# Estendendo FIONA para uma arquitetura híbrida*

**Felipe Mobus, Roberto Jung Drebes, Gabriela Jacques-Silva,
Taisy Silva Weber, Ingrid Jansch-Pôrto**

Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15064 – 90501-970 Porto Alegre, RS, Brasil

{fmobus,drebes,gjsilva,taisy,ingrid}@inf.ufrgs.br

***Abstract.*** *This papers discusses the importance of fault injection as a tool for the validation of distributed applications and points how to integrate two fault injection tools for better interoperability in a hybrid distributed fault injection architecture.*

***Resumo.*** *Este trabalho discute a importância da injeção de falhas para a validação de aplicações distribuídas e aponta como integrar duas ferramentas de injeção de falhas para obter uma melhor interoperabilidade em uma arquitetura híbrida de injeção de falhas distribuída.*

## 1. Introdução

A crescente uso de sistemas distribuídos exige que estes sejam dotados de mecanismos apropriados de tolerância a falhas, garantindo seu funcionamento seguro. Aplicações distribuídas estão sujeitas a distúrbios de transmissão, podendo sofrer inconsistências e tendo seu comportamento imprevisível. É necessário garantir que estas aplicações tenham mecanismos de tolerância a falhas adequados antes de colocá-las em ambientes de produção. Para isto, o uso de componentes de boa qualidade e de arquiteturas de sistemas bem planejadas não é suficiente: o mal funcionamento de mecanismos de detecção e recuperação pode acarretar em resultados desastrosos [1].

A injeção de falhas emula a ocorrência de falhas que, de outra forma, ocorreriam com freqüência muito baixa, assim facilitando a observação de seus efeitos sobre o sistema, apontando seus gargalos de dependabilidade e determinando a cobertura dos mecanismos de detecção e recuperação de falhas [2]. Ao desenvolver injetores de falhas, é mais adequado usar injetores genéricos ao invés de desenvolver um injetor específico para cada aplicação, visando diminuir o custo associado à fase de validação.

Este trabalho demonstra a possibilidade de estender o injetor de falhas de arquitetura distribuída FIONA (*Fault Injector Oriented to Network Applications*) [3] de forma a torná-lo interoperável com outros injetores de falhas. Como exemplo desta extensão é utilizado o injetor ComFIRM (*Communication Fault Injector through OS Resource Modification*) [4]. A Seção 2 apresenta as duas ferramentas de injeção de falhas e suas abordagens distintas. Em seguida, é discutida uma forma de integrá-las para a injeção distribuída de falhas. O artigo encerra com algumas conclusões sobre esta integração.

*Desenvolvido em parceria com HP Brasil P&D e Projeto CNPq ACERTE (#472084/2003-8)

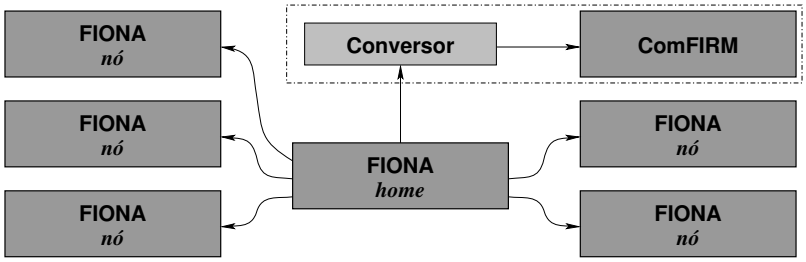


**Figura 1: Modelo de integração do FIONA com outros injetores de falhas**

## 2. Os injetores FIONA e ComFIRM

Neste artigo, são abordadas duas ferramentas de injeção de falhas distintas: FIONA e ComFIRM. FIONA é um injetor de falhas para aplicações Java distribuídas de larga escala. Sua abordagem para injeção de falhas é a instrumentação das classes de comunicação através de interfaces nativas de depuração da plataforma Java. A ferramenta permite a experimentação coordenada de injeção de falhas ocorridas em nós distintos. Para isto, uma instância de FIONA assume o papel de máquina *home* do experimento, e as demais tornam-se clientes de uma arquitetura distribuída, como injetores de nó.

ComFIRM, por sua vez, é destinado a aplicações para a plataforma GNU/Linux e funciona através da inspeção e manipulação de pacotes diretamente no interior do núcleo do sistema operacional. ComFIRM não apresenta arquitetura distribuída para injeção de falhas, mas, como mostrado a seguir, a ferramenta pode ser integrada à arquitetura FIONA como injetor de nó para aplicações não Java.

## 3. Estendendo FIONA para novos injetores de nó

FIONA é um injetor aplicável a sistemas cujos nós todos executem aplicações Java. Para usá-lo em sistemas híbridos, como um servidor de plataforma Linux e clientes em Java, é necessário estender FIONA permitindo o uso do injetor de falhas de nós distintos, compatíveis com as aplicações alvo. Se a integração for feita de forma transparente à máquina *home*, convertendo as especificações dos cenários de falhas nos nós da arquitetura distribuída e usando o mesmo protocolo de comunicação que um nó FIONA, a necessidade de modificação da ferramenta será mínima.

Para isto, foi desenvolvido um módulo responsável pela comunicação via o protocolo de um nó FIONA e pela tradução das regras de configuração de experimento transmitidas pela máquina *home*. Estas são convertidas em regras e comandos com a mesma semântica mas sintaxe da ferramenta de injeção utilizada nos nós que não utilizam FIONA, como ilustrado na Figura 1. Usando esta abordagem, não são necessárias modificações nas ferramentas envolvidas, beneficiando a modularidade. Para demonstrar a aplicabilidade dessa abordagem, exemplificamos a integração da ferramenta ComFIRM como caso de estudo. Técnicas semelhantes podem ser utilizadas para outras ferramentas.

## 4. Integração ComFIRM-FIONA

Ao desenvolver o módulo de conversão, é necessário distinguir as falhas injetadas por FIONA que necessitam de coordenação, como particionamento, das demais falhas.

```
teste  pacote.endereço == conexão.endereço
teste  pacote.porta == conexão.porta
teste  conexão.flag == 0
ação   conexão.flag := 1
ação   conexão.contador := 0
ação   conexão.temporizador := 0
ação   conexão.temporizador.modo := incrementar
```

**Figura 2: Regra de inicialização**

No primeiro caso, é preciso traduzir a mensagem de coordenação enviada aos nós envolvidos para comandos da ferramenta ComFIRM que produzam o comportamento esperado para o nó. No outro caso, é necessário traduzir do arquivo de parâmetros do cenário de falhas enviado ao nó para o formato de entrada de parâmetros exigido por ComFIRM.

### 4.1. Tradução dos cenários de falhas

ComFIRM analisa os pacotes de todas as aplicações em execução no sistema operacional. Como normalmente se deseja injetar falhas apenas em determinadas aplicações, é preciso verificar se os pacotes pertencem à aplicação alvo, testando os endereços de *socket* e só realizando a injeção se o *socket* pertence à aplicação em teste.

O teste de endereço pode ser especificado com poucas instruções da ferramenta ComFIRM. Ele consiste da leitura de campos específicos do pacote analisado e é fundamental para a seleção correta dos pacotes a serem tratados. Para traduzir para o ComFIRM o intervalo temporal de injeção de falhas, é utilizado, para cada regra traduzida, um contador e um temporizador, disponíveis no ComFIRM. Assim, para cada pacote da aplicação, é feito um incremento no contador e um teste de intervalo, ou seja, um teste do valor do contador/temporizador contra os limites inferior e superior do intervalo. Para a determinação da taxa de falhas, um valor aleatório é gerado por ComFIRM contra o qual também é feito um teste de limite. Sendo o pacote selecionado pelos testes acima descritos, a falha propriamente dita é inserida, de acordo com o tipo de falha indicada.

É necessário ainda criar uma regra de inicialização para cada falha traduzida, garantindo o início em zero dos valores do contador e do temporizador para a conexão. A construção de tal regra consiste em um teste de endereço para a conexão e um teste se o pacote atual é o primeiro da conexão, feito através de um *flag* referente a conexão. Em seguida, inicializa-se em zero o temporizador e o contador da conexão de forma que essa regra não seja novamente executada. A estrutura desta regra pode ser vista na Figura 2.

### 4.2. Coordenação de injeção de falhas

A emulação da ocorrência de falhas, como particionamento de rede, exige coordenação por se tratar de uma falha que afeta mais de um nó por vez. No caso deste tipo de falha, a máquina *home* envia a todos os nós a descrição do intervalo de ocorrência da falha e as máquinas envolvidas. A partir dessas informações, o módulo de conversão configura a ferramenta ComFIRM com as regras de injeção de falhas de colapso necessárias para causar o particionamento descrito pela máquina *home*.

Cada falha de colapso, para ser injetada, necessita apenas de uma regra de injeção, composta por um teste de endereço e uma ação de descarte, como mostrado na Figura 4.

```
teste  pacote.endereço == conexão.endereço
teste  pacote.porta == conexão.porta
teste  conexão.contador >= falha.começo
teste  conexão.contador <= falha.fim
ação   conexão.contador++
teste  rand() <= falha.taxa
ação   injeta.falha(tipo_de_falha)
```

**Figura 3: Regra de Injeção de falhas genérica**

```
teste  pacote.endereço == conexão.endereço
ação   injeta.falha(descarte)
```

**Figura 4: Regra de colapso para particionamento**

Essa regra deve ser adicionada no início do particionamento coordenado e removida ao final do mesmo. Adicionalmente, é necessário instruir a ferramenta ComFIRM a não filtrar pacotes utilizados para a coordenação das falhas, criando uma regra que permita a recepção dos pacotes destinados à porta utilizada pelo módulo de comunicação do injetor com a máquina *home*.

## 5. Considerações finais

Os injetores de falhas são importantes ferramentas de validação da dependabilidade, pois permitem a análise do comportamento das aplicações sem a necessidade de esperar pela ocorrência natural de falhas. Neste artigo, é apresentado um modelo de integração de injetores distintos de falhas de nó em uma arquitetura distribuída, aumentando a interoperabilidade.

O modelo de integração apresentado utiliza um módulo de comunicação e tradução de configuração do cenário de falhas do injetor de nó sem modificação da semântica descrita originalmente pela máquina *home* do experimento. A abordagem apresentada objetiva a manutenção da modularidade da arquitetura de injeção de falhas, se integrando transparentemente a arquitetura de FIONA sem exigir alterações dos injetores integrados.

## Referências


[1] Carreira, J. e Silva, J. G. (1998) Why do some (weird) people inject faults? *ACM SIGSOFT Software Engineering Notes*, 23 (1): 42–43.

[2] Hsueh, M.-C., Tsai, T. K. e Iyer, R. K. (1997). Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82;

[3] Jacques-Silva, G., Drebes, R. J., Gerchman, J. e Weber, T. S. (2004). FIONA: A fault injector for dependability evaluation of Java-based network applications. In *Proc. of the 3$^{rd}$ IEEE Intl. Symposium on Network Computing and Applications*, páginas 303–308, Cambridge, MA.

[4] Leite, F. O. (2000). ComFIRM – injeção de falhas de comunicação através da alteração de recursos do sistema operacional. Dissertação de Mestrado, II/UFRGS, Porto Alegre, Brasil.