Simulação de um Algoritmo de Diagnóstico Distribuído para Redes Particionáveis de Topologia Arbitrária

Andréa Weber[1,2], Aline Wolpert dos Santos[1],
Elias Procópio Duarte Jr.[1], Keiko V. O. Fonseca[2]

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

[2]Prog. de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI)
Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR - Brazil

{andrea,aws03,elias}@inf.ufpr.br, keiko@utfpr.edu.br

Abstract. *This work presents experimental results of the evaluation of the* Distributed Network Reachability *algorithm. DNR is a distributed system-level diagnosis algorithm that allows every node of a partitionable general topology network to determine which portions of the network are reachable and unreachable. Extensive simulation results of dynamic fault and repair events on nodes and links are presented. The systems considered are highly dynamic, the occurrence of the events is modeled by a Poisson process. All the results are presented within a* 95% *confidence interval. Simulated topologies included random graphs as well as regular graphs, such as meshes and hypercubes.*

Resumo. *Este trabalho apresenta resultados experimentais da avaliação do algoritmo* Distributed Network Reachability. *DNR é um algoritmo de diagnóstico distribuído em nível de sistema que permite que cada nodo de uma rede particionável de topologia arbitrária determine quais porções da mesma estão atingíveis e inatingíveis. São apresentados extensivos resultados de simulação de eventos dinâmicos de falha e recuperação de nodos e enlaces. São considerados sistemas fortemente dinâmicos, com a ocorrência de eventos modelada por um processo de Poisson. Todos os resultados apresentados têm intervalo de confiança de* 95%*. Os resultados foram obtidos para grafos aleatórios e também para topologias regulares, como* meshes *e hipercubos.*

1. Introdução

Sistemas computacionais de grande porte baseados em múltiplos processadores, tais como computadores paralelos, *clusters* e redes de computadores são plataformas críticas para diversos tipos de organizações. É altamente desejável que tais sistemas se mantenham operando apesar da falha de alguns de seus componentes. Quando partes do sistema se tornam desconectadas, atividades de reconfiguração devem ser executas com presteza. Neste trabalho são apresentados resultados de uma avaliação experimental do algoritmo *Distributed Network Reachability* (DNR) para monitoramento *on line* tolerante a falhas de sistemas particionáveis baseados em redes de topologia arbitrária. O algoritmo pode ser utilizado para construir um mapa refletindo quais porções da rede estão atingíveis e inatingíveis do ponto de vista de qualquer nodo da rede.

O algoritmo *Distributed Network Reachability* (DNR) [Weber 2008] é o primeiro algoritmo distribuído de diagnóstico em nível de sistema para redes de topologia arbitrária que executa na presença de particionamentos e reconexões da rede causados por eventos dinâmicos de falha e recuperação de nodos e enlaces. O algoritmo consiste de três fases: teste, disseminação e cálculo de alcançabilidade. Durante a fase de testes, cada enlace é testado por um de seus nodos adjacentes em intervalos de teste alternados. Quando da detecção de um novo evento, i.e., um enlace *sem-falha* se torna *não-respondendo* ou vice-versa, o testador inicia a fase de disseminação, na qual uma estratégia paralela é utilizada para informar os outros nodos alcançáveis sobre o evento. Novos eventos podem ocorrer e ser detectados antes que a disseminação seja completada. Sempre que um evento é detectado ou informado, a terceira fase é executada, na qual um algoritmo de conectividade em grafos é empregado para computar a alcançabilidade da rede.

O desempenho do algoritmo foi avaliado através de simulação, considerando múltiplos eventos de falha e recuperação, tanto de nodos como de enlaces. Além disso, tanto as situações na qual a rede sofre particionamento como na qual ela permanece sempre conectada foram avaliadas. Experimentos foram executados tanto em topologias aleatórias, como em topologias regulares como *meshes* e hipercubos. O programa de simulação foi construído usando a biblioteca de simulação de eventos discretos SMPL [MacDougall 1987]. Os experimentos foram realizados no *cluster* de alto desempenho da Universidade Federal do Paraná, composto por cerca de 60 máquinas multiprocessadas conectadas em rede *Myrinet* e *Gigabit-Ethernet*.

O restante do artigo é organizado como segue. A seção 2 apresenta trabalhos relacionados. Na seção 3 o algoritmo DNR é descrito. A seção 4 apresenta resultados de simulação da ocorrência de um grande número de eventos concorrentes em várias topologias aleatórias e regulares. A conclusão segue na seção 5.

2. Trabalhos Relacionados

Muitos algoritmos de diagnóstico em nível de sistema para redes de topologia arbitrária tem sido propostos. [Bagchi and Hakimi 1991] introduziram o primeiro algoritmo de diagnóstico para este tipo de rede, o qual era executado *off-line* a partir de resultados de testes prévios. [Stahl, Buskens and Bianchini 1992] introduziram e avaliaram através de simulação o algoritmo Adapt, que pode ser executado *on line*: quando um nodo falha, os outros nodos reconectam o grafo de testes. A estratégia de disseminação de eventos empregada é sequencial. [Rangarajan, Dahbura and Ziegler 1995] introduziram o algoritmo RDZ para diagnóstico em nível de sistema de redes de topologia arbitrária. O algoritmo constrói um grafo de testes que garante um número ótimo de testes, e utiliza uma estratégia paralela de disseminação baseada em inundação. O algoritmo, entretanto, não garante o diagnóstico de algumas configurações de falhas.

[Duarte et. al. 1997] introduziram um algoritmo que utiliza uma estratégia de testes chamada *two-way testing*, em que o nodo testado determina o estado do testador. Uma estratégia de testes baseada em *token* foi proposta posteriormente, na qual os nodos conectados por um enlace se alternam nos papéis de testado e testador. Informações sobre eventos detectados são disseminadas utilizando uma árvore distribuída de busca em largura. O algoritmo DNC (*Distributed Network Connectivity*) [Duarte Jr. and Weber 2003] baseia-se naquele algoritmo, mas suporta eventos dinâmicos, i.e. durante a fase de disseminação

novos eventos podem ocorrer e a disseminação de todos é garantida. Posteriormente uma outra estratégia de testes foi desenvolvida [Weber, Duarte Jr. and Fonseca 2006] a qual garante a resolução de casos que levam a testes ditos simultâneos.

[Subbiah and Blough 2004] introduziram o algoritmo ForwardHeartbeat, que é baseado em *heartbeats* e permite o diagnóstico em redes de topologia arbitrária numa situação dinâmica de ocorrência de eventos. O algoritmo não permite particionamento da rede. [Khanna et. al. 2007] apresentam uma aplicação que realiza diagnóstico de falhas em protocolos de redes de larga escala. A aplicação considera as entidades a serem diagnosticadas como caixas pretas e observa a troca de mensagens entre elas. Um grafo causal é construído para seguir a propagação do erro e identificar sua fonte.

Algoritmos de diagnóstico distribuído para redes sem-fio também são trabalhos relacionados. As topologias das redes sem-fio não são apenas arbitrárias como também dinâmicas. [Santi and Blough 2002] consideram o problema de calcular o quão grande o raio de transmissão dos nodos em uma rede móvel *ad hoc* deve ser de forma a assegurar que a rede resultante seja conectada; eles também computam o raio para assegurar a conectividade durante uma dada fração de tempo quando os nodos são móveis. [Chessa and Santi 2002] propuseram um protoloco distribuído para diagnóstico de falhas em redes de sensores. O protocolo é iniciado por um observador externo e tanto o diagnóstico como a disseminação de seu resultado ocorrem ao longo de uma árvore. [Ding et. al. 2005] apresentam um algoritmo tolerante a falhas para detectar nodos falhos e a borda de propagação de um evento em uma rede de sensores. [Elhadef, Boukerche and Elkadiki 2007] propõem um algoritmo adaptativo de diagnóstico distribuído baseado em comparações para redes móveis *ad hoc*, baseado no trabalho de [Chessa and Santi 2001]. O resultado dos testes realizados localmente pelos sensores é disseminado pela rede por meio de um árvore geradora mínima adaptável. [Lee, M. H. and Choi Y. H. 2007] apresentam um algoritmo distribuído eficiente baseado no modelo de comparações para isolar nodos falhos em uma rede de sensores, utilizando a premissa de que nodos sem-falha apresentam valores de leitura similares aos de seus vizinhos.

3. Descrição do Algoritmo

Nesta seção é descrito o algoritmo *Distributed Network Reachability* [Weber 2008] para redes particionáveis de topologia arbitrária. Um nodo executando o algoritmo obtém informação sobre o componente conexo da rede ao qual pertence, i.e., quais porções da rede estão alcançáveis e inalcançáveis.

Numa rede de topologia arbitrária não há necessariamente um canal de comunicação entre quaisquer pares de nodos. Tanto nodos como enlaces podem se tornar falhos, considerando falhas *crash* que podem ser reparadas. Uma falha pode particionar a rede, que pode subsequentemente reconectar. Considerando um par de nodos conectado por um enlace, se os testes executados em um nodo pelo outro determinam que o enlace não está respondendo, é impossível distinguir se é o nodo testado ou o enlace que está falho. Desta forma, falhas em redes de topologia arbitrária são ditas ambíguas [Duarte et. al. 1997]. Além disso, como o sistema não é síncrono, um *timeout* pode ser causado por um nodo falho ou muito lento. Neste caso, quando não há resposta a um teste, o enlace é considerado *não-respondendo*. A qualquer instante de tempo, um nodo

não falho executando DNR classifica os outros nodos em um de dois estados: *sem-falha* ou *inatingível*. Um enlace no qual um nodo sem-falha está conectado pode estar *sem-falha* ou *não-respondendo*, e outros enlaces podem estar *sem-falha*, *não-respondendo* ou *inatingíveis*. Um nodo sem-falha considera um enlace como *inatingível* quando o enlace é não adjacente a nenhum outro nodo alcançável sem-falha.

O algoritmo consiste de três fases: teste dos enlaces locais, disseminação de informação sobre novos eventos e cálculo local de alcançabilidade. Testes são executados periodicamente em intervalos de teste, que são intervalos de tempo. O assinalamento de testes é dinâmico, de forma que dois nodos que compartilham um enlace de comunicação se alternam nos papéis de *testador* e *testado* e um número ótimo de testes por enlace por intervalo de testes é assegurado: um teste por enlace por intervalo de testes, desde que ambos os nodos adjacentes estejam sem-falha. O algoritmo emprega uma estratégia paralela de disseminação que apresenta uma latência de disseminação proporcional ao diâmetro da rede. Ambos nodos e enlaces podem falhar e recuperar durante a execução do algoritmo. A rede pode particionar e subsequentemente reconectar.

Sob o ponto de vista de um nodo testador, um *evento* é definido como uma mudança de estado de um enlace, ou de *sem-falha* para *não-respondendo* ou de *não-respondendo* para *sem-falha*. Embora um nodo execute testes enviando uma requisição de testes para um nodo vizinho, um evento é sempre descrito em termos do enlace de comunicação correspondente. Os nodos mantém uma visão local da topologia da rede representada por um grafo, no qual um *timestamp* é mantido para cada enlace. Os *timestamps* empregados são contadores de eventos similares àqueles propostos em [Rangarajan, Dahbura and Ziegler 1995]. Os *timestamps* para todos os enlaces são inicialmente 1 e todos os enlaces são assumidos como não-respondendo. A cada vez que um novo evento é detectado, o *timestamp* para aquele enlace é incrementado. Portanto, um *timestamp* par corresponde a um enlace *sem-falha*; um *timestamp* ímpar corresponde a um enlace *não-respondendo*.

O intervalo de testes leva em conta a tecnologia e os requisitos e é estimado de forma a evitar a perda de eventos. Ao iniciar, um nodo executando o algoritmo testa todos os seus vizinhos após expirar um intervalo de tempo pré-definido chamado *recovery wait time* [Subbiah and Blough 2004]. Durante este intervalo de tempo o nodo que está recuperando não envia requisições ou respostas a testes e não processa mensagens de disseminação. Isto permite que um nodo fique falho durante um tempo mínimo e garante que enlaces *não-respondendo* sejam sempre detectados. Levando em conta o *recovery wait time* e o intervalo de testes, são formalmente calculados os intervalos de tempo que nodos e enlaces devem se manter nos estados falho e sem-falha de forma que uma mudança de estados seja sempre detectada. Estes intervalos de tempo são designados como *state holding times* [Subbiah and Blough 2004].

O procedimento de teste adotado pelo algoritmo é dito *two-way* no sentido em que permite que ambos os nodos testador e testado descubram mutuamente se estão sem-falha. O assinalamento de testes é baseado em *token* e se apóia no teste *two-way*. Cada par de nodos conectado por um enlace mantém um *token*; o nodo que tem o *token* no início de um intervalo de testes é o testador, o outro é o nodo testado. Se ambos os nodos estão sem-falha durante o teste o *token* é transferido. O processo é repetido e garante que somente um teste seja executado por enlace por intervalo de testes.

No caso em que um nodo se torna falho, este pode ser tanto o nodo testado ou o testador para o próximo intervalo de testes. Se o nodo testado falha, o testador se manterá testando a cada dois intervalos de teste. Se o testador para o próximo intervalo de testes falha, o *token* desaparece. O nodo testado detecta que não recebeu uma requisição de teste e no intervalo de testes seguinte um *token* é criado. O nodo assume o papel de testador, executando um teste a cada dois intervalos de teste. No caso de o enlace se tornar falho, ambos os nodos testam a cada dois intervalos de teste, alternadamente. Após a recuperação do enlace, o primeiro teste feito por um dos nodos faz com que o outro responda e se torne o testador para o intervalo seguinte.

Quando um nodo recupera, ele cria um *token* para si mesmo e, após expirado o *recovery wait time*, testa todos os nodos ao qual está conectado. Testes ditos *simultâneos* podem ocorrer se ambos os nodos recuperam praticamente ao mesmo tempo e se tornam testadores, ou quando um nodo recupera e testa um vizinho exatamente ao mesmo tempo em que é testado por ele. Quando ambos os nodos recebem, um do outro, uma requisição de testes, a situação é detectada e um critério baseado no identificador dos nodos é utilizado para determinar qual nodo responderá ao teste, tornando-se o nodo testado.

A fase de disseminação é iniciada com a detecção de um novo evento. A estratégia de disseminação empregada é paralela, baseada em inundação, iniciada pelo testador que detectou o novo evento. Este nodo monta uma mensagem de disseminação contendo informação de diagnóstico. Cada mensagem de diagnóstico tem: (1) o identificador do nodo testador, (2) o identificador do vizinho testado, e (3) o *timestamp* do enlace testado. O nodo que inicia a disseminação envia a mensagem correspondente em todos os seus enlaces. Um nodo que recebe uma mensagem de disseminação verifica se ela contém ou não nova informação. A informação é nova quando contém *timestamps* maiores para um ou mais enlaces que aqueles contidos na tabela de enlaces local. Se a mensagem contém nova informação, o nodo envia esta informaÃ§Ã£o em todos os seus enlaces, exceto no enlace por onde a mensagem foi recebida, e desta forma sucessivamente até que a disseminação alcance todos os nodos.

Se nodos ou enlaces falham, a rede pode se particionar em dois ou mais componentes conexos. Ambos os nodos conectados pelo enlace detectam o evento e disseminam informação sobre o mesmo a todos os nodos sem-falha alcançáveis em cada partição. Um evento de *healing* pode ocorrer e é definido por um enlace ou nodo falhos que é reparado e pode fazer com que duas ou mais partições se reconectem em um único componente conexo. Quando um nodo recupera, eventos de *healing* ocorrem em todos os seus enlaces sem-falha. Um evento de *healing* é detectado pela chegada de uma requisição de teste sobre um enlace considerado *não-respondendo*. Ao tempo em que o nodo testado responde à requisição de teste, ele também percebe que o enlace está sem-falha, com os testes *two-way*. O nodo testado responde com uma mensagem de *healing*, a qual leva informação sobre os *timestamps* dos enlaces localizados em sua partição. Quando da recepção da mensagem de *healing*, o testador atualiza sua tabela de enlaces local, incrementa o *timestamp* do enlace recuperado e inicia a disseminação de informação sobre toda a tabela. Esta estratégia permite que ambos os nodos troquem informação sobre os componentes aos quais pertenciam, de forma que *timestamps* atualizados de enlaces previamente inalcançáveis atinjam toda a rede.

A cada vez que um nodo detecta ou recebe informação sobre um novo evento,

a terceira fase é executada, na qual um algoritmo de conectividade em grafos mostra a alcançabilidade da rede do ponto de vista do nodo executando o algoritmo.

4. Resultados Experimentais

Nesta seção são apresentados resultados de simulação do algoritmo DNR. O desempenho do algoritmo foi avaliado tanto em eventos de falha como em eventos de recuperação, nos casos em que tanto nodos como enlaces se tornaram falhos ou recuperaram. Além disso, tanto as situações na qual a rede sofre particionamento como na qual ela permanece sempre conectada foram avaliadas. Experimentos foram executados tanto em topologias aleatórias como em topologias regulares como *meshes* e hipercubos.

O programa de simulação foi construído usando a biblioteca de simulação de eventos discretos SMPL [MacDougall 1987]. Os experimentos foram realizados no *cluster* de alto desempenho da Universidade Federal do Paraná, composto por cerca de 60 máquinas, as principais com 16 processadores e 16 GB de memória cada, bem como máquinas com 8 processadores e 8 GB de memória cada. Os processadores incluem *Opteron* 1.8 GHz com 1MB de *cache* L2, Intel *quad-core* 2.4 GHz e Athlon, conectados por rede *Myrinet* e *Gigabit-Ethernet*. O ambiente nas máquinas é exclusivamente de 64 bits, baseado no sistema operacional Linux. Há um servidor central de disco no qual as máquinas de cálculo fazem *boot* remoto.

Os experimentos foram conduzidos de forma a medir as latências de diagnóstico de eventos de falha e de recuperação tanto em nodos como em enlaces. A latência de diagnóstico inclui a latência de detecção do evento, i.e., o tempo para um determinado nodo detectar o evento em um enlace adjacente, e a latência de disseminação de informação sobre o mesmo, i.e., o tempo necessário para que os demais nodos alcançáveis sem-falha tomem conhecimento do evento ocorrido. De forma a obter um intervalo de confiança igual a 95% para as médias das latências, cinco rodadas independentes foram inicialmente executadas para um dado tipo e tamanho de topologia; 5000 eventos de falha e recuperação (2500 cada) foram escalonados em cada rodada. A semi-amplitude do intervalo de confiança foi avaliada até ser menor ou igual a 10% da média obtida para todas as rodadas. O dinamismo da ocorrência de eventos foi dado por um processo de Poisson. Tão logo um evento ocorre em um nodo ou em um enlace, o evento seguinte no mesmo nodo ou enlace é escalonado para ocorrer após o *state holding time* correspondente mais um período de tempo exponencialmente distribuído. Os eventos são concorrentes, escalonados para ocorrer em qualquer nodo ou enlace. Duas médias para o processo de Poisson foram utilizadas: 1 segundo e 200 segundos, portanto foram simulados sistemas altamente dinâmicos. A menor média da distribuição exponencial corresponde a um ambiente mais dinâmico, onde nodos mudam de estado com mais frequência, pouco tempo depois de permanecer em um estado durante o *state holding time* correspondente. A média de 200 segundos para a distribuição exponencial simula o ambiente oposto, no qual os nodos permanecem em cada estado por mais tempo.

Um conjunto de experimentos foi conduzido de forma a comparar o desempenho dos algoritmos DNR e *ForwardHeartbeat*. Nestes experimentos, os parâmetros de simulação e o dinamismo da ocorrência de eventos são similares àqueles utilizados em [Subbiah and Blough 2004].

Os parâmetros de simulação estão listados na Tabela 1. Os parâmetors Δ_{send_init},

Δ_{send_min} e Δ_{send_max} significam, respectivamente, o tempo para um nodo iniciar uma comunicação e os tempos mínimo e máximo de atraso num canal de comunicação. O intervalo de testes é dado por π. Seus valores foram mantidos fixos em todos os experimentos.

Tabela 1. Parâmetros de Simulação

Parâmetros de Simulação	
Simulador	SMPL
Δ_{send_init}	0.002 segundos
Δ_{send_min}	0.008 segundos
Δ_{send_max}	0.08 segundos
π	30 segundos
Médias para o processo de Poisson	1 segundo, 200 segundos
# de rodadas iniciais	5
# de eventos por rodada	2500
Intervalo de confiança	95%

Para cada uma das topologias utilizadas nos experimentos, suas conectividades de vértices e de arestas foram avaliadas utilizando o algoritmo de Ford-Fulkerson [Cormen et. al. 2001]. A conectividade de vértices (arestas) de um grafo é o menor número de vértices (arestas) cuja remoção pode desconectar o grafo. Esta propriedade é utilizada para construir dois cenários de simulação: o cenário no qual particionamentos da rede não ocorrem e o cenário no qual a rede pode se tornar particionada. Ao simular eventos em nodos no primeiro cenário, o número máximo de $k-1$ nodos foi permitido estar no estado falho simultaneamente, tendo a rede conectividade de vértices igual a k. Similarmente para eventos em enlaces. Neste cenário, se o número de nodos ou enlaces falhos é igual a $k-1$ quando um evento de falha em nodo ou enlace vai ocorrer, um período de tempo exponencialmente distribuído é utilizado para reescalonar aquele evento para um instante de tempo posterior. Por outro lado, ao simular o cenário no qual a rede pode sofrer particionamento e reconexão, eventos de falha foram escalonados em número maior que o da conectividade da rede.

Os experimentos foram executados em quatro tipos de topologias. O conjunto de topologias é mostrado na Tabela 2. Primeiramente, grafos com conectividade de vértices igual a k gerados aleatoriamente foram obtidos como em [Subbiah and Blough 2004]. Uma topologia inicial de n nodos com grau k foi obtida aleatoriamente. A conectividade de vértices da topologia foi então repetidamente avaliada. A cada vez em que a conectividade desejada não era atingida, n enlaces adicionais eram aleatoriamente introduzidos, até que uma topologia com conectividade de vértices igual a k era obtida. Topologias foram geradas para n igual a 32, 64, 128 e 256 nodos com $k = 3$ e $k = \log n$. Cinco diferentes topologias foram geradas para cada valor de n e k.

Topologias com distribuição *Power-Law* refletem a topologia da própria Internet, onde poucos nodos possuem muitos enlaces e a imensa maioria dos nodos possuem poucos enlaces. Grafos aleatórios com a distribuição *Power-Law* foram obtidas utilizando o gerador de [Bu and Towsley 2002]. Naquele gerador, o número total n de nodos e o número inicial de nodos no *backbone* são dados. A cada iteração, novos nodos são criados com dada probabilidade p. Com probabilidade $1-p$, novos enlaces são criados.

Tabela 2. Topologias utilizadas nas simulações

Topologias	
Grafos aleatórios com conectividade de vértices $k = 3$	32, 64, 128, 256 nodos
Grafos aleatórios com conectividade de vértices $k = \log n$	32, 64, 128, 256 nodos
Grafos aleatórios com distribuição *Power-Law*	32, 64, 128, 256 nodos
Topologias *Mesh*	64, 256 nodos
Hipercubos	32, 64, 128, 256 nodos

Neste caso, os nodos que serão conectados pelo novo enlace são escolhidos de acordo com um parâmetro, $\beta \in (-\infty, 1)$, o qual define o grau de preferência por nodos com muitos vizinhos. Quanto menor o valor de β, menor a preferência dada por um enlace a nodos com grau alto. Com $p = 0.6$ e $\beta = 0.2$, cinco diferentes topologias foram geradas para cada número n de 32, 64, 128 e 256 nodos. Estas topologias foram todas avaliadas como tendo conectividades de vértices e arestas iguais a 1, sendo, portanto, adequadas a experimentos com particionamento da rede.

Além destas, as topologias regulares *mesh 8x8* e *16x16* [Culler and Singh 1999] e hipercubos com 32, 64, 128 e 256 nodos foram também empregadas nos experimentos de simulação.

Como cinco diferentes topologias aleatórias de cada tipo foram geradas para cada número n de nodos, uma rodada foi simulada em cada topologia, e as médias das cinco rodadas foram testadas para a precisão do intervalo de confiança desejada. Com topologias regulares, como apenas uma topologia de cada tipo existe para cada número n de nodos, cinco rodadas independentes foram obtidas variando a semente para geração de números aleatórios em cada rodada.

Os experimentos são apresentados a seguir em duas seções; na segunda seção são apresentados resultados de simulação com particionamentos da rede.

4.1. Experimentos sem Particionamento da Rede

O primeiro conjunto de experimentos de simulação foi executado para propósitos de comparação com o algoritmo *ForwardHeartbeat*. Primeiramente, eventos de falha e de recuperação ocorrendo exclusivamente em nodos foram simulados com ambas as médias da distribuição exponencial de 1 segundo e de 200 segundos, em topologias com conectividade de vértices igual a 3. Os resultados da latência de diagnóstico são mostrados nas Figuras 1 e 2. Após isso, eventos de falha e de recuperação ocorrendo em nodos foram simulados em experimentos com média de 200 segundos para o processo de Poisson em ambas as redes com conectividade de vértices iguais a 3 e $\log n$. Os resultados são mostrados nas Figuras 3 e 4.

Como pode ser observado, as latências médias não são influenciadas nem pela conectividade da rede nem pelo dinamismo da ocorrência de eventos dado pelo processo de Poisson. Latências médias para eventos de falha são da ordem de um terço do intervalo de testes com conectividades da rede e médias da distribuição exponencial variadas. As latências médias do *ForwardHeartbeat* são da ordem de dois terços do intervalo entre *heartbeats* nas mesmas condições. As latências máximas do DNR, por outro lado, mostram um acréscimo em redes pequenas. Isto é devido ao fato de que, com poucos nodos, é maior a probabilidade de que eventos de falha em nodos contíguos ocorram. Assim, é

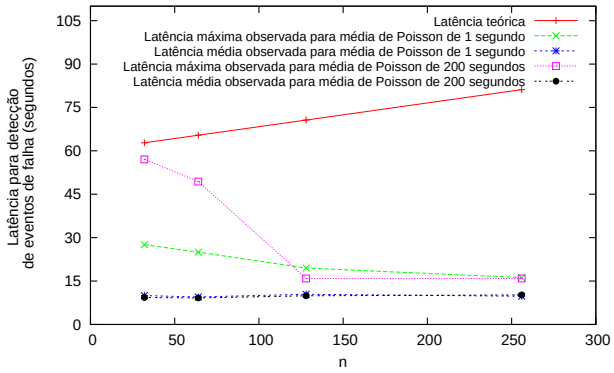

Figura 1. Eventos de falha em nodos - conectividade de vértices igual a 3 - grafos aleatórios.

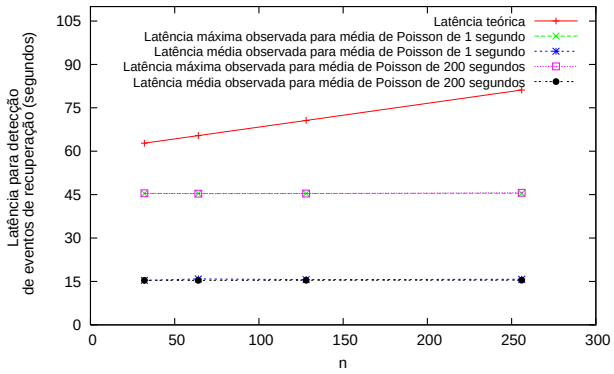

Figura 2. Eventos de recuperação em nodos - conectividade de vértices igual a 3 - grafos aleatórios.

possível que o próximo testador em um intervalo de testes esteja falho quando seu vizinho também falha. Com uma média de 200 segundos para o processo de Poisson, aquele testador permanece falho por mais tempo. Então pode levar até o máximo de dois intervalos de teste para que a falha do nodo vizinho, descrita acima, seja detectada por outro nodo.

Em experimentos de recuperação, a latência máxima ocorre em casos de testes simultâneos. Como estes casos são raros, eles não influenciam as latências médias, que são mantidas na ordem da metade do intervalo de testes. Além disto, em oposição aos experimentos com *ForwardHeartbeat*, a latência é calculada levando em conta o *recovery wait time*. De fato, tão logo expira o tempo de $W = 15.026516$ segundos, cada recuperação de um nodo é detectada. Se o *recovery wait time* não fosse levado em conta, as latências de recuperação do DNR também seriam da ordem de décimos de segundos, i.e. da ordem apenas do tempo de disseminação de cada evento. As latências médias também não crescem com o acréscimo no número de nodos na rede porque o diâmetro da rede mantém-se pequeno em comparação com n. O diâmetro máximo observado é de 25 em uma rede de 256 nodos e é da ordem de 7 em redes com 32 nodos.

Em ambos os tipos de experimentos, a latência teórica não muda com diferentes parâmetros de simulação ou tipos de eventos e é uma função do número n de nodos na rede.

Um segundo conjunto de experimentos com eventos em nodos foi executado em

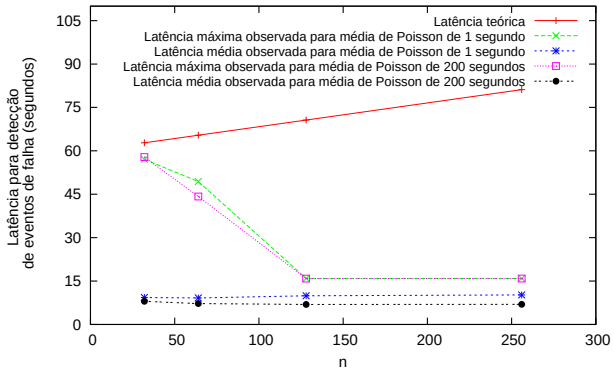

Figura 3. Eventos de falha em nodos - média de Poisson de 200 segundos - grafos aleatórios.

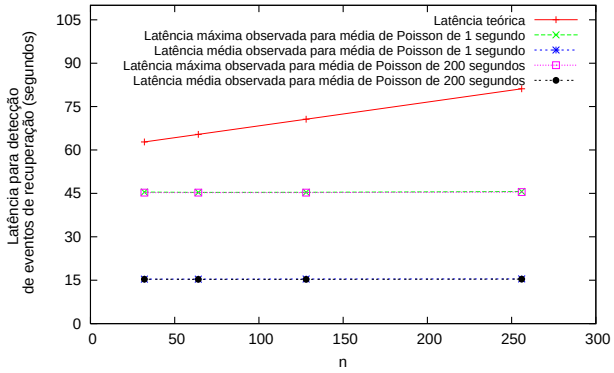

Figura 4. Eventos de recuperação em nodos - média de Poisson de 200 segundos - grafos aleatórios.

topologias regulares. As topologias foram redes *mesh* e hipercubos, e novamente as médias para o processo de Poisson foram de 1 segundo e 200 segundos. A conectividade de vértices de redes *mesh* é 4 e a conectividade de vértices de hipercubos é $\log n$. Um número máximo de nodos igual à conectividade menos 1 esteve falho simultaneamente em dado instante de tempo.

As latências médias de eventos de falha são novamente da ordem de um terço do intervalo de testes, como pode ser visto nas Figuras 5 e 6. Novamente as latências máximas mostram um acréscimo em redes pequenas para experimentos com média da distribuição exponencial de 200 segundos. Com média da distribuição exponencial igual a 1 segundo, a latência máxima apresenta um pequeno acréscimo com o acréscimo do tamanho da rede.

Os gráficos dos experimentos de *healing* exibem exatamente as mesmas curvas que aqueles do primeiro conjunto de experimentos acima e portanto não serão mostrados aqui. De fato, como um evento de recuperação de um nodo é detectado tão logo aquele nodo conclua o *recovery wait time*, nenhuma mudança na topologia influencia estes resultados, desde que o nodo em recuperação tenha ao menos um vizinho para detectar o evento.

Um outro conjunto de experimentos de simulação foi executado nas mesmas to-

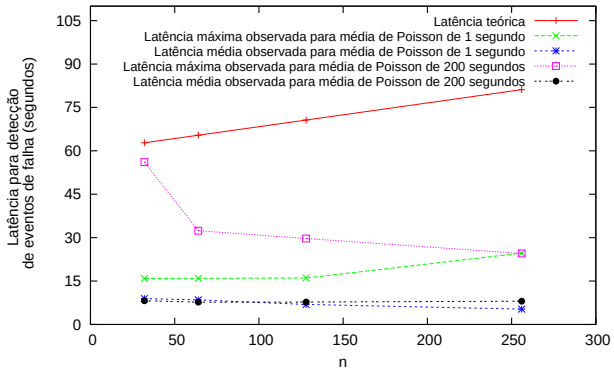

Figura 5. Eventos de falha em nodos - hipercubos.

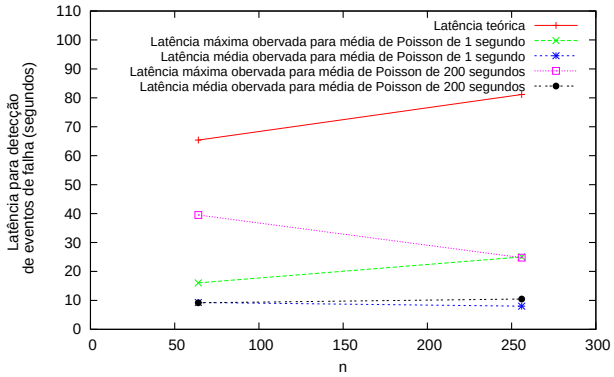

Figura 6. Eventos de falha em nodos - redes *mesh*.

pologias acima. Para este tipo de experimentos, eventos foram escalonados para ocorrer exclusivamente em enlaces. Os resultados espelham o que é teoricamente esperado: sem eventos ocorrendo em nodos, um enlace falho ou recentemente recuperado tem sempre um testador, e qualquer detecção ocorre em no máximo um intervalo de testes, independente do tipo de evento. As médias, tanto para eventos de falha como para eventos de recuperação, são iguais à metade do intervalo de testes. Esta regularidade é mantida tanto para redes aleatórias com ambas as conectividades de 3 e $\log n$ como para as redes *mesh* e os hipercubos. Os gráficos para topologias aleatórias são mostrados nas Figuras 7 e 8.

4.2. Experimentos com Particionamento da Rede

Experimentos com particionamento da rede foram executados em grafos aleatórios gerados com a distribuição *Power-Law*. Um componente conexo de uma rede pode sofrer particionamento quando um enlace ponte sofre um evento de falha ou quando um nodo falha. No primeiro caso, a rede fica particionada em dois componentes conexos, contendo cada um deles um dos nodos conectados pelo enlace recentemente falho. Um dos nodos, o próximo testador, detectará o evento em no máximo um intervalo de testes, enquanto o nodo que seria testado caso o enlace se mantivesse sem-falha detectará o evento no intervalo de testes seguinte. Por outro lado, a falha de um nodo, sobretudo daqueles com grande número de enlaces, pode criar vários componentes conexos. Em cada componente conexo, os nodos vizinhos ao nodo falho também detectarão eventos de falha em seus enlaces em um ou dois intervalos de teste, conforme a configuração da rede no momento

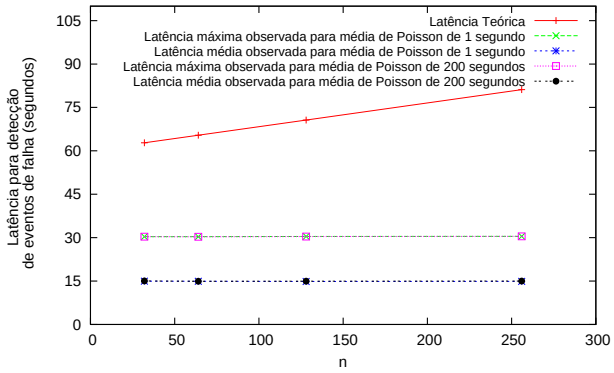

Figura 7. Eventos de falha em enlaces - conectividade de vértices igual a $\log n$ - grafos aleatórios.

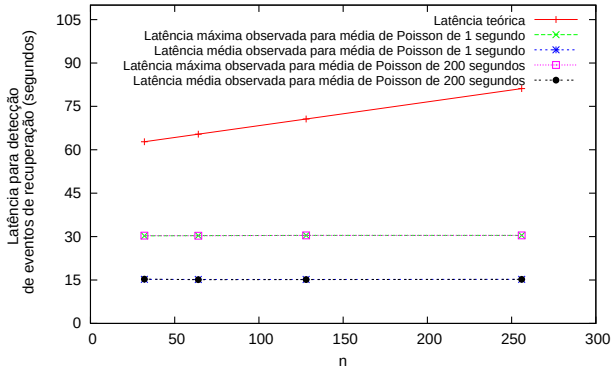

Figura 8. Eventos de recuperação em enlaces - conectividade de vértices igual a $\log n$ - grafos aleatórios.

da falha. Como a falha de um nodo pode ser simulada pela falha em todos os seus enlaces, somente eventos em enlaces foram escalonados neste tipo de experimento. O número de eventos escalonado foi sempre maior que o da conectividade da rede, porém não tão grande que a rede se reduzisse a uma rede trivial.

Foram medidas nestes experimentos apenas as latências de diagnóstico de eventos de falha. Latências de eventos de recuperação não foram medidas, por estarem suficientemente representadas nos experimentos anteriores. Para cada evento, foram medidas a latência de diagnóstico ocorrida no primeiro intervalo de testes, em um dos novos componentes conexos, e a latência de diagnóstico ocorrida para o mesmo evento no segundo intervalo de testes, no outro componente conexo. Ambas as latências foram computadas separadamente para cada evento ocorrido, e as médias das latências do diagnóstico de eventos ocorridos em cada intervalo de testes são mostradas na Figura 9. A média utilizada no processo de Poisson foi de 200 segundos.

Como num grafo aleatório gerado com a distribuição *Power-Law* um grande número de enlaces ponte são enlaces de nodos de borda, em muitos casos a disseminação de mensagens de diagnóstico fica confinada a componentes conexos com apenas um nodo. Isto é compensado pelas disseminações que ocorrem nos componentes conexos formados pelo restante da rede e, na média, as latências exibem o mesmo padrão de comportamento encontrado nos experimentos com falhas em enlace. Desta forma, tanto as

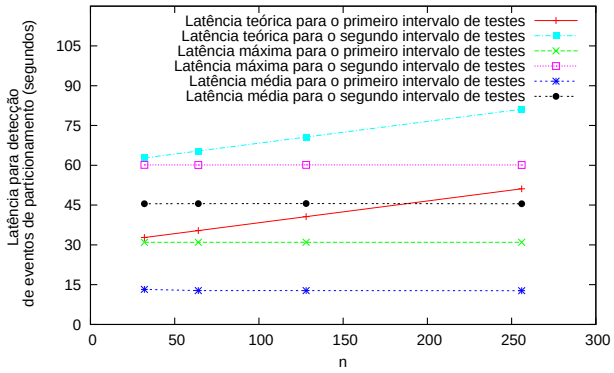

Figura 9. Eventos de falha em enlaces que particionam a rede - grafos *powerlaw*.

latências médias de diagnóstico ocorrido no primeiro como no segundo intervalo de testes se mantém na metado do intervalo de testes correspondente.

5. Conclusão

Este trabalho apresentou a avaliação do algoritmo *Distributed Network Reachability* através de técnicas de simulação de eventos discretos. Os experimentos foram realizados no *cluster* de alto desempenho da Universidade Federal do Paraná. O desempenho do algoritmo foi avaliado em várias topologias aleatórias e regulares, levando em conta múltiplos eventos de falha e recuperação, tanto de nodos como de enlaces. Situações na qual a rede sofre particionamento e na qual ela permanece sempre conectada foram avaliadas. Um conjunto de experimentos foi conduzido de forma a comparar o desempenho dos algoritmos DNR e *ForwardHeartbeat*. O dinamismo da ocorrência de eventos foi dado por processos de Poisson, com médias de 1 segundo e 200 segundos. Para obter um intervalo de confiança de 95% para as médias das latências de diagnóstico, para cada tipo e tamanho de topologia 5000 eventos de falha e recuperação (2500 de cada tipo) foram escalonados e executados em 5 rodadas independentes.

Referências

Bagchi, A. and Hakimi, S. L. (1991). "An Optimal Algorithm for Distributed System-Level Diagnosis," *Proc.* 21^{st} *Int'l Symp. Fault Tolerant Computing*, pp. 214-221, 1991.

Bu, T. and Towsley, D. (2002) "On Distinguishing between Internet Power Law Topology Generators," *IEEE INFOCOM*, vol. 2, pp 638-647, 2002.

Chessa, S. and Santi, P. (2001) "Comparison-Based System-Level Fault Diagnosis in Ad Hoc Networks," *Proc. 20th IEEE Symp. on Reliable Distributed Systems*, 2001.

Chessa, S. and Santi, P. (2002) "Crash Faults Identification in Wireless Sensor Networks," *Computer Communications*, Vol. 25, No. 14, Sept. 2002.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001) *Introduction to Algorithms*, The MIT Press, 2001.

Culler, D. E. and Singh, J. P. (1999) *Parallel Computer Architecture - A Hardware/Software Approach*, Morgan Kaufmann, 1999.

Ding, M., Chen, D., Xing, K. and Cheng, X. (2005) "Localized Fault-Tolerant Event Boundary Detection in Sensor Networks," *Proc. 24th Annual IEEE Conf. Computer and Communication Societies*, 2005.

Duarte Jr., E. P., Mansfield, G., Nanya, T. and Noguchi, S. (1997) "Non-Broadcast Network Fault Monitoring Based on System-Level Diagnosis," *Proc. IEEE/IFIP IM'97*, pp.597-609, San Diego, May 1997.

Duarte Jr., E. P. and Weber, A. (2003) "A Distributed Network Connectivity Algorithm," *Proc. IEEE/ISADS'03*, pp.285-292, Pisa, April 2003.

Elhadef, M., Boukerche, A. and Elkadiki, H. (2004) "An Adaptive Fault Identification Protocol for an Emergency/Rescue-Based Wireless and Mobile Ad Hoc Network," *Proc. Symp. Parallel and Distributed Processing*, pp.1-8, 2007.

Khanna, G., Cheng, M. Y., Varadharajan, P., Bagchi, S., Correia, M. P. and Veríssimo, P. J. (2007) "Automated Rule-Based Diagnosis through a Distributed Monitor System," *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 4, pp. 266-279, October-December 2007.

Lee, M. H. and Choi, Y. H. (2007) "Distributed Diagnosis of Wireless Sensor Networks," *Proc. IEEE Region 10 Conference*, pp.1-4, 2007.

MacDougall, M. H. (1987) *Simulating Computer Systems: Techniques and Tools*, The MIT Press, Cambridge, MA, 1987.

Rangarajan, S., Dahbura, A. T. and Ziegler, E. A. (1995) "A Distributed System-Level Diagnosis Algorithm for Arbitrary Network Topologies," *IEEE Trans. Computers*, Vol.44, pp. 312-333, 1995.

Santi, P. and Blough, D. M. (2002) "An Evaluation of Connectivity in Mobile Wireless Ad Hoc Networks," *Proc. IEEE Int'l. Conf. Dependable Systems and Networks*, pp. 89-102, Washington, 2002.

Stahl, M., Buskens, R. and Bianchini, R. (1992) "Simulation of the Adapt On-Line Diagnosis Algorithm for General Topology Networks," *Proc. IEEE* 11^{th} *Symp. Reliable Distributed Systems*, October 1992.

Subbiah, A. and Blough, D. M. (2004) "Distributed Diagnosis in Dynamic Fault Environments," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 5, pp. 453-467, May 2004.

Weber, A. (2008) *Um Algoritmo de Diagnóstico Distribuído para Redes Particionáveis de Topologia Arbitrária* Tese de Doutorado, UTFPR, 2008. (previsão de defesa - maio 2008)

Weber, A., Duarte Jr., E. P. and Fonseca, K. O. (2006) "An Optimal Test Assignment for Monitoring General Topology Networks," *Proc. 7th IEEE Latin-American Test Workshop*, pp. 131-136, Buenos Aires, 2006