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

VICTORIA RAMOS PIRES

Análise dos protocolos de comunicação segura para automação industrial com enfoque na avaliação de desempenho do PROFIsafe

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientadora: Profa. Dra. Taisy Silva Weber
Co-orientador: João de Moraes

Porto Alegre
2017

AGRADECIMENTOS

Gostaria de agradecer, em especial, às pessoas que são primordiais na minha vida, que sempre se empenharam e fizeram de tudo para que eu pudesse realizar todos os meus sonhos: o meu imenso muito obrigada aos meus pais e melhores amigos, Maria Inês Ramos Pires e Édison Araújo Pires.

Agradeço também à pessoa que faz de tudo por mim, que se desdobra em mil para me ajudar nas minhas correrias, e que é fundamental na minha vida, muito obrigada minha irmã, minha melhor amiga, Maytê Ramos Pires.

O meu muito obrigada também a uma das pessoas que mais tem paciência comigo. Obrigada pela compreensão, companheirismo e por todo amor. Muito obrigada pelo apoio incondicional, meu amor, Denis Duarte Ramos.

Gostaria de agradecer também aos meus orientadores, Taisy Weber e João de Moraes, por toda dedicação, apoio e discernimento em todos os momentos. Obrigada por todo o conhecimento e pela experiência que me foram passados durante este processo. Obrigada também aos professores João César Netto e Sérgio Cechin pelos esclarecimentos e ajuda no entendimento do trabalho.

Gostaria de ressaltar, também, a importância da parceria com a empresa ALTUS, que foi fundamental no desenvolvimento do meu trabalho. Muito obrigada pelo apoio e pelos equipamentos que me foram disponibilizados.

Obrigada aos colegas que fizeram parte dessa minha jornada. Em especial a Luiz Gustavo Gomes, Matheus Maia, Jéssica Daltrozo, Eduardo Duarte e Tiago Dreon.

Por fim, gostaria de agradecer à UFRGS por tudo que me proporcionou, por todas as oportunidades que me foram dadas, sendo uma universidade exemplar que tem papel fundamental na minha formação, o que me tornou a profissional que sou hoje.

RESUMO

O objetivo deste trabalho foi de realizar uma análise de dois protocolos de comunicação segura estudados de uma série de protocolos existentes no mercado. Como a garantia do funcionamento seguro de funções de segurança é imprescindível dentro de qualquer indústria, tais protocolos têm certificação exigida por órgãos reguladores, que comprovem que os mesmos atendem aos níveis seguros de integridade SIL (Safety Integrity Levels) de acordo com a norma de segurança funcional IEC 61508. Estão descritas todas as características, vantagens e limitações desta série de protocolos, ressaltando as definições das *stacks* dos protocolos escolhidos: openSAFETY e PROFIsafe. Ao final deste trabalho foi desenvolvida uma parte experimental mediante o uso de módulos cedidos pela parceria com a empresa Altus, o que possibilitou a simulação do protocolo PROFIsafe em um ambiente de automação industrial, demonstrando uma análise de desempenho do protocolo acerca dos tempos de resposta em uma comunicação *safety*.

Palavras-chave: Protocolos de comunicação. Certificação. Norma de segurança. Níveis seguros de integridade. Desempenho.

Analysis of safety communication protocols for industrial automation focusing on the performance evaluation of PROFIsafe

ABSTRACT

The objective of this work is to carry out an analysis of two safety communication protocols studied from a series of protocols in the market. As the guarantee of the safe operation of safety functions is essential within any industry, such protocols are required by regulatory bodies to prove that they meet the safety integrity levels SIL (Safety Integrity Levels) according to the safety standard Functional IEC 61508. All the characteristics, advantages and limitations of this series of protocols are described, highlighting the definitions of the stacks of the protocols chosen: openSAFETY and PROFIsafe. At the end of this work, an experimental part was developed through the use of modules provided by the partnership with the company Altus, which allowed the simulation of the PROFIsafe protocol in an industrial automation environment, demonstrating a protocol performance analysis on response times in safety communication.

Keywords: Communication Protocols. Certification. Safety Industrial Standard. Safety Integrity Levels. Performance.

LISTA DE FIGURAS

Figura 2.1 – Conceito de Black Channel....12
Figura 3.1 – Quadro FSoE....17
Figura 3.2 – Modelo de referência do OpenSAFETY....18
Figura 3.3 – Roteamento das mensagens....19
Figura 3.4 – Modelo estrutural do perfil de comunicação CIP Safety on Sercos....20
Figura 4.1 – Estrutura da pilha do protocolo openSAFETY....26
Figura 5.1 – Formato de uma mensagem PROFIsafe....33
Figura 5.2 – Fluxo da mensagem PROFIsafe....35
Figura 6.1 – Arquitetura do modelo de comunicação PROFIsafe....39
Figura 6.2 – Modelo de comunicação PROFIsafe proposto....39
Figura 6.3 – Bastidor contendo o mestre da comunicação....42
Figura 6.4 – Módulos escravos da comunicação....42
Figura 6.5 – Arquitetura completa....43
Figura 6.6 – Arquitetura modelada em *software*....44

LISTA DE TABELAS

Tabela 3.1 – Medidas de segurança para cada protocolo conforme erro detectado 21
Tabela 3.2 – Demais critérios para a escolha de um protocolo *safety* 23
Tabela 4.1 – Funções principais da pilha openSAFETY 29
Tabela 4.2 – Funções recomendadas da pilha openSAFETY 30
Tabela 5.1 – Funções principais da pilha PROFIsafe 36
Tabela 5.2 – Funções do *driver* PROFIsafe a serem implementadas 37
Tabela 6.1 – *Set up* para a análise do *baud rate* 45
Tabela 6.2 – Análise da taxa de *Baud* em relação ao número de escravos 46
Tabela 6.3 – *Set up* da análise dos tempos de resposta 47
Tabela 6.4 – Análise dos tempos de resposta em relação aos tempos de ciclo 48

LISTA DE ABREVIATURAS E SIGLAS

BSD	Berkeley Software Distribution
B&R	Bernercker + Rainer Industrial Automation
CIP Safety	Common Industrial Protocol Safety
CRC	Cyclic Redundancy Check
EPL Safety	Ethernet Powerlink Safety
EPSG	Ethernet Powerlink Standardisation Group
FSoE	FailSafe-over-EtherCAT
GSD	General Station Description
IEC	International Electrotechnical Commission
ODVA	Open Device Vendors Association
PROFIsafe	PROFINET safety
PST	Process Safety Time
SCFM	Safety Control Flow Monitoring
SCM	openSAFETY Configuration Manager
SD	Safety Domain
SDN	Safety Domain Number
SERR	Safety Error
SIL	Safety Integrity Levels
SIS	Sistemas Instrumentados de Segurança
SMP	Sercos Messaging Protocol
SN	openSAFETY Node
SNMTM	Safety Network Management Master
SNMTS	Safety Network Management Slave
SPDO	Safety Process Data Objects
SFS	Safety Frame Serialization
SOD	Safety Object Dictionary
SS	Safety Slave
SSC	Safety Stack Control
SSDOC	Safety Service Data Object Client
SSDOS	Safety Service Data Object Server
UDID	openSAFETY Unique Device Identification

SUMÁRIO

1 INTRODUÇÃO 9
1.1 Motivação 9
1.2 Estrutura do trabalho 10
2 SAFETY 11
2.1 Conceito 11
2.2 Black Channel 11
2.3 Norma de segurança 12
2.4 Certificação 13
3 PROTOCOLOS DE COMUNICAÇÃO SEGURA 14
3.1 Visão geral 14
3.2 Estratégias de detecção de erros de comunicação 14
3.3 PROFIsafe 15
3.4 FailSafe-over-EtherCAT 16
3.5 OpenSAFETY 17
3.6 CIP Safety 19
3.7 CIP Safety on Sercos 20
3.8 Análise comparativa 21
3.9 Critérios para a escolha de um protocolo seguro 22
4 O PROTOCOLO OPENSAFETY 25
4.1 Estrutura da stack 25
4.2 Funções principais 29
4.3 Desafios na utilização do openSAFETY 30
4.3.1 Documentação 30
4.3.2 Compilação 31
4.3.3 Suporte 31
5 O PROTOCOLO PROFISAFE 32
5.1 Características 32
5.1.1 Estrutura e formato da mensagem PROFIsafe 32
5.1.2 Drivers 33
5.2 Funções principais 35
6 TRABALHO EXPERIMENTAL 38
6.1 Modelo analítico 38
6.2 Arquitetura do ambiente 41
6.3 Testes realizados 44
6.3.1 Determinação do *baud rate* 44
6.3.2 Análise dos tempos de resposta em relação aos tempos de ciclo das CPUs 46
7 CONCLUSÃO 49
REFERÊNCIAS 50
APÊNDICE A – TRABALHO DE GRADUAÇÃO I 53

1 INTRODUÇÃO

Este capítulo apresenta as motivações para a realização deste trabalho, além da descrição da maneira como foi estruturado em capítulos de modo que fiquem concisas as informações dos estudos e experiências realizadas.

1.1 Motivação

A demanda por sistemas de comunicação industriais que suportem aplicações seguras vem crescendo diariamente. A automação industrial tem sido uma das áreas mais promissoras no campo da tecnologia, utilizando equipamentos cada vez mais robustos. Contudo, realizar a comunicação destas máquinas e serviços de maneira segura é uma tarefa árdua. Transportar as informações de forma segura é um ponto crítico em diversos setores da indústria. Por sistema seguro, entende-se detectar e prevenir falhas de processos críticos, controlando possíveis acidentes e danos, iniciando procedimentos emergenciais, como desligamentos e isolamentos instantâneos. Assim, ao detectar uma falha, o sistema deverá ser direcionado para um estado seguro ou de parada, onde o tempo de reação é o tempo de reconhecimento de uma requisição de uma função de segurança, o qual é altamente dependente da taxa de transmissão, da confiabilidade da rede e da velocidade de resposta. Funções de segurança, tais como alarmes de incêndio, bloqueio preventivo do funcionamento das máquinas, dentre outros, são aplicadas na automação industrial justamente para que as instalações garantam a execução destas funções, com um nível considerável de confiabilidade e disponibilidade, mediante o uso de técnicas de tolerância a falhas (LACHELLO L. et al.).

Nas aplicações industriais críticas são utilizados protocolos seguros de comunicação para a condução das informações de controle e monitoramento. Existem especificações, normas que devem ser atendidas e que explanam sobre como os protocolos de comunicação devem ser implementados e certificados para que possam ser considerados seguros. Todo protocolo de comunicação segura deve ser codificado e validado para cada equipamento que utilizá-lo. Muitas são as dúvidas relacionadas à não utilização da pilha TCP/IP para este tipo de comunicação. A resposta para isso é simples: esta pilha de protocolos não atende aos critérios das normas de segurança e, portanto, não pode ser utilizada para esta finalidade.

O objetivo deste trabalho foi analisar os protocolos de comunicação segura para automação industrial e, a partir disto, realizar uma avaliação de desempenho acerca dos tempos de resposta do protocolo definido como o mais apropriado, neste caso, o PROFIsafe.

1.2 Estrutura do trabalho

Nos capítulos subsequentes são tratados os aspectos que definem o que vem a ser uma comunicação segura, quais os protocolos vigentes, como escolher um deles, a investigação completa dos dois mais adequados e a experimentação elaborada acerca de um deles.

O capítulo 2 aborda os conceitos de Safety e de Black Channel, as normas de segurança que regem as regras de implementação dos protocolos e as certificações que devem ser respeitadas para garantir que os mesmos atinjam os níveis seguros de integridade almejados. No capítulo 3 são descritos os diversos protocolos de comunicação segura existentes, as características imprescindíveis para que sejam aceitos pelas normas e as particularidades de cada um, além de uma análise comparativa dos protocolos seguros, onde são ilustradas suas características através de tabelas e os critérios ao escolher um protocolo seguro. O capítulo 4 trata especificamente do protocolo openSAFETY a partir dos processos de documentação, compilação e suporte. O capítulo 5 trata da descrição do protocolo PROFIsafe trazendo suas características principais. O capítulo 6 aborda a parte experimental do trabalho, referente ao protocolo PROFIsafe. O capítulo final apresenta a conclusão, refletindo sobre os resultados obtidos e o que poderia vir a ser desenvolvido em trabalhos futuros.

2 SAFETY

Este capítulo retrata os pontos fundamentais em uma comunicação segura, apresentando seus conceitos e destacando a importância das normas para a obtenção da certificação.

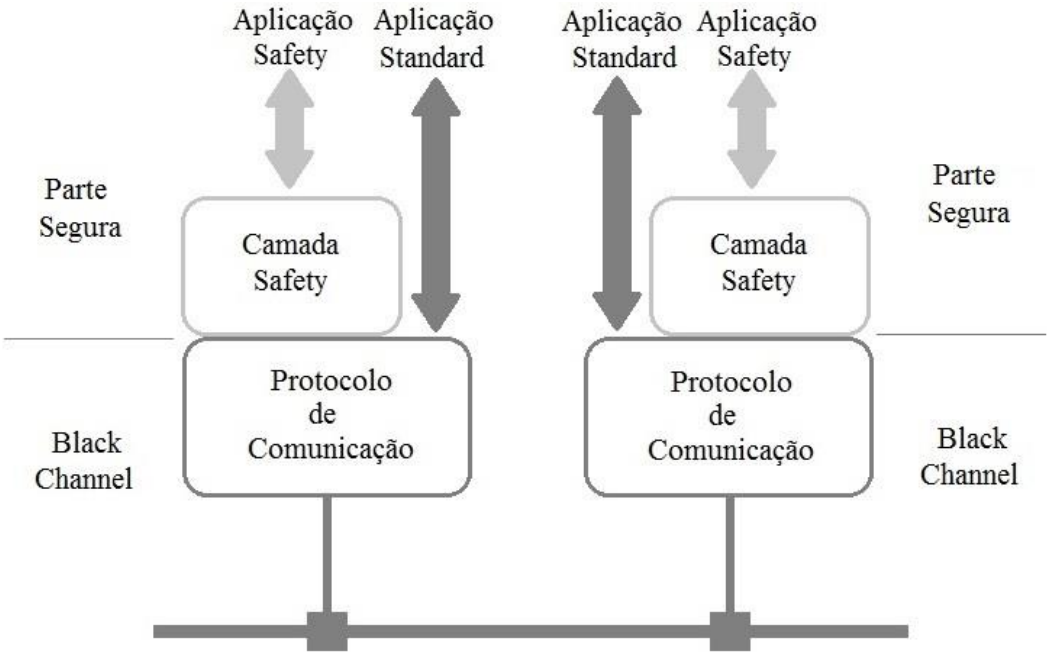
2.1 Conceito

A diferença dos conceitos de *safety* e de *security* é um pouco confusa para muitos. Ambos são traduzidos por segurança, contudo *security* está relacionado à proteção contra falhas maliciosas, no intuito de manter a privacidade, autenticidade, integridade e irrepudiabilidade dos dados. Já *safety*, tema principal deste trabalho de conclusão, garante que falhas não irão provocar danos ao sistema como um todo e nem às pessoas que o operam, sendo provável que o sistema esteja operacional e executando suas funções corretamente ou esteja descontinuando suas funções de forma a não provocar danos a outros sistemas ou a pessoas que deles dependam (AVIZIENIS et al., 2004).

2.2 Black Channel

A fim de eliminar a preocupação com os canais físicos de comunicação, os quais poderiam interferir na certificação do protocolo de comunicação segura desenvolvido, foi desenvolvido o conceito de Black Channel. Nele, os canais físicos estariam isolados numa camada não segura destinada somente para os mecanismos de transporte dos dados, assim como dos protocolos seguros, porém sem a necessidade de certificação. A divisão da comunicação em duas partes distintas torna o sistema mais fácil de ser implementado. O uso do conceito de Black Channel, desde que operando abaixo da taxa máxima de defeitos estipulada, facilita implementar segurança na camada acima, onde estão os protocolos que realizam a comunicação segura (estes sim, necessitando certificação). Ou seja, os dados seguros são transmitidos sem sofrer quaisquer alterações – mas podendo detectar erros como dado corrompido ou *frames* fora de ordem – de um dispositivo seguro até outro, sem a necessidade de se preocupar em como a transmissão é realizada, visto que comunicações *safety* e convencionais dividem o mesmo canal de comunicação (NEUMANN, 2007). A Figura 2.1 exemplifica o funcionamento do princípio Black Channel.

Figura 2.1 – Conceito de Black Channel



Fonte: Adaptado de (EPSG - The Black Channel Principle, 2016).

2.3 Normas de segurança

As normas de segurança são as responsáveis por minimizar os riscos inerentes na indústria, exigindo o emprego de técnicas de prevenção e de tolerância a falhas, o projeto detalhado, além da documentação – em todas as fases do ciclo de desenvolvimento. São elas que permitem que uma agência certificadora valide os sistemas, assumindo que os mesmos respeitem todas as regras e estejam verificados conforme requisitos.

Existem diversas normas de segurança, porém a que abrange a implementação da comunicação segura em seus aspectos gerais em diversos tipos de equipamentos, como sendo a norma "pai" da comunicação segura, é a IEC 61508 (Smith and Simpson, 2004). A aplicação da norma por si só não garante que o sistema seja realmente seguro. Para isso, além da norma bem empregada, é preciso que haja uma certificação (verificação e validação) do *software* e do *hardware* desenvolvido, após a realização de todos os testes, como os unitários, de integração e injeção de falhas.

Para tratar de segurança de uma maneira mais isolada, é possível realizar a separação dos sistemas de controle dos de segurança. Assim, as funções de segurança são executadas

nos SISs (Sistemas Instrumentados de Segurança) que utilizam apenas comunicação segura. Desta forma, somente estas funções e a implementação do protocolo de segurança como um todo têm a necessidade de serem certificados, resultando numa redução dos custos, isolamento de falhas e aumento da confiabilidade.

2.4 Certificação

Para que possamos garantir o nível de segurança de qualquer protocolo é preciso que haja um órgão que certifique que a função de segurança atingiu o SIL (nível de integridade de segurança) almejado seguindo estritamente a norma de segurança adequada. Deste modo, uma das agências internacionais autorizadas pela IEC analisa e certifica o nível de SIL alcançado em um dado protocolo. Avaliando todos os métodos utilizados para desenvolvimento e teste, como *hardware* e qualquer ferramenta de *software* aplicada, para, assim, poder determinar se a implementação dada para tal protocolo alcança o SIL requerido.

A norma IEC 61508 apresenta 4 níveis de integridade para sistemas relacionados à segurança, os chamados “Safety Integrity Levels” (SILs). O mais alto nível contido no padrão é o SIL 4, aplicado em sistemas altamente críticos, tais como usinas nucleares. Entretanto, SIL 3 é o mais alto nível exigido pelas plantas industriais nas aplicações tradicionais de fabricação e processamento. O nível SIL 3 requer que a probabilidade de falha para desempenhar uma função de segurança no modo de Baixa Demanda de operação seja $\geq 10\text{-}4$ até $< 10\text{-}3$, enquanto que no modo de Alta Demanda ou no modo contínuo de operação a probabilidade deve ser $\geq 10\text{-}8$ até $< 10\text{-}7$ (FELSER, M. et SAUTER, 2004, p. 418).

3 PROTOCOLOS DE COMUNICAÇÃO SEGURA

Neste capítulo são introduzidos os protocolos de comunicação segura, apresentando suas características em comum, assim como suas particularidades relacionadas ao funcionamento.

3.1 Visão geral

Implementar um protocolo seguro exige uma série de cuidados para que ele possa ser considerado confiável e seguro de fato. Assim, é preciso que o desenvolvimento atenda a todos os requisitos previstos na norma para que possa ser posteriormente certificado. Atualmente, existem pilhas de protocolos de segurança previamente aprovados que possibilitam a implementação de protocolos de comunicação segura sem a necessidade de criá-los por inteiro, fundamentando-se em um deles e ajustando seus respectivos protocolos de transporte e configurações de *hardware*.

São abordados a seguir o estudo realizado sobre cinco destes protocolos distintos, suas especificações, diferenças, em quais casos práticos se encaixam melhor, dentre outros fatores. Os protocolos em questão são: PROFISafe, FailSafe-over-EtherCAT, OpenSafety, CIP Safety on Sercos e CIP Safety. Todos eles possuem algumas características em comum: o uso do modelo Black Channel (que torna os protocolos totalmente independentes do protocolo de transporte), o mesmo padrão de segurança funcional (seguindo a norma IEC 61508), mesmos padrões para barramentos de campo (IEC 61158, IEC 61784-1, IEC 61784-2) e nível de SIL 3; além de que todos os protocolos ditos seguros necessitam prover mecanismos de detecção e correção de erros contidos nas mensagens transmitidas (ordenação, perdas, repetições, dentre outros). Contudo, em sua maioria, os protocolos de comunicação segura têm protocolos de transporte específicos definidos para cada um deles, conforme consta na norma que os especifica. Exceto pelo OpenSafety que não possui restrições quanto ao protocolo de transporte, podendo ser operado sobre diversos protocolos de transporte.

3.2 Estratégias de detecção de erros de comunicação

A IEC 61508 reforça algumas medidas de segurança para o desenvolvimento de um protocolo de comunicação segura (ELIA et al., 2006):

- Adicionar um número de sequência na mensagem a ser transmitida permite detectar erros de retransmissão, perda, inserção e sequência incorreta de dados;
- *Timestamp* em cada mensagem possibilita que o receptor identifique o momento em que a mensagem fora gerada para ser transmitida. Detectando, através dele, repetições, sequência de dados incorretas e atrasos. Além de permitir uma arbitragem melhor do acesso e do enfileiramento das mensagens para envio;
- Expectativa de tempo: limite de tempo estipulado entre ambos os envolvidos (emissor e receptor), que se excedido faz com que o receptor perceba que um erro (possivelmente um atraso) ocorreu;
- Mensagens de confirmação via *echo*: o receptor envia uma mensagem *echo* para o emissor, repetindo a mesma mensagem que recebeu para que o emissor verifique se a mesma está correta, identificando, assim, perdas, inserções, corrupção dos dados e mascaramento de dados. O que dificulta na utilização desta técnica é o fato da necessidade de haver uma retransmissão a cada mensagem recebida;
- Identificador para emissor e receptor: ambos reconhecem um ao outro através de um *id* adicionado a mensagem. Deste modo, detectam-se inserções de terceiros nas mensagens;
- Redundância com verificação cruzada (*cross checking*), comparando as mensagens recebidas para *crossover*, testando para transmissão correta. Detectando, de tal modo, quando houver repetições, perdas, erros de sequência e corrupção dos dados;
- Distinção das mensagens seguras (*safety related* - SR) das não seguras (*non-safety related* – NSR);
- Proteção dos dados, testando o conteúdo das mensagens antes de transmiti-las através do uso de CRC ou codificando os bits de uma maneira especial;

3.3 PROFIsafe

O protocolo de comunicação segura desenvolvido pela PI – PROFIBUS e PROFINET Internacional garante validação para segurança funcional somente quando for utilizado o protocolo PROFIBUS ou o PROFINET para o transporte. Contudo, permite o transporte das mensagens seguras no mesmo canal que o das mensagens padrão. Como dito anteriormente, um protocolo para ser considerado seguro precisa fornecer mecanismos de detecção e

correção de erros. No perfil de comunicação segura PROFIsafe essas funções são empregadas sobre as mensagens trocadas nas conexões, onde é verificada a consistência dos dados, adotando um sistema de nomes único para emissor e receptor, numerando as mensagens seguras, definindo tempos para cada uma delas.

O PROFIsafe apresenta 3 níveis de desempenho: para dados não-críticos por tempo (usa-se TCP, UDP e IP), para dados com processos críticos por tempo (Real-Time, no campo da automação industrial) e para demandas específicas (*hard real-time control*). O enfoque neste estudo é o desempenho para a área de automação industrial (Real-Time). O mecanismo de acesso à rede segue o modelo de comunicação mestre/escravo, onde o mestre é o F-Host e o escravo é o F-Device. Assim, o F-Device envia mensagens somente para responder às requisições do mestre (VIDAL et al., 2014). São adotadas algumas das medidas disponibilizadas pela norma para que o PROFIsafe atenda aos padrões de segurança. Elas são, a saber: número de sequência, *timestamp*, *watchdogs*, identificador emissor e receptor, diferença entre as mensagens SR e NSR, e a proteção dos dados (CRC, etc.) (ELIA et al., 2006).

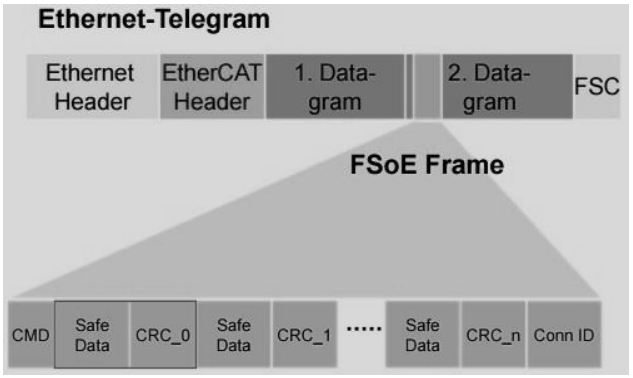
3.4 FailSafe-over-EtherCAT

Anteriormente conhecido como Safety-Over-EtherCAT, o FSoE é um protocolo aberto, especificado pelo ETG – EtherCAT Technology Group e aprovado pelo TÜV SÜD Rail GmbH, que para garantir a comunicação segura restringe a camada de transporte ao uso único e exclusivo de EtherCAT para transmissão dos dados. De maneira similar ao PROFIsafe, o FSoE envia suas mensagens seguras através de conexões e sessões exclusivas, recorre a métodos de ordenação sequencial, a fim de detectar possíveis duplicações e perdas das mensagens, e utiliza CRC para detecção de erros nos dados, diferindo do protocolo anterior somente no fato de utilizar um relógio global e rótulos de tempo nas mensagens.

Alcança ciclos de tempo real muito curtos, com alta capacidade de transferência, *jitter* na ordem de 1μs e probabilidade residual de erro $R(p) < 10\text{-}9/h$. Integra-se ao sistema TwinSAFE para desempenhar as funções de segurança, tais como paradas emergenciais e monitoramento da cerca segurança (*safety fence*), apresentando pequenas caixas de chaveamento diretamente ligadas a *safety fence*. A sua interação otimizada entre automação *standard* e tecnologia de segurança reduz os custos de *hardware* e engenharia, simplificando o cabeamento e facilitando a implementação de modificações. Utilizando a comunicação

mestre/escravo, o *frame* FSoE é mapeado em PDOs (*Process Data objects*) cíclicos, com tamanho mínimo do quadro FSoE de 6 Bytes e máximo dependendo do número do dado do processo seguro do dispositivo escravo. O ciclo FSoE consiste em um quadro mestre FSoE confirmado por um quadro escravo FSoE. O FSoE mestre envia um quadro para o FSoE escravo e, ao enviá-lo, o mestre inicia um *Watchdog-Timer* para proteger o escravo. O mestre somente iniciará um novo ciclo, criando outro quadro mestre FSoE, quando receber um quadro escravo válido. Caso o *Watchdog* expire, os dispositivos mudarão para o estado de *Reset* (LIU AND SONG, 2012).

Figura 3.1 – Quadro FSoE



Fonte: Adaptado de (Beckmann, 2016).

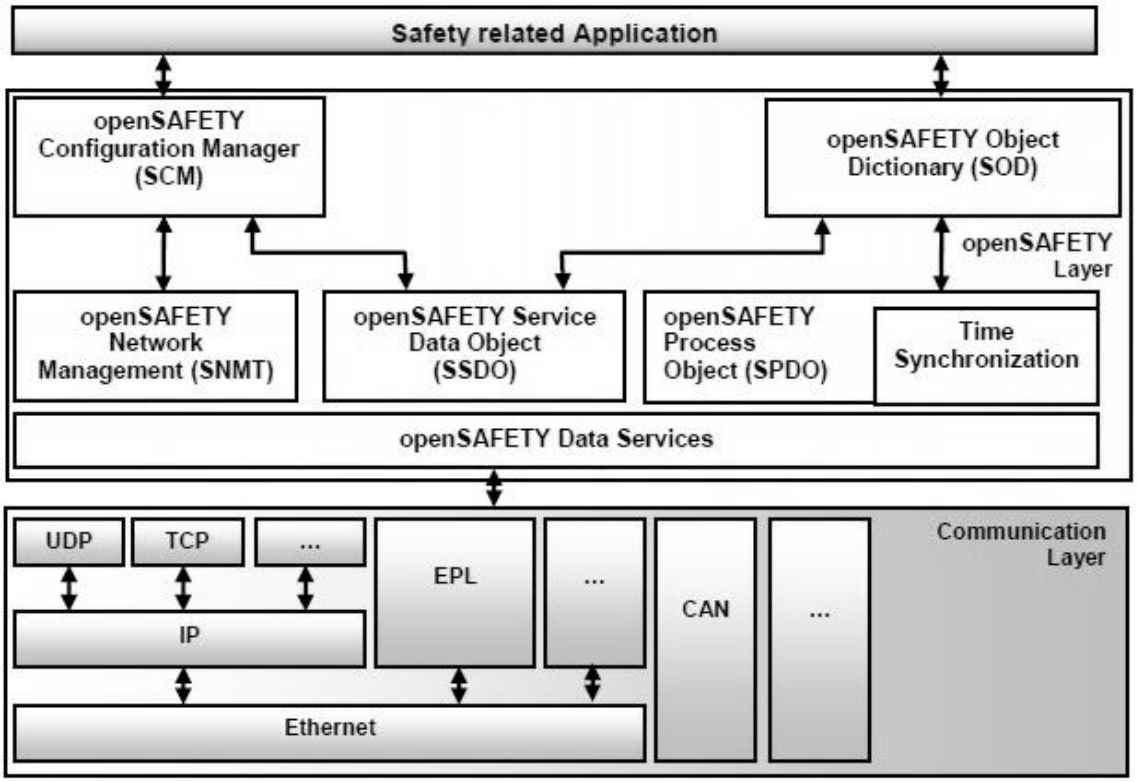
A figura 3.1 exemplifica como é formado o quadro FSoE, cujo mapeamento é feito como se fosse um contêiner dos dados do processo do dispositivo, havendo uma verificação a cada 2 *bytes* de dados seguros com um CRC de mesmo tamanho, sem impor limite quanto ao tamanho dos dados seguros processados.

3.5 OpenSAFETY

Inserido no mercado, inicialmente, com o nome de EPL safety (Ethernet Powerlink Safety), foi desenvolvido pela EPSG (Ethernet POWERLINK Standardisation Group) como um protocolo aberto e mais flexível por ser totalmente independente do protocolo de transporte, não restringindo a camada de transporte a utilização de um protocolo específico. Em 2008 foi testado pela TÜV Rheinland Group e aprovado para uso em aplicações *safety* como especificado pela norma (ZURAWSKI, 2014). Assim como o FSoE utiliza conexões e sessões únicas, onde emprega rótulos de tempo nas mensagens, verificando a integridade dos dados através de CRC.

Realiza um monitoramento da comunicação por tempo (*time-slicing*) a fim de prevenir erros causados por perdas dos dados ou *delays* excessivos, utilizando também o mecanismo de *Watchdogs*, onde os usuários poderiam identificar um mal funcionamento, verificando a sequência contínua do fluxo de dados. Assim, o OpenSafety garante transmissão dos dados de maneira segura utilizando redes "inseguras", sendo adequado para comunicações com ciclos na faixa de μs, apresentando um tempo de resposta abaixo de 100us. Ademais, habilita um gerenciamento de riscos flexível, *Smart Safe Reactions*, ao invés de paradas emergenciais, onde na maioria das vezes é suficiente reduzir a velocidade/movimentação das máquinas ou limitar o torque para um nível seguro ao invés de parar tudo simultaneamente (POWERLINK Safety - Open Safety Technology, 2009).

Figura 3.2 – Modelo de referência do OpenSAFETY



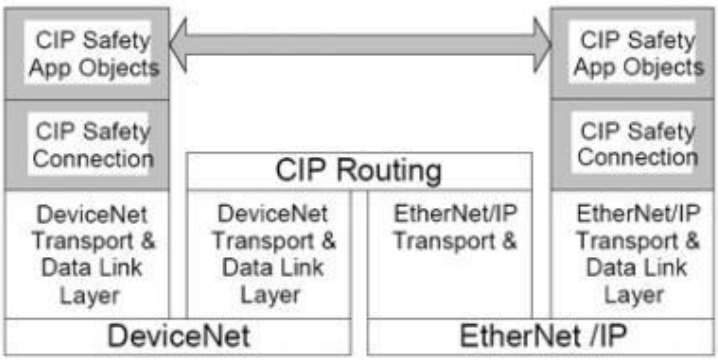
Fonte: Adaptado de (Ethernet POWERLINK Standardisation Group, 2013).

A Figura 3.2 ilustra o modelo de referência do OpenSafety. Nele podemos observar a independência da camada de transporte, além da forma mais compacta que o protocolo trata a comunicação segura, encapsulando a representação dos dados e a definição de transporte (openSAFETY Data Services, como pode-se observar na figura acima) e os serviços de nível superior de maneira similar ao POWERLINK (Ethernet POWERLINK Standardisation Group, 2013).

3.6 CIP Safety

O perfil de comunicação especificado pela ODVA (Open Device Vendors Association) é uma extensão do padrão, já certificado pela TÜV Rheinland, CIP (Common Industrial Protocol), mediante a adição da funcionalidade da camada de aplicação CIP Safety ao padrão (ZURAWSKI, 2014). Como as extensões da camada de aplicação de segurança não dependem da integridade dos serviços CIP padrão e das camadas de enlace de dados, *hardware* de canal único (não redundante) pode ser utilizado para a interface de comunicação de enlace de dados. Esta divisão de funcionalidades permite que roteadores padrão passem a rotear os dados seguros como ilustrado na Figura 3.3. O roteamento das mensagens é possível pois o dispositivo final é responsável por garantir a integridade dos dados. Então, quando houver um erro na transmissão dos dados ou no roteador intermediário, o dispositivo final detectará a falha e tomará uma ação apropriada (VASKO AND AUTOMATION, 2005).

Figura 3.3 – Roteamento das mensagens



Fonte: (Vasko and Automation, 2005).

A transmissão dos dados é realizada via Ethernet/IP ou DeviceNet, além de utilizar o mecanismo origem-destino para troca de dados entre os sistemas *safety*, definindo tempos para as mensagens através de um *timestamp*, garantindo que os dados estejam atualizados, usufruindo também de um identificador para enviar e receber os dados. O CIP *safety* utiliza códigos CRC através de objetos de validação *safety* em ambas as extremidades (emissor e receptor) de forma a verificar os dados transmitidos, organizando e garantindo a integridade dos mesmos, permitindo, também, que dispositivos *safety* e *standard* operem em uma mesma rede.

3.7 CIP Safety on Sercos

CIP Safety on Sercos é um protocolo para transmissão de dados relevantes a *safety* (safety-relevant) via Sercos. Dados *safety-relevant* são transmitidos como um contêiner de dados *safety*. O dado do contêiner é armazenado no canal do dispositivo de tempo real relevante exatamente como dados padrões. Um protocolo multiplex, SMP (*Sercos Messaging Protocol*), é utilizado para transmitir dados seguros que tenham sido escaneados sem perder largura de banda apesar de terem ciclos de barramento mais curtos (SAFETY, 2016).

Figura 3.4 – Modelo estrutural do perfil de comunicação CIP Safety on Sercos

Fonte: Adaptado de (Safety, 2016).

Baseado no protocolo CIP *safety* foi especificado pela ODVA (Open Device Vendors Association), utilizando acesso múltiplo por divisão de tempo (TDMA), onde destina-se um canal dedicado livre de colisão para a comunicação em tempo real e um canal IP adicional para transferência de dados assíncronos. Cada dispositivo possui 2 portas Ethernet, um controlador responsável pelo chaveamento entre o trânsito de dados em tempo real e o trânsito de IPs, garantindo uma sincronização sólida de todos os nodos conectados. A comunicação do tipo mestre/escravo acontece de maneira cíclica, pois o mestre atua como um controlador numérico (CN), no tempo de ciclo selecionado na inicialização; podendo ser de 31.25µs, 62µs, 125µs, 250µs ou qualquer múltiplo de 250µs até 65ms; o *jitter* encontra-se na ordem de 1µs. Além disso, o protocolo suporta comunicação direta entre os escravos. Esta funcionalidade é vantajosa nas aplicações de controle de movimentação seguras: os dados seguros podem ser trocados entre os escravos diretamente utilizando comunicação cruzada

peer-to-peer, sem a necessidade de coletar e redistribuir os dados através de um mestre centralizado. Outra grande vantagem é a definição de um mecanismo de redundância quando for empregada a topologia do anel para gerar um anel duplo. O mecanismo oferece caminhos alternados para prover uma recuperação "sem colisões" do rompimento de um dos anéis (ELIA et al., 2006).

Ademais, a interface SERCOS com *drives* digitais inteligentes fornece monitoramento preciso e proteção contra perda do controle de movimentos/velocidade, realizando um desligamento forçado em caso de falha do processador do *drive*. Mestres e escravos inserem números de sequência nas mensagens a fim de especificar o *telegram* a ser transmitido para tentar evitar erros na comunicação como repetições, corrupção dos dados, dentre outros. Cada nodo do anel mantém o cálculo do número de sequência, podendo identificar quando uma sequência for inválida. E a corrupção de um *bit* é detectada via *procedure* HDLC, código que permite reconhecer erros de *bit* com uma distância de *hamming* maior do que 4, e via monitoramento adicional dos tamanhos das mensagens e dos tempos de transmissão. Podendo utilizar, também, diversas redes de comunicação, tais como: DeviceNet e Ethernet/IP.

3.8 Análise comparativa

A coleta de dados referente aos perfis existentes para a implementação de protocolos de comunicação segura possibilitou a realização de uma análise comparativa, identificando as vantagens e limitações de cada um.

A ocorrência de erros na comunicação dá-se durante a troca de mensagens, resultando em falhas de segurança. Devido a isto, são aplicadas medidas para detectar os tipos de erros citados na tabela 4.1, para que, assim, possam ser evitadas panes nos sistemas que utilizarem tais protocolos. O quadro abaixo identifica as medidas de segurança adotadas em cada protocolo, dependendo do erro detectado.

Tabela 3.1 – Medidas de segurança para cada protocolo conforme erro detectado

Erros	PROFIsafe	FsOE	OpenSAFETY	CIP Safety	CIP Safety on Sercos
Repetição	▪ Nº de sequência ▪ Timestamp	▪ Nº de sequência ▪ CRC	▪ Timestamp	▪ Timestamp	▪ Timestamp
Perda	▪ Nº de sequência	▪ Nº de sequência ▪ Watchdog ▪ CRC	▪ Timestamp ▪ Watchdog	▪ Timestamp	▪ Timestamp

Inserção	▪ Nº de sequência ▪ Identificador	▪ Nº de sequência ▪ CRC	▪ Identificador	▪ Timestamp ▪ Identificador	▪ Timestamp ▪ Identificador
Sequência incorreta	▪ Nº de sequência ▪ Timestamp	▪ Nº de sequência ▪ CRC	▪ Timestamp	▪ Timestamp	▪ Timestamp
Corrupção dos dados	▪ Integridade dos dados via CRC	▪ CRC	▪ CRC ▪ Redundância via *cross check*	▪ CRC ▪ Redundância via *cross check*	▪ CRC ▪ Redundância via *cross check*
Atrasos	▪ Watchdog	▪ Watchdog	▪ Watchdog	▪ Timestamp	▪ Timestamp
Mascara-mento das mensagens	▪ Identificador ▪ Watchdog ▪ CRC	▪ Watchdog ▪ CRC	▪ Identificador ▪ Timestamp ▪ CRC	▪ Identificador	▪ Identificador

Como pode ser observado acima, o protocolo *safety* desenvolvido pela Sercos utiliza o CIP Safety para as funcionalidades de segurança; logo, as medidas adotadas para tratar os erros são as mesmas. É interessante ressaltar também o uso de *timestamp* como mecanismo de detecção de erros de inserção no CIP Safety e no CIP Safety on Sercos, pois, neste caso, o *timestamp* armazena a informação da "idade" dos dados. Se a "idade" dos dados for maior do que o máximo permitido, o sistema que utiliza um destes dois protocolos passa para um estado seguro.

3.9 Critérios para a escolha de um protocolo seguro

Os fatores cruciais que fazem com que uma empresa decida utilizar um protocolo para sua determinada aplicação dependem não somente das medidas de segurança, que, em sua maioria, são similares pois atendem aos requisitos da norma IEC 61508, mas principalmente em relação ao custo, aos módulos de compatibilidade disponíveis, à implementação e certificação. Assim, os que utilizam somente um protocolo de transporte e módulos específicos, como o PROFIsafe e o FSoE, embora consolidados no mercado, são os mais rígidos e inflexíveis, e, portanto, em teoria, os mais custosos para implementação. Os demais podem atender a aplicações mais variadas por se adaptarem melhor, aceitando diversos módulos e tipos de comunicação. Entretanto, por ser um código aberto e ao mesmo tempo não restringir diversos fatores para a implementação e utilização de um protocolo de comunicação segura, o que se mostrou mais flexível, em um primeiro momento, em todos os aspectos foi o OpenSafety, pois, além de não limitar a transmissão a um protocolo de transporte específico,

o investimento parecia ser mais viável, menos custoso, com garantia de uma comunicação confiável e tolerante a falhas.

Tabela 3.2 – Demais critérios para a escolha de um protocolo *safety*

Critérios	PROFIsafe	FSoE	OpenSAFETY	CIP Safety	CIP Safety on Sercos
Custo	▪ Elevado (Licença, certificação e parametrização)	▪ Elevado (Licença e certificação)	▪ Baixo: sem custo para obtenção ▪ Custo somente para certificá-lo	▪ Custos com Licença, certificação, módulos compatíveis	▪ Reduz os custos de HW e instalação ▪ Precisa de Licença e certificação
Desempenho	▪ Tempos de ciclo reduzidos para 31.25µs	▪ Tempos de ciclo ≤100µs	▪ Tempo de resposta 4x mais rápido que o FSoE	▪ Tempos de ciclo mínimos de 31.25µs	▪ Tempos de ciclo mínimos de 31.25µs
Flexibilidade	▪ Transporte restrito ao uso de PROFIBUS ou PROFINET	▪ Transporte somente via EtherCAT	▪ Totalmente flexível (independente de protocolo)	▪ Transporte via Ethernet/IP ou DeviceNet	▪ Transporte via barramento Sercos III
Topologia	▪ Estruturas flexíveis (linha, árvore, estrela ou anel)	▪ Atende a qualquer combinação de topologia	▪ Aceita qualquer topologia	▪ Atende as diversas combinações (linha, anel...)	▪ Atende as diversas combinações (linha, anel...)
Facilidade de obtenção	▪ Parte da documentação disponível no site da PI e *kit* de desenvolvimento disponível no mercado	▪ Documentos disponíveis no *site* do grupo EtherCAT e obtenção na matriz ETG	▪ Open-source, gratuito, pré-certificado (como os outros) e disponível no site para *download*	▪ Documentos disponíveis no *site* do grupo ODVA e venda nos *sites* de fabricantes	▪ Disponível para venda no *site* da Sercos direcionando para os fabricantes

Com o intuito de analisar ainda em mais detalhe os protocolos de comunicação segura para automação industrial, foi escolhido inicialmente o protocolo openSAFETY, julgado como um dos mais flexíveis (independente de protocolo de transporte, módulos simples e facilmente adaptáveis) para que fosse realizada uma análise na prática. Logo, o que havia sido colocado como objetivo original para este trabalho era a experimentação com o protocolo de comunicação openSAFETY, as dificuldades para colocá-lo em funcionamento referentes tanto à documentação quanto à sua utilização, verificando todos os aspectos que uma empresa levaria em consideração antes de utilizar tal perfil. Contudo, a experimentação de seu uso concreto se mostrou distante das percepções teóricas. Isto se deu devido às dificuldades de

execução do protocolo openSAFETY, que serão detalhadas no próximo capítulo. Deste modo, optou-se por voltar a atenção de análise para medições de desempenho de outro protocolo, o PROFIsafe.

Apesar da análise inicial ter evidenciado que este é um protocolo mais rígido em termos de utilizar somente um protocolo de transporte, a parceria existente entre o laboratório de pesquisa regido pelos orientadores deste trabalho e a empresa Altus, de automação industrial, proporcionou o uso de módulos da empresa para que fosse simulado em um ambiente real o protocolo PROFIsafe. Esta escolha também foi baseada no fato do protocolo ser bem documentado e de fácil entendimento, tornando sua usabilidade mais intuitiva. O detalhamento das atividades relacionadas ao PROFIsafe será explicitado no capítulo 5.

4 O PROTOCOLO OPENSAFETY

Este capítulo descreve em detalhes a *stack* do openSAFETY desde sua estrutura de diretórios e arquivos, principais funções até os problemas que surgiram durante o período de análise e testes da pilha do protocolo.

Como visto no capítulo anterior, o protocolo oferece várias vantagens, é aberto e independente do protocolo de transporte. Sua escolha para uma aplicação real traria vários benefícios. Por esse motivo o protocolo foi escolhido para um estudo mais detalhado.

No entanto, o item 4.3 nos mostra os desafios que surgiram na tentativa de utilizar o protocolo, devido a falta de documentação e as dificuldades de instalação. Isto, resultou em uma mudança de protocolo para que fosse realizada a análise acerca dos tempos de resposta obtidos em uma comunicação segura.

4.1 Estrutura da stack

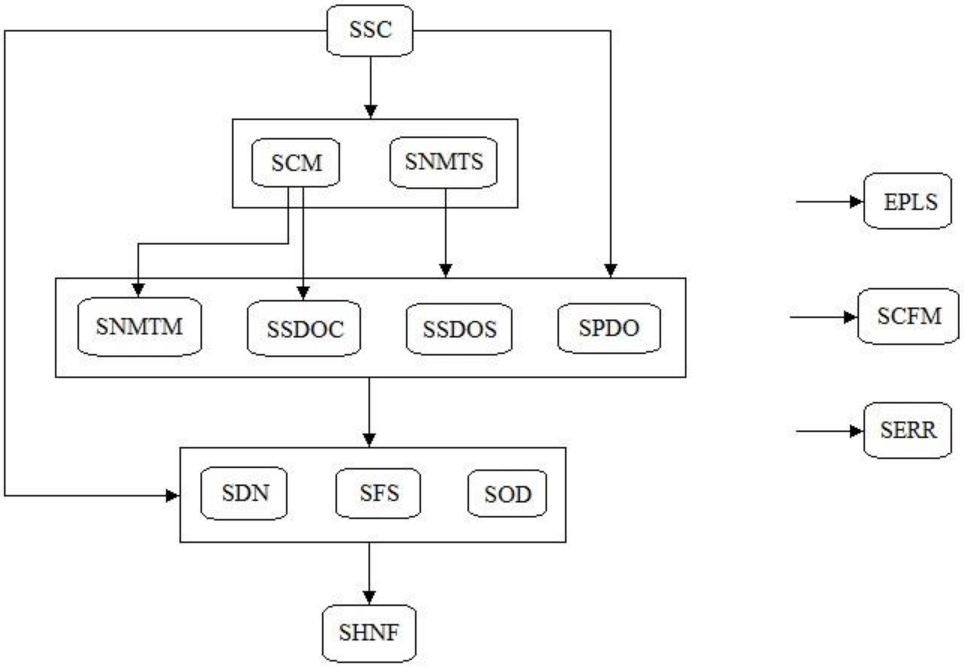
A *stack* do protocolo openSAFETY foi desenvolvida pela B&R como uma implementação referência da especificação do openSAFETY de acordo com a IEC 61508. Sendo disponibilizado com o *framework* da licença BSD (sem custo) em conjunto com uma ferramenta para calcular o CRC, tem seu código fonte desenvolvido em C, independente de *hardware*, além de fazer o uso de uma distribuição baseada em CMake com o intuito de permitir o desenvolvimento e a distribuição em *cross-plataforms*.

A estrutura de *software* modular é implementada através do modelo de comunicação mestre/escravo. O openSAFETY estabelece uma rede de comunicação segura entre dispositivos, denominada Safety Domain (SD), onde cada dispositivo é definido como um Safety Node (SN). Dentre eles, somente um poderá atuar como um SCM (Safety Master), responsável por gerenciar o protocolo, e os demais serão *n* nodos SS (Safety Slave). Toda a comunicação dentro de cada rede é identificada por um SDN (Safety Domain Number) comum, assim como o UDID do SCM principal. Como ilustrado na figura 4.1, a estrutura do openSAFETY é dividida em módulos distintos.

Os módulos SERR (Safety Error), SCFM (Safety Control Flow Monitoring) e EPLS (openSAFETY auxiliary support module) são auxiliares, podendo ser utilizados por quaisquer outros módulos. Todos os demais módulos possuem uma api própria, funções de interface

para a aplicação alvo, exportadas via arquivos de *header*, como por exemplo, o módulo SSC que fornece funções que estão definidas no arquivo *header* SSCapi.h.

Figura 4.1 – Estrutura da pilha do protocolo openSAFETY



Fonte: Adaptado de (openSAFETY: openSAFETY Software Structure).

São descritos, a seguir, os papéis exercidos por cada módulo ilustrado na figura 4.1, contido na *stack* openSAFETY.

- **SSC (Safety Stack Control)**: é a interface central para a aplicação. A pilha openSAFETY deve ser inicializada pela função de inicialização do SSC antes de qualquer outra função da API ser chamada.
- **SNMTS (Safety Network Management Slave)**: chamado internamente pelo SSC, executa as seguintes tarefas:
    - Processamento das requisições do SNMTM e SNMTS, e criação de resposta quando necessário;
    - Monitoramento dos tempos de guarda e de *refresh*, através de uma função de verificação de tempos própria que deve ser chamada ciclicamente;
    - Gerenciamento do estado do nodo.

- **SCM (Safety Configuration Manager)**: é responsável pela configuração, verificando e monitorando todos os SNs em um SD, podendo ser ativado ou desativado através de funções específicas contidas em seu arquivo header. Não permitindo a modificação de qualquer entrada do módulo SOD que seja relevante à configuração durante a etapa de configuração da *stack*. O SCM sempre configura e monitora apenas um único domínio. Por isso, ele é sempre atribuído para a instância fixa em 0. Além disso, também verifica se um SN deve ser processado ou não, utilizando uma função que a aplicação fornece. Assim, a aplicação pode excluir os SNs que não serão processados. Alguns problemas podem surgir caso o SN trave o SOD para calcular o CRC, pois o SCM também necessita acessar o SOD.
- **SSDOC (Safety Service Data Object Client)**: habilita o acesso de um servidor SSDOS e SSDOC para um diretório de objeto local de maneira segmentada ou não-segmentada, com acessos de leitura ou escrita, podendo haver múltiplos acessos simultâneos. As requisições que não receberem resposta devem ser reenviadas. Ademais, o SCM também executa acessos SSDOS e SSDOC.
- **SNMTM (Safety Network Management Master)**: fornece serviços para controlar o estado dos SNs, assim como, para monitorar e atribuir um SN a SADRs. Suas requisições são respondidas através da unidade SSC. Havendo a necessidade de reenvio das requisições quando não for recebida uma resposta.
- **SSDOS (Safety Service Data Object Server)**: este módulo interno habilita o acesso para o diretório de objeto local via rede de maneira segmentada ou não-segmentada, com acessos de leitura ou escrita. Mas somente uma transferência SSDO (Cliente e Servidor) pode ser feita por vez para cada instância. Devido ao monitoramento de timeout não estar implementado, quaisquer transferências correntes serão sempre interrompidas por uma nova transferência. Existindo somente um SCM para cada SD. E, se houver uma transferência segmentada, o SOD é acessado diretamente, sendo bloqueado durante o tempo necessário para realizar a transferência completa.
- **SPDO (Safety Process Data Objects)**: utilizado para trocas cíclicas de dados de processo e para manipular os tempos de sincronização e de validação. Como o openSAFETY suporta mapeamento dinâmico, parâmetros de mapeamento podem ser modificados durante a operação. No entanto, os parâmetros de mapeamento podem ser alterados para qualquer outra sequência somente quando estiver no estado “Pré-

operacional", pois o mapeamento não está ativado até o estado mudar para "Operacional". Ao realizar o mapeamento, a aplicação deve levar em consideração que o comprimento máximo é utilizado para objetos mapeados com comprimento variável no SPDO.

- **SDN (Safety Domain Number)**: permite atribuir um SDN a um número de instância.
- **SFS (Safety Frame Serialization)**: este módulo é utilizado para agrupar (serializar), desagrupar (desserializar) e verificar os frames openSAFETY.
- **SOD (Safety Object Dictionary)**: gerencia o acesso ao diretório do objeto. Sendo importante certificar-se de que o acesso de escrita a objetos obrigatórios no SOD (particularmente o parâmetro de mapeamento e comunicação do SPDO) não é permitido no estado "Operacional" e nem durante a transição para "Operacional". Os objetos no SOD devem estar alocados diretamente na memória. Assim, as tentativas de acesso com *delay* não serão possíveis, a não ser que sejam realizadas sucessivos readings.
- **SERR (Safety Error)**: detecta erros na pilha e informa à aplicação. Se um erro "Fail Safe" acontecer, a aplicação deverá, então, criar um estado seguro para o dispositivo, incluindo todos os objetos SODs pertinentes.
- **SCFM (Safety Control Flow Monitoring)**: atua no monitoramento do fluxo correto do programa. Para isso, o usuário deve ler e comparar o número de chamadas de função nos controladores safety. Se os valores forem diferentes, a aplicação responderá conforme este erro.
- **EPLS (openSAFETY auxiliary support module)**: é um auxiliar com variáveis globais, como defines, contendo somente arquivos de header, não fornecendo nenhuma implementação.

Cada SN tem um único número de identificação UDID. Durante o processo de inicialização, o SCM verifica o tipo de dispositivo e o UDID e automaticamente determina que a rede de configuração correta está sendo utilizada. Após a confirmação do usuário, os parâmetros requisitados são transferidos para os SNs. A configuração automática reduz os tempos de manutenção e aumenta a disponibilidade da máquina.

O SOD gerencia todos os parâmetros e a configuração de cada nodo individualmente. Após a conclusão da configuração do nodo e a fase de inicialização, a transferência cíclica de

dados entre o cliente e o servidor começa. Para a transferência de dados de processos seguros críticos emprega-se o SPDO. O *frame* openSAFETY consiste em dois sub *frames*. Podendo transportar no máximo 240 *bytes* de dados seguros, utilizando CRC 8 para *payloads* de 1 a 8 *bytes* e CRC 16 para *payloads* de 9 a 254 *bytes*. Redes muito grandes podem ser criadas. Cada SD tem até 1023 SNs conectados e endereçados pelo SCM. E a configuração total máxima de um openSAFETY é de 1023 SDs com mais de um milhão de nós seguros.

4.2 Funções principais

A implementação de um SN é realizada em quatro passos distintos. Primeiro, todas as funções ditas essenciais devem ser desenvolvidas. Em seguida, devem ser implementados, também, os *callbacks* recomendados com base na aplicação construída. O terceiro passo é dimensionar a pilha openSAFETY para, por fim, implementar e integrar um SOD. O escopo das funções é baseado na especificação atual do openSAFETY. As funções necessárias (essenciais) para a implementação de um SN são mostradas na tabela 4.1.

Tabela 4.1 – Funções principais da pilha openSAFETY

Funções principais	*Descrição*
SAPL_SERR_SignalErrorClbk()	Erros são encaminhados para esta função que permite que a pilha relate os erros
SAPL_SNMTS_CalcParamChkSumClbk()	A pilha dispara o cálculo do *checksum* do SOD
SAPL_SNMTS_SwitchToOpReqClbk()	A pilha solicita o chaveamento do SN para operacional
SAPL_SNMTS_ErrorAckClbk()	Notifica que uma confirmação (SN Ack) foi recebida
SAPL_SNMTS_ParameterSetProcessed()	Verifica se o processamento dos parâmetros recebidos está finalizado
SHNF_GetTxMemBlock()	Aloca memória para gerar um *frame safety*
SHNF_MarkTxMemBlock()	Marca uma memória alocada como utilizada
HNFiff_Crc8CalcSwp()	Calcula o openSAFETY CRC 8
HNFiff_Crc16CalcSwp()	Calcula o openSAFETY CRC 16
HNFiff_Crc16_755B_CalcSwp()	Calcula o CRC16 para dados de configuração

Além destas, existem as funções com implementações recomendadas, nas quais se faz necessária a implementação do corpo de algumas dessas funções, podendo ser definidas como um valor *default*. A tabela 4.2 descreve os empregos destas funções, tais como: inicializações de módulos, estado atual do SN, chaveamento entre estados (pré-operacional e operacional),

verificações de *timeout*, cálculo do *checksum* informando se o mesmo é válido, geração do SPDO, dentre outros.

Tabela 4.2 – Funções recomendadas da pilha openSAFETY

Funções recomendadas	*Descrição*
SNMTS_TimerCheck()	Precisa ser chamada para verificar *timeout*
SNMTS_GetSnState()	Retorna o estado atual do SN
SNMTS_PassParamChkSumValid()	Chamada após o cálculo de *checksum* do SOD para informar a pilha quando o *checksum* SOD é válido ou não
SNMTS_EnterOpState ()	Informa a pilha se será chaveado para operacional ou não (explanando o porquê em caso negativo)
SNMTS_PerformTransPreOp()	Força a pilha a ir para o estado pré-operacional
SSC_InitAll()	Chamada após a inicialização da pilha estar concluída
SSC_ProcessSNMTSSDOFrame()	Executada quando um SSDO ou um SNMT é recebido pela rede
SCFM_GetResetPath()	Retorna o contador do fluxo do programa
SCFM_TACK_PATH()	Incrementa o contador do fluxo do programa (não obrigatório)
SPDO_BuildTxSpdo()	Chamado pela aplicação para gerar o SPDO
SPDO_TxDataChanged()	Garante que o SPDO é gerado dentro do ciclo do módulo
SPDO_ProcessRxSpdo()	Chamado por cada SPDO recebido
SPDO_CheckRxTimeout()	A ser executado após o SPDO para este ciclo de módulo ser recebido
SPDO_GetRxSpdoStatus()	Retorno o *status* detalhado de um RxSPDO

4.3 Desafios na utilização do openSAFETY

Este item trata dos problemas encontrados na tentativa de fazer uma instalação do protocolo e executar comunicação segura em um ambiente formado por uma comunicação mestre/escravo independente do protocolo de transporte escolhido. A falta de documentação e as dificuldades de compilação e suporte para a utilização do protocolo impossibilitaram seu uso. São descritas a seguir as barreiras que influenciaram na troca do protocolo abordado.

4.3.1 Documentação

A fim de testar o funcionamento da pilha do protocolo openSAFETY, o passo inicial foi tentar entender a estrutura e a forma como as funções eram chamadas para que pudesse ser realizada uma comunicação segura. No entanto, não foi encontrado um *Safety Manual* de fato,

o que ocasionou diversos problemas, visto que o entendimento do código não é nem um pouco intuitivo. As únicas fontes de informação disponíveis para a compreensão do protocolo eram o *site*, que direciona para o *download* da especificação do protocolo – um tanto confusa –, e o Git que praticamente repete, de maneira mais sucinta, o que está escrito no documento de especificação. No Git foi encontrado um DEMO do protocolo que serviria para testarmos o funcionamento da comunicação segura utilizando openSAFETY.

4.3.2 Compilação

Ao compilar o DEMO surgiram outros problemas. Como não tínhamos o *hardware* necessário (plataforma FPGA com a implementação de um POWERLINK Communication Processor – PCP) para a compilação completa do DEMO no laboratório, retirou-se a parte do *hardware* tentando simular somente a parte do *software* onde o protocolo de comunicação segura atua. Porém, após inúmeras tentativas de compilação, percebeu-se que a tarefa não era tão simples. As dificuldades enfrentadas mostraram que, apesar de ser código aberto e de atender a uma série de protocolos de transporte, o investimento em tempo para vencer as dificuldades não compensaria quando comparado ao custo de aquisição e instalação relativo aos protocolos já consolidados. A falta de documentação e suporte faz com que seja preciso investir na capacitação de uma equipe, o que ocasiona em tempo dispendido, realocação de funcionários e custos, que, na maioria das vezes, não podem ser facilmente supridos pelas indústrias.

4.3.3 Suporte

Durante o processo de compreensão e utilização do openSAFETY, com o intuito de obter esclarecimentos e algum auxílio que possibilitasse um avanço na experimentação do protocolo, contatou-se o suporte do protocolo na Alemanha via e-mail, que estava disponibilizado no *site* da openSAFETY. Todavia, não foi fornecido nada além do que já havíamos encontrado, pois aquelas já eram as fontes de informações existentes a respeito da pilha openSAFETY. Foi informado que tudo o que havia de documentação era o que já estava disponível no *site*, não havendo nenhum *Safety Manual* do protocolo, como existem de outros protocolos já consolidados.

5 O PROTOCOLO PROFISAFE

Neste capítulo são apresentados detalhes do PROFIsafe, sua estrutura, principais funções e os casos de uso com estrutura de *software* de alto nível, exemplificando a utilização do protocolo.

5.1 Características

Com 5.4 milhões de nodos instalados, o protocolo PROFIsafe tem se estabelecido em uma posição de liderança no mercado de sistemas de comunicação segura (PROFIsafe, 2017). Baseado nos padrões da IEC 61784-3-3, o protocolo utiliza o mecanismo de transmissão mestre/escravo para transmissão dos dados, onde o mestre (F-Host) realiza uma troca cíclica de dados *safety* com seus escravos (F-Devices), associados através de uma relação de comunicação um-para-um (IEC 61784-3-3, 2010).

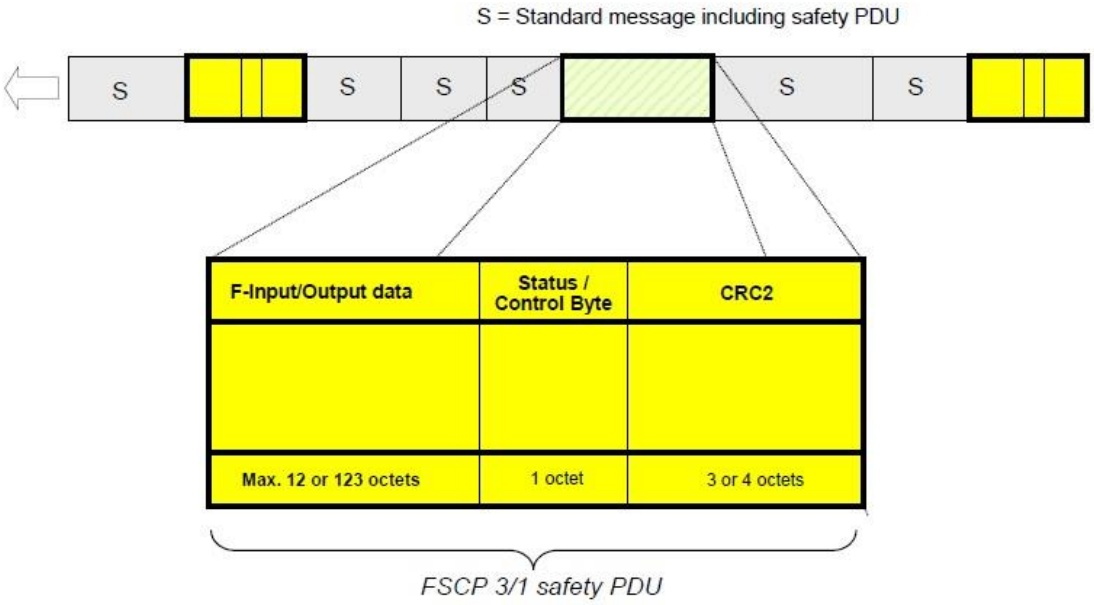
Segundo a norma, para a realização de um perfil de comunicação segura funcional, quatro medidas essenciais devem ser adotadas: número de sequência, monitoramento de tempo via *watchdog* com confirmação, verificação da integridade dos dados via CRC e identificador (codinome) nas mensagens entre o emissor e o receptor. Os números de sequência são utilizados para que os destinatários possam verificar que os telegramas *safety* chegaram na ordem correta. O *watchdog time* (reinicializado após o recebimento de um telegrama) garante que os telegramas válidos serão sempre lidos. E o identificador único entre o F-Host e o F-Device é fornecido pelos parâmetros PROFIsafe (F-Parameters).

5.1.1 Estrutura e formato da mensagem PROFIsafe

Para cada F-Device existe um F-Driver organizando a coordenação das mensagens seguras, denominadas de “Safety PDUs” (Protocol Data Unit) entre o F-Host e o F-Slave. O cálculo do CRC das PDUs depende do tamanho da mensagem a ser transferida, diferindo entre “slim PDU” (PDU pequena) de até 12 *bytes* e “long PDU” (PDU longa) de até 123 *bytes*. É utilizado CRC 24 para o cálculo de PDUs pequenas e CRC 32 para o cálculo de PDUs longas (LACHELLO L. et al.). Como ilustrado na figura 5.1, uma mensagem PROFIsafe é composta pelos dados a serem transferidos, 1 *byte* de controle ou de *status* e uma assinatura CRC. Caso os dados (F-Input/F-Output) tenham um tamanho máximo de 12

bytes, então, é necessária uma assinatura CRC de 3 *bytes* para garantir a integridade dos dados (CRC 24). Caso o tamanho máximo seja superior a este, de até 123 *bytes*, então, o CRC necessário é de 4 *bytes* (CRC 32). Se a mensagem for oriunda do F-Host, o *byte* que segue os dados é um *Control Byte* (*byte* de controle), já se vier de um F-Device é um *Status Byte* (*byte* de *status*). Esse *byte* de controle/*status* serve para sincronizar o emissor e o receptor das mensagens PROFIsafe (IEC 61784-3-3, 2010).

Figura 5.1 – Formato de uma mensagem PROFIsafe



Fonte: Norma (IEC 61784-3-3, 2010).

5.1.2 Drivers

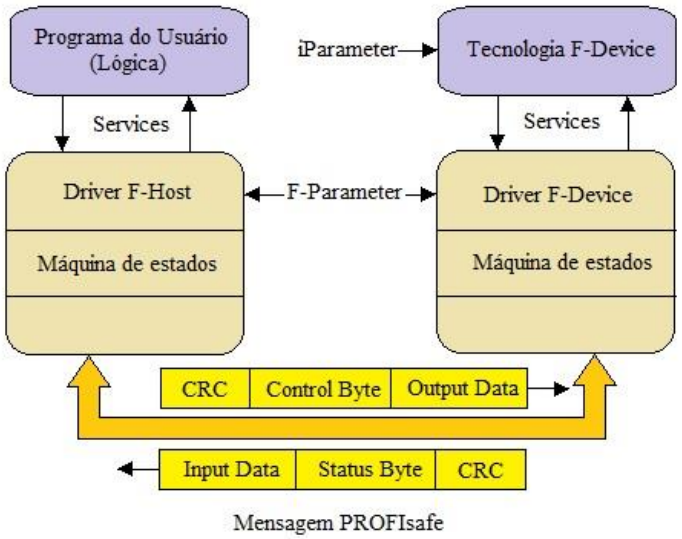
A partir da reflexão que consta em (PROFIBUS Nutzerorganisation e. V., 2010), compreende-se que as camadas PROFIsafe, onde se encontram os emissores e receptores das mensagens *safety* são realizadas em *software*, através de *drivers*. Ainda que os endereços (codinomes únicos) sejam passados automaticamente para os F-Devices, os endereços alvo (destino) precisam ser ajustados diretamente no dispositivo através de *switches* DIP (Dual In-line Package). Os F-Devices são configurados através da transferência do F-Parameter via GSD (General Station Description) e do iParameter (individual F-Device Parameter). Esses parâmetros são gerenciados com o servidor iPar, de onde eles podem ser transferidos para um dispositivo PROFIsafe utilizando interfaces padronizadas. Frequentemente, o Servidor iPar vem integrado em uma ferramenta de engenharia “CPD-Tool” (collaborative product design) (LACHELLO L. et al.). A figura 5.2 mostra como a camada PROFIsafe interage com a

tecnologia nos F-Devices e o programa do usuário nos F-Hosts, sendo descrita abaixo sua estrutura como ilustrado na imagem – tal descrição foi construída a partir dos dados obtidos em (PROFIBUS Nutzerorganisation e. V., 2010).

- ***F-Host Services*:** fornecem a troca dos dados F-Input/FOutput. Os valores de processo vigentes são substituídos por valores *default* de *fail-safe* durante a inicialização ou em caso de erros. Para forçar o receptor a um estado seguro (estado de desenergização), estes valores de segurança (*fail-safe*) devem ser todos "0". Como a desenergização não é o único estado seguro possível para os F-Devices, mas sim uma velocidade baixa, o PROFIsafe fornece serviços adicionais mediante o uso de um *flag* activate_FV no *Control Byte*. Em contrapartida, o F-Device pode informar ao programa do usuário que ativou o seu estado seguro através da *flag* FV_activated no *Status Byte*. Os erros de comunicação PROFIsafe fazem com que o *driver* F-Host mude para um estado seguro. Normalmente, não é permitido que uma função de segurança mude automaticamente de um estado seguro para um estado de operação normal sem que haja interação humana. Por isso, para informar ao programa do usuário que uma intervenção de um operador e uma confirmação estão sendo solicitadas, o PROFIsafe fornece um outro serviço, denominado OA_Req, informando o F-Device sobre uma solicitação pendente. O F-Device sinaliza a requisição e o operador confirma, passando do programa do usuário para o *driver* do F-Host esta confirmação (*ack*) por meio de um serviço correspondente, o OA_C. Além disso, os parâmetros específicos da tecnologia de um F-Device são os denominados iParameters. Se um F-Device precisar de iParameters diferentes em tempo de execução, o serviço iPar_EN permite que o programa do usuário mude o F-Device para um modo durante o qual ele aceitará iParameters novos. Já se o programa do usuário deseja retomar a operação normal de segurança, o serviço iPar_OK indica para o programa do usuário a prontidão para retornar a este estado.
- ***F-Device Services:*** além dos serviços supracitados em que os serviços do F-Device atuam (troca de dados F-Input/F-Output, ativação dos valores *fail-safe*, os indicadores dos iParameters, assim como as requisições dos operadores), a tecnologia F-Device relata falhas de dispositivo para o *driver* F-Host através do *flag* Device_Fault no *Status Byte*. Ademais, a tecnologia possibilita a transferência dos F-Parameters para a camada PROFIsafe, sendo que o F-Device recebe estes F-Parameters com todos os outros parâmetros durante a inicialização.

- ***F-Parameter:*** ajusta o comportamento da camada PROFIsafe às necessidades do cliente e verifica a exatidão das atribuições passando as informações essenciais para que a camada realize tais funcionalidades. Dentre os F-Parameters que mais se destacam estão: o *F_S/D_Address* (*short F-Address*) para garantir a autenticidade da conexão; o *F_WD_Time* que atribui o valor do *Watchdog Timer*; o *F_SIL* cuja função é indicar o SIL esperado pelo usuário para o F-Device em questão; o *F_iPar_CRC* sendo a assinatura em todos os iParameters de um F-Device; e o *F_Par_CRC* para garantir a entrega correta dos F-Parameters pois atua como uma assinatura em todos os F-Parameters.
- ***iParameter:*** são os parâmetros específicos da tecnologia F-Device que, como citado anteriormente, podem ser solicitados e confirmados, também em tempo de execução, pelos F-Device Services através das *flags* iPar_EN e iPar_OK.

Figura 5.2 – Fluxo da mensagem PROFIsafe



Fonte: Adaptado de (PROFIBUS Nutzerorganisation e. V. 2010).

5.2 Funções principais

A implementação de um *driver* PROFIsafe desde o princípio pode ser complexa, principalmente em relação à obtenção da certificação almejada. Por isso, utilizar um *kit* de desenvolvimento como os disponíveis no mercado tem certas vantagens, tais como a de obter um *software* de *driver* pré-certificado, com suporte técnico e ferramentas adicionais. Assim, este capítulo baseia-se no manual do PROFIsafe Driver (PSD) fornecido pela SIEMENS em

(SIEMENS, 2009). Nele está detalhado tudo o que é necessário para integrar o *driver* corretamente de maneira a obter-se uma comunicação segura (*safety*).

O ciclo de um PSD começa inicializando todas as instâncias do PSD através da função *psd_InitInstance(...)*. Se a execução retornar algum erro, a instância PSD passa para o estado **PSD_HARD_FAIL**, fechando o ciclo, sendo necessário inicializá-lo novamente. Caso contrário passa para o estado **PSD_INIT**. Em seguida, a aplicação F-Device indica o recebimento dos F-Parameters através de uma *flag* e atribui os parâmetros para a instância PSD chamando a função *psd_FParBuild(...)*. Se a função detectar F-Parameters inválidos, retorna ao estado anterior (**PSD_INIT**). No entanto, se os F-Parameters válidos estiverem disponíveis, passa para o estado **PSD_PARAM**. Caso a aplicação F-Device receba novos F-Parameters válidos, a mesma atribui estes novos parâmetros para a instância PSD. Ao receber a configuração do tamanho dos dados do usuário, a instância define o tamanho para os dados de usuário de entrada e saída. Quando a *flag* RUN for ativada, é indicada a transição para a troca cíclica de dados e a aplicação F-Device inicia a função *psd_Run(...)*, passando para o estado **PSD_DATAEX**. Neste estado começa a comunicação PROFIsafe de fato, inicializando, assim, a máquina de estado PROFIsafe. Para parar a execução é utilizada a função *psd_Stop(...)*, indicando a interrupção da troca cíclica de dados e retornando ao estado **PSD_PARAM**. A tabela 5.1 indica as funções necessárias para a interface PSD.

Tabela 5.1 – Funções principais da pilha PROFIsafe

Funções principais	*Descrição*
psd_InitInstance(...)	Inicializa a instância e verifica a passagem dos parâmetros
psd_FParBuild(...)	Configura o PSD, verifica a passagem dos parâmetros, lendo e verificando os F-Parameters
psd_Config(...)	Configura o tamanho dos dados de uma instância PSD
psd_Run(...)	Posiciona a instância PSD na troca cíclica de dados
psd_Stop(…)	Finaliza a troca cíclica de dados do PSD e permite parametrização
psd_RecvFOutTele(…)	Aceitação do telegrama de saída cíclica no *buffer* PSD_zykout e incrementos de 1ms
psd_RecvFOutTele_Invers(…)	Utilizada na versão *single-channel*. Realiza a aceitação dos incrementos inversos de 1ms e acumulação dos dados inversos no *buffer* PSD
psd_GetFOutData(…)	Verifica o telegrama de saída, configurando os dados do usuário F-output, assim como o *byte* de controle
psd_GetFOutData_Invers(…)	Faz o mesmo que a função anterior, porém com os dados e o *byte* de controle inversos. Função utilizada nas versões *single-channel*.

psd_SetFInData(...)	Transferência dos dados do usuário F-input e dos IDs dos *bytes* de *status* para um *buffer* PSD interno
psd_SetFInData_invers(...)	Utilizada em versões *single-channel*, a função neste caso, transfere os dados do usuário F-input e os Ids dos *bytes* de *status* inversos em um *buffer* PSD interno
psd_SendFInTele(...)	Gera e envia o telegrama de entrada (input telegram)
psd_SendFInTele_invers(…)	Gera e envia o telegrama de entrada inverso

Existem, também, funções a serem implementadas, as quais inicialmente contêm somente o escopo, mas, na realidade, são fundamentais para o *driver* PSD, devendo ser desenvolvidas para o funcionamento correto do *driver*. Tais funções estão listadas na tabela 5.2.

Tabela 5.2 – Funções do *driver* PROFIsafe a serem implementadas

Funções principais	*Descrição*
hard_err(...)	Reação específica da F-Application para erros fatais
psd_GetFParPtr(...)	O PSD solicita um ponteiro para os F-Parameters da instância
psd_GetConfigPtr(...)	Solicita um ponteiro para o tamanho dos dados da instância
psd_GetOutputTelegram(...)	Ponteiro para a saída dos dados da instância
psd_GetInputBuffer(…)	Ponteiro para a área dos dados de entrada da instância
psd_InputSendAck(…)	O PSD informa que os dados de entrada foram inseridos na área de dados de entrada da instância
enable_int(...)	O PSD informa que a trava de interrupção da instância pode ser removida novamente
disable_int(…)	Solicita o travamento das interrupções da instância pois está sem acesso aos dados

6 TRABALHO EXPERIMENTAL

Este capítulo descreve os passos realizados no âmbito da avaliação de desempenho do protocolo PROFIsafe em um ambiente experimental real. Primeiramente, foi elaborado um modelo analítico das informações, onde foram obtidos os dados teóricos que embasaram a análise. Em seguida, foi montado o ambiente para a realização dos experimentos acerca do estudo. E, por fim, são descritos todos os testes realizados e os resultados aferidos.

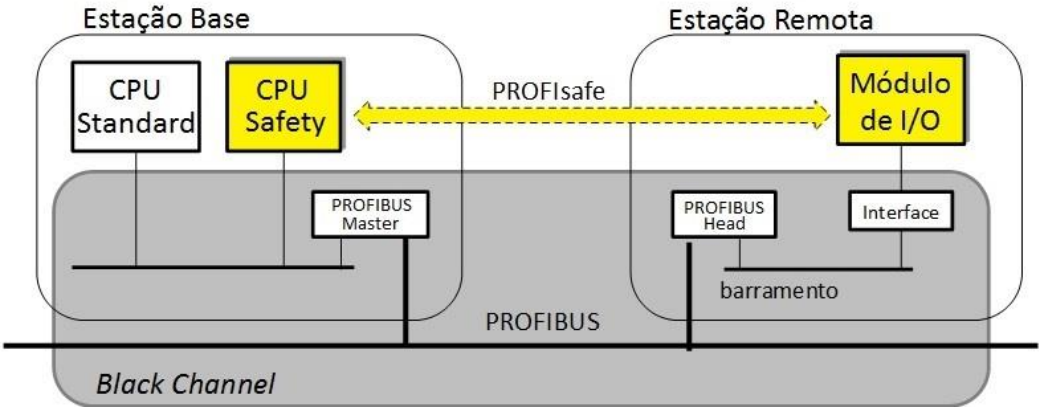
O objetivo deste trabalho experimental é avaliar os tempos de resposta da comunicação segura do protocolo PROFIsafe. Tais medidas são importantes pois refletem o desempenho do protocolo, verificando se este atende aos requisitos da especificação. Este experimento complementa o estudo realizado acerca dos protocolos, pois envolve a compreensão e aplicação dos conceitos estudados em um ambiente real.

6.1 Modelo Analítico

O modelo analítico pode variar, dependendo do fabricante do módulo *safety*, pois sua arquitetura pode ser estruturada de maneira distinta. Este trabalho baseia-se na arquitetura do modelo da Altus para comunicação segura, a qual utiliza CPU Standard Master, PROFIBUS Master, CPU Safety, PROFIBUS Head e os módulos de I/O (Slaves).

A figura 6.1 ilustra esta arquitetura, a qual podemos dividir em três partes principais: a Estação Base, o Black Channel e a Estação Remota. A Estação Base é composta do mestre da comunicação (a CPU Standard), da CPU Safety e da PROFIBUS Master. O Black Channel, como explicado no capítulo 2.2, é a parte não segura onde trafega a informação. Ele que interliga a Estação Base com a Estação Remota, mediante o protocolo de transporte PROFIBUS, utilizando as duas extremidades de cada estação (PROFIBUS Master e PROFIBUS Head). Por fim, a Estação Remota é a que contém os módulos de I/O Safety que utilizam o protocolo PROFIsafe. No caso da Altus a comunicação entre a CPU Standard Master, o PROFIBUS Master e a CPU Safety é realizada via EtherCAT dentro do módulo da Estação Base, no entanto, isso não é uma regra, para demais fabricantes, como a Siemens, essa comunicação pode ser integrada, o que resultaria em outro tipo de análise.

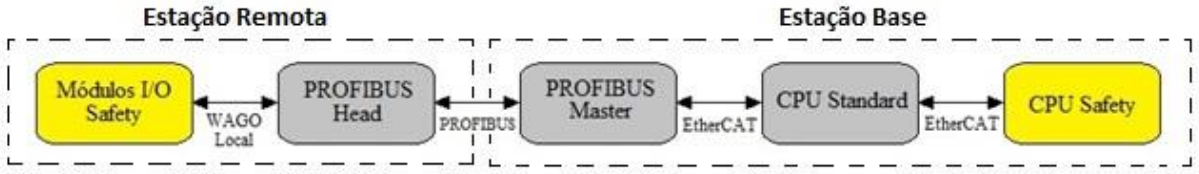
Figura 6.1 – Arquitetura do modelo de comunicação PROFIsafe



Fonte: Adaptado de (WEBER T. et al., 2017).

Com o intuito de avaliarmos os tempos de resposta em uma comunicação PROFIsafe, ou seja, o tempo entre um módulo de I/O *safety* e uma SCPU (Safety CPU), adotou-se um modelo, o qual está ilustrado no diagrama de blocos abaixo, na figura 6.2. A comunicação foi dividida em cinco blocos distintos. No módulo de I/O safety (escravos) o processamento da informação dá-se de forma cíclica em até 1ms. O PROFIBUS Head realiza leituras cíclicas, via barramento local, dos *frames* PROFIsafe contidos nos módulos I/O acoplados a ele, transferindo-os, via barramento PROFIBUS, ao PROFIBUS Master. Ele, então, é encarregado de transmitir os dados à CPU Standard, a qual atua como Master da comunicação como um todo. A CPU Standard é a responsável por enviar os *frames* para a CPU Safety. O caminho inverso é realizado quando devem ser transmitidos pacotes de confirmação do recebimento dos *frames* ou dados de *output*.

Figura 6.2 – Modelo de comunicação PROFIsafe proposto



A equação abaixo representa o modelo analítico do tempo de resposta em uma comunicação *safety*, ou seja, é o tempo que leva para a informação ser processada na CPU Safety, indo até nos módulos de I/O e retornando. Logo, o tempo começa a ser calculado a partir do momento em que o *frame* sai da CPU Safety em direção aos módulos de I/O e retorna para a mesma, ou seja, é a soma de duas vezes os tempos da CPU Standard (T_{CPU_STD}), da PROFIBUS Master (T_{PB_M}), da PROFIBUS Head (T_{PB_H}) e dos módulos de I/O (T_{IO}) mais

o tempo da CPU Safety (T_{CPU_S}) somado ao tempo que ela leva para executar o programa da aplicação ($T_{ProgSafety}$).

$$T_{Resposta} = 2 * (T_{CPU\_STD} + T_{PB\_M} + T_{PB\_H} + T_{IO}) + T_{CPU\_S} + T_{ProgSafety}$$

No entanto, para que este tempo seja válido, é preciso respeitar algumas condições. Toda aplicação/sistema que utiliza comunicação *safety* possui um tempo de PST (Process Safety Time). O PST é o tempo máximo permitido que uma aplicação pode levar para detectar uma falha e ativar uma função de segurança, desde o tempo para o sensor detectar a falha até o tempo de execução do mecanismo de segurança. Conforme descrito nas expressões abaixo, este tempo de PST deve ser o tempo de referência para todo o sistema seguro, pois o tempo de resposta do sistema como um todo (T_{Total}) deve ser menor do que o PST. O T_{Total}, como mostra a segunda expressão abaixo, abrange não só o $T_{Lógico}$ (tempo de processamento da CPU Safety mais tempos de *watchdog*), mas também os tempos do sensor (T_{Sensor}) e do atuador do mecanismo de segurança ($T_{Atuador}$).

$$T_{Total} \ll PST$$

$$T_{Total} = T_{Sensor} + T_{Lógico} + T_{Atuador}$$

Esse $T_{Lógico}$ refere-se ao tempo da CPU Safety (T_{CPU_S}) e aos tempos de *Watchdog* PROFIsafe dos módulos de entrada e saída ($WD_{Entrada}$ e $WD_{Saída}$). São estes tempos de *Watchdog* que definirão quais os valores máximos aceitos para o tempo de resposta ($T_{Resposta}$) que analisamos na equação que expressa o modelo analítico.

$$T_{Lógico} = WD_{Entrada} + T_{CPU\_S} + WD_{Saída}$$

$$WD_{PROFIsafe} = WD_{Entrada} = WD_{Saída}$$

Com isso, a partir dessas condições, podemos concluir que o tempo de resposta ($T_{Resposta}$) será válido sempre que for menor do que o tempo de *Watchdog* do PROFIsafe (ALTUS, 2015).

$$T_{Resposta} < WD_{PROFIsafe}$$

Para entender melhor como são analisados esses tempos de resposta, ponderou-se quais variáveis têm relação de dependência com os dados a serem obtidos e qual a relevância delas no resultado final. A ideia inicial era variar os tempos de ciclo e *watchdog* das CPUs (Standard e Safety) para que pudéssemos observar quais seriam os resultados. No entanto, de acordo com (PROFIBUS Nutzerorganisation e. V., 2010), o tempo de *watchdog* da CPU Safety é exatamente igual ao tempo de ciclo da sua CPU. Assim, não é necessário considerar

que os tempos de resposta sejam dependentes diretos dele, pois não podemos configurá-lo isoladamente visto que seu valor só é modificado se alterarmos o tempo de ciclo da CPU Safety (exemplificado com mais detalhes nos próximos subcapítulos). Todavia, precisamos levar em consideração, também, que a combinação das variáveis *baud rate* (taxa de transmissão do dados em bits por segundo) e número de escravos, conforme forem configuradas, podem interferir nos tempos de resposta. Portanto, as variáveis identificadas para a realização dos testes foram: o tempo de ciclo da CPU Standard, o tempo de *watchdog* da CPU Standard, o *baud rate*, o número de escravos utilizados e o tempo de ciclo da CPU Safety.

6.2 Arquitetura do ambiente experimental da avaliação

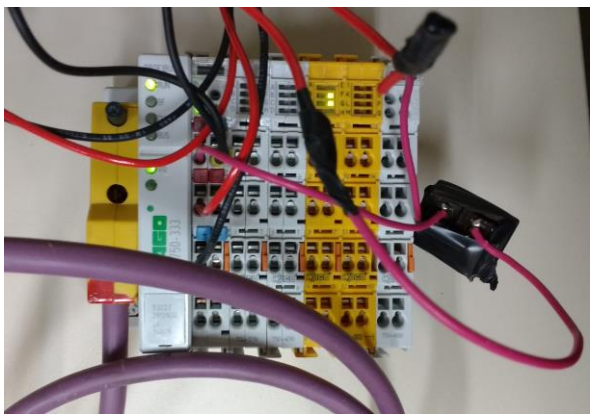
O ambiente experimental para a avaliação dos tempos de resposta foi montado com base no modelo de arquitetura para comunicação *safety* utilizado pela empresa parceira do grupo de pesquisa, a qual cedeu grande parte dos componentes descritos acima para a elaboração do ambiente. O ambiente é composto por um computador, dois sistemas de *software* empregados para a comunicação e análise do sistema, e os módulos da parte física esboçados na figura 6.2. O computador onde foi simulada a comunicação é um *Intel Core i5-6500* 3.20GHz com 16 GB de RAM e sistema operacional *Windows 10 Pro* 64 bits, estando conectado ao módulo mestre da parte física via comunicação direta ponto a ponto.

Primeiro, o sistema físico foi montado conforme o modelo citado anteriormente, dividido em 2 áreas, uma contendo o mestre da comunicação e a outra contendo os módulos de I/O (escravos). A figura 6.3 ilustra a parte do *hardware* que contém o mestre, onde temos um bastidor com 3 dispositivos acoplados a ele: o NX5210, mais à esquerda, que apresenta a conexão *ethernet* ponto a ponto com o computador, é a CPU Standard (Mestre); o NX5001 é o PROFIBUS Master, que realiza a comunicação do mestre com os escravos via barramento PROFIBUS; e a CPU Safety, mais à direita, que processa os frames PROFIsafe lidos.

Figura 6.3 – Bastidor contendo o mestre da comunicação

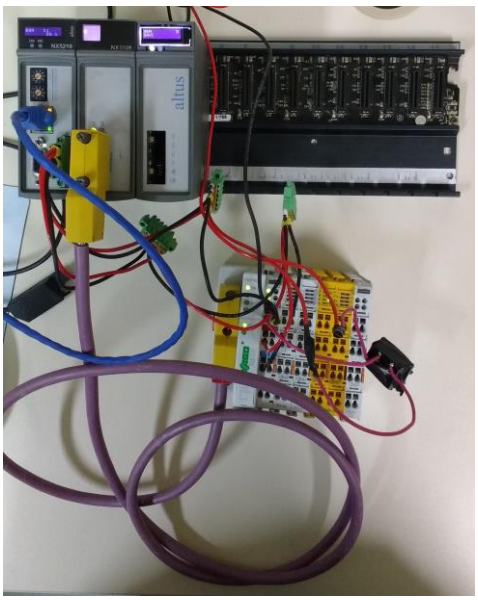
A figura 6.4 representa a parte do *hardware* referente aos módulos escravos da WAGO, onde temos: a PROFIBUS Head, que conecta todos os escravos ao mestre realizando a leitura cíclica de seus *frames* e transferindo-os via barramento; e os módulos de I/O que montam o *frame* a ser enviado. Nota-se que temos na imagem abaixo 4 LEDs acesos: os mais à esquerda, na PROFIBUS Head são para informar estado de RUN e I/Os conectados, já os mais à direita, no módulo de I/O *safety* indicados pelas letras F e G ao lado dos LEDs, correspondem respectivamente, às confirmações de que conexões PROFIBUS e com a CPU Safety estão ativas.

Figura 6.4 – Módulos escravos da comunicação



A figura 6.5 mostra, assim, o sistema físico por completo com os módulos mestre e escravos se comunicando entre si.

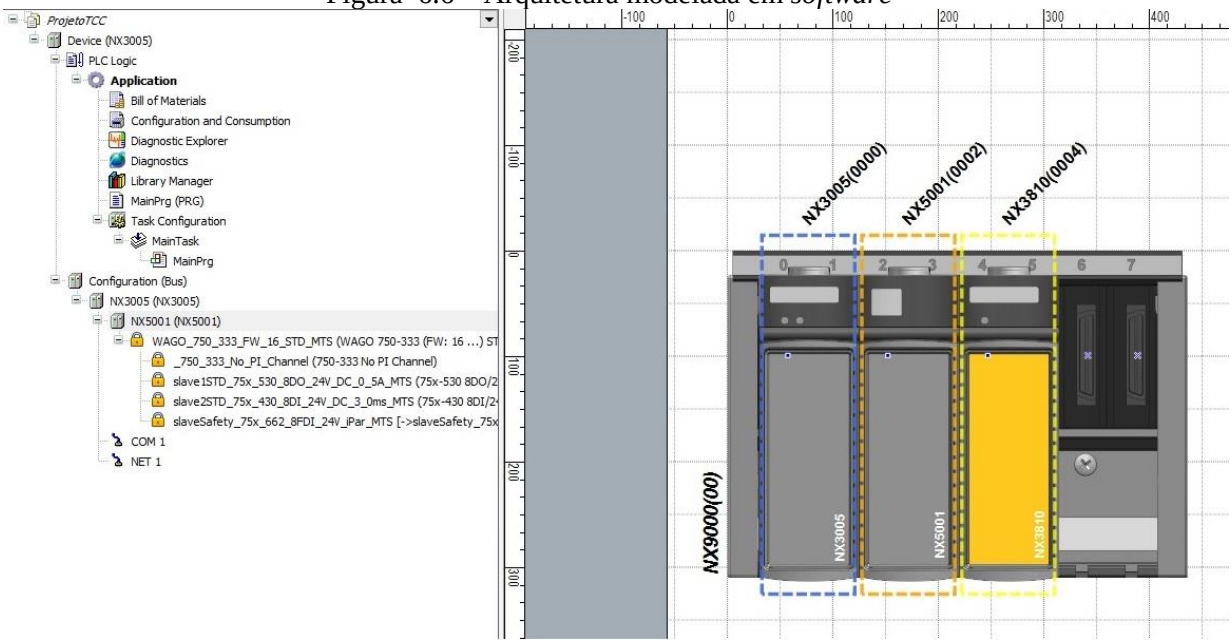
Figura 6.5 – Arquitetura completa



Mas a comunicação é estabelecida de fato e só é possível devido ao uso de 2 programas, o *Master Tool Standard* e o *Master Tool Safety*, que configuram em *software* todo o sistema físico construído. Esses programas permitem estruturar a comunicação segura que será estabelecida entre as estações (base e remota). O *Master Tool Safety* tem por finalidade estruturar toda a parte *safety* da arquitetura, ou seja, nele são descritos, de maneira fiel, todos os módulos, principalmente os *safety*, existentes em hardware. O *Master Tool Standard* unirá as estações de trabalho no modelo mestre/escravo, através do comando de importação. Nesta aplicação, a arquitetura é modelada por completo, sendo ela, o ponto de partida da comunicação. Essas ferramentas são fundamentais neste trabalho, pois auxiliam nos experimentos e na configuração de que cada cenário de teste tratado.

A figura 6.6 mostra a arquitetura completa modelada em *software* no *Master Tool Standard*. O projeto em si é montado no *Master Tool Safety*, que possibilita a inserção dos módulos *safety* no sistema, para, então, ser importado no *Master Tool Standard* de onde a comunicação é estabelecida e os testes são iniciados. Nela, enxergamos exatamente o modelo que foi proposto, onde o NX3005 é a CPU Standard, o NX5001 é o PROFIBUS Master e o NX3810 é a CPU Safety, formando a estação base definida anteriormente. Já a estação remota com os módulos de I/O e o PROFIBUS Head estão apresentados à esquerda na figura 6.6. Onde o WAGO_750_33 é o PROFIBUS Head que conecta a estação remota à estação base no *Black Channel* via PROFIBUS. E o os que estão definidos abaixo dele (os *slaves*) são os módulos de I/O da comunicação, importados para esta ferramenta.

Figura 6.6 – Arquitetura modelada em *software*



6.3 Testes realizados

Com base no modelo analítico foram identificadas cinco variáveis que modificam o tempo de resposta: o tempo de ciclo da CPU Standard, o tempo de *watchdog* desta CPU, o *baud rate*, o número de escravos utilizados e o tempo de ciclo da CPU Safety. São elas que influenciam no resultado dos tempos de resposta em uma comunicação *safety*. Mediante o uso do *Master Tool Standard* e do *Master Tool Safety* foi possível medir tais tempos alterando os valores destas variáveis. Após a modelagem do sistema em ambas as ferramentas de *software*, optou-se por dividir os testes em dois (determinação do *baud rate* e análise dos tempos de resposta em relação aos tempos de ciclo), facilitando muito a análise.

A metodologia de testes adotada a partir desta divisão da análise para avaliarmos os tempos de resposta foi: primeiramente, variar os tempos de *baud rate* e o número de escravos para determinar qual seria o melhor *baud rate* para empregarmos na análise dos tempos de resposta da comunicação segura; e, em seguida, variar somente os tempos de ciclo e de watchdog das CPUs para que pudêssemos analisar de fato os tempos de resposta da comunicação a cerca do PROFIsafe.

6.3.1 Determinação do *baud rate*

Para verificar quais os tempos de resposta obtidos conforme a configuração dos tempos de ciclo e *watchdog* das CPUs (Standard e Safety), era preciso, primeiro, definir o *baud rate* que seria utilizado nesta investigação. Sabendo que o *baud rate* influencia bastante nos tempos de resposta dependendo do número de escravos contidos no sistema, foi analisado, primeiramente, o *baud rate* em relação ao número de escravos.

A tabela 6.1 mostra como foram configuradas as ferramentas de *software* para que fosse permitida a análise somente do *baud rate*. No *Master Tool Safety* configurou-se o tempo de ciclo da CPU Safety para um valor fixo de 20ms e, como explicado anteriormente, o tempo de *watchdog* foi automaticamente configurado com o mesmo valor do seu tempo de ciclo. Após a importação do projeto no *Master Tool Standard*, configurou-se o tempo de ciclo da CPU Standard (Mestre) para 20ms e o seu tempo de *watchdog*, devido às configurações do *software* que utiliza os padrões do mercado e é automaticamente definido com 1000ms.

Tabela 6.1 – *Set up* para a análise do *baud rate*

	Set up - Análise do Baud Rate	
CPU Standard	Tempo de Ciclo (ms)	20
	Watchdog Time (ms)	1000
CPU Safety	Tempo de Ciclo (ms)	20
	Watchdog Time (ms)	20

Então, foram observados três valores de *baud rate*, o mínimo e máximo existentes e um valor médio em relação a outros três valores de número de escravos – variáveis configuradas diretamente no *Master Tool Standard*. A tabela 6.2 apresenta os resultados alcançados após a execução de 3 experimentos de aproximadamente 2min para a configuração de cada linha da tabela. O tempo de análise para cada experimento foi definido em 2min, pois observou-se que abaixo deste tempo o resultado é instável e próximo dos 2min em diante o resultado não varia mais. Além disso, repetimos cada experimento 3 vezes, pois vimos que era uma amostra suficiente para verificarmos a consistência dos resultados. Visto que, considerando os tempos de ciclo das CPUs e o intervalo dos experimentos, foram executadas cerca de 2000 amostras em cada um dos 3 experimentos para cada linha configurada da tabela.

Os dados obtidos na tabela 6.2 possibilitaram a análise do *baud rate*. Quando utilizamos um *baud rate* muito baixo, de 9.6 kBits/s, percebemos a variação dos tempos de resposta em relação ao número de escravos. Quanto maior o número de escravos utilizados, maiores são os tempos de resposta da CPU Safety. Contudo, nota-se que ao utilizar um número de escravos um pouco maior, no caso 10, por exemplo, o *baud rate* mínimo de 9.6 não consegue mais dar vazão, ou seja, apresenta o erro 49410 de w*atchdog* do PROFIsafe (F_WD), colocando o sistema em um estado seguro.

Inicialmente, ao empregarmos 10 escravos com 9.6kBits/s de *baud rate*, o *watchdog* do PROFIsafe (F_WD) padrão configurado era de 550ms, resultando no erro supracitado. Logo, a fim de analisarmos esse tempo de *watchdog* do PROFIsafe, alterou-se o *baud rate* para 19.2kBits/s (segundo valor mais baixo existente) mantendo o tempo de F_WD em 550ms, o que resultou em uma execução normal, não havendo nenhum erro. Com isso, como podemos observar na tabela 6.2, o tempo de resposta máximo da CPU Safety no caso do *baud rate* de 19.2kBits/s foi de 437ms. Assim, o tempo de F_WD necessário para o mesmo número de escravos com *baud rate* de 9.6kBits/s seria, teoricamente, de 2 vezes esse tempo de resposta máximo, neste caso, de aproximadamente 950ms. Deste modo, foi realizada mais uma tentativa para a taxa de *Baud* de 9.6kBits/s, porém agora com o F_WD de 950ms, mas o resultado não foi o esperado, pois continuou apresentando erro, mesmo se aumentássemos mais ainda o valor de F_WD.

Isso nos permite concluir que taxas de *Baud* mínimas não suportam um número de escravos elevado. No entanto, a partir da taxa de 1500kBits/s, independentemente do número de escravos definidos nos experimentos, os tempos de resposta mínimos e máximos da CPU Safety praticamente não se alteram. Portanto, a taxa de *Baud* escolhida para a análise dos tempos de resposta em relação à variação dos tempos de ciclo das CPUs é de 12000kBits/s (valor referência no mercado atual).

Tabela 6.2 – Análise da taxa de *Baud* em relação ao número de escravos

	Análise de Desempenho - PROFIsafe - Taxa de Baud					
	CPU Standard				CPU Safety	
Nº de escravos	Taxa de Baud (kBits/s)	Resp. T. Mín. (ms)	Resp. T. Máx. (ms)	Máx. Jitter (ms)	Resp. T. Mín. (ms)	Resp. T. Máx. (ms)
1	9.6	2.953	10.019	1.030	122	229
1	1500	2.901	9.888	0.981	40	63
1	12000	2.885	8.007	1.182	40	63
5	9.6	3.024	8.594	0.898	408	516
5	1500	2.998	8.738	0.870	40	84

5	12000	3.045	9.996	1.006	40	64
10	9.6	3.546	12.639	0.964	Erro 49410 - F_WD	Erro 49410 - F_WD
10	19.2	3.557	10.397	0.989	247	437
10	9.6	3.085	10.005	0.840	Erro 49410 - F_WD	Erro 49410 - F_WD
10	1500	3.078	10.003	0.802	40	84
10	12000	3.097	8.170	0.823	40	83

6.3.2 Análise dos tempos de resposta em relação aos tempos de ciclo das CPUs

Com a taxa de *Baud* definida, configuraram-se as ferramentas conforme apresenta a tabela 6.3. No *Master Tool Standard* mantivemos um único escravo e taxa de *Baud* de 12000kBits/s. O tempo de *watchdog* da CPU Standard (Mestre), como explicado anteriormente, está fixo em 1000ms de acordo com as configurações do *software*.

Tabela 6.3 – *Set up* da análise dos tempos de resposta

	Setup - Análise dos Tempos de Resposta	
CPU Standard	Watchdog Time (ms)	1000
	Taxa de Baud (kBits/s)	12000
	Nº de escravos	1

Em seguida, pudemos dar início à avaliação dos tempos de resposta a partir da variação dos tempos de ciclo e *watchdog* das CPUs (Standard e Safety). A tabela 6.4 indica os resultados aferidos para cada configuração. Estipulou-se quatro combinações diferentes para os tempos de ciclo, alternando-os entre valores pequenos, médios e grandes. Nota-se que os tempos de resposta mínimos da CPU Safety são equivalentes aos seus tempos de ciclo quando o tempo de ciclo da CPU Standard (Mestre) é menor que o tempo de ciclo da CPU Safety. E os tempos de resposta máximos da comunicação *safety* são bem pequenos. Contudo, ao utilizarmos valores menores para os tempos de ciclo da CPU Safety que os tempos da CPU Standard o mesmo não ocorre. Tanto os tempos de resposta mínimos quanto os máximos são bem elevados.

Tabela 6.4 – Análise dos tempos de resposta em relação aos tempos de ciclo

Análise de Desempenho - PROFIsafe - Tempo de Ciclo							
CPU Standard	CPU Safety		CPU Standard			CPU Safety	
Tempo de Ciclo (ms)	Tempo de Ciclo (ms)	Watchdog Time (ms)	Resp. T. Mín. (ms)	Resp. T. Máx. (ms)	Máx. Jitter (ms)	Resp. T. Mín. (ms)	Resp. T. Máx. (ms)
20	**50**	**50**	2.878	10.041	0.983	50	52
20	**100**	**100**	2.879	12.785	0.917	100	102
50	**20**	**20**	2.875	8.086	0.8	81	124
100	**20**	**20**	2.884	8.103	0.783	164	210

Assim, podemos concluir que a melhor configuração para obtermos tempos de resposta mais rápidos em uma comunicação segura é quando utilizamos tempos de ciclo da CPU Safety maiores que os tempos de ciclo da CPU Standard. Esta análise, dos tempos de resposta em uma comunicação segura usando o protocolo PROFIsafe, nos mostra o porquê do protocolo ser o mais utilizado na indústria. Devido ao acesso privilegiado que tive à documentação do PROFIsafe pela parceria com a empresa ALTUS, foi possível verificar que o protocolo reflete o esperado, as medidas são correspondentes às previstas no modelo analítico, a documentação é vasta e de fácil entendimento.

7 CONCLUSÃO

Neste trabalho, inicialmente, foram abordados cinco protocolos de comunicação segura, seus conceitos e diferenças; porém, o enfoque foi em dois deles, o openSAFETY e o PROFIsafe. Após os problemas apresentados pelo protocolo de código aberto, o estudo foi direcionado para o protocolo de comunicação segura que mostrou-se mais interessante nos aspectos mais importantes para o mercado, o PROFIsafe.

O enfoque no PROFIsafe resultou em uma análise do protocolo em relação ao seu desempenho, onde foram avaliados os tempos de resposta mínimos e máximos obtidos em uma comunicação segura. A empresa ALTUS, parceira do grupo de pesquisa, possibilitou a observação do comportamento do protocolo em um ambiente real de automação industrial através dos módulos cedidos (mestre e escravos da comunicação). As medidas resultantes da análise mostraram que o modelo analítico teorizado reflete o que foi alcançado na prática: onde o tempo que uma mensagem em uma comunicação segura leva para ser processada, encapsulada e transmitida depende de uma série de fatores. Os tempos de resposta em uma comunicação segura variam dependendo da taxa de *baud* utilizada, do número de escravos empregados na comunicação, do tempo de ciclo da CPU Standard (mestre da comunicação) e seu respectivo tempo de *watchdog* estipulado, e do tempo de ciclo da CPU Safety que realiza a comunicação segura de fato. O experimento nos programas *Master Tool Standard* e *Master Tool Safety* apresentaram amostras confiáveis e com valores quase sempre iguais. Os tempos de resposta apresentados são rápidos e o protocolo atende fielmente aos preceitos das normas para uma comunicação *safety*. No entanto, ao configurarmos as ferramentas com a taxa de *baud* mais utilizada no mercado, de 12000 kBits/s para 1 escravo, notou-se que utilizando tempos de ciclo da CPU Safety maiores que os tempos de ciclo da CPU Standard obtêm-se tempos de resposta mais rápidos em uma comunicação segura. O protocolo PROFIsafe vem sendo o mais requisitado pela indústria, pois além de atender a todas as exigências do mercado, ganha em relação a confiabilidade, garantia e certificação.

Para trabalhos futuros, sugerimos a realização de uma avaliação de desempenho semelhante a esta realizada, porém utilizando o injetor de falhas FITT (DOBLER, R. J. et al, 2016) para PROFIsafe e o injetor de falhas TANATTOS (GOMES L. G. A., 2016) para avaliarmos os tempos de resposta do protocolo PROFIsafe em um cenário onde ocorrem falhas. Desta forma seria possível medir se haveria queda de desempenho sob falhas e também a cobertura dos mecanismos de tolerância a falhas implementados no protocolo para garantir sua segurança.

REFERÊNCIAS

ALTUS. **Nexto Safety User Manual**, jul 2015. Disponível em: <www.altus.com.br>. Acesso em: 12 nov. 2016

ALTUS; UFRGS. **NX2800 - Nexto Safety Digital Output Module**, set. 2015. Disponível em: <www.altus.com.br>. Acesso em 12 de nov. 2016

AVIZIENIS, A. et al. Basic concepts and taxonomy of dependable and secure computing, **IEEE transactions on dependable and secure computing**, v. 1, n. 1, p. 11–33, 2004.

BECKMANN, G. **Overview Safety over EtherCAT**, 2016. Disponível em: <http://www.ethercat.org/pdf/english/Safety_over_EtherCAT_Overview.pdf>. Acesso em: 12 nov. 2016.

DOBLER, RODRIGO J.; CECHIN, SERGIO; WEBER, TAISY; NETTO, JOAO. A Software Fault Injector to Validate Implementations of a Safety Communication Protocol. In: **2016 Seventh LatinAmerican Symposium on Dependable Computing (LADC), 2016, Cali.** 2016 Seventh Latin-American Symposium on Dependable Computing (LADC), 2016. p. 35.

ELIA, A. et al. Analysis of Ethernet-based safe automation networks according to IEC 61508, **IEEE Conference on Emerging Technologies and Factory Automation**, 2006. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4178252>, Acesso em: 12 Nov. 2016.

EPSG - The Black Channel Principle. **EPSG**, 2016. Disponível em: <http://www.ethernet-powerlink.org/en/powerlink/industrial-ethernet-facts/safety-functionality/the-black-channel-principle/>. Acesso em: 12 nov. 2016.

Ethernet POWERLINK Facts - **The Magazine for the Industrial Ethernet Standard - Powerlink Safety - Open Safety Technology,** v. 4, p. 12, out. 2009.

Ethernet POWERLINK Standardisation Group. **EPSG_WDP_304_V-1-4-0 .pdf**, 2013.

FELSER, M. e SAUTER, T. Standardization of industrial ethernet-the next battlefield? Em **IEEE International Workshop on Factory Communication Systems**, 2004. Proceedings. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1377762>, Acesso em 12 nov. 2016.

GOMES L. G. A. (2016). Injeção de falhas de comunicação sobre implementações do protocolo EtherCAT, **UFRGS**, 2016.

IEC 61784-3-3. International Electrotechnical Commission. "IEC 61784-3-3 - Industrial Communication Networks - Functional Safety Fieldbuses". **IEC**, 29 jun. 2010.

LACHELLO L. et al. Industrial Ethernet Facts - The 5 Major Technologies. **EPSG**.

LIU, Y. et SONG, Y. EtherCAT-based functional safety-integrated communication, **Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on**. IET, 2012. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber= 6492753>. Acesso em: 12 nov. 2016.

NEUMANN, P. Communication in industrial automation—What is going on?, **Control Engineering Practice**, v. 15, n. 11, p. 1332–1347, nov. 2007.

openSAFETY: openSAFETY Software Concept. **openSAFETY**. Disponível em: <http://opensafety.sourceforge.net/doc/html/page_swconcept.html>. Acesso em: 17 jun. 2017.

openSAFETY: openSAFETY Software Structure. **openSAFETY**. Disponível em: <http://opensafety.sourceforge.net/doc/html/page_swstructure.html>. Acesso em: 17 jun. 2017.

PROFIBUS Nutzerorganisation e. V. **PROFIsafe System Descriptor**, nov. 2010.

PROFIsafe. **PROFIBUS**, 2017. Disponível em: <http://www.profibus.com/technology/profisafe/>. Acesso em: 17 jun. 2017.

SAFETY. **Sercos**, 2016. Disponível em: <http://www.sercos.org/technology/safety/>. Acesso em: 12 nov. 2016.

Safety_over_EtherCAT_Overview.pdf. **EtherCAT Technology Group (ETG)**, [s.d.]. Disponível em: <http://www.ethercat.org/pdf/english/Safety_over_EtherCAT_Overview.pdf>. Acesso em: 12 nov. 2016.

SIEMENS. **PROFIsafe driver V2.1 for F-Slaves**, jan. 2009.

SMITH, D. J. e SIMPSON, K. G. L. **Functional Safety: A Straightforward Guide to Applying IEC 61508 and Related Standards. Routledge**, 2004.

VASKO, D. A. et AUTOMATION, R. **DeviceNet Safety**: Safety networking for today and beyond, PUB 00110R0. ODVA, 2005.

VIDAL, W. et al. Software Architecture for Safety Communication in Critical Systems. **IEEE**, mar. 2014. Disponível em: <http://ieeexplore.ieee.org/document/7469506/>. Acesso em: 12 nov. 2016.

WEBER, TAISY ; CECHIN, S. L. ; DE MORAES, JOAO ; NETTO, JOAO C. . Arquitetura de hardware de uma estação remota de entrada analógica em conformidade com a IEC 61508. In: **XVIII Workshop de Testes e Tolerância a Falhas (WTF-SBRC 2017)**, 2017, Belém - PA. XVIII Workshop de Testes e Tolerância a Falhas. Porto Alegre : SBC - Sociedade Brasileira de Computação, 2017.

ZURAWSKI, R. **Industrial Communication Technology Handbook**, Second Edition. CRC Press, 2014.

APÊNDICE A – TRABALHO DE GRADUAÇÃO I

Comunicação segura para aplicações de automação industrial

Victoria R. Pires[1], Taisy S. Weber[1], João de Moraes[1]

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

{vrpires, taisy, joao.moraes}@inf.ufrgs.br

Abstract. *Ensuring the safe operation of safety functions is essential in any industry. In this way, it was developed a serie of safety communication protocols which must to be certified by the regulamentary agencies, proving that they support the SIL (Safety Integrity Levels) as described in the functional safety standard IEC 61508. However, the obligation of implementing the protocol chosen for any equipment that will use it, makes the process costly. The objective of this work is to highlight how these protocols are implemented according to the needs of the industry, what are the characteristics, advantages and limitations that each one has.*

Resumo. *Garantir o funcionamento seguro de funções de segurança é imprescindível dentro de qualquer indústria. Por esta razão, foi desenvolvida uma série de protocolos de comunicação segura, os quais devem obter certificação exigida pelos órgãos reguladores, comprovando que os mesmos atendem aos níveis seguros de integridade SIL (Safety Integrity Levels) de acordo com a norma de segurança funcional IEC 61508. Porém, a necessidade de implementar o protocolo escolhido para quaisquer equipamentos que o utilizarão, torna o processo um tanto quanto custoso. O objetivo deste trabalho é ressaltar como estes protocolos são implementados conforme a necessidade da indústria, quais as características, vantagens e limitações que cada um possui.*

1 Introdução

A demanda por sistemas de comunicação industriais que suportem aplicações seguras vem crescendo diariamente. A automação industrial tem sido uma das áreas mais promissoras no campo da tecnologia, utilizando equipamentos cada vez mais robustos. Contudo, realizar a comunicação destas máquinas e serviços de maneira segura é uma tarefa árdua. Transportar as informações de forma segura é um ponto crítico em diversos setores da indústria. Por sistema seguro, entende-se detectar e prevenir falhas de processos críticos, controlando possíveis acidentes e danos, iniciando procedimentos emergenciais, como desligamentos e isolamentos instantâneos. Assim, ao detectar uma falha, o sistema deverá ser direcionado para um estado seguro ou de parada, onde o tempo de reação é o tempo de reconhecimento de uma requisição de função de segurança, o qual é altamente dependente da taxa de transmissão, da confiabilidade da rede e da velocidade de resposta. Funções de segurança, tais como alarmes de incêndio, bloqueio preventivo do

funcionamento das máquinas, dentre outros, são aplicadas na automação industrial justamente para que as instalações garantam a execução destas funções, com um nível considerável de confiabilidade e disponibilidade, mediante o uso de técnicas de tolerância a falhas.

Nas aplicações industriais críticas são utilizados protocolos seguros de comunicação para a condução das informações de controle e monitoramento. No entanto, não existem protocolos seguros prontos para instalação. O que temos são especificações, normas que devem ser atendidas e que explanam como devem ser implementados e certificados para que possam ser considerados seguros. Todo protocolo de comunicação segura deve ser codificado e validado para cada equipamento que utilizá-lo. Muitas são as dúvidas relacionadas à não utilização da pilha TCP/IP para este tipo de comunicação. A resposta para isso é simples: esta pilha de protocolos não atende aos critérios das normas de segurança e, portanto, não pode ser utilizada para esta finalidade. A seguir, consta um estudo para identificar atributos relevantes para uma indústria que queira escolher um protocolo de segurança mais apropriado. Nele são abordados o conceito de *safety*, os diversos protocolos de comunicação segura existentes e suas particularidades, além das normas e certificações que regem as regras de implementação dos mesmos.

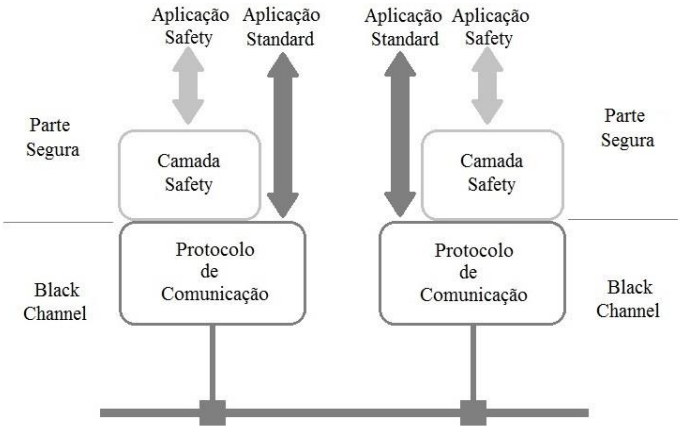
2 Safety

A diferença dos conceitos de *safety* e de *security* é um pouco confusa para muitos. Ambos significam segurança, contudo *security* está relacionada à proteção contra falhas maliciosas, no intuito de manter a privacidade, autenticidade, integridade e irrepudiabilidade dos dados. Já *safety*, tema principal deste artigo, garante que falhas não irão provocar danos ao sistema como um todo e nem às pessoas que o operam, sendo provável que o sistema esteja operacional e executando suas funções corretamente ou esteja descontinuando suas funções de forma a não provocar danos a outros sistemas ou pessoas que deles dependam [AVIZIENIS et al., 2004].

2.1 Black Channel

A fim de eliminar a preocupação com os canais físicos de comunicação, os quais poderiam interferir no protocolo de comunicação segura desenvolvido, através das taxas de transmissão e dos mecanismos de detecção de erros dos mesmos, foi desenvolvido o conceito de Black Channel. Nele, os canais físicos estariam isolados numa camada não segura destinada somente para os mecanismos de transporte dos dados, assim como dos protocolos seguros, porém sem a necessidade de certificação. A divisão da comunicação em duas partes distintas torna o sistema mais tolerante a falhas. O uso do Black Channel, desde que operando abaixo da taxa máxima de defeitos estipulada, garante a segurança implementada na camada acima, onde estão os protocolos que realizam a comunicação segura (estes sim, necessitando certificação). Ou seja, os dados seguros são transmitidos sem sofrer quaisquer alterações – mas podendo detectar erros como dado corrompido ou *frames* fora de ordem – de um dispositivo seguro até outro, sem a necessidade de se preocupar em como a transmissão é realizada, visto que comunicações *safety* e convencionais dividem o mesmo canal de comunicação [NEUMANN, 2007]. A Figura 1 exemplifica de maneira clara o funcionamento do princípio Black Channel.

Figura 1 – Conceito de Black Channel



Fonte: Adaptado de EPSG - The Black Channel Principle (2016).

2.2 Normas de segurança

As normas de segurança são as responsáveis por minimizar os riscos inerentes na indústria, exigindo o emprego de técnicas de prevenção e de tolerância a falhas, o projeto detalhado, além da documentação – em todas as fases do ciclo de desenvolvimento. São elas que permitem que uma agência certificadora valide os sistemas, assumindo que os mesmos respeitem todas as regras e estão verificados conforme requisitos. Existem diversas normas de segurança, porém a que atende melhor, de forma mais abrangente a implementação da comunicação segura em diversos tipos de equipamentos, é a IEC 61508, atingindo os níveis compatíveis com o das agências reguladoras. A aplicação da norma por si só não garante que o software seja realmente seguro. Para isso, além da norma bem empregada, é preciso que haja uma certificação (verificação e validação) do *software* desenvolvido, após a realização de todos os testes, como os unitários, de integração e injeção de falhas. Para tratar de segurança de uma maneira mais isolada, é realizada a separação dos sistemas de controle dos de segurança. Assim, as funções de segurança são executadas nos SISs (Sistemas Instrumentados de Segurança) que utilizam apenas comunicação segura. Desta forma, somente estas funções e a implementação do protocolo de segurança como um todo, têm a necessidade de serem certificados, resultando numa redução dos custos, isolamento de falhas e aumento da confiabilidade.

2.3 Certificações

Para que possamos garantir o nível de segurança de qualquer protocolo é preciso que haja um órgão certificador que certifique que a função de segurança atingiu o SIL almejado seguindo estritamente a norma de segurança adequada. Deste modo, uma das agências internacionais autorizadas pela IEC, analisa e certifica o nível de SIL (nível de integridade de segurança) alcançado em um dado protocolo. Avaliando todos os métodos utilizados para desenvolvimento e teste, como *hardware* e qualquer ferramenta de software aplicada, para assim, poder determinar se a implementação dada para tal protocolo alcança o SIL requerido.

A norma IEC 61508 apresenta 4 níveis de integridade para sistemas relacionados à segurança, os chamados "*Safety Integrity Levels*" (SILs). O mais alto nível contido no padrão é o SIL 4, aplicado em sistemas altamente críticos, tais como usinas nucleares. Entretanto, SIL 3 é o mais alto nível exigido pelas plantas industriais nas aplicações tradicionais de fabricação e processamento. O nível SIL 3 requer que a probabilidade de falha para desempenhar uma função de segurança no modo de Baixa Demanda de operação

seja ≥ 10-4 até < 10-3, enquanto que no modo de Alta Demanda ou no modo contínuo de operação a probabilidade deve ser ≥ 10-8 até < 10-7 [Felser, Max and Sauter, 2004].

3 Protocolos de comunicação segura

Implementar um protocolo seguro exige uma série de cuidados para que ele possa ser considerado confiável e seguro de fato. Assim, é preciso que o desenvolvimento atenda a todos os requisitos previstos na norma para que possa ser posteriormente certificado. Atualmente, existem pilhas de protocolos de segurança previamente aprovados que possibilitam a implementação de protocolos de comunicação segura sem a necessidade de criá-los por inteiro, fundamentando-se em um deles e ajustando seus respectivos protocolos de transporte e configurações de *hardware*. São abordados a seguir o estudo realizado sobre cinco destes protocolos distintos, suas especificações, diferenças, em quais casos práticos se encaixam melhor, dentre outros fatores. Os protocolos em questão são: PROFISafe, FailSafe-over-EtherCAT, OpenSafety, CIP Safety on Sercos e CIP Safety. Todos eles possuem algumas características em comum: o uso do modelo *Black Channel* (torna os protocolos totalmente independentes do protocolo de transporte), o mesmo padrão de segurança funcional (seguindo a norma IEC 61508), mesmos padrões para barramentos de campo (IEC 61158, IEC 61784-1, IEC 61784-2) e nível de SIL 3. Além de que todos os protocolos ditos seguros necessitam prover mecanismos de detecção e correção de erros contidos nas mensagens transmitidas (ordenação, perdas, repetições, dentre outros). Contudo, em sua maioria, os protocolos de comunicação segura têm protocolos de transporte específicos definidos para cada um deles, conforme consta na norma que os certifica. Exceto pelo OpenSafety que não possui restrições quanto a isso, podendo ser operado sobre diversos protocolos de transporte.

3.1 Características

Mesmo utilizando as pilhas de protocolos seguros já existentes, devemos ressaltar algumas características que são imprescindíveis no momento de decidir qual deles seria a melhor opção para um determinado tipo de aplicação. A IEC 61508 sugere algumas medidas de segurança para o desenvolvimento de um protocolo de comunicação segura [Elia et al., 2006]:

- Adicionar um número de sequência na mensagem a ser transmitida permite detectar erros de retransmissão, perda, inserção e sequência incorreta de dados
- *Timestamp* em cada mensagem possibilita que o receptor identifique o momento em que a mensagem fora gerada para ser transmitida. Detectando, através dele, repetições, sequência de dados incorretas e atrasos. Além de permitir uma arbitragem melhor do acesso e do enfileiramento das mensagens para envio
- Expectativa de tempo: limite de tempo estipulado entre ambos os envolvidos (emissor e receptor), que se excedido faz com que o receptor perceba que um erro (possivelmente um atraso) ocorreu
- Mensagens de confirmação via *echo*: o receptor envia uma mensagem *echo* para o emissor, repetindo a mesma mensagem que recebeu para que o emissor verifique se a mesma está correta, identificando, assim, perdas, inserções, corrupção dos dados e mascaramento de dados. O que dificulta na utilização desta técnica é o fato da necessidade de haver uma retransmissão a cada mensagem recebida

- Identificador para o emissor e receptor: ambos reconhecem um ao outro através de um id adicionado a mensagem. Deste modo, detecta-se inserções de terceiros nas mensagens
- Redundância com verificação cruzada (*cross checking*), comparando as mensagens recebidas para *crossover*, testando para transmissão correta. Detectando, de tal modo, quando houver repetições, perdas, erros de sequência e corrupção dos dados.
- Distinção das mensagens seguras (*safety related* - SR) das não seguras (*non-safety related* (NSR)
- Proteção dos dados, testando o conteúdo das mensagens antes de transmiti-las através do uso de CRC ou codificando os bits de uma maneira especial

3.2 PROFIsafe

O protocolo de comunicação segura desenvolvido pela PI – PROFIBUS e PROFINET Internacional, garante validação para segurança funcional somente quando for utilizado o protocolo PROFIBUS ou o PROFINET para o transporte. Contudo, permite o transporte das mensagens seguras no mesmo canal que o das mensagens padrão. Como dito anteriormente, um protocolo para ser considerado seguro, precisa fornecer mecanismos de detecção e correção de erros. No PROFIsafe, essas funções são empregadas sobre as mensagens trocadas nas conexões, onde são verificadas a consistência dos dados, adotando um sistema de nomes único para emissor e receptor, numerando as mensagens seguras, definindo tempos para cada uma delas.

O PROFIsafe apresenta 3 níveis de desempenho: para dados não-críticos por tempo (usa-se TCP, UDP e IP), para dados com processos críticos por tempo (Real-Time, no campo da automação industrial) e para demandas específicas (*hard real-time control*). O enfoque neste estudo é o desempenho para a área de automação industrial (Real-Time). O mecanismo de acesso à rede segue o modelo de comunicação mestre/escravo, onde o mestre é o F-Host e o escravo é o F-Device. Assim, o F-Device envia mensagens somente para responder às requisições do mestre [Vidal et al., 2014]. São adotadas algumas das medidas disponibilizadas pela norma para que o PROFIsafe atenda aos padrões de segurança. Elas são, a saber: número de sequência, *timestamp*, *watchdogs*, identificador emissor e receptor, diferença entre as mensagens SR eNSR, e a proteção dos dados (CRC, etc.) [Elia et al., 2006].

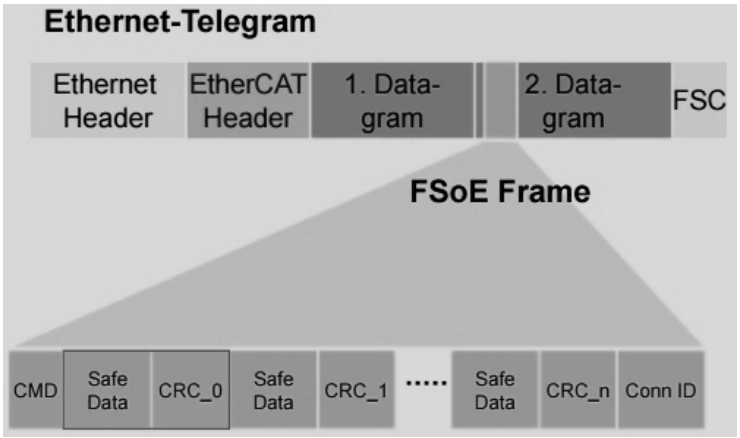
3.3 FailSafe-over-EtherCAT

Anteriormente conhecido como Safety-Over-EtherCAT, o FSoE é um protocolo aberto, especificado pelo ETG – EtherCAT Technology Group e aprovado pelo TÜV SÜD Rail GmbH, que para garantir a comunicação segura restringe a camada de transporte ao uso único e exclusivo de EtherCAT para transmissão dos dados. De maneira similar ao PROFIsafe, o FSoE envia suas mensagens seguras através de conexões e sessões exclusivas, recorre a métodos de ordenação sequencial, a fim de detectar possíveis duplicações e perdas das mensagens, e utiliza CRC para detecção de erros nos dados, diferindo do protocolo anterior somente no fato de utilizar um relógio global e rótulos de tempo nas mensagens.

Alcança ciclos de tempo real muito curtos, com alta capacidade de transferência, *jitter* na ordem de 1μs e probabilidade residual de erro $R_{(p)} < 10^{-9}$/h. Integra-se ao sistema TwinSAFE para desempenhar as funções de segurança, tais como paradas emergenciais e monitoramento da cerca segurança (*safety fence*), apresentando pequenas caixas de

chaveamento diretamente ligadas a *safety fence*. A sua interação otimizada entre automação standard e tecnologia de segurança reduz os custos de *hardware* e engenharia, simplificando o cabeamento e facilitando a implementação de modificações. Utilizando a comunicação mestre/escravo, o *frame* FSoE é mapeado em PDOs (*Process Data objects*) cíclicos, com tamanho mínimo do quadro FSoE de 6 Bytes e máximo dependendo do número do dado do processo seguro do dispositivo escravo. O ciclo FSoE consiste em um quadro mestre FSoE confirmado por um quadro escravo FSoE. O FSoE mestre envia um quadro para o FSoE escravo e, ao enviá-lo, o mestre inicia um *Watchdog-Timer* para proteger o escravo. O mestre somente iniciará um novo ciclo, criando outro quadro mestre FSoE quando receber um quadro escravo válido. Caso o *Watchdog* expire, os dispositivos mudarão para o estado de *Reset* [Liu and Song, 2012].

Figura 2 – Quadro FSoE



Fonte: Adaptado de Safety_over_EtherCAT_Overview (2016).

A figura 2 exemplifica como é formado o quadro FSoE, cujo mapeamento é feito como se fosse um contêiner dos dados do processo do dispositivo, havendo uma verificação a cada 2 *bytes* de dados seguros com um CRC de mesmo tamanho, sem impor limite quanto ao tamanho dos dados seguros processados.

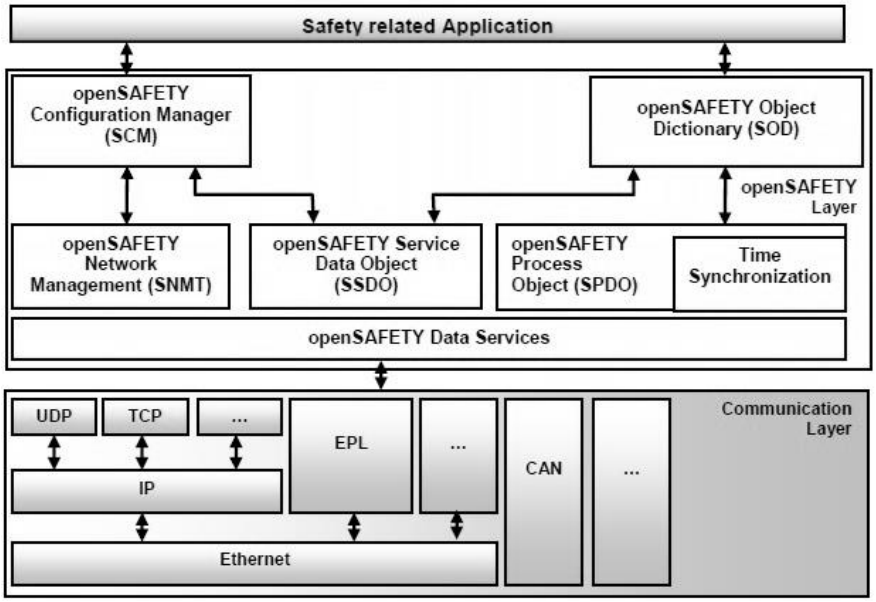
3.4 OpenSafety

Inserido no mercado, inicialmente, com o nome de EPL safety (Ethernet Powerlink Safety), fora desenvolvido pela EPSG – Ethernet POWERLINK Standardisation Group – como um protocolo aberto e mais flexível por ser totalmente independente do protocolo de transporte, não restringindo a camada de transporte a utilização de um protocolo específico. Em 2008, foi testado pela TÜV Rheinland Group e aprovado para uso em aplicações *safety* como especificado pela norma [ZURAWSKI, 2014]. Assim como o FSoE, utiliza conexões e sessões únicas, onde emprega rótulos de tempo nas mensagens, verificando a integridade dos dados através de CRC.

Realiza um monitoramento da comunicação por tempo (*time-slicing*) a fim de prevenir erros causados por perdas dos dados ou *delays* excessivos, utilizando também o mecanismo de *Watchdogs*, onde os usuários poderiam identificar um mal funcionamento, verificando a sequência contínua do fluxo de dados. Assim, o OpenSafety garante transmissão dos dados de maneira segura utilizando redes “inseguras”, sendo adequado para comunicações com ciclos na faixa de μs, apresentando um tempo de resposta abaixo de 100us. Ademais, habilita um gerenciamento de riscos flexível, *Smart Safe Reactions*, ao invés de paradas emergenciais, onde na maioria das vezes é suficiente reduzir a velocidade/movimentação das máquinas ou limitar o torque para um nível seguro ao invés

de parar tudo simultaneamente [Ethernet POWERLINK Facts - The Magazine for the Industrial Ethernet Standard - Open Safety Technology, 2009].

Figura 3 – Modelo de referência do OpenSafety



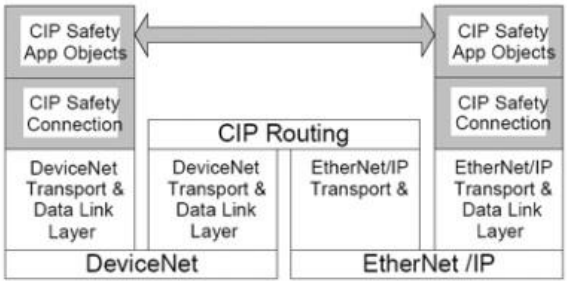
Fonte: Adaptado de [Ethernet POWERLINK Standardisation Group 2013].

A Figura 2 ilustra o modelo de referência do OpenSafety. Nele podemos observar a independência da camada de transporte, além da forma mais compacta que o protocolo trata a comunicação segura, encapsulando a representação dos dados e a definição de transporte (openSAFETY Data Services, como pode-se observar na figura) e os serviços de nível superior de maneira similar ao POWERLINK [Ethernet POWERLINK Standardisation Group, 2013].

3.5 CIP Safety

O perfil de comunicação especificado pela ODVA – Open Device Vendors Association é uma extensão do padrão, já certificado pela TÜV Rheinland, CIP (Common Industrial Protocol), mediante a adição da funcionalidade da camada de aplicação CIP Safety ao padrão [ZURAWSKI, 2014]. Como as extensões da camada de aplicação de segurança não dependem da integridade dos serviços CIP padrão e das camadas de enlace de dados, *hardware* de canal único (não redundante) pode ser utilizado para a interface de comunicação de enlace de dados. Esta divisão de funcionalidades permite que roteadores padrão passem a rotear os dados seguros como ilustrado na Figura 5. O roteamento das mensagens é possível pois, o dispositivo final é responsável por garantir a integridade dos dados. Então, quando houver um erro na transmissão dos dados ou no roteador intermediário, o dispositivo final detectará a falha e tomará uma ação apropriada [VASKO AND AUTOMATION, 2005].

Figura 4 – Roteamento das mensagens



Fonte: Artigo [Vasko and Automation 2005].

A transmissão dos dados é realizada via Ethernet/IP ou DeviceNet, além de, utilizar o mecanismo origem-destino para troca de dados entre os sistemas *safety*, definindo tempos para as mensagens através de um *timestamp*, garantindo que os dados estejam atualizados, usufruindo também de um identificador para enviar e receber os dados. O CIP *safety* utiliza códigos CRC através de objetos de validação *safety* em ambas as extremidades (emissor e receptor) de forma a verificar os dados transmitidos, organizando e garantindo a integridade dos mesmos, permitindo, também, que dispositivos *safety* e *standard* operem em uma mesma rede.

3.6 CIP Safety on Sercos

CIP Safety on Sercos é um protocolo para transmissão de dados relevantes a safety (safety-relevant) via Sercos. Dados *safety-relevant* são transmitidos como um contêiner de dados *safety*. O dado do contêiner é armazenado no canal do dispositivo de tempo real relevante, exatamente como dados padrões. Um protocolo multiplex, SMP (*Sercos Messaging Protocol*), é utilizado para transmitir dados seguros que tenham sido escaneados sem perder largura de banda apesar de terem ciclos de barramento mais curtos [SAFETY, 2016].

Figura 5 – Modelo estrutural do perfil de comunicação CIP Safety on Sercos

Fonte: Adaptado de Safety (2016).

Baseado sobre o protocolo CIP *safety*, fora especificado pela ODVA – Open Device Vendors Association, utilizando acesso múltiplo por divisão de tempo (TDMA), onde destina-se um canal dedicado livre de colisão para a comunicação em tempo real e um canal IP adicional para transferência de dados assíncronos. Cada dispositivo possui 2 portas Ethernet, um controlador responsável pelo chaveamento entre o trânsito de dados em tempo real e o trânsito de IPs, garantindo uma sincronização sólida de todos os nodos

conectados. A comunicação, do tipo mestre/escravo, acontece de maneira cíclica, pois o mestre atua como um controlador numérico (CN), no tempo de ciclo selecionado na inicialização; podendo ser de 31.25µs, 62µs, 125µs, 250µs ou qualquer múltiplo de 250µs até 65ms; o *jitter* encontra-se na ordem de 1µs. Além disso, o protocolo suporta comunicação direta entre os escravos. Esta funcionalidade é vantajosa nas aplicações de controle de movimentação seguras: os dados seguros podem ser trocados entre os escravos diretamente utilizando comunicação cruzada *peer-to-peer*, sem a necessidade de coletar e redistribuir os dados através de um mestre centralizado. Outra grande vantagem é a definição de um mecanismo de redundância quando for empregada a topologia do anel para gerar um anel duplo. O mecanismo oferece caminhos alternados para prover uma recuperação "sem colisões" do rompimento de um dos anéis (ELIA et al., 2006).

Ademais, a interface SERCOS com drives digitais inteligentes fornece monitoramento preciso e proteção contra perda do controle de movimentos/velocidade, realizando um desligamento forçado em caso de falha do processador do drive. Mestres e escravos inserem números de sequência nas mensagens a fim de especificar o *telegram* a ser transmitido para tentar evitar erros na comunicação como repetições, corrupção dos dados, dentre outros. Cada nodo do anel mantém o cálculo do número de sequência, podendo identificar quando uma sequência for inválida. E a corrupção de um bit é detectada via procedure HDLC, código que permite reconhecer erros de bit com uma distância de *hamming* maior do que 4, e via monitoramento adicional dos tamanhos das mensagens e dos tempos de transmissão. Podendo utilizar, também, diversas redes de comunicação, tais como: DeviceNet e Ethernet/IP.

4 Análise comparativa

A coleta de dados referente aos perfis existentes para a implementação de protocolos de comunicação segura possibilitou a realização de uma análise comparativa, identificando as vantagens e limitações de cada um. O quadro abaixo identifica as medidas de segurança utilizadas em cada protocolo, dependendo do erro detectado.

Tabela 1 – Medidas de segurança para cada protocolo conforme o erro detectado

Erros	PROFIsafe	FSoE	OpenSafety	CIP Safety	CIP Safety on Sercos
Repetição	-Nº de sequência -Timestamp	-Nº de sequência -CRC	-Timestamp	-Timestamp	-Timestamp
Perda	-Nº de sequência	-Nº de sequência -Watchdog -CRC	-Timestamp -Watchdog	-Timestamp	-Timestamp
Inserção	-Nº de sequência -Identificador	-Nº de sequência -CRC	-Identificador	-Timestamp -Identificador	-Timestamp -Identificador
Sequência incorreta	-Nº de sequência -Timestamp	-Nº de sequência -CRC	-Timestamp	-Timestamp	-Timestamp
Corrupção dos dados	-Integridade dos dados via CRC	-CRC	-CRC -Redundância	-CRC -Redundância	-CRC -Redundância

			via *cross check*	via *cross check*	via *cross check*
Atrasos	-Watchdog	-Watchdog	-Watchdog	-Timestamp	-Timestamp
Mascaramento das mensagens	-Identificador - Watchdog -CRC	-Watchdog -CRC	-Identificador -Timestamp -CRC	-Identificador	-Identificador

Fonte: Elaborada pela própria autora.

Como pode ser observado acima, o protocolo *safety* desenvolvido pela Sercos utiliza o CIP Safety para as funcionalidades de segurança, logo as medidas adotadas para tratar os erros são as mesmas. Os fatores cruciais que fazem com que uma empresa decida utilizar um protocolo para sua determinada aplicação dependem não somente das medidas de segurança, que, em sua maioria, são similares pois atendem aos requisitos da norma IEC 61508, mas principalmente em relação ao custo, aos módulos de compatibilidade disponíveis, a implementação e certificação. Assim, os que utilizam somente um protocolo de transporte e módulos específicos, como o PROFIsafe e FSoE, embora consolidados no mercado, são os mais rígidos e inflexíveis, e, portanto, os mais custosos para implementação. Os demais podem atender a aplicações mais variadas por se adaptarem melhor, aceitando diversos módulos e tipos de comunicação. Entretanto, por ser código aberto e ao mesmo tempo não restringir diversos fatores para a implementação e utilização de um protocolo de comunicação segura, o que se mostrou mais flexível em todos os aspectos foi o OpenSafety, pois, além de não limitar a transmissão a um protocolo de transporte específico, o investimento é bem mais viável, sendo o menos custoso, garantindo uma comunicação confiável e tolerante a falhas.

Tabela 2 – Demais critérios para a escolha de um protocolo *safety*

Critérios	PROFIsafe	FSoE	OpenSafety	CIP Safety	CIP Safety on Sercos
Custo	Elevado (Licença, certificação, parametrização)	Elevado (Licença, certificação)	Baixo: sem custo para obtenção; custo somente para certificá-lo	Custos com Licença, certificação, módulos compatíveis	Reduz custos de HW e instalação; precisa de Licença e certificação
Desempenho	Tempos de ciclo reduzidos para 31.25µs	Tempos de ciclo ≤ 100µs	Tempo de resposta 4x mais rápido que o FSoE	Tempos de ciclo mínimos de 31.25µs	Tempos de ciclo mínimos de 31.25µs
Flexibilidade	Transporte restrito ao uso de PROFIBUS ou PROFINET	Transporte somente via EtherCAT	Totalmente flexível (independente de protocolo)	Transporte via Ethernet/IP ou DeviceNet	Transporte via barramento Sercos III
Topologia	Estruturas flexíveis (linha, árvore, estrela ou anel	Atende a qualquer combinação de topologia	Aceita qualquer topologia	Atende as diversas combinações (linha,anel...)	Atende as diversas combinações (linha, anel...)
	Parte da documentação disponível no	Documentos disponíveis no site do	Open-source, gratuito, pré-certificado	Documentos disponíveis no site do	Disponível para venda no site da sercos

Facilidade de obtenção	site da PI e kit de desenvolvimento disponível no mercado	grupo EtherCAT e obtenção na matriz ETG	(como os outros) e disponível no site p/ download	grupo ODVA e venda nos site de fabricantes	direcionando para os fabricantes

Fonte: Elaborada pela própria autora

5 Proposta

Com o intuito de analisar ainda em mais detalhe os protocolos de comunicação segura para automação industrial, foi escolhido o protocolo OpenSafety, julgado como um dos mais flexíveis (independente de protocolo de transporte, módulos simples e facilmente adaptável) para que fosse realizada uma análise na prática. Logo, o que está sendo proposto para o Trabalho de Gradução II (TG-II) é a experimentação com o protocolo de comunicação OpenSafety, as dificuldades para colocá-lo em funcionamento referentes tanto à documentação quanto à sua utilização, verificando todos os aspectos que uma empresa levaria em consideração antes de utilizar tal perfil. Assim, as atividades referentes ao TG-II foram divididas em seis etapas principais: estudo da documentação do protocolo escolhido, instalação dele em um laboratório de pesquisa, experimentação clássica com carga de trabalho sintética, análise, escrita e apresentação.

Tabela 3 – Cronograma do plano de atividades para o Trabalho de Graduação II

Fase	Janeiro	Fevereiro	Março	Abril	Maio	Junho
Estudo	X					
Instalação	X	X				
Experimentação		X	X			
Análise			X	X		
Escrita			X	X	X	
Apresentação						X

Fonte: elaborada pela própria autora.

8 Bibliografia

Avizienis, A. et al. (2004). “Basic concepts and taxonomy of dependable and secure computing”, *IEEE transactions on dependable and secure computing*, v. 1, n. 1, p. 11–33.

BECKMANN, G. (2016). “Overview Safety over EtherCAT”, . Disponível em: <http://www.ethercat.org/pdf/english/Safety_over_EtherCAT_Overview.pdf>.

Elia, A. et al. (2006). “Analysis of Ethernet-based safe automation networks according to IEC 61508”, Em *2006 IEEE Conference on Emerging Technologies and Factory Automation*. . IEEE. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4178252>, Acesso em 12 Nov. 2016.

EPSG - The Black Channel Principle (2016). Disponível em: <http://www.ethernet-powerlink.org/en/powerlink/industrial-ethernet-facts/safety-functionality/the-black-channel-principle/>, Acesso em 12 Nov. 2016.

Ethernet POWERLINK Facts - The Magazine for the Industrial Ethernet Standard - Powerlink Safety - Open Safety Technology (Out 2009). v. 4, p. 12.

Ethernet POWERLINK Standardisation Group (2013). “EPSG_WDP_304_V-1-4-0 .pdf”,

Felser, M. e Sauter, T. (2004). “Standardization of industrial ethernet-the next battlefield?”, Em *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*. . IEEE. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1377762>, Acesso em 12 Nov. 2016.

Liu, Y. e Song, Y. (2012). “EtherCAT-based functional safety-integrated communication”, Em *Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on*. . IET. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6492753>, Acesso em 12 Nov. 2016.

Neumann, P. (Nov 2007). “Communication in industrial automation—What is going on?”, *Control Engineering Practice*, v. 15, n. 11, p. 1332–1347.

openSAFETY: openSAFETY Software Concept ([S.d.]). Disponível em: <http://opensafety.sourceforge.net/doc/html/page_swconcept.html>, Acesso em 17 Jun. 2017.

openSAFETY: openSAFETY Software Structure ([S.d.]). Disponível em: <http://opensafety.sourceforge.net/doc/html/page_swstructure.html>, Acesso em 17 Jun. 2017.

PROFIBUS Nutzerorganisation e. V. (Nov 2010). “PROFIsafe System Descriptor”,

PROFIsafe (2017). Disponível em: <http://www.profibus.com/technology/profisafe/>, Acesso em 17 Jun. 2017.

Safety (2016). Disponível em: <http://www.sercos.org/technology/safety/>, Acesso em 12 Nov. 2016.

Safety_over_EtherCAT_Overview.pdf. Disponível em: <http://www.ethercat.org/pdf/english/Safety_over_EtherCAT_Overview.pdf>, Acesso em 12 Nov. 2016.

Smith, D. J. e Simpson, K. G. L. (2004). *Functional Safety: A Straightforward Guide to Applying IEC 61508 and Related Standards*. Routledge.

Vasko, D. A. e Automation, R. (2005). “DeviceNet Safety: Safety networking for today and beyond”, *PUB 00110R0. ODVA*,

Vidal, W. et al. (Mar 2014). “Software Architecture for Safety Communication in Critical Systems”, . IEEE. Disponível em: <http://ieeexplore.ieee.org/document/7469506/>, Acesso em 12 Nov. 2016.

Zurawski, R. (2014). *Industrial Communication Technology Handbook, Second Edition*. CRC Press.