Suporte a Injetores de Falhas em um Ambiente para Descrição de Cargas de Falhas

Ruthiano S. Munaretti, Taisy S. Weber

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

{rsmunaretti,taisy}@inf.ufrgs.br

Resumo. *No contexto de sistemas distribuídos complexos, injeção de falhas ainda não é utilizada de forma efetiva como uma técnica de testes. Isto ocorre devido a baixa usabilidade dos injetores existentes, uma vez que cada injetor utiliza abordagens distintas de funcionamento. Assim, este artigo estende o modelo de um ambiente para descrição de carga de falhas, detalhando a realização desta carga no ambiente. Será apresentada a extensão desta arquitetura e a descrição de seus componentes, juntamente com um exemplo de funcionamento.*

1. Motivação

A ocorrência de falhas em sistemas computacionais é inevitável, independentemente dos níveis de confiabilidade que estes sistemas possuem. Por este motivo, a execução de testes é uma tarefa essencial, tendo os injetores de falhas como ferramentas fundamentais para a execução destes testes sob falhas. Injeção de falhas, por sua vez, é uma técnica que provoca falhas de forma controlada em um sistema. Comparada a outras técnicas (como *modelagem analítica*), a injeção de falhas mostra-se mais adequada para a investigação do comportamento sob falhas de sistemas complexos [Arlat et al. 2003].

No contexto de sistemas distribuídos, as falhas mais relevantes são de *comunicação*. Por falhas de comunicação, entende-se todas as falhas que envolvem, no todo ou em parte, a rede de comunicação no qual o sistema em questão é executado. Como exemplos de falhas de comunicação, podem ser destacados o colapso de algum *caminho* ou *nodo* na rede de comunicação, assim como o atraso, descarte e duplicação de mensagens em nodos e caminhos da respectiva rede.

Apesar de inevitável, a probabilidade de ocorrência de falhas em sistemas distribuídos é pequena, se considerarmos a execução real do respectivo sistema. Por este motivo, em um ambiente operacional, a identificação das causas de uma determinada falha torna-se complexa, bem como a reprodução das mesmas, considerando sistemas complexos [Hsueh et al. 1997]. Assim, o uso de injetores de falhas permite acelerar este processo, de forma a verificar se os mecanismos de tolerância a falhas estão corretamente implementados.

Com o intuito de resolver este problema, é possível constatar a existência de inúmeros injetores de falhas no contexto de sistemas distribuídos. Estes injetores utilizam diversas abordagens, com o objetivo de facilitar a tarefa de injeção de falhas. As abordagens, por sua vez, diferem-se principalmente na definição de carga de falhas, ou seja, do conjunto de falhas a serem incluídas em um dado experimento de injeção de falhas.

Esta existência de várias abordagens ocasiona uma queda na usabilidade dos injetores, fazendo com que a injeção de falhas não seja utilizada efetivamente como uma técnica complementar de testes. Para isso, este artigo aborda um ambiente para descrição de cargas de falhas, com foco em falhas de comunicação. Assim, o ambiente fornecerá as cargas de falhas necessárias aos injetores a serem utilizados, baseando-se em uma carga genérica, conforme esquematizado na figura 1.

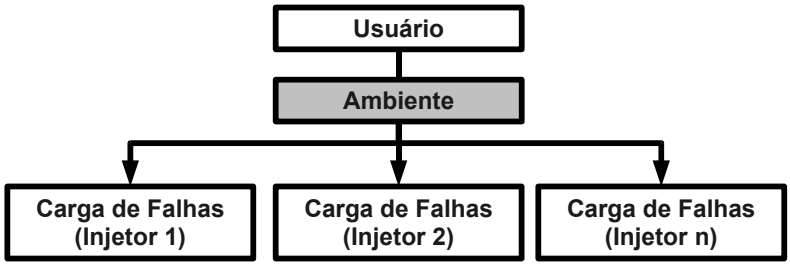

Figura 1. Ambiente para descrição de cargas de falhas.

Considerando o ambiente proposto, um modelo do mesmo já foi previamente definido [Munaretti and Weber 2008]. O presente artigo aborda uma *extensão* deste modelo, com foco na descrição da carga de falhas, visando fornecer suporte aos injetores de falhas a serem utilizados. Desta forma, será detalhada a metodologia empregada pelo ambiente em transformar uma dada carga de falhas (especificada pelo usuário) em uma carga específica de um determinado injetor.

O restante do artigo está organizado da seguinte maneira: a seção 2 apresenta os injetores de falhas que se aproximam ao trabalho proposto. A seção 3 explica a arquitetura do ambiente abordado, visando contextualizar a contribuição deste artigo. A extensão da arquitetura do ambiente, a partir do suporte a injetores de falhas, é realizada na seção 4, seguido de um exemplo de execução na seção 5. Finalmente, conclusões sobre o trabalho realizado, bem como a delimitação dos próximos passos, são apresentados na seção 6.

2. Trabalhos Relacionados

Referente a descrição de carga de falhas, como já mencionado na seção anterior, os injetores de falhas existentes fazem uso de abordagens distintas para tal especificação. A seção atual visa descrever injetores que utilizam abordagens facilitadas para esta descrição, de forma a compará-las com o ambiente proposto. Neste sentido, serão abordados dois injetores do Grupo de Tolerância a Falhas da UFRGS (FIONA e FIRMAMENT, respectivamente), além de três injetores conhecidos na literatura (DOCTOR, Mendosus e FAIL/FCI).

FIONA [Jacques-Silva et al. 2004] é um injetor de falhas em sistemas distribuídos de larga escala, com foco no protocolo UDP. A abordagem utilizada no mesmo consiste na instrumentação de código, utilizando-se para isso de uma ferramenta de instrumentação em Java, denominada JVMTI. O modelo de falhas utilizado neste injetor é o proposto por Birman [Birman 1996], com destaque ao suporte a particionamento de rede. Para a criação de cargas de falhas, FIONA utiliza um *arquivo de configuração*, instanciando-se um objeto para cada falha especificada, o que torna simplificada a especificação das falhas a serem injetadas.

FIRMAMENT [Drebes 2005] é um injetor de falhas que trabalha no nível de sistema operacional, injetando falhas a partir de módulos do kernel Linux. Para a especificação de cargas de falhas, FIRMAMENT utiliza a abordagem de *faultlet*, uma linguagem tipo *assembler* utilizada para emular comportamentos de falhas. Apesar de existir uma certa complexidade na aprendizagem da ferramenta (devido à linguagem adotada pelos *faultlets*), esta abordagem proporciona um grande poder de expressividade à ferramenta.

DOCTOR [Han et al. 1995] consiste em um ambiente de injeção de falhas, com foco em sistemas distribuídos de tempo real. Quanto ao modelo de falhas adotado, DOCTOR é capaz de emular falhas de hardware tradicionais, bem como falhas de comunicação. Referente aos quesitos de carga de falhas e extensibilidade, DOCTOR apresenta um mecanismo adequado para a elaboração de carga de falhas, assim como pontos de extensão bem definidos. Entretanto, a extensibilidade em si é limitada, devido ao foco específico do injetor (em sistemas de tempo real), além da complexidade inerente ao mesmo.

O injetor Mendosus [Li et al. 2002] trata de uma ferramenta para injeção de falhas em redes do tipo *SAN* (System-Area Networks). Assim como FIRMAMENT, este injetor aplica as falhas a nível de kernel, através da extensão de classes do kernel Linux. Diferentemente de FIRMAMENT, a ferramenta Mendosus utiliza chamadas de alto nível para a especificação de carga de falhas. Entretanto, o modelo de falhas adotado por Mendosus é limitado, admitindo apenas falhas relacionadas a colapso.

A ferramenta FAIL/FCI [Hoarau and Tixeuil 2005] é utilizada para injeção de falhas em aplicações de *cluster* e *peer-to-peer* (P2P). Por utilizar uma abordagem de *linguagem de autômatos* para a especificação de carga de falhas, esta ferramenta se apresenta como a mais próxima à abordagem proposta neste artigo. Entretanto, assim como o injetor Mendosus, esta ferramenta possui um modelo de falhas limitado apenas às falhas de colapso.

3. Arquitetura Geral do Ambiente

Um ambiente de cargas de falhas deve refletir todos os casos possíveis de falhas, de acordo com o tipo de aplicação alvo desejado. Neste sentido, o modelo do ambiente foi elaborado visando a divisão em elementos, de forma a modularizar as funcionalidades e promover a extensibilidade. A figura 2 ilustra uma visão macro da arquitetura do ambiente, sendo cada elemento descrito nos itens a seguir.

- **Usuário do Ambiente:** elemento externo ao ambiente, o usuário é o responsável pelo gerenciamento do ambiente como um todo. Assim, cabe ao usuário especificar uma *Carga de Falhas*, bem como uma *Composição de Falhas* (caso seja necessária ao experimento) e o *Injetor Específico* a ser utilizado, juntamente com sua respectiva *Interface*.
- **Carga de Falhas:** representando uma *entrada de dados* ao ambiente pelo usuário, a carga de falhas visa especificar *quais* falhas estarão ativas em um dado experimento, bem como qual a *configuração* de cada falha no respectivo experimento. Esta carga de falhas é genérica, não sendo assim restrita às peculiaridades de uma determinada ferramenta.

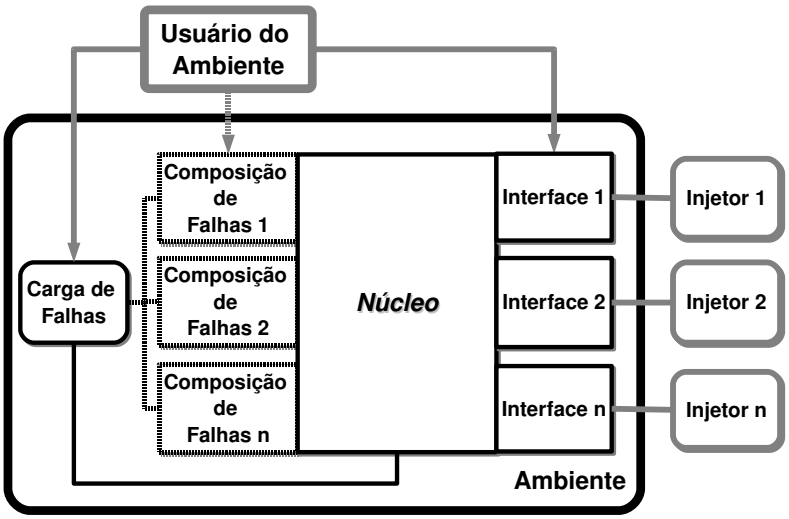

Figura 2. Arquitetura do Ambiente.

- **Composição de Falhas:** define um conjunto de falhas, composto por uma ou mais falhas existentes no elemento *Núcleo* (descrito no próximo item). Assim, este elemento possui o intuito de permitir a criação de novos tipos de falhas, com a finalidade de facilitar a descrição da carga de falhas pelo usuário. Vale ressaltar que, ao usuário do ambiente, a utilização deste elemento é *opcional*, como indicam as linhas pontilhadas na figura 2.
- **Núcleo:** sendo o elemento de mais baixo nível e, conseqüentemente, o mais importante do ambiente, o *núcleo* define uma série de *falhas primitivas* existentes em sistemas baseados em troca de mensagens. Logo, o principal objetivo consiste em mapear a carga de falhas, definida pelo usuário, às falhas primitivas existentes, iniciando-se assim a transformação da carga de falhas genérica para a carga de falhas específica de cada injetor utilizado.
- **Interface:** elemento externo ao núcleo, a interface tem por objetivo *finalizar* a transformação da carga de falhas genérica para a carga de falhas específica do injetor utilizado. Para isso, o ambiente define a interface como uma outra entrada de dados definida pelo usuário e, assim como a composição de falhas, é fornecida uma forma *intuitiva* para que o usuário especifique a *forma* pelo qual o seu injetor específico trabalha.
- **Injetor:** finalmente, o elemento *injetor* representa um injetor de falhas específico que o usuário deseja utilizar no ambiente. O ambiente permite a utilização de vários injetores simultaneamente, sendo que os mesmos recebem como entrada suas respectivas cargas de falhas específicas, geradas pelas interfaces descritas no item anterior.

Desta forma, pode-se observar que o ambiente possui três entradas de dados, correspondentes a *Carga de Falhas*, a *Composição de Falhas* e à *Interface*. Referente a estes dois últimos, percebe-se que os mesmos são integrados ao *núcleo* do ambiente, sendo descritos pelo usuário utilizando a abordagem de *plugins*. Assim, o ambiente torna-se modularizado e extensível, facilitando o seu uso.

4. Extensão do Ambiente

Conforme o modelo descrito na seção anterior, é possível constatar que a descrição de carga de falhas e o suporte a injetores são funcionalidades essenciais ao ambiente proposto. Por este motivo, as mesmas são descritas em detalhes nesta seção. Neste sentido, o modelo existente passou por uma extensão, através de um detalhamento que envolveram os elementos *Núcleo* e *Interface*, como esquematizado na figura 3.

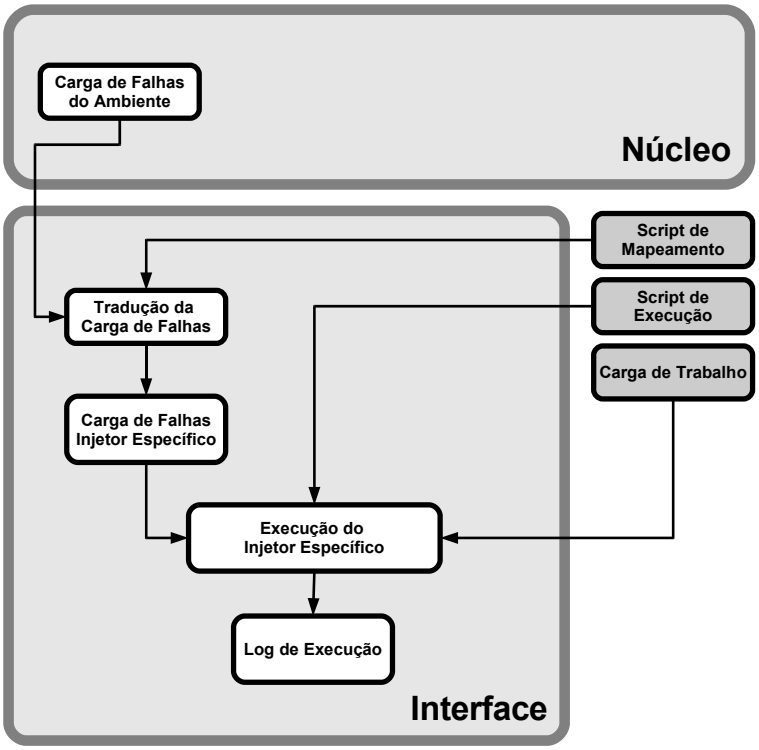

Figura 3. Arquitetura da Interface para Injetores Específicos.

De forma a facilitar a explicação desta abordagem, a seção atual foi dividida em três partes. A seção 4.1 aborda como a *carga de falhas* é especificada e utilizada, enquanto que a seção 4.2 indica como o *suporte a injetores de falhas* é realizado no contexto do ambiente. Finalmente, a seção 4.3 descreve o *resultado* desta extensão, indicando o que pode ser obtido a partir de uma execução deste modelo.

4.1. Carga de Falhas

No contexto do ambiente, a descrição da carga de falhas é realizada a partir de elementos externos, especificados pelo usuário do ambiente. Considerando a extensão do ambiente, fazem parte desta descrição os componentes **Carga de Falhas do Ambiente**, **Script de Mapeamento**, **Script de Execução** e **Carga de Trabalho**, descritos a seguir.

- *Carga de Falhas do Ambiente:* corresponde às falhas a serem injetadas em um dado experimento. Para isso, as respectivas falhas são especificadas através de um *script*. A estrutura deste script, previamente definida no modelo do ambiente, é ilustrada na figura 4, sendo formada pelo **rótulo** da falha a ser injetada, juntamente com as configurações da respectiva falha (caso as mesmas sejam necessárias).

Figura 4. Estrutura utilizada no script para carga de falhas

- *Script de Mapeamento:* componente responsável pelo *relacionamento* entre os comandos do ambiente e os comandos do injetor. Este script é formado pelo rótulo de uma falha, acompanhado do comando corresponde a um injetor, conforme a figura 5. Assim como a carga de falhas, a estrutura deste script foi definida no modelo do ambiente. Nesta extensão, o mapeamento foi dividido em dois tipos:
  - **Gerais:** neste tipo de script, todos os comandos do injetor são mapeados para comandos do ambiente. Assim, pode-se cobrir todas as funcionalidades do injetor no ambiente. Entretanto, a efetividade do script é proporcional ao número de comandos mapeados, tornando esta etapa complexa.
  - **Específicos:** em um mapeamento específico, *somente* os comandos utilizados no dado experimento são mapeados para comandos do ambiente. Em comparação ao mapeamento geral, a construção deste script torna-se mais simplificada. No entanto, várias readaptações podem ser necessárias, de acordo com a natureza do experimento.

Figura 5. Script de mapeamento - falha do ambiente : especificação do injetor

- *Script de Execução:* correspondem aos comandos e parâmetros necessários à execução de um injetor específico. Como esta especificação é utilizada diretamente na execução do injetor (conforme ilustrado na figura 3), a mesma não possui uma estrutura definida. Logo, o script é informado ao ambiente de forma direta.
- *Carga de Trabalho:* componente que implementa a carga de trabalho do injetor de falhas utilizado. Assim como o script de execução, este componente também é utilizado de forma direta pelo injetor específico, para fins de coleta de dados, como geração de *logs* (descrito na próxima seção).

4.2. Suporte a Injetores de Falhas

No contexto do modelo estendido, o suporte a injetores de falhas permite viabilizar a execução do experimento de injeção de falhas em um determinado injetor. Neste sentido, são envolvidos os seguintes componentes: **Tradução da Carga de Falhas**, **Execução do Injetor Específico** e **Geração de Log de Execução**, apresentados a seguir.

- *Tradução da Carga de Falhas:* responsável pela conversão da carga de falhas do ambiente em uma carga de falhas específica do injetor utilizado. Para isso, é utilizado o *script de mapeamento* descrito na seção anterior.

- *Execução do Injetor Específico:* consiste na realização do experimento propriamente dito, a partir da carga específica gerada no item anterior, utilizando também o **Script de Execução** e a **Carga de Trabalho** passados como entrada (mencionados anteriormente).
- *Geração de Log de Execução:* efetua uma coleta de dados, referente ao experimento realizado no injetor, considerando possíveis erros/problemas na execução.

4.3. Resultados

A partir da carga de falhas e do suporte aos injetores, o modelo estendido realiza a geração de dados, referentes a *saídas*. Estas saídas, por sua vez, correspondem aos elementos **Carga de Falhas do Injetor Específico** e **Log de Execução**, ambos descritos a seguir.

- **Carga de Falhas do Injetor Específico:** principal saída do ambiente, corresponde a carga de falhas real, a ser inserida no respectivo injetor específico para execução e posterior análise de resultados.
- **Log de Execução:** permite visualizar os resultados obtidos na injeção realizada, com o intuito de gerar dados estatísticos, obter indicadores de uso, comparações de execução/desempenho, entre outros itens relevantes.

5. Execução do Ambiente

A partir da arquitetura estendida, a execução do ambiente modificado pode ser realizada a partir de entradas simples, tendo assim um baixo nível de complexidade. Neste sentido, o principal objetivo está voltado a usabilidade da interface, principal premissa do ambiente.

```
stop()
dropPackets(0.1)
```

Figura 6. Exemplo de carga de falhas genérica.

Para efeito de exemplo, nas figuras 6 e 7 são ilustradas a execução de uma falha de *colapso*, seguida de uma falha de *omissão* com descarte de 10% dos pacotes. Para isso, são necessárias apenas a criação de duas entradas: a **carga de falhas** e o **mapeamento de falhas**. Referente ao mapeamento de falhas, vale ressaltar a realização de um mapeamento para cada injetor específico - no contexto deste exemplo, para os injetores FIONA (figura 7(a)) e FAIL/FCI (figura 7(b)).

```
stop() : UdpCrashFault
dropPackets() : UdpOmissionFault
```

(a) FIONA

```
stop() : halt
dropPackets() : stop, continue
```

(b) FAIL/FCI

Figura 7. Exemplos de scripts de mapeamento.

6. Conclusões

Este artigo apresentou uma extensão ao modelo de um ambiente para descrição de cargas detalhadas de falhas, com foco ao suporte a injetores de falhas. Neste contexto, foi abordada a arquitetura desta extensão, envolvendo seus principais componentes. Em seguida, foi mostrado um exemplo de execução deste modelo estendido, abrangendo possíveis dados de entrada para o mesmo.

Comparando-se às abordagens similares, o ambiente implementa um modelo de falhas mais detalhado que os suportados por Mendosus e FAIL/FCI. Outra característica presente no ambiente diz respeito a sua independência de protocolo/aplicação, diferentemente de injetores como FIONA (que, por exemplo, possui o foco voltado ao protocolo UDP). Finalmente, o ambiente suporta injetores de diversas finalidades, além de possuir um *script* simplificado para definição das cargas de falhas, características não presentes nos injetores DOCTOR e FIRMAMENT.

Em relação à interface do ambiente, constata-se que a mesma apresenta uma boa usabilidade, se comparada aos mecanismos existentes nos injetores. Isto pode ser observado nos scripts adotados, que necessitam de pouca ou nenhuma manutenção na realização de experimentos - apenas o script referente a carga de falhas precisa ser modificado com freqüência. Neste sentido, o ambiente proposto torna-se adequado para a execução de vários injetores de falhas, envolvendo o teste de diferentes protocolos, uma vez que tal abordagem possibilita uma comparação direta entre estes injetores.

Atualmente, este trabalho encontra-se em fase de execução, com a implementação de um protótipo em andamento. Como próximos passos, estão previstos um detalhamento maior dos outros elementos do modelo do ambiente (tais como a *Carga/Composição de Falhas* e o *Núcleo*), além do término do protótipo e a execução de experimentos.

Referências

Arlat, J., Crouzet, Y., Karlsson, J., Folkesson, P., Fuchs, E., and Leber, G. H. (2003). Comparison of Physical and Software-Implemented Fault Injection Techniques. *IEEE Transactions on Computers*, 52:1115–1133.

Birman, K. (1996). *Building Secure and Reliable Network Applications*. Manning Publications, Co, Greenwich.

Drebes, R. J. (2005). FIRMAMENT: Um Módulo de Injeção de Falhas de Comunicação para Linux. Master's thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Han, S., Shin, K., and Rosenberg, H. (1995). *DOCTOR: An Integrated Software Fault Injection Environment for Distributed Real-Time Systems*. In *Int. Computer Performance and Dependability Symposium. (IPDS'95)*, pages 204–213, Erlangen, Germany. IEEE Computer Society Press.

Hoarau, W. and Tixeuil, S. (2005). *A Language-Driven Tool for Fault Injection in Distributed Systems*. In *Proc. of the 6th IEEE/ACM Intl. Workshop on Grid Computing*, pages 194–201, Grand Large, França.

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

Jacques-Silva, G., Drebes, R., Gerchman, J., and Weber, T. (2004). *FIONA: A Fault Injector for Dependability Evaluation of Java-Based Networks*. In *Proc. of the 3rd IEEE Intl. Symposium on Network Computing and Applications*, pages 303–308, Cambridge, MA.

Li, X., Martin, R., Nagaraja, K., Nguyen, T. D., and Zhang, B. (2002). Mendosus: A SAN-Based Fault-Injection Test-Bed for the Construction of Highly Available Network Services. In *In Proceedings of the 1st Workshop on Novel Uses of System Area Networks (SAN-1)*.

Munaretti, R. S. and Weber, T. S. (2008). *Modelo de um Ambiente para Descrição de Cenários Detalhados de Falhas*. In SBC, editor, *IX Workshop de Testes e Tolerância a Falhas - WTF2008*, pages 71–84, Rio de Janeiro, RJ.