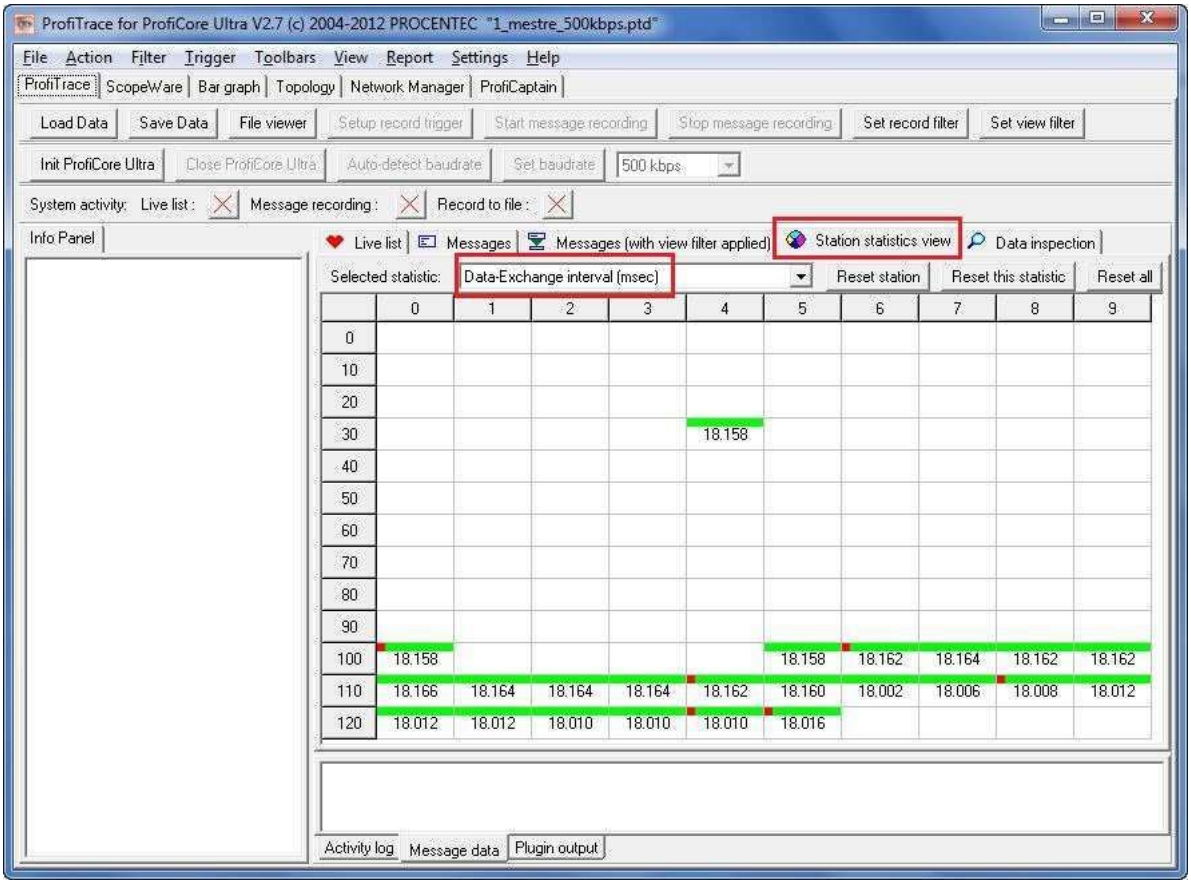

Figura 5.20: Detalhe do software de captura para medição do tempo de ciclo.

No simulador, o tempo de ciclo é obtido através do somatório do valor de Delta Time de cada frame compreendido dentro de um ciclo de Token. Este cálculo é realizado pelo próprio simulador (continuamente) e apresentado na aba *Simulation Statistics*, chamado *TMessageCycle*. Observando a figura 5.21 nota-se que o Simulador calcula o valor instantâneo e o pior caso (mais lento), isso é útil para fazer um planejamento sobre possíveis falhas na rede, mantendo o tempo de ciclo dentro dos requisitos de projeto.

Para a configuração 1 (23 instrumentos - total 322 bytes de entrada e 40 bytes de saída) a tabela 5.5 mostra a comparação entre o tempo de ciclo medido e o tempo de ciclo obtido através da simulação das mesmas configurações.

As Figuras ?? e ?? a seguir representam graficamente a Tabela 5.5.

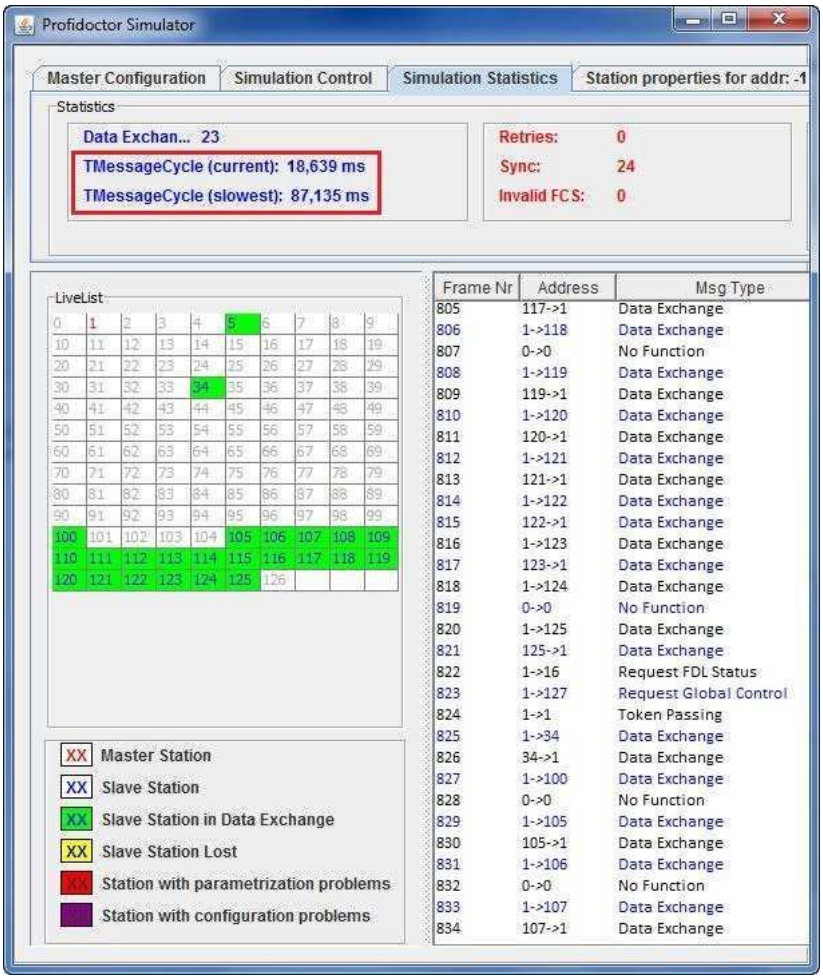

Figura 5.21: Aba do simulador onde se obtém o tempo de ciclo.

Tabela 5.5: Tabela comparativa de Tempo de ciclo real e simulado

Baud Rate (bps)	TCiclo Real (ms)	TCiclo Simulado (ms)	|Dif.| (ms)	|Dif./Real| (%)
9600	932,5	949,9	17,4	2
19200	466,5	474,9	8,4	2
45450	198,5	200,65	2,15	1
93750	93,8	97,3	3,5	4
187500	47,4	48,6	1,2	3
500000	18,2	18,6	0,4	2
1500000	6,66	6,35	0,31	5

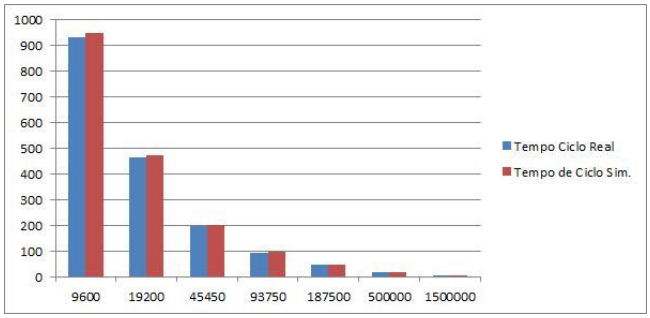

Figura 5.22: Gráfico do Slot Time (ms) x Baud Rate (bps).

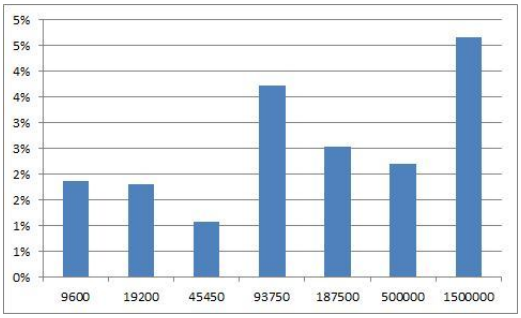

Figura 5.23: Erro(%) para o Tempo de Ciclo Real (ms) x Baud Rate (bps).

5.6.3 Análise dos Resultados

A validação temporal transmite a viabilidade de uso do simulador para análises quantitativas de tempo em relação à diferentes configurações, permitindo uma possível validação de projeto de arquitetura de automação quanto ao tempo de resposta projetado e validação em relação à erros de configuração realizados pelo usuário.

Na aba de estatísticas (Simulation Statistics), pode se obter além do tempo de ciclo, a validação de demais parâmetros de configuração em relação ao TRR - Veja na Figura 5.24, como por exemplo se o TWdg está configurado de maneira adequada e se o TTR está com uma margem muito reduzida em relação ao TRR.

Figura 5.24: Aba de estatísticas para uso em validação de configuração

Capítulo 6

Conclusões

Este software permite explorar o funcionamento de uma rede Profibus DP através de simulações de diferentes configurações de rede. A flexibilidade proporcionada pelo arquivo de configuração permite avaliar o desempenho das redes segundo a variação dos equipamentos na configuração do mestre e dos parâmetros de rede. O uso de um arquivo baseado em um configurador de mercado permite comparar imediatamente o comportamento simulado do comportamento real.

Observou-se, durante o desenvolvimento do trabalho, que grande parte do tempo em foi destinado ao desenvolvimento de infraestrutura para comunicação, montagem de frames e simulação até possibilitar alguma obtenção de resultados.

Este trabalho propõe um recurso didático interessante, que permite auxiliar a formação de usuários no campo de redes industriais com foco específico em Profibus. O formato da interface do simulador semelhante à uma ferramenta de captura de mercado compartilha o aprendizado de uma ferramenta com a outra.

Foi realizada a variação quanto à exatidão dos frames, funcionamento do protocolo e a exatidão temporal obtendo erros máximos de 5% em relação aos valores reais.

A possibilidade de incluir erros de configuração e falha de equipamentos permite simular casos reais e verificar, por exemplo, qual a variação do tempo de reação do sistema para um determinado cenário de falha de instrumentos e auxiliar o serviço técnico de projetos a dimensionar TTR, Watchdog Time e outros parâmetros de tempo explicados neste trabalho e presentes no simulador.

O simulador já permite carregar configurações de plantas reais, não necessitando dos GSDs utilizados, pois já obtem todos os dados necessários da configuração.

Observa-se ainda, no mercado, uma variação quanto ao comportamento dos mestres Profibus DP não obedecer a norma. Isso ocorre devido à não obrigatoriedade na certificação de Mestres Profibus DP. Acredita-se que com a evolução da norma e devido ao crescimento do número de fabricantes e usuários, um processo de certificação do mestres será indispensável para evitar futuros problemas de interoperabilidade.

Um módulo de comunicação com rede real foi desenvolvido prematuramente para se comunicaçar como um mestre classe 1, com comandos selecionáveis pelo usuário para permitir estudos e acesso à equipamentos reais. Esse módulo não foi abordado neste trabalho porque seu detalhamento estava fora do escopo de simulação.

Como trabalho futuro, sugere-se a implementação de um simulador de escravos que possa ser colocado em uma rede real para que se possa avaliar problemas de controle oriundos de falhas de escravos.

Referências Bibliográficas

ANTONOPOULOS, C., V. KAPSALIS, e S. KOUBIAS (2004). An efficient simulator for the profibus-dp mac-layer protocol. In *Conference on Manufacturing, Modelling, Management and Control (MiM04) Conference*, Athens, Greece.

CARVALHO, J., A. CARVALHO, e P. PORTUGAL (2005). Assessment of profibus networks using a fault injection framework. In *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, Volume 1, pp. 9 pp.–423.

CASANOVA, V., J. J. SALT, A. CUENCA, e V. MASCAROS (2006). Networked control systems over profibus-dp: Simulation model. In *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, pp. 1337–1342.

HE, F. e K. GUO (2008). Modeling and simulation of profibus-dp network control system. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pp. 1141–1146.

HOSSEINPOUR, F. e H. HAJIHOSSEINI (2009). Importance of simulation in manufacturing.

HUANG, Y. (2005). Time characteristics of profibus on windows xp.

Klang, H. e A. Lindholm (2005). Modelling and simulation of a gas turbine.

MCGREGOR, I. (2002). The relationship between simulation and emulation. In *Simulation Conference, 2002. Proceedings of the Winter*, Volume 2, pp. 1683–1688 vol.2.

MOSSIN, R. C. d. (2012). *Diagnostico automático de redes Profibus.* Tese de Doutorado, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

NARDUCI, F., R. TORRES, e D. BRANDAO (2010). Desenvolvimento de um mestre classe

2 pertencente a uma ferramenta de diagnostico em redes profibus. In *Industry Applications (INDUSCON), 2010 9th IEEE/IAS International Conference on*, pp. 1–6.

PROFIBUS (1998). Normative parts of profibus -fms, -dp, -pa according to the european standard en 50 170 volume 2. Technical report, PROFIBUS Nutzerorganisation e.V. (PNO).

PROFIBUS (2010). Profibus system description - technology and application. Technical report, PROFIBUS Nutzerorganisation e.V. (PNO).

SOUZA, R. C. d. (2012). Diagnostico da camada fisica de redes profibus dp baseado em redes neurais artificiais. Dissertação de Mestrado, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

TORRES, R., F. NARDUCI, E. MOSSIN, e D. BRANDAO (2010). Analise do periodo de aquisicao de uma variavel de processo atraves de uma rede profibus dp: Comparacao entre caso real e simulado. In *Industry Applications (INDUSCON), 2010 9th IEEE/IAS International Conference on*, pp. 1–5.

Apêndice A

Telas do Simulador Profibus DP

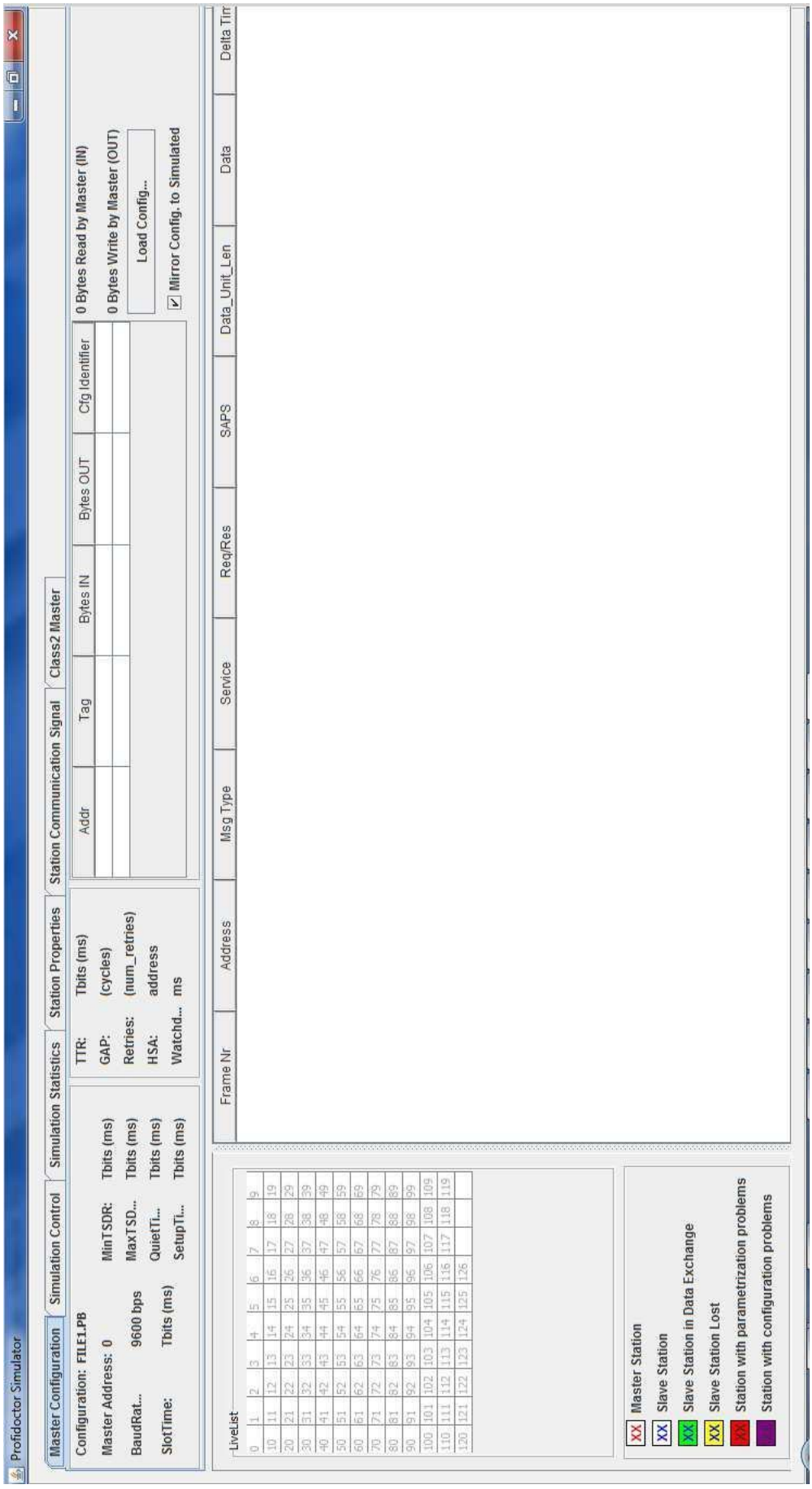

Figura A.1: Tela Inicial do Simulador

Figura A.2: Aba de Controle do Simulador

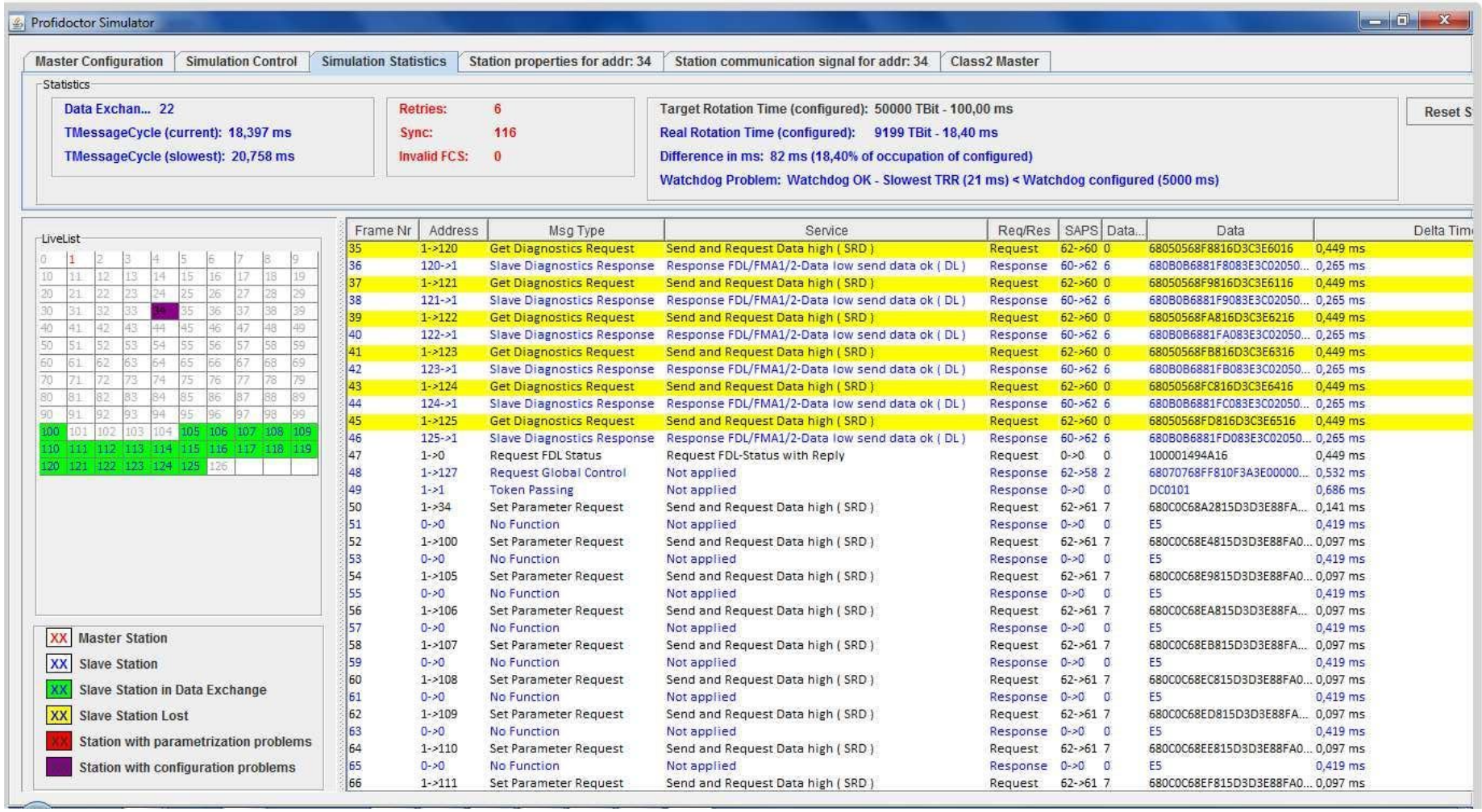

Figura A.3: Aba de Estatísticas do Simulador

Figura A.4: Aba de Diagnóstico e Configuração do Escravo

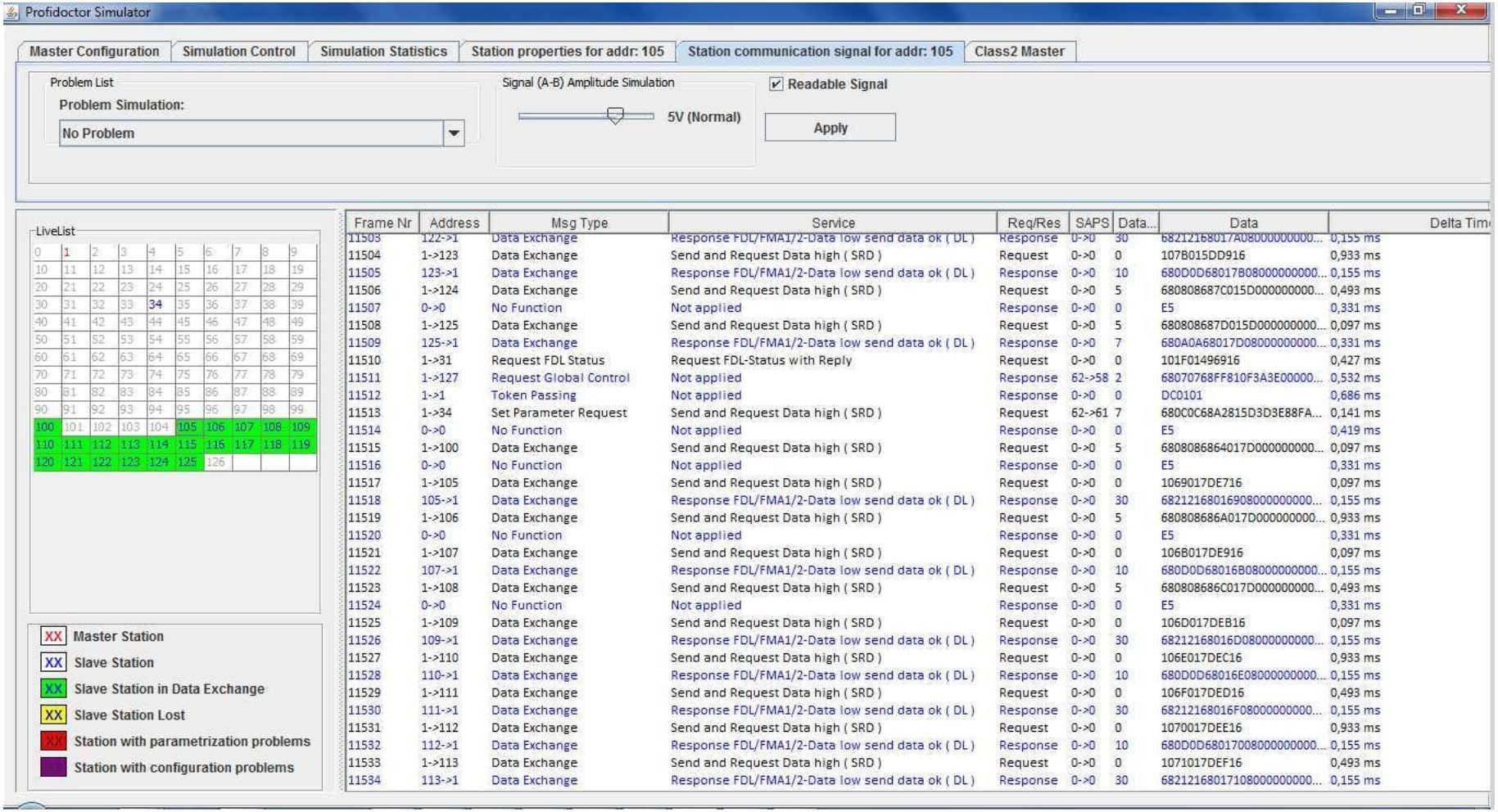

Figura A.5: Aba de modificações no sinal de comunicação