Modelos para injeção de falhas em ambiente móveis

Heinrich Felix Marmitt[1], **Sérgio Luis Cechin** [1], **Taisy Silva Weber**[1]

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

{hfmarmitt, cechin, taisy}@inf.ufrgs.br

Resumo. *Este artigo apresenta parte da pesquisa atual do Grupo de Tolerância a Falhas da UFRGS. O objetivo deste trabalho é pesquisar modelos adequados para a injeção de falhas em ambiente móveis, em especial para o sistema Android. Serão apresentados dois modelos de falhas de comunicação já pesquisados. Além disso será apresentado um* faultlet *escrito para o injetor de falhas FIRMAMENT que descreve o modelo de perda de pacotes em uma rede IEEE 802.11 e o resultado do primeiro teste executado com este* faultlet*.*

1. Introdução

A dependabilidade de um sistema é definida como a habilidade de evitar defeitos de serviços que são mais frequentes ou mais severos que o aceitável. Um dos meios para atingir esse critério é o uso de mecanismos de tolerância a falhas [Avizienis et al. 2004]. Contudo, os mecanismos de tolerância a falhas precisam ser testados e é inviável que o teste seja dependente da ocorrência natural de uma falha específica. A injeção de falhas é um dos métodos aceitos para testar esses mecanismos e está consolidado na literatura [Arlat et al. 1990] [Hsueh et al. 1997].

A computação móvel está cada vez mais presente no nosso cotidiano. Grande parte dos dispositivos móveis (notebooks, netbooks e smartphones) já saem de fábrica equipados com um ou mais dispositivos para a comunicação em rede. Com a popularização destes aparelhos, a quantidade de aplicativos que necessitam atender a critérios de dependabilidade aumenta. Estes aplicativos podem utilizar técnicas de tolerância a falhas, que precisam ser devidamente testadas.

Este trabalho tem como objetivo pesquisar e definir modelos adequados para a injeção de falhas de comunicação em ambientes móveis. Estes serão utilizados posteriormente para a validação de aplicações desenvolvidas para o sistema Android. O Android é uma plataforma desenvolvida pela Open Handset Alliance para dispositivos móveis e possui um sistema operacional baseado no kernel Linux versão 2.6 [Android-ADG 2010].

O artigo será organizado seguindo a seguinte ordem. A seção 2 revisará rapidamente alguns conceitos de injeção de falhas e de redes móveis, incluindo os modelos matemáticos existentes para falhas nessas redes. A seção 3 apresentará a plataforma Android e suas principais características. A seção 4 apresentará o injetor de falhas utilizado no projeto e os métodos de entrada para os modelos de falhas utilizados por ele. A seção 5 contém a implementação da carga de falhas já escrita para o injetor e o teste executado para verificar se atende as funcionalidades esperadas. Finalmente, serão apresentadas algumas conclusões e os passos a serem desenvolvidos em trabalhos futuros.

2. Injeção de falhas e redes móveis

A injeção de falhas é um dos modos de validação de sistemas que precisam atender a critérios de dependabilidade. Para cada fase utilizam-se métodos diferentes, que testam diferentes características do sistema [Hsueh et al. 1997]. É uma abordagem antiga, utilizada por Arlat há 20 anos, por exemplo [Arlat et al. 1990]. A injeção de falhas pode ser utilizada em diferente fases do desenvolvimento de um sistema.

Ainda segundo Hsueh, uma das técnicas de injeção de falhas é baseada em simulações, que assume que os erros ocorrem seguindo uma determinada distribuição de probabilidade [Hsueh et al. 1997]. Esta técnica é útil para avaliar a eficácia dos mecanismos de tolerância a falhas e a dependabilidade do sistema, contudo é importante que as entradas reproduzam fielmente o modelo de falhas do sistema. Por esta razão, este trabalho se concentrará em definir modelos de falhas que testem a interface IEEE 802.11 da plataforma Android.

O IEEE 802.11 é um padrão para comunicação sem fio definido pelo Institute of Electrical and Electronics Engineers (IEEE) e é amplamente utilizado em redes locais sem fio (*Wireless Local Area Networks*, WLANs). Neste trabalho não serão abordados detalhes específicos do padrão, se o leitor desejar um maior aprofundamento a especificação completa do padrão pode ser encontrada na literatura especializada [IEEE 2007].

Segundo Boggia, as WLAN's possuem naturalmente uma tendência de apresentar erros mais frequentemente que redes com fio (IEEE 802.3) e um dos principais desafios consiste em modelar o padrão dos erros que ocorrem em forma de rajadas [Boggia et al. 2009]. Isto acontece principalmente devido aos fenômenos físicos que afetam o meio de transmissão, como atenuação do sinal e interferências externas.

Neste trabalho duas classes de modelos serão estudadas e, posteriormente, implementadas em um injetor de falhas para fornecer a prova de conceito. A primeira é uma classe de perda de pacotes (*packet loss*), que é baseada no modelo de Gilbert-Elliot para canais de rádio-frequência [Gilbert 1960] [Elliott 1963]. A outra classe trata dos atrasos de entrega de pacotes, que ocorrem principalmente por colisões [Raptis et al. 2009].

2.1. Perda de pacotes

Khayam realizou testes experimentais utilizando uma rede IEEE 802.11b [Khayam et al. 2003]. Neste estudo o autor propõe que o modelo básico de Gilbert-Elliot (GE), que consiste em uma Cadeia de Markov de dois estados, pode ser utilizado para modelar os padrões de perdas de pacotes corretamente. Os estados foram nomeados como *Good* (g) e *Bad* (b), que simbolizam, respectivamente, a ausência e a presença de falha. A tabela 1 mostra os resultados experimentais de Khayam com o tamanho médio das rajadas (*Mean*) e a tabela 2 mostra os parâmetros das Cadeias de Markov que modelam estes padrões de falhas.

Na figura 1, pode-se ver a cadeia de Markov do modelo GE. O estado B representa o estado de falhas (*Bad*), com a probabilidade de falhas igual a α. O estado G (*Good*) representa o estado de operação normal, com probabilidade de falhas igual a (1 - β).

De acordo com o modelo, a probabilidade de ocorrência de uma rajada de tamanho *n* pode ser calculada pela fórmula:

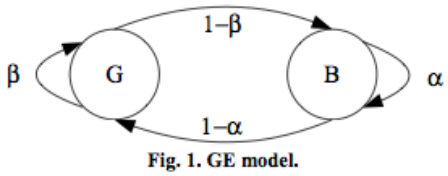

Fig. 1. GE model.

Figura 1. Modelo de Gilbert-Elliot [Carvalho et al. 2005]

$$P[X = n] = \alpha^{n-1} * (1 - \alpha) \tag{1}$$

Carvalho propõe que seja utilizada uma distribuição de probabilidade logarítmica ao invés da distribuição geométrica utilizada pelo modelo de Gilbert-Elliot. Além disso, o autor mostra que, apesar de nenhum dos modelos conseguir reproduzir fielmente os padrões de falhas, o modelo proposto produz resultados mais aproximados da realidade do que o modelo básico (GE) [Carvalho et al. 2005].

Tabela 1. Estatísticas de perdas de pacotes para diferentes taxas de transmissão na rede IEEE 802.11b [Khayam et al. 2003]

Taxa de Transmissão	Tamanho médio da rajada	Desvio-padrão	Taxa de transferência (%)
2 Mbps	0,00029	0,02329	99,9825
5,5 Mbps	1,67930	2,94070	63,8613
11 Mbps	16,3493	12,3343	14,2361

Tabela 2. Probabilidades da cadeia de Markov de dois estados para diferentes taxas de transmissão na rede IEEE 802.11b [Khayam et al. 2003]

Taxa de transmissão	P(gg)	P(gb)	P(bg)	P(bb)	P(pacote)
2 Mbps	0,999	0,0001	0,6470	0,3529	0,0002
5,5 Mbps	0,820	0,1794	0,3163	0,6836	0,3614
11 Mbps	0,376	0,6235	0,1034	0,8966	0,8576

2.2. Atraso de entrega

Raptis propõe um modelo computacionalmente simples para o atraso de entrega de pacotes em redes IEEE 802.11 [Raptis et al. 2006]. Este modelo é baseado em cadeias de Markov, assim como os modelos de perda de pacotes. Além disso, assume que o meio de transmissão é livre de erros e que os pacotes transmitidos possuem tamanho fixo.

O ponto-chave do modelo proposto por Raptis é que cada pacote possui uma probabilidade fixa de colisão p, que é independente de fatores como o *backoff* presente no mecanismo CSMA/CA (utilizado no padrão IEEE 802.11). Assim, os autores chegam a uma distribuição de probabilidades que serve para o cálculo de quantas colisões ocorreram antes que o pacote seja corretamente entregue, o que, pelo modelo, indica qual o atraso que afetará este pacote [Raptis et al. 2006].

3. Android

O sistema Android [Android-ADG 2010] é uma plataforma para ambientes móveis. Desenvolvido inicialmente pela Google, atualmente o projeto é coordenado pela Open Handset Alliance (OHA). A OHA é formada por um conjunto de empresas de comunicação e tecnologia que tem por objetivo criar padrões para a telefonia móvel.

O Android é baseado no kernel Linux e possibilita que desenvolvedores independentes criem aplicações que utilizem diretamente todas as funcionalidades do aparelho móvel. Deste modo, as aplicações têm acesso direto a funcionalidades como: fazer chamadas telefônicas, utilizar a câmera fotográfica, utilizar a rede sem fio, entre outras. Em maio de 2010, foi lançada a versão 2.2 rev 1, que utiliza o kernel Linux 2.6.32.

4. FIRMAMENT

O FIRMAMENT [Drebes et al. 2006] é um injetor de falhas que utiliza a interface de programação Netfilter [Russell and Welte 2002], disponível nas versões posteriores à versão 2.4 do núcleo Linux. Ele foi projetado para ser um injetor de falhas por software, atuando principalmente sobre o protocolo de rede IP. Segundo Drebes, devido às características do protocolo IP, este é um ponto ideal para a injeção de falhas de comunicação.

Uma das principais preocupações em relação a injetores de falhas por software se refere à intrusividade do injetor no sistema. Assim, o injetor de falhas será eficiente quando tiver o menor impacto possível no desempenho do sistema. Um dos modos de alcançar essa eficiência é estar localizado o mais próximo possível do hardware.

O FIRMAMENT procura reduzir a sua intrusividade se localizando no nível do sistema operacional. Na prática, o injetor alcança a baixa intrusividade por ser um módulo disponível para o núcleo Linux. Além disso, essa abordagem permite que o injetor seja executado com o mínimo de alterações possível no sistema original. Ele utiliza os ganchos disponíveis na interface Netfilter para interceptar pacotes que utilizem o protocolo IPv4 ou IPv6.

Quando o pacote é interceptado, a máquina virtual FIRM_VM executa microprogramas chamados de *faultlets*, que expressam os cenários de falhas utilizados nos testes. Após a execução, a máquina virtual retorna à pilha de protocolos a ação a ser realizada sobre o pacote. Entre as ações possíveis encontram-se: NF_ACCEPT (o pacote continua pelo caminho normal), NF_DROP (o pacote é descartado) e NF_QUEUE (o pacote é enviado para uma função definida pelo usuário). A ação NF_QUEUE é utilizada pelo FIRMAMENT para emular atrasos do pacote.

O FIRMAMENT já foi portado com sucesso para o sistema Android [Acker et al. 2010].

4.1. Faultlets

Um *faultlet* é uma aplicação com alto poder de expressão que descreve uma carga de falhas [Drebes et al. 2006]. No FIRMAMENT, pode ser definido um *faultlet* para cada um dos fluxos disponíveis (IPv4_IN, IPv4_OUT, IPv6_IN, IPv6_OUT). Este *faultlet* é executado cada vez que um pacote é capturado pelos ganchos do Netfilter, podendo, por exemplo, alterar a rota do pacote ou modificar o seu conteúdo.

Uma característica que contribui para o alto poder de expressão de um *faultlet* é a presença de registradores de uso geral na máquina virtual FIRM_VM que o executa. Deste modo, um *faultlet* pode possuir memória de estados anteriores, o que permite alterar o seu fluxo de execução a partir de informações dos pacotes em processamento.

5. Faultlet de perdas em rajada

Foram executados testes com o objetivo de verificar se o injetor utilizado possui a expressividade necessária para descrever a carga de falhas baseada no modelo de Gilbert-Elliot. Inicialmente foi criado um *faultlet* que emula situações de perda de pacotes, que foi compilado e executado em duas máquinas virtuais.

O sistema base é composto de um Intel Core2Duo 2.26 GHz e 2 GB de RAM, executando o sistema operacional Mac OS X 10.6. As máquinas virtuais foram executadas utilizando-se o programa Sun VirtualBox 3.1.8. A primeira máquina virtual consiste de uma instalação Fedora 10 sobre um núcleo Linux versão 2.6.27. A segunda máquina virtual utiliza o sistema operacional Fedora 12 sobre um núcleo Linux versão 2.6.32. Em ambas as máquinas virtuais as versões do FIRMAMENT utilizadas possuíam as modificações necessárias para sua compilação ter sucesso.

O teste do *faultlet* foi executado utilizando-se uma aplicação cliente-servidor sobre UDP. Tanto o cliente quanto o servidor foram programados em Python. Cada pacote enviado pelo servidor continha um número de sequência em sua área de dados. Conforme o cliente recebia os pacotes, as rajadas de perdas eram calculadas pela diferença entre o número de sequência atual e o número de sequência do último pacote recebido. Apesar do tráfego de rede não estar sujeito a interferências pela utilização de uma rede virtual, o servidor fechava o socket após a transmissão de cada pacote e aguardava 10 ms antes de iniciar a transmissão do próximo pacote.

O *faultlet* executado emula o modelo básico de Gilbert-Elliot. Este modelo consiste em uma cadeia de Markov com dois estados (normal [G, *Good*] e falha [B, *Bad*]). Inicialmente os valores probabilísticos das transições foram arbitrários e mais altos que os esperados para uma rede 802.11 em uma situação normal de operação. Isto foi feito para facilitar a verificação do funcionamento dos modelos e do injetor de falhas nos núcleos escolhidos. No *faultlet*, foram utilizados os valores de transição contidos na tabela 3.

Neste modelo foram executadas 4 transmissões em horários e execuções diferentes do sistema para que o gerador de números randômicos do FIRMAMENT fosse alterado. Em cada transmissão foram enviados 100.000 pacotes. A tabela 4 mostra uma comparação entre as probabilidades de ocorrência calculadas pela fórmula (1) e os resultados das transmissões. As probabilidades de ocorrência de rajadas de até 4 pacotes foram muito próximas das probabilidades calculadas teoricamente, o que indica que o *faultlet* criado descreve corretamente a carga de falhas desejada.

Código 1. Faultlet que descreve o modelo de Gilbert-Elliot

```
1           SET 9 R0          ;Filtra pacotes do tipo especificado
2           READB R0 R1
3           SET 17 R0         ;17 = UDP, 6 = TCP, 1 = ICMP
4           SUB R0 R1
5           JMPZ R1 UDP       ;Se for do tipo, segue adiante
6           ACP               ;senao, aceita o pacote
7   UDP:
8           SET 16 R0         ;Filtra pacotes pelo endereco IP destino
9           READW R0 R1
10          SET 0xc0a80002 R0 ;testa se o IP = 192.168.0.2 em hexadecimal
11          SUB R0 R1
12          JMPZ R1 IP        ;Se for o IP correto, segue adiante
13          ACP               ;senao, aceita o pacote
14  IP:
15          JMPZ R9 G_TESTE   ;Teste do estado atual 0 = Good, 1 = Bad
16  B_TESTE:
17          SET 5000 R2       ;Gera um numero aleatorio entre -5000 e 5000
18          RND R2 R3
19          SET 4000 R2       ;muda o intervalo => -1000 a 9000
20          ADD R2 R3
21          JMPN R3 B_B_TRAN;se negativo (10% de chance), executa uma
22          SET 0 R9          ;transicao B_B, senao altera o estado e aceita
23          ACP               ;o pacote
24
25  B_B_TRAN:
26          DRP               ;Continua no estado B e rejeita o pacote
27
28  G_TESTE:
29          SET 5000 R2       ;Gera um numero aleatorio entre -5000 e 5000
30          RND R2 R3
31          SET 4500 R2       ;Muda o intervalo => -500 a 9500
32          ADD R2 R3
33          JMPN R3 G_B_TRAN;Se negativo (5%) executa uma transicao G_B
34          SET 0 R9          ;senao, mantem o estado e aceita o pacote
35          ACP
36  G_B_TRAN:
37          SET 1 R9          ;Muda o estado para B e rejeita o pacote
38          DRP
```

6. Conclusão

Mesmo com a popularização de redes móveis e a importância do teste de dependabilidade em várias aplicações, existem poucos trabalhos publicados que relacionam diretamente as duas áreas. Grande parte dos modelos matemáticos para transmissões sem fio são baseados nos trabalhos de Gilbert e Elliot sobre canais de rádio-frequência.

Tabela 3. Probabilidades de transição entre os estados

	G	B
G	0,95 (ß)	0,05
B	0,90	0,10 (α)

Tabela 4. Comparação entre a distribuição esperada de rajadas e os resultados do experimento com o *faultlet*

Tamanho da rajada (pacotes)	Probabilidade teórica	Probabilidade calculada sobre a média de rajadas	Primeiro experimento (pacotes)	Segundo experimento (pacotes)	Terceiro experimento (pacotes)	Quarto experimento (pacotes)
1	0,9	0,9011	4245	4259	4247	4247
2	0,09	0,0874	429	407	417	395
3	0,009	0,01039	52	38	57	49
4	0,0009	0,001007	4	4	6	5
5	0,00009	0,00005301	0	1	0	0
6	0,000009	0	0	0	0	0
7	0,0000009	0,00005301	0	0	1	0
Total Pacotes	--	5242,5	5275	5208	5283	5204
Total Rajadas	--	4715,75	4730	4709	4728	4696

Apesar dos modelos pesquisados não reproduzirem fielmente as ocorrências de falhas de redes 802.11, eles possuem baixa complexidade computacional e isto os torna especialmente atrativos para serem utilizados como modelos para injeção de falhas de comunicação.

Com a implementação bem sucedida do *faultlet* que utiliza Cadeias de Markov, se comprovou que o injetor escolhido é adequado para injeção de falhas de acordo com os modelos pesquisados, mesmo que não tenha sido projetado inicialmente para redes móveis.

Os trabalhos futuros estarão relacionados com a implementação de um *faultlet* que emule falhas de atraso e validação de aplicativos para o Android utilizando o FIRMAMENT e os *faultlets* criados.

Referências

Acker, E. V., Weber, T. S., and Cechin, S. L. (2010). Injeção de falhas para validar aplicações em ambientes móveis. In *Anais / XI Workshop de Testes e Tolerância a Falhas*, pages 61–74, Gramado, RS.

Android-ADG (2010). What is android?. http://developer.android.com/guide/index.html. [acessado em maio/2010].

Arlat, J., Aguera, M., Amat, L., Crouzet, Y., Fabre, J.-C., Laprie, J.-C., Martins, E., and Powell, D. (1990). Fault injection for dependability validation: A methodology and some applications. *IEEE Trans. Softw. Eng.*, 16(2):166–182.

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

Boggia, G., Camarda, P., and D'Alconzo, A. (2009). Performance of Markov models for frame-level errors in IEEE 802.11 wireless LANs. *Int. J. Commun. Syst.*, 22(6):695–718.

Carvalho, L., Angeja, J., and Navarro, A. (2005). A new packet loss model of the IEEE 802.11g wireless network for multimedia communications. *Consumer Electronics, IEEE Transactions on*, 51(3):809 – 814.

Drebes, R. J., Jacques-Silva, G., Trindade, J., and Weber, T. S. (2006). A kernel based communication fault injector for dependability testing of distributed systems. In Springer-Verlag, editor, *First Int. Haifa Verification Conf.*, volume 3875, pages 177–190.

Elliott, E. O. (1963). Estimates of error rates for codes on burst-noise channels. *Bell Syst. Tech. J*, (42):1977–1997.

Gilbert, E. (1960). Capacity of a burst-noise channel. *Bell Systems Technical Journal*, (39).

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

IEEE (2007). IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pages C1 –1184.

Khayam, S. A., Karande, S., Radha, H., and Loguinov, D. (2003). Performance analysis and modeling of errors and losses over 802.11b lans for high-bit-rate real-time multimedia. *Signal Processing: Image Communication*, 18(7):575 – 595.

Raptis, P., Banchs, A., and Paparrizos, K. (2006). A simple and effective delay distribution analysis for IEEE 802.11. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, pages 1 –5.

Raptis, P., Vitsas, V., and Paparrizos, K. (2009). Packet delay metrics for IEEE 802.11 Distributed Coordination Function. *Mob. Netw. Appl.*, 14(6):772–781.

Russell, R. and Welte, H. (2002). Linux netfilter hacking howto. www.netfilter.org.