Arquitetura para injeção de falhas em protocolos de comunicação segura em aplicações críticas

Sergio Cechin, Taisy Silva Weber, João Cesar Netto

Instituto de Informática – UFRGS
Caixa Postal 15064 - 91501-970 Porto Alegre, RS
{taisy@inf.ufrgs.br},{cechin@inf.ufrgs.br}

Resumo: *Em aplicações industriais críticas, falhas podem provocar a morte de pessoas ou danos irreparáveis ao meio ambiente. Devido ao ambiente hostil, a comunicação é um dos elos mais frágeis do sistema. A comunicação através de protocolos TCP/IP ou barramentos de campo apresenta taxas de defeito incompatíveis com os requisitos impostos a aplicações críticas. Quando o risco de acidentes fatais é muito alto, protocolos de comunicação segura, tais como o PROFIsafe, openSafety e o Safety-over-EtherCAT, devem ser empregados. Entretanto, para cada novo equipamento, o protocolo deve ser implementado e validado obedecendo estritamente às recomendações de normas de segurança como a IEC 61508 e a IEC 61784-3. As normas exigem injeção de falhas em todas as fases de teste. Para facilitar a aplicação destas normas por parte dos desenvolvedores e testadores, o artigo propõe a arquitetura de um ambiente de injeção de falhas para validação de protocolos de comunicação sugeridos pelas normas de segurança. Para atender aos requisitos de baixo custo e alta precisão, a arquitetura proposta baseia-se no uso de hardware genérico, com o emprego eventual de adaptadores de hardware, e de software específico.*

Abstract*: In critical industrial applications, faults can cause deaths or irreparable damage to the environment. Due to the harsh environment, communication is one of the weakest components in the system. Communication via TCP/IP or fieldbus features failure rates incompatible with the requirements of the critical applications. When the risk of fatal accidents is very high, secure communication protocols, such as PROFIsafe, openSafety and the Safety-over-EtherCAT, should be employed. However, for each new device, the protocol must be implemented and validated in strict obedience to the safety standards IEC 61508 and IEC 61784-3. The standards require fault injection in all test phases. To facilitate the implementation of these standards by developers and testers, the paper proposes the architecture of a fault injection environment for validation of safety protocols. To meet the low cost and high accuracy requirements, the proposed architecture is based on the use of generic hardware, with the possible use of hardware adapters, and specific software.*

Trabalho desenvolvido no Projeto SDCD - FAURGS/UFRGS/BNDES – (13.2.0646.1 - projeto 8108).

1 Introdução

Aplicações industriais requerem o monitoramento de variáveis do processo, o processamento dessas informações segundo alguma lógica programável e a atuação sobre esse processo. Como apresentado na figura 1, o monitoramento do processo é realizado por sensores; a lógica programável é realizada por equipamentos controladores e a atuação é conduzida por dispositivos atuadores. Em sistemas mais simples, o sistema é implementado através da conexão entre sensores, controladores e atuadores através de cabos de cobre, por onde trafegam sinais elétricos proporcionais às grandezas do processo. Entretanto, nos sistemas atuais, mais complexos, a comunicação é realizada através de redes de dados.

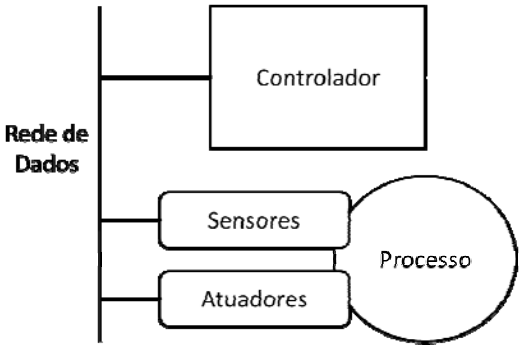

Figura 1 – Arquitetura dos Sistemas de Controle de Aplicações Industriais

Falhas de comunicação na rede de dados em aplicações *críticas*, como geração de energia, extração de petróleo, e transporte público, podem conduzir a acidentes fatais. Os controladores do processo ou controladores específicos podem ser aplicados para aumentar a segurança de instalações industriais executando funções necessárias para prevenir ou mitigar acidentes (Dunn 2003). Exemplos de funções de segurança são: controle de parada emergencial; bloqueio preventivo em maquinários pesados; sinais de alerta; e alarmes de incêndio e detecção de gases tóxicos.

Portanto, devido ao risco inerente a essas funções, elas devem receber uma atenção especial. Essa atenção aparece na forma de normas de segurança (*safety*), onde uma função de segurança é avaliada pelo seu nível integridade de segurança, SIL (Bell 2011). Integridade de segurança que é definida como a probabilidade de ocorrência de defeitos ao realizar a função de segurança, e apresenta 4 níveis. Um função SIL 4 reduz o risco de um acidente fatal em 10.000 vezes. SIL 3 reduz o risco em 1000 vezes e SIL 2 em 100 vezes. O setor de óleo e gás exige SIL 2 a SIL 3, dependendo da função. Em um cenário em que a explosão de uma plataforma de petróleo poderia provocar a morte de até 100 pessoas, se as funções de segurança forem todas SIL 3, é esperado que o dano fique restrito a menos do uma pessoa, o que é considerado tolerável.

Protocolos da pilha TCP/IP não podem ser usados para comunicação em sistemas de segurança. Eles não atendem os critérios das normas. Não há argumentação aceitável que justifique o emprego desses protocolos em ambientes críticos. Funções de segurança se executadas diretamente sobre TCP/IP impedem que os órgãos reguladores autorizem a operação do sistema. A situação não melhora se foram empregadas redes de automação como Profibus, Profinet, ou EtherCAT. A solução é o emprego de protocolos seguros de comunicação. Mas não existem protocolos seguros prontos para serem instalados em ambientes críticos, apenas especificações ou *tool kits*. Os protocolos

devem ser codificados seguindo rigorosamente as normas de segurança e tanto o processo de desenvolvimento como o código resultante devem ser certificados por uma entidade competente.

O artigo propõe uma arquitetura para um sistema de validação por injeção de falhas que permita aplicar testes sob falhas na implementação dos protocolos mais usados em ambientes industriais (como PROFIsafe, openSafety e Safety-over-EtherCAT). A arquitetura facilita a aplicação dos testes por partes dos desenvolvedores e testadores e permite reuso de cargas de falhas. A arquitetura foi recentemente implementada para validação do protocolo PROFIsafe e está sendo no momento estendida para comportar o Safety-over-EtherCAT. Este trabalho é desenvolvido com o financiamento do projeto SDCD, FAURGS/UFRGS/BNDES, e apoio da empresa Altus, que produz controladores lógicos programáveis. O SDCD visa desenvolver ambientes de controle e monitoramento distribuídos, incluindo sistemas críticos onde segurança (*safety*) é um requisito não funcional a ser atendido.

O artigo se desenvolve como segue: inicialmente são apresentadas normas de segurança que tratam de comunicação segura e é fornecida uma visão geral sobre protocolos seguros. A seguir são apresentados conceitos de injeção de falhas e alguns trabalhos relacionados e a proposta de arquitetura para o ambiente de injeção de falhas. Uma breve descrição da experiência já adquirida com o ambiente finaliza o artigo.

2 Normas de segurança

Nas últimas décadas, uma grande quantidade de normas de segurança foi proposta para vários setores de atividades (Esposito, Cotroneo, and Silva 2011). Normas diferentes são adotadas em diferentes domínios de aplicação (como óleo e gás, aviação, energia elétrica e nuclear, indústria automobilística). As normas de segurança têm em comum a obrigatoriedade do emprego de técnicas de prevenção e de tolerância a falhas para a redução de risco, além do projeto criterioso e documentado, desde as primeiras fases do ciclo de desenvolvimento do sistema. As normas permitem certificação de sistemas construídos segundo suas recomendações e que tenham sido devidamente verificados e validados.

Normas de segurança tratam tanto do hardware como do software. Software é o principal problema. Muitos acidentes graves devidos a falhas de software são reportados na literatura (Zhivich and Cunningham 2009), (Alemzadeh et al. 2013), (Hardy 2014). Aplicar criteriosamente normas de segurança não fornece garantia absoluta de evitar acidentes, mas reduz consideravelmente os riscos.

2.1 Perfis de comunicação segura em aplicações críticas

Entre as várias normas de segurança, algumas apresentam uma natureza mais genérica, como a IEC 61508 ((Fowler and Bennett 2000), (Bell 2006)). Seu principal objetivo é orientar o trabalho de equipes técnicas no desenvolvimento de sistemas computacionais de segurança, visando alcançar níveis compatíveis com os exigidos por agências reguladoras (Gall and Wen 2010). A IEC 61784-3, que padroniza perfis de comunicação segura, é derivada da IEC 61508. As normas são extensas, com centenas de medidas prescritivas baseadas em uma vasta experiência prática, pesquisas e discussões. Como consequência, tanto desenvolver um projeto em conformidade com a norma como obter certificação são tarefas árduas (Bilich and Hu 2009).

2.2 Comunicação em sistemas instrumentados de segurança

Um sistema instrumentado de segurança, SIS, é encarregado de executar a função de segurança. Geralmente, é formado por um componente lógico conectado por uma rede de comunicação a uma central e a portas remotas de entrada e saída. Essas portas remotas estão conectadas aos sensores e/ou aos atuadores, que estão por sua vez diretamente ligados ao processo. Até recentemente, SIS eram construídos com componentes discretos. A partir do ano 2000, com a publicação da IEC 61508, foi permitido que SIS fossem construídos com componentes programáveis. Atualmente, as portas remotas também são inteligentes, formando assim um sistema distribuído de instrumentação. Na figura 2 é apresentado o sistema de controle com o SIS. Nota-se a inclusão de um Controlador (*safety*), um conjunto de sensores e atuadores separados daqueles usados para o processo e uma rede segura de dados, separada da rede de dados de controle do processo.

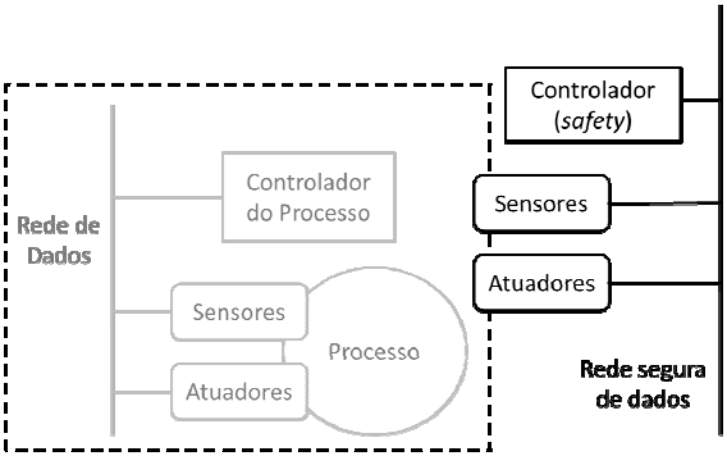

Figura 2 – SIS: sistema instrumentado de segurança

Sistemas de controle há muito tempo são programáveis, e muitos alcançam alta confiabilidade. A diferença é que apenas SIS precisam ser certificados. A separação entre sistemas de controle e de segurança reduz os custos em uma instalação industrial e, pelo isolamento de falhas, aumenta a confiabilidade. Regulamentos de vários setores, como os do setor de óleo e gás, proíbem que funções de segurança sejam executadas nos mesmos equipamentos responsáveis pelas funções de controle.

SIS programáveis são extremamente flexíveis e adaptáveis. Mas onde existe comunicação e software, existem falhas. O desafio é manter o sistema operando com segurança mesmo na ocorrência de falhas. Para evitar acidentes, deve ser garantida alta confiabilidade do próprio SIS incluindo, também, a comunicação entre seus componentes.

As normas especificam que todos os recursos de hardware, software e comunicação que implementam uma função de segurança devem ser certificados. Inicialmente, deveriam ser validadas todas as camadas de comunicação e todos os protocolos e mídias legados. Isso implicava na certificação do desenvolvimento da pilha de protocolos, desde a parte física até a aplicação, o que se mostrou inviável na prática devido a limitações de custo e de fornecedores. O problema foi resolvido na versão de 2010 da IEC 61508 com a adoção do conceito de *Black Channel*.

2.3 Parte segura e insegura da comunicação: *black channel*

É possível reduzir o risco de acidentes a valores aceitáveis dividindo a comunicação em uma parte segura e outra não segura. Na parte não segura estão os mecanismos usados para o transporte de dados. Esses operam como transportadores (*carrier*) dos pacotes de dados dos protocolos seguros e não necessitam de certificação. Esse canal é "opaco" (*black*), uma vez que, a princípio, não afeta as características de segurança da comunicação.

Os protocolos seguros são, então, implementados sobre um *black channel*, o que os torna independente dos canais físicos de transmissão, sejam eles fios de cobre, fibras ópticas, *wireless* ou *backplanes*. As taxas de transmissão e os mecanismos de detecção de erros do *black channel* não têm qualquer interferência sobre o protocolo seguro. Entretanto, o *black channel* deve garantir uma taxa máxima de defeitos. Se não operar abaixo desta taxa, não é possível garantir a segurança da camada acima.

O conceito de *black channel* é extremamente útil para o desenvolvedor. É possível, assim, usar formas de comunicação já disponíveis e se concentrar apenas na codificação da camada segura. Protocolos como EtherCAT, PROFIsafe, openSafety, INTERBUS, e FOUNDATION Fieldbus se apoiam no conceito de *black channel* (Neumann 2007).

3 Visão geral dos protocolos seguros

Os protocolos PROFIsafe, openSafety e Safety-over-EtherCAT, chamados de “perfis” de comunicação (*communication profiles*) pela IEC 61784-3, onde estão definidos, baseiam-se nos padrões das IEC 61158, IEC 61784-1 e IEC 61784-2 para barramentos de campo industriais, e da IEC 61508 para segurança funcional. Todas as especificações dos protocolos seguros mencionados neste item são pré-aprovadas para o nível de segurança funcional 3 (SIL 3).

PROFIsafe é um protocolo de comunicação criado pela PI – PROFIBUS e PROFINET Internacional. OpenSafety foi especificado e certificado pela EPSG – Ethernet POWERLINK Standardisation Group – como um protocolo aberto: quanto a especificação e documentação. Também existem algumas pilhas de protocolos disponíveis, mas que devem ser adaptadas ao hardware alvo e, então, certificadas. Por último, o Safety-over-EtherCAT foi especificado pelo ETG – EtherCAT Technology Group. É considerado um protocolo aberto. Também é conhecido pela sigla FSoE – *FailSafe-over-EtherCAT*.

Todos estes protocolos seguem o modelo de *black channel*, o que significa que são totalmente independentes do protocolo de transporte, no que diz respeito a sua capacidade em detectar e corrigir erros. Entretanto, o PROFIsafe e o Safety-over-EtherCAT estão definidos para serem transportados, a princípio, por certos protocolos específicos. No caso do PROFIsafe, a PI só garante a pré-aprovação do PROFIsafe para segurança funcional, caso seja usado o PROFIBUS ou o PROFINET como protocolo de transporte. No caso do Safety-over-EtherCAT, as características de tempo real da comunicação só são garantidas quando se usa uma infraestrutura EtherCAT para o transporte. Por outro lado, o openSafety não tem essas restrições, podendo operar sobre vários protocolos de transporte, incluindo PROFINET e Ethernet/IP.

3.1 Modelo de falhas e mecanismos de detecção dos protocolos seguros

Segundo as normas, os protocolos seguros devem prover mecanismos para detectar e corrigir erros provocados por falhas nas mensagens transmitidas: corrupção, repetição, perda, inserção, troca de ordem, atraso e mascaramento (interpretação de mensagens seguras como sendo não-seguras e vice-e-versa). O modelo de falhas é semelhante ao modelo adotado por protocolos convencionais de comunicação e se concentra sobre os problemas que afetam as mensagens que transitam na rede de dados. A detecção de erros também é semelhante, mas a cobertura dos mecanismos de detecção deve ser maior, porque é a cobertura que vai definir a segurança do sistema.

Para fazer frente à tarefa de detecção, o PROFIsafe utiliza o conceito de conexão e, sobre as mensagens trocadas nessa conexão, são aplicados mecanismos para verificar a consistência das mensagens, um sistema de nomes único para emissor e receptor, temporização das mensagens e um mecanismo virtual para numeração das mensagens seguras.

O Safety-over-EtherCAT também trafega suas mensagens seguras sobre uma conexão, em que são empregados mecanismos de numeração sequencial, para identificação de perdas, inserções ou repetições e CRC, para detectar a eventual corrupção das mensagens. Ainda, de forma semelhante ao PROFIsafe, as conexões e sessões de operação são unicamente identificadas. Finalmente, de forma diferente do PROFIsafe, o Safety-over-EtherCAT utiliza um relógio global e rótulos de tempo nas mensagens. No caso do openSafety, à semelhança do Safety-over-EtherCAT, são usados rótulos de tempo nas mensagens, identificadores únicos para as mensagens e a integridade dos dados das mensagens é verificada através de CRC.

3.2 Implementação de protocolos de comunicação seguros

O protocolo seguro (norma IEC 61784-3) deve ser implementado seguindo as cláusulas de desenvolvimento de software da IEC 61508. Alguns autores (Mayr, Plosch, and Saft 2011) afirmam não existir evidências suficientes de que a aplicação das normas conduza a software mais seguro. Lloyd e Reeve (Lloyd and Reeve 2009) listam várias dificuldades que as empresas enfrentam para se adaptar a IEC 61508 e obter a certificação do primeiro produto. As dificuldades passam pela falta de qualificação e competência da equipe de desenvolvimento e validação, pouca familiaridade com tolerância a falhas e engenharia de software, falhas na documentação, práticas usuais de desenvolvimento muito diferentes das preconizadas pela norma, impossibilidade de rastrear requisitos, uso de código legado, e adoção de ciclo de vida inadequado. Uma vez, entretanto, que consigam a primeira certificação, os demais produtos são mais facilmente certificados devido à mudança na cultura de desenvolvimento introduzida na empresa.

Para conseguir certificação, entretanto é preciso seguir as normas. Uma questão é a necessidade de desenvolvimento dos protocolos "a partir do zero". Existem fortes restrições de codificação (Vidal et al. 2014). Um determinado código só pode ser certificado se todo o seu processo de desenvolvimento foi realizado segundo as técnicas recomendadas na norma e se foi verificado pelos órgãos certificadores. Para evitar uma implementação completa, existem pilhas de protocolos de segurança pré-aprovadas ("aprovado" é diferente de "certificado"). Estas, por sua vez, devem ser ajustadas ao protocolo de transporte e ao hardware que lhe dará suporte. Essa forma de operação tem consequências: (1) alguns protocolos de segurança requerem que sejam utilizados

protocolos específicos para transporte. É o caso do EtherCAT e o PROFIsafe; e (2) custo de desenvolvimento, mesmo com o uso de pilhas aprovadas não é tão fácil e rápido quando se poderia esperar, pois é necessário desenvolver (e certificar) o processo de adaptação dessas pilhas ao hardware de suporte.

As normas obrigam obedecer requisitos para implementação e validação seguindo um ciclo de vida que enfatiza verificação e validação da implementação do protocolo. Entre as estratégias de validação determinadas pela norma está a injeção de falhas. Finalmente, depois da implementação, validação e verificação, o código executando em um dado equipamento poderá ser certificado. Note-se que só é possível certificar uma implementação existente. Ou seja, uma pilha genérica de protocolos não pode ser certificada. No máximo, ela será "aprovada" para uso no desenvolvimento que leva a uma implementação, que então poderá ser certificada.

3.3 Ciclo de vida de desenvolvimento

A IEC 61508 apresenta um ciclo de vida global para o sistema de segurança e ciclos de vida para a realização do software e do hardware do SIS. A implementação do protocolo de segurança deve seguir o ciclo de vida de desenvolvimento de software. Os ciclos de hardware e software são semelhantes e envolvem as fases de especificação dos requisitos de segurança, planejamento e validação de segurança, projeto e desenvolvimento, integração hardware e software, procedimentos para operação e modificação e finalmente validação de segurança. Todos os ciclos e fases são extremamente convencionais. Técnicas inovadoras, como métodos ágeis, projeto orientado a objetos, aprendizagem de máquina, reconfiguração dinâmica, não são bem aceitas na área de desenvolvimento de sistemas seguros.

A fase de desenvolvimento de software se desdobra no *diagrama V* (figura 3) com um maior detalhamento de subfases.

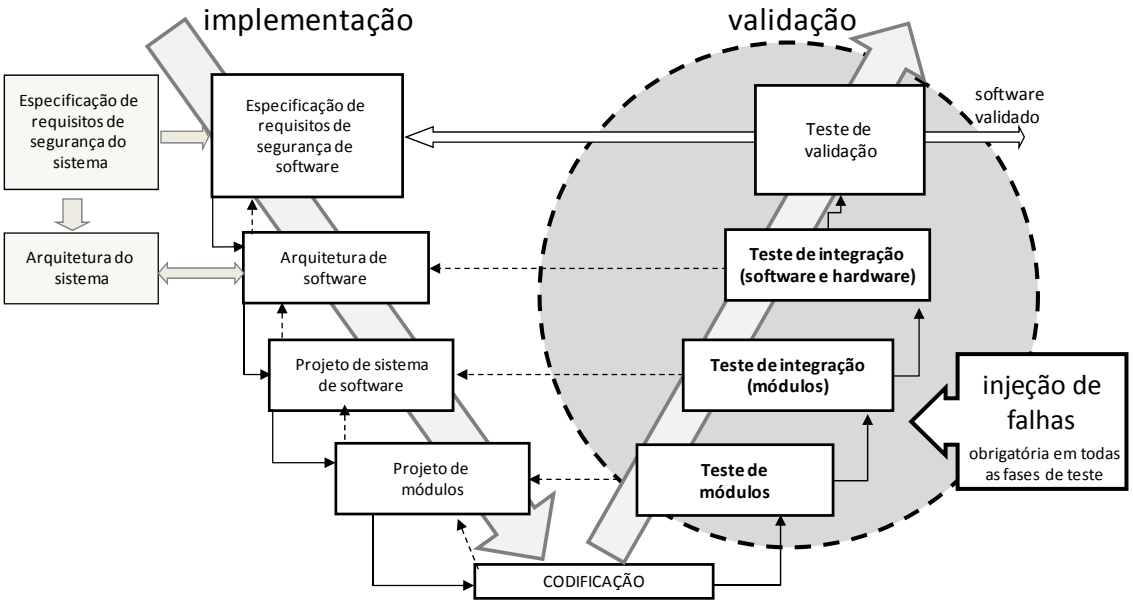

Figura 3 – Diagrama V para desenvolvimento de software na IEC 61508

Fases importantes do ciclo de vida de desenvolvimento do protocolo seguro são relacionadas a tarefas de validação, que corresponde ao lado direito do diagrama V

(figura 3). Todas as formas tradicionais de teste são aplicadas para validação, incluindo teste unitário, teste de integração dos módulos e teste de integração no equipamento alvo (hardware e software). Todos os requisitos devem ser rastreáveis desde a especificação até o código relacionado a cada requisito e ao teste do requisito, assim como no sentido inverso.

Uma técnica de teste imposta pelas normas para as fases de validação é a injeção de falhas. As falhas previstas no modelo de falhas de comunicação devem ser emuladas e o comportamento do protocolo seguro sob falhas deve ser avaliado. Desta forma é possível medir se a cobertura de falhas dos mecanismos de detecção e recuperação de erros está em conformidade com a especificada para um dado SIL.

A injeção de falhas não substitui as demais estratégias de teste e verificação, mas as complementa. Ela é essencial para determinar se os requisitos de segurança especificados foram atingidos e se a tolerância a falhas implementada aumenta a segurança, confiabilidade e disponibilidade do sistema (Pintard et al. 2013).

4 Injeção de falhas e trabalhos relacionados

As estratégias de tolerância a falhas do protocolo de comunicação devem ser ativadas para avaliar sua correção. Como a detecção de falhas só entra em operação na ocorrência de falhas, tais falhas devem estar presentes durante a avaliação do protocolo. Para não depender da ocorrência natural de falhas no ambiente real de operação, uma técnica eficaz (Hsueh, Tsai, and Iyer 1997) é emular falhas por software e injetá-las no protocolo alvo do teste. Através da introdução controlada de falhas, é possível determinar se o protocolo tolera ou não as falhas injetadas.

Um experimento de injeção de falhas é um teste executado em condições controladas na qual são introduzidas falhas. Seguindo uma carga de falhas pré-definida, falhas são introduzidas no sistema durante a execução de uma carga de trabalho. Ambas a carga de trabalho e a carga de falhas devem ser representativas do uso do sistema em condições reais de operação. Para avaliação de protocolos de comunicação, a carga de trabalho são as mensagens trocadas entre os componentes da rede e a carga de falhas inclui as falhas que afetam a troca de mensagens e estão previstas na especificação do protocolo, como por exemplo: corrupção, repetição, perda, inserção, troca de ordem, atraso e mascaramento de mensagens.

Em um experimento, parâmetros que podem ser variados para a carga de trabalho incluem número, tamanho dos pacotes e frequência de transmissão. Parâmetros de carga de falhas incluem: tipos de falhas, instantes de ativação, frequência de ocorrência de falhas e distribuição de probabilidade. Um ambiente experimental de injeção de falhas injeta falhas na troca de mensagens e monitora o comportamento do protocolo enquanto executa uma carga de trabalho.

Um problema neste tipo de teste é a falta de injetores adequados. Apesar dos vários injetores reportados (Natella, Cotroneo, and Madeira 2016), ferramentas funcionais e com usabilidade adequada são difíceis de serem encontradas. A maioria das ferramentas apresenta problemas de portabilidade (Schirmeier et al. 2012) que vão desde terem sido desenvolvidas para ambientes muito específicos, a exigirem sistema operacional ou bibliotecas específicas. Grande parte dos injetores não tem código disponível para uso ou não é mais mantido por seus desenvolvedores.

Na literatura, são encontrados alguns trabalhos de avaliação de tolerância a falhas em protocolos de comunicação ((Dawson et al. 1996), (Hurtig and Brunstrom 2008), (Siqueira et al. 2009)). Também são encontrados trabalhos que discutem a importância da injeção de falhas no desenvolvimento e certificação de sistemas críticos, ((Pintard et al. 2013), (Cotroneo and Natella 2013)). Os trabalhos mais próximos do relatado aqui são injetores de falhas para redes de controle. *Fault Injection Framework for PROFIBUS*, uma rede de campo que suporta PROFIsafe, é um framework de injeção de falhas desenvolvido para avaliar o PROFIBUS (Carvalho, Carvalho, and Portugal 2005). Seu principal objetivo é analisar o comportamento do PROFIBUS na ocorrência de falhas e também a interferência dos mecanismos de tolerância a falhas no desempenho do protocolo. O sfiCAN - *star-based physical fault injector for CAN* (Gessner et al. 2014) é um injetor de falhas físicas para o protocolo CAN, que tem por objetivo testar o comportamento dos nodos de uma rede CAN na presença de erros de canal, em particular dos controladores de nodos CAN e do software executando neles. Apesar da proximidade, nenhum dos trabalhos citados contempla os requisitos necessários para o nosso ambiente de avaliação.

5 Arquitetura do ambiente de avaliação

A arquitetura deve permitir a validação por injeção de falhas de qualquer protocolo seguro construído sobre um *black channel*, definido conforme a IEC 61784-3. Portanto, com modelos de falhas idênticos, a mesma carga de falhas dos experimentos de teste pode ser reutilizada na validação desses vários protocolos. Entretanto, os protocolos seguros executam sobre infraestruturas diferentes. Não existe hardware comum nem software comum de apoio ao protocolo. A solução adotada em injetores de uso geral, como é o caso do Firmament (Drebes et al. 2006), que operam interceptando mensagens no kernel do sistema operacional da máquina que executa o protocolo, não é adequada para protocolos seguros.

Assim, para sistematizar o estudo dos aspectos relacionados ao emprego dos injetores de falhas em sistemas críticos, os ambientes de avaliação foram classificados segundo dois aspectos:

- quanto ao tipo de interferência causada pelo injetor: pode ser classificada como “interferência espacial” ou “interferência temporal”;
- quando a localização de aplicação do injetor: pode ser de aplicação interna aos dispositivos ou externa aos dispositivos.

5.1 Sobre o tipo de interferência do injetor

As máquinas onde são implementados o transmissor e o receptor das mensagens dos protocolos seguros podem ter diferentes tipos de *kernel* de sistema operacional ou mesmo não ter sequer sistema operacional. Além disso, mesmo se houvesse um *kernel* ou outro módulo de software comum a todos os protocolos, o injetor de falhas não poderia estar embutido nesse software. O motivo disso é que todo o software embarcado que estiver associado à função de segurança tem que ser certificado. Portanto, se o injetor de falhas estiver embutido na aplicação, ele deveria ser certificado, o que não faz sentido, pois não é necessário para a implementação da função de segurança, nem é economicamente viável, pois a certificação tem custos que crescem com a complexidade do software empregado. Esse tipo de interferência onde o injetor

representa a ocupação de memória é denominado de *interferência espacial*, que não se pode aceitar no caso dos sistemas críticos.

O segundo aspecto que deve ser considerado na definição do ambiente de injeção de falhas é a *interferência temporal*: interferência do injetor no tempo de propagação das mensagens trocadas pela rede, o que pode levar a erros de temporização do protocolo sob teste. Notar que essa interferência é típica dos protocolos de tempo real, que é o caso dos protocolos seguros, devido a sua grande aplicação em ambientes industriais. A interferência temporal do injetor deve ser compensada para que o protocolo não acuse falhas no seu comportamento funcional durante os experimentos de injeção de falhas.

5.2 Sobre a localização de aplicação do injetor

Um injetor pode ser desenvolvido para *aplicação interna* aos dispositivos do sistema. Esse enfoque oferece a vantagem do baixo custo de implementação, pois não requer o acréscimo de hardware específico. Entretanto, essa técnica sempre causa interferência espacial, que deve ser mantida em níveis aceitáveis. Por outro lado, pode-se reduzir significativamente a interferência temporal, buscando formas de compensar eventuais atrasos ou avanços introduzidos pelo injetor no tratamento das mensagens do protocolo.

Como opção para a aplicação interna, pode-se recorrer à aplicação *externa do injetor*. Nesse caso, a desvantagem está na necessidade de um hardware específico, onde deve rodar uma pilha de protocolos capaz de decodificar as mensagens, injetar a falha e recodifica-las. A vantagem desse enfoque é a inexistência de interferência espacial e a possibilidade de se controlar a interferência temporal usando as técnicas semelhantes àquelas empregadas em um injetor de aplicação interna.

Em linhas gerais, a aplicação interna dos injetores oferece um baixo custo de implementação, quando comparado com a aplicação externa. Entretanto, a aplicação interna dos injetores apresenta níveis de interferência maiores do que a aplicação externa.

Com relação aos protocolos de segurança empregados nos sistemas críticos, é importante ressaltar a questão do custo de certificação: a aplicação interna pode requerer que o injetor seja certificado, o que não faz sentido, conforme já explicado. Então, a localização de aplicação do injetor deve ser tal que não seja necessária essa certificação. Isso só é possível se o injetor de falhas de comunicação estiver aplicado no *black channel*. Dessa forma, pode-se ter injetores de aplicação externa, pois os canais de comunicação sempre estão no *black channel*, ou injetores de aplicação interna, desde que na parte interna do *black channel*.

Mesmo com a possibilidade de aplicar os injetores internamente aos dispositivos, no *black channel*, essa opção apresenta dificuldades de aplicação. A injeção de falhas pode ser usada, entre outras, para verificação de dispositivos para os quais não se tem acesso ao seu interior. Dessa forma, não é possível aplicar o injetor à parte do *black channel* existente no dispositivo. Assim, a arquitetura do ambiente requer um injetor externo aos dispositivos, tornando-o agnóstico às características do dispositivo sob teste (fabricante, modelo, etc).

5.3 Arquitetura dos injetores externos

Os injetores externos são implementados através de equipamentos específicos e aplicados externamente aos dispositivos sob teste. Portanto, não apresentam interferência espacial. Esses injetores externos podem ser construídos de várias formas, no que diz respeito às características de seus componentes de hardware e software. Cada forma de implementação oferece diferentes níveis de interferência temporal.

Injetores com hardware específico usam software apenas para a sua configuração. Toda a atuação é realizada pelo próprio hardware. São os injetores que oferecem a menor interferência temporal. Entretanto, em geral, são os menos flexíveis, permitindo a aplicação de poucos tipos de falhas. Dessa forma, oferecem muitas dificuldades para injetar uma carga de falhas como aquela necessária para o teste de protocolos de segurança. Esses são injetores que não utilizam software no processo de injeção de falhas.

Quando se acrescenta software na implementação do injetor, encontra-se a categoria dos injetores com hardware e software específicos. No caso dos injetores de falhas em protocolos de comunicação, esses equipamentos são construídos de maneira que o hardware e o software se conjugam para aplicar cargas de falhas com a maior precisão possível. Apesar desse ótimo desenho, são também os mais caros.

No outro extremo encontram-se os injetores construídos a partir de hardware e software genéricos. Esses injetores permitem ao analista a construção de sua própria carga de falhas, a um custo significativamente menor do que aqueles com hardware específico: são extremamente flexíveis. Entretanto, dependendo do protocolo a ser testado, é necessário acrescentar pequenos adaptadores de hardware. Outra desvantagem desses injetores é que se tem pouco controle da interferência temporal, o que faz dele um injetor aceitável para teste de protocolos comuns de comunicação, mas pouco adequados para a injeção de falhas em protocolos de segurança.

5.4 Ambiente de avaliação

Para acomodar os requisitos de baixo custo e alta precisão, a arquitetura proposta para o ambiente baseia-se no uso de hardware genérico, com o uso eventual de adaptadores de hardware, e software específico. Com essa arquitetura pode-se construir injetores de custo comparável àquele dos injetores de hardware e software genéricos. Apesar disso, os injetores oferecem precisão comparável àquela dos injetores de hardware e software específicos, inclusive com a um bom controle sobre a interferência temporal.

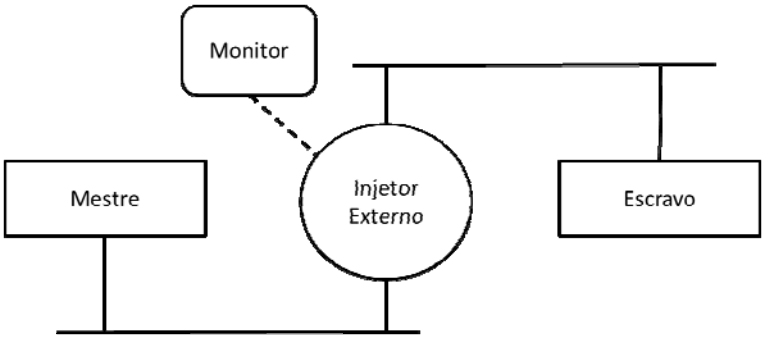

Figura 4 – Aplicação dos injetores externos

Na figura 4 está esquematizado o ambiente de aplicação proposto. No ambiente, aparecem um nodo “Mestre”, responsável por iniciar qualquer comunicação, e um “Escravo”, que sempre deve responder às solicitações do mestre. Esse modelo de comunicação visa garantir um alto nível de segurança das informações. O injetor de falhas externo aparece interceptando as mensagens trocadas entre mestre e escravo, de tal forma que nenhum deles é capaz de perceber a existência do injetor. Finalmente, os procedimentos de injeção de falhas podem ser configurados, controlados e monitorados, de maneira a promover a realização dos testes especificados.

Essa arquitetura foi utilizada em um trabalho de mestrado (Dobler 2016) do grupo de pesquisas, na implementação de um injetor de falhas para o protocolo “PROFIsafe”. Esse injetor será utilizado nas etapas de validação das implementações do protocolo e na verificação da compatibilidade entre equipamentos de diferentes fabricantes.

6 Conclusão

As normas de segurança estabelecem os requisitos necessários para assegurar que os sistemas sejam projetados e operados para suprir as exigências do SIL requerido para uma dada função de segurança. A comunicação é um componente importante de um sistema de instrumentação distribuído e a implementação dos protocolos de segurança precisam seguir as normas.

No Brasil, a crescente demanda por equipamentos certificados faz com o desenvolvimento de software seguro seja igualmente impulsionado. No momento em que as normas de segurança funcional permitiram o uso de controladores programáveis, uma vasta gama de oportunidades se abriu para desenvolvedores de software. Entretanto a norma restringe a aplicação das técnicas e procedimentos limitando-as àquelas comumente usadas por desenvolvedores e testadores de aplicações convencionais, o que torna a implementação e validação de protocolos seguros uma tarefa árdua.

As normas de segurança exigem o emprego de injeção de falhas. Para facilitar essa tarefa, no âmbito do projeto SDCD-NG, está em fase final de desenvolvimento um injetor de falhas para validação da implementação de protocolos seguros, que segue o ambiente de validação proposto neste artigo. Nesta primeira versão do ambiente escolheu-se validar implementações do PROFIsafe seguindo demandas da empresa parceira do projeto. A próxima extensão prevista no projeto para o ambiente deverá permitir validação do Safety-over-EtherCAT.

Para atender as necessidades do projeto, a arquitetura atende aos requisitos de baixo custo e alta precisão. Estes requisitos foram alcançados através de uma criteriosa fusão entre as características de injetores com hardware e software específicos, que são muito precisos porem caros, com injetores com hardware e software genéricos, que oferecem os menores custos. A arquitetura proposta neste artigo foi empregada no desenvolvimento de um injetor real para o protocolo PROFIsafe e está atualmente em fase final de depuração.

7 Bibliografia

Alemzadeh, Homa, Jai Raman, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2013. “Analysis of Safety-Critical Computer Failures in Medical Devices.” *IEEE Security & Privacy* 99 (1): 1.

Bell, Ron. 2006. “Introduction to IEC 61508.” In *Proceedings of the 10th Australian Workshop on Safety Critical Systems and software-Volume 55*, 3–12. Australian Computer Society, Inc.

———. 2011. “Introduction and Revision of IEC 61508.” In *Advances in Systems Safety*, edited by Chris Dale and Tom Anderson, 273–91. Springer London.

Bilich, Carlos, and Zaijun Hu. 2009. “Experiences with the Certification of a Generic Functional Safety Management Structure According to IEC 61508.” In *Computer Safety, Reliability, and Security*, edited by Bettina Buth, Gerd Rabe, and Till Seyfarth, 5775:103–17. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Carvalho, José, Adriano Carvalho, and Paulo Portugal. 2005. “Assessment of PROFIBUS Networks Using a Fault Injection Framework.” *Proceedings of 10th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2005*, 415–23.

Cotroneo, Domenico, and Roberto Natella. 2013. “Fault Injection for Software Certification.” *Security & Privacy, IEEE* 11 (4): 38–45.

Dawson, S., F. Jahanian, T. Mitton, and T. L. Tung. 1996. “Testing of Fault-tolerant and Real-time Distributed Systems via Protocol Fault Injection.” In *Fault Tolerant Computing, 1996., Proceedings of Annual Symposium On*, 404–14.

Dobler, R.J. 2016. "FITT: Uma Ferramenta de Injeção de Falhas para Validar Protocolos de Comunicação Seguros". Dissertação de mestrado. UFRGS.

Drebes, R.J., Gabriela Jacques-Silva, Joana Matos Fonseca Da Trindade, and Taisy Silva Weber. 2006. “A Kernel-based Communication Fault Injector for Dependability Testing of Distributed Systems.” In *1st International Haifa Verification Conference on Hardware and Software, Verification and Testing, November 13, 2005 - November 16, 2005*, 3875 LNCS:177–90. Lecture Notes in Computer Science. Haifa, Israel: Springer Verlag.

Dunn, W.R. 2003. “Designing Safety-critical Computer Systems.” *Computer* 36 (11): 40– 46.

Esposito, Christian, Domenico Cotroneo, and Nuno Silva. 2011. “Investigation on Safety-Related Standards for Critical Systems.” In *Software Certification (WoSoCER), 2011 First International Workshop On*, 49–54. IEEE.

Fowler, Derek, and Phil Bennett. 2000. “IEC 61508 — A Suitable Basis for the Certification of Safety-Critical Transport-Infrastructure Systems ??” In *Computer Safety, Reliability and Security*, edited by Floor Koornneef and Meine van der Meulen, 1943:250–63. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Gall, Heinz, and Joachim Wen. 2010. “Functional Safety IEC 61508 and Sector Standards for Machinery and Process Industry the Impact to Certification and Users Including IEC 61508 2nd Edition.” In *23rd International Congress on Condition Monitoring and Diagnostic Engineering Management, COMADEM 2010, June 28, 2010 - July 2, 2010*, 73–81. Nara, Japan: Sunrise Publishing Limited.

Gessner, D., M. Barranco, A. Ballesteros, and J. Proenza. 2014. “sfiCAN: A Star-Based Physical Fault-Injection Infrastructure for CAN Networks.” *IEEE Transactions on Vehicular Technology* 63 (3): 1335–49.

Hardy, Terry L. 2014. “Case Studies in Process Safety: Lessons Learned from Software related Accidents.” *Process Safety Progress* 33 (2): 124–30.

Hsueh, Mei-Chen, T.K. Tsai, and R.K. Iyer. 1997. “Fault Injection Techniques and Tools.” *Computer* 30 (4): 75–82.

Hurtig, Per, and Anna Brunstrom. 2008. “Enhancing SCTP Loss Recovery: An Experimental Evaluation of Early Retransmit.” *Computer Communications* 31 (16): 3778–88.

Lloyd, M. H., and P. J. Reeve. 2009. “IEC 61508 and IEC 61511 Assessments-some Lessons Learned.” *4th IET International Conference on Systems Safety*, 2A1.

Mayr, Alois, Reinhold Plosch, and Matthias Saft. 2011. “Towards an Operational Safety Standard for Software: Modelling IEC 61508 Part 3.” In *18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, ECBS 2011, April 27, 2011 - April 29, 2011*, 97–104. Las Vegas, NV, United states: IEEE Computer Society.

Natella, Roberto, Domenico Cotroneo, and Henrique S. Madeira. 2016. “Assessing Dependability with Software Fault Injection: A Survey.” *ACM Computing Surveys (CSUR)* 48 (3): 44.

Neumann, Peter. 2007. “Communication in Industrial automation—What Is Going On?” *Control Engineering Practice* 15 (11): 1332–47.

Pintard, Ludovic, Jean-Charles Fabre, Karama Kanoun, Michel Leeman, and Matthieu Roy. 2013. “Fault Injection in the Automotive Standard ISO 26262: An Initial Approach.” In *Dependable Computing*, edited by Marco Vieira and João Carlos Cunha, 126–33. Lecture Notes in Computer Science 7869. Springer Berlin Heidelberg.

Schirmeier, H., M. Hoffmann, R. Kapitza, D. Lohmann, and O. Spinczyk. 2012. “Fail #x2217;: Towards a Versatile Fault-injection Experiment Framework.” In *ARCS Workshops (ARCS), 2012*, 1–5.

Siqueira, Torgan, Bruno Fiss, Raul Weber, Sergio Cechin, and Taisy Weber. 2009. “Applying FIRMAMENT to Test the SCTP Communication Protocol Under Network Faults.” In *2009 10th Latin American Test Workshop, LATW 2009, March 2, 2009 - March 5, 2009*. Rio de Janeiro, Brazil: Inst. of Elec. and Elec. Eng. Computer Society.

Vidal, William, Rodrigo Dobler, Sérgio Cechin, Taisy Weber, and João Netto. 2014. “Aplicação Da IEC 61508 Na Prototipação de Protocolos Seguros de Comunicação.” In *Workshop de Testes e Tolerância a Falhas*, 147–59. Florianópolis: Sociedade Brasileira de Computação.

Zhivich, Michael, and Robert K. Cunningham. 2009. “The Real Cost of Software Errors.” *Security & Privacy, IEEE* 7 (2): 87–90.