Distribuições de Probabilidade na Descrição de Carga de Falhas para Injetores de Falhas de Comunicação *

Juliano C. Vacaro, Gabriela Jacques-Silva, Taisy Silva Weber, Ingrid Jansch-Pôrto

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

{jcvacaro,gjsilva,taisy,ingrid}@inf.ufrgs.br

Abstract. *Fault injection is an efficient technique for evaluating systems and fault tolerance mechanisms. In order to obtain accurate results from fault injection based experiments, it is important to reproduce real conditions in the test environment. The aim of this paper is to extend FIONA to provide the use of probabilistic distributions and obtain faultloads that can simulate the behavior observed in real system environments.*

Resumo. *Injeção de falhas é uma técnica eficiente na validação do funcionamento de sistemas computacionais, bem como dos mecanismos de tolerância a falhas. Para que os resultados de experimentos baseados nesta técnica sejam válidos, é importante que o ambiente de teste reproduza as condições observadas na realidade. Este artigo tem por objetivo expandir a ferramenta de injeção de falhas FIONA para prover o uso de distribuições de probabilidade e assim obter cenários de falhas capazes de simular o comportamento observado na realidade.*

1. Introdução

Aplicações devem ser projetadas para operar em ambientes onde falhas podem ocorrer. Mecanismos de tolerância a falhas são a melhor garantia para prover dependabilidade (habilidade de prover serviços que possam ser justificadamente confiáveis [Avizienis et al. 2004]) a estes sistemas. A validação de tais aplicações não é tarefa trivial. Essa tarefa se torna ainda mais crítica em sistemas distribuídos onde, além do comportamento sob falhas de cada nó do sistema, também a sua reação a falhas no subsistema de troca de mensagens deve ser determinada. Deste modo, procedimentos convencionais de teste de cada nó isolado não são suficientes para garantir propriedades como confiabilidade e disponibilidade de todo o sistema ou estabelecer medidas como cobertura de falhas e queda de desempenho sob falhas.

Ferramentas de injeção de falhas mostram-se um método eficiente para a validação de sistemas. A abordagem visa a emulação de falhas e a análise do comportamento do sistema na presença destas falhas. FIONA é uma ferramenta de injeção de falhas de comunicação que suporta a criação de cenários de falhas para condução de experimentos. A manifestação de falhas pode ser definida baseada em distribuição uniforme. Porém, em ambientes de rede, nem todas as aplicações seguem este padrão de manifestação de erros

*Financiado pelos projetos ACERTE/CNPq (472084/2003-8) e DepGriFE/HP P&D Brasil

de comunicação [Yoma et al. 2005]. O uso de modelos probabilísticos capazes de reproduzir tais características é relevante para a criação de cenários de falhas mais significativos na condução de experimentos.

Este artigo tem por objetivo descrever a expansão da ferramenta FIONA para possibilitar a injeção de falhas baseadas em distribuições de probabilidade, proporcionando maior flexibilidade e generalidade na elaboração de cenários de falhas. Esta expansão permite que experimentos sejam conduzidos de modo a refletir a ocorrência de falhas observadas em ambientes reais.

A Seção 2. mostra as principais características dos injetores de falhas. A Seção 3. descreve a arquitetura da ferramenta FIONA. Detalhes da implementação para a realização do trabalho proposto são mostrados na Seção 4. Considerações finais são feitas na Seção 5.

2. Injetores de Falhas

Injeção de falhas é uma técnica eficiente para o teste de estratégias de tolerância a falhas implementadas em sistemas. Nesta técnica, falhas são inseridas no ambiente sob teste e o comportamento do sistema alvo na presença de falhas é observado. As técnicas de injeção de falhas são classificadas em injeção de falhas por simulação, por *hardware* e por *software* [Hsueh et al. 1997]. Na injeção por *software* falhas são inseridas no ambiente de execução das aplicações em teste. Esta abordagem tem como característica a flexibilidade e o baixo custo, porém, o nível de perturbação gerado ao ambiente de teste deve ser levado em consideração.

Na próxima Seção será descrita a ferramenta FIONA, que utiliza a técnica de injeção de falhas por *software* para inserir falhas de comunicação em sistemas distribuídos.

3. FIONA - *Fault Injector Oriented to Network Applications*

FIONA (*Fault Injector Oriented to Network Applications*) é uma ferramenta baseada em JVMTI desenvolvida para a condução de experimentos de injeção de falhas de comunicação em aplicações Java. FIONA pode ser usado para a verificação do funcionamento correto de mecanismos de tolerância a falhas em aplicações Java distribuídas de maneira fácil e flexível [Jacques-Silva et al. 2004].

A ferramenta FIONA permite a execução em ambientes locais e distribuídos. Neste trabalho será focada a arquitetura local de FIONA, devido a facilidade na condução de experimentos para a validação da implementação proposta. É importante salientar que as alterações realizadas na ferramenta são válidas tanto no ambiente de operação local quanto no ambiente de operação distribuído.

Nas próximas Seções são descritas as características da ferramenta FIONA. Na Seção 3.1. é mostrado o modelo de falhas adotado pela ferramenta. A arquitetura local de FIONA é mostrada na Seção 3.2. A Seção 3.3. descreve o mecanismo de construção de cenários de falhas.

3.1. Modelo de Falhas

O modelo de falhas para sistemas distribuídos assumido em FIONA é baseado no definido por Birman [Birman 1996], que descreve falhas de colapso (*halt*), parada segura (*fail-*

stop), omissão na transmissão e na recepção, rede, particionamento de rede, temporização e bizantinas. Como o foco da ferramenta é o teste de sistemas distribuídos de larga escala, foram consideradas as falhas mais comuns nesse ambiente. FIONA emula falhas de colapso, de omissão, de temporização, de duplicação e de particionamento de redes.

FIONA injeta falhas em aplicações que usam o protocolo UDP. Apesar desse protocolo ser não confiável, é comumente usado como base para implementação de protocolos de nível mais alto usados em aplicações com características de tolerância a falhas. Como exemplo pode-se citar o *middleware* de comunicação em grupo JGroups [Ban 1998], que por padrão usa UDP como base para sua pilha de protocolos.

3.2. Arquitetura Local

FIONA realiza a injeção de falhas através da instrumentação das classes de comunicação do sistema. Falhas em comunicação usando o protocolo UDP são ativadas através da instrumentação da classe `java.net.DatagramSocket`, que contém os métodos `send()` e `receive()` para envio e recebimento de datagramas. O injetor é composto de três partes: o agente JVMTI, as classes de sistema instrumentadas e as classes auxiliares. O agente JVMTI realiza a instrumentação das classes de sistema e mantém a comunicação com os demais injetores da rede; as classes instrumentadas realizam a injeção das falhas; as classes auxiliares armazenam a configuração do experimento e os logs de monitoramento. A Figura 1 ilustra a arquitetura local do injetor.

Assim, toda a vez em que os métodos `send()` e `receive()` são invocados, a classe instrumentada `java.net.DatagramSocket` acessa as funcionalidades das classes auxiliares de injeção de falhas para decidir quais falhas devem ser aplicadas à mensagem sendo enviada ou recebida.

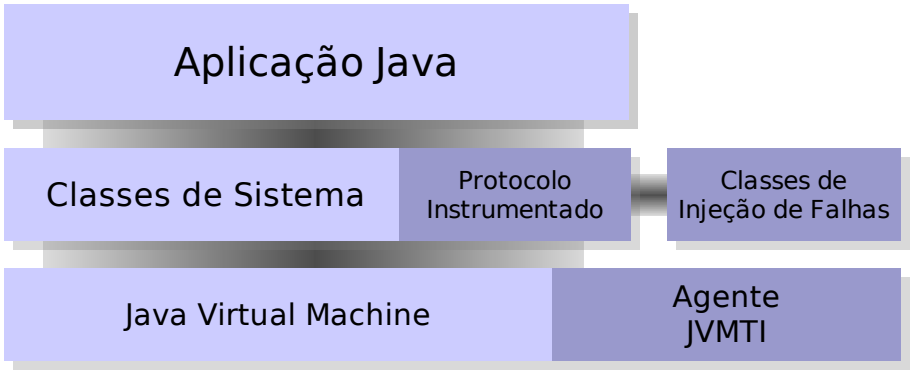

Figura 1. Arquitetura local do injetor FIONA

3.3. Cenário de Falhas

FIONA ainda permite que sejam definidos cenários de falhas. Através destes cenários é possível definir a carga de falhas que será injetada durante a condução dos experimentos. Ao final da execução de um experimento a ferramenta gera informações sobre as falhas injetadas, as quais podem ser visualizadas nos *logs* disponibilizados pelo programa. Estes cenários são descritos através de arquivos de configuração, os quais são analisados pelas classes de injeção de falhas responsáveis pela configuração da ferramenta.

Para cada falha do cenário de falhas pode ser especificada a maneira com que esta se manifestará. A ferramenta suporta três modos para a manifestação de falhas: permanentes (onde falhas ocorrem em todas as mensagens interceptadas), transientes (onde falhas ocorrem apenas uma vez) e intermitentes (falhas se manifestam de acordo com uma taxa de falhas). Neste último modo, FIONA foi implementada permitindo apenas uma distribuição uniforme de falhas no tempo.

Na Seção 4.2. é detalhado o fluxo de execução no processo de inicialização de cenários de falhas e as modificações efetuadas sobre os arquivos de configuração para a adição de distribuições de probabilidade na definição destes cenários.

4. Expansão da Ferramenta FIONA

Na Seção 2. foi mostrada a importância do mecanismo de injeção de falhas. A condução de experimentos baseada nesta abordagem visa reproduzir falhas reais em ambientes controlados. Para que os resultados destes experimentos sejam significativos, é proposto o uso de distribuições de probabilidade. Distribuições modelam comportamentos em determinadas situações. Por exemplo, Yoma [Yoma et al. 2005] propõe o uso de um modelo de Gilbert com restrições de duração de tempo baseado em distribuições gamma e geométrica para descrever a perda de pacotes UDP em redes utilizadas para a transmissão de dados de aplicações de tempo real. O uso de distribuições também permite que o comportamento das falhas injetadas seja variado de maneira controlada, pois, cada distribuição de probabilidade possui características bem definidas. Ainda é possível especificar o comportamento de uma distribuição através de informações como média e desvio padrão, fato que aumenta o controle do processo de condução de experimentos.

Nesta Seção são descritos os procedimentos adotados para adicionar a possibilidade de FIONA trabalhar com o uso de distribuições de probabilidade.

4.1. DESMO-J - *Discrete-Event Simulation and MOdelling in Java*

DESMO-J (*Discrete-Event Simulation and MOdelling in Java*) [Page 1999], é um *framework* de simulação de sistemas desenvolvido na Universidade de Hamburg. DESMO-J é totalmente desenvolvido na plataforma Java e é distribuído sobre a Licença GPL. A biblioteca suporta simulação discreta de sistemas baseada em eventos, e também possui funcionalidades de probabilidade e estatística. No escopo deste trabalho, as funcionalidades utilizadas do *framework* estão localizadas no pacote `desmoj.core.dist`. Este pacote contém todo o suporte da biblioteca às funções de probabilidade e, em especial, aos geradores de números randômicos.

Dentre os geradores de números randômicos disponíveis neste *framework* estão os geradores baseados em distribuições Bernoulli, Poisson, Normal, Exponencial etc. A utilização dos geradores citados, no ambiente FIONA, é mostrado na Seção 4.2.

4.2. Implementação

A primeira decisão a ser tomada é como adicionar as novas funcionalidades à ferramenta. Para garantir que não haja dependência de FIONA em relação ao *framework* DESMO-J, foi desenvolvida uma camada de abstração que possibilita o uso de um gerador de números randômicos genérico. Assim, as classes da biblioteca DESMO-J foram inseridas sobre esta camada de abstração, possibilitando aos demais módulos de FIONA usar os

recursos oferecidos sem estabelecer dependências. A Figura 2 é um diagrama de classes que ilustra de maneira simplificada a organização dos módulos de injeção de falhas da ferramenta.

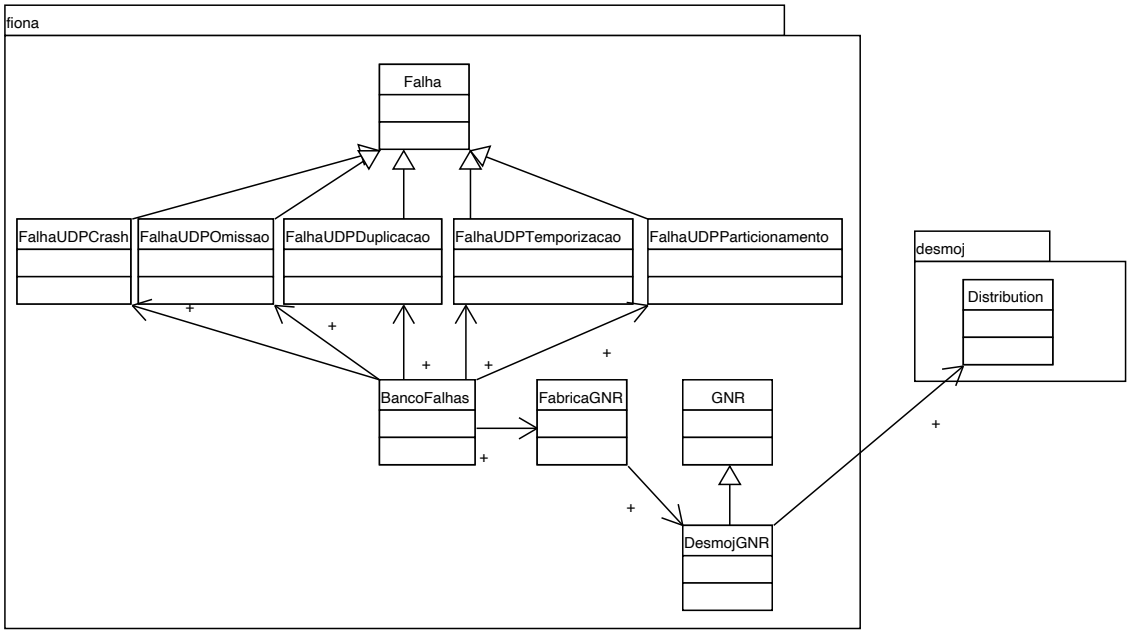

Figura 2. Diagrama simplificado das classes de injeção de falhas de FIONA

Como mencionado na Seção 3.3., FIONA permite a definição de cenários de falhas para a condução de experimentos via arquivo de configuração. Assim, as distribuições de probabilidade podem ser informadas neste arquivo. A Figura 3 mostra o formato de um arquivo de configuração para falhas de omissão de mensagens. O campo `padrão_de_repetição` indica como a falha irá se repetir durante o experimento, podendo ser permanente, transiente ou intermitente. Os campos `distribuição`, `semente` e `taxa_de_falhas` definem a frequência com que falhas serão manifestadas quando o modo de repetição for intermitente. O próximo parâmetro é referente ao tipo de ativação da falha (`tipo_ativação(tempo|mensagem)`). Uma falha pode se tornar ativa por um tempo definido em segundos ou pelo número da mensagem que está sendo trocada por um determinado *socket*. Os campos de `início` e `fim` de falha definem o intervalo da execução da aplicação que a falha estará ativa. A parte final da configuração indica em qual *socket* será injetada a falha.

```
UdpOmissionFault:<padrão_de_repetição>:[<distribuição>:<semente>:]
<taxa_de_falhas>:<tipo_ativação(tempo|mensagem)>:<início>:<fim>:
<nodo_origem>:<porta_origem>:<nodo_destino>:<porta_destino>:
<envio|recepção>:
```

Figura 3. Configuração para falhas de omissão

Na inicialização de FIONA uma das primeiras tarefas executadas é a leitura das informações mencionadas. A classe `BancoFalhas` é a responsável por esta tarefa. Durante o processamento do arquivo o módulo solicita um gerador de números randômicos à `FabricaGNR`, caso indicado no arquivo de configuração. A classe `FabricaGNR`

então inicializa o gerador especificado e retorna o objeto para o `BancoFalhas`, o qual configura a falha em questão com este gerador.

O processo de criação de um gerador é transparente para a classe `BancoFalhas`. A tarefa de criação é feita pela classe `FabricaGNR` que acessa as bibliotecas de geração de números aleatórios como DESMO-J. Esta abstração permite que bibliotecas adicionais sejam incorporadas ao programa sem que alterações sejam propagadas para os demais módulos. Cada gerador deve implementar a interface `GNR`, meio pelo qual geradores são manipulados pelos demais componentes do sistema. Portanto, quando mensagens são enviadas ou recebidas, as classes instrumentadas de FIONA acessam a classe `BancoFalhas` com o objetivo de verificar quais falhas devem ser acionadas para aquela mensagem. Caso a falha for do tipo intermitente, esta irá invocar a sua referência à `GNR` e assim obter valores randômicos da distribuição escolhida para determinar a ativação ou não desta falha.

5. Conclusão

Esforços visando a melhoria das técnicas de teste e validação de sistemas mostram-se válidos na medida em que a necessidade por sistemas confiáveis é exigida. Com o uso de distribuições de probabilidade para definir o comportamento do processo de injeção de falhas, é possível elaborar cenários de falhas capazes de representar a ocorrência de falhas observadas em ambientes reais.

A implementação da arquitetura para o uso de distribuições de probabilidade foi finalizada, portanto, a próxima etapa deste trabalho é a validação do mecanismo proposto. Testes serão realizados a fim de verificar o comportamento de aplicações na presença de falhas injetadas sobre diferentes distribuições de probabilidade. Também é importante analisar em quais situações uma certa distribuição de probabilidade mostra-se mais adequada para modelar a manifestação de falhas naquele ambiente.

Referências

Avizienis, A., Laprie, J.-C., Randell, B., and Landwehr, C. E. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Ban, B. (1998). JavaGroups - group communication patterns in Java. Technical report, Department of Computer Science, Cornell University.

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

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

Jacques-Silva, G., Drebes, R. J., Gerchman, J., and Weber, T. S. (2004). FIONA: A fault injector for dependability evaluation of java-based network applications. *3rd IEEE Intl. Symposium on Network Computing and Applications*, pages 303–308.

Page, B. (1999). DESMO-J a framework for discrete-event modelling and simulation. *http://www.desmoj.de*.

Yoma, N. B., Busso, C., and Soto, I. (2005). Packet-loss modelling in ip networks with state-duration constraints. *IEE Proceedings Communications*, 152(1):1–5.