XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos

30 de maio a 3 de junho de 2011

Campo Grande • Mato Grosso do Sul

http://sbrc2011.facom.ufms.br

SBRC 2011

XII Workshop de Testes e Tolerância a Falhas (WTF 2011)

ANAIS

XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos
30 de maio a 3 de junho de 2011
Campo Grande, MS

XII Workshop de Testes e Tolerância a Falhas (WTF)

Editora
Sociedade Brasileira de Computação (SBC)

Organizadores
Luiz Eduardo Buzato (UNICAMP)
Fábio Moreira Costa (UFG)
Ronaldo Alves Ferreira (UFMS)

Realização
Faculdade de Computação
Universidade Federal de Mato Grosso do Sul (UFMS)

Promoção
Sociedade Brasileira de Computação (SBC)
Laboratório Nacional de Redes de Computadores (LARC)

Copyright © 2011 da Sociedade Brasileira de Computação
Todos os direitos reservados

Capa: Venise Melo
Produção Editorial: Lucilene Vilela Gonçalves, Ronaldo Alves Ferreira

Cópias Adicionais:
Sociedade Brasileira de Computação (SBC)
Av. Bento Gonçalves, 9500 – Setor 4 – Prédio 43.412 – Sala 219
Bairro Agronomia – CEP 91.509-900 – Porto Alegre – RS
Fone: (51) 3308-6835
E-mail: sbc@sbc.org.br

Dados Internacionais de Catalogação na Publicação (CIP)

Workshop de Testes e Tolerância a Falhas (12.: 2011 : Campo Grande, MS).

Anais / XII Workshop de Testes e Tolerância a Falhas; organizadores Luiz Eduardo Buzato... et al. – Porto Alegre : SBC, c2011.
90 p.

ISSN 2177-496X

1. Redes de computadores. 2. Sistemas distribuídos. I. Buzato, Luiz Eduardo. II. Título.

Promoção

Sociedade Brasileira de Computação (SBC)

Diretoria

Presidente
José Carlos Maldonado (USP)

Vice-Presidente
Marcelo Walter (UFRGS)

Diretor Administrativo
Luciano Paschoal Gaspary (UFRGS)

Diretor de Finanças
Paulo Cesar Masiero (USP)

Diretor de Eventos e Comissões Especiais
Lisandro Zambenedetti Granville (UFRGS)

Diretora de Educação
Mirella Moura Moro (UFMG)

Diretora de Publicações
Karin Breitman (PUC-Rio)

Diretora de Planejamento e Programas Especiais
Ana Carolina Salgado (UFPE)

Diretora de Secretarias Regionais
Thais Vasconcelos Batista (UFRN)

Diretor de Divulgação e Marketing
Altigran Soares da Silva (UFAM)

Diretor de Regulamentação da Profissão
Ricardo de Oliveira Anido (UNICAMP)

Diretor de Eventos Especiais
Carlos Eduardo Ferreira (USP)

Diretor de Cooperação com Sociedades Científicas
Marcelo Walter (UFRGS)

Promoção

Conselho

Mandato 2009-2013
Virgílio Almeida (UFMG)
Flávio Rech Wagner (UFRGS)
Silvio Romero de Lemos Meira (UFPE)
Itana Maria de Souza Gimenes (UEM)
Jacques Wainer (UNICAMP)

Mandato 2007-2011
Cláudia Maria Bauzer Medeiros (UNICAMP)
Roberto da Silva Bigonha (UFMG)
Cláudio Leonardo Lucchesi (UFMS)
Daltro José Nunes (UFRGS)
André Ponce de Leon F. de Carvalho (USP)

Suplentes – Mandato 2009-2011
Geraldo B. Xexeo (UFRJ)
Taisy Silva Weber (UFRGS)
Marta Lima de Queiroz Mattoso (UFRJ)
Raul Sidnei Wazlawick (PUCRS)
Renata Vieira (PUCRS)

Laboratório Nacional de Redes de Computadores (LARC)

Diretoria

Diretor do Conselho Técnico-Científico
Artur Ziviani (LNCC)

Diretor Executivo
Célio Vinicius Neves de Albuquerque (UFF)

Vice-Diretora do Conselho Técnico-Científico
Flávia Coimbra Delicato (UFRN)

Vice-Diretor Executivo
Luciano Paschoal Gaspary (UFRGS)

Membros Institucionais
CEFET-CE, CEFET-PR, IME, INPE/MCT, LNCC, PUCPR, PUC-RIO, SESU/MEC, UECEM UERJ, UFAM, UFBA, UFC, UFCG, UFES, UFF, UFMG, UFMS, UFPA, UFPB, UFPE, UFPR, UFRGS, UFRJ, UFRN, UFSC, UFSCAR, UNICAMP, UNIFACS, USP

Realização

Comitê de Organização

Coordenação Geral
Ronaldo Alves Ferreira (UFMS)

Coordenação do Comitê de Programa
Artur Ziviani (LNCC)
Bruno Schulze (LNCC)

Coordenação de Palestras e Tutoriais
Nelson Luis Saldanha da Fonseca (UNICAMP)

Coordenação de Painéis e Debates
José Augusto Suruagy Monteiro (UNIFACS)

Coordenação de Minicursos
Fabíola Gonçalves Pereira Greve (UFBA)

Coordenação de Workshops
Fábio Moreira Costa (UFG)

Coordenação do Salão de Ferramentas
Luis Carlos Erpen De Bona (UFPR)

Comitê Consultivo
Antônio Jorge Gomes Abelém (UFPA)
Carlos André Guimarães Ferraz (UFPE)
Francisco Vilar Brasileiro (UFCG)
Lisandro Zambenedetti Granville (UFRGS)
Luci Pirmez (UFRJ)
Luciano Paschoal Gaspary (UFRGS)
Marinho Pilla Barcellos (UFRGS)
Paulo André da Silva Gonçalves (UFPE)
Thais Vasconcelos Batista (UFRN)

Realização

Organização Local
Brivaldo Alves da Silva Jr. (UFMS)
Edson Norberto Cáceres (UFMS)
Eduardo Carlos Souza Martins (UFMS/POP-MS)
Hana Karina Sales Rubinstejn (UFMS)
Irineu Sotoma (UFMS)
Kátia Mara França (UFMS)
Luciano Gonda (UFMS)
Lucilene Vilela Gonçalves (POP-MS)
Márcio Aparecido Inácio da Silva (UFMS)
Marcos Paulo Moro (UFGD)
Massashi Emilson Oshiro (POP-MS)
Nalvo Franco de Almeida Jr. (UFMS)
Péricles Christian Moraes Lopes (UFMS)
Renato Porfírio Ishii (UFMS)

Mensagem do Coordenador Geral

Sejam bem-vindos ao XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2011) em Campo Grande, MS. É um prazer e uma distinção organizar um simpósio de tamanha relevância para a Computação no Brasil, mais ainda por ser a primeira vez que a Região Centro-Oeste tem o privilégio de sediá-lo. O SBRC é um evento anual promovido pela Sociedade Brasileira de Computação (SBC) e pelo Laboratório Nacional de Redes de Computadores (LARC). Ao longo dos seus quase trinta anos, o SBRC tornou-se o mais importante evento científico nacional em Redes de Computadores e Sistemas Distribuídos e um dos maiores da área de Informática no país.

O SBRC 2011 está com uma programação bastante rica, de qualidade diferenciada e que consiste em: 18 sessões técnicas de artigos completos que abordam o que há de mais novo nas áreas de redes de computadores e sistemas distribuídos; três sessões técnicas para apresentação de ferramentas selecionadas para o Salão de Ferramentas; cinco minicursos, com quatro horas de duração, sobre temas atuais; três palestras e três tutoriais com pesquisadores de alto prestígio internacional; e três painéis sobre assuntos de interesse da comunidade. Além dessas já tradicionais atividades do simpósio, ocorrerão em paralelo oito workshops: XVI Workshop de Gerência e Operação de Redes e Serviços (WGRS), XII Workshop da Rede Nacional de Ensino e Pesquisa (WRNP), XII Workshop de Testes e Tolerância a Falhas (WTF), IX Workshop em Clouds, Grids e Aplicações (WCGA), VII Workshop de Redes Dinâmicas e Sistemas P2P (WP2P), II Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF), I Workshop on Autonomic Distributed Systems (WoSIDA) e I Workshop de Redes de Acesso em Banda Larga (WRA).

O desafio de organizar um evento como o SBRC só pode ser cumprido com a ajuda de um grupo especial. Eu tive a felicidade de contar com a colaboração de inúmeras pessoas ao longo desta jornada. Meus sinceros agradecimentos aos membros dos Comitês de Organização Geral e Local por realizarem um trabalho de excelente qualidade e com muita eficiência, a qualidade da programação deste simpósio é fruto do trabalho dedicado dessas pessoas. Sou grato a Faculdade de Computação da UFMS por ter sido uma facilitadora ao longo de todo o processo de organização, desde a nossa proposta inicial até o fechamento da programação. Gostaria de agradecer, também, ao Comitê Gestor da Internet no Brasil (CGI.br), às agências governamentais de fomento e aos patrocinadores por reconhecerem a importância do SBRC e investirem recursos financeiros fundamentais para a realização do evento. Com o apoio financeiro recebido, foi possível manter os custos de inscrição baixos e oferecer um programa social de alta qualidade.

Em nome do Comitê Organizador, agradeço a todos os participantes pela presença em mais esta edição do SBRC e desejo uma semana produtiva, agradável e com estabelecimento de novas parcerias e amizades.

Ronaldo Alves Ferreira
Coordenador Geral do SBRC 2011

Mensagem do Coordenador de Workshops do SBRC 2011

Os workshops são uma parte tradicional do que hoje faz do SBRC o principal evento da área no país, sendo responsáveis por atrair uma parcela cada vez mais expressiva de participantes para o Simpósio todos os anos. O SBRC 2011 procurou manter essa tradição, com a realização de workshops já considerados parte do circuito nacional de divulgação científica nas várias subáreas de Redes de Computadores e Sistemas Distribuídos, como o WTF (Workshop de Testes e Tolerância a Falhas), o WCGA (Workshop em Clouds, Grids e Aplicações), o WGRS (Workshop de Gerência e Operação de Redes e Serviços) e o WP2P (Workshop de Redes Dinâmicas e Sistemas P2P). Incluímos também nesta lista de iniciativas bem sucedidas o WRNP (Workshop da Rede Nacional de Ensino e Pesquisa), que cumpre o importantíssimo papel de fazer a ponte entre as comunidades técnica e científica da área.

Como novidade em 2011, e reconhecendo o surgimento e o fortalecimento de novas linhas de pesquisa de expressiva importância dentro da comunidade brasileira de Redes e Sistemas Distribuídos, procuramos incentivar a criação de novos workshops dentro do Simpósio. Foi com esse intuito que introduzimos pela primeira vez no SBRC a chamada aberta de workshops, por meio da qual membros da comunidade foram convidados a submeter propostas de workshops inéditos para realização em conjunto com o SBRC 2011. Em resposta à chamada, recebemos nove propostas de alta qualidade, das quais oito foram aceitas e seus respectivos proponentes convidados a organizarem os workshops no SBRC em Campo Grande. Das oito propostas aceitas, cinco tratavam dos workshops já tradicionais acima mencionados, e uma referia-se à segunda edição de um workshop mais recentemente criado, mas que teve sua primeira edição realizada de forma muito bem sucedida no SBRC 2010, o WPEIF (Workshop de Pesquisa Experimental da Internet do Futuro). As outras duas propostas foram resultado direto da chamada aberta de workshops e resultaram na adição de dois novos eventos ao leque do SBRC, o WRA (Workshop de Redes de Acesso em Banda Larga) e o WoSiDA (Workshop on Autonomic Distributed Systems), ambos com ótima aceitação pela comunidade, a julgar pelos números de submissões de trabalhos recebidos.

Esperamos que 2011 seja mais um ano de sucesso para os workshops do SBRC, em particular para aqueles criados nesta edição do Simpósio, e para que eles continuem contribuindo como importantes fatores de agregação para os avanços promovidos pela comunidade científica da área de Redes e Sistemas Distribuídos no Brasil.

Aproveitamos para agradecer o inestimável apoio recebido de diversos membros da comunidade e, em particular, da Organização Geral do SBRC 2011.

A todos, um excelente SBRC em Campo Grande!

Fábio M. Costa
Coordenador de Workshops do SBRC 2011

Mensagem do Coordenador do WTF

O Workshop de Testes e Tolerância a Falhas chega à sua décima segunda edição. Ele tem representado, desde sua primeira edição em 1998, um fórum nacional importante para a troca de idéias, discussão e avaliação de projetos teóricos e práticos em sua área de concentração.

A programação do XII WTF inclui 6 artigos completos selecionados pelo Comitê de Programa e duas palestras convidadas. Na primeira sessão os artigos tem seu principal foco em testes e na segunda sessão o foco principal é a tolerância a falhas em sistemas sujeitos a falhas arbitrárias (bizantinas). Em complemento as sessões técnicas ocorrerão as palestras convidadas, que tratam de temas atuais associados a consenso, uma abstração importante para a construção de sistemas tolerantes a falhas. A primeira, ministrada pela Profa. Fabíola Greve (UFBA) abordará as condições necessárias para o estabelecimento de consenso em ambientes móveis. Na segunda palestra, o Prof. Miguel P. Correia (IST/UTL, Lisboa) abordará o uso de aleatorização para permitir que consenso seja implementado em sistemas conectados por rede sem fio sujeita a falhas dinâmicas. É gratificante constatar que o programa do XII WTF reflete a preocupação dos pesquisadores brasileiros em manterem projetos e pesquisa compatível com o desenvolvimento mundial da área de Tolerância a Falhas.

Gostaria de agradecer a todos os membros do Comitê de Programa pelo seu compromisso em retornar avaliações de qualidade em um prazo justo. Sem a pronta atuação de cada membro o WTF não teria acontecido. Agradeço de forma especial os membros do Comitê Gestor da Comissão Especial de Tolerância a Falhas (CE-TF) da Sociedade Brasileira de Computação (SBC), eles sempre estiveram prontos para conversar sobre a melhor forma de organizar o Workshop. Finalmente, gostaria de agradecer aos organizadores do SBRC 2011 pelo excelente apoio oferecido aos coordenadores dos Workshops associados ao Simpósio pela organização do SBRC.

Luiz Eduardo Buzato
Coordenador do WTF 2011

Comitê de Programa do WTF

Alcides Calsavara (PUCPR)
Alexandre Sztajnberg (UERJ)
Alysson Bessani (Universidade de Lisboa – Portugal)
Avelino Zorzo (PUCRS)
Cecilia Rubira (UNICAMP)
Daniel Mossé (Universidade de Pittsburgh – EUA)
Eduardo Bezerra (UFSC)
Eliane Martins (UNICAMP)
Elias P. Duarte Jr. (UFPR)
Fabíola Greve (UFBA)
Fernando Dotti (PUCRS)
Fernando Pedone (Universidade de Lugano – Suiça)
Frank Siqueira (UFSC)
Henrique Madeira (Universidade de Coimbra – Portugal)
Ingrid Jansch-Pôrto (UFRGS)
Irineu Sotoma (UFMS)
Joni da Silva Fraga (UFSC)
Jose Pereira (Universidade do Minho – Portugal)
Lau Cheuk Lung (UFSC)
Lívia Sampaio (UFCG)
Luciana Arantes (Universidade de Paris VI – França)
Miguel Correia (Universidade de Lisboa – Portugal)
Patricia Machado (UFCG)
Raimundo Barreto (UFAM)
Raimundo J. de Araújo Macêdo (UFBA)
Raul Ceretta Nunes (UFSM)
Regina Moraes (UNICAMP)
Rogerio de Lemos (Universidade de Kent – UK)
Rui Oliveira (Universidade do Minho – Portugal)
Sérgio Gorender (UFBA)
Taisy Weber (UFRGS)
Udo Fritzke Jr. (PUCMG)
Vidal Martins (PUCPR)

Revisores do WTF

Alcides Calsavara (PUCPR)
Alexandre Sztajnberg (UERJ)
Alysson Bessani (Universidade de Lisboa – Portugal)
Avelino Zorzo (PUCRS)
Cecilia Rubira (UNICAMP)
Daniel Mossé (Universidade de Pittsburgh – EUA)
Eduardo Bezerra (UFSC)
Eliane Martins (UNICAMP)
Elias P. Duarte Jr. (UFPR)
Fabíola Greve (UFBA)
Fernando Dotti (PUCRS)
Fernando Pedone (Universidade de Lugano – Suiça)
Frank Siqueira (UFSC)
Henrique Madeira (Universidade de Coimbra – Portugal)
Ingrid Jansch-Pôrto (UFRGS)
Irineu Sotoma (UFMS)
Joni da Silva Fraga (UFSC)
Jose Pereira (Universidade do Minho – Portugal)
Lau Cheuk Lung (UFSC)
Lívia Sampaio (UFCG)
Luciana Arantes (Universidade de Paris VI – França)
Miguel Correia (Universidade de Lisboa – Portugal)
Patricia Machado (UFCG)
Raimundo Barreto (UFAM)
Raimundo J. de Araújo Macêdo (UFBA)
Raul Ceretta Nunes (UFSM)
Regina Moraes (UNICAMP)
Rogerio de Lemos (Universidade de Kent – UK)
Rui Oliveira (Universidade do Minho – Portugal)
Sérgio Gorender (UFBA)
Taisy Weber (UFRGS)
Udo Fritzke Jr. (PUCMG)
Vidal Martins (PUCPR)

Sumário

Sessão Técnica 1 – Injeção de Falhas e Testes 1

MoDiVHA: Uma Estratégia Hierárquica para Assinalamento de Testes Distribuídos
Jefferson Paulo Koppe, Luis Carlos E. Bona e Elias P. Duarte Jr. (UFPR) 3

Estabelecendo Sequências de Teste de Integração de Classes: Um Estudo Comparativo da Aplicação de Três Algoritmos Evolutivos Multiobjetivos
Wesley Klewerton Guez Assunção (UFPR), Thelma Elita Colanzi (UEM), Silvia Regina Vergilio e Aurora Trinidad Ramirez Pozo (UFPR) 15

J-Attack: Injetor de Ataques para Avaliação de Segurança de Aplicações Web
Plinio C. S. Fernandes, Tânia Basso e Regina Moraes (UNICAMP) 29

Palestras Convidadas 43

Palestra 1: Connectivity Conditions for the Solvability of Fault-Tolerant Consensus in Asynchronous Unknown Networks
Profª. Fabíola Greve (UFBA) 45

Palestra 2: Randomization Can Be a Healer: Consensus with Dynamic Omission Failures
Profº. Miguel P. Correia (Universidade Técnica de Lisboa) 45

Sessão Técnica 2 – Tolerância a Falhas 47

Reviewing Dependability Issues and Suitable Solutions for Emerging Wireless Sensor Networks Applications
Carlos Oberdam Rolim, Edison Pignaton de Freitas (UFRGS), Tales Heimfarth (UFLA), Cláudio F. R. Geyer, Carlos Eduardo Pereira (UFRGS), Armando Morado Ferreira (IME), Flávio Rech Wagner (UFRGS) e Tony Larsson (Halmstad University) 49

Aspectos Práticos Sobre o Consenso Bizantino Entre Participantes Desconhecidos
Eduardo Adilio Pelinson Alchieri, Luiz Renato Tomelin (UFSC), Alysson Neves Bessani (Universidade de Lisboa) e Joni da Silva Fraga (UFSC) 63

Medidas de Conectividade Baseadas em Cortes de Vértices para Redes Complexas
Karine Pires (UFPR), Jaime Cohen (UEPG) e Elias P. Duarte Jr. (UFPR) 77

Índice por Autor 90

XII Workshop de Testes e Tolerância a Falhas

♦

Sessão Técnica 1

Injeção de Falhas e Testes

MoDiVHA: Uma Estratégia Hierárquica para Assinalamento de Testes Distribuídos

Jefferson Paulo Koppe[1], **Elias P. Duarte Jr.**[1], **Luis C. E. Bona**[1]

[1] Departamento de Informática –Universidade Federal do Paraná (UFPR)
Caixa Postal 19081 – 81531-990 –Curitiba – PR – Brazil

{jpkoppe,elias,bona}@inf.ufpr.br

Abstract. *Distributed diagnosis allows fault-free nodes of a system to determine the state of all nodes. The diagnosis is performed based on tests assigned to system nodes. In this work we present a hierarchical testing strategy that is scalable by definition. This strategy, called MoDiVHA, is based on DiVHA's test assignment [1], but produces less tests. Reducing the number of tests is important because it can represent a significant reduction on resources required by the diagnosis algorithm. Experimental results were obtained from three series of simulations conducted to compute the number of tests and diagnosis latency for various system sizes and fault situations.*

Resumo. *O diagnóstico distribuído permite que nodos sem-falha de um sistema determinem o estado de todos os nodos do sistema. O diagnóstico é realizado a partir dos resultados de testes assinalados aos nodos do sistema. Neste trabalho apresentamos uma estratégia de testes hierárquica que permite o diagnóstico escalável. Essa estratégia, denominada MoDiVHA, é baseada naquela do algoritmo DiVHA [1], mas resulta em um número de testes inferior. A diminuição do número de testes é importante pois pode representar uma redução significativa de recursos de sistema alocados para realização do diagnóstico. Os resultados experimentais foram obtidos através de três séries de simulações, conduzidas para computar o número de testes e a latência em diversos tamanhos de sistemas e configurações de falhas.*

1. Introdução

Um sistema de gerenciamento de falhas deve ser tolerante a falhas, uma vez que roda na rede e precisa ser confiável mesmo quando a rede apresentar falhas. O diagnóstico em nível de sistema é uma abordagem eficaz para construção de sistemas de gerenciamento distribuídos tolerantes a falhas [2], [3].

Considere um sistema com N nodos onde cada nodo pode estar em um de dois estados, falho ou sem-falha. Considere ainda que existe um enlace de comunicação entre qualquer par de nodo do sistema e que os enlaces de comunicação nunca falham. O objetivo do diagnóstico distribuído em nível de sistema é permitir que os nodos não-falhos determinem o estado, falho ou não-falho, de todos os outros nodos pertencentes ao sistema [4], [5], [6]. Utilizando algoritmos de diagnóstico distribuído é possível construir sistemas de monitoração de rede tolerante a falhas [8].

Os nodos em um sistema de diagnóstico são capazes de testar outros nodos e determinar seu estado corretamente. O conjunto de testes assinalados para cada um dos nodos, bem como a realização do diagnóstico a partir do resultado dos testes, são funções elementares de um algoritmo de diagnóstico distribuído.

O sistema é representado por um grafo direcionado $G = (V, E)$, onde os vértices do grafo representam os nodos do sistema e as arestas indicam os testes que são realizados. A figura 1 mostra um exemplo de sistema com 4 nodos representado por um grafo, o vértice na cor preta indica que o nodo está falho. Uma aresta saindo do nodo u para o nodo v significa que o nodo u testa o nodo v. Cada teste corresponde a um estímulo enviado cuja resposta classificará o nodo testado em falho ou sem-falha.

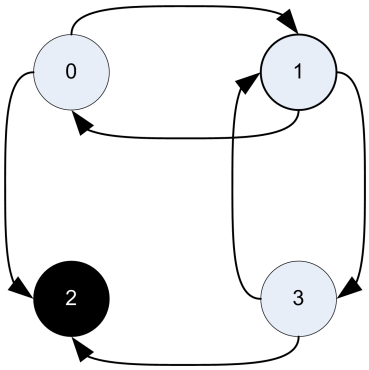


Figura 1. Um exemplo de assinalamento de testes; o nodo 2 está falho.

O desempenho de um algoritmo de diagnóstico distribuído é mensurado, principalmente, por dois valores: (1) a quantidade de testes necessários por rodada; (2) a latência de propagação de um evento. A latência é expressa em rodadas de testes e corresponde ao tempo necessário para que todos os nodos sem-falha descubram a ocorrência de um novo evento [1]. Uma rodada de testes é definida como o período de tempo em que todos os nodos no estado sem-falha executam seus testes. Quanto menor forem esses valores (quantidade de testes e latência) mais eficiente é o algoritmo. No entanto, a busca por um algoritmo ideal não é tão simples pois, o ganho com uma métrica, geralmente implica em perdas na outra métrica. O Adaptive-DSD [8] é um bom exemplo desta contraposição, sua necessidade de apenas N testes para concluir uma rodada de testes o torna muito eficiente neste quesito. Em contra partida este algoritmo tem uma latência de propagação de evento muito longa, podendo ser necessário até N rodadas de testes para que um novo evento seja conhecido por todos os nodos participantes do sistema.

Este trabalho apresenta uma abordagem para assinalamento de testes alternativa para algoritmo DiVHA [1], onde o número de testes por rodada é significativamente menor. Esta nova abordagem foi denominada MoDiVHA (*Modified Distributed Virtual Hypercube Algorithm*). O MoDiVHA mantêm intactas as propriedades principais do DiVHA, em particular a latência no pior caso, de forma que pode ser usado em sua substituição.

Implementações de algoritmos de diagnóstico distribuído, de forma geral, usam a troca de mensagens em rede como ferramenta para testar e compartilhar informações sobre o estado de um ou mais nodos. Desta forma, a otimização da quantidade de testes e,

consequentemente, dos *links* para troca de informações, é importante pois pode representar uma redução significativa de recursos de rede alocados para realização do diagnóstico.

O algoritmo foi implementado através de simulação e os resultados experimentais são apresentados mostrando o funcionamento do algoritmo para diversos tamanhos de sistemas e configurações de falhas. Foram conduzidas três séries de simulações. A primeira simulação mostra o número de testes necessários sem considerar a ocorrência de eventos e/ou nodos falhos. O resultado deste experimento é então comparado com os dados obtidos de experimento similar baseado no algoritmo DiVHA. A segunda série de experimentados apresentados considera diversas configurações de falhas e computa o número de testes necessários para cada uma delas. Finalmente, o terceiro experimento tem como objetivo determinar a latência do MoDiVHA para uma série de situações de falhas geradas aleatoriamente.

O restante deste artigo está organizando da seguinte maneira. A sessão 2 apresenta os principais modelos de diagnóstico distribuído em nível de sistema. Na sessão 3 o algoritmo MoDiVHA é especificado. Os resultados de simulações e uma avaliação empírica são apresentados na sessão 4. A sessão 5 conclui este trabalho.

2. Trabalhos Relacionados

O primeiro modelo de diagnóstico foi o PMC, cujo nome deriva das iniciais dos autores [7]. O modelo PMC assume a existência de um observador central que coleta todos os resultados de testes realizados pelos nodos, analisa esses resultados e então determina os estados dos nodos do sistema. Os testes entres os nodos são fixados previamente e o conjunto de testes possibilita que o diagnostico correto do sistema seja possível. Além do diagnóstico, os autores também definem a *diagnosticabilidade* da seguinte maneira: um sistema é *t-diagnosticável* se dada uma síndrome, o supervisor central completa o diagnóstico se o número de unidades falhas não exceder t.

Ao contrário do modelo PMC, o Adaptive-DSD (BIANCHINI; BUSKENS, 1991, 1992) realiza diagnóstico distribuído e adaptativo, de forma que não necessita de uma unidade central para a realização do diagnóstico e a configuração dos testes é determinada dinamicamente. No algoritmo Adaptive-DSD cada nodo testa outros nodos até encontrar um nodo não-falho, do qual colhe dados para atualizar suas informações de diagnóstico. Em termos de quantidade de testes, o algoritmo Adaptive-DSD é considerado eficiente sendo executados no máximo N testes por rodada. Quanto à latência, podem ser necessárias até N rodadas para que todos os nodos realizem o diagnóstico de um evento [1].

O algoritmo Hi-ADSD (DUARTE Jr.; NANYA, 1998) realiza os testes de forma hierárquica utilizando o conceito de *clusters*. A figura 2 mostra um sistema de oito nodos organizado em *clusters*. Durante a execução dos testes os nodos são agrupados em *clusters*, cujo tamanho é sempre uma potência de 2. Um nodo inicialmente realiza testes em *clusters* de tamanho 2 (2^1), nas próximas rodadas os *clusters* têm tamanho 4 (2^2), 8 (2^3) e assim sucessivamente até atingir tamanho $N/2(2^{(log_2 N)-1})$. No pior caso o número de testes executados pelo Hi-ADSD [9] é de $N^2/2$. A latência é de $log_2 N$ rodadas de testes.

O Algoritmo DiVHA (*Distributed Virtual Hypercube Algorithm*) [1], determina um grafo de testes $T(S)$ com topologia baseada em um hipercubo. O algoritmo de

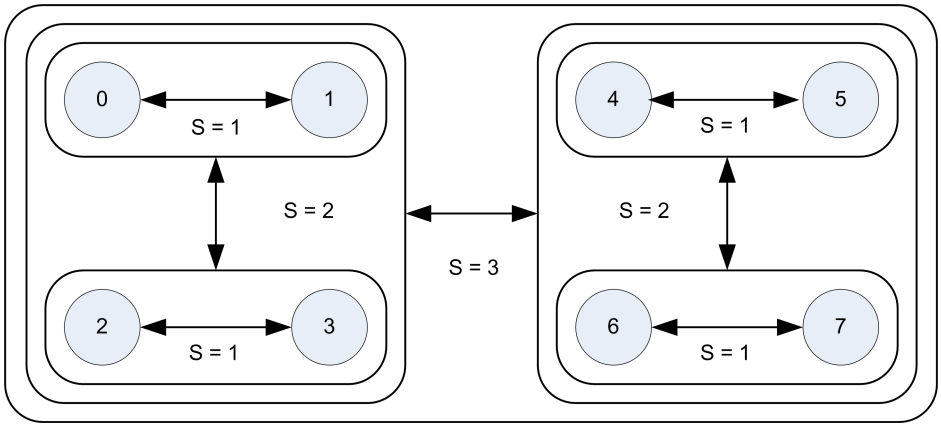


Figura 2. Sistema de 8 nodos agrupados em clusters.

diagnóstico utiliza o algoritmo DiVHA para atribuir os testes e determinar o fluxo de informação no sistema [1]. Na existência de nodos falhos testes adicionais são incluídos de forma a preservar o diâmetro logarítmico do hipercubo. O algoritmo DiVHA, da mesma forma que o algoritmo Hi-ADSD, utiliza o conceito de *clusters* para agrupar os nodos e organizar os testes. A figura 2 ilustra como o agrupamento em *clusters* é estruturado.

A construção de $T(S)$ pode ser dividida em duas fases. Na primeira fase $T(S)$ recebe todas as arestas correspondentes ao hipercubo dos nodos sem falha, de forma que cada nodo sem falha fica responsável por testar seus $log_2 N$ nodos vizinhos. A ocorrência de nodos falhos pode exigir a inclusão de arestas de teste adicionais. A inclusão de um teste adicional entre um par de nodo qualquer i e j baseia-se na distância no hipercubo entre este par de nodos bem como a atual distância dos mesmos no grafo de testes $T(S)$. A figura 3 mostra os assinalamentos de testes em um sistema de 8 nodos. Tome o nodo 7 como exemplo. O nodo 1 é escolhido como testador porque está a distância 2, enquanto 0 está a distância 3.

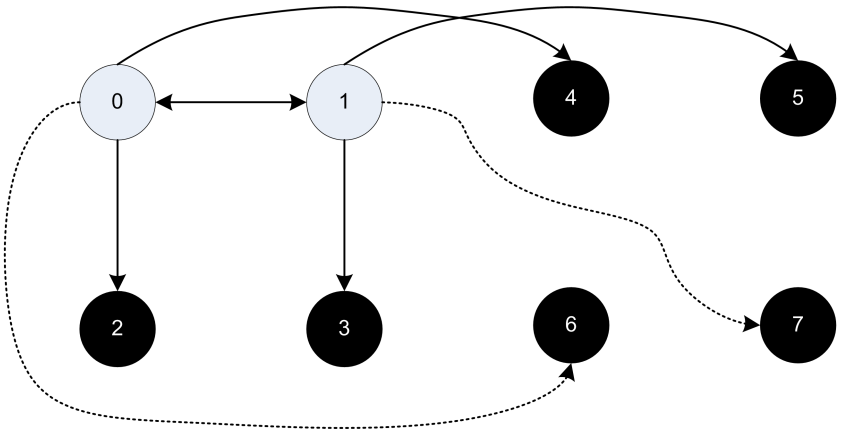


Figura 3. Assinalamento de testes do DiVHA; linha pontilhada indica teste adicional.

O DiVHA permite que o algoritmo de diagnóstico determine o estado de todos os nodos do sistema em no máximo $log_2 N$ rodadas de testes. O número máximo de testes

por rodada é de $N \times log_2 N$. A latência média do algoritmo DiVHA é, em geral, menor que sua latência máxima de $log_2 N$. Resultados experimentais indicam que a maior parte dos eventos são diagnosticados em cerca de $(log_2 N)/2$ rodadas [1].

3. MoDiVHA

O algoritmo MoDiVHA tem como função principal determinar quais serão os testes que cada um dos nodos devem executar. De forma similar ao algoritmo DiVHA, os nodos e os testes são representados por um grafo cuja a topologia é baseada em um hipercubo. No entanto, o grafo de testes produzido pelo MoDiVHA possui menos arestas que o equivalente gerado pelo DiVHA. A intuição é incluir no grafo de testes apenas as arestas estritamente necessárias, de forma a reduzir a redundância de caminhos entre os pares de nodos mas, ao mesmo tempo, garantir as propriedades que permitem que o diagnóstico do sistema seja realizado. Na sessão de resultados experimentais é realizado um comparativo da quantidade de testes necessários para diferentes tamanhos de sistema.

3.1. O Algoritmo MoDiVHA

A topologia do sistema segue a estruturada de um hipercubo e é representada pelo grafo $H(S)$, conforme a figura 4. Para contextualizar e exemplificar o funcionamento do algoritmo usaremos um sistema simples, composto por apenas 8 nodos. Os vértices de $H(S)$ correspondem aos nodos e as arestas direcionadas representam as conexões virtuais entre nodos, formando uma rede *overlay*. A distância mínima entre o nodo i e um nodo qualquer j é definida por $h(i, j)$, e é tamanho do menor caminho entre o nodo i e o nodo j em $H(S)$.

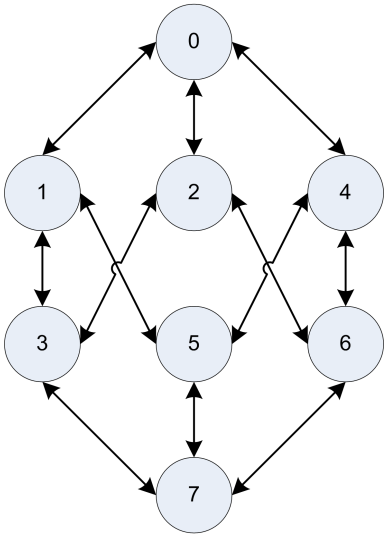


Figura 4. Grafo $H(S)$ para um sistema de 8 nodos.

O MoDiVHA faz uso de $H(S)$ para determinar os testes que devem ser realizados por todos os nodos sem-falha do sistema. O resultado desta operação é o grafo $T(S)$. Uma aresta direcionada de $T(S)$ entre, por exemplo, os nodos i e j indica que o nodo i testa o nodo j. Na ocorrência de um evento (falha ou recuperação de um nodo), todos os nodos do sistema devem recalcular $T(S)$ de forma a, novamente, determinar seus testes

considerando os testes realizados pelos demais nodos e pelas informações que consegue obter dos nodos testados. Na condição do sistema estar estabilizado $T(S)$ é igual para todos os nodos.

A construção do grafo $T(S)$ é realizada de forma a garantir que cada nodo falho seja testado por pelo menos um nodo sem-falha, e que a distância entre quaisquer nodos em $T(S)$ seja de no máximo $log_2 N$. A manutenção da distância garante a latência de diagnóstico. Desta forma uma aresta entre os nodos i e j é adicionada ao grafo $T(S)$ ou quando não existe um caminho entre i e j, ou quando o caminho que existe é maior que caminho mínimo esperado. O algoritmo verifica progressivamente o atendimento das distâncias mínimas entre os nodos. Inicialmente são verificados os nodos conectados por distância mínima 1, aumentando esta distância até alcançar $log_2 N$. Quando a verificação da distância mínima entre dois nodos falha, é inserida uma aresta ligando diretamente estes nodos.

A figura 5 mostra aspecto final de $T(S)$ para um sistema de 8 nodos, onde o nodo de índice 3 está falho. Os testes em linha contínua tem origem das conexões de $H(S)$. Neste exemplo o nodo falho compromete a condição do caminho mínimo, pois não existe rota de tamanho menor ou igual a $log_2 N$ entre os nodos 1 e 7. Desta forma testes adicionais devem ser incluídos, indicados pela linha tracejada.

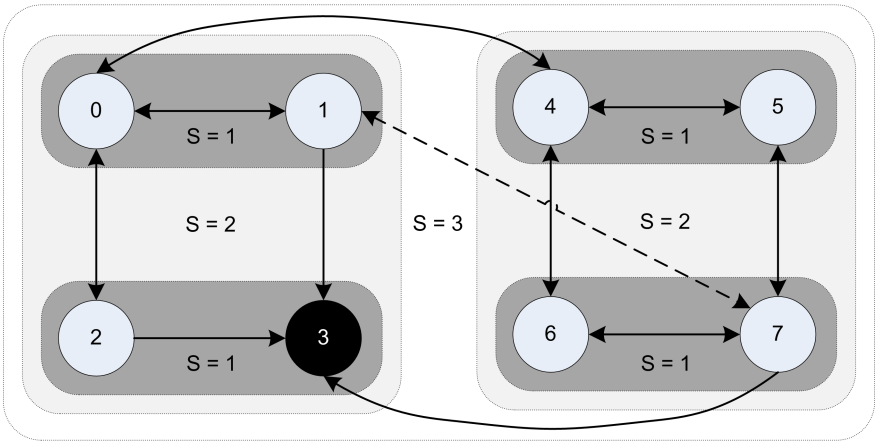


Figura 5. Grafo $T(S)$ representando o assinalamento de testes.

O MoDiVHA utiliza o conceito de *clusters* [9] para agrupar os nodos. O tamanho de um *cluster* é definido pelo índice s e determinado por uma potência de 2 (2^s). Um *cluster* de índice $s = 3$ tem tamanho igual a 8 (2^3) e é composto por dois *clusters* de índice 2 e, por conta da recursividade, contêm quatro *clusters* de índice 1. A figura 5 ilustra como esse agrupamento é realizado para um sistema de 8 nodos.

A lista de nodos de um determinado *clusters* é calculada pela função $C(i, s)$ [9] e pode ser definida recursivamente pela expressão: $C_{i,s} = C_{j,s-1} \cup C_{i,s-1}$, $j = i \oplus 2^{s-1}$ e $C_{i,1} = j$, $j = i \oplus 1$. A tabela da figura 6 lista os possíveis resultado da $C(i, s)$ para nosso exemplo de sistema.

O pseudo-código do MoDiVHA é apresentado na figura 7. O algoritmo começa sua operação reinicializando $T(S)$ para o estado no qual o número de arestas é 0, conforme indicado pela linha 2. Em seguida, nas linhas 3 a 6, são executados quatro *loops*

s	$C(0,s)$	$C(1,s)$	$C(2,s)$	$C(3,s)$	$C(4,s)$	$C(5,s)$	$C(6,s)$	$C(7,s)$
1	1	0	3	2	5	4	7	6
2	2,3	3,2	0,1	1,0	6,7	7,6	4,5	5,4
3	4,5,6,7	5,6,7,4	6,7,4,5	7,4,5,6	0,1,2,3	1,2,3,0	2,3,0,1	3,0,1,2

Figura 6. Resultado da função $C(i,s)$ para um sistema de 8 nodos.

em cascata, onde cada *loop* varre uma determinada propriedade do modelo do sistema. O laço mais externo percorre progressivamente todos os possíveis índices de *clusters* para o sistema S. O laço seguinte varre todas as possíveis distâncias d em $H(S)$ para o índice de *cluster* corrente.

```
1. MoDiVHA(system S)
2.    T(S) initially does not have any edge
3.    FOR s = 1 TO log2(N) DO
4.        FOR d = 1 TO s DO
5.            FOREACH fault free node i = [0 ... N-1] DO
6.                FOREACH node j that belongs C(i, s) DO
7.                    IF h(i, j) == d AND d(i, j) > s THEN
8.                        add edge(i, j) to T(s)
```

Figura 7. O algoritmo MoDiVHA.

O terceiro *loop* faz uma busca por todos os nodos i sem falha do sistema S e, finalmente, o último *loop* percorre os nodos j pertencentes a $C(i,s)$ atual. Desta forma são verificadas as arestas entre o nodo 0 e nodos pertencentes ao *cluster* $C(0,s)$, entre o nodo 1 e os nodos pertencentes ao *cluster* $C(1,s)$, e assim por diante até o nodo $N-1$ e os nodos pertencentes ao *cluster* $C(N-1,s)$.

A linha 7 verifica as condições necessária para inclusão de uma nova aresta em $T(S)$ e pode ser lida da seguinte maneira: ***SE** a distância de i para j no hipercubo for igual ao d corrente **E** a distância de i para j no grafo T for maior que o s corrente **ENTÃO***. A primeira condicional assegura que as arestas dos *clusters* de menor índice são verificadas antes que as arestas dos *clusters* de índice maior. A segunda condicional garante que uma aresta é adicionada entre o nodo i e o nodo $j \in C(i,s)$ sempre que <u>não</u> existir um caminho do nodo i para o nodo j em $T(S)$ com distância $\leq s$.

A função $d(i,j)$ retorna a distância entre os nodos i e j em $T(S)$ e corresponde ao tamanho do menor caminho entre o nodo i e o nodo j em $T(S)$. Se não existir um caminho entre um par de nodos a distância é tida como infinita.

Satisfeitas as condições a inclusão da aresta de i para j é realizada pela operação da linha 8 do pseudo-código. Esta nova aresta é um assinalamento que indica que nodo i deve testar no nodo j.

A redução de testes obtida por este algoritmo é consequência do fato que, diferentemente do DiVHA, o MoDiVHA submete, inclusive as arestas de $H(S)$ ao teste da existência de um caminho mínimo. Observe a figura 8, considere que o MoDiVHA está no exato momento onde verifica a necessidade de incluir uma aresta de teste entre os no-

dos 1 e 5. Pelo grafo $H(S)$ este teste deveria existir, mas como é possível encontrar um caminho entre 1 e 5 de tamanho menor ou igual a 3 ($s = 3$) o teste é ignorado. Neste caso o nodo 1 consegue obter informação sobre o nodo 5 pelo caminho $1 \rightarrow 0 \rightarrow 4 \rightarrow 5$.

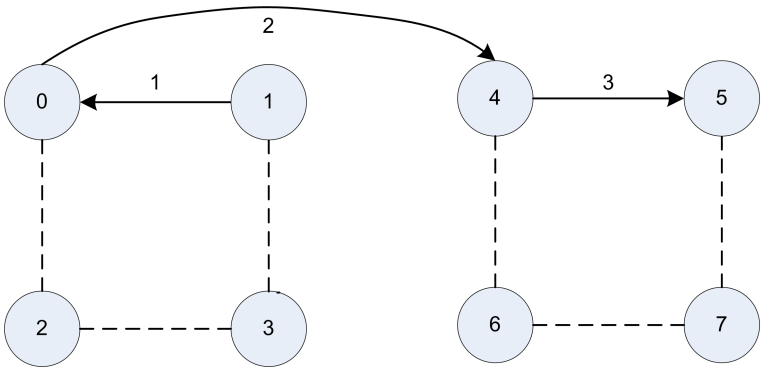


Figura 8. Construção do grafo $T(S)$.

Em um sistema onde todos os nodos estão sem falha o grafo $T(S)$ resultante do algoritmo DiVHA é um hipercubo completo, uma cópia exata de $H(S)$. O mesmo não se aplica ao MoDIVHA, conforme ilustrado pela figura 9.

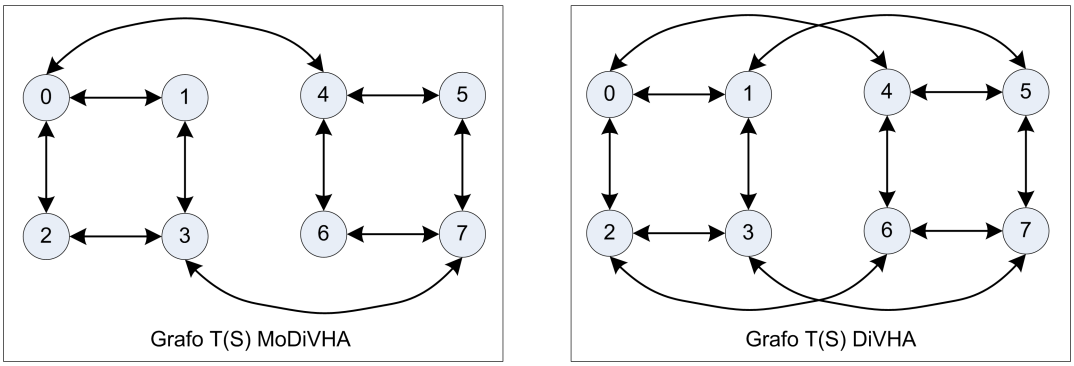


Figura 9. Grafos $T(S)$ gerados pelo MoDiVHA e DiVHA para um sistema de 8 nodos sem falhas.

4. Resultados Experimentais

Nesta seção são apresentados resultados obtidos de experimentos realizados com algoritmo MoDiVHA. A implementação do algoritmo foi realizada em linguagem de programação C e os experimentos foram conduzidos utilizando a biblioteca de simulação de eventos discretos SMPL (*Simple Portable Simulation Language*) [10].

Foram realizados três séries de simulações. A primeira simulação mostra o número de testes necessários sem considerar a ocorrência de eventos e/ou nodos falhos. O resultado deste experimento é então comparado com os dados obtidos de experimento similar baseado no algoritmo DiVHA. A segunda série de experimentados apresentados considera diversas configurações de falhas e computa o número de testes necessários para cada uma delas. Finalmente, o terceiro experimento tem como objetivo determinar a latência do MoDiVHA para uma série de situações de falhas geradas aleatoriamente.

4.1. Número de Testes Necessários

O propósito deste experimento é comparar o número de testes necessários em um sistema executando os algoritmos de assinalamento MoDiVHA e DiVHA. Foram simulados sistemas de 2 até 1024 nodos. A tabela da figura 10 reproduz os resultados obtidos.

N	Testes DiVHA	Testes MoDiVHA	Economia de Testes
2	2	2	0%
4	8	8	0%
8	24	20	17%
16	64	48	25%
32	160	102	36%
64	384	220	43%
128	896	458	49%
256	2048	944	54%
512	4608	1926	58%
1024	10240	3922	62%

Figura 10. Tabela comparativa; *DiVHA vs MoDiVHA*.

A primeira coluna da tabela indica o tamanho do sistema simulado, na segunda coluna está a quantidade de testes exigidos pelo DiVHA para cada tamanho de sistema. Em seguida a quantidade de testes referente ao MoDiVHA e, na última coluna, temos a economia de testes percentual do MoDiVHA em relação ao DiVHA.

Em sistemas pequenos, com poucos nodos, ambos algoritmos executam o mesmo número de testes. Conforme o tamanho do sistema aumenta o algoritmo MoDiVHA começa a apresentar uma redução significativa, e crescente, do número de testes. Em um sistema de 1024 nodos o algoritmo MoDiVHA executa 62% menos testes que o DiVHA. A figura 11 representa graficamente estes resultados.

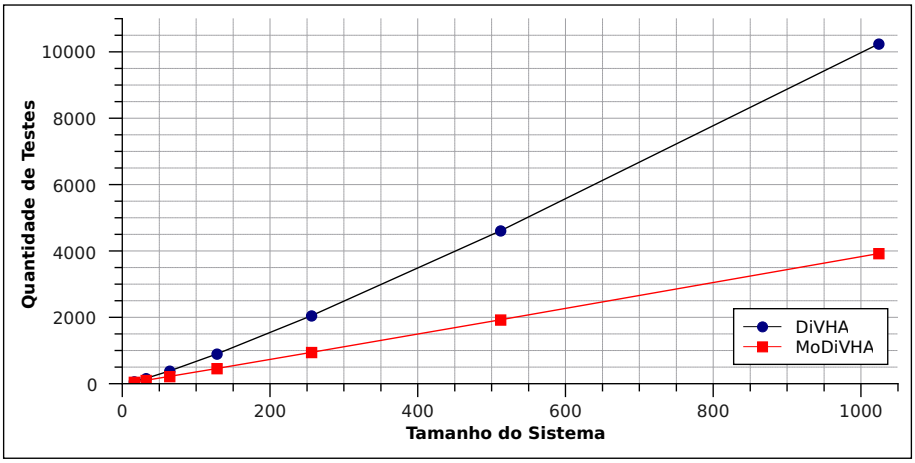


Figura 11. Número de testes necessários para diversos tamanhos de sistema.

4.2. Número de Testes Considerando Falhas

O segundo experimento considera um sistema de 256 nodos onde foram simuladas diversas configurações falhas. O propósito é medir o número de testes necessários na

ocorrência de 1 até $N - 1$ falhas em nodos aleatórios. O gráfico da figura 12 apresenta os resultados obtidos pela simulação. O eixo X representa a quantidade de nodos falhos e o eixo Y apresenta o número de testes que foram executados para cada situação de falha simulada. Estes resultados mostram a tendência da quantidade de testes diminuir conforme o número de nodos falhos aumenta.

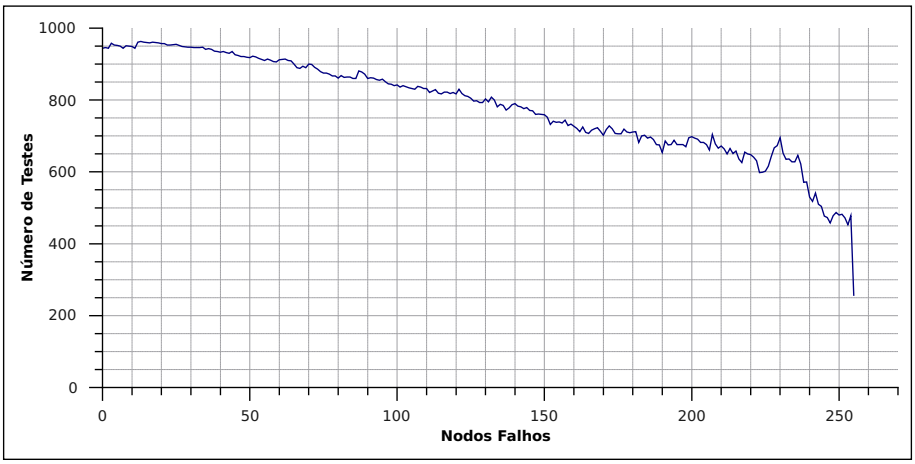


Figura 12. Número de testes *versus* falhas para um sistema com 256 nodos.

4.3. Latência para Detecção de Eventos

A latência é o número de rodadas de testes necessárias para que todos nodos sem-falha realizem o diagnostico de um evento. O objetivo do terceiro experimento é verificar a latência para detectar um evento sob diferente situações de falhas. O sistema simulado é composto por 128 nodos inicialmente sem-falhas. Foram introduzidos neste sistema falhas em nodos aleatórios. A latência de propagação destes eventos foram monitoras e registradas tanto para o algoritmo MoDIVHA quanto para o DiVHA, conforme exposto nas figuras 13 e 14 respectivamente.

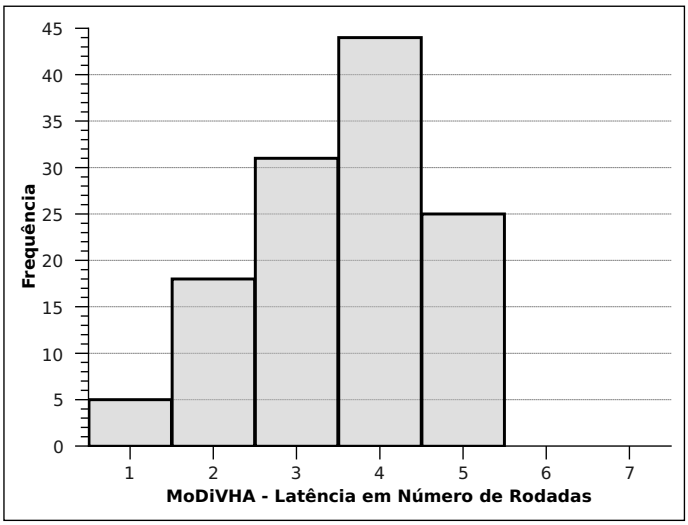


Figura 13. Latência MoDiVHA para um sistema com 128 nodos.

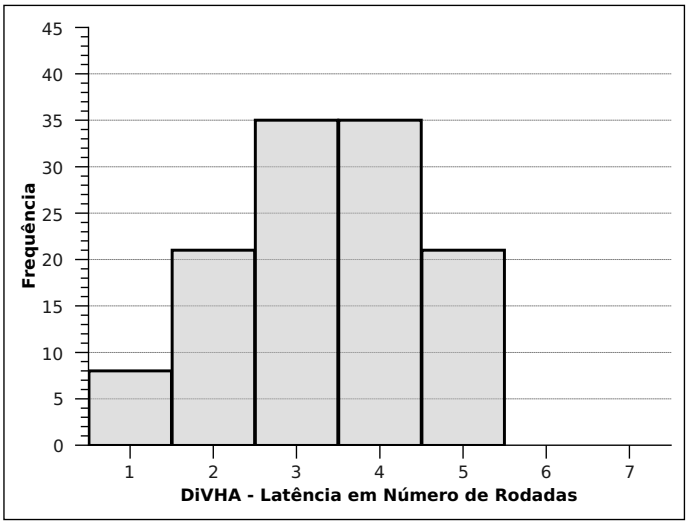


Figura 14. Latência DiVHA para um sistema com 128 nodos.

O histograma na figura 13 mostra a distribuição de frequência da latência observada para os eventos simulados usando o MoDiVHA. A maioria dos eventos foi diagnosticada em um número de rodadas abaixo da latência máxima teórica de $log_2 N$ rodadas, mas, na média, acima dos valores observados para o algoritmos DiVHA (figura 14). Este comportamento já era esperado e é consequência direta da redução de arestas no grafo de testes $T(S)$. As arestas de $T(S)$, além de indicarem o assinalamento dos testes, também são usadas para o determinar a rota de troca de informações entre os nodos do sistema.

5. Conclusões

Este trabalho apresentou uma estratégia de assinalamento de testes hierárquica que permite o diagnóstico escalável de sistemas distribuídos. Esta nova estratégia foi denominada MoDiVHA sendo baseada naquela do algoritmo DiVHA [1], mas visa, principalmente, a redução dos recursos de sistema necessários para a realização do diagnóstico. Os dados experimentais demostram que quanto maior o sistema maior será a redução do número de teste necessários pelo MoDiVHA em relação ao DiVHA. Este resultado evidencia o fato que o MoDiVHA é escalável, tornando-o uma opção para aplicações de diagnóstico distribuído quando é necessário reduzir o número de testes.

Trabalhos futuros incluem a aplicação da estratégia distribuída hierárquica de testes MoDiVHA em redes *Ad hoc*, que implicará em mudanças inclusive no modelo de sistema considerado [11]. E está sendo considerada a aplicação do MoDiVHA para testes hierárquicos em serviços distribuídos na Internet [12]; neste caso um dos desafios é adaptar o modelo às características do sistema.

Referências

[1] Luis C. E. Bona, Elias P. Duarte Jr., Keiko V. O. Fonseca, Samuel L. V. Mello, "HyperBone: A Scalable Overlay Network Based on a Virtual Hypercube", *The 8th International Symposium on Cluster Computing and the Grid (CCGRID'2008)*, pp. 58-64, Lyon, France, 2008.

[2] N. Blum, P. Jacak, F. Schreiner, D. Vingarzan, and P. Weik, "Towards Standardized and Automated Fault Management and Service Provisioning for NGNs", *Journal of Network and Systems Management*, Vol. 16, No. 1, 2008.

[3] Hui Yang, Mourad Elhadef, Amiya Nayak, Xiaofan Yang, "An evolutionary approach to system-level fault diagnosis", *IEEE Congress on Evolutionary Computation*, 2009.

[4] Elias P. Duarte Jr., Luiz C. P. Albini, Alessandro Brawerman, Andre L. P. Guedes, "A Hierarchical Distributed Fault Diagnosis Algorithm Based on Clusters with Detours", *The 6th IEEE Latin American Network Operations and Management Symposium (LANOMS)*, pp. 1-6, Punta del Este, Uruguay, 2009.

[5] S.-Y. Hsieh and Y.-S. Chen, "Strongly Diagnosable Systems Under the Comparison Model", *IEEE Transactions on Computers*, Vol. 57, No. 12, 2008.

[6] X. Yang, and Y. Y. Tang, "Efficient Fault Identification of Diagnosable Systems under the Comparison Model", *IEEE Transactions on Computers*, Vol. 56, No. 12, 2007.

[7] Elias P. Duarte Jr., Roverli P. Ziwich, Luiz C. P. Albini, "A Survey of Comparison-Based System-Level Diagnosis", *ACM Computing Surveys*, ISSN 0360-0300, Aceito para publicação.

[8] R.P. Bianchini, and R. Buskens, "An Adaptive Distributed System-Level Diagnosis Algorithm and Its Implementation", *Proc. FTCS-21*, pp. 222-229, 1991.

[9] E. P. Duarte Jr., and T. Nanya, "A Hierarchical Adaptive Distributed System-Level Diagnosis Algorithm", *IEEE Transactions on Computers*, Vol. 47, pp. 34-45, No. 1, Jan 1998.

[10] M.H. MacDougall, "Simulating Computer Systems: Techniques and Tools", *The MIT Press*, Cambridge, MA, 1987.

[11] Andréa Weber, Alexander Robert Kutzke, Stefano Chessa, "Energy-Aware Test Connection Assignment for the Diagnosis of a Wireless Sensor Network", *The 5th IEEE Latin American Dependable Computing Symposium (LADC)*, São José dos Campos, SP, 2011.

[12] Alan Nakai, Edmundo Madeira, Luiz Eduardo Buzato, "Load Balancing for Internet Distributed Services using Remote Resource Reservation", *The 5th IEEE Latin American Dependable Computing Symposium (LADC)*, São José dos Campos, SP, 2011.

Estabelecendo Sequências de Teste de Integração de Classes: Um Estudo Comparativo da Aplicação de Três Algoritmos Evolutivos Multiobjetivos

Wesley Klewerton Guez Assunção[1], **Thelma Elita Colanzi**[1 2], **Silvia Regina Vergilio**[1], **Aurora Trinidad Ramirez Pozo**[1]

[1]Departamento de Informática, Programa de Pós-Graduação em Informática, Universidade Federal do Paraná (UFPR)– Curitiba, PR – Brasil

[2]Departamento de Informática
Universidade Estadual de Maringá(UEM) – Maringá, PR – Brasil

{wesleyk, thelmae, silvia, aurora}@inf.ufpr.br

Resumo. *Com o intuito de minimizar os custos associados à criação de stubs, durante o teste de integração de sistemas orientados a objetos é preciso determinar a melhor ordem de integração e teste das classes. Em alguns estudos recentes, Algoritmos Evolutivos Multiobjetivos (MOEA) têm resolvido eficientemente este problema porque eles consideram vários fatores que afetam o processo de construção de stubs. Geralmente esses fatores são conflitantes e por isto, nem sempre é possível obter uma solução (ordem) única. Dentre os trabalhos publicados, os MOEAs NSGA-II e SPEA2 alcançaram bons resultados para o problema em questão. No entanto, existem outros MOEAs que não foram aplicados neste contexto e que podem melhorar o desempenho desses dois algoritmos. Com este objetivo, o presente trabalho explora o uso do algoritmo PAES neste contexto. Foi realizado um experimento com quatro sistemas reais no qual o PAES foi comparado com o NSGA-II e o SPEA2. Os resultados mostram que ele consegue superar os outros dois MOEAs no caso do sistema mais complexo.*

Abstract. *During the integration testing of object-oriented software it is necessary to determine the best class integration and test order (CITO) aiming at minimizing the costs of stubs creation. Recently, Multi-Objective Evolutionary Algorithms (MOEA) have efficiently solved this problem because they consider several factors which affect the stub creation. These factors are usually in conflict, so it is not possible to find a single solution (order). Related works showed the MOEAs NSGA-II and SPEA2 have achieved good solutions for CITO problem. Although, there are other MOEAs that were not applied in this context and they can improve the performance of NSGA-II and SPEA2. In this sense, this work explores the use of the MOEA PAES in the CITO problem. It was performed an experiment involving four real systems and PAES was compared with NSGA-II and SPEA2. The results show that PAES overcomes the other two algorithms regarding to the more complex system.*

1. Introdução

O teste de software é considerado uma atividade fundamental para o desenvolvimento de software de qualidade. Geralmente esta atividade de teste é conduzida em diferentes fases. Durante a fase de teste de integração de software orientado a objetos, é comum que

algumas classes necessitem de recursos de outras classes que ainda estão em desenvolvimento. Isso leva à necessidade de construção de *stubs* para simular recursos necessários mas não disponíveis e indispensáveis para o teste, aumentando a complexidade e o custo desta atividade.

A construção de *stubs* é um processo que aumenta o custo do teste e por isto é desejável minimizar o número de *stubs* a serem construídos. O número de *stubs* está relacionado à ordem utilizada para teste e integração das classes. Determinar a melhor ordem é um problema conhecido como CITO (*Class Integration and Test Order*) [Abdurazik and Offutt 2006], e devido à existência de ciclos de dependência entre classes, bastante comum em linguagens tais como Java [Melton and Tempero 2007], determinar essa sequência não é trivial.

A maioria dos trabalhos que visam a resolver este problema propõe estratégias baseadas em grafos [Kung et al. 1995a, Kung et al. 1995b, Tai and Daniels 1997, Traon et al. 2000, Briand and Labiche 2003], entretanto estas estratégias não consideram fatores específicos da construção de *stubs* e, em geral, encontram apenas soluções subótimas. Para contornar estas limitações, estratégias baseadas em algoritmos genéticos (AG) são mais promissoras [Briand et al. 2002a, Briand et al. 2002b]. Elas permitem o uso de diferentes medidas de acoplamento para medir a complexidade e custo dos *stubs*, tais como acoplamento de atributos e métodos. Entretanto, os melhores resultados foram alcançados com algoritmos multiobjetivo [Cabral et al. 2010], os quais apresentam uma variedade de boas soluções quando comparadas às estratégias baseadas em AG com uma agregação de funções para cálculo de *fitness*.

Um estudo recente [Pozo et al. 2011] mostrou que o algoritmo NSGA-II (*Non-dominated Sorting Genetic Algorithm*) [Deb et al. 2002], que é um algoritmo evolutivo multiobjetivo (MOEA - *Multi-Objective Evolutionary Algorithm*), se adapta melhor do que outras meta-heurísticas, tais como busca tabu e colônia de formigas para solucionar o problema CITO. Motivados por este resultado, Assunção et al [Assunção et al. 2011a, Assunção et al. 2011b] realizaram um experimento aplicando o NSGA-II juntamente com um outro algoritmo evolutivo multiobjetivo, o SPEA2 (*Strength Pareto Evolutionary Algorithm*) [Zitzler et al. 2001], obtendo resultados que atestam a eficácia da aplicação de ambos MOEAs. Dando continuidade aos trabalhos de Assunção et al e buscando melhorar ainda mais os resultados, este trabalho descreve um experimento de aplicação ao problema CITO de um outro MOEA, o algoritmo PAES (*Pareto Archived Evolution Strategy*) [Knowles and Corne 2000]. O PAES é um algoritmo multiobjetivo que usa uma estratégia de evolução um tanto diferente da estratégia adotada pelos outros dois algoritmos utilizados anteriormente. Saber qual algoritmo se comporta melhor para o problema é uma questão que só pode ser respondida através de experimentos. Por isto, este trabalho descreve os resultados de aplicação dos três MOEAs a quatro sistemas reais. O custo dos *stubs* é medido em relação a dois objetivos a serem minimizados, avaliados segundo duas medidas de acoplamento comumente usadas: número de métodos e número de atributos.

Este artigo está organizado como segue. Na Seção 2 é dado o contexto de otimização e algoritmos multiobjetivos utilizados. Na Seção 3 descreve-se como foi realizada a aplicação dos MOEA ao problema CITO. Na Seção 4 descreve-se o estudo experimental realizado cujos resultados são apresentados e analisados na Seção 5. Na Seção 6 apresenta-se um exemplo de uso das soluções geradas pelos MOEAs para o problema em

questão. Finalmente, na Seção 7 são apresentadas as conclusões do trabalho.

2. Algoritmos multiobjetivos

Problemas de otimização com duas ou mais funções objetivo são chamados de multiobjetivo. Neste tipo de problema os objetivos costumam estar em conflito e, por isso, não há uma solução única. Assim, para solucionar este tipo de problema, procura-se encontrar soluções que melhor representem o compromisso entre os objetivos. Estas soluções são chamadas de não-dominadas e formam a fronteira de Pareto [Pareto 1927]. Em muitas aplicações a busca pela fronteira de Pareto ótima é NP-difícil [Knowles et al. 2006], então o problema de otimização busca encontrar a aproximação (PFApprox) mais próxima possível da fronteira de Pareto.

Algoritmos Evolutivos Multiobjetivos têm sido usados cada vez mais para resolver problemas de otimização multiobjetivo. MOEAs são adequados para esta tarefa porque eles evoluem uma população de potenciais soluções simultaneamente, obtendo um conjunto de soluções próximas à fronteira de Pareto em uma única execução do algoritmo. Existem alguns variantes dos AGs tradicionais adaptados para problemas multiobjetivo. Cada um adota estratégias distintas para evoluir e diversificar as soluções. Como mencionado, neste trabalho os seguintes algoritmos foram avaliados: NSGA-II [Deb et al. 2002], SPEA2 [Zitzler et al. 2001] e PAES [Knowles and Corne 2000].

O NSGA-II (*Non-dominated Sorting Genetic Algorithm*) [Deb et al. 2002] ordena a população em várias fronteiras não-dominadas. A cada iteração ele utiliza as soluções não dominadas formando uma nova fronteira. Para cada solução i determina-se o número de soluções que não dominam i e o conjunto de soluções que i domina. As fronteiras representam a estratégia de elitismo adotada pelo NSGA-II. Além disso, este algoritmo usa um operador de diversidade (*crowding distance*) que ordena os indivíduos de acordo com a sua distância em relação aos vizinhos da fronteira para cada objetivo, visando garantir maior espalhamento das soluções.

O algoritmo SPEA2 (*Strength Pareto Evolutionary Algorithm*) [Zitzler et al. 2001], além de ter sua população regular, mantém um arquivo externo que armazena soluções não-dominadas encontradas a cada geração. Para cada solução que está no arquivo e na população é calculado um valor de *strength*, que é usado no cálculo do *fitness* do indivíduo e durante o processo de seleção. O valor de *strength* de uma solução i corresponde a quantidade de indivíduos j, pertencentes ao arquivo e a população, dominados por i. O tamanho do arquivo é fixo, então, caso este tamanho seja excedido, um algoritmo de agrupamento é utilizado a fim de reduzi-lo [Coello et al. 2006].

No processo evolutivo do algoritmo PAES (*Pareto Archived Evolution Strategy*) [Knowles and Corne 2000] o conceito de população é diferente das estratégias tradicionais de AGs, pois apenas uma solução é mantida em cada geração, e a criação de novos indivíduos é feita somente através da aplicação do operador de mutação. Uma vez que o algoritmo trabalha com apenas uma solução por geração, não existe motivo para a utilização do operador de cruzamento. Assim como no SPEA2, existe um arquivo externo de soluções que é populado com as soluções não dominadas encontradas durante o processo. A cada geração, o PAES cria uma nova solução filha que é comparada com a solução pai, se a solução filha domina a solução pai, a filha toma o lugar da pai e a filha

é acrescentada ao arquivo externo; se a solução filha é dominada pela pai ou por alguma solução do arquivo externo, a filha é descartada; e caso nenhuma das soluções domine a outra, a escolha da solução que vai permanecer no processo evolutivo é feita considerando a diversidade entre pai, filha e as soluções do conjunto externo. Caso o tamanho do arquivo externo exceda, é aplicada uma estratégia de diversidade (*crowded region*) sobre este conjunto de soluções, eliminando soluções similares e mantendo a exploração de um espaço de busca maior.

Portanto, apesar de se tratar de três algoritmos evolutivos a estratégia de evolução do PAES é bastante diferente das outras duas, o que pode levar a resultados diferentes na resolução do problema CITO.

3. MOEAs aplicados ao problema CITO

Para aplicar os MOEAs a fim de resolver o problema CITO é preciso encontrar uma boa representação para o problema. E esta representação influencia na implementação de todos os estágios do algoritmo. Como o problema CITO é relacionado a permutação de classes, as quais formam as ordens de teste, o cromossomo é representado por um vetor cujas posições contém inteiros que representam as classes. O tamanho do vetor é igual ao número de classes de cada sistema. Então, se cada classe é representada por um inteiro, uma solução válida para um problema de 8 classes seria $(3, 5, 4, 2, 1, 7, 6, 8)$. Neste exemplo, a primeira classe a ser testada e integrada seria a classe representada pelo número 3. Esta representação é a mesma utilizada em [Cabral et al. 2010, Assunção et al. 2011a].

Outra questão relacionada aos MOEAs é a escolha das funções objetivo. Como mencionado anteriormente, podem ser usados no problema CITO várias medidas e fatores como acoplamento, coesão e restrições de tempo. No experimento realizado no presente trabalho foram usadas duas funções baseadas nas duas medidas de acoplamento usadas na maioria dos trabalhos relacionados encontrados na literatura [Briand et al. 2002a, Briand et al. 2002b, Cabral et al. 2010]: acoplamento de atributos e de métodos.

Tais medidas foram definidas para medir as dependências entre classes acopladas em termos de número de atributos usados e número de métodos chamados que influenciam diretamente na complexidade de criação de *stubs*. Então, considerando-se que c_i e c_j são duas classes acopladas, as medidas de acoplamento são definidas como segue:

- Acoplamento de atributos (A) = Número de atributos localmente declarados em c_j quando referências ou ponteiros para uma instância de c_j aparecem na lista de argumentos de alguns métodos de c_i, como o tipo de seu valor de retorno, na lista de atributos de c_i, ou como parâmetros locais de métodos de c_i (adaptado de [Briand et al. 2002b]). Assim como no experimento realizado por [Briand et al. 2002b], no caso de herança o número de atributos herdados das classes ancestrais não foi contado.
- Acoplamento de Métodos (M) = Número de métodos (inclusive construtores) localmente declarados em c_j que são invocados por métodos de c_i (adaptado de [Briand et al. 2002b]). No caso de herança, também são contados os métodos invocados herdados das classes ancestrais.

Os dados de entrada de cada MOEA são formados por três matrizes: (i) matriz de dependência entre as classes, (ii) matriz de acoplamento de atributos (medida A) e

(iii) matriz de acoplamento de métodos (medida M). A matriz de dependência popula um vetor de duas dimensões onde linhas e colunas correspondem a identificadores das classes, na intersecção das linhas e colunas das classes dependentes é informado o tipo de dependência, sendo que a classe identificada da linha depende da classe identificada na coluna. Com base nessa matriz são definidas as restrições para cada um dos algoritmos. A restrição que deve ser considerada durante o processamento do algoritmo é não quebrar as dependências relacionadas a herança[1].

As métricas de acoplamento A e M são usadas para calcular o *fitness* de cada solução, sendo que a soma das dependências entre as classes de cada métrica corresponde a um objetivo. Durante o processo de otimização, busca-se minimizar todos os objetivos.

4. Descrição do Estudo Experimental

O estudo experimental, descrito nesta seção, envolve a aplicação dos três MOEAs descritos anteriormente, que foram implementados usando as versões disponíveis no *framework* JMetal [Durillo et al. 2010].

4.1. Sistemas Utilizados

Foram utilizados quatro sistemas reais desenvolvidos em Java na realização do estudo experimental: MyBatis, JHotDraw, BCEL e JBoss. MyBatis é um *framework* de mapeamento de classes de aplicações orientadas a objetos em base de dados relacionais. JHotDraw é um *framework* de gráficos bidimensionais para editores de desenho escritos em Java. JBoss AS é um servidor de aplicações Java. O último sistema é BCEL (*Byte Code Engineering Library*), uma biblioteca que possibilita aos seus usuários analisar, criar e manipular binários Java.

Considerando a dificuldade em obter documentação arquitetural de sistemas complexos para usar no experimento, optou-se por desenvolver um *parser* para realizar a engenharia reversa do código dos software a fim de identificar os tipos de relacionamentos existentes entre as classes. O *parser* desenvolvido teve como base o software AJATO[2] (*AspectJ and Java Assessment Tool*). Na Tabela 1 são apresentadas a versão e a quantidade de classes, dependências e linhas de código (LOC) de cada sistema.

Tabela 1. Sistemas utilizados no estudo experimental

Software	Versão	Classes	Dependências	LOC	Disponível em:
BCEL	5.0	45	289	2999	http://archive.apache.org/dist/jakarta/bcel/old/v5.0/
JBoss	6.0.0M5	150	367	8434	http://www.jboss.org/jbossas/downloads.html
JHotDraw	7.5.1	197	809	20273	http://sourceforge.net/projects/jhotdraw/
MyBatis	3.0.2	331	1271	23535	http://code.google.com/p/mybatis/downloads/list

4.2. Configuração dos parâmetros dos algoritmos

Neste experimento foram adotados os mesmos valores de parâmetros utilizados em [Assunção et al. 2011a] para o NSGA-II e o SPEA2. Como mencionado na Seção 2 o algoritmo PAES não recebe parâmetro de tamanho de população e não usa operador

[1]Diferente do trabalho de Briand et al [Briand et al. 2002b] neste trabalho dependências relacionadas a Agregação não são consideradas restrições.

[2]AJATO disponível em: http://www.teccomm.les.inf.puc-rio.br/emagno/ajato/

de cruzamento em seu processo evolutivo. Após validações empíricas, decidiu-se por adotar como taxa de mutação para o PAES o valor 1,0. No caso da taxa de mutação não era interessante adotar o mesmo valor utilizado para o NSGA-II e o SPEA2 em [Assunção et al. 2011a], já que estes dois algoritmos utilizam dois operadores evolutivos (cruzamento e mutação) e o PAES utiliza somente o operador de mutação. O número de avaliações de *fitness* foi utilizado como critério de parada e foi estabelecido em 20.000, mesmo número utilizado no experimento descrito em [Assunção et al. 2011a]. A Tabela 2 contém os valores dos parâmetros utilizados para cada um dos três algoritmos.

Tabela 2. Parâmetros para os MOEAs

Parâmetro	NSGA-II	SPEA2	PAES
Tamanho da população	300	300	-
Número de avaliações de *fitness*	20000	20000	20000
Taxa de mutação	0,02	0,02	1,00
Taxa de cruzamento	0,95	0,95	-
Tamanho do arquivo	-	250	250

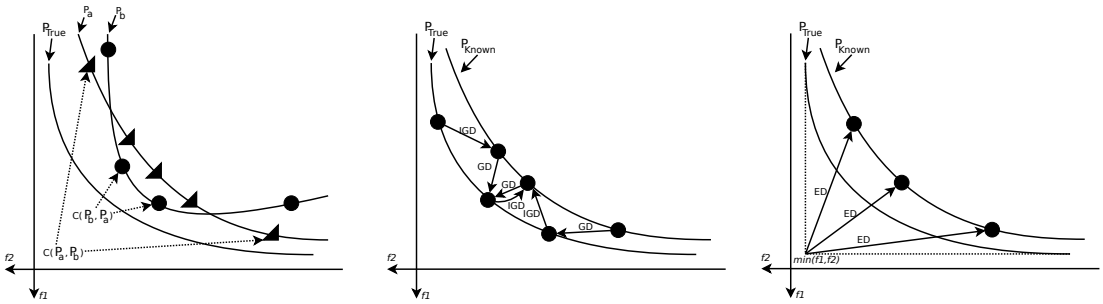
4.3. Indicadores de Qualidade

Para análise e comparação dos resultados alcançados pelos MOEAs utilizaram-se quatro indicadores de qualidade: Cobertura, Distância Geracional (GD), Distância Geracional Inversa (IGD) e Distância Euclidiana a Solução Ideal. Como os indicadores utilizam um conjunto formado pelas melhores soluções conhecidas para o problema (P_{true}), este foi obtido para cada sistema por meio da união dos conjuntos de todas as soluções encontradas por cada MOEA, eliminando-se as soluções dominadas e repetidas.

O indicador de qualidade Cobertura (C) [Knowles and Corne 2000] é usado para medir a dominância entre dois conjuntos. Comparando-se C(P_a, P_b) obtém-se um valor entre 0 e 1 referente a quanto o conjunto P_b é dominado pelo conjunto P_a, por outro lado, analisa-se C(P_b, P_a) para obter o quanto P_a é dominado por P_b. A Figura 1(a) apresenta um exemplo desse indicador, para um problema de minimização com dois objetivos. No primeiro caso analisa-se C(P_a, P_b) que retorna o valor 0.5, pois o conjunto P_b possui dois de seus quatro elementos dominados pelo conjunto P_a. No segundo caso analisa-se C(P_b, P_a) que retorna o valor 0.6 já que três dos quatro elementos do conjunto P_a são dominados pelo conjunto P_b.

O indicador GD [van Veldhuizen and Lamont 1999] é usado para calcular a distância de um conjunto de Pareto encontrado (P_{known}) em relação ao conjunto P_{true}, ou seja, é uma medida de erro na qual verifica-se o quão distante está um ponto encontrado do seu correspondente mais próximo no conjunto P_{true}. Já o indicador IGD [Radziukyniene and Zilinskas 2008] observa o inverso da GD, pois tem o objetivo de calcular a distância do conjunto P_{true} em relação ao conjunto P_{known}. Tanto para GD quanto para IGD deseja-se obter valores próximos a 0. A Figura 1(b) apresenta um exemplo das métricas de GD e IGD.

O indicador Distância Euclidiana da Solução Ideal (ED) tem como objetivo encontrar a solução que mais se aproxima dos objetivos ótimos, e consequentemente identificar o algoritmo que mais se aproximou da melhor solução possível para um determinado problema. Assim, este indicador consiste em determinar os melhores valores para cada um



(a) Cobertura (b) GD e IGD (c) Distância Euclidiana

Figura 1. Exemplos dos Indicadores de Qualidade

dos objetivos entre todas as soluções de P_{true}, encontrando um ponto considerado como uma solução ideal para o problema, então calcula-se a partir deste ponto a distância euclidiana em relação a todas as soluções de P_{known} (Figura 1(c)). Este indicador tem como base a técnica de classificação *Compromise Programming* [Cochrane and Zeleny 1973] que é usada em otimização multiobjetivo como técnica de apoio ao tomador de decisão para selecionar uma solução entre as várias encontradas.

5. Resultados e Análise dos Algoritmos

Após a execução dos três MOEAs, os resultados permitem estimar a complexidade do problema CITO para cada sistema. A Tabela 3 apresenta alguns resultados do experimento como a cardinalidade da PFApprox após todas as 30 execuções dos MOEAs (segunda coluna) e, na terceira, quarta e quinta colunas, a cardinalidade média de PFApprox e, entre parênteses, o total de diferentes soluções de PFApprox retornadas por cada MOEA. Corroborando as evidências detectadas em [Assunção et al. 2011a], (i) JBoss e BCEL se mostraram mais simples, uma vez que os três algoritmos encontraram todas as soluções em PFApprox em quase todas as execuções, e (ii) JHotDraw e Mybatis são mais complexos pois os algoritmos encontraram algumas das soluções em PFApprox em cada execução. MyBatis é ainda mais complexo do que o JHotDraw pois tem uma PFApprox com maior cardinalidade. Neste caso, o NSGA-II encontrou um maior número de diferentes soluções (53) seguido pelo PAES (49).

Tabela 3. Número de soluções não dominadas

Sistema	# PFApprox	NSGA-II	SPEA2	PAES
BCEL	29	28,70 (29)	28,57 (29)	25,13 (29)
JBoss	1	1,00 (1)	1,00 (1)	1,00 (1)
JHotDraw	3	1,80 (3)	1,73 (3)	1,97 (3)
MyBatis	49	42,53 (53)	39,97 (44)	38,47 (49)

Os três algoritmos encontraram uma única e igual solução para o sistema JBoss com os seguintes valores para cada objetivo: A = 10, M = 6. Então, esta solução envolve criar *stubs* para simular dependências de 10 atributos e 6 métodos. Apesar de alcançar uma solução final ótima, as soluções iniciais geradas pelos algoritmos multiobjetivos não são ótimas. Inicialmente os valores de *fitness* são altos e ao longo das gerações eles conseguem baixar o *fitness* até alcançar a solução supracitada.

O número de classes, dependências e ciclos influencia a cardinalidade da PFApprox obtida pelos algoritmos. Por exemplo, apesar de o JBoss ter mais classes e dependências do que o BCEL, os três MOEAs alcançaram uma única solução para ele, enquanto para o BCEL foram alcançadas 29 soluções. Portanto, o número de ciclos tem a maior influência sobre o número de soluções encontradas, já que o JBoss tem 8 ciclos e o BCEL tem 416.091 ciclos. Mas é possível notar que os dois sistemas que têm mais dependências, JHotDraw e Mybatis, são mais complexos para serem resolvidos. Entretanto, a respeito do JHotDraw, parece que as métricas não são altamente interdependentes e conflitantes já que o número de soluções alcançadas pelos MOEAs é menor, talvez porque os métodos do JHotDraw sejam menos acoplados ou mais simples que os do MyBatis.

Quando os MOEAs são usados, o testador pode usufruir da variedade de soluções encontradas de acordo com suas preferências e necessidades. O testador pode conduzir o teste de integração priorizando alguma das métricas de acoplamento e a decisão sobre a priorização de objetivos pode ser influenciada por diferentes fatores, tais como restrições do negócio, econômicas ou contratuais.

Nas próximas subseções são apresentados resultados obtidos para cada um dos quatro indicadores de qualidade analisados, além da discussão desses resultados. Para os indicadores GD, IGD e ED, não se analisam os resultados para o sistema JBoss, uma vez que os MOEAs obtiveram uma única solução para este sistema. Após obter os resultados dos indicadores GD e IGD, foi executado o teste de Friedman [Gárcia et al. 2009], através do software R, de modo a verificar se os MOEAs são considerados estatisticamente iguais ($p - value$ maior que o nível de significância α = 0,05). O teste de Friedman pode ser usado para comparar os resultados de três ou mais amostras relacionadas. Para o indicador C foi executado o teste pareado de Wilcoxon [Gárcia et al. 2009], também através do software R, de modo a verificar se os algoritmos são considerados estatisticamente diferentes ($p - value <= 0,05$).

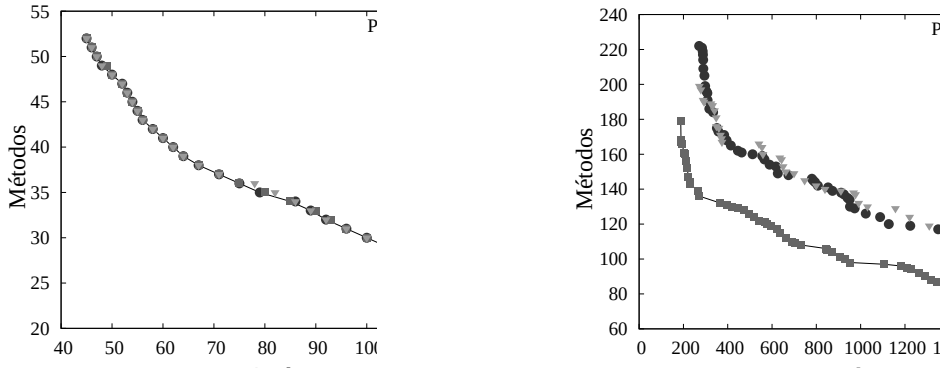
5.1. Cobertura

A Tabela 4 apresenta os valores do indicador C do conjunto final de soluções encontradas por cada algoritmo. A leitura é feita por linha-coluna, portanto o MOEA de determinada linha exerce uma determinada dominância sobre o MOEA da coluna. Para o sistema BCEL os três algoritmos são equivalentes, já para os sistemas JHotDraw e MyBatis o PAES domina em 100% tanto o NSGA-II como o SPEA2. Além disso, nestes dois sistemas o NSGA-II domina o SPEA2 com mais de 65% de dominância. Esses resultados são confirmados ao analisar a distribuição das soluções alcançadas pelos algoritmos no espaço de busca. Na Figura 2(a) fica evidente a equivalência entre os MOEAs no caso do BCEL; e, na Figura 2(b) é perceptível que o PAES alcança soluções que dominam as soluções dos outros dois MOEAs, e que o NSGA-II têm várias soluções que dominam soluções do SPEA2.

A Tabela 5 apresenta os valores de $p - values$ da comparação do indicador de C entre as 30 rodadas de cada MOEA utilizando o teste de Wilcoxon. Aparecem destacados em negrito os valores significativos, menores que 0,05, da comparação par a par entre os algoritmos. Portanto, com base neste teste estatístico, pode-se concluir que para o sistema JHotDraw os três algoritmos são equivalentes já que os $p - values$ não apontam diferença estatística entre eles. Por outro lado, no caso dos sistemas BCEL e MyBatis há diferença

Tabela 4. Análise de cobertura dos algoritmos

Sistema	MOEA	NSGA-II	SPEA2	PAES
BCEL	NSGA-II	-	0,0689655	0,275862
	SPEA2	0	-	0,241379
	PAES	0,0344828	0,103448	-
JHotDraw	NSGA-II	-	0,666667	0
	SPEA2	0	-	0
	PAES	1	1	-
MyBatis	NSGA-II	-	0,727273	0
	SPEA2	0,301887	-	0
	PAES	1	1	-



(a) BCEL (b) MyBatis

Figura 2. Soluções no espaço de busca

significativa entre os algoritmos. A análise dos *boxplots* (Figura 3) permite identificar qual deles é o melhor. Como o indicador C analisa a cobertura de um conjunto sobre o outro, então o algoritmo que tiver um maior valor de cobertura pode ser considerado melhor. No caso do BCEL, o NSGA-II e o SPEA2 superam o PAES neste indicador (Figuras 3(a) e 3(b)). Em relação ao MyBatis, nas Figuras 3(c) e 3(d) é perceptível a dominância absoluta do PAES sobre os outros dois algoritmos.

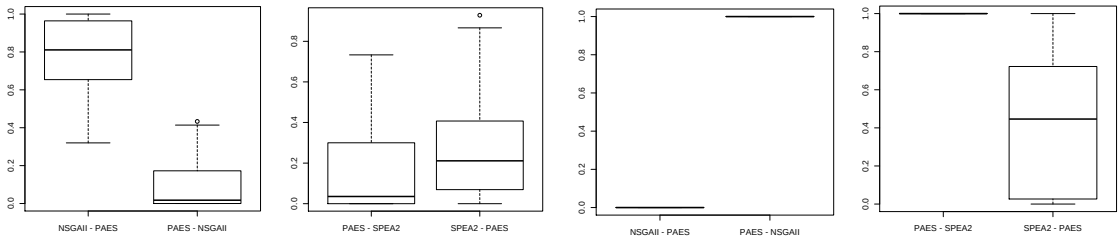
Tabela 5. $P - values$ **retornados para o indicador C**

	NSGA-II x SPEA2	NSGA-II x PAES	SPEA2 x PAES
BCEL	0,09879	**4,562e-11**	**0,04647**
JHotDraw	0,07083	0,3737	0,5279
MyBatis	0,3097	**1,685e-14**	**1,887e-10**

5.2. Distância Geracional (GD) e Distância Geracional Inversa (IGD)

Para os cálculos dos indicadores GD e IGD são utilizados: um conjunto de Pareto Conhecido (P_{known}) para cada algoritmo, formado pelas soluções encontradas pelo MOEA nas 30 rodadas, eliminando-se repetidas e dominadas; e um conjunto único chamado Pareto Real (P_{true}) que é formado por todas as soluções encontradas pelos 3 algoritmos, nas 30 rodadas, eliminando-se repetidas e dominadas.

Pela Tabela 6 pode-se observar os resultados dos algoritmos para o indicador GD. O NSGA-II obteve melhor média de GD para os sistemas BCEL e JHotDraw, enquanto



(a) BCEL (b) BCEL (c) MyBatis (d) MyBatis

Figura 3. *Boxplots* para o indicador C

o PAES obteve melhor média de GD para o sistema MyBatis. O teste de Friedman retornou os $p - values$: 2.41615449499253e-06 para o BCEL, 0.0564904201243948 para o JHotDraw, e 7.35295806145384e-11 para o MyBatis, indicando que existe diferença significativa entre os algoritmos somente para os sistemas BCEL e MyBatis.

No caso do indicador IGD, os resultados foram semelhantes ao do GD: O NSGA-II obteve o melhor GD para os sistemas BCEL e JHotDraw e o PAES para o MyBatis, conforme Tabela 6. Mas, assim como no indicador GD, o teste de Friedman só atestou a diferença significativa entre os algoritmos para os sistemas BCEL e MyBatis. Os $p - values$ retornados são: 3.26990952923229e-07 para o BCEL, 0.0564904201243948 para o JHotDraw e 1.13411309337498e-10 para o MyBatis.

Tabela 6. Média e Desvio Padrão dos indicadores GD e IGD

Indicador	Sistema	NSGA-II		SPEA2		PAES	
		Média	Desvio Padrão	Média	Desvio Padrão	Média	Desvio Padrão
GD	BCEL	0,00456987	0,00210061	0,00562624	0,00238220	0,01033035	0,00412745
	JHotDraw	1,05060163	2,06375473	1,06093700	2,16080866	3,04802903	4,01798222
	MyBatis	0,04827702	0,01292301	0,05158728	0,01275482	0,01656948	0,00826484
IGD	BCEL	0,00457060	0,00220206	0,00569144	0,00259717	0,01187230	0,00526975
	JHotDraw	1,04954454	2,25400178	1,08284691	2,38655655	3,39360395	4,58535582
	MyBatis	0,05124519	0,01045320	0,05537378	0,01089798	0,01923484	0,00955634

A análise dos *boxplots* gerados pelo R e os resultados retornados pelo teste de Friedman indicam que, no caso do BCEL, não há diferença estatística entre o NSGA-II e o SPEA2 e que os dois são melhores que o PAES tanto nos indicadores GD como IGD. Já em relação ao MyBatis, a situação é inversa, o PAES alcança o melhor desempenho nestes dois indicadores e não há diferença entre os outros dois MOEAs.

5.3. Distância Euclidiana a Solução Ideal (ED)

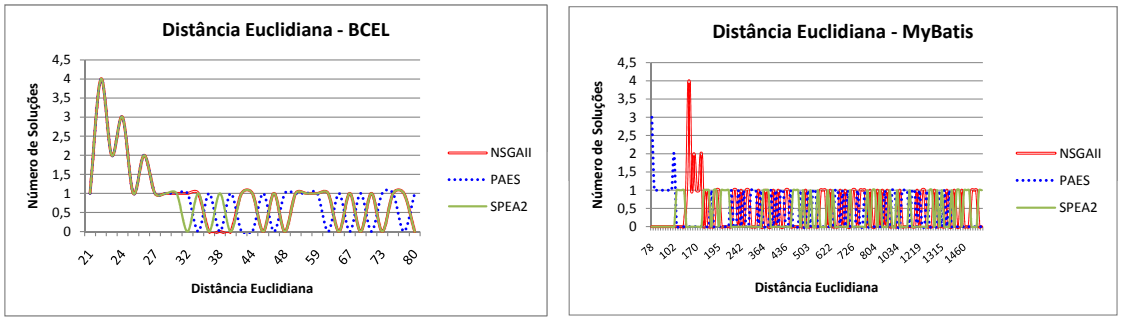
Para o cálculo da ED o custo da solução ideal para cada sistema é apresentado na segunda coluna da Tabela 7. Esta tabela também apresenta a menor distância encontrada e seu respectivo *fitness* para os sistemas usados no experimento. A menor distância encontrada está destacada em negrito. O PAES alcançou as soluções que tem a menor ED a partir das soluções ideais para o JHotDraw e o MyBatis. Para os outros dois sistemas os três MOEAs alcançaram a mesma ED.

Os gráficos da Figura 4 mostram o número de soluções encontradas por cada um dos MOEAs em relação a ED. No caso do BCEL (Figura 4(a)) é possível notar que não

Tabela 7. Custo da Solução Ideal e menor ED

Sistema	Custo da Solução Ideal	MOEA	Menor ED	*Fitness* da solução de menor ED
BCEL	45, 24	NSGA-II	**21,954498**	**56, 43**
		SPEA2	**21,954498**	**56, 43**
		PAES	**21,954498**	**56, 43**
JBoss	10, 6	NSGA-II	**0**	**10, 6**
		SPEA2	**0**	**10, 6**
		PAES	**0**	**10, 6**
JHotDraw	27, 10	NSGA-II	3,162278	28, 13
		SPEA2	3,162278	28, 13
		PAES	**1,414214**	**28, 11**
MyBatis	188, 77	NSGA-II	164,268074	298, 199
		SPEA2	148,121572	272, 199
		PAES	**78,230429**	**230, 143**

há diferença entre os algoritmos em termos de distribuição das soluções. Já no caso do MyBatis (Figura 4(b)), o PAES tem soluções mais perto da solução ideal do que os outros dois, seguido pelo SPEA2 que alcança duas soluções de ED baixa. No caso do JBoss e do JHotDraw foram encontradas 1 e 3 soluções, respectivamente, inviabilizando a análise de distribuição das soluções de acordo com a ED. Sendo assim, os três algoritmos têm um desempenho bem parecido para o sistema BCEL e o PAES alcança os melhores resultados para o sistema MyBatis neste indicador de qualidade (Tabela 7).



(a) BCEL (b) MyBatis

Figura 4. Distância Euclidiana das Soluções Encontradas pelos MOEAs

5.4. Discussão dos resultados

Após a apresentação dos resultados dos quatro indicadores de qualidade é possível analisá-los. Os três MOEAs alcançaram uma única solução para o sistema JBoss, indicando que neste caso os objetivos não estão em conflito. Apesar de ser um sistema complexo os três algoritmos também se mostraram equivalentes para o sistema JHotDraw considerando os resultados dos quatro indicadores.

Em relação ao BCEL, o NSGA-II alcançou os melhores valores de GD e IGD e teve resultados de cobertura equivalentes aos do SPEA2. Em termos de ED, os três algoritmos tiveram a mesma distribuição de soluções e alcançaram a mesma distância. Ainda assim pode-se apontar o NSGA-II como o melhor para este caso já que teve melhor desempenho em dois indicadores e dominou as soluções do PAES em relação à cobertura.

No sistema MyBatis, o mais complexo, o PAES alcançou os melhores resultados em todos os indicadores de qualidade e sua superioridade foi atestada estatisticamente. Sendo assim, há evidências de que o PAES se ajusta melhor para tratar problemas multi-objetivos bastante complexos.

6. Exemplo de Uso

O uso de MOEAs para resolver o problema CITO é factível e eficaz como relatado acima; e boas ordens de teste puderam ser obtidas automaticamente por meio deles. O testador pode escolher entre elas de acordo com algum objetivo que ele precise priorizar ou outros fatores associados ao desenvolvimento de software. Para ilustrar isso, na Tabela 8 são apresentados os custos de algumas soluções encontradas pelos MOEAs para o MyBatis.

Nessa tabela, a solução a tem a menor ED e a solução b é uma alternativa da solução a uma vez que tem a segunda menor ED. Mas, como este resultado pode ser utilizado pelo testador a fim de escolher entre a e b? Se a solução a do PAES fosse escolhida para teste do sistema MyBatis (Tabela 8), seria preciso criar *stubs* para simular dependências de 230 atributos e 143 métodos. Por outro lado, se a solução b do mesmo algoritmo fosse escolhida seria preciso criar *stubs* para simular dependências de 223 atributos e 147 métodos. Desta forma, no primeiro caso, o objetivo priorizado seria a medida M e, no segundo caso, a medida A. As soluções a e b têm o melhor *trade-off* entre os objetivos e, por isso, possuem menores EDs do que as soluções que contêm valores menores (porém extremos) para as medidas A ou M.

Tabela 8. Custo de algumas soluções próximas a solução ideal para o MyBatis

	NSGA-II			SPEA2			PAES			Solução
	A	M	ED	A	M	ED	A	M	ED	Ideal
a	298	199	164,268	272	199	148,122	230	143	78,230	188, 77
b	304	196	166,184	279	197	150,602	223	147	78,262	

Já que os MOEAs encontraram uma única solução para o sistema JBoss, o testador não precisa estabelecer qualquer critério para escolher a ordem de integração e teste de classes porque a melhor sequência é conhecida (solução gerada pelos algoritmos).

Apesar de neste estudo a análise e exemplos considerarem três MOEAs, na vida real o testador provavelmente terá resultados de um único algoritmo. A partir de todas as soluções obtidas, o testador poderá decidir qual objetivo priorizar e escolher a ordem de teste de acordo com esta prioridade. A decisão de priorização de objetivos pode ser influenciada por diversos fatores, tais como restrições econômicas e de negócios.

7. Conclusões

Neste trabalho foi analisado o emprego de três MOEAs (NSGA-II, SPEA2 e PAES) para resolver o problema CITO visando a minimização de dois objetivos que representam fatores que influenciam na criação de *stubs* durante o teste de integração de classes.

Foi realizado um experimento envolvendo quatro sistemas reais: BCEL, JBoss, JHotDraw e MyBatis. Três destes sistemas são exemplos característicos do problema CITO, pois contêm objetivos conflitantes e, portanto, os algoritmos encontraram um conjunto de diferentes alternativas que representam uma solução de compromisso entre os

objetivos. Apesar disso, o sistema JBoss é um caso particular onde isso não ocorre, e por esta razão os três algoritmos encontraram uma única solução.

Os resultados do experimento mostraram que os MOEAs são eficazes para resolver o problema em questão e se mostraram equivalentes quando da análise dos indicadores de qualidade para alguns sistemas. Entretanto, o algoritmo PAES apresentou o melhor desempenho para o sistema mais complexo utilizado no experimento. Isso fornece evidências de que talvez a sua estratégia de evolução seja mais eficaz do que as estratégias adotadas pelos outros dois MOEAs quando existem objetivos muito conflitantes no sistema a ser testado. Apesar disso, como vários fatores podem influenciar o custo de construção de *stubs* durante a fase de teste de integração, é importante analisar o comportamento deste algoritmo quando se utilizam mais de dois objetivos.

Neste contexto, pretende-se realizar novos experimentos utilizando esses três MOEAs com um número maior de objetivos, e, possivelmente utilizando outros sistemas, a fim de confirmar as evidências identificadas neste trabalho.

8. Agradecimentos

Os autores agradecem a Edison Klafke Fillus por sua contribuição no experimento. Este trabalho foi apoiado pela Fundação Araucária, CAPES/Reuni e CNPq.

Referências

Abdurazik, A. and Offutt, J. (2006). Coupling-based Class Integration and Test Order. In *International Workshop on Automation of Software Test*, pages 50–56, Shanghai, China. ACM.

Assunção, W. K. G., Colanzi, T. E., Pozo, A. T. R., and Vergilio, S. R. (2011a). Establishing integration test orders of classes with several coupling measures. In *Genetic and Evolutionary Computation Conference*. To appear.

Assunção, W. K. G., Colanzi, T. E., Pozo, A. T. R., and Vergilio, S. R. (2011b). Reduzindo o custo do teste de integração com algoritmos evolutivos multiobjetivos e diferentes medidas de acoplamento. In *VIII Encontro Nacional de Inteligência Artificial*. Submitted.

Briand, L. and Labiche, Y. (2003). An investigation of graph-based class integration test order strategies. *IEEE Transactions on Software Engineering*, 29(7):594–607.

Briand, L. C., Feng, J., and Labiche, Y. (2002a). *Experimenting with Genetic Algorithms and Coupling Measures to Devise Optimal Integration Test Orders*. Carleton University, Technical Report SCE-02-03.

Briand, L. C., Feng, J., and Labiche, Y. (2002b). Using genetic algorithms and coupling measures to devise optimal integration test orders. In *14th International Conference on Software Engineering and Knowledge Engineering*, Ischia, Italy.

Cabral, R., Pozo, A., and Vergilio, S. (2010). A Pareto Ant Colony Algorithm Applied to the Class Integration and Test Order Problem. In *22nd IFIP International Conference on Testing Software and Systems (ICTSS'10)*. Springer.

Cochrane, J. and Zeleny, M. (1973). *Multiple Criteria Decision Making*. University of South Carolina Press, Columbia.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2):182 –197.

Durillo, J., Nebro, A., and Alba, E. (2010). The jMetal framework for multi-objective optimization: Design and architecture. In *2010 IEEE Congress on Evolutionary Computation(CEC)*, pages 4138–4325, Barcelona, Spain.

Gárcia, S., Molina, D., Lozano, M., and Herrera, F. (2009). A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. *Journal of Heuristics*, 15(6):617–644.

Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland. Revised version.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8:149–172.

Kung, D., Gao, J., Hsia, P., Toyoshima, Y., and Chen, C. (1995a). A test strategy for object-oriented programs. In *19th International Computer Software and Applications Conference*. IEEE Computer Society.

Kung, D. C., Gao, J., Hsia, P., Lin, J., and Toyoshima, Y. (1995b). Class firewal, test order and regression testing of object-oriented programs. *Journal of Object-Oriented Program*, 8(2).

Melton, H. and Tempero, E. (2007). An empirical study of cycles among classes in Java. *Empirical Software Engineering*, 12:389–415.

Pareto, V. (1927). *Manuel D'Economie Politique*. Ams Press, Paris.

Pozo, A., Bertoldi, G., Árias, J., Cabral, R., and Vergilio, S. (2011). Multi-objective optimization algorithms applied to the class integration and test order problem. *Software Tools for Technology Transfer*. Submitted.

Radziukyniene, I. and Zilinskas, A. (2008). Evolutionary Methods for Multi-Objective Portfolio Optimization. In *Proceedings of the World Congress on Engineering 2008 Vol II*.

Tai, K.-C. and Daniels, F. J. (1997). Test order for inter-class integration testing of object-oriented software. In *21st International Computer Software and Applications Conference*, pages 602–607. IEEE Computer Society.

Traon, Y. L., Jéron, T., Jézéquel, J.-M., and Morel, P. (2000). Efficient object-oriented integration and regression testing. *IEEE Transactions on Reliability*, pages 12–25.

van Veldhuizen, D. A. and Lamont, G. B. (1999). Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, SAC'99, pages 351–357, New York, NY, USA. ACM.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

J-Attack - Injetor de Ataques para Avaliação de Segurança de Aplicações Web

Plinio C. S. Fernandes, Tânia Basso, Regina Moraes

Faculdade de Tecnologia – Universidade Estadual de Campinas (UNICAMP)
Caixa Postal 456 – 13484-332 - Campus I – Limeira – SP – Brasil
{pliniocsfernandes,taniabasso}@gmail.com, regina@ft.unicamp.br

Abstract. *The security of software systems is mandatory in current software development scenario. It requires expertise in several areas of knowledge and includes the study of malicious faults, in order to know them and use this knowledge to allow providing computer systems ability to detect and avoid them without compromising the security required by the users. This paper presents the design and development process of a tool that is able to inject attacks in Web applications aims to use it to evaluate the security of the applications.*

Resumo. *A busca por sistemas seguros é uma necessidade no cenário de software atual. A segurança de sistemas computacionais exige conhecimentos específicos em diversas áreas de conhecimento e inclui o estudo de falhas maliciosas, para que, conhecendo-as, seja possível prover aos sistemas computacionais capacidade de detectá-las e evitá-las sem comprometer a segurança exigida pelos usuários. Este artigo apresenta o projeto e o processo de desenvolvimento de uma ferramenta capaz de injetar ataques em aplicações Web para utilização em avaliações de segurança de aplicações.*

1. Introdução

Vulnerabilidades de segurança de software têm sido amplamente exploradas com o intuito de obter informações confidenciais e invadir redes corporativas. Isto ocorre devido à complexidade dos produtos de software, que é cada vez maior, e resulta em vulnerabilidades de segurança produzidas por má codificação. Também ocorre devido restrições de tempo e custo, pois, diante desses fatores, desenvolvedores, muitas vezes, mantêm o enfoque principal do desenvolvimento da aplicação na funcionalidade e na satisfação dos requisitos do cliente, negligenciando aspectos de segurança. Assim, normalmente, não é adotado um ciclo de desenvolvimento seguro para o software. O desenvolvimento de software seguro visa contribuir com melhorias ao cenário da falta de segurança em aplicações Web e reduzir a quantidade de falhas residuais no código, que podem levar a vulnerabilidades de segurança.

Sabe-se que mecanismos tradicionais de segurança de redes como *firewalls*, criptografia e sistemas de detecção de intrusão podem proteger a rede, mas não evitam ataques às aplicações. Por este motivo, o foco de ataques tem mudado da rede para as aplicações *Web*, onde codificação insegura representa grande risco [Singhal *et al* 2007].

Sabendo da importância da segurança de software para as organizações e da expressiva quantidade de vulnerabilidades de segurança que a grande maioria das aplicações *Web* apresenta, provedores de soluções voltadas para o desenvolvimento de software colocam no mercado um número significativo de ferramentas para detectar as vulnerabilidades dessas aplicações, os *scanners* de vulnerabilidades. No entanto, estas ferramentas apresentam baixa cobertura (o scanner deixa de detectar vulnerabilidades que realmente existem na aplicação) e alta taxa de falsos positivos (o resultado da análise indica vulnerabilidade de segurança que efetivamente não existe na aplicação).

Visando avaliar as ferramentas de vulnerabilidades (*scanners*) e sistematizar a validação de aplicações *Web* quanto às vulnerabilidades de segurança, foi desenvolvido em um trabalho prévio [Fernandes *et al* 2010] uma metodologia com base em árvores de ataque. As árvores de ataque foram usadas para descrever cenários que pudessem revelar vulnerabilidades nas aplicações e para guiar a injeção de ataques para avaliar se os scanners eram eficazes na detecção de vulnerabilidades (cobertura) sem apresentar uma alta taxa de falsos positivos.

Durante o desenvolvimento do trabalho foi percebido que poderia ser criada uma ferramenta para automatizar, ao menos em parte, a injeção dos ataques através do modelo das árvores. Dessa forma, o objetivo deste artigo é relatar o projeto e desenvolvimento de um Injetor de Ataques (*J-Attack*) para avaliação de segurança de aplicações *Web*. O objetivo a ser atingido é a extensão do estudo de vulnerabilidades de segurança de sistemas computacionais, procurando prover uma maneira automática de emular ataques de segurança.

Depois da Introdução que esta Seção apresentou, o restante deste artigo apresenta na Seção 2 os trabalhos relacionados que serviram de embasamento teórico para o desenvolvimento da pesquisa relatada neste artigo; a Seção 3 apresenta as Árvores de Ataque que serviram de base para o desenvolvimento do Injetor de Ataque "*J-Attack*" que, por sua vez, é apresentado na Seção 4; um Estudo de Caso é apresentado na Seção 5 e a Seção 6 apresenta os Resultados e Discussões; finalmente a Seção 7 apresenta algumas Considerações e Trabalhos Futuros.

2. Trabalhos Relacionados

A interação do usuário com uma aplicação *Web* é feita através de um navegador que é responsável por interpretar arquivos contendo a linguagem de marcação *Hypertext Markup Language* (HTML) e *Cascading Stylesheets* (CSS), como também códigos de script em algumas linguagens, sendo a mais conhecida e utilizada o *Javascript*. O *Hypertext Transfer Protocol* (HTTP) é o protocolo de comunicação mais comumente usado por aplicações Web, sendo que as requisições contêm seu método, o *Uniform Resource Identifier* (URI) e a versão do protocolo, seguidos por uma mensagem do tipo *Multipurpose Internet Mail Extensions* (MIME) com os modificadores da requisição, informação do cliente e um conteúdo. [Fielding *et al*. 1999].

Aplicações *Web* são alvos constantes de ataques, pois são aplicações com alta exposição na rede mundial. Segundo o *Open Web Application Security Project* [OWASP 2010], uma vulnerabilidade é uma brecha ou uma fraqueza na aplicação, que pode ser uma falha de *design* ou um *bug* de implementação, que permite que um atacante cause problemas para seus *stakeholders*, ou seja, que ataque a aplicação vulnerável. Devido à

alta exposição, qualquer vulnerabilidade nas aplicações *Web* será inevitavelmente atacada em algum momento. Estes ataques não podem ser protegidos por mecanismos tradicionais de segurança de redes, tais como *firewalls*, criptografia e sistemas de detecção de intrusão. A razão é que esses ataques são feitos através de portas que são usadas pelo tráfego regular da rede [Singhal *et al* 2007] e para prover a proteção necessária, grande conhecimento sobre o contexto de negócio é necessário [OWASP 2010]. Por este motivo, o foco de ataques tem mudado da rede para as aplicações *Web*, onde codificação insegura representa grande risco [Singhal *et al* 2007].

Muitos trabalhos publicados mostram que aplicações *Web* apresentam falhas graves de segurança [Christey and Martin 2007, Acunetix 2007, NT Monitor 2008]. Acunetix (2007) apresentou o resultado de um experimento que inspecionou (*scan*) 3200 *websites* durante um ano. De acordo com esses resultados, 70% dos *websites* apresentaram vulnerabilidades de alto ou médio risco para as organizações. O relatório anual sobre segurança da NTA Monitor (2008) relata que 25% das organizações que participaram dos testes, apresentaram uma ou mais vulnerabilidades de alto risco e 78% delas contêm vulnerabilidades de médio risco. Esses fatos mostram que as aplicações *Web* são entregues sem o devido cuidado em relação à segurança e ressalta a importância da pesquisa na área de segurança para esse segmento de aplicações.

Uma maneira que as organizações provedoras de soluções para o desenvolvimento de software acharam para amenizar o problema foi o desenvolvimento de ferramentas para detectar as vulnerabilidades dessas aplicações, os *scanners* de vulnerabilidades [HP 2011, Acunetix 2011, IBM 2011]. Porém, como experimentalmente comprovado por diversos trabalhos científicos, estas ferramentas apresentam baixa cobertura e alta taxa de falsos positivos [Fonseca *et al* 2007, Vieira *et al* 2009, Bau *et al* 2010, Basso *et al* 2010].

Diversas ferramentas foram desenvolvidas para injetar ataques, normalmente voltadas para um determinado tipo de ataque, por exemplo, SQL Injection [SqlMap (2011), Havij (2011), SqlHelper (2011)]. Porém, a ferramenta que está sendo proposta procura ser flexível para que se possa acoplar injetores para diferentes vulnerabilidades além de implementar a visão do atacante através da árvore de ataques. Em Fernandes *et al* (2010), a utilidade das árvores de ataques foram investigadas e modeladas para três principais vulnerabilidades. Este trabalho complementa o anterior, implementando a ferramenta que usa os modelos desenvolvidos no primeiro trabalho.

Na próxima seção árvores de ataque são abordadas para introduzir a maneira como o injetor de ataques consegue automatizar os ataques injetados.

3. Árvores de Ataque

O modelo de árvores de ataque foi desenvolvido por SCHNEIER (1999) e tem como vantagem permitir a visualização da aplicação do ponto de vista de um atacante. Neste modelo o nó raiz representa o sucesso em alcançar a meta final de um ataque. Cada nó filho representa submetas que devem ser alcançadas para que a meta do nó pai seja alcançada. Nós pai podem estabelecer uma relação entre seus filhos utilizando um relacionamento “OU” ou “E”. Em um relacionamento OU, se qualquer meta de um nó filho for alcançada a meta do nó pai também é alcançada. Já em um relacionamento “E”, as metas de todos os nós filhos devem ser alcançadas para que a meta do nó pai seja

alcançada. Cenários individuais de invasão são gerados ao se navegar na árvore de um modo *depth-first* (busca em profundidade).

Não existem orientações específicas para a construção das árvores de ataque e no trabalho de Fernandes *et al* (2010) foi utilizada uma abordagem *bottom up*, onde os passos para alcançar a meta principal são identificados e representados de acordo com a ordem na qual eles devem ser executados (dos nós folha para o nó raiz). Para não sobrecarregar as árvores, as marcações de relacionamentos do tipo "OU" foram omitidas.

O OWASP mantém uma lista, anualmente atualizada, dos dez tipos de vulnerabilidade que apresentam o maior risco para as organizações. Desta lista [OWASP 2010] foram escolhidos, para o presente trabalho, três tipos de vulnerabilidades a serem investigadas: SQL Injection, XSS e CSRF. As vulnerabilidades escolhidas são respectivamente a primeira, a segunda e a quinta da lista. As duas primeiras foram escolhidas por estarem nos primeiros lugares da lista enquanto a quinta foi escolhida por ser uma vulnerabilidade que apareceu em muitos resultados dos *scanners* comerciais nos trabalhos anteriores do grupo e é um tipo de vulnerabilidade que já causou problemas em grandes sites como Gmail e Yahoo. A próxima Seção apresenta estas vulnerabilidades.

3.1. *SQL Injection*

Explorar vulnerabilidades do tipo *SQL Injection* consiste em enganar o interpretador de SQL para que este execute comandos manipulados. Isto é feito através da entrada de dados do usuário, que é enviada como uma parte da requisição para a aplicação *Web*. Um atacante pode obter acesso ao banco de dados da aplicação e, de forma arbitrária, criar, ler, alterar ou apagar quaisquer dados disponíveis para a aplicação. Aplicações *Web* se tornam vulneráveis para este tipo de ataque quando dados providos pelo usuário para o interpretador não são validados ou recodificados.

A Figura 1 mostra a árvore de ataque modelada para representar as etapas que permitem explorar vulnerabilidades do tipo *SQL Injection*. As etapas de 1 a 4 mostram as diferentes maneiras de injetar códigos maliciosos, ou seja, as diferentes entradas de dados [Halfond 2006]. A etapa representada pelo nó 6 consiste em encontrar mecanismos de envio de dados do *site* que transmitem informações para o banco de dados, encontrando meios adequados de ataque. Após identificar os possíveis dados de entrada para injeção, é necessário criar o trecho de código SQL que será utilizado no ataque (representado pelo nó 5). O código SQL é construído de acordo com a intenção do ataque e deve considerar algumas técnicas para evitar mecanismos de detecção de assinatura, caso necessário. Após criar o código que será injetado, este pode ser testado em diferentes mecanismos de entrada (nó 7). Validações podem ser feitas pela aplicação apenas no lado do cliente e não no servidor. Nestes casos o atacante deve utilizar métodos para passar pela validação (nó 8). Os nós 9 e 10 representam os passos para verificar a resposta do ataque e, consequentemente, se este foi bem sucedido.

3.2. *Cross Site Scripting*

O ataque do tipo *Cross Site Scripting* (XSS) consiste em executar *scripts* no navegador da vítima permitindo roubo de sessão, inserção de conteúdo hostil, roubo de

informações pessoais e outros danos. Este tipo de ataque pode ser executado quando aplicações recebem dados do usuário e enviam estes dados para o navegador sem validação ou recodificação apropriada.

Os ataques de XSS podem ser classificados em três tipos (OWASP, 2010): (i) XSS armazenado (*Stored XSS*): dados de script são injetados na aplicação e armazenados por ela e quando a aplicação é acessada mais tarde, o script é executado; (ii) XSS refletido (*Reflected XSS*): o script injetado é imediatamente mostrado, portanto executado, logo após ser enviado para a aplicação; (iii) XSS baseado em DOM (*DOM-based XSS*): ataques deste tipo utilizam scripts para manipular o *Document Object Model* (DOM) do navegador da vítima, sendo assim, executado dinamicamente mesmo quando a aplicação não sofreu modificações.

A Figura 2 mostra a árvore de ataques modelada para ataques de XSS. O primeiro passo para efetuar este tipo de ataque é encontrar os chamados *XSS holes* (nó 1), que são indicadores de que a aplicação pode ser vulnerável. Um *XSS hole* pode ser descrito como uma parte da aplicação onde é possível injetar informações para serem exibidas pela aplicação. O próximo passo é testar a injeção de *scripts* na aplicação. Para isto, é necessário verificar se a aplicação valida caracteres especiais nos dados inseridos. Se não houver validação (nó 2), o *script* malicioso pode ser injetado (nó 3).

Uma forma simples de testar se a aplicação é vulnerável seguindo os passos 1, 2 e 3, é injetar, em todos os campos de entrada de dados, um *script* simples, que apenas exibe uma mensagem, como uma das variações abaixo [Snake 2010]:

```
<script>alert('XSS')</script>
&{alert('XSS')};
<img src=javascript:alert('XSS')>
```

Caso a aplicação valide os dados inseridos, ainda pode ser possível executar o ataque com um *script* específico para evasão de filtros e acessar a aplicação alvo utilizando um navegador compatível com a sintaxe do *script* (nó 4).

Os últimos passos para fazer o ataque XSS consistem em acessar a parte da aplicação onde o *script* é executado (nó 5), pois esta parte pode não ser diretamente a próxima a ser acessada caso o ataque seja de XSS armazenado. O nó 6 consiste em acessar a aplicação com um navegador adequado, pois diferentes navegadores processam *scripts* de formas diferentes e alguns possuem mecanismos de segurança contra ataques de XSS (Browser Scope Project, 2010). Isto é importante para identificar em quais navegadores o ataque será bem sucedido.

3.3. *Cross Site Request Forgery*

Um ataque de *Cross Site Request Forgery* (CSRF) força o navegador da vítima, que está autenticado na aplicação alvo, a enviar uma requisição à aplicação *Web* vulnerável. Como a vítima tem uma sessão autenticada no navegador, a aplicação *Web* executa a ação desejada em nome da vítima, de acordo com a requisição recebida. Este tipo de ataque é possível, pois aplicações *Web* autorizam requisições baseadas em credenciais que são enviadas automaticamente, como *cookies* de sessão, endereços IP, certificados SSL entre outros [Auger 2010]. A Figura 3 mostra a árvore modelada para esta

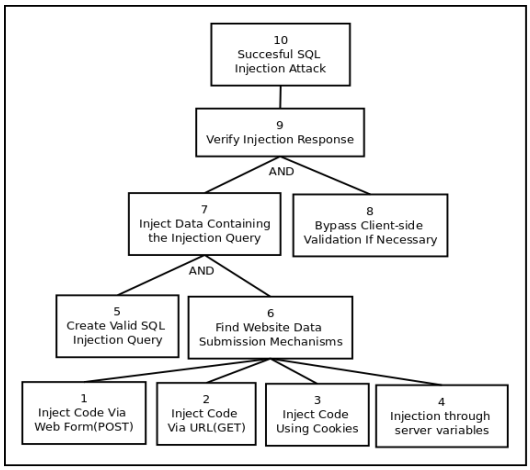


Figura 1. Árvore de Ataque - SQL Injection

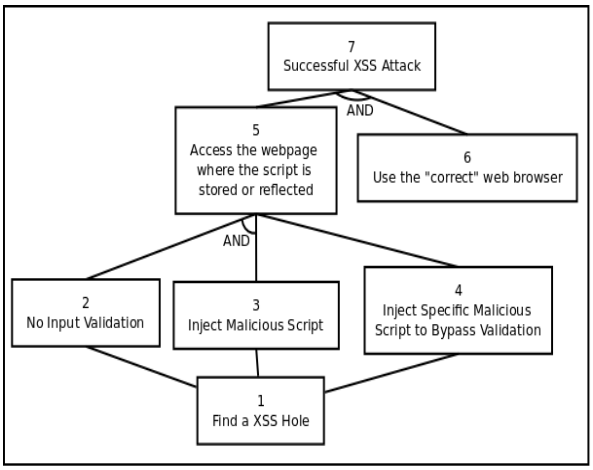


Figura 2.Árvore de Ataque–Cross Site Scripting

vulnerabilidade, cujo percurso segue a mesma lógica apresentada para vulnerabilidades precedentes.

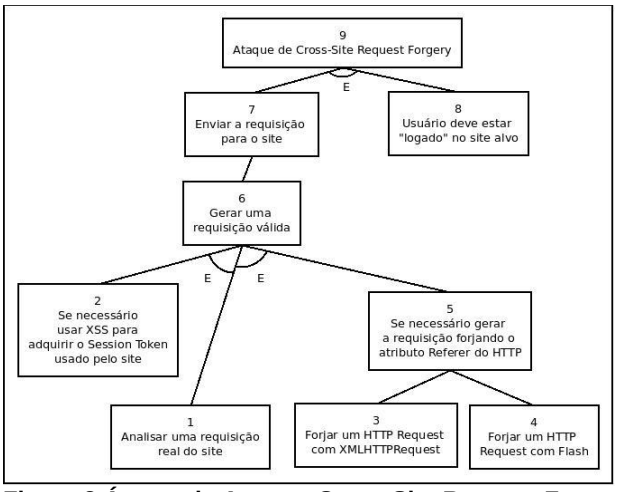


Figura 3.Árvore de Ataque–Cross Site Request Forgery

4. *J-Attack*

Esta seção tem como objetivo apresentar as soluções que foram utilizadas para o desenvolvimento da ferramenta proposta, descrevendo a aplicação, ferramentas, requisitos, soluções de implementação e procedimentos de teste.

4.1. O desenvolvimento do Projeto

Foi utilizado o modelo *Test Driven Development* (TDD) para o desenvolvimento do projeto. O modelo foi proposto por Kent Beck, mesmo autor da proposta da metodologia ágil *Extreme Programming* (XP) e a premissa básica do TDD pode ser resumida na frase de Ron Jeffries "*Clean code that works*" [Beck 2002].

Apesar de os modelos ágeis muitas vezes não possuírem uma fase de modelagem muito bem definida e de não gerarem artefatos baseados na *Unified Modeling Language* (UML) para documentação do sistema, neste projeto foram utilizados dois diagramas UML: diagrama de casos de uso e diagrama de classes. Estes dois diagramas foram selecionados pois permitem um planejamento inicial do sistema, que é importante para garantir uma boa arquitetura.

Outra característica utilizada no desenvolvimento foi a utilização de padrões de projeto (*design patterns*). Padrões de projeto são soluções padronizadas para problemas que ocorrem com frequência em projetos de software. Estas soluções definem apenas o conceito principal da solução e deve ser implementada de acordo com cada caso. Neste projeto foi utilizado o padrão "Strategy" [Gamma *et. al*., 1995]. O padrão Strategy define uma família de algoritmos, encapsula cada algoritmo e torna possível a utilização de qualquer um deles em determinado ponto do código. A utilização do Strategy no projeto permite que diferentes algoritmos sejam selecionados de acordo com a necessidade do teste em execução, i.e., de acordo com o ataque que se pretende injetar. Com isso a ferramenta ganha em flexibilidade (pode executar os testes com base no que está cadastrado no banco de dados) e escalabilidade (facilita a expansão da ferramenta, bastando para isso criar um novo algoritmo na família do Strategy).

As ferramentas que foram utilizadas durante o desenvolvimento do projeto foram: (i) Ubuntu Linux, como sistema operacional; (ii) Java, como linguagem de programação; (iii) JUnit, para os testes unitários; (iv) Netbeans IDE; (v) Apache Subversion, para o controle de versões; (vi) MySQL, como sistema gerenciador de banco de dados; (vii) MySQL Workbench, para modelagem; (viii) Dia, para modelagem; (ix) Eclipse, como ambiente de desenvolvimento integrado (ADI).

4.2. Os requisitos do Projeto

Os requisitos do injetor, entendidos pelos integrantes do Grupo de Engenharia de Sistemas e Informação da Faculdade de Tecnologia são descritos a seguir:

- Enviar requisições HTTP e verificar a resposta do *site* para as requisições;
- Mapear um *site* através dos *links* da página inicial;
- Procurar padrões nas respostas de requisições (todo o site ou em páginas específicas);
- Comparar duas ou mais respostas HTTP do servidor;
- Encontrar mecanismos de entrada de dados do *site* (*inputs*);
- Executar testes baseados em um banco de dados que contenha variações de ataques;
- Executar testes em uma determinada ordem;
- Possuir uma estrutura que possibilite a ampliação do sistema futuramente;
- Utilizar autenticação para verificar *sites* que necessitem desse recurso.

A partir do levantamento de requisitos mencionados, foi gerado o diagrama de casos de uso que é apresentado na Figura 4. Nele, definimos a interação do usuário com o sistema e as ações básicas que a aplicação deve ser capaz de executar.

O banco de dados tem a finalidade de armazenar casos de teste extraídos das árvores de ataque que serviram para guiar a ferramenta na execução dos testes. Foram identificadas as entidades principais: Vulnerabilidade (armazena tipos de vulnerabilidade), Teste (armazena testes para um determinado tipo de Vulnerabilidade) e Ação (armazena parâmetros que deve ser executada como passo de um teste).

A Ação pode ser de um dos seguintes tipos: Requisição, Procurar Padrão, Encontrar Entradas, Comparar Respostas. A Figura 5 apresenta o modelo relacional do Projeto. No modelo relacional há sete tabelas representando as principais entidades identificadas no Diagrama Entidade Relacionamento, sendo que quatro tabelas são

complementares à tabela “acao” e contém os dados específicos para cada tipo de Ação identificada.

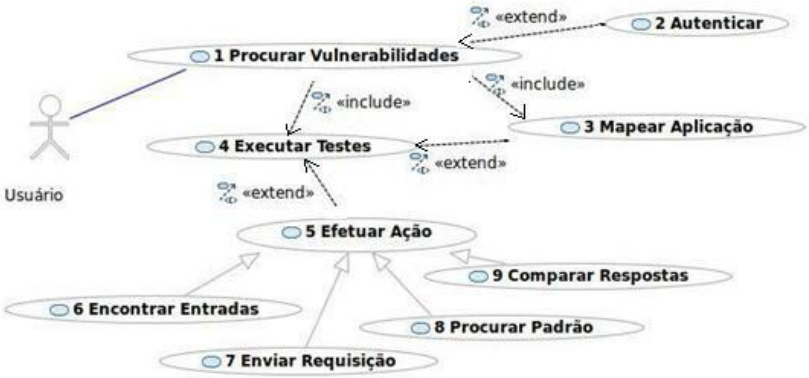


Figura 4. Diagrama de Casos de Uso da *J-Attack*

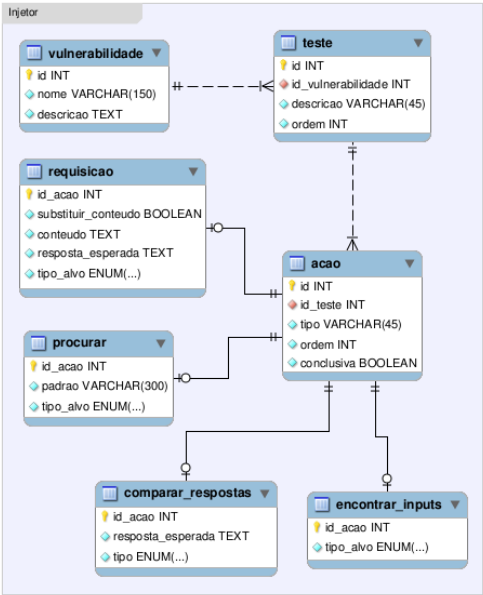


Figura 5. Modelagem Relacional do Banco de Dados da *J-Attack*

A Figura 6 apresenta o diagrama de classes do sistema. Os métodos de acesso aos atributos (“get” e “set”) das classes foram omitidos para simplificar a visualização. No diagrama é possível identificar a utilização do padrão de projeto Strategy que tem como tarefa a seleção de algoritmos em tempo de execução, pois a ferramenta precisa variar a execução de acordo com a definição do banco de dados.

4.3. Implementação e Testes

Inicialmente, foram criadas as classes com os atributos e as assinaturas dos métodos. Testes unitários foram criados, a seguir, para cada classe projetada, contendo testes para todos os métodos da respectiva classe, exceto “gets” e “sets” que não executam nenhuma validação ou processamento adicional além do acesso aos atributos.

Os testes unitários criados foram baseados na especificação e depois da criação dos testes unitários cada método correspondente foi implementado e validado até que se obtivesse sucesso na validação.

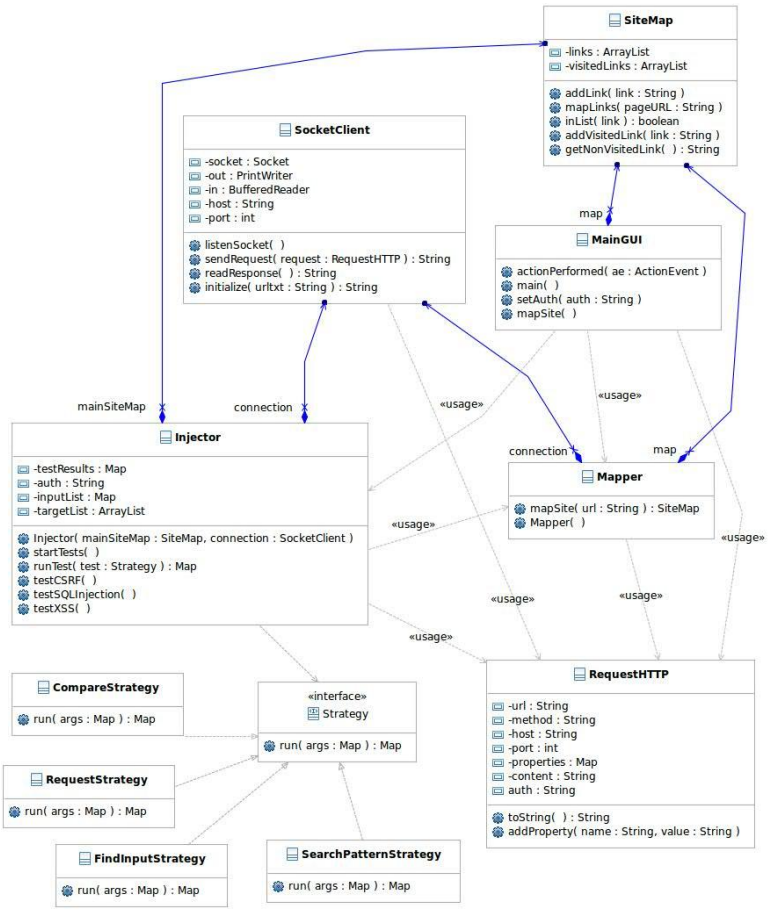


Figura 6. Diagrama de Classes da *J-Attack*

O uso de alguns *stubs* (classe que substitui a classe chamada) foi necessário para possibilitar os testes de integração da ferramenta a cada parte implementada. Os testes unitários foram utilizados, nesta fase, para os testes de regressão, executados a cada nova funcionalidade desenvolvida.

A ferramenta, depois de totalmente integrada, foi validada com um sistema simples. O banco de dados foi populado com casos de teste básicos (para maiores detalhes ver Fernandes *et al* (2010)) para a validação inicial.

5. Estudo de Caso

Como estudo de caso, três aplicações *Web* foram testadas para detectar vulnerabilidades de segurança. Os testes foram projetados com base nas árvores de ataque e os resultados obtidos foram comparados com os resultados de um *scanner* de vulnerabilidade comercial. A ferramenta foi utilizada para realizar, de forma automática, os testes realizados manualmente em Fernandes *et al.* (2010) e se mostrou eficiente para a automatização dos testes.

As três aplicações *Web* selecionadas foram desenvolvidas em Java e têm códigos fonte disponíveis na Internet. A primeira aplicação é um Gerenciador de Relacionamento com o Cliente (CRM) (Eberom 2010), usa MySQL Server 5.0 e servidor de aplicação Tomcat 5.5.28. Além disso, usa tecnologia Hibernate, *framework* Struts e JasperReports para a geração de relatórios. A segunda aplicação é um Gerenciador de Ensino a Distância (Amadeus 2010), usa PostgreSQL 8.4 e servidor de aplicação Tomcat 5.5.28. Também usa Hibernate e AJAX (*Asynchronour Javascript and XML*). A terceira aplicação é um gerenciador de *Call Center* de uma clínica de saúde, usa *framework* Java *Server Faces*, servidor de aplicação JBoss 4.2, MySQL Server 5.0, Hibernate e AJAX. As três aplicações foram selecionadas porque os serviços providos são aplicáveis para áreas comerciais e acadêmicas e usam tecnologias recentes que podem impactar a presença de vulnerabilidades de segurança. Os nomes das ferramentas são preservados, pois não permitem a publicação de resultados relacionados à marca e foram nominadas App1, App2 e App3 sem nenhuma ordem.

Os experimentos foram executados usando um *scanner* de vulnerabilidades comercial, que também por questões de sigilo tem seu nome preservado. Primeiramente, o *scanner* comercial foi utilizado para inspecionar as aplicações. Depois, os testes que usaram os casos de teste projetados com base na árvore de ataque foram executados com a *J-Attack* em cada uma das vulnerabilidades detectadas. Os resultados obtidos com os *scanners* comerciais foram comparados, minuciosamente validados manualmente e classificados como corretos, falsos positivos ou falha de cobertura. O objetivo dos testes era validar a eficácia da *J-Attack* e a validação manual foi necessária para saber se a ferramenta foi incapaz de explorar a vulnerabilidade ou se a vulnerabilidade não existia.

1. Localize um mecanismo de submissão de dados (por exemplo, um campo de busca).
2. Insira um caractere especial (por exemplo, “’”) e submeta a consulta.
3. Observe a resposta normal. Se a aplicação retornar uma mensagem de erro de sintaxe pode ser que esteja vulnerável.
4. Injete uma consulta SQL com uma condição verdadeira adicional para ser concatenada com a consulta original. Isso fará que a consulta retorne um resultado manipulado (por exemplo, ‘ union select load_file (‘/etc/passwd’),1 #)
5. Observe a resposta e verifique se a consulta foi executada e retornou resultados esperados

Figura 7. Caso de Teste para *SQLInjection*

A Figura 7 apresenta um caso de teste para a vulnerabilidade *SQL Injection*. O caso de teste apresentado foi derivado dos passos 5 AND 6, 7 AND 8, 9 e 10 da árvore apresentada na Figura 1.

1. Localize uma entrada de dados de usuário que tenha sido apresentada em uma das páginas da aplicação (i.e., tabela de dados, relatórios de tela, etc).
2. Insira um código *javascript* (por exemplo, <script>alert(‘vulnerável a XSS’</script>) e submeta a requisição.
3. Observe se o *javascript* foi executado (se aparecem mensagens *popups* na tela).

Figura 8. Caso de Teste para *XSS*

A Figura 8 apresenta um caso de teste para a vulnerabilidade *XSS*. O caso de teste apresentado foi derivado dos passos 1, 2 AND 3, 5 AND 6 e 7 da árvore apresentada na Figura 2. Por questões de restrição de espaço, o caso de teste para a vulnerabilidade CSRF foi omitido e pode ser encontrados em Fernandes *et al.* (2010).

6. Resultados e Discussão

Na Figura 9 é apresentada a interface da *J-Attack* quando executava um caso de teste de XSS. As interfaces para os demais tipos de vulnerabilidades foram omitidas devido a restrição de espaço. Em todos os casos, a ferramenta efetua o mapeamento da aplicação Web alvo e depois executa os testes cadastrados no banco de dados.

A aplicação encontrou os mecanismos de entrada de dados do site alvo, depois enviou requisição contendo um *script* para todas as páginas encontradas no passo anterior e verificou se o *script* estava presente na resposta. Neste passo, o ataque se mostrou bem sucedido. Para finalizar, a ferramenta efetuou uma busca pelo *script* injetado anteriormente em todo o site para verificar a existência de XSS armazenado. Na resposta o conjunto requisição-resposta aponta onde a vulnerabilidade foi encontrada.

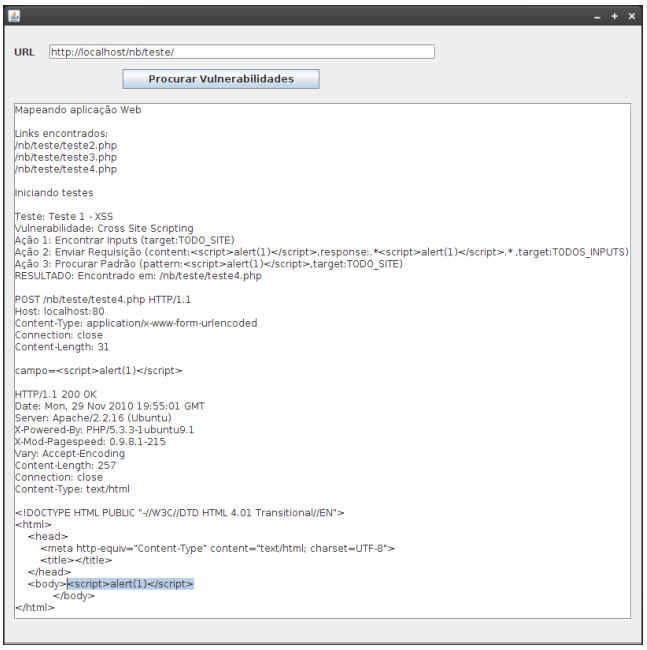


Figura 9. Interface da *J-Attack* – *Execução de ataque XSS*

A Figura 10 apresenta a execução da ferramenta com um teste de SQL Injection carregado. Primeiramente, a aplicação encontrou os mecanismos de entrada de dados do site alvo, em seguida enviou requisição com uma apóstrofe para as páginas encontradas no passo anterior. Depois a ferramenta enviou uma nova requisição contendo o valor "1" para as mesmas páginas. Finalmente, foram comparadas as respostas das duas requisições anteriores e verificado que houve diferença, pois uma das respostas retornou uma mensagem de erro do banco de dados e o ataque não pode ser feito por este cenário.

Considerando as três aplicações inspecionadas, o *scanner* comercial detectou 3 XSS, 4 SQL Injection e 15 CSRF. Os resultados foram validados manualmente e 2 vulnerabilidades XSS e 12 CSRF foram detectadas corretamente, as demais 8 vulnerabilidades eram falsos positivos. Além dessas vulnerabilidades, outras 9 novas

vulnerabilidades foram detectadas quando os casos de teste baseados na árvore de ataque foram aplicados. Essas novas vulnerabilidades se configuram como falha na cobertura da ferramenta comercial. A Tabela 1 apresenta estes resultados.

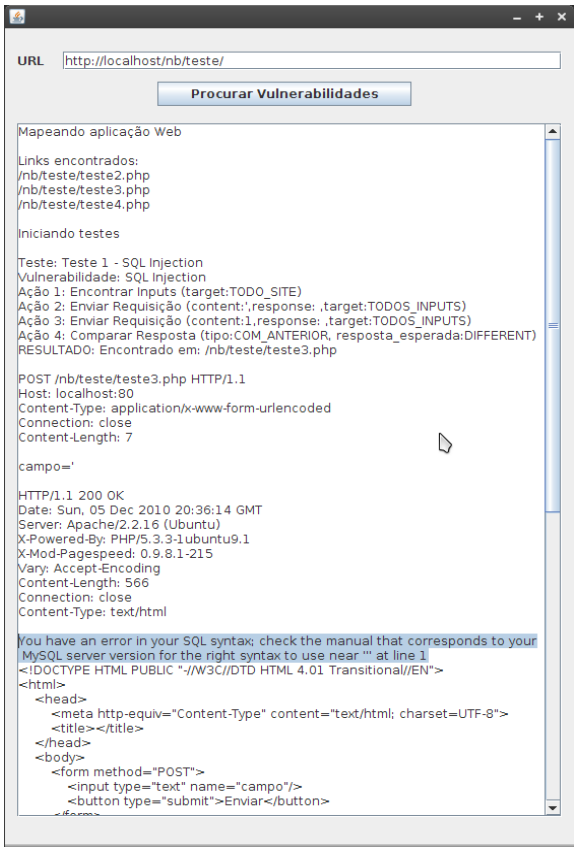


Figura 10. Interface da *J-Attack* – *Execução de ataque SQL Injection*

Os resultados por aplicação podem ser encontrados em Fernandes *et al*.(2010) e não foram apresentados aqui por restrição de espaço. A *J-Attack* em seus testes iniciais conseguiu efetuar os ataques planejados nos casos de teste e em casos de falsos positivos acusou a impossibilidade do ataque. Ao reexecutar os teste manuais utilizando a *J-Attack* algumas falhas de implementação foram encontradas e, neste momento, estão sendo corrigidas para gerar uma versão que se possa disponibilizar para uso de terceiros.

Tabela 1. Resultados dos Testes por tipo de vulnerabilidade

	XSS	SQLInj	CSRF	Total
Detectada pelo *scanner* comercial	3	4	15	22
Detectada corretamente	2	0	12	14
Falso positive	1	4	3	8
Novas vulnerabilidades (falha de cobertura)	0	0	9	9

7. Considerações e Trabalhos Futuros

A ferramenta desenvolvida possibilita a automatização de testes de vulnerabilidades de segurança em aplicações *Web*. A versão inicial da ferramenta trabalha apenas com algumas ações básicas, mas já é capaz de cobrir muitos tipos de testes. A ferramenta

apresenta uma grande flexibilidade devido à sua operação de acordo com o banco de dados. O banco de dados pode ser preenchido com vários tipos de ataques e preferencialmente com uma grande quantidade de variações, pois isto é importante para ataques como XSS e SQL Injection e outros tipos de ataques que usam a injeção como meio de ataque.

A ferramenta foi desenvolvida tendo como objetivo a escalabilidade. Sendo assim, é possível expandir a ferramenta para que ela passe a cobrir mais vulnerabilidades de segurança. Novos tipos de ações podem ser definidos e adicionados ao padrão Strategy do projeto. Novos testes e vulnerabilidades podem ser adicionados ao banco de dados.

A interface gráfica da ferramenta se mostrou bastante simples, facilitando o uso, porém pode ser melhorada e mecanismos de parametrização de execução podem ser adicionados, possibilitando a limitação de execução para testes específicos, por exemplo.

A adição de uma sessão de controle do banco de dados na ferramenta também seria interessante para evitar a necessidade de manipulação de dados diretamente no sistema gerenciador de banco de dados. A ferramenta também precisa ser testada e validada com outras aplicações *Web*.

Referências

Acunetix (2007), "70% of websites at immediate risk of being hacked!" Disponível em http://www.acunetix.com/news/security-audit-results.htm, acesso março/ 2011.

Acunetix (2011), "Acunetix Web Vulnerability Scanner" Disponível em http://www.acunetix.com/vulnerability-scanner/, acesso março/2011.

Amadeus (2010). "Portal do software público brasileiro". Disponível em http://www.softwarepublico.gov.br/ver-comunidade?community_id=9677539. Último acesso em dezembro/2010

Auger, R. (2010) "The Cross-Site Request Forgery (CSRF/XSRF) FAQ". Disponível em http://www.cgisecurity.com/csrf-faq.html, acesso novembro/2010.

Basso, T., Fernandes, P.C.S., Jino, M., Moraes, R. (2010) "Analysis of the Effect of Java Software Faults on Security Vulnerabilities and Their Detection by Commercial Web Vulnerability Scanner Tool". In: Proceedings of IEEE DSN Workshops (DSNW), Chicago, EUA.

Bau, J., Bursztein, E., Gupta, D, Mitchell, J. (2010) "State of the Art: Automated Black-Box Web Application Vulnerability Testing". IEEE Symposium on Security and Privacy, Oakland, USA. Páginas 332-345.

Beck, K. (2002) "Test Driven Development: By Example". (5ª edição). Addison-Wesley Professional.

Browser Scope Project (2010) "Security Test Results". Disponível em http://browserscope.org/?category=security, acesso novembro/2010.

Christey, S. and Martin, R.A. (2007) "Vulnerability type distributions in CVE", Technical Report 1.1, vol. 10, p.04, MITRE Corporation, May 2007.

Eberom (2010). “A CRM and Project Management Tool”. Disponível em http://sourceforge.net/projects/eberom/. Último acesso em dezembro/2010.

Fernandes, P. C. S., Basso, T., Moraes, R., Jino, M. (2010) “Attack Trees Modeling for Security Tests in Web Applications” 4th. Brazilian Workshop on Systematic and Automated Software Testing (SAST). Natal - RN, Brasil.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T. (1999) “RFC 2616: Hypertext Transfer Protocol - HTTP/1.1”. Disponível em: http://www.w3.org/Protocols/rfc2616/rfc2616-sec1.html, acesso novembro de 2010.

Fonseca, J., Vieira, M., Madeira, H. (2007) “Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks”, 13th IEEE Pacific Rim Dependable Computing Conference (PRDC 2007), Melbourne, Victoria, Australia.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1995) “Design Patterns: Elements of Reusable Object-Oriented Software”. 1st. Ed. Estados Unidos da América: Addison-Wesley.

Halfond, W. G. J., Viegas, J., Orso, R. (2006) “A Classification of SQL-Injection Attacks and Countermeasures”. International Symposium on Secure Software Engineering – ISSSE 2006, Arlington, Virginia, 2006.

Havij. “ITSecTeam”. Disponível em http://www.itsecteam.com/en/projects/project1.htm, acesso abril/2011.

HP (2011), “HP WebInspect” Disponível em https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-201-200%5E9570_4000_100__, acesso março/2011.

IBM (2011), “IBM Rational AppScan” Disponível em http://www-01.ibm.com/software/ awdtools/appscan/, último acesso março/2011.

NTA Monitor (2008), Annual Web Application Security Report 2008, Disponível em http://www.nta-monitor.com, último acesso fevereiro/2011.

OWASP (2010) “Top 10 Project”. Disponível em http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, acesso novembro de 2010.

Schneier, B (1999) “Attack Trees: Modeling Security Threats”, Dr. Dobb’s Journal.

Singhal, A., Winograd, T. and Scarfone, K. (2007) “Guide to Secure Web Services: Recommendations of the National Institute of Standards and Technology,” Report, National Inst. of Standards and Tech, US Dep.of Commerce, 2007, pp. 800–95.

Snake, R. “XSS (Cross Site Scripting) - Cheat Sheet Esp: for filter evasion”. Disponível em http://ha.ckers.org/xss.html, acesso novembro/2010.

SqlHelper “Stored Procedure and Documentation Tool”. Disponível em http://www.pikauba.com/sqlhelp/details.htm, acesso abril/2011.

SqlMap “Automatic Sql Injection and Database Takeover Tool”. Disponível em http://sqlmap.sourceforge.net, acesso abril/2011.

Vieira, M., Antunes, N., Madeira, H. (2009) "Using Web Security Scanners to Detect Vulnerabilities in Web Services". Pract.Exp.Report. DSN 2009, Lisboa, Portugal.

XII Workshop de Testes e Tolerância a Falhas

♦

Palestras Convidadas

Palestras Convidadas

Palestra 1: Connectivity Conditions for the Solvability of Fault-Tolerant Consensus in Asynchronous Unknown Networks.

Profa. Fabíola Greve
Universidade Federal da Bahia
Salvador, Brasil

Abstract: The consensus problem is at the heart of solutions related to the development of modern reliable distributed systems. This talk presents necessary and sufficient conditions under which fault-tolerant consensus become solvable in dynamic systems and self-organizing networks. Those conditions are related to the synchrony requirements of the environment, to the connectivity of the knowledge graph constructed by the nodes in order to communicate with their peers, as well as to the knowledge about global parameters in the system, such as, the total number of participants and the maximum number of node crashes.

Palestra 2: Randomization Can Be a Healer: Consensus with Dynamic Omission Failures.

Prof. Miguel P. Correia
Instituto Superior Técnico
Universidade Técnica de Lisboa
Lisboa, Portugal

Abstract: Wireless ad-hoc networks are being increasingly used in diverse contexts, ranging from casual meetings to disaster recovery operations. A promising approach is to model these networks as distributed systems prone to dynamic communication failures. This captures transitory disconnections in communication due to phenomena like interference and collisions, and permits an efficient use of the wireless broadcasting medium. This model, however, is bound by the impossibility result of Santoro and Widmayer, which states that, even with strong synchrony assumptions, there is no deterministic solution to any non-trivial form of agreement if $n-1$ or more messages can be lost per communication round in a system with n processes. In this paper we propose a novel way to circumvent this impossibility result by employing randomization. We present a consensus protocol that ensures safety in the presence of an unrestricted number of omission faults, and guarantees progress in rounds where such faults are bounded by $f \leq \lceil \frac{n}{2} \rceil (n-k) + k - 2$, where k is the number of processes required to decide, eventually assuring termination with probability 1.

XII Workshop de Testes e Tolerância a Falhas

♦

Sessão Técnica 2

Tolerância a Falhas

Reviewing Dependability Issues and Suitable Solutions for Emerging Wireless Sensor Networks Applications

Carlos Oberdan Rolim[1,3], Edison Pignaton de Freitas[1,2], Tales Heimfarth[4], Carlos Eduardo Pereira[1], Cláudio F. R. Geyer[1] , Armando Morado Ferreira[5], Flávio Rech Wagner[1], Tony Larsson[2]

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

[2]IDE – Halmstad University, Halmstad, Sweden.

[3]Departamento das Engenharias e da Ciência da Computação – Universidade Regional Integrada (URI) – Santo Ângelo, RS – Brazil

[4]Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA) – Lavras, MG – Brazil

[5]Programa de Pós-Graduação em Engenharia de Defesa – Instituto Militar de Engenharia (IME) – Rio de Janeiro, RJ, Brazil

{carlos.oberdan,epfreitas,flavio,geyer}@inf.ufrgs.br,tales@dcc.uflr.br
cpereira@ece.ufrgs.br, armando@ime.eb.br, tony.larsson@hh.se,

Abstract. *Some Wireless Sensor Networks applications can be considered safe critical due to the important impact caused in occurrence of their malfunctioning. Motivated by this issue, a research area in dependability applied to WSN is gaining strength. As highly distributed systems, WSN present a number of issues that have to be handled in order to achieve a dependable behavior. These issues are stressed considering the current sophisticated WSN, composed by static and/or mobile nodes. This paper proposes a review of the main problems and trends in the area of dependability for WSN, aiming to find suitable solutions to the problems faced by emerging WSN composed of static and mobile sensor nodes. Such applicability is analyzed for an application scenario of area surveillance.*

1. Introduction

Wireless Sensor Networks (WSNs) are being employed in a number of applications nowadays [Kuorilehto et al 2005], which reflects the result of advances in the sensor technologies and wireless communications [Mini et al 2004]. An interesting approach in WSN is the combination of different types of sensors in a single network. The different sensors can provide complementary data which are combined to provide the users with information with more aggregated value, as the example provided in [Heinzelman et al 2004]. Within this context, a new trend that is starting to be explored is the combination of static and mobile sensor nodes [Erman et al 2004]. As an example, a surveillance system can be composed of a number of unattended static ground sensor nodes and mobile sensor nodes carried by Unmanned Ground Vehicles (UGVs) or Unmanned Aerial Vehicles (UAVs) [Erman et al 2004]. A problem in this scenario is how to ensure the correctness of the overall system if some sensors fail. This issue is common to all

WSNs and is being considered a challenge by the research community in the dependability area [Koushanfar et al 2002]. Thus, this is a research area with great potential to be explored, as by now not many proposals address the problem.

Wireless communications per se are already subject to a number of circumstances that leads to failures [Mini et al 2004]. Moreover, sensor nodes typically used in WSNs are very constrained in resources, fact that has its rationale in the attempt to keep both their size and cost as small as possible [Koushanfar et al 2002]. Having these resource constraints in mind, ordinary Triple Modular Redundancy (TMR) based solutions for the sensor hardware are out of consideration. One may argue that if the hardware used in sensor nodes is cheap; providing redundancy of something cheap would still result in something cheap. This is a misleading idea, because the sensor nodes are cheap because they use only the minimum set of components necessary to their operation. Once these components are duplicated or triplicated, the statement that the sensor node is cheap may not hold. Besides the fact that the redundant hardware would require space, compromising the requirement of keep them in a small size. These facts make WSNs very susceptible to faults in the individual nodes, which if not properly handled, may lead to catastrophic consequences depending on the application.

Conversely, besides the mentioned fragilities that WSNs present, which make them susceptible to failures, they also count with a feature that makes them able to be robust and fault tolerant. This characteristic is related to the great number of nodes used in WSN deployments. This fact provides support to redundancy, which can be used to overcome faults in the sensor nodes that lead to errors in the communications which finally can manifest a system failure in the final service that the network delivers.

Still important to consider in this context is the fact that many WSNs are autonomous systems. This fact makes part of both the problem and the solution in terms of dependability, at the same time. On one hand, as autonomous (complex) systems, equipped with decision making mechanisms, they are able to operate without human intervention. However, a concern about dependability is raised exactly because of these decision mechanisms, which may contain faults that trigger the rest of the error-failure-fault-error… chain compromising the system functionality [Avizienis et al 2001]. On the other hand, autonomous WSNs are able to provide self-healing features and thus recover from failures of individual nodes, handled as faults from the system perspective. This allows the system deliver correct functionality, in spite of presence of faulty nodes.

This paper proposes a review of the main problems faced in the operation of WSN problems that may compromise the system availability and reliability. These two components of the dependability concept are considered the primary ones for WSNs, as stated in [Taherkordi et al 2006], and represent a great potential threat to emerging WSN applications. Additionally, it presents prominent approaches to address some of these related issues and an analysis of their applicability to a surveillance system. Section 2 presents the surveillance system considered in the present study. In Section 3 an overview of the considered dependability problems in WSN is presented. Section 4 provides a small survey of outstanding approaches that propose solutions to handle these problems, focusing on the fault detection aspect, while Section 5 discusses the applicability of a selected approach to the presented system. In Section 6 the conclusions are drawn and the future works are outlined.

2. Application Scenario: Area Surveillance System

A surveillance system considered in this study is composed of two main types of sensors: static on ground and mobile in the air (UAV-carried, from now on just called "UAV"). There are G static ground sensor nodes spread on the area of interest, which are individualized by an identifier sn_i, (i = 1,…, G). The ground static sensor nodes are spread according to a given distribution, which can be random or uniform following a defined pattern. N UAVs fly over the area, following a random or predefined movement pattern. These UAVs are identified by u_i (i = 1,…, N). G, the number of ground sensor nodes, is assumed to be much bigger than the total number of UAVs, i.e. N. The two groups of sensor nodes are further divided according to their sensing capabilities. The considered capabilities are the type of sensing measurements that they can perform and the sensing range. For the ground sensor nodes, the considered measurements are, for instance, vibration, temperature, humidity, difference in the magnetic field, and acoustic signature. Examples for the UAV-carried sensors are visible light cameras, infrared cameras, and SAR/ISAR radars. The sensing range is a tunable parameter that remains constant for the ground sensor nodes, but may vary for the UAV-carried ones, according to the sensing device. The sensor nodes wirelessly communicate with each other within a given tunable range, called communication range. Due to the broadcast nature of wireless communications, all nodes inside the range of another receive a message issued by the sender node.

The system works as follows: The ground sensor nodes are configured to detect possible targets, which is defined by a set of threshold levels of its measurements. When the acquired measurements reach a configured threshold level, a "match" with the detecting criteria is achieved. In the occurrence of a match, the sensor node issues an alarm, which is received by all nodes in its vicinity that are within its communication range.

Alarms are represented by single communication packets. These packets contain a timestamp, the position of the issuer node, and the type of the possible target. The two first components of the alarm allow its unique identification, avoiding alarm duplications. For the purposes of this work, each alarm indicates one target. This means that if it is a group of persons or vehicles, they are handled as a unique entity.

The main elements of the scenario and the described system are presented in Fig. 1. In the figure it is possible to observe the occurrence of a detection of a possible target by a ground sensor node. This node issues an alarm that is received by all other nodes in its communication range. One of these neighbors relays the alarm which is received by a UAV, characterizing the alarm delivery.

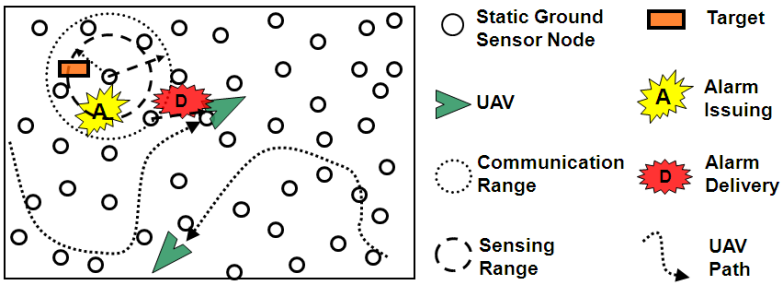


Figure 1. Overview of the application scenario.

By the occurrence of an alarm, the goal of the system is to allocate one of the UAVs, which are equipped with more sophisticated sensors, to fly towards the area where the alarm was issued, gather further information about the possible target, and confirm it as a target, i.e. an intruder or a threat.

Due to an uncontrolled or even harsh or hostile environment, static on ground sensors can become faulty and unreliable. A faulty sensor node can propagate erroneous collected data and affect the correctness of overall system. Another possible problem is that faulty nodes may affect the alarm forwarding. Therefore, the surveillance system needs to perform the identification of fault nodes. This is not a trivial task and encompasses the selection of a suitable algorithm that can be applied in the current scenario.

In next section the background information for the dependability problem in WSN is presented to support the selection of the algorithm further discussed.

3. Dependability Issues in WSN

The notion of dependability is broken down into six elementary component properties [Avizienis et al 2001] as follows: a) Availability – the system will be operational when needed; b) Reliability – the system will keep operating correctly while being used; c) Safety – the system operation will not be dangerous; d) Confidentiality – there will not be disclosure information when not authorized; e) Integrity – there will be no unauthorized modification of the information used by the system; and f) Maintainability – the capability to perform a successful repair action within a given time.

Talking about WSN, the two first aspects of dependability are a must [Taherkordi et al 2006]. It does not mean that the other ones are not important, but these two are the ones that concentrate the focus of attention of the research community, as shown in the literature [Taherkordi et al 2006]. Some of the others aspects correspond to complete different research areas, which is what happens for instance with confidentiality. On the other hand, maintainability may be also considered as a "hot topic", as there are several proposals that aim at reprogram the entire network, which can be seen as a way to fix the software pre-installed in the sensor nodes, such as [Fok et al 2005] [Liu et al 2003]. However, they do not talk about dependability issues in their work, so it is possible to say that this capability is a "side-effect" of such proposals, even its application with this goal is definitely reasonable and valid.

Most of WSN applications should run continuously and correctly during their operation. They are characterized by being mission critical, complexly structured, composed of components that have significantly fault rates, such as semiconductor integrated circuits-based systems and sensor devices, besides their software part represented by the decisional mechanisms [Koushanfar et al 2002]. All these together reveal a big threat to WSN operation, thus compromising the system availability and reliability, supporting the strong interest by and focus on these two topics.

A WSN-based system is basically composed of four elements: 1) the individual sensor nodes; 2) the network formed by these nodes; 3) the sink node(s); and 4) the backend system. The first two elements are the essential parts of a WSN, while the third is the way the data collected by the sensor nodes can be accessed from the outside of the WSN, and finally the fourth is the information system that presents the data to the final user. In this work the considered problems are only the faults that occur in the first two.

Problems related to the sink(s) and the information systems are out of the scope, even considering that they are important parts of the system.

3.1. Faults

Within the delimited scope, the source of faults in WSN can be then classified as: 1) Node Faults and 2) Network Faults.

3.1.1. Node Faults: Node faults are considered to be internal faults that occur in individual sensor nodes. These faults may have their source in the hardware or in the software component of the nodes. In the hardware, common problems are related to antennas that are fragile and cannot resist to impacts [Langendoen et al 2006] and to batteries that provides incorrect voltage or current outputs damaging other internal components [Tolle et al 2005].

In spite of hardware faults representing an important problem, software ones are very common and may even exceed the first ones, depending on the complexity of the WSN application. As deeply embedded systems, sensor nodes are hard to program due to the inherent resource constraints of such systems. Incautious programming of sensor nodes may easily lead to memory overruns, buffer overflows, deadlocks and to all sorts of trick and hard to debug programming problems. Incorrect handling of data provided by the sensor unit can be caused by faults in the application programs. These faults in one node may trigger a fault in data aggregation software in other nodes, as the WSN in general run distributed applications.

3.1.2. Network Faults: Network faults are those related to the interaction among nodes and that lead to an error that affects the nodes' communication and coordination. One of the very common sources of fault is radio interference. This problem may occur by obstacles present in the environment, or by the presence of other devices operating in similar frequencies, or even by other nodes of the network displaced close by. In fact this last case, even a legitimate emission by a neighbor node may cause undesired interferences when frequent collisions of messages happen. Link stability is also reported as a problem in sensor nodes networking. This instability affects the sensor nodes routing capabilities, which per se is another source of problems.

Routing issues can also be considered as a significant source of networking malfunctioning. In WSN, routing has a great importance, specially related to the semantics of the applications that run over the networks. Faults in the routing protocols may compromise the network as a whole, or parts of it, in a number of ways (e.g. the wrong cluster-head election in hierarchically organized WSN due to delayed, dropped or misrouted messages).

It is important to observe that this classification complies with the fundamental concepts in dependability presented in [Avizienis et al 2001]. For example, the software faults discussed above, such as buffer overflow, can be classified as a fault due: Development; Internal; Human-made; Software; Non-Malicious. It can be classified as presented because it is an internal part of the system, a software part, programmed by a person that did not consider checking the input value, so it was human-made introduced, and not necessarily malicious. It could be malicious if the programmer intentionally considered the hypothesis of such condition happens, nevertheless did not introduce the value checking. However, this work focuses on the sensor network perspective. This means that the categorization of the faults tries to emphasize the elements present in

these systems, and it is why it does not mention much the classification presented in [Avizienis et al 2001].

3.2. Errors

Errors are deviations of the normal way that a part of a system should work, and they are activated by faults. The types of considered errors in this work follow the consequences of the faults presented above, as they are the ultimate responsible for the occurrence of the errors.

As an example of an error caused by a node fault, it can be mentioned the miscalculation of a value based on wrong data provided by the sensor device. The sensor device may provide an unexpected value, which is out of the range of tractable values for the piece of software that manipulates this data. As a result, an erroneous output is generated being a manifestation of two original faults: the former is a hardware fault, related to the sensor device that provided the wrong value; and the latter is the fault due to lack of protection in the program code against out of bounds values, which leads to an erroneous state.

Errors due to network faults can be exemplified by selection of wrong routes to forward messages due to routing faults. The consequence of these faults can be erroneous data aggregation, for example, or undelivery of expected messages. Malformed or incomplete communication packets can be presented as examples of errors due to interferences.

3.3. Failures

A failure represents an incorrect delivery of a service provided by a system. Originally motivated by one or more faults, it is the external manifestation of an error.

In WSN a failure could be considered the final erroneous information provided to the end user, which is ultimately the desired service from a WSN. However, for the scope of this work, the considered failures are related to the nodes and the network, as discussed above and classified according to [Souza et al 2007]:

Crash or Omission Failures: These are failures in which the nodes or parts of the network stop to respond to requests, e.g. due to the exhaustion of nodes' batteries.

Timing Failures: They are failures that occur due to mismatch between the timing requirements specified for the WSN services and the actual observed timing behavior.

Value or Semantic Failures: These types of failures represent deviations in the expected outputs of the system in terms of the content of data or information.

Arbitrary or Byzantine Failures: These are types of failures that do not follow a consistent pattern, such as an intermittently correct and erroneous sensor output.

4. Prominent Approaches to Handle Dependability in WSN

The study about dependability in WSN is gaining importance as the applicability of this technology is increasing, but there is still a gap between problems and solutions when it comes to assure dependability in WSN operation.

This section presents some selected prominent approaches to deal with dependability problems in WSN. The bibliography review presented here focuses on the

fault detection and forecast techniques. Fault recovery and removal are for sure important, but the goal of the paper is kept in the mentioned subject. The idea is analyze to suitable fault detection and/or forecast techniques to be used in emerging applications of WSN. It does not mean that other existing techniques, especially for fault recovery, could not be considered to be (re)used. However, they are kept as related works instead.

The works described in this section are classified following the taxonomy presented in [Souza et al 2007].

Automatic fault detection and forecast techniques have the goal to provide a system with the ability to assess if its functionalities are correct and if it will continue working in the future. Some proposals provide both techniques together.

Following a similar reasoning which based the classification of faults according to node or network ones, the analysis of the fault detection techniques presented here considers also the elements involved in this process. This is related to the considered granularity. Faults detected by the nodes themselves are classified as self-diagnosis. Fault detection performed by the interaction of nodes is classified in two classes: group detection and hierarchical detection. The former consists of mechanism in which a number of nodes are used to monitor other node(s). The latter considers the usage of a hierarchical detection tree, among other mechanisms, which shifts the performance of the detection to more powerful nodes outside the WSN, for instance a sink. Group detection techniques are especially interesting for the type of WSN target of this work, so more attention will be given to this type of technique.

4.1. Self-Diagnosis

Self-diagnosis techniques consider the use of the nodes themselves to detect and in many proposals forecast the occurrence of a fault. These techniques consist of performance of tests in the system elements so that the faults can be detected. In [Hart et al 2005] tests using accelerometers are used to detect possible faults due to impacts.

An important source of node failure is energy exhaustion. Keeping track of the energy resources may indicate the faults in the nodes functionalities. In [Rakhmatov et al 2001], an estimation of the nodes' batteries lifetime is performed. This approach uses the measurement of the current battery voltage, analyzing the discharge curve and the current rate to determine estimation for the battery exhaustion.

In [Chen et al 2006] nodes are able to detect faults in their sensor devices and inform other nodes in the network that they are not able to perform sensing services anymore. However, as their transceivers and batteries are still working, they are able to perform routing, and provide forwarding of incoming packets.

4.2. Group Detection

A technique to detect faulty sensors in a network of mobile sensors used for target tracking is presented in [Kim et al 2008]. The technique is used to identify faults in sensors readings when estimating the targets positions. In order to detect the nodes that are providing an incorrect target position, two algorithms run concurrently: the former is a semi-decentralized dynamic data fusion, which employs a median-consensus using non-faulty nodes to provide the estimate target position, while the latter one, is a fault detection algorithm, which uses the estimation provided by the former to decide if a given node should be considered as faulty or not.

Clustering is an important mechanism largely used in WSN. In [Taherkordi et al 2006] a mechanism to detect faulty nodes in clustered WSN is proposed. This approach uses an event-based middleware software architecture for the services provided by the nodes, and distinguishes those services according to the hole of a node in the network, i.e. cluster-head or cluster member. The former are considered event-brokers while the latter are event-sources. By using a publish-subscribe scheme for the nodes' interaction, the proposal is to detect node disconnection faults due to node mobility or energy depletion. Additional mechanisms make possible fault recovery during reconnections.

Exploring the fault detection abilities of grouped nodes, [Chen et al 2006] proposes an approach in which the nodes in agiven area test each other to detect the occurrence of faults. According to the specific deployments, the nodes generate test results for all their neighbors. Based on this information, they decide about their own tendency. Then a consistency check is performed to decide about the obtained results. This proposal allows the detection of faults in the sensing measurements, which generate erroneous values, but also faults that lead to crash failure.

ASFALT (A Simple Fault-tolerant signature-based Localization Technique for emerging sensor networks) [Jadliwala et al 2007] proposes a fault-tolerant location mechanism for sensor networks. The work as whole deals with problems from nodes' deployment to the reduction of the error in signature-based location algorithms. The part related to the fault detection is performed in a group fashion, in which a group head keeps the information about the health status of the nodes in its group. A group with healthy nodes is capable to deliver correct values to solve the location of a target. Group heads exchange their status among them via a periodic broadcast. A fault is detected when less than three groups are declared healthy to solve the location of a given target.

In [Ni et al 2009] a Bayesian model to detect erroneous data provided by faulty sensor nodes is presented. The approach is divided in two phases, in which the former uses a Bayesian MAP criterion to determine a set of aggregation node subsets, which will perform the decision in the latter whether a given data is erroneous or not. Suspicious data is tagged as so, and by passing through the aggregating nodes, the audition of its content is done. As the proposal is highly computing intensive, the authors relaxed the timing requirements providing a semi-realtime approach, in which the data integrity audit is performed much less frequently than the sensor samples are taken.

Neural networks are used to identify the misbehavior of sensor nodes in [Obst 2009]. The authors propose a neural network training based on spatially organized distributed computation in which a local interaction among a group of nodes is explored. The training is performed in two steps: 1) a sample matrix of internal network states and the corresponding output matrix with activations are used; and 2) the output weights are computed. The fault detection is done by the comparison of the predicted values provided by the neural network and the actual readings. If the difference between them exceeds a threshold, a fault it detected.

4.3. Hierarchical Detection

A fault detection mechanism using management architecture called MANNA is proposed in [Ruiz et al 2004]. This work uses a manager located externally to the WSN, which has the global view of the entire network. This vision allows the execution of

complex management tasks. Every change in the nodes status is informed to the manager, which is responsible to test the sensor nodes by sending commands to them. As drawback, besides the centralized manager, this approach has a great overhead due to the number of communications among the sensor nodes and the manager.

In [Ringwald et al 2006] a debugging tool that uses a decision binary tree is proposed. Verifications in the tree are performed so that possible faults can be detected. The authors claim that the one of the major pros of their approach is the fact that no additional traffic has to be transported among the sensor nodes to provide the detection.

A detection tree in which the child nodes report to their parents and these continuously forward up to the sink is presented in [Rost et al 2006]. When a node receives the status of its child nodes, it performs an aggregation with its own status and forwards the result. This mechanism continues until the information arrives to the sink.

In [Chitnis et al 2009], a study proposing the use of some powerful nodes together with resource constrained ones in a WSN is analyzed. This proposal uses such a heterogeneous composition of the WSN to improve reliability of data aggregation, by the use of the powerful nodes to detect and recover from faults. The authors compare a detection aggregation tree structure to a hybrid one in terms of cost-benefits and performance, regarding faults. Their conclusions show benefits in using a number of expensive powerful nodes to achieve the desirable behavior against faults.

5. Case Study – Analysis of Suitable Approaches

This section presents the used fault model and the algorithm that has these characteristics and can be used in the proposed problem-scenario presented above.

5.1. Network and Fault Model

The model used in this work considers a scenario in which the ground sensor nodes are deployed in a rectangular area where they are randomly distributed.

The instance of the model studied in this paper considers an area of 10 Km x 10 Km, and sensor nodes with 350 meters communication range. For these parameters, the minimum number of nodes that provide a connected network is 5000. This is a requirement to guarantee the correct system functionality, as discussed in [Freitas et al 2010]. This number is obtained by using:

$$P(d_{min} > 0) = \left(1 - e^{-\rho\pi r^2}\right)^n \quad (1)$$

where $P(d_{\min} > 0)$ is the probability that the minimum degree of the network is higher than zero, "ρ" is the node density, "r" is the communication range, and "n" is the total number of nodes in the network.

The formula presented in (1) determines the probability that a network of nodes randomly deployed with independent uniform probability (homogeneous Poisson point process in two dimensions, which generates a geometrical random graph) has a degree greater than zero, which means that each node in the network has at least one node via which it may communicate with any other node in the network [Bettstetter 2002]. Thus each node can locate neighbors within its transmission range to infer about faulty ones.

To attend an alarm occurrence the system can allocate one of six UAVs of three different types, equally distributed, that patrol the area, flying at altitudes not higher

than 250 meters and with speeds from 100 Km/h up to 120 Km/h. The UAVs have a communication range of 1.5 Km.

This work considers two different types of failures for the ground static sensor nodes: crash failures and value failures. The former considers that the node is permanently enable to perform its activities which can be detected, for instance, if it does not respond any request for more than a certain time period. The latter considers that a node is providing erroneous sensor readings and thus is not trustable to detect a possible event of interest and issue a corresponding alarm. For this second type of considered failure, the node is still capable to route messages, so incoming messages with alarms from other nodes can still be forwarded via this node.

For each of the considered failures it is established a possible situation. A number of such failures occur in a given region of the network, so that it creates an island of faulty nodes. For the first type of failure, it results in incomplete absence of the network in that region, as the nodes cannot even forward alarms. For the second one, the network still exists, but the sensing ability is out of order, so the system functionality is broken.

The goal is to use a fault detection mechanism that is able to provide the information about the occurrence of such situations, so that fault recovery measures can be applied. This is not a goal of this paper to deeply analyze such recovery measures, but as an example of a possible solution can be the use of the mobile sensors, the UAVs, to cover the problematic areas. The nodes that detected such problems can issue an alarm calling a UAV to fly close by the faulty region in order to cover the region, or at least make the UAV have that region as a preferred flying zone. It is important to highlight that the aim is to preserve the system functionality in providing the final user information about possible targets. Thus, the use of the UAVs to cover the areas with faulty nodes is a reasonable solution. However, this paper just focuses on fault detection on ground static sensors and this mentioned possible solution was given as an example.

5.2. Applied Fault Group Detection

After presenting prominent approaches in dependability research area for WSN and the considered model for this research work, the proposal is to use group based fault detection to handle the problems of the scenario depicted above in Section 2. This choice is due to the large amount of static sensor nodes on the area where the system is deployed, in which each node is able to locate neighbors within its transmission range and use its results to infer about faulty neighbor nodes.

The scenario also requires that the fault detection needs to be conducted in a real-time mode, with low latency, high throughput and, due to the limited energy resources of the sensor nodes, the detection of faulty nodes cannot be expensive in terms of energy. Such restriction excludes centralized approaches in which a base station is used to collect information from all sensors and calls for a distributed and more generic algorithm that explores local decision taken by the sensor nodes. A suitable solution with the desired distributed characteristics is an algorithm presented in [Chen et al 2006].

The proposed algorithm is depicted is Figure 2. It uses five steps to infer about status of grouped sensors. In step one, every sensor S_i tests the set of its neighbors $N(S_i)$. This test is done using measurements of each sensor´s neighbor with two

variables, d_{ij} and Δd_{ij}, and two predefined threshold value $\theta 1$ and $\theta 2$. If the modulus of measurement difference between S_i and S_j at a time t is greater than threshold $\theta 1$ then the historical data is used to find if the current measurement changes over the time significantly from previous measurement. In affirmative case, it is more likely the sensor is faulty. If this variation of measurement is greater than threshold $\theta 2$ the test results in 1 (indicating that S_i and S_j are most likely in different status) otherwise in 0 (indicating that either both S_i and S_j are most likely good or faulty). In step two S_i generates a tendency value T_i based upon its neighboring sensors' test value. If the sum of test results is less than half of the number of its neighbors, the sensor is likely good (LG). Otherwise, it's likely faulty (LF). To finish this step T_i is communicated to neighbors. In step three, the number of the LG sensors with coincident test results determines whether the sensors are good (GD) or faulty (FT). If the summation of LG sensors is greater or equal than half of the number of its neighbors then T_i is set to GD and so T_i is communicated to neighbors. In step four GD sensors are used to determine status of the remaining undetermined sensors. Finally, step five does a validation check to make sure the diagnosis is consistent throughout the entire network. The next subsection presents the analysis of the application of this algorithm to detect faults in the surveillance system presented in Section 2.

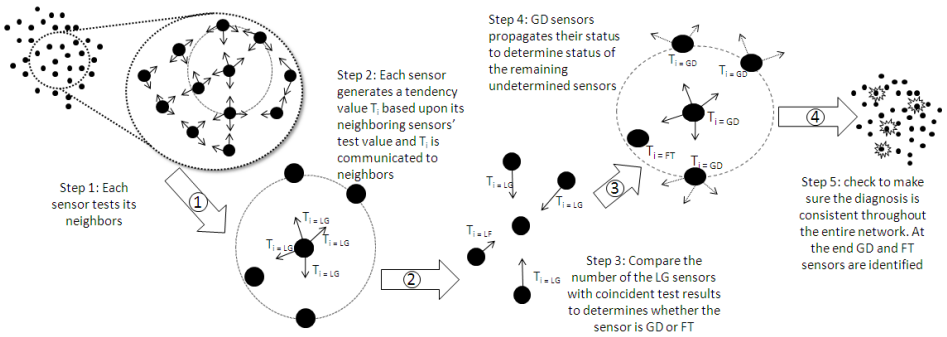


Figure 2. Overview of the Fault Detection Algorithm.

5.3. Analytical Evaluation

Based on the results presented by the authors of the technique in [Chen et al 2006], and on the results about the proposed surveillance system degradation by a non-sufficient number of ground sensor nodes according to (1) [Freitas et al 2010], it is possible to analyze the applicability of the proposed fault group detection to this particular system.

The results presented in [Freitas et al 2010] analyze an instance of the network with the same parameters as described above. The achieved results showed that with a number below 50% of the ideal number of nodes, the system performance becomes really degraded, approaching values below 60%, and dropping fast from this point to lower number of nodes. For a number of nodes above 50% of the ideal amount, the system responds with values above 70% of efficiency, which can be considered a fairly good result. Table 1 presents this data. This data emphasizes the problem in losing sensor nodes, especially from the level of 50% of the ideal number of nodes. Assuming the first type of the considered failures, the node crash, a detection of a faulty region should be triggered when a group of nodes detects in their vicinity a number of approximately 50% of faulty nodes. For the parameters presented in the example instance of the considered system, each node has in average 18 neighbors, which is a

value deduced from the average number of nodes in a region within an area of the circle determined by the communication range of a sensor node, calculated by:

$$n = \left({}^{N}/_{A}\right) \cdot \pi r_c^2 \quad (2)$$

Where n is the average number of nodes in a region with area equivalent to the circle determined by the communication range of a sensor node, N is the total number of nodes distributed in the area, A is the considered area, and $r_c$is the communication range of sensor node. From the achieved value, it is subtracted one unit, which is the reference node in the center of the circle and the result is the number of its neighbors.

Table 1. Relation Among Number of Nodes, P(d_{min} > 0) and System Efficiency

Number of Nodes	% of Nodes in Relation to the Ideal Number	$P(d_{min} > 0)$	Normalized System Efficiency in Average
5000	100%	1.0	85%
4500	90%	0.999863268	85%
4000	80%	0.999168572	85%
3500	70%	0.995031912	83%
3000	60%	0.971209180	76%
2500	50%	0.846539727	65%
2250	45%	0.675534269	53%
2100	42%	0.521033789	39%
2000	40%	0.401629927	34%
1500	30%	0.009203929	<10%

Considering the area covered by the neighbors of these neighbors, the number of nodes may come to around 70. This number may vary depending on the nodes' distribution, but in average this is a good approximation for the simulation parameters mentioned above. Considering a sensing range of 100 meters, which is a realistic value considering available sensor technology nowadays [Wilson 2005], this amount of nodes covers a representative part of the total area, and as so, a failure in covering it, may result in a degraded system functionality. For this concrete case, applying the approach presented in Section 5.2, which was proposed in [Chen et al 2006], if a number of faulty nodes above 35 is detected in a given region (50% of the nodes of the considered area), a fault can be notified. This threshold was established based on the fact that when a percentage below 50% of the total number of nodes in the network the functionality is seriously degraded, the same holds for a part of the network.

Considering de second type of failure, the same reasoning described above can be applied. Even if the nodes are still able to communicate; they are only able to forward alarms from other regions. This means that the region where they are located will become without sensing coverage. Thus, even that the alarm delivery mechanism is not affected, the detection of possible targets is, which indeed represents a malfunctioning of the system.

6. Conclusions and Future Work

This paper proposes a review of the efforts in handling dependable issues in WSN, and their applicability in emerging WSN applications. A review of important concepts was presented, along with a survey of prominent works in the area. Finally a highlight on

one of them was done, which was deeply described and had its applicability analyzed to an application scenario of a surveillance system.

As future works, other mechanisms have to be combined in order to address the different types of nodes that compose the network. In the presented application, this difference was the nodes' mobility. A second issue is the need for a more accurate metrics to determine the dependability qualities of the proposed solutions. Finally, more than combined solutions, unified solutions that consider different types of sensors need to come up to address these types of emerging applications.

References

Kuorilehto, M., Hännikäinen, M. and Hämäläinen, T.D. (2005) "A Survey of Application Distribution in Wireless Sensor Networks", EURASIP Journal on Wireless Communications and Networking, 38(5), p.774-788.

Mini, R., Loureiro, A., Nath, B. (2004) "The Best Energy Map of a Wireless Sensor Network", In: Proceedings of the 7th ACM Intl Symposium on Modeling, Analysis and Simulation of Wireless and Mobile System, Venice, Italy, p.165-169.

Heinzelman, W. B., Murphy, A. L., Carvalho, H. S. and Perillo, M. A. (2004) "Middleware to support sensor network applications, Network", IEEE, vol. 18, no. 1, p.6-14.

Erman, A.T., Hoesel, L. and Havinga P. (2008) "Enabling Mobility in Heterogeneous Wireless Sensor Networks Cooperating with UAVs for Mission-Critical Management", IEEE Wireless Communications, vol. 15, is. 6, p.38-46.

Koushanfar, F., Potkonjak, M. and Sangiovanni-Vincentelli, A. (2002) "Fault Tolerance in Wireless Ad-Hoc Sensor Networks", IEEE Sensors, vol. 2, p.1491-1496.

Avizienis, A., Laprie, J. and Randell, B. (2001) "Fundamental Concepts of Computer System Dependability", IARP/IEEE-RAS Workshop on Robot Dependability: Technological Challenge of Dependable Robots in Human Environments, Korea.

Taherkordi, A., Alkaee Taleghan, M. and Sharifi, M. (2006) "Dependability Considerations in Wireless Sensor Networks Applications", In: Journal of Networks, v.1, n. 6, Academy Publisher.

Fok, C.L., Roman G.C. and Lu, C. (2005) "Rapid Development and Flexible Deployment of Adaptive Wireless Sensor Network Applications", In: Proceedings of the 24th International Conference on Distributed Computing Systems, p.653-662.

Liu, T. and Martonosi, M. (2003) "Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems", In: Proceedings of the 9th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, p.107-118.

Langendoen, K., Baggio, A. and Visser, O. (2006) "Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture", In IPDPS 20th International Parallel and Distributed Processing Symposium.

Tolle, G., Polastre, J., Szewczyk, R., Culler, D., Turner, N., Tu, K., Burgess, S., Dawson, T., Buonadonna, P., Gay D., and Hong, W. A. (2005) "Macroscope in the Redwoods", In: SenSys '05: Proceedings of the 3rd international conference on embedded networked sensor systems, New York, NY, USA, p. 51–63.

Souza, L. M. S., Vogt, H. and Beigl M. (2007) "A Survey on Fault Tolerance in WSN", Interner Bericht. Fakultät für Informatik, Universität Karlsruhe.

Harte, S. and Rahman A. (2005) “Fault Tolerance in Sensor Networks Using Self-Diagnosing Sensor Nodes”, In: IEEE International Workshop on Intelligent Environment, p.7–12.

Rakhmatov D. and Vrudhula S. B. (2001) “Time-to-Failure Estimation for Batteries in Portable Electronic Systems”, In: Proceedings of the 2001 International symposium on Low power electronics and design, p.88-91.

Chen, J., Kher, S. and Somani A., (2006) Distributed fault detection of wireless sensor networks. In Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks, p.65-72.

Kim, Y., Gu, D., and Postlethwaite, I. (2008) “Fault-Tolerant Cooperative Target Tracking in Distributed UAV Networks”, In: Proceedings of 17 IFAC conference, p.8878-8883.

Jadliwala, M., Upadhyaya S. and Taneja, M. (2007) “ASFALT: A Simple Fault-Tolerant Signature-based Localization Technique for Emergency Sensor Networks”, In: Proceedings of 26th IEEE International Symposium on Reliable Distributed Systems, p.3-12.

Ni, K.S., Pottie, G.J. (2009) “Sensor Network Data Fault Detection Using Bayesian Maximum a Posterior Sensor Selection and Hierarchical Bayesian Space-Time Models”, Technical Reports, Center for Embedded Network Sensing, UCLA.

Obst, O. (2009) “Distributed Fault Detection using a Recurrent Neural Network”, In: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks (ISPN'09), San Francisco, CA, USA, p. 373-374.

Ruiz, L. B., Siqueira, I. G., Oliveira, L. B., Wong, H. C., Nogueira, J. M. S. and Loureiro, A. A. F., (2004) “Fault management in event-driven wireless sensor networks”, In: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems, p.149-156.

Ringwald, M., Römer, K. and Vitaletti, A. (2006) “Snif: Sensor network inspection framework”, Technical Report 535, ETH Zurich, Institute for Pervasive Computing.

Rost, S. and Balakrishnan, H. Memento. (2006) “A health monitoring system for wireless sensor networks”, In: Proceedings of SECON.

Chitnis, L. Dobra, A. and Ranka S. (2009) “Fault tolerant aggregation in heterogeneous sensor networks”, In: Journal of Parallel and Distributed Computing, Vol. 69, Is. 2, p. 210-219.

Freitas, E.P., Heimfarth, T., Pereira, C.E., Ferreira, A.M., Wagner, F.R. and Larsson, T. (2010) “Experimental Analysis of Coordination Strategies to Support Wireless Sensor Networks Composed by Static Ground Sensors and UAV-carried Sensors”, In: Proceedings of IEEE International Symposium on Parallel and Distributed Processing with Applications. Taipei, Taiwan, p. 152-161.

Bettstetter, C. (2002) “On the Minimum Node Degree and Connectivity of a Wireless Multihop Network”, In: Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing, ACM, New York, NY, USA, p. 80-91.

Wilson, J. S. (2005), Sensor technology handbook, Vol 1, Elsevier.

Aspectos Práticos sobre o Consenso Bizantino entre Participantes Desconhecidos

Eduardo Adilio Pelinson Alchieri[1], **Luiz Renato Tomelin**[1], **Alysson Neves Bessani**[2], **Joni da Silva Fraga**[1]

[1]DAS, Universidade Federal de Santa Catarina, Florianópolis - Brasil
[2]LaSIGE, Universidade de Lisboa, Lisboa - Portugal

Abstract. *Agreement protocols form the basis for the solution of most problems involving reliable distributed systems. Although the consensus has been widely studied in classic environments, few studies have considered this problem in dynamic and self-organizing systems, where the set of participants are previously unknown. Recently, a solution to Byzantine fault-tolerant consensus with unknown participants (BFT-CUP) was proposed, which identifies the degree of knowledge, about the system composition, necessary and sufficient to solve the consensus. This paper complements these theoretical works by analyzing practical aspects about the BFT-CUP. First, this problem is analyzed in Mobile Ad Hoc Networks through simulations of several realistic scenarios, where we can identify what are the parameters and settings required to solve BFT-CUP with a high probability. Moreover, this paper studies the use of BFT-CUP protocols in Vehicular Ad Hoc Networks, by analyzing the vehicular connectivity of the city of Porto - Portugal.*

Resumo. *Protocolos de acordo formam a base para a solução da maioria dos problemas que envolvem sistemas distribuídos e confiáveis. Apesar de o consenso ter sido amplamente estudado em ambientes clássicos, onde o conjunto de participantes é conhecido, poucos trabalhos consideram este problema em ambientes dinâmicos e auto-organizáveis, onde os participantes da computação são, a priori, desconhecidos. Recentemente foi proposta uma solução para o problema do consenso Bizantino entre participantes desconhecidos, chamada BFT-CUP (Byzantine Consensus with Unknown Participants), a qual busca definir o grau de conhecimento, sobre a composição do sistema, necessário e suficiente para que este problema admita solução. Este trabalho complementa os resultados teóricos obtidos até então e busca analisar aspectos práticos da realização do BFT-CUP. Primeiramente, este problema é analisado em redes MANETs (Mobile Ad Hoc Networks) através de simulações de vários cenários realistas, onde é possível verificar quais são os parâmetros e configurações necessários para que os participantes do sistema consigam resolver o BFT-CUP com uma alta probabilidade. Além disso, também é apresentado um estudo acerca da utilização do BFT-CUP em redes VANETs (Veicular Ad Hoc Networks), através de uma analise sobre a conectividade veicular da cidade de Porto - Portugal.*

1. Introdução

O problema do consenso, Lamport *et al.* [Lamport et al. 1982], e de um modo geral os algoritmos de acordo, formam a base para a solução da maioria dos problemas encontrados no desenvolvimento de sistemas distribuídos confiáveis, pois possibilitam que os participantes da computação distribuída coordenem suas ações de forma a manter a consistência em seus estados e garantir o progresso do sistema. Este problema foi vastamente estudado em redes clássicas, onde o conjunto de processos que participam de determinada computação é estático e conhecido

por todos os participantes do sistema. Mesmo nestes ambientes, o problema de consenso não tem solução determinista na presença de uma única falha, considerando que as entidades se comportam de forma assíncrona [Fischer et al. 1985].

Considerando sistemas dinâmicos e auto-organizáveis, como redes MANETs (*Mobile Ad Hoc Networks*) e VANETs (*Veicular Ad Hoc Networks*), as dificuldades encontradas na elaboração de protocolos para resolver o consenso aumentam. Um conhecimento inicial sobre a composição do sistema é uma premissa que não pode ser adotada nestes ambientes. Assim, os participantes da computação (e os seus conhecimentos sobre a composição do sistema) não podem ser previamente determinados.

Neste sentido, Cavin *et al.* [Cavin et al. 2004, Cavin et al. 2005] realizaram os primeiros esforços para resolver o consenso em redes desconhecidas (ou sistemas dinâmicos e auto-organizáveis), definindo quais são as condições necessárias e suficientes para resolver o FT-CUP (*fault-tolerant consensus with unknown participants*), considerando o conhecimento sobre a composição do sistema e os requerimentos de sincronia (encapsulados em um detector de faltas). No trabalho apresentado em [Cavin et al. 2005], prova-se que para resolver o FT-CUP em um cenário com o mínimo de conhecimento sobre a composição do sistema é necessário que o mesmo seja síncrono. Greve *et al.* [Greve and Tixeuil 2007] comprovam que realmente existe uma relação entre o grau de conhecimento obtido pelos participantes e os requisitos de sincronia, apresentando uma solução para o FT-CUP em sistemas parcialmente assíncronos (mínimo de sincronia necessária para resolver o consenso em sistemas estáticos [Dwork et al. 1988], desconsiderando protocolos que empregam randomização [Aspnes 2003]), onde os participantes devem obter um maior conhecimento sobre a composição do sistema.

Recentemente, Alchieri *et al.* [Alchieri et al. 2008] estenderam este modelo e estudaram o consenso bizantino entre participantes desconhecidos, BFT-CUP (*Byzantine fault-tolerant consensus with unknown participants*), o qual define as condições necessárias e suficientes para resolver o consenso nestes ambientes, considerando que processos podem agir maliciosamente. Estas condições referem-se ao grau de conhecimento obtido pelos participantes e ao modelo de sincronia apresentado pelo sistema. As condições suficientes são representadas por um conjunto de algoritmos capaz de resolver o BFT-CUP dado que o sistema apresente as condições identificadas como necessárias para que o BFT-CUP admita solução.

Este trabalho apresenta uma abordagem prática sobre a realização do BFT-CUP [Alchieri et al. 2008]. Primeiramente, o comportamento dos protocolos que resolvem o BFT-CUP [Alchieri et al. 2008] é avaliado em uma rede MANET de topologia arbitrária, onde o grau de conhecimento teoricamente necessário para resolver este problema nem sempre é obtido pelos participantes. Nossas avaliações são baseadas em simulações onde os parâmetros tanto do BFT-CUP quando da rede MANET são variados. Além disso, este trabalho também apresenta um estudo acerca da possibilidade de utilização do BFT-CUP em redes VANETs, através de uma análise sobre a conectividade veicular apresentada pela cidade de Porto-Portugal.

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta os principais conceitos relacionados com o problema do consenso bizantino entre participantes desconhecidos (BFT-CUP) e descreve brevemente a solução para este problema proposta em [Alchieri et al. 2008]. O entendimento destes conceitos é fundamental para a compreensão das análises realizadas nas seções seguintes. As seções 3 e 4 analisam o comportamento do BFT-CUP em redes MANETs e VANETs, respectivamente. Por fim, a Seção 5 discute alguns trabalhos relacionados e a Seção 6 apresenta as conclusões deste trabalho.

2. O Problema do Consenso em Redes Desconhecidas

2.1. Modelo de Sistema

Considera-se um sistema distribuído composto por um conjunto finito Π de processos retirados do conjunto universo U. Em uma *rede conhecida*, Π é conhecido por todos os processos do sistema, enquanto que em uma *rede desconhecida*, um processo $i \in \Pi$ poderá conhecer apenas um subconjunto $\Pi_i \subseteq \Pi$ (visão parcial do sistema).

Os processos estão sujeitos a faltas Bizantinas [Lamport et al. 1982], onde o limite de falhas f é conhecido. Comunicação entre processos conhecidos dá-se através de canais ponto a ponto confiáveis e autenticados, onde é necessário um protocolo de roteamento tolerante a faltas bizantinas [Kotzanikolaou et al. 2005]. Já a comunicação entre processos desconhecidos é realizada através de redundância de mensagens [Alchieri et al. 2008].

Além disso, os protocolos do BFT-CUP são projetados para sistemas assíncronos mas utilizam um camada subjacente de consenso bizantino clássico que pode ser implementada sobre sistemas parcialmente síncronos [Dwork et al. 1988] (ex., Paxos Bizantino [Castro and Liskov 2002]) ou sobre sistemas completamente assíncronos (ex., *randomized consensus* [Correia et al. 2006, Bracha 1984, Ben-Or 1983]). Desta forma, o BFT-CUP exige o mesmo nível de sincronia do protocolo de consenso clássico utilizado. Nas simulações realizadas neste trabalho, utilizamos o protocolo Paxos Bizantino [Castro and Liskov 2002], considerando um sistema parcialmente síncrono.

2.2. Definição

O problema do consenso consiste em garantir que todos os processos corretos acabarão por decidir pelo mesmo valor, que foi previamente proposto por algum processo. Formalmente, o consenso é definido pelas seguintes propriedades [Chandra and Toueg 1996, Castro and Liskov 2002]:

- *Validade*: se um processo correto decide pelo valor v, então v foi proposto por algum processo;
- *Acordo*: dois processos corretos não decidem por valores diferentes;
- *Terminação*: todo processo correto acaba por decidir por algum valor;
- *Integridade*: todo processo correto decide uma única vez.

2.3. Detectores de Participação

Para resolver qualquer problema distribuído não trivial, é necessário que os processos obtenham um conhecimento ao menos parcial sobre a composição do sistema caso cooperação entre os mesmos seja necessária. Os detectores de participação (PD) são oráculos distribuídos que fornecem informações sobre participantes do sistema. Cada participante i possui o seu detector de participação $i.PD$ e uma consulta a este detector retorna um subconjunto de processos de Π com os quais i pode colaborar [Cavin et al. 2004].

Detectores de participação fornecem um contexto inicial sobre a composição do sistema através do qual é possível expandir o conhecimento sobre Π. Assim, esta abstração enriquece o sistema com um grafo de conectividade por conhecimento. Este grafo é orientado, pois o conhecimento fornecido pelos detectores de participação não é necessariamente bidirecional [Cavin et al. 2004].

Definição 1 ***Grafo de Conectividade por Conhecimento****: Seja $G_{di} = (V, \xi)$ um grafo orientado representando a relação de conhecimento determinada pelo oráculo PD. Então, $V = \Pi$ e $(i, j) \in \xi$ se e somente se $j \in i.PD$ (i conhece j).*

Definição 2 ***Grafo Não-Orientado de Conectividade por Conhecimento****: Seja $G = (V, \xi)$ um grafo não orientado representando a relação de conhecimento determinada pelo oráculo PD. Então, $V = \Pi$ e $(i, j) \in \xi$ se e somente se $j \in i.PD \vee i \in j.PD$ (i conhece j ou j conhece i).*

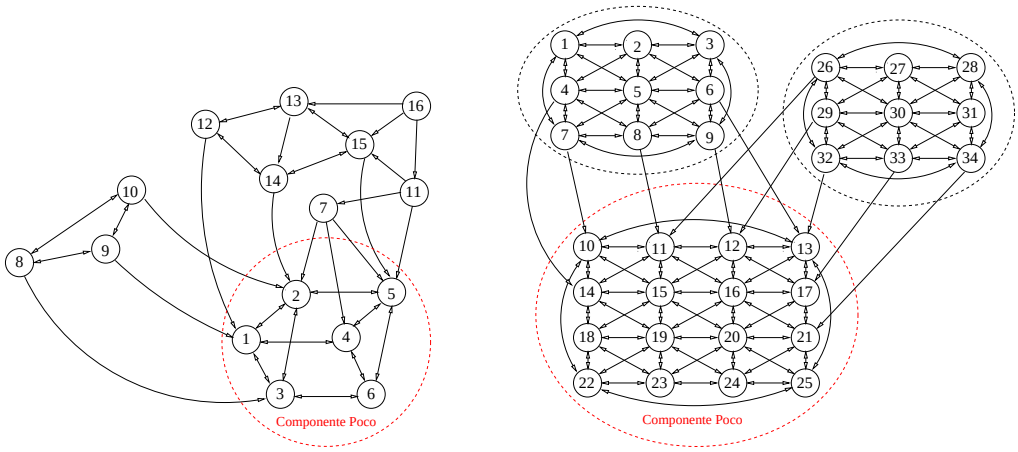
Baseado nas propriedades apresentadas pelos grafos de conectividade por conhecimento, algumas classes de detectores de participação foram propostas para resolver o consenso entre participantes desconhecidos. Neste sentido, o mínimo de conhecimento necessário para que o BFT-CUP admita solução é encapsulado em um PD k-OSR (k-One Sink Reducibility ou k-Redutível a Único Poço) [Greve and Tixeuil 2007, Alchieri et al. 2008].

Antes de definir como um detector de participação k-OSR encapsula o conhecimento sobre a composição do sistema, vamos definir algumas notações sobre grafos. Uma componente G_c de G_{di} é *k-fortemente conexa* se para qualquer par (i,j) de nós em G_c, i pode alcançar j através de pelo menos k caminhos disjuntos nos nós. Além disso, uma componente G_{sink} de G_{di} é uma *componente poço* quando não existe caminhos partindo de nós pertencentes à G_{sink} para outros nós de G_{di}, com exceção dos nós na própria componente G_{sink}.

Definição 3 ***PD k-Redutível a Único Poço (k-OSR)****: O grafo de conectividade por conhecimento G_{di}, que representa a relação de conhecimento induzida pelo oráculo PD, satisfaz as seguintes condições:*

1. *O grafo não orientado G obtido de G_{di} é conexo;*
2. *A redução de G_{di} as suas componentes fortemente conexas tem apenas uma componente poço, chamada G_{sink};*
3. *G_{sink} é k-fortemente conexa;*
4. *Para cada $i \notin G_{sink}$ e $j \in G_{sink}$, existem pelo menos k caminhos disjuntos nos nós de i para j.*

Para melhor ilustrar a Definição 3, a Figura 1 apresenta dois grafos G_{di} induzidos por um detector de participação k-OSR. As figuras 1(a) e 1(b) mostram relações de conhecimento induzidas por detectores de participação da classe 3-OSR e 5-OSR, respectivamente. Por exemplo, na Figura 1(a), o valor retornado por $1.PD$ é o subconjunto $\{2, 3, 4\}$ de Π.



(a) 3-OSR (b) 5-OSR

Figura 1. Grafos de Conectividade por Conhecimento.

2.4. BFT-CUP: Consenso Bizantino entre Participantes Desconhecidos

Esta seção apresenta os principais protocolos que formam a solução para o BFT-CUP [Alchieri et al. 2008], a qual é modular: primeiramente, um conjunto de participantes que compartilham a mesma visão parcial do sistema é identificado; após, um algoritmo de consenso para redes tradicionais (estáticas) é executado por estes participantes que difundem o valor de decisão pelo sistema, permitindo que todos os participantes decidam pelo mesmo valor.

Seguindo esta abordagem e utilizando os requisitos mínimos de sincronia para executar o algoritmo de consenso estático (clássico) Paxos Bizantino [Castro and Liskov 2002] (sistema parcialmente síncrono [Dwork et al. 1988]), é possível resolver o BFT-CUP utilizando um PD k-OSR, onde $k \geq 2f + 1$ e G_{sink} sendo composta por no mínimo $3f + 1$ participantes [Alchieri et al. 2008], que executam o algoritmo de consenso Paxos Bizantino.

Para analisar os aspectos práticos acerca do comportamento do BFT-CUP (seções 3 e 4), é fundamental entender o funcionamento de seus protocolos bem como das relações de dependência entre os mesmos. Desta forma, os algoritmos que solucionam o BFT-CUP são brevemente descritos a seguir. Os pseudocódigos destes algoritmos, bem como as provas de que os mesmos funcionam conforme especificado, podem ser encontrados em [Alchieri et al. 2008].

1 - *Reachable Reliable Broadcast*: Este protocolo é usado para comunicação entre participantes ainda não conhecidos, i.e., para comunicações realizadas antes dos participantes do sistema determinarem suas relações de conhecimento. Este protocolo basicamente realiza *flood* de mensagens no sistema, onde cada participante envia/encaminha as mensagens para todos os seus vizinhos em G_{di} (i.e., nós retornados pelo seu detector de participação). Todos os processos alcançáveis em G_{di} entregam estas mensagens assim que as autenticidades das mesmas sejam comprovadas através da redundância no recebimento, o que ocorre quando as mesmas forem recebidas através de pelo menos $f + 1$ caminhos disjuntos nos nós [Alchieri et al. 2008].

2 - *Discovery*: O primeiro passo para resolver o consenso entre participantes desconhecidos é fazer com que cada participante obtenha o máximo de conhecimento possível sobre a composição do sistema. Neste sentido, partindo do conhecimento inicial retornado pelos detectores de participação, este protocolo é usado pelos participantes do sistema para expandir seus conhecimentos sobre a composição do mesmo, através de uma espécie de busca em G_{di}.

Como os processos ainda não determinaram suas relações de conhecimento, este procedimento utiliza o algoritmo de difusão anteriormente descrito para garantir a integridade das mensagens recebidas pelos participantes [Alchieri et al. 2008]. Após a execução deste algoritmo, e por consequência da determinação das relações de conhecimento, os participantes comunicam-se através de canais ponto a ponto confiáveis e autenticados, onde um protocolo de roteamento tolerante a faltas bizantinas (ex.: [Kotzanikolaou et al. 2005]) deve ser empregado. Esta forma de comunicação também é utilizada pelo protocolo de difusão (*reachable reliable broadcast*) para enviar/encaminhar mensagens para nós vizinhos (que sempre são conhecidos pelo participante que está enviando/encaminhando uma mensagem).

Esta é a parte mais importante do BFT-CUP, onde cada participante obtém a sua visão parcial do sistema. Neste sentido, um dos grandes desafios do BFT-CUP é a busca de uma solução para o consenso Bizantino entre processos que possuem diferentes visões parciais sobre a composição do sistema.

3 - *Sink*: Após um participante obter a sua visão parcial do sistema, este algoritmo é utilizado para determinar se o mesmo pertence à componente G_{sink} de G_{di}. O princípio de funcionamento deste protocolo baseia-se no fato de que, através do algoritmo *Discovery*, cada participante em G_{sink} obtém a mesma visão parcial do sistema que é composta justamente pelos participantes de G_{sink}. Desta forma, este algoritmo isola um conjunto de participantes que compartilham a mesma visão do sistema, i.e., os participantes em G_{sink}.

4 - *Consensus*: Este algoritmo é executado após cada participante determinar se pertence à componente G_{sink} e apresenta comportamento distinto, dependendo da sua localização em G_{di}:

- Participantes em G_{sink}: Estes participantes executam um consenso bizantino clássico (ex. Paxos Bizantino [Castro and Liskov 2002]) e difundem o valor de decisão para os outros nós do sistema, que apenas solicitam este valor. Isso é possível graças ao fato dos participantes em G_{sink} compartilharem a mesma visão parcial do sistema. Este também é o motivo pelo qual G_{sink} deve possuir pelo menos $3f + 1$ participantes (o mínimo necessário para executar o consenso bizantino clássico [Castro and Liskov 2002]).
- Participantes que não pertencem à G_{sink}: Estes participantes apenas enviam uma requisição aos nós em G_{sink} solicitando o valor de decisão. Cada participante decide por um valor v quando receber pelo menos $f + 1$ respostas indicando que v foi o valor decidido pelos participantes em G_{sink}, garantindo a presença de pelo menos uma resposta produzida por um participante correto.

Relações de Dependência entre os protocolos do BFT-CUP. A Figura 2 apresenta as relações de dependência entre os protocolos utilizados para resolver o BFT-CUP. Note que o protocolo de difusão de mensagens (*reachable reliable broadcast*), juntamente com a camada subjacente de roteamento tolerante a faltas bizantinas, abstraem todos os aspectos relacionadas com a comunicação entre os processos. Um processo i deve invocar uma execução do consenso através do protocolo *consensus*, que utiliza o protocolo *sink* para verificar se i pertence à G_{sink}. Caso $i \in G_{sink}$, então i executa um consenso Bizantino clássico. De outra forma, i apenas solicita o valor de decisão conforme anteriormente descrito.

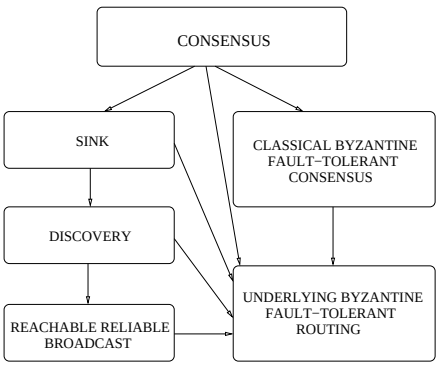


Figura 2. Relações de Dependência entre os Protocolos do BFT-CUP.

O protocolo *sink*, por sua vez, utiliza o protocolo *Discovery* para determinar a visão parcial que i obtém sobre o sistema. O algoritmo *Discovery* utiliza o conhecimento reportado pelo detector de participação de i ($i.PD$) para realizar uma busca em G_{di} obtendo o máximo de conhecimento possível sobre a composição do sistema. Para isso, também utiliza o algoritmo de difusão de mensagens, pois neste estágio os processos ainda não definiram completamente

as suas relações de conhecimento e i não pode enviar mensagens diretamente para todos os processos alcançáveis em G_{di} (nem todos estes processos já são conhecidos por i neste momento).

3. BFT-CUP em Redes MANETs

Esta seção busca analisar, através de uma série de simulações, o comportamento dos protocolos do BFT-CUP quando executados em MANETs. Para isso, tais protocolos foram implementados e simulados no *Jist/SWANS*. O *Jist* (*Java in Simulation Time*) [Barr et al. 2004] é um simulador de alta performance baseado em eventos discretos que executa sobre uma máquina virtual *Java*. Já o *SWANS* (*Scalable Wireless Ad hoc Network Simulator*) [Barr et al. 2005] é um módulo para simulação de MANETs construído sobre o *Jist*, sendo portanto também implementado em *Java*.

3.1. Implementação dos Detectores de Participação

Os detectores de participação são abstrações que fornecem conhecimento sobre a composição do sistema, sendo que para a solução do BFT-CUP o mínimo exigido é o detector de participação da classe k-OSR (Seção 2.3). No entanto, não existe uma proposta de implementação para um detector de participação que garante os atributos definidos para esta classe em uma rede MANET. A possibilidade de existência de tal implementação continua sendo uma questão em aberto [Costa et al. 2009]. Deste modo, em nossas simulações utilizamos o detector de participação **PD-1hop** [Costa et al. 2009] e buscamos avaliar a possibilidade de solução do BFT-CUP em um cenário onde o detector de participação não garante as propriedades do PD k-OSR. Note que caso o detector de participação assegure as propriedades do PD k-OSR, então o BFT-CUP sempre admite solução [Alchieri et al. 2008], i.e., a conectividade por conhecimento fornecida pelo detector é suficiente para resolver o BFT-CUP.

O detector de participação **PD-1hop** é muito simples e retorna para os processos uma lista contendo os nós que se encontram no seu alcance de transmissão (nós vizinhos). Sempre que este detector é iniciado em algum processo, o mesmo envia mensagens de "hello" através de um *broadcast* local, por meio da camada MAC da rede MANET, que atinge todos os seus vizinhos dentro de 1 salto. Ao receber estas mensagens, cada vizinho i adiciona a fonte (emissor) a sua lista de nós conhecidos, que é retornada em $i.PD$.

Esse procedimento de descoberta inicial termina quando um tempo é atingido. Durante este período, os processos podem difundir novas mensagens de "hello". Este detector é implementado de forma independente dos parâmetros do BFT-CUP e é ativado no início de cada simulação para retornar uma aproximação da vizinhança real na rede.

3.2. Configurações das Simulações

As configurações gerais utilizadas nas simulações foram adotadas com base nos trabalhos de Costa *et al.* [Costa and Greve 2007, Costa et al. 2009]. Estas configurações podem ser divididas em configurações da rede MANET (Tabela 1) e configurações do BFT-CUP (Tabela 2).

Os principais parâmetros da rede MANET variados nas simulações e que influenciam na possibilidade de solução do BFT-CUP são: *(1)* o alcance de transmissão e *(2)* a densidade de nós no sistema (quantidade de nós). Estes dois parâmetros estão diretamente relacionados com o grau de conhecimento que os participantes obtêm sobre a composição do sistema, i.e., com a conectividade por conhecimento apresentada em G_{di}, que é o fator determinante para a possibilidade de solução do BFT-CUP.

Os detectores de participação utilizados nestas simulações foram ativados durante $60s$ antes de cada execução do BFT-CUP. Durante este período, cada participante esperou um tempo

aleatório de até $20s$ para reenviar a mensagem de "hello" indicando sua presença no sistema (Seção 3.1). Vale ressaltar que estes parâmetros fizeram com que cada participante enviasse em média de 3 a 4 destas mensagens.

Parâmetros das Simulações (Simulador)	
Simulador	*Jist/SWANS*
Quantidade de nós	de 0 à 50
Área	$500X500\ m^2$
Alcance *wireless*	de $0m$ à $300m$
Repetições	10
Padrão de Mobilidade	*Randon WayPoint*
Velocidade dos nós	de $0m/s$ à $5m/s$
Tempo de pausa	de até $2s$

Tabela 1. Configurações da rede MANET.

Além disso, também vale destacar que apesar do sistema estar configurado para suportar um determinado número de faltas, que varia de acordo com as configurações utilizadas em determinada simulação (de 1 até 10 faltas), não foram introduzidos nós faltosos no sistema.

Parâmetros das Simulações (BFT-CUP)	
Tempo de PD	$60s$
Tempo para reenvio de "hello"	de até $20s$
Número máximo de faltas suportadas	de 1 à 10

Tabela 2. Configurações do BFT-CUP.

Por fim, o tempo de término das simulações foi definido como sendo o instante de tempo em que todos os eventos produzidos pelos nós foram processados pelo simulador, i.e., as simulações permaneceram em execução enquanto tinham eventos a serem processados.

3.3. Métricas Analisadas

Estas simulações têm por objetivo verificar em quais condições os participantes do sistema conseguem adquirir um conhecimento suficiente para resolver o BFT-CUP. Para atingir este objetivo, não é importante analisar o tempo necessário para a execução do BFT-CUP, mas sim as métricas diretamente relacionadas com a convergência do BFT-CUP. Neste sentido, as seguintes métricas serão analisadas nestas simulações:

1. *Conhecimento retornado pelos detectores de participação*: Esta métrica define a quantidade de conhecimento sobre a composição do sistema que os nós recebem diretamente dos detectores de participação. Deste modo, esta métrica refere-se ao conhecimento adquirido pelos participantes antes da execução do agoritmo *Discovery*.
2. *Convergência para Poço*: Esta métrica refere-se a quantidade de participantes que acabam por determinar que pertencem ao G_{sink} de G_{di}. No entanto, como veremos a seguir, em determinadas configurações é possível ocorrer execuções do BFT-CUP onde diferentes participantes acreditam que pertencem a diferentes componentes G_{sink}. Neste cenário é teoricamente possível ocorrer uma de duas coisas: *(1)* ou os participantes não terminam a execução do BFT-CUP (porque cada G_{sink} não possui nós suficientes para executar o consenso clássico); *(2)* ou ocorre desacordo nos valores das decisões (participantes em diferentes G_{sink} podem decidir por valores diferentes [Alchieri et al. 2008]).

3. *Terminação*: Esta métrica define a quantidade de participantes que terminam a execução dos protocolos do BFT-CUP. Rigorosamente falando, os nós em G_{sink} não podem terminar o BFT-CUP quando decidem por algum valor, pois os mesmos devem responder às requisições de outros nós solicitando este valor da decisão. Assim, tais nós devem aguardar por estas requisições enquanto o valor de decisão é importante para a aplicação que está utilizando o BFT-CUP como base para algum algoritmo distribuído. Nestas simulações, os nós em G_{sink} aguardaram por requisições de valor de decisão enquanto tinha algum evento para ser processado pelo simulador, i.e., enquanto existia a possibilidade de algum processo ter solicitado este valor de decisão.
4. *Acordo*: Esta métrica especifica a quantidade de participantes que terminam a execução dos protocolos do BFT-CUP decidindo pelo mesmo valor.

3.4. Resultados e Análises

Esta seção apresenta os resultados obtidos através das simulações, bem como as análises sobre estes resultados. Primeiramente, a Figura 3 apresenta os resultados para simulações onde fixou-se o número de faltas suportadas pelo sistema (f) em 1 falta e variou-se a densidade e o alcance de transmissão dos nós.

Na Figura 3(a) é possível observar o grau de conhecimento sobre a composição do sistema que é obtido através dos detectores de participação (conhecimento obtido antes da execução do algoritmo *Discovery*). Considerando a porcentagem de conhecimento inicial fornecido aos participantes (e não os números absolutos), é possível verificar que a porcentagem de conhecimento está diretamente relacionada com o alcance de transmissão dos nós, sendo que a densidade de nós presentes no sistema exerce pouca influência sobre este percentual. O conhecimento reportado pelos detectores de participação variou de aproximadamente 0% (para alcance de $30m$) a aproximadamente 50% (para alcance de $300m$). Considerando o cenário para alcance de $300m$, isto significa que quando o sistema foi composto por 4 nós, o conhecimento sobre a presença de mais 2 outros nós em média foi retornado pelos detectores de participação. Já para o cenário de 50 nós, este valor salta para 25 outros nós em média.

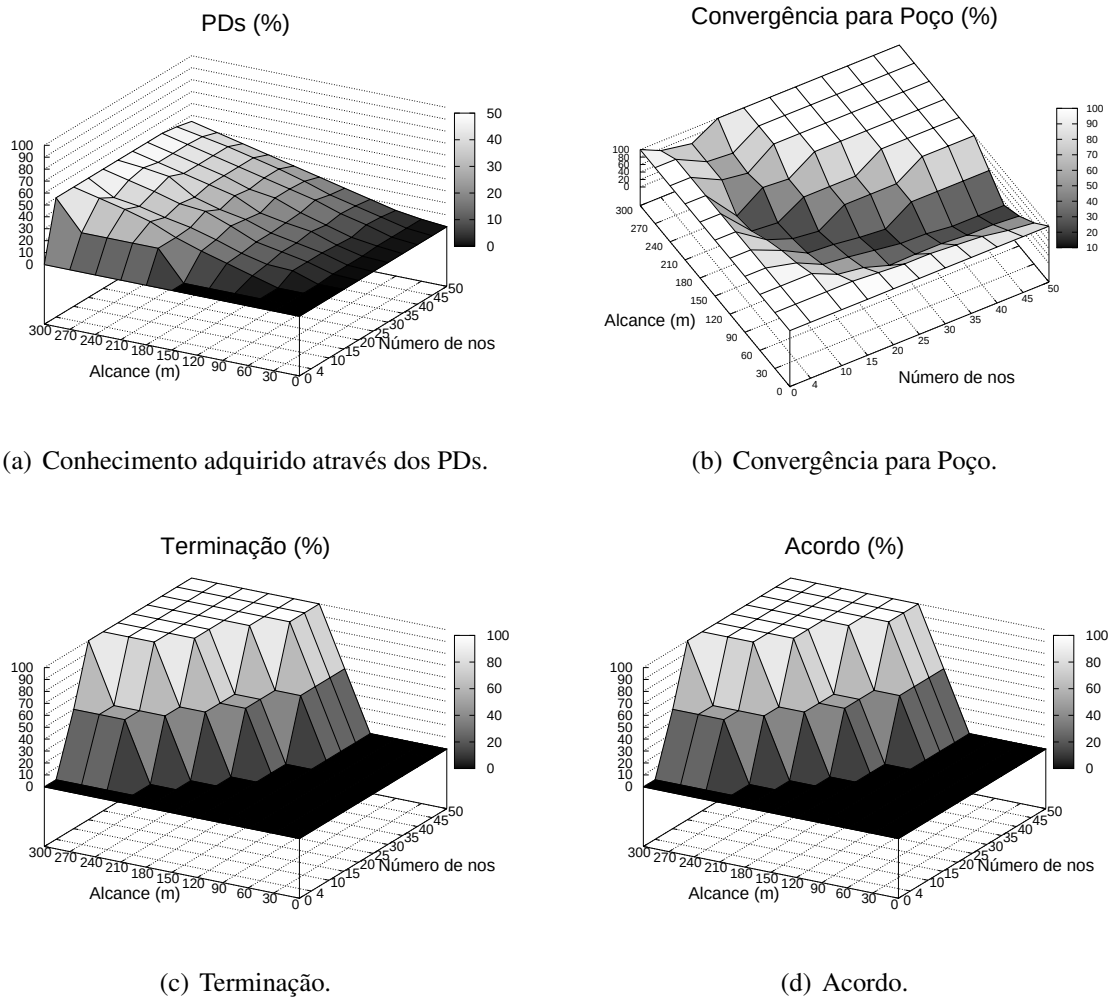
É importante notar que a união destes conhecimentos, obtidos em cada participante do sistema através de seus detectores de participação, gera o grafo de conectividade por conhecimento G_{di}, o qual indicará se a conectividade obtida é suficiente para resolver o BFT-CUP.

Após esta etapa de descoberta inicial, os nós executam o algoritmo *Discovery* para expandir este conhecimento e na sequência *Sink* para determinar se pertencem à G_{sink} de G_{di}. Neste sentido, a Figura 3(b) apresenta a porcentagem de participantes que acreditaram pertencer à G_{sink} nos mais variados cenários. Este gráfico apresenta três zonas distintas:

1. Alcance de transmissão e/ou densidade de nós baixo: Nestes cenários todos os nós acreditam pertencer a G_{sink}. No entanto, como os conhecimentos obtidos sobre a composição do sistema (depois do algoritmo *Discovery*) são muito reduzidos, os nós acreditam pertencer a diferentes componentes G_{sink}. Em alguns cenários os nós descobriram apenas outro ou até mesmo nenhum outro participante.
2. Alcance de transmissão e/ou densidade de nós moderada: Estes cenários formam o fundo da "vala" na Figura 3(b), onde o conhecimento obtido pelos nós é tão divergente entre os mesmos (embaralhado) que nenhum nó (ou poucos) acreditam pertencer a G_{sink}.
3. Alcance de transmissão e/ou densidade de nós alta: Nestes cenários todos os nós decidiram que pertencem ao mesmo G_{sink}, pois adquiriram um grande grau de conhecimento sobre a composição do sistema. Em praticamente todos estes cenários, os nós acabaram conhecendo todos os outros participantes do sistema.

As figuras 3(c) e 3(d) apresentam os gráficos para terminação e acordo, respectivamente. Como podemos perceber, estas métricas estão diretamente relacionadas com a convergência para um único G_{sink}, sendo que não é possível resolver o BFT-CUP nos cenários descritos anteriormente pelos itens 1 e 2. Nestes casos, o consenso tradicional não é executado pelos seguintes motivos:

1. Cenários do item 1: Apesar de todos os nós acreditarem pertencer a G_{sink}, o consenso tradicional não é executado porque são diferentes componentes G_{sink} visto que os seus conhecimentos são muito reduzidos. De fato, nenhum nó consegue descobrir pelo menos outros 3 nós para executar o consenso clássico que exige $3f + 1$ participantes [Castro and Liskov 2002] (como $f = 1$, é necessário 4 participantes).
2. Cenários do item 2: O consenso clássico não é executado porque um número insuficiente de nós (menos de $3f + 1$ [Castro and Liskov 2002]) acreditam pertencer à G_{sink}.



(a) Conhecimento adquirido através dos PDs.

(b) Convergência para Poço.

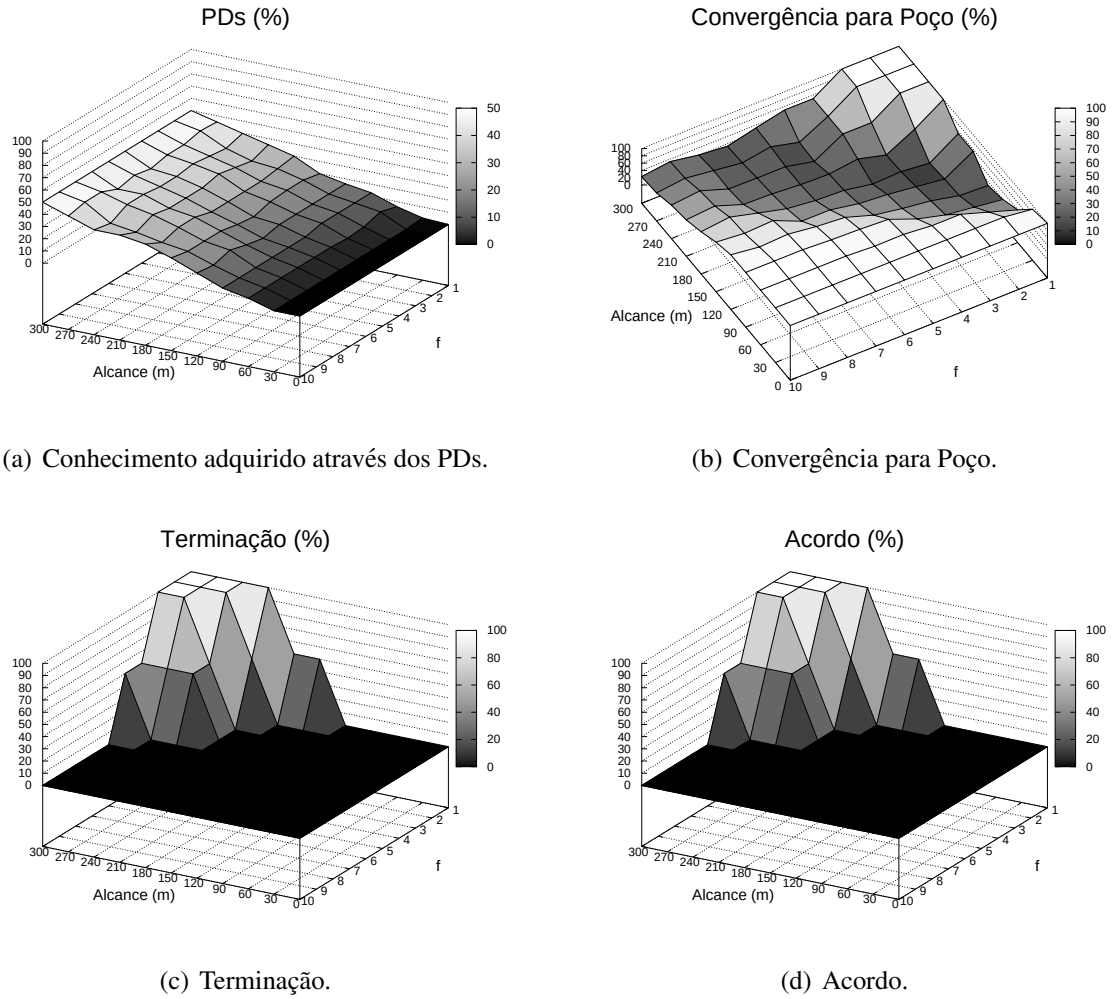
(c) Terminação.

(d) Acordo.

Figura 3. Resultados das simulações para $f = 1$. Tanto o aumento da densidade de nós no sistema, quanto o aumento do alcance de transmissão dos nós, faz com que os participantes obtenham um melhor conhecimento sobre a composição do sistema, aumentando a conectividade em G_{di} e com isso também aumentando a probabilidade do BFT-CUP admitir solução.

Por estes motivos, os gráficos de acordo e terminação são idênticos representando que quando os algoritmos do BFT-CUP terminaram, a propriedade de acordo do consenso foi atendida. Este é um resultado importante, pois nestas simulações sempre que os nós terminaram a execução dos protocolos acabaram decidindo pelo mesmo valor.

A fim de analisar o comportamento do BFT-CUP em relação ao número de faltas suportadas pelo sistema, outros cenários foram simulados para o sistema composto por 30 nós. Os resultados são apresentados na Figura 4, onde é possível perceber que o número de faltas suportadas não têm relação com a quantidade de conhecimento obtida através dos detectores de participação 4(a). De fato, os detectores de participação trabalham de forma independente do BFT-CUP e este parâmetro não deve interferir em seu funcionamento.



(a) Conhecimento adquirido através dos PDs.

(b) Convergência para Poço.

(c) Terminação.

(d) Acordo.

Figura 4. Resultados das simulações para 30 nós. A probabilidade do BFT-CUP admitir solução em uma rede MANET é diretamente proporcional ao alcance de transmissão dos nós e inversamente proporcional ao número de faltas toleradas pelo sistema.

Nestas simulações foi possível resolver o BFT-CUP em cenários para f de até 5 faltas e alcance de transmissão a partir de $150m$. O BFT-CUP nunca terminou nos cenários com f maior do que 5 faltas, pois nestes casos não foi possível formar um único G_{sink}. A Figura 4(b) apresenta a porcentagem de nós que acreditaram pertencer a G_{sink}, cuja interpretação é da mesma forma que o anterior (Figura 3(b)): *(1)* alcance de transmissão baixo – os nós acreditaram que pertenciam à diferentes componentes G_{sink}; *(2)* alcance de transmissão moderado e/ou f alto – poucos nós acreditaram pertencer à G_{sink};*(3)* alcance de transmissão alto e/ou f baixo – os nós decidiram que pertenciam à mesma componente G_{sink}.

Novamente, sempre que os protocolos do BFT-CUP terminaram (Figura 4(c)) os nós acabaram decidindo pelo mesmo valor (Figura 4(d)), indicando que não ocorreu desacordo nos cenários simulados.

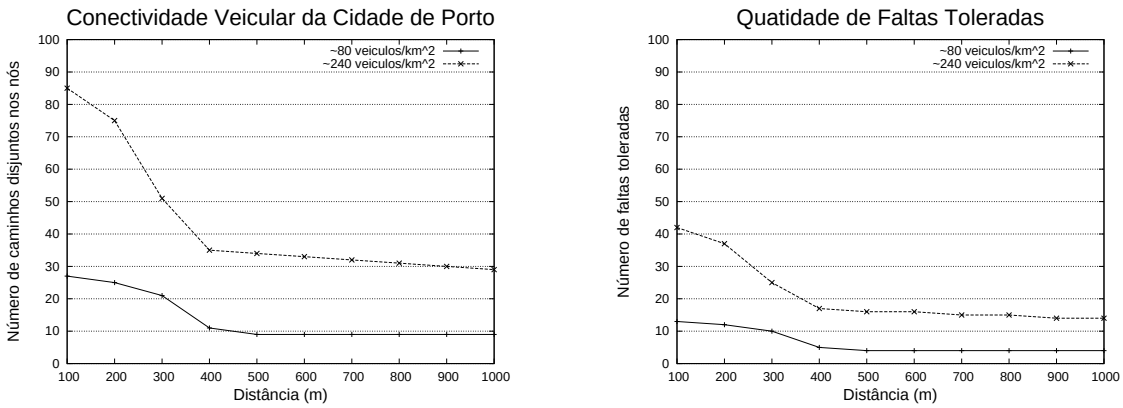
4. BFT-CUP em Redes VANETs: Exemplo Prático da Cidade de Porto - Portugal

Esta seção apresenta uma análise acerca da realização do BFT-CUP entre veículos (ou uma parte deles) que trafegam nas vias de uma cidade, ou seja, em uma rede VANET. O cenário escolhido foi a cidade de Porto - Portugal, cujo tráfego veicular foi modelado por Ferreira *et al.* [Ferreira et al. 2009]. A mobilidade veicular e a comunicação entre os veículos são dois fatores fundamentais desta modelagem, que baseou-se no uso de fotografias aéreas (*stereoscopic aerial photography*) e mostrou-se mais realista que outros simuladores para VANETs [Ferreira et al. 2009]. De fato, a modelagem do tráfego urbano é uma tarefa árdua, sendo que cada cidade possui suas peculiaridades. Por exemplo, os motoristas preferem trafegar por determinadas vias em determinados horários, de acordo com o movimento dos pedestres e/ou de outros veículos nestes locais.

Nesta análise consideramos o grau de conectividade veicular desta cidade, onde as rotas escolhidas pelos veículos constituem o fator que mais interfere nesta métrica, pois a concentração de veículos em um número reduzido de vias certamente afetará a conectividade de veículos que trafegam em outras vias.

A Figura 5(a) apresenta o grau de conectividade entre dois veículos em relação à distância entre os mesmos (dados extraídos de [Ferreira et al. 2009]). Para este cálculo, fixou-se um veículo i e calculou-se a média de caminhos disjuntos nos nós que ligam os outros veículos ao i, de acordo com a distância que os mesmos se encontram. O gráfico apresenta o grau de conectividade para cenários com aproximadamente 80 e 240 veículos/km^2. Além disso, o alcance de transmissão foi especificado em $250m$ para comunicações diretas e $140m$ para comunicações onde alguma construção impediu a comunicação direta.

A partir destes dados, podemos derivar o número máximo de faltas toleradas na execução do BFT-CUP entre estes veículos: como é necessário $k \geq 2f + 1$ (Seção 2.4), onde k é o grau de conectividade, temos que $f \leq \frac{k-1}{2}$. A Figura 5(b) mostra que é possível tolerar até aproximadamente 40 faltas em um cenário de 240 veículos/km^2, que se encontram em uma distância de até $100m$. Além disso, este número diminui bruscamente até os $400m$ de distância quando então tende a estabilizar.



(a) Conectividade Veicular da Cidade de Porto (extraído de [Ferreira et al. 2009])

(b) Quantidade de Faltas Toleradas

Figura 5. O número de faltas toleradas está diretamente relacionado com o grau de conectividade do sistema.

Existem diversas aplicações para o consenso em redes VANETs. Por exemplo, existe uma proposta de substituição dos semáforos (infraestrutura localizada na via) pela execução de

um consenso entre veículos [Ferreira et al. 2010], que determinará qual destes veículos deve atravessar por primeiro em um cruzamento, implementando uma espécie de semáforo virtual. Esta abordagem visa otimizar o fluxo de veículos em cruzamentos, pois neste caso um veículo apenas irá parar no cruzamento quando realmente for necessário devido à presença de outros veículos competindo pelo cruzamento.

5. Trabalhos Relacionados

Existem alguns trabalhos na literatura que buscam analisar analiticamente o grau de conectividade apresentado por uma rede MANET em função de suas configurações [Bettstetter 2002, Santi 2005]. Estes trabalhos buscam soluções para controlar a topologia da rede e podem ser empregados na obtenção de uma rede com características (principalmente relacionadas com o grau de conectividade) que permitam a solução do BFT-CUP.

Além disso, encontramos alguns trabalhos que buscam analisar o comportamento do FT-CUP (consenso tolerante a faltas por *crash*) [Costa and Greve 2007, Costa et al. 2009] através de simulações em uma rede MANET. Este trabalho complementa os anteriores através da análise dos mais diversos cenários e considera protocolos que admitem a presença de participantes maliciosos, o que também exige maior conectividade no sistema [Alchieri et al. 2008].

6. Conclusões

Este trabalho apresentou resultados e análises envolvendo aspectos práticos do BFT-CUP. Primeiramente, os protocolos do BFT-CUP foram simulados para diversos cenários em uma rede MANET. A análise destes resultados torna possível avaliar a probabilidade do BFT-CUP admitir solução dada uma determinada configuração da rede. Além disso, uma análise acerca da realização do BFT-CUP em uma rede VANET foi apresentada, onde tomou-se como base o tráfego veicular da cidade de Porto-Portugal.

A partir dos dados discutidos neste artigo é possível identificar quais são os parâmetros e configurações do sistema que devem ser considerados quando estes protocolos forem utilizados no desenvolvimento de aplicações: *(1)* a escolha do parâmetro f é determinante para a convergência de BFT-CUP; *(2)* uma escolha adequada entre o alcance de transmissão e a densidade de nós no sistema também determinará a possibilidade de convergência de BFT-CUP.

Além disso, nas simulações realizadas conseguimos resolver o BFT-CUP com um detector de participação mais simples do que o k-OSR. Nas configurações onde o BFT-CUP admitiu solução, geralmente o sistema apresentou um grafo de conectividade por conhecimento G_{di} k-fortemente conexo (apenas uma componente k-fortemente conexa). Outro ponto a destacar é que a propriedade de acordo não foi violada em nenhum cenário analisado, i.e., sempre que os protocolos do BFT-CUP terminaram, os participantes acabaram decidindo pelo mesmo valor.

Agradecimentos

Gostaríamos de agradecer à professora Fabíola Greve pelas discussões sobre os protocolos do BFT-CUP. Além disso, Eduardo Alchieri, Luiz Tomelin e Joni Fraga são bolsistas do CNPq.

Referências

Alchieri, E. A. P., Bessani, A. N., da Silva Fraga, J., and Greve, F. (2008). Byzantine consensus with unknown participants. In *12th International Conference On Principles Of DIstributed Systems*, pages 22–40.

Aspnes, J. (2003). Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2-3):165–175.

Barr, R., Haas, Z. J., and van Renesse, R. (2004). Jist: Embedding simulation time into a virtual machine. In *Proceedings o EuroSim Congress on Modelling and Simulation*.

Barr, R., Haas, Z. J., and van Renesse, R. (2005). *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*, chapter Scalable Wireless Ad hoc Network Simulation, pages 297–311. CRC Press.

Ben-Or, M. (1983). Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing*, pages 27–30.

Bettstetter, C. (2002). On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing - MobiHoc'02*.

Bracha, G. (1984). An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In *Proceedings of the 3rd ACM symposium on Principles of Distributed Computing*, pages 154–162.

Castro, M. and Liskov, B. (2002). Practical Byzantine fault-tolerance and proactive recovery. *ACM Transactions on Computer Systems*, 20(4):398–461.

Cavin, D., Sasson, Y., and Schiper, A. (2004). Consensus with unknown participants or fundamental self-organization. In *Proceedings of the 3rd International Conference on Ad hoc Networks and Wireless - ADHOC-NOW 2004*, pages 135–148.

Cavin, D., Sasson, Y., and Schiper, A. (2005). Reaching agreement with unknown participants in mobile self-organized networks in spite of process crashes. Technical Report IC/2005/026, EPFL - LSR.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267.

Correia, M., Neves, N. F., and Veríssimo, P. (2006). From consensus to atomic broadcast: Time-free Byzantine-resistant protocols without signatures. *The Computer Journal*, 49(1).

Costa, V. F., Greve, F. G. P., , and Tixeuil, S. (2009). Consenso ft-cup em redes manets: Uma abordagem prática. In *Anais do 27º Simpósio Brasileiro de Redes de Computadores - SBRC 2009*.

Costa, V. F. and Greve, F. G. P. (2007). Aspectos práticos da realização do consenso ft-cup em redes móveis ad-hoc. In *Anais VIII Workshop de Teste e Tolerância a Falhas- WTF 2007*, Belém, PA, Brasil.

Dwork, C., Lynch, N. A., and Stockmeyer, L. (1988). Consensus in the presence of partial synchrony. *Journal of ACM*, 35(2):288–322.

Ferreira, M., Conceição, H., Fernandes, R., and Tonguz, O. K. (2009). Stereoscopic aerial photography: an alternative to model-based urban mobility approaches. In *Proceedings of the sixth ACM international workshop on VehiculAr InterNETworking - VANET'09*.

Ferreira, M., Fernandes, R., Conceição, H., Viriyasitavat, W., and Tonguz, O. K. (2010). Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking - VANET'10*.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382.

Greve, F. G. P. and Tixeuil, S. (2007). Knowledge connectivity vs. synchrony requirements for fault-tolerant agreement in unknown networks. In *Proceedings of the International Conference on Dependable Systems and Networks - DSN*, pages 82–91.

Kotzanikolaou, P., Mavropodi, R., and Douligeris, C. (2005). Secure multipath routing for mobile ad hoc networks. *Wireless On-demand Network Systems and Services - WONS*, pages 89–96.

Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine generals problem. *ACM Transactions on Programing Languages and Systems*, 4(3):382–401.

Santi, P. (2005). Topology control in wireless ad hoc and sensor networks. *ACM Comput. Surv.*, 37:164–194.

Medidas de conectividade baseadas em cortes de vértices para redes complexas

Karine Pires[1], **Jaime Cohen**[1,2], **Elias P. Duarte Jr.**[1]

[1]Departamento de Informática – Universidade Federal do Paraná

[2]Departamento de Informática – Universidade Estadual de Ponta Grossa

Abstract. *This paper presents connectivity measures for complex networks that allows the identification of well connected network nodes. We define the vertex-connectivity of nodes and we show how to compute the measure. The vertex-connectivity is compared to other centrality measures such as betweenness, closeness, eccentricity, degree and edge-connectivity measures. We analyze the measures in extreme cases and in random graphs.*

Resumo. *Neste trabalho apresentamos medidas de conectividade para redes complexas. Essas medidas identificam os nodos importantes em uma rede de acordo com a conectividade dos mesmos em relação aos demais nodos. Mostramos como calcular o valor da medida que chamamos de vértice-conectividade e que denotamos por* $\kappa(v)$*. Relacionamos o valor da vértice-conectividade com outras medidas como grau de intermediação, closeness, excentricidade, grau e medidas baseadas em cortes de arestas. Analisamos as medidas em casos extremos e em redes sintéticas aleatórias.*

1. Introdução

Redes complexas tem por objetivo modelar qualquer sistema, natural ou artificial, que possua muitas partes discretas que interagem entre si. Por conta dessa generalidade, o conhecimento sobre redes complexas pode ser aplicado a inúmeras áreas do conhecimento como física, computação, telecomunicações, astronomia e outras [Barabási 2003, Costa et al. 2006]. Também podem ser encontradas aplicações de sistemas complexos em redes reais, como, por exemplo, aquelas que modelam relações existentes entre pessoas [Leskovec et al. 2010, Liljeros et al. 2001], redes de energia [Watts and Strogatz 1998] e a Internet [Faloutsos et al. 1999, Albert et al. 1999, Albert et al. 2000].

Dentre as aplicações de redes complexas na área da computação, está a simulação de eventos, como a propagação de vírus em uma rede [Balthrop et al. 2004] ou o estudo do comportamento de algoritmos de roteamento que possibilitam torná-los mais eficientes ou mais seguros [Echenique et al. 2005].

A área de grafos, apesar de já amplamente estudada, conta com enorme atividade científica atualmente quando relacionada às aplicações em redes complexas. Dentre os conceitos estudados e aplicados às redes complexas estão medidas para a caracterização de redes. As medidas propostas não são individualmente suficientes para caracterizar as redes complexas e há a necessidade de considerar um conjunto de várias medidas para encontrar resultados satisfatórios [Costa et al. 2006]. É com este intuito que propomos a medida apresentada neste trabalho, a qual esperamos mostrar ser capaz de contribuir para a compreensão das redes complexas.

Dentre as medidas existentes, destacamos algumas. São elas: grau, grau de intermediação (*betweenness centrality*), *closeness* e excentricidade. O grau é a medida mais óbvia e consiste no número de arestas incidentes ao vértice. O grau de intermediação (*betweenness*) [Freeman 1977] mede quantos caminhos mínimos entre pares de vértices passam pelo vértice avaliado. A medida *closeness* [Okamoto et al. 2008] é definida como a média das distâncias entre um vértice e todos os outros vértices do grafo. A excentricidade [Hage 1995] de um vértice é o máximo das distâncias desse vértice a qualquer outro vértice no grafo.

As medidas de conectividade baseadas em cortes de arestas [Cohen et al. 2011] têm por objetivo encontrar um valor que determine a conectividade de um nodo em relação aos demais nodos da rede. Para tal, elas são fundamentadas no conceito de conectividade de arestas de conjuntos de nodos. Com estas medidas é possível caracterizar efeitos de falhas de enlace em redes de computadores. Em particular, elas permitem encontrar vértices que são mais difíceis de serem desconectados da rede.

A *vértice-conectividade*, nome dado à medida proposta, tem como conceito base a conectividade entre os nodos levando em conta a possibilidade de falha em outros nodos. Ela nos permite encontrar os nodos de maior importância numa rede em relação a vértice-conectividade destes com outros nodos. Uma possível aplicação dessas medidas é a seleção de um conjunto de nodos com máxima confiabilidade em relação a comunicação entre eles.

As medidas de conectividade são menos locais do que o grau. Com efeito, elas são parametrizadas de forma a torná-las ainda mais globais dependendo desse parâmetro. O grau do vértice determina o número de arestas que necessitam ser removidas para desconectá-lo totalmente de todos os demais. Porém, é possível que a falha de uma única aresta seja suficiente para separar esse vértice de grau alto de qualquer outro vértice específico da rede, como é o caso em uma rede com topologia de estrela. Por outro lado, uma medida de conectividade alta garante que o nodo é difícil de ser isolado de pelo menos um outro nodo da rede através de falhas de enlaces ou de nodos dependendo do tipo de conectividade escolhido.

Este trabalho está organizado da seguinte forma: na seção 2 são apresentadas as medidas grau, grau de intermediação, *closeness*, excentricidade e aquelas baseadas em cortes de arestas. Na seção 3 são descritas as medidas de conectividade baseadas em cortes de vértices. Na seção 4 é apresentado o algoritmo para o cálculo da medida. Experimentos e comparações com as medidas citadas anteriormente são apresentados na seção 5. A última seção apresenta as conclusões.

2. Medidas de Conectividade

Nesta seção definimos algumas importantes medidas de centralidade de nodos de redes.

Dado um grafo $G = (V, E)$ onde V é o conjunto de vértices e E o conjunto de arestas, o *grau* de um vértice v, denotado por $deg(v)$, é igual ao número de arestas incidentes ao vértice v.

O *grau de intermediação* ou *betweenness* [Freeman 1977] de um vértice mede quantos caminhos mínimos entre pares de vértices passam por ele. O grau de intermediação é calculado através da seguinte fórmula:

$$B_u = \sum_{ij} \frac{\sigma(i,u,j)}{\sigma(i,j)}$$

onde $\sigma(i,u,j)$ é o número de caminhos mínimos entre os vértices i e j que passam pelo vértice u e $\sigma(i,j)$ é o número total de caminhos mínimos entre os vértices i e j. Por convenção fazemos $\sigma(i,j) = 1$ quando $i = j$.

A medida *closeness* de um vértice é definida como o inverso da média das distâncias entre o vértice e todos os outros vértices do grafo, ou seja:

$$C_u = \frac{|V| - 1}{\sum_{i \in V, i \neq u} d_G(u,i)}$$

onde $d_G(u,i)$ é a distância entre os vértices u e i, isto é, o número de arestas de um caminho mínimo entre u e i.

A excentricidade de um vértice é a distância máxima do vértice aos outros vértices do grafo, definida por:

$$E_u = \max\{d_G(u,i) \mid i \neq u, i \in V\}$$

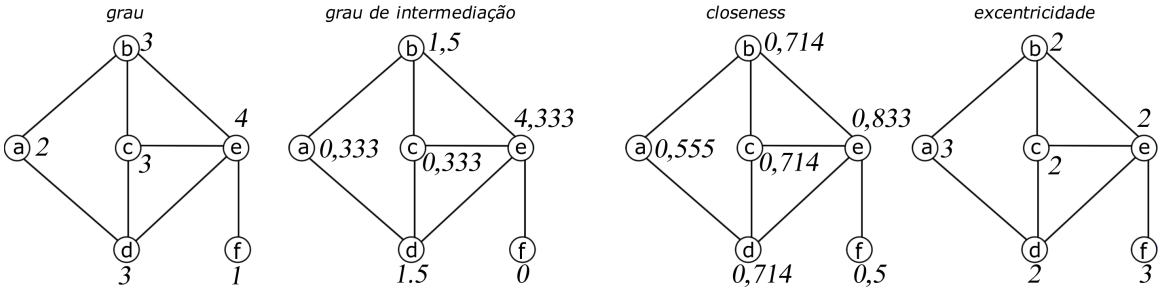


Figura 1. Exemplos de grafos com os valores das medidas de centralidade.

A figura 1 mostra os valores das medidas de centralidade para um mesmo grafo. No primeiro grafo temos o grau dos vértices. O grafo seguinte apresenta o valor do grau de intermediação dos vértices. Por exemplo, o cálculo do grau de intermediação do vértice a é feito assim:

$$\begin{aligned}
B_a &= \sum_{ij} \frac{\sigma(i,a,j)}{\sigma(i,j)} \\
&= \frac{\sigma(a,a,a)}{\sigma(a,a)} + \frac{\sigma(a,a,b)}{\sigma(a,b)} + \cdots + \frac{\sigma(b,a,d)}{\sigma(b,d)} + \cdots + \frac{\sigma(e,a,f)}{\sigma(e,f)} + \frac{\sigma(f,a,f)}{\sigma(f,f)} \\
&= \frac{0}{1} + \frac{0}{1} + \cdots + \frac{1}{3} + \cdots + \frac{0}{1} + \frac{0}{1} \\
&= 0,333
\end{aligned}$$

Note que todos os valores omitidos no cálculo anterior são zeros, pois o único caminho mínimo que passa pelo vértice a é o caminho entre os vértices b e d. Portanto, $\sigma(b, a, d) = 1$. Por outro lado, existem três caminhos mínimos entre os vértices b e d, logo, $\sigma(b, d) = 3$.

O terceiro grafo trás os valores da medida *closeness*. Para o vértice a, o cálculo é feito da seguinte forma:

$$\begin{aligned} C_a &= \frac{|V| - 1}{\sum\limits_{i \epsilon V, i \neq a} d(a, i)} \\ &= \frac{6 - 1}{d(b, a) + d(c, a) + d(d, a) + d(e, a) + d(f, a)} \\ &= \frac{5}{1 + 2 + 1 + 2 + 3} \\ &= 0.55555555555555... \end{aligned}$$

O último grafo da figura 1 mostra os valores de excentricidade. Para o vértice a, temos que o valor da excentricidade é:

$$\begin{aligned} E_a &= \max\{d(a, i) \mid i \neq a, i \in V\} \\ &= \max\{d(b, a), d(c, a), d(d, a), d(e, a), d(f, a)\} \\ &= \max\{1, 2, 1, 2, 3\} \\ &= 3 \end{aligned}$$

Medidas de Conectividade Baseadas em Cortes de Arestas Descrevemos a seguir as medidas de conectividade para nós de uma rede baseadas em cortes de arestas. Estas medidas foram apresentadas em [Duarte Jr. et al. 2004] e [Cohen et al. 2011] e estatísticas e experimentos em redes reais foram apresentados em [Cohen et al. 2011]. O objetivo destas medidas é encontrar um valor significativo a fim de avaliar a conectividade de um nodo em relação à rede. A medida de conectividade mais óbvia é o grau do nodo. Entretanto, em muitas aplicações ela não é ideal. O conceito base para a nova medida é a definição da conectividade de aresta entre pares de nodos do grafo.

A topologia da rede é representada por um grafo não direcionado $G = (V, E)$, onde V são vértices que representam os nodos da rede e E é o conjunto de arestas que representam as ligações entre os nodos da rede. Seja $X \subseteq V$. Denota-se por $\delta(X)$ o conjunto de arestas $\{u, v\}$ tal que $u \in X$ e $v \in V \setminus X$. Chama-se o conjunto de arestas $C \subseteq E$, $C \neq 0$, um *corte* se $C = \delta(X)$ para algum $X \subseteq V$. Um corte $C = \delta(X)$ *separa* dois vértices s e t se $s \in X$ e $t \in V \setminus X$. Um *s-t-corte mínimo* é um corte de cardinalidade mínima que separa s de t.

Em grafos sem pesos, o tamanho do corte mínimo que separa dois vértices s e t é igual ao número máximo de caminhos aresta-disjuntos que conectam s e t. Em grafos ca-

pacitados essa relação é dada pelo conhecido teorema do Corte Mínimo - Fluxo Máximo [Korte and Vygen 2002].

Uma definição importante é:

Definição 1 *Seja $G = (V, E)$ um grafo não direcionado. Considere um conjunto de vértices $X \subseteq V$. A aresta-conectividade de X em relação a G é o tamanho do corte mínimo que separa quaisquer pares de vértices em X. Denotamos a aresta conectividade de X por $\lambda(X)$.*

É importante observar que a aresta-conectividade de um conjunto de nodos X é diferente da aresta-conectividade de um subgrafo induzido por X. A razão é que os cortes mínimos são obtidos de G e não do subgrafo induzido por X. O subgrafo induzido por X pode não ter arestas e ainda X possuir a maior aresta-conectividade. A figura 2 ilustra tal situação, onde o conjunto $X = \{a, b, c, d\}$ tem aresta-conectividade 3 e o grafo induzido por X não possui nenhuma aresta [Cohen et al. 2011].

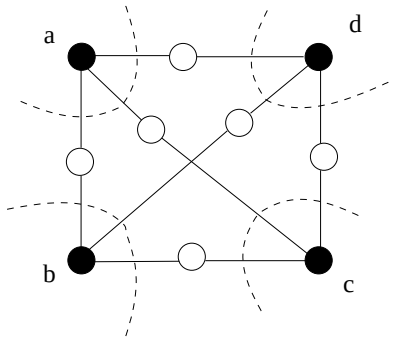


Figura 2. Conjunto de vértices com indicação dos cortes mínimos [Cohen et al. 2011]. A aresta-conectividade de $\{a, b, c, d\}$ é 3.

Em seguida definimos as medidas de conectividade baseadas em cortes de arestas.

Definição 2 *O número de conectividade de índice i de um nodo v, denotado por $\lambda_i(v)$, é o maior valor da aresta-conectividade de um conjunto $X \subseteq V$ satisfazendo:*

i. $v \in X$, *e*
ii. $|X| \geq i$

A figura 3 mostra um exemplo do critério de conectividade. Os números associados aos nodos são os números de conectividade $\lambda_2(v)$. As áreas circuladas identificam os componentes de maior conectividade. Para calcular os números de conectividade com índice 2, é necessário verificar a máxima aresta-conectividade de conjuntos com 2 nodos. Por exemplo, $\lambda_2(a) = 4$ pois os nodos a e b não podem ser separados por um corte com menos do que 4 arestas e a aresta-conectividade entre a e outro nodo qualquer é de no máximo 4. Quando o índice i em $\lambda_i(v)$ é 3, somente conjuntos com pelo menos 3 nodos são considerados. Por exemplo, $\lambda_3(d) = 4$ porque qualquer conjunto que contenha d com pelo menos três nodos tem aresta-conectividade pelo menos 4. Observe, entretanto, que $\lambda_2(d) = 5$.

Por convenção, $\lambda_1(v)$ é definido como o grau de v e $\lambda_n(v)$, onde $n = |V|$, é, por definição, o tamanho do menor corte de aresta global. O número de conectividade é uma medida unificada que, conforme i aumenta de 1 a $|V|$, $\lambda_i(v)$ inicia como uma propriedade local e termina como uma propriedade global da rede. Propriedades dos números de conectividade são discutidas em [Cohen et al. 2011].

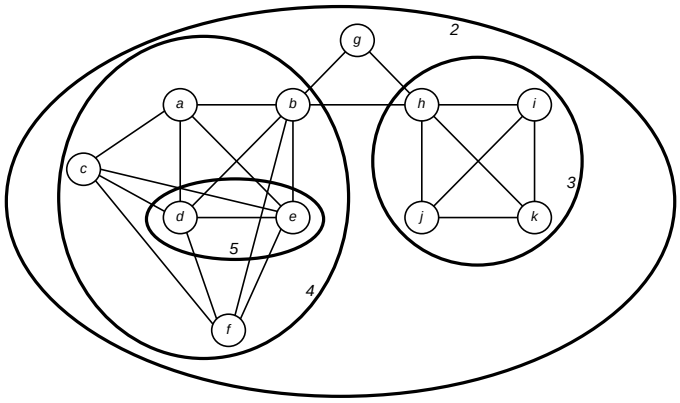


Figura 3. Conectividade baseada em cortes de arestas [Cohen et al. 2011].

3. Medidas de Conectividade Baseadas em Cortes de Vértices

As medidas baseadas em cortes de vértices tem por objetivo encontrar um valor que determine a importância de um nodo em relação a sua vértice-conectividade a outros nodos da rede. A medida é baseada no conceito de conectividade de vértices.

A rede que queremos caracterizar é representada por um grafo não direcionado $G = (V, E)$, onde V é um conjunto de vértices que representam os nodos desta rede e E o conjunto de arestas que representam os enlaces entre os nodos da rede.

Definimos a seguir a *vértice-conectividade* de um conjunto de vértices [Nagamochi and Ibaraki 2008].

Definição 3 *Dados dois vértices $s, t \in V$, a vértice-conectividade entre s e t, denotada por $\kappa(s, t)$, é o número de caminhos vértice-disjuntos entre s e t.*

Definição 4 *A vértice-conectividade de um conjunto $X \subseteq V$ é a menor vértice-conectividade entre quaisquer pares de vértices em X e é denotada por $\kappa(X)$.*

A medida de conectividade de nodos baseada em vértice-conectividade está definida a seguir:

Definição 5 *Definimos a i-vértice-conectividade de um vértice v, denotada por $\kappa_i(v)$, como a maior vértice-conectividade de um conjunto $X \subseteq V$ satisfazendo:*

i. $v \in X$, *e*
ii. $|X| \geq i$

Para ilustrar a diferença entre as medidas de conectividade baseadas em arestas e as medidas de conectividade baseadas em vértices, observe a figura 4. Ela mostra um exemplo de grafo com valores de aresta-conectividade diferentes do valor da vértice-conectividade. Note que os vértices 1 e 5 são os que possuem valores com maior discrepância. Por exemplo, $\lambda_2(1) = 5$ pois a máxima aresta-conectividade dos conjuntos da forma $X = \{1, u\}$, $u \in V - \{1\}$, é 5. Caso quiséssemos calcular a 3-aresta-conectividade, então tomaríamos conjuntos X com 3 vértices. Por outro lado, $\kappa_2(1) = 3$ pois a máxima vértice-conectividade dos conjuntos da forma $X = \{1, u\}$, $u \in V - \{1\}$, é 3. Note que o conjunto de vértices $\{2, 3, 4\}$ separam os vértice 1 e 5. Esse exemplo deixa claro que a vértice-conectividade se relaciona com os caminhos vértice-disjuntos e que a aresta-conectividade é relacionada com os caminhos aresta-disjuntos e que pode haver discrepância entre as duas medidas.

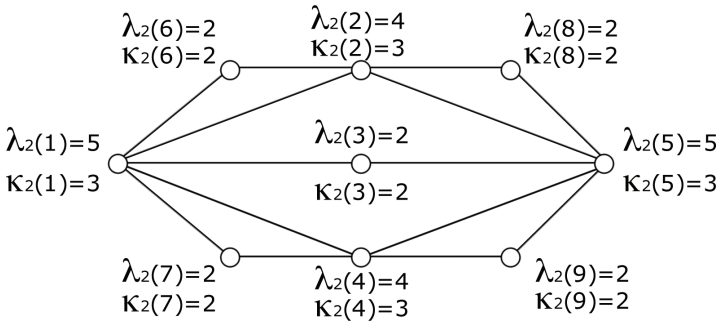


Figura 4. Grafo com discrepância entre os valores da aresta-conectividade e da vértice-conectividade.

Algorithm 1 Gera grafo com discrepância entre aresta-conectividade e vértice-conectividade

---

Entrada: k, discrepância desejada
Saída: grafo com discrepância k entre a aresta-conectividade e a vértice-conectividade
  for i de $k+1$ até 1 passo -1 **do**
    for j de 0 até 2^{k+1} passo 2^{k-i+1} **do**
      imprime aresta $\{j+1, j+2^{k-i+1}+1\}$
    end for
  end for
  imprime arestas $\{1, 2^{k+1}+2\}$, $\{2^{k+1}+2, 2^{k+1}+1\}$, $\{1, 2^{k+1}+3\}$ e $\{2^{k+1}+3, 2^{k+1}+1\}$

Lema 1 *Dado um grafo $G = (V, E)$ e $v \in V$, tem-se que $\kappa_i(v) \leq \lambda_i(v)$.*

Esse lema decorre do fato que todo caminho vértice-disjunto também é um caminho aresta-disjunto. Portanto, a vértice-conectividade de um par de vértices é menor do que ou igual à aresta-conectividade entre eles. O mesmo vale para a conectividade de conjuntos de vértices.

O Algoritmo 1 produz grafos com uma discrepância arbitrária entre a vértice-conectividade e a aresta-conectividade. Esses grafos generalizam o grafo da figura 4.

4. Algoritmo para Cálculo do Valor $\kappa_2(v)$

Para calcular o critério de conectividade baseado em vértices, que chamamos de $\kappa_i(v)$, necessita-se encontrar cortes de vértices mínimos. Ao invés de resolver diretamente o problema dos cortes de vértices, utilizamos uma estratégia descrita em [Nagamochi and Ibaraki 2008] que consiste em reduzir o problema dos cortes de vértices ao problema dos cortes de arestas. Essa redução se aplica a grafos orientados, portanto os grafos não-orientados são, primeiramente, transformados em grafos orientados.

Um grafo não orientado pode ser transformado em grafo orientado através da duplicação das arestas do grafo original, fazendo com que uma aresta $\{a, b\}$ corresponda a duas arestas direcionadas $\{(a, b), (b, a)\}$ no novo grafo.

Para encontrar os cortes de vértices do grafo orientado, criamos um novo grafo de forma que os cortes de arestas mínimos do grafo novo correspondam aos cortes de vértices mínimos do grafo original. Veja a figura 5. Para cada vértice v do grafo original, o grafo transformado contém dois vértices, v' e v''. As arestas com origem no vértice v do grafo

original têm como origem no grafo novo o vértice v''. As arestas com destino no vértice v no grafo original têm como destino v' no grafo transformado. Por fim, adicionamos ao novo grafo arestas direcionadas (v', v'') entre todos os pares de vértices associados. As arestas do tipo (v', v'') recebem capacidade de valor 1 e as demais arestas recebem capacidade $n - 1$. Essas capacidades garantem que os cortes de arestas de capacidade mínima utilizem somente as arestas que correspondem aos vértices do grafo original, o que justifica a redução entre os problemas.

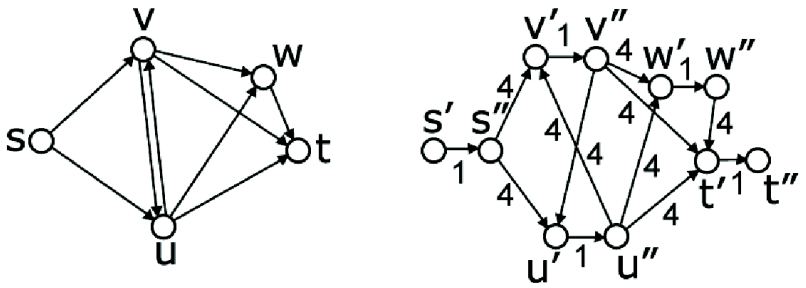


Figura 5. Transformação de grafo orientado para aplicação de corte de arestas [Nagamochi and Ibaraki 2008].

Nesse novo grafo aplicamos o algoritmo de fluxo máximo para encontrar o corte mínimo de arestas entre pares de vértices s e t. Para que a transformação corretamente encontre a vértice-conectividade, que foi definida em termos de caminhos vértice-disjuntos entre s e t, ainda é necessário tratar o caso em que a aresta $(s, t) \in E$. Nesse caso, a aresta (s'', t') recebe capacidade 1 para que seja incluída no corte mínimo. Assim, o valor do fluxo máximo entre s'' e t' no grafo transformado corresponde à vértice-conectividade entre s e t no grafo original.

Para calcular $\kappa_2(v)$ utilizamos a transformação descrita acima entre v e todos os demais vértices do grafo. A máxima vértice-conectividade encontrada é o valor $\kappa_2(v)$.

A complexidade de tempo do algoritmo para o cálculo de 2-vértice-conectividade requer o cálculo de $n - 1$ fluxos máximos para cada vértice. O cálculo do fluxo máximo é facilitado pela estrutura do grafo transformado que possui vértices que são o destino de apenas uma aresta ou são a origem de apenas uma aresta. Nesse caso, o algoritmo de Dinitz tem complexidade de tempo de $O(m\sqrt{n})$ [Nagamochi and Ibaraki 2008]. Logo, a complexidade de tempo para o cálculo de $\kappa_2(v)$ é $O(nm\sqrt{n})$.

Os valores de $\kappa_i(v)$ para $i > 2$ podem ser calculados considerando conjuntos de tamanho i, porém tal abordagem não é eficiente. Nós não conhecemos algoritmos eficientes para calcular $\kappa_i(v)$ em geral, ao contrário do que ocorre com $\lambda_i(v)$ que pode ser calculado em tempo polinomial para qualquer valor de i.

5. Resultados Experimentais

Nesta seção apresentamos alguns resultados experimentais. O algoritmo que calcula as medidas de conectividade baseadas em cortes de vértices foi implementado em linguagem *C++*. Um *script* na linguagem *python* foi utilizado para executar os experimentos e gerar os gráficos.

Utilizamos nos experimentos um grafo do modelo de rede de *Barabási-Albert* [Albert et al. 2000]. O modelo *Barabási-Albert* define um algoritmo que gera redes da classe *power-law*, livres de escala. O modelo possui dois importantes conceitos: o de crescimento e de preferência de conexão. O crescimento é o aumento do número de nodos na rede ao passar do tempo. A preferência de conexão é calculada da seguinte maneira: quanto maior é o grau do nodo, maior é a probabilidade dele receber novas conexões.

A rede utilizada nos experimentos tem 100 nodos. Apresentamos a seguir os gráficos das correlações entre a medida proposta e outras medidas de conectividade.

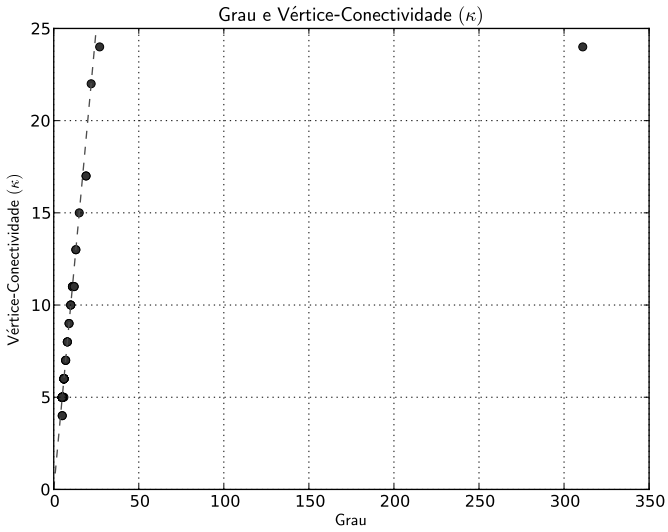


Figura 6. Relação entre medida de conectividade baseada em vértices e grau.

A figura 6 ilustra a diferença entre a medida proposta e o grau. O eixo x corresponde aos valores obtidos da medida grau e o eixo y corresponde aos valores obtidos com as medidas de conectividade baseadas em vértices. É possível perceber que para diversos pontos os valores do grau e da medida resultam próximos. Entretanto, em particular para o vértice de maior grau, a diferença é grande. Isso se deve ao fato de que o modelo utilizado gera várias conexões do primeiro nodo com o resto da rede. Nesse caso específico, enquanto o grau chega a 311, λ_2 não passa de 24. A medida proposta nos releva que existe também outro nodo muito importante na rede, que possui conectividade 22, e não somente o primeiro.

A figura 7 mostra a relação entre a medida de conectividade proposta e o grau de intermediação e releva uma diferença ainda maior para o primeiro nodo. Isso se deve ao fato de haver um único valor muito alto gerado pelo grau de intermediação.

A relação entre a medida de conectividade baseada em vértices e a medida *closeness* é ilustrada na imagem 8. A medida *closeness* tem todos os valores entre 0 e 1.

A excentricidade é um valor praticamente constante no grafo analisado. No caso, temos um nodo de valor 1 e todos os outros valores são 2, como pode ser visto no gráfico da figura 9.

Temos pequenas diferenças que podemos notar no gráfico da figura 10 entre as

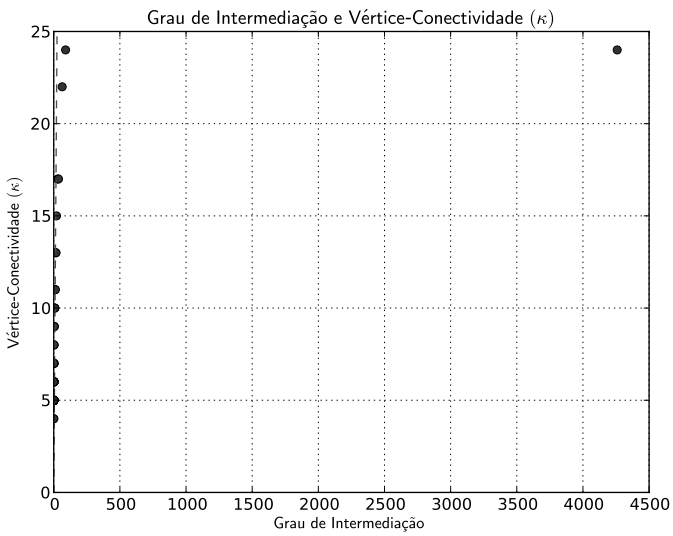


Figura 7. Relação entre medida de conectividade baseada em vértices e grau de intermediação.

medidas de conectividade baseadas em arestas e as medidas propostas. A diferença entre os valores médios é pequena mas está presente.

Na figura 11 temos o resultado dos valores gerados a partir de um experimento que utilizou um grafo gerado com o Algoritmo 1. Esse grafo tem como objetivo demonstrar que as medidas de conectividade baseadas em arestas e as medidas de conectividade baseadas em vértices podem ter discrepâncias grandes.

6. Conclusão

Nesse trabalho foram propostas medidas de conectividade baseadas em cortes de vértices. As medidas propostas podem contribuir para caracterização de redes complexas, uma vez que, para caracterizá-las, é necessário a seleção de um conjunto de medidas.

A i-vértice-conectividade de um vértice v, denotada como $\kappa_i(v)$, determina o valor da conectividade baseada em cortes de vértices para um vértice v. Essas medidas caracterizam os efeitos de falhas de nodos em uma rede e fundamenta-se no cálculo cortes de vértices. Para calcular o corte de vértices utilizamos uma redução do problema de cortes de vértices ao problema de cortes de arestas.

Observamos que o grau e o grau de intermediação (*betweenness*) amplificam casos extremos e raros que nem sempre são relevantes.

A desvantagem da medida proposta é o alto custo computacional do seu cálculo e a semelhança dos valores em relação a aresta-conectividade em diversos grafos. Em outras palavras, se estamos interessados em encontrar vértices com alta vértice-conectividade, talvez seja suficiente computar as aresta-conectividade dos mesmos.

Como trabalho futuro serão realizados mais experimentos e outras comparações com outros valores relacionados.

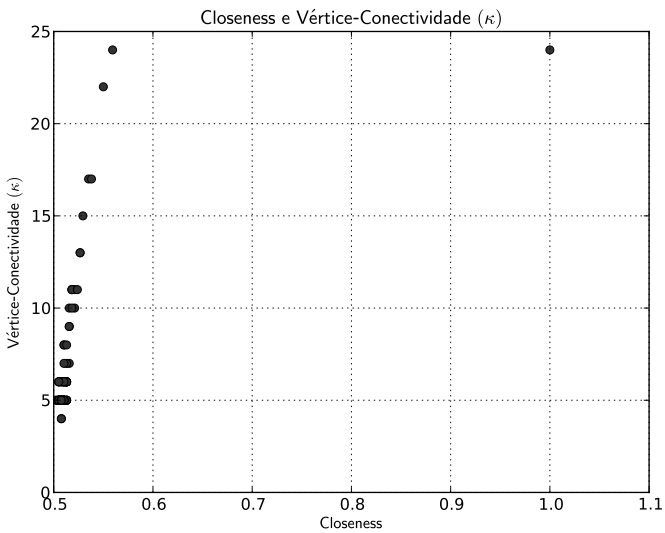


Figura 8. Relação entre medida de conectividade baseada em vértices e *closeness*.

Agradecimentos

Jaime Cohen é bolsista de doutorado da Fundação Araucária/SETI sob Edital No. 16/2008. Elias P. Duarte teve financiamento do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) sob contrato núm. 304013/2009-9.

Referências

Albert, R., Barabási, A. L., and Jeong, H. (2000). Scale-free characteristics of random networks: The topology of the world wide web. *Physica A*, 281.

Albert, R., Jeong, H., and Barabási, A. L. (1999). The diameter of the world wide web. *Nature*, 401.

Balthrop, J., Forrest, S., Newman, M. E. J., and Williamson, M. M. (2004). Technological networks and the spread of computer viruses. *Science*, 304.

Barabási, A. L. (2003). *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume.

Cohen, J., Schroeder, J., and Duarte Jr., E. P. (2011). Connectivity criteria for ranking network nodes. *Communications in Computer and Information Science (CCIS) - aceito para publicação*.

Costa, L. F., Rodrigues, F. A., Travieso, G., and Boas, P. R. V. (2006). Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1).

Echenique, P., Gómez-Gardeñes, J., and Moreno, Y. (2005). Dynamics of jamming transitions in complex networks. *EPL (Europhysics Letters)*, 71(2).

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, volume 29 of *SIGCOMM '99*.

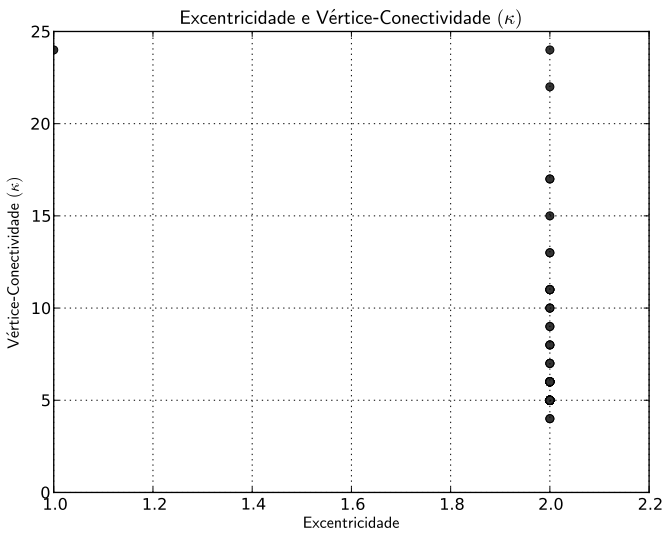


Figura 9. Relação entre medida de conectividade baseada em vértices e excentricidade.

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40.

Hage, P. (1995). Eccentricity and centrality in networks. *Social Networks*, 17.

Korte, B. and Vygen, J. (2002). *Combinatorial Optimization: Theory and Algorithms*. Springer, 2nd edition.

Leskovec, J., Huttenlocher, D., and Kleinberg, J. (2010). Signed networks in social media. *Proceedings of the 28th international conference on Human factors in computing systems*.

Liljeros, F., Edling, C. R., Amaral, L. A., Stanley, E. H., and åberg Y. (2001). The web of human sexual contacts. *Nature*, 411.

Duarte Jr., E. P., Santini, R., and Cohen, J. (2004). Delivering packets during the routing convergence latency interval through highly connected detours. In *DSN*. IEEE Computer Society.

Nagamochi, H. and Ibaraki, T. (2008). *Algorithmic Aspects of Graph Connectivity*. Cambridge University Press.

Okamoto, K., Chen, W., and Li, X. (2008). Ranking of closeness centrality for Large-Scale social networks. *Frontiers in Algorithmics*, 5059.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of small-world networks. *Nature*, 393.

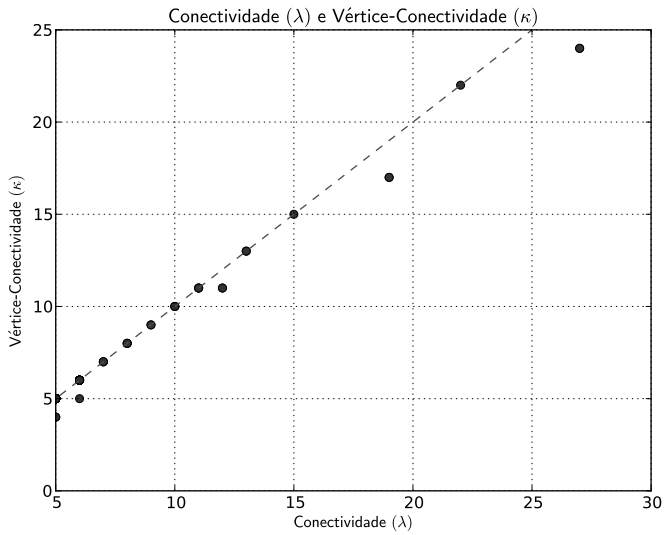


Figura 10. Relação entre a medida de conectividade baseada em vértices e a medida de conectividade baseada em arestas.

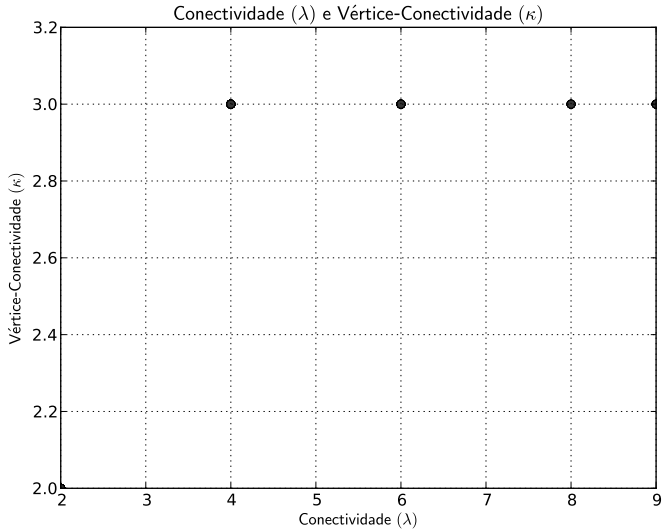


Figura 11. Relação entre a medida de conectividade baseada em vértices e a medida de conectividade baseada em arestas em modelo criado.

Índice por Autor

A

Alchieri, E. A. P. **63**
Assunção, W. K. G. **15**

B

Basso, T. **29**
Bessani, A. N. **63**
Bona, L. C. E. **3**

C

Cohen, J. **77**
Colanzi, T. E. **15**
Correia, M. P. **45**

D

de Freitas, E. P. **49**

F

Fernandes, P. C. S. **29**
Ferreira, A. M. **49**
Fraga, J. da S. **63**

G

Geyer, C. F. R. **49**
Greve, F. **45**

H

Heimfarth, T. **49**

J

Júnior, E. P. D. **3**, **77**

K

Koppe, J. P. **3**

L

Larsson, T. **49**

M

Moraes, R. **29**

P

Pereira, C. E. **49**
Pires, K. **77**
Pozo, A. T. R. **15**

R

Rolim, C. O. **49**

T

Tomelin, L. R. **63**

V

Vergilio, S. R. **15**

W

Wagner, F. R. **49**

ISSN 2177-496X

Organização:

FACOM-UFMS

UFMS

Realização:

SBC

LARC

Patrocínios:

egi.br nic.br

CNPq

CAPES

Fundect

VALSPE GROUP

SERPRO

POLYCOM

DIGITALNETBR

Google

Netware Enterprise

SICREDI Gente que coopera cresce.