Modelagem de Falhas para Simulação de Sistemas Distribuídos*

Ruthiano S. Munaretti **Marinho P. Barcellos**

[1]PIPCA - Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Unisinos - Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 - São Leopoldo, RS - 93022-000

{ruthiano,marinho}@exatas.unisinos.br

Resumo. *É reconhecida a dificuldade de se testar experimentalmente sistemas distribuídos tolerantes a falhas. Simulação fornece um ambiente controlado para investigação e testes perante situações atípicas nestes sistemas. O presente artigo aborda uma extensão do framework Simmcast de forma a permitir a simulação de sistemas distribuídos tolerantes a falhas. Para tal, define um modelo de falhas suportado, e como as mesmas são mapeadas nos componentes fundamentais do Simmcast, nodo e caminho. Por fim, introduz o conceito de cenários de falhas: combinações pré-concebidas de comportamentos de falhas e objetos que facilitam a especificação de falhas a serem simuladas durante um experimento.*

1. Introdução

É reconhecida a dificuldade de se testar experimentalmente sistemas distribuídos tolerantes a falhas, particularmente em sistemas de maior porte na Internet. Simulação, nesse caso, pode ajudar a remediar o problema, pois representa um passo intermediário antes da implementação completa e utilização do sistema. Simulação fornece um ambiente controlado e facilita a análise da interação entre os componentes, permitindo testar cenários difíceis de serem reproduzidos.

O Simmcast ([1]) é um *framework* para simulação discreta de protocolos e sistemas distribuídos. Sua arquitetura é adequada à simulação de sistemas distribuídos, no entanto presentemente não existe suporte algum a experimentos com falhas. Almeja-se estender o Simmcast de forma a permitir a execução de experimentos com sistemas distribuídos tolerantes a falhas, tal como modelagem de protocolos de consenso e detectores de defeitos. A versão estendida do Simmcast, denominada Simmcast-FT, permitirá criar ambientes simulados onde falhas ocorram nos componentes fundamentais de uma topologia em momentos específicos e/ou segundo condições especificadas no arquivo que descreve a simulação. Este artigo representa o primeiro passo nesse sentido, descrevendo o projeto da arquitetura do sistema, sua API e as justificativas para as principais decisões de projeto.

O artigo está organizado da seguinte forma: a Seção 2 indica quesitos necessários para a simulação de sistemas distribuídos tolerantes a falhas e revisa trabalhos relacionados. A Seção 3 mostra a arquitetura do *framework* de simulação Simmcast através de uma ótica hierárquica, em camadas. A Seção 4 discute quais são os componentes do sistema subjacente sujeitos a falhas, e como as mesmas podem ser modeladas. A Seção 5 encerra o trabalho e faz considerações sobre trabalhos futuros.

*Este trabalho foi desenvolvido com apoio do CNPq.

2. Simulação de Falhas em Sistemas Distribuídos

Simulação discreta é uma prática popular no dimensionamento de redes de computadores, e tem sido amplamente usada no projeto de novos protocolos de comunicação ou avaliação de desempenho de protocolos existentes em novos cenários de rede. Exemplos de simuladores de redes são OPNET ([2]), VINT NS-2 ([3]), SSF ([4]) e Simmcast ([1]). Simuladores oferecem um ambiente controlado para validar o comportamento de protocolos existentes, fornecem infra-estrutura para desenvolvimento de novos protocolos e oportunizam o estudo de suas interações. É natural que esta técnica seja empregada também no desenvolvimento de protocolos e sistemas distribuídos tolerantes a falhas; exemplos incluem [5], [6] e [7]. Este último oferece um ambiente orientado a objetos para teste de implementações de protocolos distribuídos tolerantes a falhas. Seu foco é em sistemas de tempo-real, cujo processo de testes em ambientes reais é especialmente complexo.

O NS-2 é o simulador mais popular no ambiente acadêmico nos dias de hoje, sendo bastante usado para investigação de novos protocolos de comunicação e seus mecanismos de controle. NS-2 e Simmcast possuem diversos aspectos em comum. Particularmente, ambos envolvem a construção de novos componentes e descrição de um cenário de simulação, conforme a seguir. Primeiro, um simulador pode ser estendido criando-se novos componentes, modificando-se ou aumentando-se os existentes, resultando em novos protocolos, geradores de carga e tecnologias de transmissão. Isto é feito de forma a refletir as características e funcionamento desejados do experimento realizado. Segundo, a descrição de um cenário de simulação consiste na indicação dos diversos componentes que serão utilizados no experimento, bem como da ligação e interação entre eles. Isto é realizado através de uma linguagem de configuração específica, usualmente do tipo *script*.

Na simulação de sistemas distribuídos tolerantes a falhas, são especificadas pelo usuário quais tipos de falhas ocorrerão no ambiente e sob que circunstâncias. Portanto, além de uma "categoria de falha", deve ser indicada uma condição que define o momento em que o componente falhará, quando uma falha é ativada, e o momento que a falha cessará. Vislumbra-se casos em que falhas são ativadas e/ou desativadas de forma *determinística*, segundo uma situação específica, e *probabilísticas*, quando determinados componentes do sistema subjacente tem a si associadas propriedades de falha. Exemplos são, respectivamente, quando um nodo falha estando os demais nodos em um determinado estado, e quando um determinado nodo é configurado com tempo médio entre falhas e tempo médio de recuperação. Objetiva-se um modelo de especificação de falhas conciso, mas que seja flexível e poderoso.

3. Arquitetura do Simmcast

Esta seção descreve a arquitetura do Simmcast, mostrando suas camadas, e enfatizando os conceitos que servem de base para a extensão do *framework* com suporte à simulação de falhas; não é objetivo da mesma descrever o Simmcast como um todo (vide [1, 8, 9]). O Simmcast é um *framework* de simulação cuja construção pode ser descrita através de um conjunto de camadas, onde cada camada é desenvolvida de forma independente das superiores, fazendo o uso de serviços oferecidos pelas inferiores. A arquitetura, um histórico e análise das principais decisões de projeto podem ser encontrados em [9].

Na arquitetura, a **camada 0 (Java)** oferece o suporte a *threads*, unidade básica de execução do simulador, enquanto a **camada 1 (Simmcast Engine)** é o motor de execução do simulador, uma máquina de eventos discretos baseada em processos. Estes processos são uma especialização das *threads* Java. A **camada 2 (kernel do simulador)** é a primeira camada visível ao usuário do simulador, sendo na mesma definidos os diversos componentes que formam o *framework*. Nodos (`Node`) e caminhos (`Path`) são definidos

nesta camada, sendo que a execução de um nodo é definida através de um conjunto de processos definidos na camada 1. A **camada 3 (roteamento)** define componentes como roteadores (`RouterNodes`) e estações (`HostNodes`), bem como algoritmos de roteamento necessários em simulações de redes com modelagem da topologia física. A **camada 4 (protocolo)** contém a especialização dos diversos componentes definidos em camadas inferiores (como `Node` e `HostNode`, por exemplo), a fim de se adicionar a lógica do protocolo desejado. A **camada 5 (aplicação)** é onde se situa a aplicação de usuário para execução no simulador. A **camada 6 (configuração da simulação)** é utilizada para modelagem de componentes e comportamento da rede, o que se dá através de um arquivo de configuração, definindo assim um cenário de simulação.

Para suporte à simulação de falhas, os itens mais importantes da arquitetura do simulador estão localizados na camada 2. Esta contém os dois componentes fundamentais do Simmcast, utilizados na composição de topologias: nodos e caminhos, ou **Node** e **Path**, respectivamente. Um nodo é uma entidade abstrata no Simmcast que pode representar elementos diferentes de um sistema real, contendo uma ou mais *threads* (classe **NodeThread**, especialização de um processo definido na camada 1), que compartilham memória e "rodam sobre o mesmo hardware". Um caminho (classe **Path**), por sua vez, representa a conexão lógica entre dois nodos quaisquer, criando um caminho através do qual pacotes (classe **Packet**, também na camada 2) podem ser transmitidos.

4. Suporte à Simulação de Falhas no Simmcast

Esta seção discute a extensão do framework para suporte à simulação de falhas. A interface utilizada para especificar falhas no sistema é ponto chave para o sucesso do simulador, e ela se manifesta de duas formas: através da estrutura de classes do *framework* (como instâncias de seus componentes são criados e acessados), e do arquivo que descreve um experimento de simulação. Isto reflete no projeto do Simmcast-FT, conforme apresentado nas subseções a seguir.

4.1. Extensão de Componentes com um Modelo de Falhas

Um sistema distribuído tolerante a falhas admite um determinado conjunto de falhas, conforme sua especificação, em termos de funcionalidade, de premissas sobre o ambiente e do modelo de falhas adotado. O modelo de falhas, tal como [10, 11, 12, 13], define um conjunto de categorias de falhas; a especificação de um sistema distribuído tolerante a falhas pode então se basear na nomenclatura apresentada pelo modelo, visando definir precisamente quais condições são tratadas, o que assume-se não ocorrer, e o que não é tratado. O modelo de falhas adotado no Simmcast-FT, largamente baseado em [13], é apresentado na Tabela 1.

Tabela 1: Tipos de falha.

Tipo de Falha	Descrição
Colapso	Componente pára silenciosamente de funcionar
Colapso com Recuperação	Componente pára silenciosamente de funcionar, mas pode retornar mais tarde (com amnésia de estado)
Omissão	Componente omite resultados, de forma completa ou parcial
Temporização	Funcionamento com tempo arbitrário
Sintática	Comportamento incorreto, detectável
Semântica	Comportamento correto com sentido incorreto

Conforme indicado anteriormente, o suporte à simulação de falhas no *framework* é realizado através da extensão dos componentes básicos do Simmcast, nodo e caminho. No

Simmcast-FT, a interface destes componentes passa a oferecer métodos que são usados para estabelecer um comportamento de falhas para a instância do objeto correspondente. Existe um método para cada tipo de falha, de acordo com a Tabela 1.

Para especificação da condição de ativação da falha, métodos tomam como argumento de entrada um *predicado de ativação*: o mesmo é periodicamente avaliado, e quando resulta verdadeiro, a falha é ativada no componente. Quando a falha é ativada, o componente muda de estado e duas ações ocorrem instantaneamente: atuação da falha sobre o componente (por exemplo, causando a perda de mensagens), e avaliação de um *predicado de desativação*. Caso este seja verdadeiro, a falha será imediatamente desativada, sendo percebida como uma *falha efêmera*. Caso contrário, a falha continua ativa e o predicado de desativação continua a ser avaliado de forma periódica, até que se torne verdadeiro ou a simulação acabe. A peridiocidade para avaliação de predicados é definida pelo usuário, sendo o valor padrão configurado em 1 (unidade de tempo de simulação). No momento em que uma falha é ativada em um componente, o tempo é registrado como atributo do componente; enquanto permanecer ativa, o predicado de ativação não é avaliado.

Na expressão booleana que representa um predicado podem aparecer as constantes *True* e *False*, o valor atual do relógio, o tempo de ativação da última falha do componente, um valor extraído de uma distribuição aleatória definida pelo usuário, e um método a ser executado sobre o componente. Portanto, o usuário aciona a falha indicando no arquivo .sim o seguinte método:

```
<node>.<FaultType>(<on>, <off> [, <args>])
```

onde `<node>` é a identificação do objeto nodo, `<FaultType>` é o tipo de falha, `<on>` é o predicado de ativação e `<off>` de desativação. Dependendo do tipo de falha, argumentos opcionais (indicados acima por `[, <args>]`) são utilizados. A seguir, descreve-se o mapeamento das diferentes categorias de falha nos componentes fundamentais do Simmcast, nodo e caminho.

4.2. Falhas de Nodo

O tipo mais simples de falha de um nodo é a de **colapso**, quando o mesmo silenciosamente pára de funcionar. Todas as mensagens que se encontravam em propagação em conexões diretas com o nodo, ou nas filas de envio na camada subjacente do mesmo, são silenciosamente perdidas; o estado do nodo é apagado (amnésia). Em caso de colapso sem recuperação, o predicado off será *False*. De outra forma, será especificado em função do relógio atual, o tempo da última falha e o tempo (médio) de recuperação. Com falha de **temporização** em nodos, é possível configurar um valor arbitrário para um relógio de um nodo, bem como uma dada velocidade de avanço, de forma que divirja do relógio da simulação.

O restante das falhas de nodos refletem na troca de mensagens. Quando um nodo falha por **omissão**, ele deixa de ser visto pelos demais, o que corresponde ao descarte de todas as mensagens enviadas pelo nodo. Quando uma falha **sintática** em um nodo é ativada, mensagens enviadas pelo nodo assumem uma sintaxe incorreta; predicados probabilísticos podem ser usados para que parte das mensagens seja afetada. A falha **semântica** de um nodo, quando ativada, faz com que o mesmo passe a enviar mensagens que são sintaticamente corretas (e portanto não geram exceção), mas contendo valores incorretos (semântica incorreta).

4.3. Falhas de Caminhos

A arquitetura de falhas do Simmcast-FT estende, de forma elegante, o modelo de falhas definido para nodos, para os componentes que representam caminhos. Assim, caminhos

estão sujeitos às mesmas categorias de falhas do que nodos, e podem também ser ativados ou desativados via predicados. Falhas em caminho, como na vida real, impactam na transmissão de mensagens.

Quando uma falha de **colapso** é ativada em um caminho, o transporte de mensagens pelo mesmo é encerrado, com descarte silencioso de quaisquer mensagens futuras ou em propagação, em ambas as direções. Assim como em nodos, colapsos podem ser modelados com e sem recuperação, dependendo do predicado de desativação. O tipo de falha mais comumente associado a caminhos é o de **omissão**, com descarte de parte das mensagens. Falhas de **temporização** aplicadas a caminhos fazem com que mensagens transportadas sejam atrasadas segundo um valor constante ou obtido de uma distribuição.

Falhas **sintáticas** são aplicadas a caminhos de forma a embaralhar, ou seja, corromper, mensagens: tal como para nodos, assume-se que mensagens afetadas sejam detectáveis por receptores. Por fim, caminhos podem ser alvo de falhas **semânticas**: quando tal falha é ativada, todas as mensagens que são transportadas pelo caminho, incluindo aquelas que se encontram em propagação, podem ter seus valores internos alterados.

4.4. Cenário de Falhas

Nas seções anteriores, foram apresentadas as diversas categorias de falhas e como as mesmas são mapeadas nos componentes do Simmcast. Embora a interface resultante seja concisa (extensão de apenas duas classes), elegante (simétrica, aplica os mesmos métodos às classes) e flexível (devido a forma com que falhas são ativadas e desativadas), vislumbra-se a combinação de um certo conjunto de falhas em componentes chamados "cenários de falhas". Os cenários podem ser usados para representar casos típicos, que expressem um modelo comum, ou que sejam próximos da realidade. Estas configurações prontas ou personalizadas, fornecidas junto ao *framework*, tem o intuito de facilitar o uso do modelo de falhas no Simmcast-FT.

Em diversas situações, é desejável que uma determinada falha seja aplicada a um conjunto de nodos. Para tal, a classe `Network` é estendida de forma a oferecer métodos para seleção de um conjunto de nodos ou um conjunto de caminhos. Os critérios para seleção passíveis de aplicação são os seguintes: **aleatório**, segundo uma dada probabilidade; **aleatório**, porém aplicado a um **subconjunto** de nodos/caminhos; **regional**, selecionando um conjunto de nodos segundo o identificador de um nodo central e um valor de escopo. Esta facilidade pode ser usada então, via arquivo ".sim", para selecionar uma região da topologia, ou um conjunto aleatório de nodos, e então sobre os mesmos aplicar um determinado comportamento de falhas.

5. Conclusões

Simulação fornece um ambiente controlado e facilita o teste das condições específicas que podem ocorrer em sistemas distribuídos tolerantes a falhas e que são difíceis de se reproduzir. Este artigo apresentou o projeto da API do Simmcast-FT, uma extensão do Simmcast para execução de experimentos com sistemas distribuídos tolerantes a falhas. A interface do Simmcast-FT permite descrever ambientes onde falhas ocorrem em componentes individuais ou em conjuntos de componentes em momentos específicos e/ou segundo condições especificadas no arquivo que descreve a simulação.

Este é um trabalho em andamento, e limita-se a apresentar a extensão da arquitetura do Simmcast, definindo um modelo de falhas suportado e um mapeamento em componentes do Simmcast-FT. Por fim, introduz-se o conceito de cenários de falhas, combinações de comportamentos de falhas e objetos que facilitam a especificação de falhas a

serem simuladas durante um experimento. A arquitetura se encontra em fase de prototipação, porém os resultados iniciais são animadores e serviram como "prova de conceito" e incentivo para a realização da mesma.

Referências

[1] M. P. Barcellos, H. H. Muhammad, and A. Detsch. Simmcast: a simulation tool for multicast protocol evaluation. In *XIX Simpósio Brasileiro de Redes de Computadores, SBRC 2001*, volume 1, pages 418 – 433, Florianópolis, Brasil, 2001.

[2] Opnet technologies. http://www.opnet.com/.

[3] The Network Simulator VINT ns-2. http://www.isi.edu/nsnam/ns.

[4] Scalable simulation framework - ssf. http://www.ssfnet.org/.

[5] C. Ciarfella, L. Moser, P. Melliar-Smith, and D. Agarwal. The Totem Protocol Development Environment. In *International Conference on Network Protocols, ICNP'94*, 1994.

[6] B. Weiss, G. Gridling, U. Schmid, and K. Schossmaier. The SimUTC Fault-Tolerant Distributed Systems Simulation Toolkit. In *7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, March 1999.

[7] G. A. Alvarez and F. Cristian. Cesium: Testing hard real-time and dependability properties of distributed protocols. In IEEE, editor, *3rd Workshop on Object-Oriented Real-Time Dependable Systems - (WORDS '97)*, Newport Beach, 1997.

[8] H. H. Muhammad, G. B. Bedin, G. Facchini, and M. P. Barcellos. Quebrando a barreira entre simulação e experimentação prática em redes de computadores. In SBC, editor, *XXII Simpósio Brasileiro de Redes de Computadores, SBRC 2004*, volume 1, Gramado, Brasil, maio 2004. SBC. a ser publicado.

[9] M. P. Barcellos, G. Facchini, L. F. Cintra, and H. H. Muhammad. Projeto do framework de simulação simmcast: uma arquitetura em camadas com ênfase na extensibilidade. In SBC, editor, *XXII Simpósio Brasileiro de Redes de Computadores, SBRC 2004*, volume 1, Gramado, Brasil, maio 2004. SBC. a ser publicado.

[10] K. Birman. *Building Secure and Reliable Network Applications*. Prentice Hall, 1996. 500 p.

[11] Pankaj Jalote. *Fault Tolerance in Distributed Systems*. Prentice-Hall, 1998.

[12] V. Hadzilacos and S. Toueg. *Distributed Systems*, chapter 5, Fault-Tolerant Broadcasts and Related Problems, pages 97–146. Addison-Wesley, 2nd. edition, 1998.

[13] P. Veríssimo and L. Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.