Um Toolkit para Avaliação da Intrusão de Métodos de Injeção de Falhas

Patrícia Pitthan A. Barcelos[1] Taisy Silva Weber[2] Roberto Jung Drebes[3]

{pitthan@exatas.unisinos.br}
Instituto de Informática, UNISINOS
C. Postal 275, São Leopoldo – RS, Brasil

{pitthan, taisy, drebes@inf.ufrgs.br}
Instituto de Informática, UFRGS
C. Postal 15064, Porto Alegre – RS, Brasil

Abstract

This paper presents implementation and experiments of fault injection used to validate mechanisms of fault tolerance in communication protocols with temporal restrictions. It describes two approaches. The first implements fault injection through modifications in the application libraries, which ones support the target protocol. The second approach uses operating system resources. Also, the paper shows practical experiments using the fault injectors to validate a timeout mechanism of a communication protocol. The fault injectors are part of the INFIMO project, a toolkit to fault injection experimentation developed under Linux platform.
Keywords: Fault-tolerance, Validation, Fault Injection, Communication Protocol, Linux.

1 Introdução

Técnicas de tolerância a falhas visam aumentar a dependabilidade dos sistemas nos quais são empregadas. Entretanto, há necessidade de garantir a confiança na capacidade do sistema em fornecer o serviço especificado. A validação possui como objetivo propiciar esta garantia. Uma técnica de validação bastante utilizada é a injeção de falhas, que consiste na introdução controlada de falhas no sistema para observar seu comportamento. Injeção de falhas acelera a ocorrência das falhas em um sistema. Ao invés de esperar pela ocorrência espontânea das falhas, pode-se introduzi-las intencionalmente, controlando o tipo, a localização, o disparo e a duração. Injeção de falhas pode ser implementada por hardware, software ou simulação.

O artigo apresenta a validação, através da injeção de falhas por software, de um protocolo de comunicação. Enfoque especial é dado à intrusão resultante da inclusão do injetor junto ao protocolo, já que o mesmo apresenta restrições temporais que podem ser comprometidas pela atuação do injetor. Um *toolkit* para experimentos de intrusão da injeção de falhas é apresentado. O *toolkit*, denominado INFIMO (***IN****trusiveless* ***F****ault* ***I****njector* ***MO****dule*), visa analisar, de forma experimental, a intrusão temporal de injetores de falhas sobre um protocolo com característica tempo real.

A seção seguinte apresenta aspectos referentes a injeção de falhas por software e a intrusão de injetores de falhas. A seção 3 descreve o INFIMO *toolkit* e a arquitetura dos injetores de falhas. Na seção 4 é apresentado o protocolo INFIMO_TAP, enquanto nas seções 5 e 6 são descritos os injetores INFIMO_LIB e INFIMO_DBG, respectivamente. A seção 7 descreve experimentos de intrusão e a seção 8 esboça algumas conclusões.

2 Injeção de Falhas por Software

Injeção de falhas por software emula falhas de hardware através de software corrompendo o estado do programa em execução. Sob o ponto de vista de implementação, o injetor é um processo, conjunto de rotinas ou objeto, que interrompe ou altera a execução do sistema e executa seu próprio código, emulando falhas pela inserção de erros no sistema.

Em sistemas com característica tempo real, onde as aplicações por eles controladas devem ocorrer em instantes de tempo relativos a uma base de tempo externa, a injeção de

[1] Doutora em Ciência da Computação (UFRGS, 2001)
[2] Doutora em Informática (Univ. Karlsuhe, 1986)
[3] Bacharel em Ciência da Computação (UFRGS, 2001)

falhas se torna mais crítica. As restrições temporais impostas pelos sistemas aliadas a sobrecarga provocada pelo injetor de falhas justificam este fato.

2.1 Intrusão

Por representar uma carga extra no sistema, o injetor de falhas pode alterar o tempo de execução do protocolo, comprometendo suas restrições temporais. Esta execução de carga extra é referenciada na literatura como intrusão [CUN00]. Assim, durante o desenvolvimento de um injetor, cuidados especiais devem ser tomados para determinar a carga que esse representa e minimizar seus efeitos. Na maioria dos modelos de falhas, há necessidade de executar um código adicional no mesmo processador que executa a aplicação alvo. Como conseqüência, pode-se chegar a situações onde alguns limites de tempo são perdidos devido ao tempo de execução extra por parte das atividades de injeção de falhas [CAR98].

A intrusão do injetor de falhas no protocolo alvo pode ser observada de forma temporal e espacial. Na intrusão temporal tem-se o tempo de execução aumentado em virtude do acréscimo das atividades de injeção de falhas, as quais devem ser executadas juntamente com o protocolo alvo. Já na intrusão espacial a necessidade de modificar o código do protocolo alvo pode consistir em comportamento intrusivo. O artigo enfoca a análise da intrusão temporal do injetor de falhas sobre o protocolo alvo.

A intrusão temporal é avaliada de acordo com o *tempo de execução do protocolo*. No trabalho apresentado neste artigo, este tempo é medido através do relógio local do processador no qual o protocolo está executando. Para cada experimento toma-se o valor do relógio antes da execução do protocolo e após o término da mesma. Para se analisar a carga do sistema executa-se, para os mesmos dados de entrada, os seguintes experimentos: execução somente do protocolo alvo, execução do protocolo alvo e do injetor inativo e execução do protocolo alvo e do injetor ativo.

3 INFIMO *Toolkit*

O INFIMO (***IN****trusiveless* ***F****ault* ***I****njector* ***MO****dule*) é um *toolkit* para experimentos de injeção de falhas na validação da comunicação em protocolos com restrições temporais. O principal objetivo da implementação do INFIMO é analisar a intrusão imposta pelas atividades de injeção de falhas no comportamento temporal do protocolo alvo.

Este trabalho compreende dois métodos de validar um protocolo de comunicação através da injeção de falhas. Com isso, busca-se analisar a interferência das atividades de injeção de falhas no comportamento do protocolo alvo, o qual possui característica tempo real. O primeiro método de validação utiliza-se das API's do protocolo alvo (biblioteca JRTPLIB), enquanto o segundo faz uso de recursos do sistema operacional.

O INFIMO é implementado em plataforma Linux. Os experimentos apresentados, assim como a implementação do *toolkit*, foram realizados em computadores PC Pentium MMX 233Mhz, 64MB de RAM, conectados via Ethernet 10Mbps, rodando Linux 2.2.14.

O *toolkit* é formado por três componentes: um protocolo alvo e dois injetores de falhas. O protocolo, denominado INFIMO_TAP (***IN****trusiveless* ***F****ault* ***I****njector* ***MO****dule* – ***TA****rget* ***P****rotocol*), foi desenvolvido para experimentos com o *toolkit* e encontra-se descrito na seção 4. Os injetores de falhas INFIMO_LIB (***IN****trusiveless* ***F****ault* ***I****njector* ***MO****dule – by* ***LIB****rary modifications*) e INFIMO_DBG (***IN****trusiveless* ***F****ault* ***I****njector* ***MO****dule – using* ***DeBuG*** *resources*) são apresentados nas seções 5 e 6.

Neste contexto, cabe ressaltar três objetivos: do INFIMO, do protocolo alvo e dos injetores de falhas. O INFIMO visa analisar a intrusão dos injetores de falhas INFIMO_LIB e INFIMO_DBG sobre o protocolo alvo INFIMO_TAP. O protocolo INFIMO_TAP viabiliza a

troca de pacotes entre processos, controlando, através do mecanismo de detecção do *timeout*, a recepção destes pacotes. Os injetores INFIMO_LIB e INFIMO_DBG têm como objetivo validar o mecanismo de detecção do *timeout* implementado pelo protocolo alvo.

3.1 Arquitetura do INFIMO

A Figura 1 apresenta a arquitetura dos injetores INFIMO_LIB e INFIMO_DBG, baseada na arquitetura genérica para injetores de falhas proposta por Hsueh [HSU97].

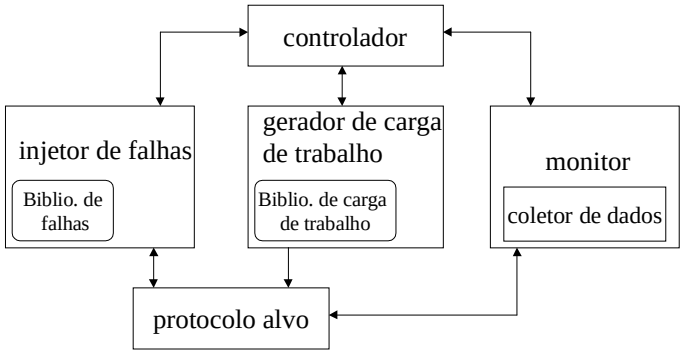

Figura 1 – Arquitetura de Injeção de Falhas do INFIMO

Os injetores do INFIMO implementam os componentes da arquitetura da Figura 1 através de procedimentos incorporados aos códigos dos mesmos. O protocolo INFIMO_TAP oferece procedimentos que realizam as funções do *gerador de carga de trabalho* e da *biblioteca de carga de trabalho*. Os injetores INFIMO_LIB e INFIMO_DBG possuem procedimentos que executam as funções dos componentes: *injetor de falhas*, *biblioteca de falhas*, *controlador*, *monitor* e *coletor de dados*.

O *protocolo alvo* é a um protocolo de comunicação que apresenta restrições temporais e possui um mecanismo de tolerância a falhas, o mecanismo de detecção de *timeout*.

Em ambos os injetores de falhas, a *biblioteca de falhas* pertence ao *injetor de falhas*. Esta biblioteca possibilita a especificação do modelo de falhas, composto pelas informações referentes ao tipo, localização, duração e disparo das falhas.

Para o INFIMO_LIB, o *injetor de falhas* é implementado através de alterações nas funções da biblioteca JRTPLIB, a qual é implementada sobre o protocolo RTP. As alterações realizadas visam a injeção de falhas de acordo com a especificação da biblioteca de falhas. No INFIMO_DBG, o *injetor de falhas* usa o *ptrace( )* para injetar as falhas no protocolo alvo. O *ptrace( )* permite que o injetor controle e manipule a execução do protocolo.

O *gerador de carga de trabalho* estabelece o número de processos participantes da comunicação e o número de pacotes envolvidos. O gerador de carga de trabalho conta com o auxílio da *biblioteca de carga de trabalho*.

O *controlador* é um procedimento que dispara a injeção de falhas. No INFIMO_LIB o disparo é feito a cada acesso à biblioteca alterada. No INFIMO_DBG o disparo é realizado a cada chamada ao *ptrace( )*. Em ambos os injetores, o disparo é realizado com base na especificação do modelo de falhas a ser validado.

O *monitor* é um procedimento que observa o andamento do experimento, coleta dados sobre o mesmo e salva-os em arquivos de *log*. O *coletor de dados* pertence ao *monitor*.

Em ambas as implementações, os injetores de falhas são processos criados pelo sistema operacional através da linha de comando. O processo que implementa o injetor de falhas INFIMO_LIB incorpora em seu código fonte o código do protocolo INFIMO_TAP. O

processo que implementa o injetor INFIMO_DBG cria um processo filho, o processo a ser monitorado. Este processo executa o código do protocolo alvo INFIMO_TAP.

4 Protocolo alvo: INFIMO_TAP

O INFIMO_TAP é o alvo da validação por injeção de falhas. O INFIMO_TAP é um protocolo coordenado de troca de pacotes, que possui restrições temporais e tolerância a falhas. Para sua implementação, faz-se uso das funções da biblioteca JRTPLIB [JRT99], a qual é implementada sobre o protocolo RTP (*Realtime Transport Protocol*) [SCH97]. O mecanismo de tolerância a falhas do INFIMO_TAP consiste na detecção do *timeout* associado aos pacotes. O objetivo da validação é a detecção do *timeout*.

A execução do protocolo é disparada pela linha de comando, através da qual o sistema cria os processos do protocolo. Um processo do INFIMO_TAP corresponde a um conjunto de *threads*. Para cada processo existe uma *thread* principal, uma *thread* de resposta e *n*-1 *threads* de envio, onde *n* é o número de processos. Associado à *thread* principal há o *signal_handler*, uma espécie de vetor de interrupções responsável pela coordenação do *timer*.

O *gerador de carga de trabalho* é responsável pela criação das *threads*, sua associação aos processos, e geração do número de pacotes a serem enviados. O destino dos pacotes é irrelevante. Após a geração da carga de trabalho, dá-se início à comunicação propriamente dita. A Figura 2 ilustra a seqüência de envio de um pacote A do processo 2 para o processo 4. O envio se dá através de uma *thread* de envio do processo origem (processo 2), enquanto a recepção é feita pela *thread* de resposta do processo destino (processo 4). A *thread* de envio do processo 2, responsável pelo destino 4 é a *thread* de envio 4.

Conforme a Figura 2, no tempo t1, a *thread* de envio 4 (processo 2) envia o pacote A. Ao enviar o pacote, a *thread* 4 adiciona um *timer* ao pacote e coloca este *timer* em uma fila. No tempo t2, a *thread* de resposta (processo 4) recebe o pacote A, armazena-o e envia um ACK ao processo 2. O ACK é recebido pela *thread* de resposta (processo 2) no tempo t3. Então, a *thread* de resposta (processo 2) remove o *timer* do pacote A da fila. Toda a seqüência de envio de pacote e recebimento do ACK é realizada antes do *timeout* expirar (tempo t4). As ações de cada processo do protocolo são armazenadas em arquivos de *log*.

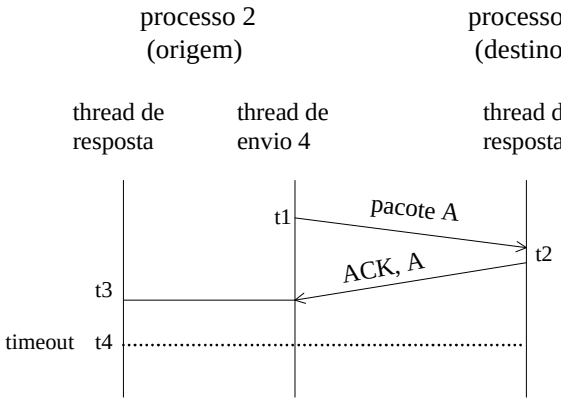

Figura 2 – INFIMO_TAP: Envio de pacote

5 INFIMO_LIB

O INFIMO_LIB é um injetor de falhas implementado através de alterações nas funções da biblioteca JRTPLIB sobre a qual está implementado o protocolo alvo (INFIMO_TAP). Assim como no INFIMO_TAP, no INFIMO_LIB também existe um conjunto de processos, onde cada um possui um conjunto de *threads*. Entretanto, enquanto os

processos do INFIMO_TAP utilizam as funções da biblioteca JRTPLIB, os processos do INFIMO_LIB usam funções modificadas da JRTPLIB.

Enquanto o INFIMO_TAP corresponde a um protocolo de troca de pacotes, o INFIMO_LIB corresponde ao INFIMO_TAP acrescido de atividades de injeção de falhas. O INFIMO_LIB mantém a estrutura de funcionamento do INFIMO_TAP referente a: disparo dos processos, criação das *threads*, geração da carga de trabalho, seqüência de envio e reenvio, manipulação dos *timers* e geração de *logs*.

A execução do INFIMO_LIB é disparada pela linha de comando. O disparo do INFIMO_LIB deve ser feito para cada processo do protocolo. Ao ser executado, o INFIMO_LIB, através da *biblioteca de falhas*, interpreta o modelo de falhas, possibilitando assim, a execução efetiva da injeção de falhas. Os processos do INFIMO_LIB executam conforme os processos do INFIMO_TAP. De acordo com a especificação do modelo de falhas, a comunicação se dá através da execução de funções da JRTPLIB modificadas, ou seja, são executadas as funções de injeção de falhas do INFIMO_LIB.

Em ambos os injetores de falhas, para emular uma falha de *crash* no processo alvo, todos os pacotes oriundos deste processo são desviados para um destino não válido. O mesmo acontece na emulação de uma falha por omissão, entretanto, neste caso o desvio não ocorre para todos os pacotes do processo alvo, porém somente para alguns. O atraso pode ser emulado através da retenção do pacote por um determinado período de tempo.

O histórico de ações de cada processo do INFIMO_LIB é armazenado em arquivos de *log*. Este procedimento é realizado pelo *coletor de dados*, que atua em conjunto com o *monitor*. O *controlador* é o responsável por todo o gerenciamento do INFIMO_LIB, desde a inicialização do protocolo alvo até a coleta de dados do experimento.

Um aspecto a ser considerado na implementação desta abordagem se refere à integridade e, neste sentido, destacam-se duas preocupações. A primeira relaciona-se à integridade do protocolo alvo. Alterações nas funções da JRTPLIB não devem interferir no contexto do protocolo. A segunda preocupação refere-se à manutenção da semântica das funções da biblioteca JRTPLIB. Deve-se garantir que as alterações impostas pelas atividades de injeção de falhas não comprometam o comportamento das funções originais. Outro aspecto a ser considerado é a portabilidade da abordagem. A idéia é poder utilizar esta abordagem para validação de outros protocolos alvo, implementados também sobre a JRTPLIB.

Entretanto, a abordagem apresenta diminuição do controle da carga de trabalho, uma vez que se está implementando o injetor em um nível mais alto. Assim, a carga de trabalho é provavelmente mais alta do que se a implementação fosse desenvolvida em baixo nível, como no nível do sistema operacional.

6 INFIMO_DBG

O INFIMO_DBG baseia-se na utilização dos recursos de depuração apresentados pelo sistema operacional. O injetor utiliza o *ptrace( )* (padrão Unix) para desviar das operações normais do protocolo alvo, o INFIMO_TAP, para as atividades de injeção de falhas.

O mecanismo de injeção do INFIMO_DBG é manipulado através da interceptação da comunicação de dois processos que executam concorrentemente: processo injetor de falhas e processo alvo. O controle de execução exercido pelo injetor de falhas permite a injeção das falhas através das chamadas de sistema executadas pelo processo alvo.

A inicialização do INFIMO_DBG é similar a do INFIMO_LIB. A execução do INFIMO_DBG é disparada pela linha de comando. O disparo do INFIMO_DBG deve ser feito para cada processo do protocolo.

Ao ser executado, o INFIMO_DBG, através da *biblioteca de falhas*, interpreta o modelo de falhas. Um dos processos do protocolo assume a responsabilidade de injetar falhas,

tornando-se o processo injetor de falhas. O processo injetor de falhas tem total controle sobre o processo alvo podendo, portanto, atuar sobre o mesmo.

A Figura 3, que ilustra o funcionamento do INFIMO_DBG, apresenta um conjunto de processos: processo injetor de falhas, processo alvo e demais processos do protocolo. O processo injetor de falhas cria, através da chamada de sistema *fork( )*, o processo alvo. O processo alvo executa o mesmo código dos demais processos do protocolo. As ações do processo alvo podem ser interceptadas pelo processo injetor de falhas. Através da interceptação, o processo injetor de falhas decide, com base no modelo de falhas, sobre as atividades de injeção de falhas.

Na Figura 3 são mostradas duas situações de intercepção do processo alvo por parte do processo injetor de falhas. Na primeira, o injetor de falhas decide pela omissão do pacote enviado pelo processo alvo para um dos processos do protocolo. Esta situação ilustra os tipos de falha *crash* ou omissão. Na segunda interceptação, o injetor supõe o atraso no envio do pacote. Esta situação ilustra uma falha de temporização.

Ao ser executado, o processo alvo recebe do processo injetor de falhas os argumentos referentes ao modelo de falhas, os quais possibilitam a execução efetiva da injeção de falhas. O histórico de ações de cada processo do protocolo é armazenado em arquivos de *log*. O *monitor* e o *coletor de dados* são os responsáveis por esta função.

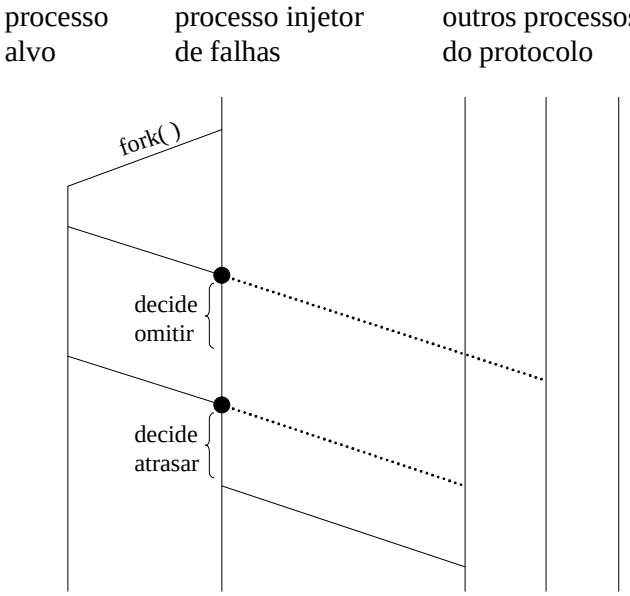

Figura 3 – Interceptação do processo alvo

O INFIMO_DBG baseia-se no uso do depurador do sistema operacional através da execução do protocolo alvo da validação até a chamada de sistema *sendto( )*. No momento em que encontra a chamada *sendto( )*, o INFIMO_DBG pode alterar os parâmetros da mesma para injetar a falha desejada. A coordenação das atividades do INFIMO_DBG é realizada pelo *controlador*, de acordo com a arquitetura exibida na seção 3.1.

Para utilizar o *ptrace( )*, o INFIMO_DBG alterou a estrutura de *threads* do INFIMO_TAP. Para o INFIMO_TAP há uma *thread* principal, que cria uma *thread* de resposta, a qual cria *n threads* de envio. A necessidade do *ptrace( )* atuar sobre o nível mais alto de *threads* fez com que as *threads* de envio fossem incorporadas à *thread* principal.

Esta abordagem de implementação apresenta como vantagens a portabilidade e a integridade. Por ser um injetor embasado no padrão Unix e utilizar recursos do sistema operacional para a injeção de falhas, é possível sua utilização para validação de outros

protocolos, implementados ou não sobre a JRTPLIB, porém compatíveis com o ambiente Unix. Quanto à integridade pode-se afirmar que o contexto do protocolo alvo não sofre interferência em função da subordinação ao injetor de falhas.

7 Intrusão temporal dos injetores de falhas

Esta seção descreve experimentos que mostram os *tempos de execução* dos injetores INFIMO_LIB e INFIMO_DBG, ou seja, a carga imposta pelas atividades dos injetores de falhas sobre o protocolo INFIMO_TAP. Consideram-se 3 processos, onde os processos livres de falhas enviam 27 e 24 pacotes e processo alvo envia 30 pacotes.

Os experimentos baseiam-se no modelo de falhas apresentado na Tabela 1. O tipo de falha corresponde ao que corromper e como corromper. O INFIMO enfoca falhas de comunicação, cuja localização inclui processos ou *links*. As falhas de comunicação ocorrem no envio de pacotes. A forma pela qual um processo (ou *link*) pode ser corrompido compreende a classe de falhas a ser validada, que pode ser falha de *crash* ou omissão. O INFIMO possui ainda suporte para implementação de falhas de temporização.

O disparo da falha relaciona-se a uma condição espacial ou temporal. Uma falha disparada espacialmente é ativada quando o protocolo alcança determinado estado. Uma falha disparada temporalmente torna-se ativa após um determinado período de tempo. O INFIMO possibilita ainda a definição da duração da falha, que pode ser transiente ou intermitente.

Tabela 1 – Modelo de falhas dos experimentos de intrusão

Argumento	**Descrição**
Tipo de falha	Falhas por omissão (falhas simples)
Localização da falha	Processo
Disparo da falha	Disparo espacial (a cada 2 pacotes)
Duração da falha	Falhas intermitentes

A Tabela 2 apresenta os experimentos relativos a intrusão dos injetores de falhas INFIMO_LIB e INFIMO_DBG. Os valores de tempo de execução exibidos são expressos em milisegundos. O valor definido para o *timeout* é de 500 ms.

Tabela 2: Experimentos de intrusão dos injetores

	INFIMO_LIB		**INFIMO_DBG**	
	Processos livres de falha	Processo alvo	Processos livres de falha	Processo alvo
a) somente protocolo	37,443	-	37,443	-
b) protocolo + injetor inativo	36,876	-	40,177	-
c) protocolo + injetor ativo				
1 falha	40,091	666,641	40,152	647,589
2 falhas	39,590	691,139	39,825	678,759
4 falhas	39,748	674,899	39,818	679,683
8 falhas	39,321	682,603	39,680	687,102
12 falhas	39,516	703,460	38,809	688,042

O experimento “a” (linha 3 da Tabela 2) apresenta a carga do protocolo alvo, ou seja, é medido o tempo de execução do protocolo alvo. Todos os processos estão livres de falhas, uma vez que o protocolo não está sendo validado por nenhum injetor de falhas.

No experimento “b” (linha 4 da Tabela 2) o protocolo e o injetor estão executando, porém o injetor de falhas executa com máscara de injeção nula. Assim, todas as atividades do injetor são realizadas, exceto a injeção propriamente dita. Em ambos os injetores de falhas, os

valores de tempo de execução mantém-se razoavelmente próximos. O INFIMO_DBG apresenta um pequeno acréscimo no seu tempo de execução, porém não implica em sobrecarga significativa. Este acréscimo está provavelmente relacionado à criação do processo injetor de falhas e as trocas de contexto entre os processos participantes do experimento.

O experimento "c" (linhas 5-9 da Tabela 2), apresenta a carga imposta pela injeção de falhas. Tanto no INFIMO_LIB como no INFIMO_DBG, para os processos livres de falhas, os tempos de execução obtidos para o envio de 27 e 24 pacotes, mantêm valores aproximados. Para todos os experimentos, os quais introduzem de 1 a 12 falhas, o *timeout* expirou. Considerando o processo alvo da injeção de falhas, percebe-se que os valores de tempo apresentados sofrem um elevado acréscimo logo após a injeção da primeira falha. Nesse acréscimo há que se considerar o valor do *timeout* (500 ms). Logo, o processo leva, em média, 183,748 ms para manipular a falha. Esta manipulação inclui, além da detecção da falha, a cópia do pacote, a remoção de seu *timer* da fila de *timers*, o reenvio do pacote e o acréscimo do novo *timer* no final da fila de *timers*. Observa-se ainda que o aumento progressivo no número de falhas injetadas implica em mínima variação no tempo de execução.

8 Conclusões

Diversas abordagens de implementação de injetores de falhas têm sido propostas na literatura. O diferencial do INFIMO concentra-se na característica tempo real do protocolo alvo, no escopo de falhas a ser validado e na idéia de *toolkit*, a qual possibilita a comparação entre duas abordagens de implementação.

Protocolos com característica tempo real constituem um desafio no contexto de injeção de falhas. Diferentemente de grande parte dos injetores de falhas ([CAR98], [CUN00], [KAN95], [KRI98]), os injetores do INFIMO enfocam falhas de comunicação, injetando falhas no envio de pacotes. RT_Xception [CUN00], apesar de tratar tempo real, se restringe a falhas de memória e processador, não se prestando à validação de protocolos de comunicação.

Além da condução de experimentos de injeção de falhas, o INFIMO caracteriza-se principalmente pela sua modularidade, possibilitando a expansão do *toolkit*. Por estar implementado sobre plataforma Linux e utilizar biblioteca não comercial, pode-se propor outros protocolos alvo e outras abordagens de implementação para o INFIMO. O INFIMO *toolkit* pode auxiliar ao desenvolvedor de protocolos de comunicação com característica tempo real na utilização dos injetores de falhas como técnica de validação.

Bibliografia

[CAR98] CARREIRA, J.; MADEIRA, H.; SILVA, J. G. Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers. IEEE TOSE, v. 24, n. 2, Fev. 1998.

[CHI89] CHILLAREGE, R.; BOWEN, N. S. Understanding Large-System Failures: A Fault-Injection Experiment. FTCS 19, Los Alamitos, California, p.356-363, 1989.

[CUN00] CUNHA, J. C.; RELA, M. Z.; SILVA, J. G. RT-Xception: Fault-Injection for Real-Time. Coimbra: Centro de Informatica e de Sistemas da Univ. de Coimbra, 2000. (TR-2000/002).

[HAN95] HAN, S. et. al. DOCTOR: an integrateD sOftware fault injeCTion enviRonment for Distributed RT systems. Int. Comp. Performance and Dependability Symp., Erlangen, Germany, 1995.

[HSU97] HSUEH, M. et. al. Fault Injection Techniques and Tools. IEEE Computer, v. 30, n. 4, Apr. 1997.

[JRT99] LIESENBORGS, J. Manual da JRTLIB. http://limumba.luc.ac.be/jori/jrtplib/jrtplib.html

[KAN95] KANAWATTI, G. A. et. al. FERRARI: A flexible software-based fault and error injection system. IEEE Transactions on Computers, v. 44, n. 2, Feb. 1995.

[KRI98] KRISHNAMURTHY, N. et. al. A design methodology for software fault injection in embedded systems. Int. Workshop Dependable Comp. and Applications, Johannesburg, South Africa, 1998.

[SCH97] SCHULZRINNE, H. G. Protocolo RTP. http://www.cs.columbia.edu/~hgs/RTP.