INFIMO: Um Injetor de Falhas de Comunicação para Aplicações RT-Linux

Patrícia Pitthan de A. Barcelos[1] Taisy Silva Weber[2]

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

Resumo

Este artigo discute abordagens de implementação de injetores de falhas para validação de mecanismos de tolerância a falhas em protocolos de comunicação voltados para aplicações tempo real. O trabalho concentra-se em minimizar o impacto da carga do injetor de falhas no sistema sob teste e a intrusão tanto no protocolo quanto na plataforma que o executa. O artigo apresenta ainda o *INFIMO (INtrusiveless Fault Injection MOdule)*, um injetor de falhas que está sendo desenvolvido para a plataforma RT-Linux usando os recursos desse sistema operacional.

Palavras-chave: Tolerância a Falhas, Sistemas Distribuídos, Injeção de Falhas, Tempo Real.

Abstract

This article discusses approaches to implement fault injectors that will be used to validate mechanisms of fault tolerance in communication protocols directed to real time applications. The work focuses on minimizing the impact of the load of the fault injector in the system under test and the intrusion so in the protocol as in the platform that executes the protocol. The article also presents the *INFIMO (INtrusiveless Fault Injection MOdule)*, a fault injector under development to RT-Linux platform using these operating system resources.

Key Words: Fault-tolerance, Distributed Systems, Fault Injection, Real Time.

1 Introdução

Um sistema tolerante a falhas caracteriza-se pela capacidade de fornecer, de forma contínua, o serviço especificado, mesmo na ocorrência de falhas. No projeto desses sistemas surgem problemas relacionados à validação, uma vez que, além das funcionalidades normais, os mecanismos de tolerância a falhas devem ser validados [MAR93].

Para testar um sistema distribuído deve-se forçar o mesmo a determinados estados para garantir que caminhos de execução específicos sejam seguidos. Por outro lado, pode-se forçar o sistema a um comportamento que, sob condições normais, não aconteceria [DAW95].

Validação visa garantir a confiança na capacidade do sistema em fornecer o serviço especificado. Este é o principal objetivo da injeção de falhas. Injetar falhas consiste em introduzir falhas no sistema controladamente, a fim de observar seu comportamento [STE97].

Este artigo discute a implementação de injetores de falhas por *software* para validação de mecanismos de tolerância a falhas em protocolos de comunicação. O artigo apresenta ainda um injetor de falhas, o qual está sendo desenvolvido buscando uma solução que minimize o impacto no injetor no sistema. Para tanto, está sendo utilizado o sistema operacional de tempo real RT-Linux, que viabiliza acesso e alteração nas rotinas de manipulação de prioridades e escalonamento de tarefas.

2 Injeção de Falhas

Como falhas de especificação, falhas de projeto e falhas inevitáveis nos componentes de *hardware* podem ser fatais durante a operação de um sistema computacional, é preciso garantir o comportamento confiável desses sistemas. Técnicas de verificação formal não são ainda aplicáveis, com custo aceitável, a sistemas distribuídos com alto grau de interação, sujeitos a falhas imprevisíveis. Nesse caso, o teste assume um papel imprescindível. Uma forma de testar sistemas quanto ao seu comportamento sob falhas é através da injeção de falhas [ARL90].

---

[1] Doutoranda do PPGC da UFRGS; Professora da Faculdade de Informática da PUCRS
[2] Doutora em Computação (Karlsruhe, 1986); Professora orientadora do PPGC da UFRGS

O injetor de falhas é um processo, um conjunto de rotinas ou um objeto, que emula falhas de *hardware* corrompendo o estado do protocolo em execução. O objetivo de injeção de falhas por *software* é modificar o estado do sistema, levando-o a se comportar como se falhas de *hardware* estivessem presentes. O injetor interrompe ou altera a execução do protocolo e executa seu próprio código, emulando falhas pela inserção de erros no sistema.

As falhas são injetadas durante a execução do protocolo e o injetor fornece dados sobre o comportamento do protocolo em presença das falhas injetadas. A análise destes dados indica se o protocolo atende à especificação, ou seja, se realmente mascara ou recupera-se das falhas a que se propôs na fase de projeto.

Entretanto, a validação tanto em sistemas convencionais, como em sistemas tempo real, se depara com limitações espaciais e temporais inerentes a estes sistemas. Por representar uma carga extra no sistema, o injetor de falhas pode alterar o tempo de execução das tarefas comprometendo, desta forma, suas restrições temporais. Assim, durante o desenvolvimento de um injetor, cuidados especiais devem ser tomados para determinar a carga que esse representa ao sistema e minimizar seus efeitos sobre as medidas de confiabilidade e cobertura de falhas.

3 Abordagens para implementação de injetores de falhas

Injetores de falhas por *software* não podem ser implementados sem alterar a carga do sistema e sem influir em maior ou menor grau nas características temporais do sistema. Usualmente o injetor é construído alterando o código do protocolo sob validação, inserindo procedimentos que interrompem o processamento normal do protocolo para provocar a falha desejada. Esses procedimentos (rotinas, processos ou objetos) podem ser executados como aplicações ou como componentes do sistema operacional.

Pode-se estabelecer três níveis possíveis para implementação do injetor de falhas: no meta-nível, no nível da aplicação e no nível do sistema operacional [BAR99a]. Este artigo explora as abordagens de implementação nos níveis da aplicação e do sistema operacional.

A inserção do injetor de falhas no nível da aplicação introduz falhas na própria aplicação sob teste. Esta abordagem apresenta vantagens principalmente quando o protocolo e o injetor são desenvolvidos em conjunto. Pode-se implementar de várias formas: a aplicação e o injetor formam um único processo; o injetor de falhas e a aplicação correspondem a dois processos distintos interativos; ou o injetor e o protocolo são *threads* de um mesmo processo.

Na implementação por rotinas de um único processo o protocolo sob teste deve incluir, em seu código, chamadas às rotinas do injetor. A cada chamada ao injetor, o protocolo desvia sua operação normal, executa ações do injetor e então retorna a sua execução. Para injeção de falhas de comunicação, todas as chamadas a rotinas de comunicação, como *send, receive, broadcast, multicast, deliver* e outras funcionalmente semelhantes podem ser substituídas por rotinas do injetor. Alternativamente, as próprias rotinas de comunicação do protocolo podem ser alteradas.

Outra abordagem é quando o processo correspondente ao protocolo sob teste e o processo injetor executam concorrentemente. Somente haverá interação entre os dois processos na inserção de uma falha e na posterior coleta de resultados da inserção. Entretanto, tais procedimentos não implicam em alterações no código dos processos. Para interagir com o protocolo sem alterar o código do mesmo, o processo injetor deve receber privilégios especiais para alterar determinadas áreas de memória do processo protocolo. Determinar as áreas e obter privilégios de acesso a outros processos não é uma tarefa trivial para processos no nível de aplicação e está relacionada aos recursos que um sistema operacional deve oferecer. Se entretanto, for prevista no desenvolvimento do protocolo a validação por um injetor, estas áreas podem ser declaradas como áreas de acesso comum a mais de um processo.

A implementação por *threads* combina características das duas abordagens anteriores e visa construir o protocolo e o injetor como *threads* de um mesmo processo. Esta abordagem permite, como a primeira, acesso ao espaço de endereçamento de memória do protocolo sob teste, sem necessidade de privilégios especiais, e evita o custo do chaveamento de contexto na troca de processos da segunda abordagem. A intrusão do injetor no código fonte do protocolo sob teste é mínima, se restringindo a criação das *threads*.

Com injeção de falhas no nível do sistema operacional o injetor poderá atuar, por exemplo, através de bibliotecas do sistema, através de *system calls* ou através da manipulação de outros recursos executáveis do sistema. Cada vez que o protocolo utilizar recursos do sistema operacional, o injetor de falhas poderá alterar as ações em seu favor. Esta abordagem permite que o protocolo sob teste execute sem qualquer interferência do injetor de falhas, o que aumenta o reuso do injetor. A portabilidade da ferramenta está vinculada à portabilidade do sistema operacional usado como plataforma. Entretanto, por encontrar-se no nível do sistema operacional e utilizar-se dele para atuar, o injetor de falhas pode interferir em sua integridade. Além disso, há necessidade de alterações do próprio sistema operacional, que deve suportar a presença de intrusos.

4 INFIMO

A implementação do INFIMO combina a utilização dos recursos do escalonador com o uso de *threads* e *system calls*. A idéia é inicialmente gerar duas *threads*, a *thread* protocolo e a *thread* injetora. Com esta implementação o protocolo de comunicação sob teste e o INFIMO correspondem a *threads* de um mesmo processo. O conceito de *threads* permite acesso ao espaço de endereçamento de memória do protocolo sob teste, sem a exigência de privilégios especiais. Além disso, com *threads* evita-se o custo do chaveamento de contexto na execução alternada de processos [BAR99b].

Esta implementação exige que o sistema operacional suporte *threads*. Neste caso, quando a *thread* protocolo sob teste abandona a CPU, a *thread* do INFIMO pode executar e liberar a CPU antes do protocolo voltar a executar. Com isso, o impacto sobre a temporização do sistema é baixo, considerando que o INFIMO executa tarefas rápidas cada vez que assume a CPU.

A *thread* injetora deve fazer uso dos momentos em que a *thread* protocolo realiza uma chamada à rotina do sistema operacional (*system call*). O RT-Linux possibilita esta implementação, uma vez que permite acesso a suas rotinas. Com esta implementação, o INFIMO deve atuar de forma a desviar a operação normal do protocolo sob teste. No lugar de utilizar as primitivas de comunicação convencionais dos sistemas operacionais (*send, receive, deliver, ...*) o protocolo sob teste é desviado para a execução das primitivas de comunicação do INFIMO, cujo objetivo é implementar o modelo de falhas de comunicação a ser validado.

Os procedimentos entre *threads* devem ser controlados pelo escalonador do sistema, sob o qual o INFIMO deve possuir acesso. Os objetivos desta implementação são priorizar as atividades do INFIMO em relação as atividades do protocolo sob teste e garantir mínima interferência nas restrições temporais do protocolo. A intrusão do INFIMO no código fonte do protocolo sob teste é pequena, se restringindo à criação das *threads*.

O INFIMO utiliza ainda os recursos do escalonador para priorizar as atividades do injetor de falhas em relação ao protocolo sob teste, tentando não ferir as restrições temporais inerentes ao protocolo. O escalonador dispara o injetor. O injetor em si deve ter características especiais que o habilitem a avaliar e alterar prioridades de processos dinamicamente. As rotinas que injetam falhas, uma vez escalonadas e de posse da CPU, atuam de forma convencional selecionando ou manipulando mensagens conforme especificado no experimento.

5 Plataforma de Implementação do INFIMO

A plataforma escolhida para implementação do INFIMO é o sistema operacional RT-Linux. O RT-Linux permite a criação de tarefas de tempo real que têm a sua execução garantida mesmo se o sistema estiver executando uma chamada de sistema do *kernel*. Estas tarefas são implementadas como módulos do *kernel*, tendo assim acesso a todas as suas estruturas internas.

A idéia básica do RT-Linux é executar o Linux convencional sob controle do *kernel* tempo real. Este *kernel* deve ser não preemptivo. Quando há tarefa tempo real a ser realizada, o sistema operacional tempo real deve executá-la. No caso de não haver tarefa tempo real a ser realizada, o *kernel* tempo real escalona o Linux convencional para execução. Portanto, o Linux é a tarefa de mais baixa prioridade do *kernel* tempo real. Assim, o Linux convencional somente é executado quando o Linux tempo real não tiver nenhuma tarefa para executar.

O INFIMO consiste em três módulos: gerente, injetor e monitor. O módulo gerente recebe comandos de processos do usuário, dispara e controla a execução do módulo injetor. O módulo injetor executa efetivamente a inserção de falhas no protocolo alvo, enquanto o monitor informa aos processos do usuário o estado da experimentação.

O módulo gerente é responsável por garantir que a execução do módulo monitor não interfira nos *deadlines* das tarefas responsáveis pelo protocolo. Assim, é necessário especificar quando e por quanto tempo o módulo injetor pode ser executado.

Para a comunicação dos módulos com os processos do usuário existem quatro alternativas: chamada de sistema específica, extensão de uma chamada de sistema existente, *named pipes* (FIFO's) e memória compartilhada. A escolha da alternativa mais apropriada tem sido alvo de investigação, uma vez que depende de fatores como latência na comunicação, nível de interferência no escalonamento das tarefas de tempo real, compatibilidade com padrões existentes e portabilidade.

A idéia central é ter o módulo gerente como a tarefa de mais alta prioridade, de forma que este tenha total controle sobre o escalonamento das demais tarefas. Através do uso de *timers* de alta resolução, também providos pelo RT-Linux, o módulo gerente terá o controle exato do tempo de duração de cada tarefa. Desta forma, é garantida não interferência da injeção de falhas sobre os *timings* do protocolo.

6 Conclusões

Com injeção de falhas é possível medir a eficiência dos mecanismos de detecção, correção e recuperação de erros no sistema. Vantagens da injeção de falhas por sofware estão associadas ao baixo custo, baixa complexidade de desenvolvimento, portabilidade e fácil expansibilidade para novos tipos de falhas. Além disso, não apresenta problemas com interferências externas e riscos de danificar o sistema sob teste.

O INFIMO explora uma combinação de abordagens de implementação visando a diminuição da intrusão provocada por um módulo extra no sistema: o módulo injetor de falhas. O trabalho visa a utilização de recursos do sistema operacional para otimização do tempo de execução exigido pelo injetor de falhas. Neste sentido, a utilização do RT-Linux como plataforma de implementação apresenta dois principais propósitos. O primeiro está relacionado ao aproveitamento dos recursos de escalonamento embutidos no sistema. Com isso, pode-se manipular as prioridades das tarefas em função da aplicação tempo real. O segundo propósito se refere à possibilidade de acesso (e alteração, quando necessário) às rotinas do *kernel*. Sendo assim, pode-se implementar a abordagem de injeção de falhas através de alterações nas chamadas de sistema do RT-Linux.

A opção por uma versão tempo real do sistema operacional Linux visa impor as limitações temporais exigidas pelas aplicações tempo real.

Referências Bibliográficas

[ARL90] ARLAT, J. et.al. Fault Injection for Dependability Validation: A Methodology and Some Applications. IEEE TOSE, v. 16, n.2, Feb. 1990.

[BAR99a] BARCELOS, P. P. A.; LEITE, F. O.; WEBER, T. S. Building a Fault Injector to Validate Fault Tolerant Communication Protocols, PCS'99 (Parallel Computing Symposium), Ensenada, Baja California, México, 1999.

[BAR99b] BARCELOS, P. P. A. Análise da Viabilidade da Implementação de um Injetor de Falhas de Comunicação Não Intrusivo. Proposta de Tese de Doutorado, PPGC da UFRGS, 1999.

[BEC98] BECK, M. et.al. Linux Kernel Internals. 2nd Ed., Eddison Wesley, 1998.

[DAW95] DAWSON, S.; JAHANIAN, F. Probing and Fault Injection of Protocol Implementations. Technical Report CSE-TR-217-94. The University of Michigan, 1994.

[MAR93] MARTINS, E. Validação Experimental da Tolerância a Falhas: a técnica de injeção de falhas. V SCTF, Mini Curso, p. 56-70, 1993.

[STE97] STEFANI, M. R.; MARTINS, E. Análise de Traço e Geração de Diagnósticos para Testes Baseados em Injeção de Falhas por Software. VII SCTF, p. 321-336, 1995.