LUIZ FERNANDO LEGORE DO NASCIMENTO

UM ALGORITMO DISTRIBUÍDO E ADAPTATIVO PARA DIAGNÓSTICO DE REDES AD HOC MÓVEIS COM BASE EM INFORMAÇÕES GEOGRÁFICAS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Luiz Carlos Pessoa Albini

CURITIBA

2012

LUIZ FERNANDO LEGORE DO NASCIMENTO

UM ALGORITMO DISTRIBUÍDO E ADAPTATIVO PARA DIAGNÓSTICO DE REDES AD HOC MÓVEIS COM BASE EM INFORMAÇÕES GEOGRÁFICAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Luiz Carlos Pessoa Albini

CURITIBA

2012

AGRADECIMENTOS

Primeiramente à Deus pelo direcionamento do melhor caminho a seguir. Ciente pois, que não seria o mais simples e tranquilo, mas que certamente traria grandes benefícios.

Ao professor Luiz Carlos Pessoa Albini, que aceitou orientar-me. Ressaltando a humildade, apoio, dedicação à docência e principalmente a paciência com este acadêmico que um dia resolveu galgar uma nova área de trabalho. Dificuldades foram e sempre serão muitas, contudo são necessárias para que haja o amadurecimento, tanto como pessoa, quanto como pesquisador.

A minha mãe, a qual sinto imensamente falta. Àquela na qual tanto me incentivou e que durante seus últimos dias de vida mostrava-se forte nas despedidas, mesmo estando em um leito hospitalar. Dedico a ela a realização deste trabalho.

A minha esposa Márcia e as minhas filhas, Fernanda, Jéssica Luiza e Yasmin. Mulheres que com muita paciência e compreensão entenderam a ausência paterna, além dos aborrecimentos ocorridos durante dura jornada de estudos.

Aos colegas de trabalho, amigos docentes que sempre apoiaram para a obtenção desse título. Os incentivos, as viagens rumo aos estudos e as trocas de experiências foram me imprescindíveis. Grato também aos amigos técnicos, que além do apoio, assumiram com grande dedicação as infrequentes tarefas de um ambiente de TI.

Agradeço e orgulho-me de ter sido aluno de grandes professores durante o mestrado. Ensinamentos aqui adquiridos que certamente serão repassados.

Aos nobres colegas que conheci durante esta jornada, integrantes do grupo de pesquisa NR2. Amizade dessas pessoas que guardarei com muito carinho e que buscarei sempre

manter contato.

A todos os amigos que me deram o prazer de viver a vida com alegria. Amigos os quais não se resume em uma simples lista de nomes, mas que encontram-se muito bem guardadas em um forte sentimento.

Muito Obrigado!

RESUMO

Algoritmos de diagnóstico de falhas em nível de sistema são comumente utilizados como uma estratégia de tolerância à falhas. Neles, a partir de uma série de testes, pode-se determinar quais unidades estão com falha e quais estão sem-falha. Os primeiros modelos de diagnóstico de falhas que surgiram eram do tipo centralizado. Nesses modelos, considera-se que uma unidade central é a única responsável por diagnosticar o estado de todas as unidades do sistema. Posteriormente, surgiram os modelos distribuídos, nos quais o diagnóstico é atribuído a algumas ou todas as unidades do sistema. Todos estes modelos foram desenvolvidos para redes estáticas e cabeadas, não sendo facilmente adaptáveis à ambientes móveis. A identificação de unidades com falha em uma rede Ad Hoc móvel é uma tarefa muito difícil. Nestas redes, as unidades se utilizam de comunicações sem fio e podem se mover livremente e de forma imprevisível. Nesse trabalho é apresentado o primeiro algoritmo distribuído e adaptativo para diagnóstico em nível de sistema baseado no modelo PMC para redes Ad Hoc, que não impõe restrições ao movimento dos nós durante todo o diagnóstico. Todos os nós podem testar, serem testados e diagnosticar os demais nós. Além disso, os nós podem ser classificados como falho, sem falha e suspeitos. Um nó é classificado como suspeito quando este deixa de responder a solicitação de teste por sair do raio de alcance das transmissões de radio do seu testador. Como a mobilidade dos nós implica em variação de distâncias entre os nós, testador e testado, as solicitações de testes e suas respostas nem sempre ocorrem de forma satisfatória. Para isso, o algoritmo utiliza-se de coordenadas geográficas para monitorar a mobilidade dos nós de forma que seja possível diferenciar falhas reais, de falhas ocorridas devido a mobilidade.

Palavras-chave: Diagnóstico de falhas, PMC, Ad Hoc, Coordenadas geográficas.

ABSTRACT

System-level fault diagnosis algorithms are commonly used as a strategy for fault tolerance, in which, from a series of tests, it is possible to determine faulty units. The first fault diagnosis models were centralized, considering that a central unit is the only responsible for diagnose the status of all system units. Subsequently, there were distributed models, in which diagnosis is assigned to some or all units of the system. All these models were developed for static and wired networks, not being easily adaptable to mobile environments. The identification of faulty units in a mobile Ad Hoc network is a very difficult task because units can move freely and unpredictably. In this paper we present the first distributed algorithm for adaptive system-level diagnosis based on the PMC model for Ad Hoc networks, which imposes no restrictions on the movement of the units throughout the diagnosis. All nodes can test, be tested and diagnose other nodes. Furthermore, nodes can be classified as faulty, fault-free and suspect. A node is considered suspect when it moves away from the tester transmission range, thus not responding to tests anymore. As node mobility implies on distances variation between nodes, tester and tested, tests requests and their answers do not always occur in a satisfactory manner. Hence, geographic coordinates are considered by the algorithm in order to keep track of the nodes mobility, so that it is always possible to distinguish real faults from mobility-generated faults.

Keywords: Fault diagnosis, PMC, Ad Hoc, geographical coordinates.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BFS	Breadth-First Search (Busca em Largura - Grafos)
BGM	Barsi, Grandoni e Maestrini
DSDP	Distributed Service Directory Protocol (Protocolo de Serviços de Diretório Distribuído
GPS	Global Position System (Sistema de Posicionamento Geográfico)
MANET	Mobile Ad-Hoc Network (Rede Ad-Hoc Móvel)
MM	Maeng e Malek
NS	Network Simulator (Simulador de Redes)
PMC	Preparata, Metze e Chien
RSSF	Redes de Sensores Sem Fio
WSN	Wireless Sensor Network (Rede de Sensores sem fio)

CAPÍTULO 1

INTRODUÇÃO

Uma rede de computadores pode ser constituída por diversos equipamentos computacionais, tanto fixos quanto portáteis, no entanto, equipamentos portáteis conectados à rede Internet através de tecnologia sem fio são, segundo [26], um caminho sem retrocesso. A mobilidade desses equipamentos é fator preponderante, uma vez que a facilidade de acesso à informação está presente em qualquer localidade, sem que haja a necessidade de algum fio conectado a esses dispositivos. Embora essa facilidade ofereça conforto e praticidade, essas no entanto requerem mais atenção quanto aos itens de segurança, gerenciamento e conectividade. Dentre os pontos citados, observa-se que a ocorrência de determinados eventos podem comprometer toda a rede. Caso uma rede sem fio, utilize vários equipamentos trabalhando como roteadores e mantendo entre si uma via de comunicação entre diferentes pontos, a ocorrência de falha em um desses roteadores poderá interromper uma comunicação existente.

Ao referir-se a conectividade, uma rede é dita conexa se todos os dispositivos podem se comunicar, independentemente desses dispositivos serem fixos ou móveis, [6]. Caso uma rede seja composta por somente duas unidades, sendo uma delas fixa e a outra móvel, a conectividade desses dispositivos fica condicionada a uma determinada área de cobertura fixa. Nesse caso, o dispositivo não pode se mover para longe da unidade fixa sem que haja perda de comunicação. Ainda nesse exemplo, a rede pode ser centralizada nesse ponto fixo, o qual é o responsável por manter e gerenciar todos os dispositivos móveis conectados a ela.

A mobilidade pode ser maior se cada dispositivo puder enviar ou receber informações sem que haja a existência desse ponto fixo central. Em redes sem fio Ad Hoc os equipamentos portáteis podem se comunicar permitindo que uma determinada informação seja repassada por vários deles até chegar ao seu destinatário. Nessas redes, não há uma co-

municação centralizada [10, 23]. Esse trabalho aborda os termos equipamentos, nós e unidades como sinônimos. A comunicação entre os nós móveis é feita por sistemas computacionais distribuídos e autônomos que se encarregam de manter a rede conexa. Assim, nós pertencentes a uma rede sem fio denominada Ad Hoc ou redes multi-saltos podem se mover livremente, desde que cada equipamento (nó) possa ter pelo menos um outro dentro da sua área de cobertura.

Para exemplificar a importância do diagnóstico em redes Ad Hoc, considera-se a existência de uma comunicação previamente estabelecida entre dois nós distintos u e v os quais não vizinhos entre si. Observando a representação matemática, u, x_1, x_2, x_3,...,x_k, v, observa-se que vários outros nós x_i são utilizados como elos de ligação. Essa comunicação no entanto, poderá deixar de existir caso um dos nós pertencentes a esse caminho torne-se falho. Diagnosticar rapidamente e corretamente o estado desses nós torna-se imprescindível. Nesse caso, e a partir de um rápido diagnóstico, pode-se invocar um outro procedimento o qual crie um caminho alternativo entre os nós u e v, garantido portanto a continuidade dessa comunicação.

Em outro exemplo, uma rede de sensoriamento, onde um nó com falha poderá emitir sinais errôneos de monitoramento, sinalizando eventos inexistentes, como por exemplo um princípio de incêndio, tempestades, entre outros. A partir desses exemplos citados, é possível compreender a importância de se realizar o diagnóstico de falhas, principalmente em sistemas de natureza crítica.

A mobilidade dos nós implica em alguns casos em perda de comunicação. Um nó ao mover-se para longe de outro nó poderá ser incapaz de responder aos testes solicitados. Todos os trabalhos aqui apresentados classificam um nó como falho quando esse deixa de responder aos testes após o tempo limite de resposta ter ocorrido. Esse trabalho, busca monitorar o deslocamento dos nós afim de diferenciar nós em movimento de falhas tradicionais de *hardware* e de *software*. Um monitoramento utilizando coordenadas geográficas permite que seja investigado se a ocorrência de uma falha se deu pela quebra do equipamento ou devido a migração para fora da área de cobertura de sinal. O algoritmo DAINGeo - Algoritmo Distribuído e Adaptativo com base em Informações Geográficas, o

qual é apresentado nesse trabalho, classifica os nós em três diferentes conjuntos: Falhos, sem falha e suspeitos. Ressaltando a classificação de nós suspeitos em função da sua mobilidade.

Muitos estudos referentes ao diagnóstico de falhas são apresentados na literatura. Dentre eles, destaca-se o modelo PMC proposto por Preparata, Metze e Chien, [22]. Esse foi o primeiro modelo proposto, tendo introduzido o conceito de diagnóstico em nível de sistema. Esse é provavelmente o modelo de diagnóstico mais bem estudado. No modelo PMC original, assume-se a existência de um supervisor central que determina quais os testes a serem realizados, coleta os resultados e realiza o diagnóstico com base nas respostas dos testes recebidos.

Posterior ao modelo PMC foram apresentados outros modelos de diagnóstico, nos quais os algoritmos utilizados permitem que cada nó possa decidir qual será o próximo teste a ser realizado tendo como base os resultados de testes anteriores. Algoritmos dessa natureza são classificados como adaptativos [11, 16]. Além disso, surgiram os algoritmos que consideram que o diagnóstico deve ser realizado de forma distribuída, ou seja, deixa portando de ser centralizado, como no modelo PMC original, e passa a ser feito através dos próprios nós da rede [2, 14, 15, 16].

Nesse trabalho é apresentado o primeiro algoritmo distribuído e adaptativo para diagnóstico em nível de sistema baseado no modelo PMC para redes Ad Hoc, que não impõe restrições ao movimento das unidades durante todo o diagnóstico. A detecção de falhas do algoritmo proposto baseia-se em distâncias geográficas do testador e das unidades testadas. Para análise do algoritmo, optou-se por fazer de forma matemática. Além disso, são apresentados comparativos entre o algoritmo DAINGeo com o algoritmo de Madhu Chouhan [7] e Chessa e Santi [6].

1.1 Objetivos

Objetiva-se:

- o desenvolvimento de um algoritmo distribuído de diagnóstico, baseado no modelo

PMC, que não apresente restrições quanto a mobilidade das unidades durante todo o processo de diagnóstico;

- provar de forma teórica que o algoritmo apresentado poderá diagnosticar de forma correta a rede mesmo que esta possua falhas dinâmicas;

1.2 Organização do trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta os conceitos sobre os diferentes tipos de falhas e sua classificação. No Capítulo 3, são apresentados os diversos modelos de diagnósticos pertinentes ao trabalho. No Capítulo 4 são apresentados os trabalhos publicados para redes Ad Hoc. O Capítulo 5 é apresentado o algoritmo proposto e seu funcionamento, sendo a avaliação teórica desse algoritmo vista no Capítulo 6. Por fim, o Capítulo 7 apresenta as conclusões e sugestões de trabalhos futuros.

CAPÍTULO 2

TIPOS DE FALHAS

Um equipamento deixa de funcionar quando o *hardware*, o *software*, ou ambos, cessam suas atividades ou apresentam falhas parciais que produzem respostas incorretas. No caso de falhas ocorridas em uma rede sem fio, falhas de *hardware* podem ser exemplificadas através do esgotamento de sua fonte de energia (bateria), ou a quebra da antena de um equipamento portátil. Em ambos os exemplos, tais falhas tornam a comunicação com qualquer outro equipamento da rede impossível. Falhas referentes ao *software*, podem a exemplo, apresentar travamento ou sobrecarga de processamento. Desta forma, o *hardware* ainda poderá se manter em funcionamento, porém apresentando processamento irregular.

Além das falhas de *hardware* e *software*, falhas de conexão poderão ocorrer. Nestas, a comunicação entre os equipamentos participantes deixa de existir, como exemplo pode-se citar elevados níveis de interfência capaz de interromper uma comunicação previamente estabelecida. Embora falhas dessa natureza possam ocorrer, estas são consideradas como falhas de *hardware* e serão tratadas como tal neste trabalho.

A categorização dos diversos tipos de falhas é apresentada a seguir e se baseia em: (i) no tempo de duração da falha; (ii) no comportamento após a ocorrência da falha; e (iii) na ocorrência de falha durante a sessão de diagnóstico.

1. Quanto ao tempo de duração da falha, elas podem ser de três tipos [7]:

    - Transitória: Falha de duração limitada, causada por um mau funcionamento temporário ou em consequência de alguma interferência externa.
    - Intermitente: Trata-se de uma falha transitória, porém ocorre repentinamente e sempre por uma duração muito curta.
    - Permanente: Falhas dessa natureza podem ocorrer devido a uma quebra física

do equipamento ou pela ausência de energia necessária para manter o equipamento funcionando. Assim, essas falhas demoram a serem reparadas e quase sempre são reparadas por algum administrador externo.

Falhas de comportamento recebem ainda uma outra classificação. Segundo Jalote em [13], essas são classificadas em quatro tipos: *Permanente, Omissão, Temporal e Bizantina.* Segundo o autor, há uma relação de inclusão entre essas falhas, formando uma hierarquia. Na Figura 2.1, é apresentada essa hierarquia, onde as falhas do tipo *Permanente* são consideradas as mais simples e as mais restritivas, e *Bizantina* as menos restritivas.

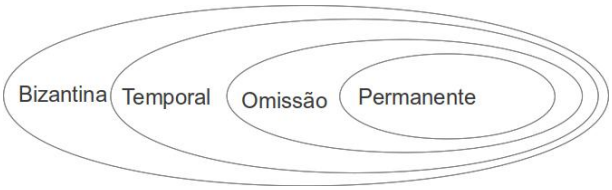

Figura 2.1: Classificação quanto aos tipos de falhas - [13]

- Falhas do tipo permanente (*crash*), são consideradas *hard fault*, ou seja, o dispositivo deixa de funcionar por completo e portanto não produz uma resposta incorreta quando testado por algum outro dispositivo.
- Falhas referentes à omissão (*Omission Fault*), referem-se a obtenção de respostas a algumas solicitações feitas e omissão a outras.
- A falha temporal ou *Timing Fault* refere-se ao retardamento ou adiantamento das respostas.
- A ocorrência de falhas arbitrárias, são também denominadas de falhas bizantinas ou *Byzantine Fault.* Nesse tipo de falha os nós não param de funcionar, eles ainda podem continuar respondendo, porém processam requisições de forma incorreta e inconsistente, i.e. respondem de maneira imprevisível aos testes solicitados.

2. A classificação quanto a ocorrência de falha durante a sessão de diagnóstico pode ser de dois tipos [7]:

- Falha estática: Nós que sofreram algum tipo de falha *hardware* ou *software* ou tiveram sua falha reparada não podem ter seu seu estado alterado durante uma sessão de diagnóstico.
- Falha dinâmica: São permitidos aos nós a mudança do seu estado durante a seção de diagnóstico. Portanto, nós sem-falha podem tornarem-se falhos e nós com falha, podem ser reparados mudando em seguida seu estado para sem-falha.

2.1 Conclusão

Os diversos tipos de falhas foram apresentadas nesse capítulo como o objetivo de familiarizar o leitor sobre suas diversas classificações. Pode-se observar que algumas dessas falhas dificultam o procedimento para a sua descoberta. A exemplo disso, pode-se citar a falha intermitente a qual pode não seguir um intervalo de tempo padronizado para sua ocorrência. Outro tipo de falha de difícil classificação é a falha bizantina, a qual apresenta características de todas as demais falhas, sendo esse tipo de falha costumeiramente observado em processos maliciosos. Nesse trabalho, com exceção das falhas bizantinas todas as demais falhas poderão ser observadas, contudo as falhas dinâmicas, estáticas e permanentes serão as mais discutidas.

CAPÍTULO 3

MODELOS DE DIAGNÓSTICO

Modelos de diagnóstico tem por objetivo a obtenção de informações sobre o estado das unidades computacionais que compõem um sistema. Denota-se como modelos de diagnóstico um conjunto de condições e métodos, que de forma estruturada possam diagnosticar o estado desses ambientes computacionais [16, 27]. Esses modelos são utilizados de acordo com o ambiente a ser investigado.

O primeiro modelo a utilizar o conceito de diagnóstico em nível de sistema foi proposto por Preparata, Metze e Chien [22] (Modelo PMC). Além desse, outra abordagem surgiu de forma alternativa ao PMC. O primeiro trabalho proposto o qual baseava-se em um modelo de diagnóstico de falhas o qual utilizava a comparação de resultados obtidos por dois nós distintos da rede, foi proposto por Malek [19] e por Chwa e Hakimi [8].

Nesse trabalho, é descrito o modelo PMC além dos seguintes modelos baseados em comparação: o modelo MM [19] e o modelo de Comparações Generalizado [24].

3.1 Modelo PMC

No modelo PMC [22], um sistema é definido como um conjunto de unidades indivisíveis. Afim de diagnosticar o estado das unidades de um sistema, é necessário que os nós (unidades) pertencentes ao sistema possuam a capacidade de executar testes e reportar resultados. O modelo não considera a existência de falhas de enlaces. Um nó somente é considerado falho quando ocorre a "morte" dessa unidade. Considera-se "morte" ou *hard fault*, quando um nó não mais responde a qualquer estímulo.

No modelo PMC, o sistema é representado através de um grafo, no qual os nós são os vértices e as arestas são as ligações lógicas que interligam esses nós. A Figura 3.1 apresenta um grafo representando um sistema com 4 nós.

Para a realização dos testes, um outro grafo, denominado Grafo de Testes é apresentado

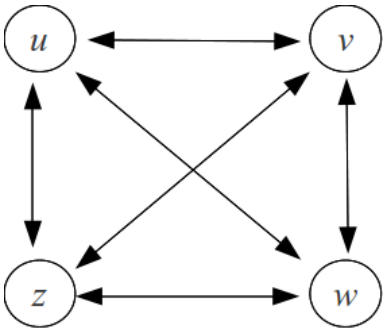

Figura 3.1: Grafo do Sistema.

(Figura 3.2). Este grafo representa os testes a serem feitos. O sistema é modelado através de um grafo $G = (V,L)$, onde V são os vértices representando os nós do sistema e L as arestas. As arestas são testes direcionados, $u \rightarrow v$, ou seja, o nó u testa o nó v.

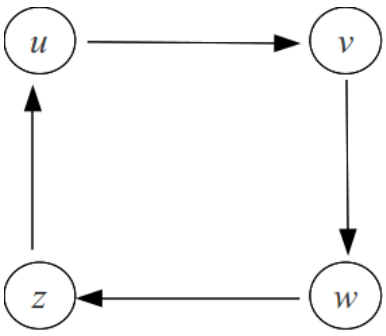

Figura 3.2: Grafo direcionado de teste.

Os nós devem ser capazes de testar todos os demais nós do sistema e determinar se eles estão ou não funcionando corretamente, além disso, nenhum nó testa a si mesmo. A saída de um teste é simplesmente 1 (falho) ou 0 (sem-falha). Um teste completo sempre retorna o estado correto do nó testado, isto é, o resultado de um teste completo sempre reflete a condição atual do nó testado. Os testes realizados são fixados previamente e o conjunto destes testes determina se um sistema pode ou não ser diagnosticado.

Os resultados de todos os testes são submetidos a um observador central, encarregado de fazer o diagnóstico. O observador central é uma entidade externa ao sistema, deve ser confiável e não pode falhar em hipótese alguma. Esse modelo assume ainda que, durante o processo de diagnóstico os nós não podem mudar seu estado.

Uma saída é associada a cada teste que o nó u executa sobre o nó v. Portanto $a_{u,v}$ é a saída do respectivo teste. Considerando o nó u como unidade testadora e sem-falha, o

teste $a_{u,v}$ realizado sobre o nó v será 0 se v estiver sem-falha. Caso v esteja com falha, $a_{u,v}$=1. Nos casos em que u é falho, o resultado do teste não é confiável e $a_{u,v}$ pode assumir qualquer valor, independente do estado de v. Esse conjunto de resultados de todos os testes que foram executados é apontado como síndrome do sistema [13].

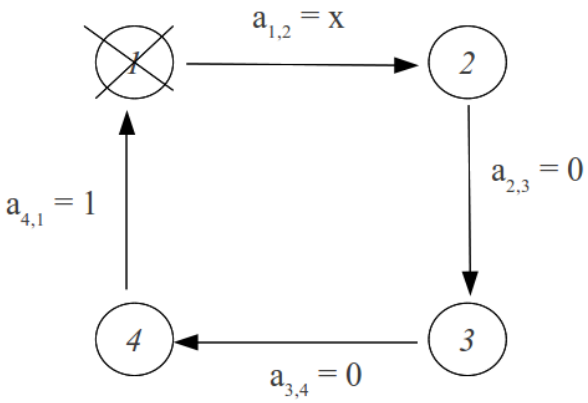

Figura 3.3: Síndrome de um sistema com 4 nós.

Para exemplificar, considera-se um modelo do sistema com 4 nós, cujos testes são atribuídos conforme mostrado na Figura 3.3. Os testes são realizados tendo como início o nó 1, o qual testa o nó 2, o nó 2 testa o nó 3, e assim por diante, até que o último testa o primeiro. A síndrome desse sistema é o seguinte vetor de 4 bits:

$$(a_{1,2}, a_{2,3}; a_{3,4}; a_{4,1})$$

Neste exemplo, observa-se que o nó 1 está falho, logo a síndrome do sistema é:

$$(x; 0; 0; 1)$$

Observa-se que o nó 2 identifica o nó 3 como sem-falha, $a_{2,3} = 0$, o nó 3 identifica o nó 4 como sem-falha, $a_{3,4} = 0$, o nó 4 identifica o nó 1 como falho, $a_{4,1} = 1$. Como o nó 1 está falho, o valor de $a_{1,2}$ pode ser 0 ou 1, nesse caso um valor x é atribuído representando qualquer resultado possível.

Denominada como regra de invalidação, essa regra é apresentada na Tabela 3.1.

Considerando σ como sendo a diagnosticabilidade do sistema, um sistema é dito ser σ-diagnosticável quando é possível identificar todos os nós com falhas através da análise da síndrome.

Testador	Nó Testado	Diagnóstico
sem-falha	sem-falha	(0) sem-falha
sem-falha	falha	(1) falha
falha	sem-falha	(x)resultado qualquer
falha	falha	(x) resultado qualquer

Tabela 3.1: Modelo PMC - Regra de Invalidação.

O trabalho de Preparata, Metze e Chien [22] demonstra duas condições necessárias para que um sistema seja σ-diagnosticável:

- (c1) a diagnosticabilidade do sistema é dada por: $\sigma \leq \frac{(n-1)}{2}$, onde n é o número de nós do sistema;
- (c2) a unidade deve ser testada por pelo menos σ outras unidades.

Observando as condições apresentadas, um sistema pode ser então corretamente diagnosticado se a quantidade de nós com falha não exceder a σ. Em Hakimi e Amin os autores demonstraram que as condições acima, (c1) e (c2) são necessárias e suficientes para que um sistema seja σ-diagnosticável, desde que não haja testes mútuos entre as unidades do sistema [12].

Muitos algoritmos, baseados no modelo PMC, foram propostos posteriormente. A exemplo, os algoritmos adaptativos, [11, 16], nos quais as unidades decidem os próximos testes a serem realizados baseando-se em resultados de testes anteriores [11]. Os algoritmos de diagnósticos distribuídos [2, 14, 15, 16] permitem que o diagnóstico seja feito pelas próprias unidades. Assim, observa-se que para essa abordagem a figura do observador central deixa de existir. Neste caso, cada nó aceita informações de testes de outros nós do sistema que tenham sido previamente testados como sem-falha e realiza o diagnóstico do sistema.

3.2 Modelos de Diagnóstico Baseado em Comparações

Os primeiros modelos de diagnóstico baseado em comparações foram proposto por Malek [20], e Chwa e Hakimi [8, 19, 20]. Nesses modelos, os testes realizados no sistema são

executados através da duplicação de uma tarefa de teste designada à dois nós distintos. Os resultados dos testes são então enviados a uma entidade externa e confiável, denominada observador central. A esse nó confiável é atribuída a responsabilidade de diagnóstico do sistema, que é realizado através da comparação dos resultados apresentados pelos dois nós distintos.

Como esse modelo assume a existência de um observador central, confiável, esse não poderá falhar. Além disso, esse observador central coleta e mantém as saídas de todas as unidades testadas.

Ao comparar as saídas geradas por dois nós testados, elas podem apresentar resultados iguais, ou seja, ambas as saída podem resultar em nós sem-falha. Caso um desses nós esteja com falha, o resultado da comparação será uma divergência. Dessa forma, o observador central não conseguirá afirmar qual dos nós está falho, ou seja, é possível afirmar que existe pelo menos um falho, mas não qual dos dois é o nó com falha.

Para tanto, o modelo impõe as seguintes asserções:

- as saídas produzidas por dois nós sem-falha para uma mesma tarefa sempre são idênticas;
- uma saída produzida por um nó falho tem que ser diferente de qualquer outra saída produzida para essa mesma tarefa, tanto por nós sem-falha, como por outros nós falhos.

Ao respeitar essas asserções, a única forma de duas saídas geradas por nós distintos em uma mesma tarefa serem idênticas é quando os dois nós estão sem-falha.

Unidade 1	Unidade 2	Resultado da comparação
sem-falha	sem-falha	0 (igualdade)
sem-falha	com falha	1 (diferença)
com falha	sem-falha	1 (diferença)
com falha	com falha	1 (diferença)

Tabela 3.2: Possíveis resultados do modelo baseado em comparação.

A Tabela 3.2 apresenta os possíveis resultados das saídas produzidas por unidades com falha e sem-falha.

O sistema é modelado através de um grafo $G = (V,L)$, onde V é um conjunto de n vértices, sendo que cada um deles corresponde a uma unidade do sistema. O conjunto de arestas L, corresponde as ligações lógicas entre as unidades do sistema. Além disso, G é um grafo conexo, ou seja, há pelo menos um caminho entre quaisquer dois vértices.

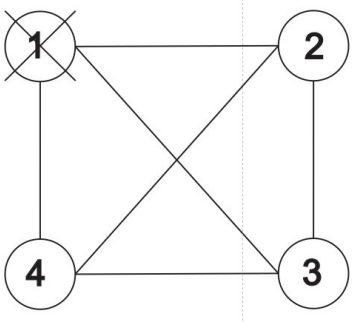

Figura 3.4: Grafo representando um sistema com quatro unidades.

Na Figura 3.4 é apresentado um grafo com 4 nós. Para esta figura observa-se a falha do nó 1. Os resultados de todas as comparações são em seguida apresentadas na Tabela 3.3 em relação ao grafo apresentado na Figura 3.4.

ID da Unidade 1	ID da Unidade 2	Resultado da comparação
1	2	1 (diferença)
2	3	0 (igualdade)
3	4	0 (igualdade)
1	3	1 (diferença)
1	4	1 (diferença)
2	4	0 (igualdade)

Tabela 3.3: Todos os resultados obtidos pelo nós comparadores.

3.2.1 Modelo MM

Maeng e Malek [19], apresentam uma melhoria para o modelo baseado em comparações, conhecido como o modelo MM. Esse modelo foi proposto para sistemas compostos por múltiplos processadores. Nesse modelo de diagnóstico as comparações são feitas nos próprios nós do sistema. A existência do observador central como um nó externo e confiável também é empregada neste modelo sendo o nó responsável pelo diagnóstico. No entanto, foi eximido do observador a tarefa de comparar as saídas geradas pelos nós tes-

tadores. A tarefa de comparação é agora atribuída aos próprios nós. Esses nós também poderão estar com falha.

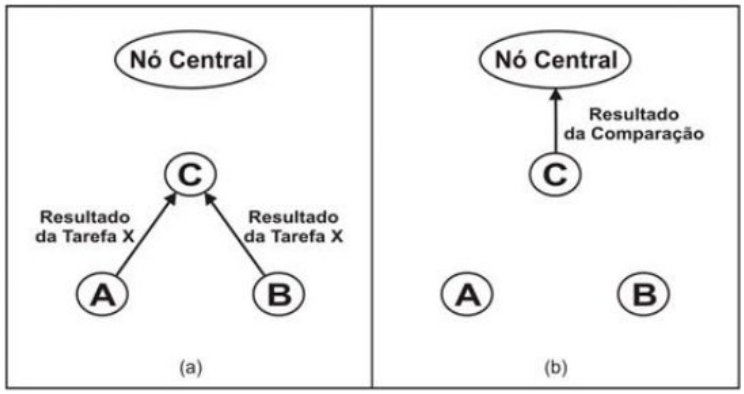

Figura 3.5: Modelo baseado em comparações

No exemplo da Figura 3.5, um dado nó testador C envia para os nós A e B uma mesma tarefa e coleta os resultados produzidos (a). A seguir, o nó C repassa o resultado da comparação ao nó central que realiza o diagnóstico (b).

Se o resultado de uma comparação constatou que as duas saídas eram idênticas e o comparador está sem-falha, o observador central sabe que os dois nós que realizaram a tarefa estão sem-falha. Mas, se o resultado da comparação constatou que as duas saídas eram diferentes e o comparador está sem-falha, o observador central não consegue determinar qual dos dois nós está falho. Além disso, como o comparador agora também é um nó do sistema, ele próprio pode estar falho. Nesse caso, nada é possível constatar sobre as comparações feitas por esse nó.

Este modelo assume algumas asserções como:

- Todas as falhas de qualquer nó são permanentes;
- Todo nó falho sempre gera resultados incorretos para cada tarefa de entrada, isto é, a comparação das saídas de tarefas produzidas por um nó falho e qualquer outro nó (falho ou sem-falha) sempre resulta em diferença;
- A comparação feita por um nó falho não é confiável;
- Existe um limite superior σ, sobre a quantidade de nós que podem estar falhos no sistema, para que o diagnóstico desse sistema seja possível.

Todos os possíveis resultados de uma comparação são apresentados na Tabela 3.4.

Comparador	Unidade 1	Unidade 2	Resultado da Comparação
sem-falha	sem-falha	sem-falha	0 (igualdade)
sem-falha	sem-falha	falha	1 (diferença)
sem-falha	falha	sem-falha	1 (diferença)
sem-falha	falha	falha	1 (diferença)
falha	sem-falha	sem-falha	0 ou 1
falha	falha	sem-falha	0 ou 1
falha	falha	sem-falha	0 ou 1
falha	falha	falha	0 ou 1

Tabela 3.4: Possíveis resultados de comparações para o modelo MM.

3.2.2 Modelo MM Generalizado

Neste modelo, proposto por Sengupta e Dahbura [24], o nó que vai comparar as saídas produzidas para uma mesma tarefa, pode ser um dos dois nós que realizam a tarefa de testador. Por exemplo, é possível ter uma comparação *(u, v)u* ou *(u, v)v*. A Figura 3.6 ilustra o modelo MM Generalizado com os nós u e v testando os demais nós do sistema. Essa Figura ainda mostra que um dos nós u ou v podem comparar os resultados obtidos e enviar ao observador central para a realização do diagnóstico.

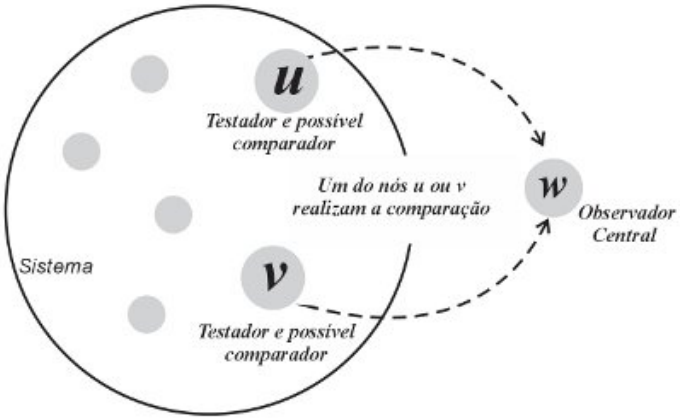

Figura 3.6: Modelo MM Generalizado

O resultado dessa comparação é enviado ao observador central para que este realize o diagnóstico. O observador ainda é o único nó responsável pelo diagnóstico, mesmo que as comparações sejam distribuídas. Todo o restante do modelo de comparações generalizado é similar ao modelo MM.

3.3 Conclusão

Neste capítulo, foram apresentados alguns dos principais modelos de diagnóstico de falhas. Nesta breve conclusão, ressalta-se o modelo PMC original, no qual é apresentado a figura do observador central. Este é o único nó do sistema responsável por diagnosticar todos os nós da rede, tendo este recebido todas as respostas de testes solicitadas. Diferentemente desse modelo, nos modelos baseados em comparação, os testes realizados através da duplicação de uma mesma tarefa de teste é designada à dois nós distintos do sistema.

CAPÍTULO 4

DIAGNÓSTICO EM REDES AD HOC

Nesta seção são apresentados os modelos de diagnóstico de falhas para redes Ad Hoc [4, 5, 6, 7, 17, 18]. O primeiro modelo apresentado, refere-se ao sensoriamento de níveis de fumaça. Nesse trabalho os autores utilizam como base o modelo de diagnóstico PMC para a realização do diagnóstico em nós estacionários e dispostos em regiões geográficas. Os demais trabalhos apresentados são baseados nos modelos de comparação. Nesses trabalhos são permitidos aos nós se moverem, porém há pelo menos alguma restrição de mobilidade desses nós durante todo o processo de diagnóstico.

4.1 Diagnóstico em Redes de Sensores Através do modelo PMC

Em [25] os autores empregam o modelo PMC buscando uma estratégia para avaliação sobre a diagnosticabilidade em nível de sistema em Redes de Sensores Sem Fio (RSSF). O cenário é descrito como uma rede de sensoriamento que monitora a ocorrência de sinais de fumaça. No caso desses sinais ultrapassarem determinados níveis, um alarme é disparado. Afim de garantir que um alarme não seja disparado de forma errônea (falso positivo), é empregado o diagnóstico de falhas para assegurar a veracidade desse evento. Nesse modelo, é considerado que uma RSSF é composta por sensores implantados em uma determinada região (R), com distribuição uniforme. Os resultados obtidos através dos testes são utilizados como entrada para um algoritmo de diagnóstico que identifica os sensores falhos e, assim, confirma ou rejeita o alarme reportado.

O sistema é modelado como um grafo $G=(V,L)$, em que todos os sensores são vértices e pertencem ao conjunto V. As conexões lógicas entre os pares de sensores são representadas pelas arestas do grafo, o conjunto L. A comunicação entre os nós é feita por radiofrequência onde em uma determinada área de sensoriamento também denominada como região geográfica R, estão dispostos diversos nós sensores. Um conjunto VR é de-

notado por nós sensores V, os quais são pertencentes a uma determinada região R. Essa região R é dividida em 4 quadrantes.

Tendo sido observado um evento, isto é, um alerta de incêndio, dois quadrantes vizinhos a essa região na qual ocorreu o evento, participam do processo de diagnóstico, os quadrantes denominados sucessor e antecessor.

O conjunto de nós que conseguiram detectar esse evento, passam a integrar o conjunto de nós sensores T, onde $T \subset VR$. A fim de impedir que haja um alerta errôneo, essa informação precisa portanto ser validada por outros nós dessa rede. Duas diferentes estratégias de teste são apresentadas com o objetivo de gerar, de forma eficiente, assinalamentos de testes com diagnosticabilidade suficiente para validar um conjunto de alarmes. Essas estratégias garantem ainda que, para que um alarme possa ser gerado por σ sensores, um grafo de testes no mínimo σ-diagnosticável deve ser considerado.

A primeira estratégia, consiste em testes sem reciprocidade, i.e. os nós $T \subset VR$ solicitam para que outros nós e de outra região realizem testes sobre eles mesmos. Nessa estratégia, o nós sensores presentes em um mesmo quadrante não executam testes entre si. Os nós do conjunto T solicitam que alguns nós do quadrante antecessor realizem testes sobre eles, dessa forma, os testes ocorrem no sentido anti-horário. Para a realização desses testes deve ser considerado as condições (c1) e (c2), descritas na seção 3.1

A segunda estratégia aborda os testes com reciprocidade, ou seja, trata-se de uma estratégia simples, que consiste em efetuar todos os testes possíveis entre os sensores da região R. Nesse caso, para que o sistema seja σ-diagnosticável, uma terceira condição deverá existir, nos casos de testes mútuos.

- (c3) Seja G um grafo direcionado representando um sistema com n unidades, e κ(G) representa a conectividade do grafo G, ou seja, o número mínimo de vértices cuja remoção desconecta o grafo G; se κ(G)$\geq \sigma$, então o sistema é dito ser σ-diagnosticável, [9].

A estratégia consiste em tornar todos os testes possíveis entre os nós sensores da região R. Para isso, são utilizadas algumas conclusões apresentadas por Penrose [21], cujo

trabalho aborda a relação entre a conectividade alcançada por grafos e seus graus mínimos. Penrose prova que quando n tende ao infinito, a probabilidade de que o grafo seja κ-conexo tende a 1, e quando a probabilidade de qualquer vértice do grafo apresentar grau mínimo igual a κ esse também tende a 1. Os resultados apresentados por Penrose sugerem que se a densidade da rede for suficientemente alta então a condição (c3) é alcançada.

Ao final, todos os resultados dos testes realizados são recolhidos por um nó denominado coletor, que usando um algoritmo adequado decodifica os resultados e realiza o diagnóstico.

O modelo [25] segue o modelo PMC original [22]. Nesse caso, observa-se a existência da figura do observador central (coletor ou *sink*) que realiza o diagnóstico, assumindo ainda que testes feitos por um nó sensor com falha produz respostas arbitrárias. A Figura 4.1 ilustra um exemplo de organização para redes de sensores no qual pode ser observado os nós sensores em uma região de sensoriamento e o *sink* ou coletor, como sendo o observador central.

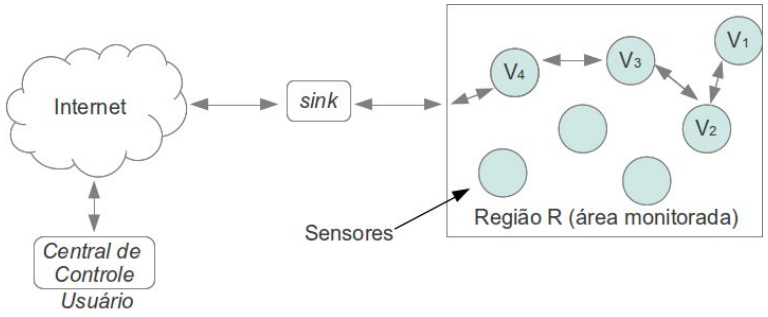

Figura 4.1: Exemplo de organização de redes de sensores - [1].

Os resultados das simulações apresentadas pelos autores desse trabalho, mostram que as duas estratégias alcançaram os níveis necessários de diagnosticabilidade, mesmo que à custa de uma crescente interferência de sinais de radiofrequência.

Embora seja empregado o modelo PMC para o diagnóstico em redes Ad Hoc, os nós sensores utilizados são nós estacionários, ou seja, o algoritmo não considera nenhuma mobilidade desses nós, o que difere do algoritmo DAINGeo. Algoritmo esse apresentado no Capítulo seguinte.

4.2 Modelo de Chessa e Santi

Chessa e Santi [6] abordam o problema de se identificar falhas em redes Ad Hoc móveis. O modelo apresentado baseia-se em comparações de resultados obtidos por diferentes unidades que realizam uma mesma tarefa. Nesse trabalho, os autores apresentam duas diferentes abordagens para o diagnóstico. Na primeira, considera-se que a topologia da rede não muda durante o diagnóstico, além disso, as falhas de *hardware* e *software* podem ser facilmente detectadas. Na segunda abordagem, é permitido que haja mudanças na topologia da rede durante o diagnóstico.

O modelo é descrito como um grafo direcionado $G = (V,L)$, denominado grafo de comunicações, onde V é o conjunto de unidades móveis (nós) e L é o conjunto de ligações lógicas. Assim, um nó qualquer denotado por u poderá se comunicar diretamente com outro nó v próximo a ele. Nesse caso, o conjunto de unidades vizinhas a esse nó u é denotado por $N(u)$.

Para esse modelo é pressuposto que:

1. Cada unidade deverá possuir um identificador único;
2. Exista um protocolo de nível de enlace que forneça o seguinte:
   (a) A solvência de conflitos;
   (b) A comunicação a um salto de distância deverá utilizar uma transmissão primitiva confiável;
   (c) O receptor da mensagem conhece a identidade do remetente.

Dando atenção a assertiva (2.b), Chessa e Santi [6] consideram que dada a natureza da comunicação em redes sem fio, a confiabilidade pode ser obtida através da detecção de erros, correção de códigos e /ou por retransmissão de mensagens corrompidas.

O modelo proposto pelos autores baseia-se na regra de invalidação do modelo MM generalizado. Essa regra é descrita a seguir através da Tabela 4.1.

Nesse modelo um nó u envia à todos os seus vizinhos um teste e espera suas respostas. Na medida em que as respostas são recebidas, u compara os resultados obtidos e com base

na regra de invalidação, apresentada na Tabela 4.1, realiza o diagnóstico. Dada a natureza compartilhada da comunicação, os resultados dos testes gerados pelos nós vizinhos de u são recebidos por muitos outros nós.

u	v	w	resultado da comparação das unidades v e w gerado por u
sem-falha	sem-falha	sem-falha	0 (igualdade)
sem-falha	sem-falha	falha	1 (diferença)
sem-falha	falha	sem-falha	1 (diferença)
sem-falha	falha	falha	1 (diferença)
falha	qualquer	qualquer	x (qualquer resultado)

Tabela 4.1: Regra de invalidação do modelo MM generalizado

Topologia fixa durante o diagnóstico

Suponha que a topologia da rede não é alterada durante a execução dos testes, i.e. se um nó u envia um teste no instante t, e T_{out} é o tempo limite para esse teste, então os nós vizinhos a u no instante t devem ser os mesmos no instante $t+T_{out}$. Esta hipótese não significa que a rede seja estática, mas que a sua topologia não muda durante o diagnóstico. Os nós são autorizados a circular, mas não podem migrar para fora da área de transmissão dos seus vizinhos. A Figura 4.2 ilustra o processo de geração do pedido de teste feito pelo nó u aos seus vizinhos.

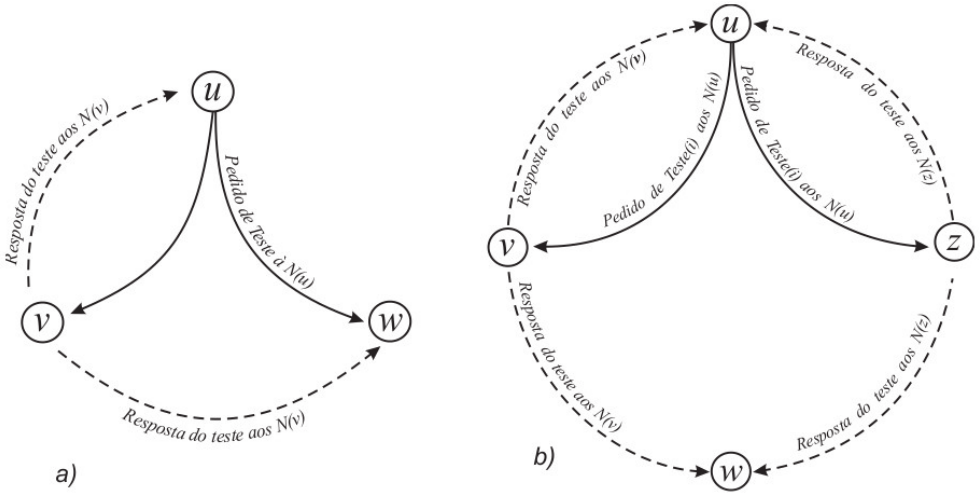

Figura 4.2: Modelo Chessa e Santi - Geração do pedido de teste.

Geração do pedido de teste: No tempo t, a unidade u (nó testador) gera um número de teste sequencial i, ou seja, uma tarefa de teste $Teste_i$. O pedido de teste é então enviado para todos os vizinhos de u a um salto de distância, Figura 4.2-a (As unidades v, $w \in \mathrm{N}(u)$ recebem o pedido de teste do nó u).

Recebimento do pedido de teste: Qualquer unidade $v \in N\ (u)$, ao receber o pedido de teste, gera o resultado do $Teste_i$. Esse resultado é encaminhado para todos os vizinhos de v a um salto de distância.

Recebimento da resposta de teste: qualquer unidade $w \in N\ (v)$, ao receber uma resposta de teste, faz o seguinte: Se $w = u$, ou seja, se w é o nó testador, ele compara com o resultado esperado, e gera o resultado da comparação. A unidade v é diagnosticada como sem-falha em caso de igualdade dos resultados, caso contrário o nó v é falho. No caso de $w \neq u$, as seguintes opções poderão ocorrer:

1. Se $w \neq u$ e $w \in N(u)$, então w também recebe o pedido de $Teste_i$ do nó u. Os resultados dos nós v e do nó w são comparados e o nó v é diagnosticado como falho ou sem-falha, segundo a regra de invalidação do modelo MM generalizado.
2. Se $w \neq u$ e $w \notin N(u)$, então w recebe de v a resposta do $Teste_i$, Figura 4.2-b. A resposta do teste é comparada com as respostas dos testes recebidos para a mesma tarefa (se houver). Se houver algum $z \in N(u)$ tal que as respostas de z e v após serem comparadas, sejam uma igualdade então ambos os nós são diagnosticados como sem-falha. Caso contrário, se o nó z for diagnosticado como sem-falha, então a unidade v será diagnosticada como falha.

Tempo limite de recepção: Todas as unidades que não responderam ao pedido de teste dentro do tempo limite T_{out} são diagnosticadas como falha.

Os autores mostram que ao assumir uma topologia de rede fixa, um nó u sem-falha gera um pedido de teste no tempo t, então, no instante $t + T_{out}$:

- A unidade u diagnostica corretamente o estado de todas as unidades vizinhas $N(u)$;

- Qualquer unidade $v \in N(u)$ e sem-falha consegue diagnosticar corretamente o estado das unidades como sem-falha e *soft fault*, para os nós que são vizinhos em comum de u e v.

- Qualquer unidade $z \notin N(u)$ considerada sem-falha, consegue diagnosticar corretamente o estado das unidades, sem-falha e *soft fault* para os nós comuns a u e z, ou seja, $N(u) \cap N(z)$, se pelo menos duas dessas unidades em comum estiverem sem-falha.

Topologia não-fixa durante o diagnóstico

Suponha agora que os nós estão autorizados a mover-se durante o tempo de execução do teste. As comparações são realizadas de acordo com o seguinte protocolo:

Geração do pedido de teste: No tempo t, a unidade u, gera um teste $Teste_i$. O teste é então enviado para todos os vizinhos de u a um salto de distância.

Recebimento do pedido de teste: Qualquer unidade $v \in N(u)$, ao receber o pedido de teste, gera o resultado do $Teste_i$.

Recebimento da resposta de teste: Qualquer unidade $x \in N(v)$, ao receber uma resposta de teste, faz o seguinte: Se $x = u$, ele compara com o resultado esperado, e gera o resultado recebido. A unidade v é diagnosticada como sem-falha se houver a igualdade dos resultados. Caso contrário, o nó v será considerado com falha. No caso de $x \neq u$, surgem as seguintes opções:

1. Se $x \neq u$ mas $x \in N(u)$, então w também recebe o pedido de $Teste_i$ do nó u. Os resultados dos nós v e x são comparados, e o nó v é diagnosticado como falho ou sem-falha, segundo a regra de invalidação do modelo MM generalizado.

2. $x \notin N(u,t)$. A unidade v é classificada como ativa. Além disso, sua resposta ao teste é comparada com as respostas de testes recebidos para a mesma tarefa (se houver). Se existir algum $z \in N(u)$ tal que as respostas de z e v após serem

comparadas resultem em uma igualdade, então ambos os nós são diagnosticados como sem-falha. Caso contrário, se o nó z foi diagnosticado como sem-falha, então a unidade v é diagnosticada como falha.

Tempo limite de recepção: Todas as unidades que não responderam ao pedido de teste dentro do tempo limite T_{out} são diagnosticadas como unidades com falha.

Uma vez que a topologia da rede varia com o tempo, em geral $N(u,t) \neq N(u,t + T_{out})$, ou seja, os vizinhos de u no instante t, podem não ser os mesmos no instante $t + T_{out}$. Como consequência, as unidades com *hard fault*, não podem ser distinguidas das unidades sem-falha que migraram para fora da faixa de transmissão das unidades de teste. Portanto, o diagnóstico deixa de ser correto.

Uma restrição adicional sobre o comportamento de nós com falha do tipo *soft-fault* deve ser imposta de forma a garantir a corretude do algoritmo de comparação. Nesse caso, considera-se um nó z sem-falha o qual envia um pedido de teste para todos seus vizinhos $N(z)$. Como o nó $v \in N(z)$, possui uma falha do tipo *soft-fault*, essa unidade v poderá gerar uma resposta incorreta ao teste, alterando o cabeçalho da mensagem, de forma que sua identificação passa a ser s e não mais v, isto é, um falso nó s. Em seguida essa resposta poderá ainda ser transmitida para seus vizinhos.

Considerando ainda a existência de um verdadeiro nó s presente na rede, o nó u, ao receber a resposta de s poderá não ser capaz de identificar corretamente o verdadeiro nó s, mesmo que as tarefas de testes possam ser diferentes. Isto significa que uma unidade u pode diagnosticar erroneamente uma unidade v como sem-falha. Este tipo de comportamento apresentado por um *soft-fault* é expressamente proibido nesse modelo de diagnóstico. A probabilidade de ocorrência desse comportamento indesejado dos nós móveis com *soft-fault* pode ser significativamente reduzido com a realização de uma verificação de consistência no cabeçalho da mensagem antes de processá-lo. Por exemplo, ao considerar $\{0, .., n\text{-}1\}$ como sendo o conjunto de identificadores dos nós móveis, uma unidade i somente poderá gerar sequência de testes com números congruentes, i.e. com o mesmo formato de sua identificação. Deste modo, ao receber uma mensagem com um cabeçalho

(i, j), qualquer nó sem-falha u poderá realizar uma verificação de consistência.

Os próprios resultados obtidos nesse trabalho apresentam pouca eficiência desse algoritmo para nós em movimento. Além disso, os autores afirmam ser necessário atribuir algumas restrições referente a mobilidade desses nós para que eles possam ser diagnosticados. Em ambas as abordagens apresentadas são utilizados *flooding* para a disseminação dos resultados locais.

4.3 Auto-Diagnóstico Distribuído

Em [4, 5], os autores estendem o modelo de diagnóstico proposto por Chessa e Santi [6] apresentando um protocolo de diagnóstico para redes Ad Hoc móveis permitindo que a topologia da rede varie em função do tempo. Dois novos protocolos são apresentados: o adaptativo e distribuído de auto-diagnóstico (Adaptive-DSDP) para topologia de redes; e o protocolo distribuído de auto-diagnóstico para redes móveis (Mobile-DSDP).

A ideia-chave de ambos os protocolos é que um nó, ao responder a uma solicitação de teste, encaminhe juntamente com a saída produzida, a tarefa de teste. Dessa forma, qualquer receptor será capaz de diagnosticar seu estado através da simples comparação da saída com resultados semelhantes a um mesmo teste realizado. Além disso, é possível comparar o resultado recebido com o seu próprio resultado.

Para a topologia em rede fixa, os autores comparam o protocolo de Adaptive-DSDP com modelo de Chessa e Santi [6]. As principais diferenças entre eles são:

- (i) em Chessa e Santi [6], um nó não é obrigado a incluir a tarefa de teste quando responde ao teste. No Adaptive-DSDP este deverá encaminhar a tarefa de teste dentro das respostas do teste;

- (ii) em Chessa e Santi, uma vez que um nó recebe a resposta de todos os seus vizinhos, ele encaminha para todos os outros nós em uma MANET usando *flooding*. No Adaptive-DSDP é utilizado uma árvore de expansão para o *broadcast* de informações de diagnóstico [5].

Geração de um pedido de teste: Quando a unidade u decide testar os seus vizinhos em um dado tempo t, ela transmite um pedido de teste. O pedido de teste inclui uma tarefa, T_i. Após o envio, $< Test;\ T_u >$, o primeiro *timer* é ativado, este é definido como T_{out}. Além disso, um segundo *timer* também é iniciado e definido como $T_{DiagnosisSession}$. Este último *timer* refere-se ao pior tempo de execução de uma sessão de diagnóstico, ou seja, o tempo previsto para que todas as unidades sem-falha possam responder. O segundo *timer* permite reconhecer que um nó dinâmico v deixou de responder ao pedido de teste por ter se movido além da área de cobertura dos outros nós, nesse caso, este não poderá ser diagnosticado.

Recepção de um pedido de teste: Quando o nó v recebe um pedido de teste de um dos seus vizinhos, u, este se comporta da seguinte forma: Se ele conhece o resultado R da tarefa de teste T_i, então ele define $R_u^v = R$. Caso contrário, ele realiza a tarefa de teste T_i e gera um resultado R_u^v. A resposta do teste é transmitida para todos os seus vizinhos, $< Resposta,\ T_i,\ R_u^v >$. A resposta do teste é armazenada em um conjunto de respostas, denotada por $Validated_v$, na qual todas as respostas corretas dos testes são mantidas. Tanto as respostas geradas pela própria unidade quanto as respostas obtidas durante a sessão de diagnóstico. Nesta fase, o nó v gera sua própria solicitação de teste. Caso ainda não tenha sido feita, ele envia o pedido de teste para todos os seus vizinhos. Considerando σ como sendo a síndrome do sistema, cada unidade móvel é obrigada a responder em no máximo $\sigma + 1$ requisições de testes, ou seja, deve informar a no máximo, $\sigma + 1$ vizinhos sobre o seu estado.

Recepção de uma resposta do teste: As resposta aos teste solicitados poderão ser originadas tanto por nós estáticos como por nós dinâmicos. Além disso, alguns desses nós dinâmicos poderão ser nós que migraram para a vizinhança de w. Nesse caso, nó w poderá receber tanto os resultados de teste quanto a tarefa de testar. A partir dos resultados recebidos o diagnóstico é realizado. Caso o nó w não consiga classificar corretamente o estado desses nós, estes são armazenados em um conjunto pendente, denominado$Pending_w$.

Tempo Limite: Após a ocorrência do primeiro *timeout*, i.e. T_{out}, o nó u será capaz de diagnosticar o estado de seus vizinhos estáticos, bem como os novos vizinhos que migraram para a sua vizinhança e que já receberam pelo menos uma mensagem de resposta. Nesta fase, o nó u divulga o resultado do diagnóstico local que recolheu de todos os seus vizinhos. Quando o segundo *timeout* ocorre, i.e. $T_{DiagnosisSession}$, o nó u irá diagnosticar todos os outros nós como falhos.

Nesse trabalho, os autores apresentam uma melhoria do modelo de Chessa e Santi [6], que segundo eles é o primeiro protocolo distribuído e adaptativo de auto-diagnóstico (Adaptive-DSDP) para o diagnóstico de falhas em uma topologia de rede fixa, além disso considera-se a existência de uma árvore de expansão mínima, na qual sua raiz é um nó da rede denominado iniciador. O objetivo dessa árvore é manter os nós da rede conectados a ela, facilitando ao fim do diagnóstico local, a disseminação desses resultados. Nesse caso, um procedimento para manutenção dessa árvore deve ser feito constantemente, a fim de manter todos os nós conectados a ela. Nós em movimento podem se desconectar momentaneamente de um dos galhos dessa árvore e conectar-se a outro, isto é, um nó ao perder seu pai, devido a seu deslocamento, deve em um momento seguinte solicitar sua reconexão junto a árvore. A busca desse nó (orfão) por um novo pai é feita de forma que o pai escolhido esteja dentro da sua área de transmissão e que este esteja o mais próximo possível do nó iniciador. Esse procedimento permite reduzir a profundidade da árvore e, portanto, a latência de diagnóstico.

O segundo protocolo apresentado é também distribuído e de auto-diagnóstico (Mobile-DSDP), no qual os autores fazem uma adaptação do modelo de Chessa e Santi [6]. Nesse caso, eles consideram que a topologia varia em função do tempo. A ideia é, mesmo que um nó v não possa mais alcançar o seu antigo vizinho u (seu testador), qualquer outro nó móvel próximo a ele será capaz de diagnosticar o seu estado, uma vez que qualquer nó da rede poderá possuir a tarefa de teste e o resultado dessa saída. Essa segunda abordagem é considerada pelos autores o primeiro trabalho em que se aborda o problema de diagnóstico em MANETs, sem contudo impor qualquer restrição sobre a mobilidade dos nós, isto é,

em comparação com o que já havia sido desenvolvido anteriormente com os protocolos apresentados em [6] e [10].

4.4 Diagnóstico para Redes Ad Hoc em Larga Escala

Um modelo para detecção de falhas em para redes Ad Hoc em larga escala usando uma abordagem baseada em comparações é apresentado em [17]. Nesse modelo é assumido que o ambiente é dinâmico, ou seja, os nós podem mudar seu estado durante a execução do algoritmo de diagnóstico. O modelo proposto é auto-diagnosticável e distribuído, e o conjunto de nós presentes no sistema pode ser populado tanto por nós móveis (MANETs) quanto por nós sensoriais (WSN). Os nós ativos fazem parte de uma rede MANET e são encarregados pelo processo de diagnóstico e agrupamento. Nós passivos (sensores), não realizam a tarefa de diagnóstico e não precisam guardar informações globais sobre toda a rede. Desta forma, não são todos os nós que diagnosticam a rede e portanto o tempo de diagnóstico é menor.

Nesse trabalho, os autores consideram a ocorrência de falhas do tipo *hard fault* e *soft fault*, onde *hard fault* são também denotadas como falhas do tipo *crash*. A reparação dessas falhas deve ser feita por algum administrador, como a reposição da bateria esgotada ou mesmo a manutenção física do equipamento.

Para as falhas do tipo *soft fault*, o modelo assume que cada nó móvel possui um valor estimado (pode ser valor energético ou valor de uma tarefa de diagnóstico) já anteriormente conhecida. Ao observar o valor obtido durante os testes, este é comparado com o valor estimado e sua diferença é calculada. Se o desvio for maior do que um determinado limiar Θ, o algoritmo diagnostica o nó como falho.

Os autores propõem a criação de *clusters* hierárquicos que resolvem o problema de escalabilidade sendo o processo iniciado por vários nós da rede, porém não todos, uma vez que isso iria gerar a demora no processo de diagnóstico. A formação de um *cluster* é feita através da construção de uma árvore. Para isso, um grafo é modelado utilizando um algoritmo de busca em largura.

Uma menor sobrecarga em relação à tradicional técnica de diagnóstico distribuído

é observado. Somente alguns nós iniciam o processo de agrupamento e diagnóstico do sistema, o que é feito após o processo de eleição de nós líderes em cada *cluster*. O algoritmo também evita que somente um nó desencadeie o processo, evitando assim o gargalo, que nesse caso todos os outros nós seriam obrigados a aguardar por informações de diagnóstico.

Os autores referem-se aos nós sensores que poderão estar em grande número, dificultando o processo de diagnosticabilidade. A corretude passa a ser questionável uma vez que esses nós sensores não armazenam informações dos demais nós presentes nesta rede mista. Nota-se que se a quantidade de nós ativos tornar-se muito pequena e espaça, fica, praticamente impossível o diagnóstico.

4.5 Avaliação de um Algoritmo de Detecção de Falhas em Larga Escala

Em [18] os autores avaliam um algoritmo de detecção de falhas usando uma abordagem baseada em clusters, onde é ressaltada a eficiência do agrupamento em casos onde as redes possuem muitos nós. Os autores consideram uma rede Ad Hoc grande e conexa em que todos os nós são conhecidos e identificados cardinalmente, 1, 2, 3, ..., N. Esses nós são distribuídos aleatoriamente em um determinado espaço físico. Esses nós são *hosts* estacionários em que o alcance de transmissão de cada nó é fixo e idêntico, sendo que a ligação entre eles é bi-direcional.

Os nós são agrupados em forma de clusters, onde a responsabilidade pela manutenção desses clusters bem como a detecção dos nós falhos é feita por um nó denotado como *clusterhead* (CH). Outro nó considerado importante e presente em um cluster é o *gateway* (GW). Esse nó é o responsável pelo encaminhamento de mensagens para clusters vizinhos. Nós pertencentes a um determinado cluster normalmente trabalham em modo promíscuo, e utilizam a técnica de *heartbeat* para a troca de mensagens entre seus vizinhos.

O algoritmo proposto pelos autores [18], divide toda a rede em dois níveis de arquitetura de comunicação, sendo elas: intra-cluster e inter-cluster. A detecção de falhas ocorre

no nível de intra-cluster, ou seja, cada cluster realiza seu próprio diagnóstico entre seus nós. Caso seja detectada uma falha em um cluster local, as informações são encaminhadas à outro cluster através da comunicação hierárquica inter-cluster.

Afim de manter os nós em clusters, mensagens de alto custo (*overhead*) são trocadas nos clusters, tanto intra-cluster quanto inter-cluster. Todos os membros de um determinado cluster precisam trocar mensagens entre si através de *heartbeat* a fim de manter o grupo em funcionamento.

Uma rede Ad Hoc organizada em clusters, sendo esta constituída por k número de clusters necessita trocar mensagens para sua manutenção inter-cluster. Nesse caso, um cluster i qualquer, com n membros, também denominado n_i, terá no pior caso um *overhead* de $n_i{}^2$ para sua manutenção inter-cluster. Além disso, o custo de manutenção desse cluster i é (k-1). Assim, o custo total para a manutenção de um cluster i é $n_i^2 + k - 1$. Logo, o *overhead* total da rede Ad Hoc é $\sum_{i=1}^{k} n_i^2 + k - 1$.

O serviço de detecção de falhas é tratado usando o mesmo mecanismo de *heartbeat* dentro de cada cluster. Cada nó envia periodicamente uma mensagem para o *clusterhead* (CH) do cluster local. Se o CH não receber uma mensagem de *heartbeat* dentro de período de tempo limite, então o nó é classificado como falho.

Os resultados obtidos pelos autores, concluem que o algoritmo é eficiente. A complexidade de mensagens trocadas, aumenta linearmente com o número de nós. O tempo de detecção de um evento local não depende do número de nós pertencentes ao cluster. A abordagem é linearmente escalável em termos de tempo de consenso, ou seja, o tempo para que todos os nós da rede conheçam um o nó falho, cresce de forma linear.

4.6 Diagnóstico de Falhas Dinâmicas em Redes Ad Hoc Móveis

Nesse trabalho [7], os autores utilizam o modelo de diagnóstico baseado em comparação para diagnosticar nós falhos em MANETs. O modelo proposto busca identificar falhas dinâmicas que surgem durante a fase de testes da sessão de diagnóstico. É assumido que cada nó tem um conjunto fixo de vizinhos, ou seja, o grafo representando a topologia é estático durante toda a sessão de diagnóstico.

Esse trabalho considera a existência tanto de *hard faults* quanto de *soft faults*. Falhas ocorridas durante o diagnóstico são consideradas permanentes. São ainda consideradas falhas estáticas e dinâmicas, sendo que falhas estáticas não podem surgir durante a sessão de diagnóstico, mas falhas dinâmicas podem.

Os autores consideram, que o número máximo de nós com falhas (σ) não deverá ultrapassar o valor da conectividade da rede κ(G), isto é, o número mínimo de nós que ao ser retirado desconecta o grafo. Dessa forma, $\sigma \leq \kappa$(G)-1 para que a rede ainda se mantenha conexa. Dentro do diâmetro (D_G) do grafo, uma MANET é denotada como σ-diagnosticável se todos os nós com falha podem ser inequivocadamente diagnosticados.

Nesse trabalho são apresentadas duas diferentes abordagens para a disseminação dos resultados diagnosticados. A primeira utiliza uma inundação simples, denominada Diagnóstico Distribuído Dinâmico com Inundações (*DDD Flooding*). Esse modelo é dividido em duas fases: fase de teste e fase de disseminação.

A segunda abordagem baseia-se em árvore de expansão, denominada de Diagnóstico Distribuído Dinâmico com *Spanning Tree* (*DDD Spanning Tree*). Nessa, o modelo é dividido em três fases principais: fase de testes, fase de construção e fase de divulgação.

A fase de teste de ambos os modelos, *DDD Flooding* e *DDD Spanning Tree* são iguais. Ao fim dessa fase, o modelo *DDD Flooding* continua utilizando o procedimento de *flooding* para a disseminação dos resultados obtidos, enquanto que em *DDD Spanning Tree*, uma árvore de expansão é primeiramente construída, a qual será em seguida utilizada como via para que a informação sobre todos os nós possa ser disseminada.

Fase de Testes: Nessa fase, um nó considerado como iniciador gera um pacote com o seguinte formato <INIT_DIAGNOSTIC, Θ, T_{out} >, onde Θ é o intervalo de tempo concedido para que o pacote LIFE_RESPONSE seja gerado e T_{out} é o tempo de duração da fase de testes. Cada nó, após receber o pacote INIT_DIAGNOSTIC, chama o procedimento de resposta SEND_LIFE_RESPONSE dentro do intervalo Θ e retransmite o INIT_DIAGNOSTIC à seus vizinhos. Usando o formato do pacote <LIFE_RESPONSE, id_y, *idtask* ,R_v, seq_no_v>, onde o id_v é o ID do remetente, *idtask* é o ID de tarefa e o R_v é a resposta gerada pelo nó v. Esse procedimento

pode ser observado na Figura 4.3-a, a qual ilustra um nó iniciador u enviando o pacote <INIT_DIAGNOSTIC, Θ, T_{out} > a todos os nós da rede através de *flooding*. Além disso, observa-se também que um nó v retransmitindo esse pacote a todos os demais vizinhos e respondendo ao seu remetente, o nó u.

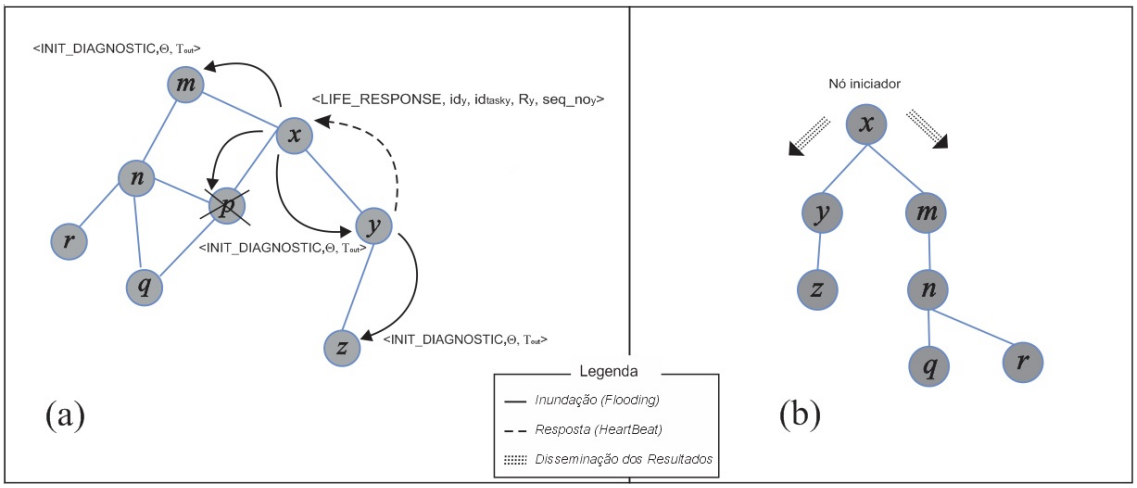

Figura 4.3: *Spanning Tree DDD* [7].

Cada nó ao receber o pacote <INIT_DIAGNOSTIC, Θ, T_{out} >, irá retransmitir o mesmo pacote ao seu vizinho, o qual irá responder ao remetente através de *heartbeat*, isto é, ele irá enviar periodicamente uma mensagem com as informações sobre seu estado ao nó remetente, até que o tempo limite T_{out} ocorra. O nó remetente ao receber esse pacote de resposta irá analisar se já não há outra resposta com o mesmo ID da tarefa de teste. Caso não haja, o nó remetente irá registrar a informação, caso contrário, o registro existente será atualizado e incorporado ao seu conjunto de nós sem-falha. Ao fim desse tempo limite T_{out}, os nós que não responderam terão seus ID adicionados ao conjunto de nós com falha. Após essa fase de testes, não serão permitidos aos nós a ocorrência de qualquer mudança sobre o seu estado.

Fase de Construção: Essa fase é iniciada a partir do momento em que o nó iniciador u envia o pacote de mensagem ST_MSG, para todos os nós da rede. Um método denominado de árvore de expansão - *spanning tree*(ST) é utilizado, o qual os autores [3, 7, 10], afirmam ser esse um método que consome menos energia em função da menor complexidade para o encaminhamento das mensagens. Somente os nós

diagnosticados durante a fase de teste com o estado de sem-falha podem fazer parte da construção dessa árvore, além disso, é necessário que cada nó filho conheça seu nó pai. Caso exista um nó sem pai, o nó emissor da mensagem de construção, torna-se seu pai, isso se o nó remetente não está com falha. Um outro temporizador é iniciado afim de que todos os filhos respondam à seus pais. Ao fim desse segundo, T_{out}, os nós que não tem filhos começam a enviar as informações de diagnóstico local para seus pais, iniciando assim a fase de disseminação.

Fase de Divulgação: Depois que a árvore geradora tiver sido construída, e todos os nós filhos tiverem enviado as informações locais à seus pais, e estes, por sua vez, coletarem as informações de todos seus filhos e retransmitem à seus pais, até que se atinja o nó iniciador do processo, ou seja, até a raiz da árvore. O nó iniciador, que agora possui a informação global e atualizada de diagnóstico de todos os nós da rede, passa agora a enviar, para baixo, (de pai para filho) a informação global sobre o estado de todos os nós da rede, Figura 4.3-b. Dessa forma, todos os nós sem-falha passam a ter a visão global de todos os nós da rede [3].

Neste trabalho foi possível observar que os nós somente poderão mudar seu estado durante a fase de teste. Além disso, os autores apresentaram e avaliaram duas próprias abordagens para a realização do diagnóstico de falhas em nós móveis. Porém após a realização das simulações, os autores optaram pela segunda abordagem como sendo esta o melhor método a ser utilizado para divulgação dos resultados de diagnóstico local, isto é, o Diagnóstico Distribuído Dinâmico com *Spanning Tree* (*Spanning Tree DDD*). Esse método foi mais eficiente em termos de complexidade da mensagem e latência do diagnóstico.

4.7 Conclusão

No trabalho apresentado sobre redes de sensores [25], são empregadas duas técnicas pertinentes ao modelo proposto: a localização do nó em uma determinada região geográfica e o modelo PMC como base. No modelo de Diagnóstico de Falhas em Redes de Sensores

os nós são formados por equipamentos fixos e geograficamente mapeados afim de reduzir as mensagens trocadas entre os nós em uma determinada área. Diferentemente de [25], o algoritmo DAINGeo, o qual é apresentado no capítulo seguinte, considera que os nós possuem a capacidade de se moverem e que a falhas podem ser dinâmicas. Além disso, ele contempla a ocorrência de falhas durante o diagnóstico.

O modelo que mais se assemelha ao modelo DAINGeo é o modelo de Chessa e Santi [6] por considerar uma topologia de rede não-fixa, na qual os nós podem mudar seu estado durante o tempo de execução do teste.

Observa-se que nos modelos baseados em comparação, as técnicas de agrupamento para diagnóstico em larga escala são interessantes quando se refere a uma rede composta por um muitos nós. O uso de *flooding* para inundação das mensagens, tanto para solicitação de testes quanto para a disseminação dos resultados é considerado simples e rápido no processo de disseminação, porém custa muito em relação a quantidade de mensagens trocadas. Além disso, a grande maioria dos trabalhos apresentados fazem algum tipo de restrição quanto a mobilidade dos nós.

Em [17] a construção e manutenção dos clusters considera que os nós encontram-se posicionados em um determinado espaço físico de forma que seja possível a construção de clusters hierárquicos. Embora a avaliação do algoritmo seja eficiente os nós com tecnologia sem fio tendem a ser móveis e dinâmicos tornando a técnica de construção de clusters ainda mais complexa.

Em [7] é apresentado o Diagnóstico de Falhas Dinâmica em Redes Ad Hoc Móveis o qual considera nós com falhas dinâmicas. Nesse trabalho, é considerado que os nós devem permanecer fixos durante toda a sessão de diagnóstico. Assim, um grafo representando a topologia MANET é estático. Além disso, esse modelo também se baseia em comparação.

CAPÍTULO 5

ALGORITMO DE DIAGNÓSTICO DE FALHAS DISTRIBUÍDO E ADAPTATIVO COM BASE EM INFORMAÇÕES GEOGRÁFICAS

Falhas de *hardware* e *software* são previstas em qualquer tipo de rede. No entanto, outros tipos de falhas são previstas em redes Ad Hoc móveis. Nessas redes, além das falhas comuns de *hard fault* e *soft fault*, poderão ainda ocorrer falsos positivos causados pela mobilidade desses nós.

O DAINGeo – algoritmo Distribuído e Adaptativo com base em Informações Geográficas para redes Ad Hoc, busca diferenciar as falhas reais dos possíveis falsos positivos. Para isso, o algoritmo utiliza as posições geográficas dos nós como referencial, dessa forma é possível diferenciar falhas de *hardware* e *software* das faltas de respostas que podem ocorrer em função da mobilidade dos nós.

Considera-se uma rede na qual a quantidade de nós é conhecida e a comunicação entre eles é feita através da transmissão de pacotes utilizando para isso, sinais de radiofrequência. Cada nó possui um identificador único e portanto poderá ser facilmente identificado por todos os nós da rede. Além disso, o posicionamento geográfico de cada nó é registrado a cada resposta de teste recebida. O algoritmo DAINGeo permite que a topologia da rede varie em função do tempo. Essa topologia é descrita como um grafo direcionado $G(t)$ $=(V,L(t))$, onde V é o conjunto de equipamentos móveis e $L(t)$ são as ligações lógicas existentes no instante t. Além disso, o algoritmo considera que os nós podem se mover durante todo o diagnóstico.

Esse algoritmo assume as seguintes asserções:

A-1 o número total de unidades móveis presentes no sistema é conhecido por todas as unidades do sistema;

A-2 todas as unidades tem o mesmo alcance de transmissão;

A-3 cada unidade se movimenta de forma aleatória e independente das outras unidades, podendo ainda se manter estáticas;

A-4 todas as unidades possuem um sistema de localização geográfica;

A-5 o sistema é considerado σ-diagnosticável. Para que isso seja possível, considera-se κ como a menor quantidade de nós que se removido desconecta a rede. Dessa forma, a quantidade de nós com falha deverá ser $< \kappa$-1.

Além dessas asserções, o algoritmo considera que, se um nó testador envia à todos seus vizinhos, os quais estejam a um salto de distância, solicitações de teste, estes devem responder dentro de um limite de tempo previamente determinado. Juntamente com as respostas, são agregadas informações sobre o posicionamento geográfico desses nós testados. O algoritmo considera ainda, que cada nó do sistema possui seu sistema de localização geográfica. Note que qualquer mecanismo de localização poderá ser utilizado.

Uma vez que os nós u, (nó testador) e v (nó testado), conhecem suas respectivas coordenadas geográficas, é possível calcular a distância Euclidiana entre eles. A partir da realização dos próximos testes, torna-se possível monitorar o deslocamento de cada nó em relação ao nó testador. Caso seja observado que um determinado nó deixou de responder ao pedido de teste devido ao possível distanciamento do seu testador, esse nó será classificado como suspeito.

5.1 Modelo de Falhas

Este modelo, admite falhas tanto de *hardware* quanto de *software*. Porém, não são admitidas falhas bizantinas. Além disso, o diagnóstico é dividido em rodadas de teste. Durante essas rodadas, são permitidas a ocorrência de eventos sobre os estados dos nós, ou seja, nós sem-falha podem se tornar falhos e vice-versa.

Os nós dessa rede podem, em determinado momento, possuir somente uma ligação lógica entre cada par de nós ativos como podem em momento seguinte possuir múltiplas

conexões entre si, conforme Figura 5.1.

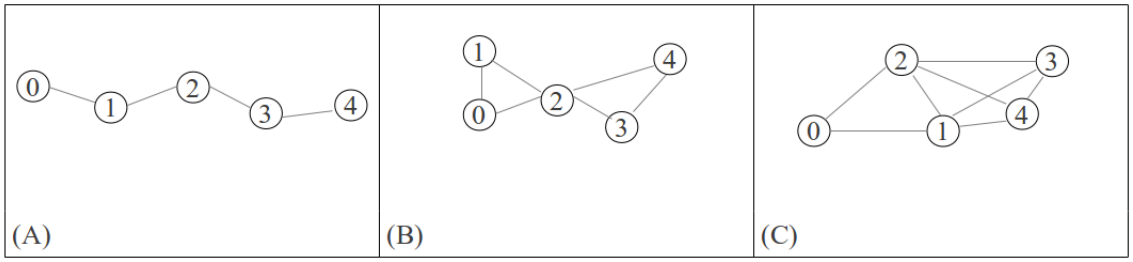

Figura 5.1: Variação da Topologia

A partir dessas topologias é possível analisar a capacidade do algoritmo em diagnosticar a rede. Considerando κ(G) como sendo a conectividade do grafo G, ou seja, a quantidade de nós que se removidos desconecta a rede, pode-se analisar os casos apresentados na Figura 5.1 como parâmetros para o melhor e pior caso possível. O caso ilustrado na Figura 5.1-A, apresenta um grafo no qual os nós estão dispostos em linha, nesse caso, o grau de conexão é igual a 1, isto é, κ(G)=1.

No caso da Figura 5.1-B, a conectividade desse sistema também apresenta um grau de conexão igual a 1, ou simplesmente, κ(G)=1. Embora esse caso apresente várias ligações para um determinado nó, um vértice de articulação é observado. Caso haja a ocorrência de falha nesse vértice de articulação, a rede se tornará desconexa.

Para o caso apresentado na Figura 5.1-C, é possível observar que serão necessários mais de um nó com falha para que a rede se torne desconexa, ou seja, $\kappa(G) > 1$.

Pode-se afirmar que, uma rede será σ-diagnosticável se a conectividade da rede $\kappa(G) \geq 1$, isto é, a diagnosticabilidade da rede dependerá da topologia do grafo existente a cada rodada de teste. Como esse modelo não considera que haja desconexão do grafo, o valor de κ(G) somente poderá ser maior ou igual a 1. Logo, a rede é σ-diagnosticável, ou seja, todos os nós com falha poderão ser corretamente identificados, desde que o número de nós com falha não exceda σ.

5.2 Algoritmo de Diagnóstico

O algoritmo DAINGeo foi proposto para o diagnóstico em redes Ad Hoc, sendo ele distribuído e dinâmico. Baseia-se no modelo de diagnóstico PMC [22], sem a figura do

observador central. Cada nó é capaz de recolher os resultados de suas solicitações de teste e de efetuar por si só o diagnóstico. Nesse algoritmo, cada nó tem a capacidade de testar, ser testado e também realizar o diagnóstico.

Em DAINGeo, os testes são realizados através de rodadas de teste. Todos os nós enviam aos seus vizinhos a 1 salto de distância uma solicitação de teste no instante t. O diagnóstico do sistema é feito após todos os nós obterem os resultados de todos os demais nós da rede. Cada nó analisa os resultados obtidos e diagnostica seu estado como falho, sem-falha ou suspeito. Nota-se que os nós conseguem diagnosticar todos os demais nós do sistema embora os testes sejam feitos apenas com seus vizinhos adjacentes.

Não há nesse algoritmo uma fase especial para a disseminação da informação. A disseminação é feita juntamente com as rodadas de teste e torna-se completa quando todos os nós obtiverem a informação sobre todos. Cada nó poderá diagnosticar todos os nós baseados em suas próprias informações e em informações recebidas de seus vizinhos adjacentes.

Considerando o cenário, no qual um nó testador u e um nó $v \in \mathrm{N}(u)$, ao enviar uma solicitação de teste ao nó vizinho, esse também solicita que seja informado o seu posicionamento geográfico. O nó u ao receber essas coordenadas converte em distâncias Euclidianas entre ele e o nó vizinho v. A cada nova resposta recebida de v, tanto as informações geográficas quanto os valores referentes a distância entre os nós são atualizados.

Considerando ainda, que todos os nós de um determinado cenário são conhecidos e que um deles deixou de responder a uma solicitação de teste após T_{out}, uma possível localização desse nó poderá ser estimada pelo seu testador. Nesse caso, o nó v o qual deixou de responder ao teste terá sua distância prevista calculada. Como se trata de uma distância estimada, o nó v, em movimento, poderá estar situado em qualquer ponto dentro de uma área (α). As Figuras 5.2-ab ilustram essa área α bem como a área β a qual se refere a área de cobertura do nó testador u e a área γ, a qual é denominada como área de incerteza.

A partir da apresentação desse cenário pode-se apresentar 3 definições:

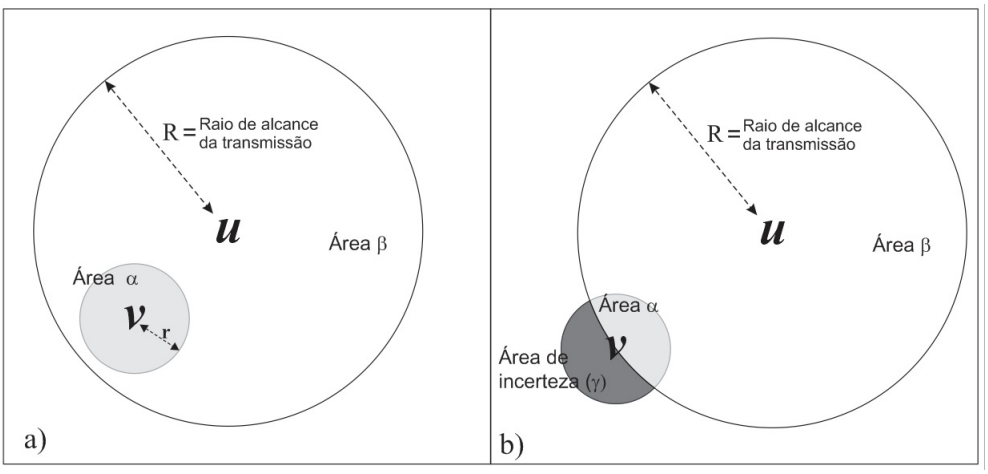

Figura 5.2: Possibilidade de deslocamento dos nós móveis.

Primeira Definição: Considerando $\Delta t = (t_1\text{-}t_0)$, onde, t_0 é o tempo em que o nó u comunicou-se com o nó v pela última vez e que t_1 é o tempo atual. Pode-se estimar a distância(r) em que o nó v poderá viajar no intervalo de tempo Δt, Figura 5.2-a. Essa distância(r) é dada pela velocidade * (t_1-t_0).

Segunda Definição: Se toda área α estiver contida em β e o nó v não respondeu ao teste durante T_{out}, então ele será considerado falho.

Terceira Definição: Se alguma parte da área α não estiver contida na área β e v deixou de responder ao teste, então ele poderá estar dentro da área de incerteza γ. Devido a essa incerteza, o nó v será classificado como suspeito.

5.3 Fase de Teste

Nesta fase os nós são organizados em três conjuntos distintos, sendo eles: falho $\{F\}$, sem-falha $\{FF\}$ e suspeito $\{S\}$. O algoritmo 1, procura utilizar as mesmas variáveis utilizadas em [7]. A divulgação dos resultados é feita de forma compartilhada e ocorre durante o envio das respostas aos testes solicitados, ou seja, por *piggybacking*. No instante em que todos os nós obtiverem a informação sobre todos, o diagnóstico estará completo.

Neste algoritmo observa-se:

INICIO_DIAGNOSTICO: Cada nó considerado sem-falha poderá diagnosticar todos seus nós vizinhos os quais estejam a um salto de distância. Para iniciar a fase de teste, um pacote denominado INICIO_DIAGNOSTICO envia através de *broadcast* à todos os nós que estiverem dentro da área de transmissão do nó testador com o seguinte formato: <INICIO_DIAGNOSTICO,T_i,T_{out} >, onde T_i é o teste identificado a cada rodada de teste, T_{out} é o tempo limite para v responder a u. Esse procedimento se repete a cada rodada de teste, até que todos os nós da rede possam ter a informação sobre todos.

Cada nó vizinho de u após receber o pacote INICIO_DIAGNOSTICO, realiza os seguintes procedimentos:

- Chama o procedimento ENVIA_RESPOSTA_TESTE com limite de tempo T_{out};
- O procedimento RESPOSTA_TESTE é usado para gerar a resposta solicitada pelo nó testador.

O algoritmo 2, apresenta os procedimentos utilizados para o envio da resposta solicitada pelo nó testador u.

RESPOSTA_TESTE: Após o início do diagnóstico ocorrido no instante t, o nó v deverá responder ao nó testador u com a resposta do teste no instante t'. O tempo limite T_{out} é usado para o encerramento do envio dessas respostas. Portanto $t \leq t' \leq t + T_{out}$.

Como a disseminação da informação está implicita no próprio procedimento de resposta, o formato do pacote de resposta terá como parâmetros < RESPOSTA_TESTE, id_v, R_v, t_uR_v, $lat_t'(v)$, $long_t'(v)$ > juntamente com todas as informações que v possui.

O nó v que já havia testado outros nós e recolhido seus resultados, agora responde ao nó testador u com as informações de v juntamente com todas as informações que v já possuía de outros nós. Além disso, como cada nó registra a entrada das mensagens recebidas em t_uR_v, somente serão aceitas as informações mais atuais.

Cada nó, após receber a solicitação de teste, irá proceder da seguinte forma:

Algoritmo 1: Fase de Teste

id_v: Identificador do nó v, *N(u)*
$\mathrm{t}_u R_v$: instante t em que u recebe a resposta de v
$\mathrm{lat}_t(v)$ e $\mathrm{long}_t(v)$: Coordenadas geográficas de v no instante atual t
R_v: Resposta do nó v
$\mathrm{R}_{N(v)}$: Resposta do nó *N(v)* compartilhada pelo nó v
$\hat{d}(uv)$: Distância prevista do nó u ao nó v
Raio_u: Alcance de transmissão do nó u

<INICIO_DIAGNOSTICO,T_i,T_{out} >
RESPOSTA_TESTE: < RESPOSTA_TESTE, id_v, R_v, $t_x R_v$, $\mathrm{lat}_t(v)$, $\mathrm{long}_t(v)$, $\mathrm{id}_{N(v)}$, $\mathrm{R}_{N(v)}$ >
se *R(v)* $\in$ { *FF* } **então**
  registra info de v: id_v, R_v $t_x R_v$, $\mathrm{lat}_t(v)$, long_t'*(v)*;
  /* Calcula a distância Euclidiana entre u a v e atualiza o valor da média anteriormente registrada */
  calcula $d(u,v) = \sqrt[2]{(u'_t - u_t)^2 + (v'_t - v_t)^2}$
  /* *se houver um* $z \in N(v)$ *e* $z \notin N(u)$ */
  registra info de *N(y)*: $\mathrm{id}_N(v)$, $\mathrm{R}_{N(v)}$, $\mathrm{t}_x\mathrm{R}_N(v)$;
  quando houver $id_v == id_v$ anteriormente registrado, guarda o registro completo de v no instante t e atualiza: R_v $\mathrm{lat}_t(v)$; $\mathrm{long}_t(v)$
fim se
discarta as informações registradas de *N(v)*;
TIMEOUT $\leftarrow$ true;
/* TIMEOUT: após a ocorrência do timeout, u calcula a distância prevista dos nós que deveriam ter respondido à solicitação de teste dentro do limite de tempo esperado */
se $R(y) = \emptyset$ *e* $\hat{d}(uv) > Raio_u$ **então**
  /* *Será considerado um nó suspeito se após* $t + T_{out}$ *a distância prevista* $\hat{d}(uv)$ *for > alcance de transmissão de u* */
  retira v do conjunto { FF }; (caso esteja)
  retira v do conjunto { F }; (caso esteja)
  adiciona v ao conjunto { S };
fim se
se $R(y) = \emptyset$ *e* $\hat{d}(uv) \leq Raio_u$ **então**
  /* *Será considerado um nó com falha aquele que deixar de responder ao teste após* $t + T_{out}$ *e cuja distância prevista* $\hat{d}(uv) \leq$ *alcance de transmissão de u* */
  retira v do conjunto { FF }; (caso esteja)
  retira v do conjunto { S }; (caso esteja)
  adiciona v ao conjunto { F };
fim se

Algoritmo 2: ENVIA_RESPOSTA_TESTE

procedimento ENVIA_RESPOSTA_TESTE;
Responde ao testador u em t < T_{out};
Envia para u o pacote com os parâmetros < RESPOSTA_TESTE, id_v, R_v, lat_t'*(v)*, $long_t$'*(v)* >
se *N(v)* $\neq \emptyset$ **então**
    envia RESPOSTA_TESTE: $id_{N(v)}$, $R_{N(v)}$;
    /* Envia por *piggybacking* as informações dos N(v) */
fim se
descarta o pacote de info N(v);
fim procedimento

- Verifica o tempo limite, T_{out};
- Empacota os resultados anteriormente obtidos dos seus vizinhos;
- Envia o pacote de informações dos *N(v)*, juntamente com o resultado do teste solicitado pelo nó u.

5.4 Funcionamento

Com a finalidade de apresentar o funcionamento do algoritmo, um modelo de cenário é apresentado na Figura 5.3. A partir desse cenário, as tabelas seguintes, apresentam as informações obtidas a cada rodada de teste.

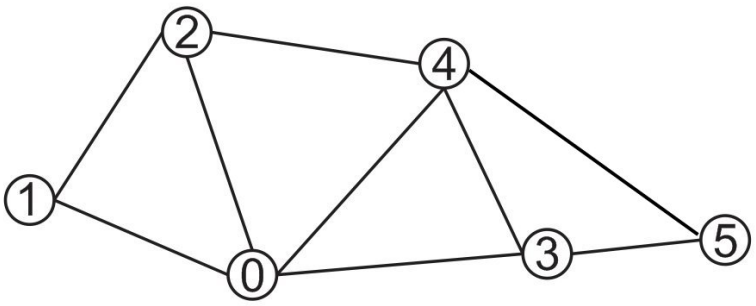

Figura 5.3: Cenário

Embora todos os nós realizem exatamente as mesmas tarefas, a Tabela 5.1 apresenta somente os resultados obtidos pelo nó 0, na qual observa-se os nós 1, 2, 3 e 4 diagnosticados pelo nó 0 como sem-falha e com suas respectivas coordenadas geográficas.

id_v	$id_{N(v)}$	t_uR_v	R_v	lat(v)	long(v)	$R_{N(v)}$
0		0,0000		422,28	257,26	
1		0,0018	FF	399,70	271,46	
2		0,0027	FF	430,41	258,32	
3		0,0051	FF	419,28	345,26	
4		0,0062	FF	423,11	329,38	

Tabela 5.1: Primeira Rodada de Teste

Como em uma única rodada não foi possível obter informação sobre todos os nós, uma nova rodada de teste foi iniciada. O nó 5 é agora visto na Tabela 5.2, uma vez que este nó é vizinho de 3 e não vizinho adjacente de 0. Observa-se para este caso, que o nó 3 já havia testado o nó 5 na rodada passada, tendo em seguida encaminhando seus resultados de diagnóstico para o nó testador 0.

id_v	$id_{N(v}$	t_uR_v	R_v	lat(v)	long(v)	$R_{N(v)}$
0		2,0000		428,74	262,27	
1		2,0023	FF	423,11	259,38	
2		2,0051	FF	408,03	300,93	
3		2,0051	FF	410,21	355,11	
3	5	2,0051	FF			FF
4		2,0061	FF	483,11	355,98	

Tabela 5.2: Segunda Rodada de Teste

A Tabela 5.3 apresenta os resultados referentes ao grafo ilustrado na Figura 5.4. Nestes, objetiva-se explicar a ocorrência de nós falhos e nós suspeitos. Para tanto, observa-se que as coordenadas geográficas foram convertidas em distâncias. Além disso, dois diferentes eventos ocorridos após T_{out} são também exemplificados. Para melhor demonstrar o funcionamento desse modelo, introduziu-se a coluna $\hat{d}(uv)$, com os resultados dos cálculos efetuados pelo nó 0 referente à distância prevista dos nós vizinhos os quais deixaram de responder ao teste.

No primeiro evento, observa-se a ocorrência de falha do nó 3 após a última rodada de teste. O nó 3 deixou de responder à solicitação de teste mesmo tendo sua distância prevista menor que o raio de alcance da transmissão que é de 130 metros. Como esse nó não mais encontra-se ativo, todas as informações anteriormente repassadas por esse nó agora perdem a confiança. Nesse caso, o nó 3 é diagnosticado como falho e as informações

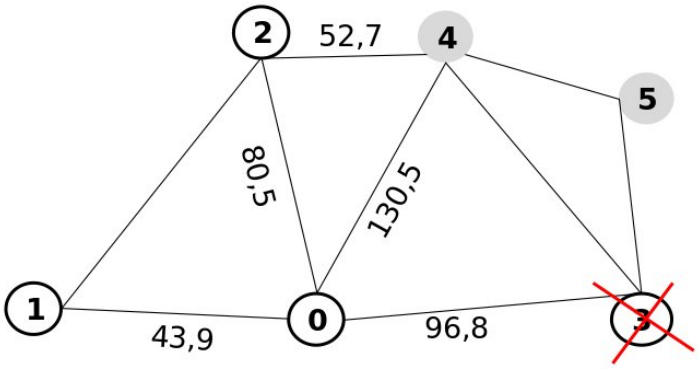

Figura 5.4: Cenário 2 - Ocorrência de eventos

id_v	$id_{N(v)}$	$t_u R_v$	R_v	$lat(v)$	$long(v)$	$R_{N(v)}$	$\hat{d}(uv)$
0		4,0000		421,12	262,27		
1		4,00279	FF	387,12	234,60		
2		4,00282	FF	354,12	217,60		
3		5,00000	F				96,789
3	5	5,00000				S	
4		5,00000	S				130,466

Tabela 5.3: Eventos ocorridos e localização prevista

de diagnóstico anteriormente repassadas são agora classificadas como suspeitas. Nesse exemplo, o nó 5 é agora um nó classificado como suspeito.

No segundo evento, o nó 4 ao deixar de responder ao teste, é diagnosticado como suspeito, uma vez que sua distância prevista está na área limite do alcance de transmissão. Durante a realização do teste, esse nó pode até ter recebido a solicitação de teste mas foi impedido de responder ao testador devido ao seu distanciamento.

Após a ocorrência de T_{out}, os nós que não responderam ao teste, têm sua distância prevista calculada pelo nó testador. Para o cálculo da distância prevista, utilizou-se a sua última coordenada geográfica registrada a qual é somada ao deslocamento do próprio nó testador e ao valor médio da distância que v percorre constantemente.

Considerando o cenário apresentado na Figura 5.4, o algoritmo analisa a cada rodada de teste e busca solver possíveis conflitos de resultados de diagnóstico obtidos entre nós suspeitos e nós sem-falha. Para esse caso, é considerado três nós distintos u, v e z os quais são todos vizinhos entre si. O nó v ao testar o nó z classifica-o como sem-falha num instante t. Como essa informação é compartilhada durante a fase de teste, o nó u

também receberá essa mesma informação. O nó u ao testar o nó z em um instante $t+1$, poderá classificar esse mesmo nó z como suspeito, isto é, caso o distanciamento entre eles seja além da área de cobertura.

Observa-se que os nós u e v possuem informações diferentes sobre um mesmo nó. Além disso, poderá ocorrer de um nó ficar entrando e saindo da área de cobertura de outro nó, tornando-se intermitente. O algoritmo verifica durante a entrada de registro, que se houver um nó z classificado como sem-falha e a próxima entrada de registro para esse mesmo nó for a de nó sem-falha, o algoritmo irá descartar a classificação anteriormente dada ao nó z como suspeito e registrará a entrada desse nó como sendo um nó sem-falha.

5.5 Quando um nó é classificado como falho

Um nó é classificado pelo algoritmo como falho nos seguintes casos:

Caso 1: Considerando um nó u como testador, este envia um pedido de teste ao nó v que não responde dentro do tempo limite pré estabelecido, T_{out}. A partir disso, o algoritmo invoca o procedimento para estimar a atual localização do nó v. Como se trata de uma distância prevista, não se pode afirmar que esse nó esteja posicionado em um ponto bem definido, mas sim dentro de um área prevista, denominada por área α e ilustrada na Figura 5.2-a. Caso toda essa área α esteja localizada dentro da área de cobertura do nó testador, $\alpha \in \beta$, o nó é considerado falho, pois v estaria em condições de responder ao testador.

Caso 2: Como as informações são compartilhas por todos os nós do sistema, um nó classificado como falho por outro nó sem-falha irá repassar o estado dele para todos seus vizinhos adjacentes. Após o fim das rodadas de teste, todos terão a informação sobre o estado desse nó.

5.6 Quando um nó é classificado como suspeito

Observando o algoritmo DAINGeo, este classifica um nó como suspeito após a ocorrência do tempo limite T_{out}. Para diferenciar nós com falha, de nós suspeitos, é necessário observar a localização do nó em relação ao seu testador, i.e. analisar o posicionamento da área α em relação a área β, áreas essas anteriormente descritas e ilustradas na Figura 5.2-b. Os casos em que um nó é classificado como suspeito são a seguir apresentados:

Caso 1: Um nó em movimento poderá no instante t possuir toda sua área α pertencente a área β, i.e. um nó $v \in N(u)$ no instante t. Nesse caso, o nó testador envia o pedido de teste que é recebido pelo nó v. Porém, o nó v não mais consegue responder ao testador em função da continuidade do seu deslocamento e consequentemente do aumento da distância entre eles. Nesse caso, a área α do nó v deixou parcialmente ou totalmente de pertencer a área β. Dessa forma, o nó é v é classificado como suspeito.

Caso 2: Para esse caso, considera-se um nó $v \in N(u)$ no instante t. O testador envia uma solicitação de teste ao nó v o qual não consegue receber o pedido uma vez que este se tornou inalcançável no exato momento do envio da solicitação. Além disso, a área β de ambos os nós entraram no limite do alcance de transmissão entre si, isto é, criou-se uma distância constante. Nesse caso, ambos os nós irão reportar diagnósticos de nós suspeitos. Esses nós poderão permanecer estáticos ou em movimento síncrono sem que haja alteração de distâncias entre eles.

Caso 3: Para esse último caso, um nó $v \in N(u)$ e também $N(z)$, porém $z \notin N(u)$ terão entre si as informações de respostas compartilhas. Dessa forma, o nó u obterá as informações do nó z através do nó v. Neste caso, o nó v será o nó responsável pelas informações sobre o estado do nó z. Caso o nó v torne-se falho ou mova-se de forma a tornar-se suspeito, o nó u não mais poderá confiar nas informações a ele repassadas. Neste caso, o nó z será considerado suspeito até que uma nova informação sobre seu estado seja encaminhado ao testador u.

5.7 Quando um nó é classificado como sem-falha

O algoritmo classifica um nó sem-falha nos seguintes casos:

Caso 1: Se um nó v o qual possua toda sua área α pertencente a área β, i.e. $v \in \mathrm{N}u$, então o testador u poderá enviar uma solicitação de teste a v. O nó v será considerado sem-falha caso ele responda dentro do tempo limite T_{out}. Observa-se para este caso que o nó testador u não invoca o cálculo da distância prevista, poupando processamento.

Caso 2: Assim como anteriormente apresentado sobre o compartilhamento das informações, um nó sem-falha deverá repassar a informação sobre todos seus vizinhos testados a seu testador.

5.8 Conclusão

Nesse algoritmo DAINGeo, não se considera a existência de um observador central, como acontece no modelo PMC original. Aos nós diagnosticados, o algoritmo considera três diferentes conjuntos onde se classifica o estado desses nós em falho, suspeito e sem-falha. O emprego do conjunto de nós suspeitos é apresentado neste algoritmo como forma de diferenciar falhas ocorridas por *hardware* ou *software* de ausência de resposta ocorrida devido ao deslocamento do nó móvel.

Unidades móveis podem se movimentar para a zona limite de transmissão do nó testador e ali permanecerem indefinidamente, onde também são classificadas como suspeitas. Ao acompanhar as distâncias percorridas pelos nós em movimento é possível estimar sua próxima zona de localização. Assim, se um determinado nó tiver sua distância menor que o alcance da transmissão, e este não responder ao teste, sua falha é verídica. Caso contrário, se a localização prevista for maior que o alcance da transmissão, esse nó poderá estar sem-falha, porém inalcançável. Logo, será classificado como suspeito.

Cada nó poderá neste algoritmo, testar, ser testado e realizar o diagnóstico de seus vizinhos a um salto de distância. A disseminação dos resultados de diagnóstico para todos os demais nós da rede é feita na medida que os testes são realizados. Como não há uma fase exclusiva para disseminação da informação, a precisão dos resultados de diagnóstico

em cada um dos nós poderá ser incorreta em determinado instante, mas volta a ser correta novamente na medida que os nós atualizam seus resultados.

Embora seja explicado o valor da latência da rede no capítulo seguinte, pode-se observar que o tempo necessário para que as informações sejam atualizadas entre os nós, será o tempo gasto para que todos os nós da rede possam ter as informações sobre todos. Além disso, a topologia apresentada a cada instante poderá contribuir para reduzir ou aumentar esse tempo. No caso da rede possuir seus nós com múltiplas ligações, as informações de diagnósticos serão mais rapidamente disseminadas do que nos casos onde a topologia da rede for um grafo em forma de linha.

CAPÍTULO 6

ANÁLISE DO ALGORITMO DAINGEO

A avaliação é apresentada em forma de grafo *G(t)=(V,L(t))*. O grafo é denotado como um conjunto de unidades móveis (V), onde ligações lógicas (L) variam em função do tempo t. A posição dos nós é registrada através de coordenadas geográficas e organizadas através de ligações lógicas entre os nós.

Neste capítulo, é apresentada a análise sobre modelo proposto fornecendo, a prova da complexidade de tempo e da mensagem, a prova da diagnosticabilidade, corretude e completude.

6.1 Complexidade de tempo

Como no modelo PMC, neste algoritmo a complexidade de tempo é medida em forma de rodadas de teste. A latência do algoritmo é a quantidade de rodadas de teste necessárias para que todos os nós do sistema obtenham informações sobre todos. Assim, pode-se medir a complexidade de tempo do diagnóstico através da análise do pior caso de latência da rede.

O posicionamento dos nós é feito de forma alinhada tal que os nós encontram-se no limite máximo do alcance de transmissão entre seus vizinhos. A Figura 6.1 ilustra essa topologia, onde é possível observar as informações obtidas por cada nó durante cada rodada de teste. O processo é assíncrono, isto é, cada nó testa a seu próprio tempo. O diagnóstico estará completo quanto todos os nós do sistema obtiverem as informações sobre todos. Neste exemplo, para a latência do diagnóstico foram necessárias 3 rodadas de teste.

Teorema 1: *A latência do algoritmo DAINGeo é n-1 rodadas de teste para o pior caso.*

Prova: A prova é trivial para o pior caso de latência. Considerando o grafo *G=(V,L)*,

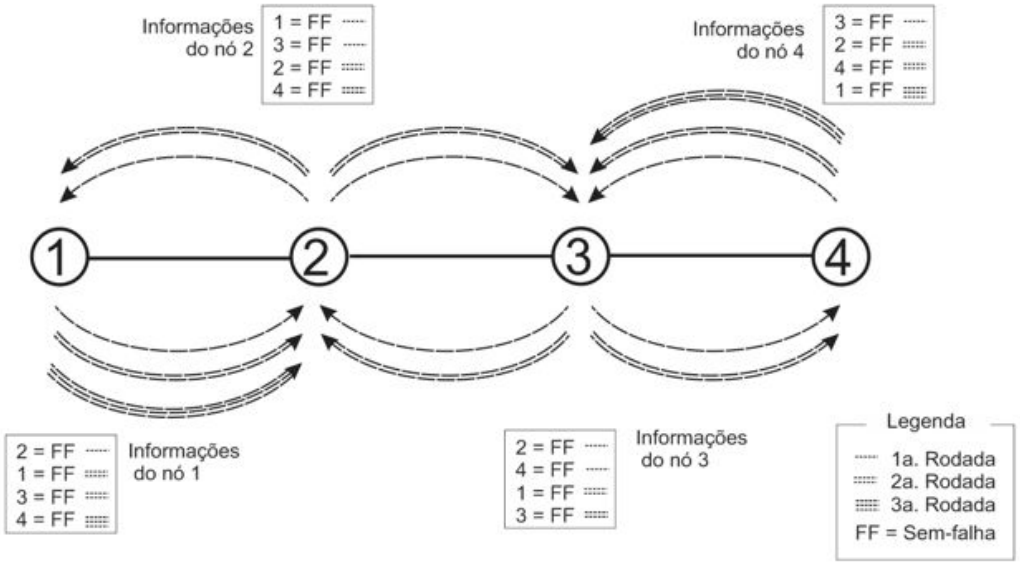

Figura 6.1: Rodadas de teste em um grafo com 4 nós dispostos em linha.

onde V_G=(u_0, u_1, u_2,...,u_{u-1}, u_k), tal que $u_k \neq u_0$. Uma falha ocorrida com o último nó desta linha, u_k, poderá levar até *n-1* rodadas de teste para ser diagnosticada pela unidade u_0. Nesse caso, o alcance da transmissão é a distância máxima, isto é $d_{max}(u,v)$, tal que $v \in N(u)$.

6.2 Complexidade da Mensagem

Teorema 2 (*Broadcast*): *A quantidade de mensagens enviadas por cada nó, é n por rodada de teste.*

Prova: A prova é trivial. Cada nó da rede envia uma mensagem de solicitação de teste para todos seus nós vizinhos utilizando *broadcast*, os quais devem responder ao seu testador dentro do tempo limite T_{out}. Dessa forma, a quantidade de mensagens que cada nó irá trafegar por rodada de teste é *1 + (n-1)*. Logo, é $n *$ o número de rodadas de teste, pois no pior caso, cada nó poderá possuir até *n-1* vizinhos.

Teorema 3: *A quantidade total de mensagens trocadas entre os nós é* $O(n^2)$.

Prova: A prova é trivial. Com base no Teorema 2, a quantidade de mensagens

trafegadas por cada nó da rede é n mensagens por rodada de teste. Como todos os nós realizam o mesmo procedimento, então é $n*(n)$. Dessa forma, a quantidade total de mensagens trocadas na rede é n^2 por rodada de teste. Logo, é O(n^2).

6.3 Prova da diagnosticabilidade

Teorema 4: *O algoritmo é σ-diagnosticável se a quantidade de nós com falha for $< \kappa$-1.*

Prova: A prova é trivial. Se o número de nós com falha for $\leq \kappa$-1, então haverá um caminho entre quaisquer dois nós sem-falha na rede de forma que a informação poderá trafegar. Além disso, todos os nós falhos possuem pelo menos um vizinho sem-falha. Portanto, os nós sem-falha constituem uma rede conexa.

Caso contrário, se o número de nós com falha for $\geq \kappa - 1$, então haverá nós falhos em quantidade suficiente para particionar a rede em pelo menos dois subgrafos (G' e H'). Dessa forma, os nós contidos no subgrafo G' não conseguirão obter informações dos nós contidos no subgrafo H' e vice-versa, consequentemente o diagnóstico deixa de ser completo.

6.4 Prova de correção

Premissa 1: Cada unidade somente classifica como uma unidade falha ou sem-falha com base em seus resultados de teste, ou com base em informações recebidas de outras unidades sem-falha.

Lema 1: *Se a rede é conexa, então um nó v é corretamente identificado por pelo menos um nó vizinho sem-falha.*

Prova: A prova é trivial. Invocando as informações apresentadas no Teorema 4 e na assertiva A-5, ambos anteriormente apresentados. A rede somente poderá ser conexa. Portanto, cada nó terá pelo menos um nó vizinho sem-falha capaz de testar e ser testado.

Lema 2: *Um nó ao ser diagnosticado, somente poderá ser classificado em apenas um dos seguintes conjuntos: nós sem-falha* $\{FF\}$*, nós com falha* $\{F\}$*, ou nós suspeitos* $\{S\}$.

Prova: A prova é trivial e para tanto são apresentados os seguintes casos:

Caso 1: Dada uma solicitação de teste, o nó que deixa de responder a essa solicitação, terá sua distância prevista calculada. Considerando β como a área de cobertura do no testador e α a área de possível localização do nó móvel, esse nó será classificado como suspeito nos seguintes casos: (i) se pelo menos uma parte da área $\alpha \notin \beta$, então parte dessa área α irá pertencer a área de incerteza γ, Figura 5.2-b. Como não é possível afirmar com precisão se o nó está em α ou γ, então ele é classificado como suspeito; (ii) quando um nó $v \in \mathrm{N}(u)$ torna-se falho, toda a informação anteriormente repassada por v ao nó u deixa de ser confiável. Nesse caso, os $\mathrm{N}(v)$ tornam-se suspeitos.

Caso 2: Um nó será classificado como sem-falha, quando: (i) o nó testado responde a seu testador dentro do tempo limite T_{out}; (ii) quando houver o compartilhamento das informações de um nó sem-falha por outro igualmente sem-falha.

Caso 3: Um nó somente será classificado como falho, quando: (i) um nó deixa de responder a solicitação de teste e a área correspondente a sua possível localização (α) estiver totalmente contida na área β; (ii) houver o compartilhamento das informações de um nó com falha por outro sem-falha.

Logo, não havendo outras abordagens daquelas apresentados nos casos 1, 2 e 3, um nó somente poderá se classificado como suspeito, sem-falha ou com falha.

6.5 Prova de Completude

Lema 3: *O algoritmo não é completo se houver falha em um vértice de corte existente.*

Prova: Caso uma rede possua um vértice de corte, e este venha a falhar, a rede poderá ser dividida em pelo menos dois subgrafos (G' e H'). O diagnóstico será realizado

entre cada subgrafo, sem que haja a troca de informações entre os subgrafos. Nesse caso, o algoritmo será correto porém incompleto.

Teorema 5: *Se o grafo for conexo então não haverá nós suspeitos depois do período de latência.*

Prova: Considerando que o grafo da rede é conexo o qual foi anteriormente provado através do Lema 1. Invocando ainda a assertiva A1 a qual garante que a quantidade de nós do sistema é conhecida, pode se afirmar que: se a união dos conjuntos, sem-falha $\{FF\}$ e falho $\{F\}$ tiver como resultado o total de nós do sistema, o conjunto de nós suspeitos $\{S\}$ será nulo. Dessa forma, após o período de latência, todos os nós puderam ser diagnosticados, portanto ele é completo.

Teorema 6: *O algoritmo é completo se todos os nós estiverem sem-falha.*

Prova: A prova é trivial. Como o algoritmo faz a asserção de que a rede é conexa, e neste caso todos os nós possuem um vizinho sem-falha capaz de testar e ser testado, todos os nós poderão diagnosticar seus vizinhos e ainda compartilhar seus resultados. Logo, o algoritmo será completo se todos os nós do sistema estiverem sem-falha.

6.6 Conclusão

Para concluir essa seção, um comparativo referente a complexidade do tempo e da mensagem entre o algoritmo DAINGeo e os modelos apresentados por Madhu Chouhan[7] e Chessa e Santi [6] são apresentados.

Complexidade do Tempo

DAINGeo: Como o algoritmo utiliza o conceito de rodadas de teste para o cálculo da latência, a complexidade do tempo é dada como sendo o número de rodadas de teste a qual está diretamente relacionada com topologia da rede encontrada. Nesse caso, a

complexidade de tempo é *(n-1)*T_{OUT}.

Chessa e Santi: Nesse modelo, a complexidade de tempo é definido em termos de latência do diagnóstico isto é, o tempo decorrido entre o início e o fim da sessão de diagnóstico. Considerando T_{GEN} como o tempo máximo decorrido entre a recepção da primeira mensagem de diagnóstico e a geração do pedido de teste, em no máximo $D_G T_{GEN}$ todos os nós sem-falha geram seu pedido de teste. Além disso, uma vez que a solicitação de teste é enviada, qualquer unidade sem-falha pode diagnosticar seus vizinhos no tempo máximo de T_{OUT}. Como $D_G T_F$ é o tempo máximo necessário para que uma mensagem seja propagada por toda a rede, então a complexidade de tempo desse protocolo de diagnóstico é $D_G T_{GEN} + D_G T_F + T_{OUT}$.

DDD Flooding: Esse modelo, apresentado em [7] é divido em duas fases, fase de teste e fase de disseminação, as quais são descritas a seguir.

- *Fase de Teste*: Nesta fase, considera-se que $D_G T_{INIT}$ seja o tempo máximo decorrido entre o envio do pacote inicial e o recebimento da primeiro mensagem de resposta T_{INIT} o qual ocorre em função do diâmetro D_G, que a rede apresenta. Além disso, considera-se que $\beta D_G T_{GEN} + T_{OUT}$ seja o tempo máximo em que o nó remetente recebe a mensagem de resposta, calcula a sua própria tarefa e avalia a resposta recebida, a qual também ocorre em função do diâmetro D_G da rede e do intervalo de tempo β o qual representa o intervalo de tempo em que as respostas são enviadas periodicamente ao remetente até que o tempo limite T_{OUT} seja atingido. A complexidade para a fase de teste é portanto $D_G T_{INIT} + \beta D_G T_{GEN} + T_{OUT}$.

- *Fase de Disseminação*: Uma vez que cada nó possui suas informações de diagnóstico local, esse pacote pode então ser disseminado na rede através de *flooding*. O tempo gasto para que esse pacote de diagnóstico local seja disseminado por inundação e este atinja todos os nós da rede é dado por $D_G T_F$.

Considerando a soma das variáveis anteriormente descritas, a complexidade de tempo

total gasta no modelo *DDD Flooding* é dada por: $D_G T_{INIT} + \beta D_G T_{GEN} + D_G T_F + T_{OUT}$.

DDD Spanning Tree: Considerando que esse modelo possui três diferentes fases, essas são a seguir descritas.

- *Fase de Teste*: Como a fase de teste dos modelos *DDD Flooding* e *DDD Spanning Tree* são iguais, a complexidade de tempo desse modelo é também dada por: $\beta D_G T_{GEN} + T_{OUT}$.

- *Fase de Construção*: Para essa fase, o tempo máximo T_{PROP} necessário para que o pacote de mensagem o qual inicia a construção da árvore seja propagado, levando em consideração a profundidade da árvore $D_S T$ é acrescido ainda do tempo gasto para que cada nó filho inicie o envio das informações locais ao seu pai. A complexidade de tempo para essa fase é portanto, $T_{PROP} D_S T + T_{OUT}$.

- *Fase de Disseminação*: Nessa fase, inicialmente cada nó filho envia suas informações locais a seus pais, os quais agregam as próprias informações obtidas e encaminham até que todas elas atinjam o nó iniciador. Em seguida, o nó iniciador o qual passa ter a informação atualizada e completa sobre todos os nós, inverte o processo e devolve para todos os nós da rede e via árvore, o diagnóstico completo de todos os nós da rede. Dessa forma, o procedimento para disseminação dos resultados é executado duas vezes. Portanto são necessários $2 D_{ST} T_{PROP}$.

Para o modelo *DDD Spanning Tree*, a complexidade de tempo total é: $D_G T_{INIT} + \beta D_G T_{GEN} + 3 D_{ST} T_{PROP} + 2 T_{OUT}$.

Modelo de Diagnóstico	Complexidade do Tempo
Algoritmo DAINGeo	$(n\text{-}1) T_{OUT}$
Chessa e Santi [6]	$D_G T_{GEN} + D_G T_F + T_{OUT}$
DDD Flooding [7]	$D_G T_{INIT} + \beta D_G T_{GEN} + D_G T_F + T_{OUT}$
DDD Spanning Tree [7]	$D_G T_{INIT} + \beta D_G T_{GEN} + 3 D_{ST} T_{PROP} + 2 T_{OUT}$

Tabela 6.1: Complexidade da Mensagem - Comparativo

Observando os dados apresentados na Tabela 6.1, é possível concluir que o algoritmo DAINGeo apresenta melhores resultados que os demais.

Complexidade da Mensagem

DAINGeo: Utilizando o conceito de rodadas de teste, poderá no pior caso ser necessários *n-1* rodadas de teste para o diagnóstico completo da rede. Ao considerar que cada nó sem-falha, envia uma mensagem de solicitação de teste para seu vizinho adjacente e esse responde ao solicitado, a quantidade de mensagens será $n(n)$. Além disso, esse procedimento poderá ocorrer até *n-1* rodadas de teste. Portanto a complexidade da mensagem será $n^2(n-1)$.

Chessa e Santi: Em Chessa e Santi também é utilizado somente uma fase para o diagnóstico do sistema. Cada nó sem-falha gera no máximo uma mensagem de solicitação de teste e no máximo uma mensagem de resposta, $n(n)+n$. A quantidade de respostas recebidas dependerá do grau de conectividade dos nós vizinhos sem-falha, i.e. uma mensagem de resposta enviada para cada nó vizinho sem-falha, $n(d_{max})$. Portanto, a complexidade total de mensagens do sistema é $n(n+1+d_{max})$.

DDD Flooding:

- *Fase de Teste*: Na fase de teste desse modelo, todos os nó geram no máximo, uma mensagem. Um nó iniciador gera um pacote inicial o qual enviado para todos seus vizinhos adjacentes que em seguida são por eles retransmitidos para seus outros vizinhos. Nesse procedimento, a complexidade da mensagem é dada por n. Em seguida, cada nós envia as informações sobre o seu estado, utilizando o procedimento conhecido como *heartbeat*. A complexidade da mensagem para esse processo é βn. Nesse caso, β representa o intervalo de tempo em as respostas são enviadas periodicamente ao remetente.

- *Fase de Disseminação*: A rede é inundada para que a disseminação dos resultados

locais seja feita, sendo a complexidade dada por n^2.

Portanto, a complexidade da mensagem para o modelo *DDD Flooding* é $n(n+1+\beta)$.

DDD Spanning Tree:

- *Fase de Teste*: Ocorre assim como apresentado em *DDD Flooding*. Um nó iniciador gera o pacote inicial, que envia à todos seus vizinhos adjacentes, os quais retransmitem esse mesmo pacote para os demais vizinhos. Cada nó também utiliza o *heartbeat* para responder ao seu nó remetente.

- *Fase de Construção*: Nessa fase, ocorre o processo de construção da árvore de expansão. Considerando que o pior caso a ser apresentado é quando todos os nós estão sem-falha. Nesse caso, a complexidade para a construção dessa árvore é dada por n.

- *Fase de Disseminação*: Nessa terceira e última fase, cada nó filho envia suas informações locais a seus pais, os quais agregam as próprias informações e reencaminham até que o nó iniciador seja atingindo. Quando este tiver todas as informações sobre todos os nós da rede, este dissemina a informação global para todos os nós sem-falha pertencentes a árvore. A complexidade da mensagem para esses algoritmo de diagnóstico é dada através da soma total de todas essas fases, nesse caso é $4n\text{-}2+\beta n$.

Modelo de Diagnóstico	Complexidade da Mensagem
Algoritmo DAINGeo	$n^2(n\text{-}1)$
Chessa e Santi [6]	$n(n+1+d_{max})$
DDD Flooding [7]	$n(n+1+\beta)$
DDD Spanning Tree [7]	$4n\text{-}2+\beta n$

Tabela 6.2: Complexidade da Mensagem - Comparativo

Ao observar a complexidade da mensagem dos modelos anteriormente apresentados e resumidos na Tabela 6.2, pode-se concluir que o modelo *DDD Spanning Tree* utiliza uma quantidade de mensagens bem menor que os demais modelos.

Por fim, como os modelos apresentados em Madhu Chouhan [7], *DDD Flooding* e *DDD Spanning Tree* utilizam o procedimento de inundação em pelo menos uma de suas fases da seção de diagnóstico, isso faz com que a complexidade da mensagem seja reduzida, porém aumenta o tempo de realização do diagnóstico. Observa-se ainda que, tanto no modelo de Chessa e Santi [6] quanto no algoritmo DAINGeo, a seção de diagnóstico é realizada em uma única fase, reduzindo assim o tempo de realização do diagnóstico. Observa-se portanto que algoritmo DAINGeo demonstra ser mais rápido que os demais.

CAPÍTULO 7

CONCLUSÃO E TRABALHOS FUTUROS

Considerado por vários autores como um dos modelos de diagnóstico mais bem estudados, o modelo PMC pouco tem sido empregado como modelo de diagnóstico de falhas para redes Ad Hoc móveis. Entre esses poucos trabalhos, destaca-se o diagnóstico de falhas em redes de sensores sem fio [25].

Como forma de contribuição, o algoritmo apresentado nesse trabalho, o algoritmo DAINGeo - Distribuído e Adaptativo com base em Informações Geográficas para redes Ad Hoc, é um desses poucos algoritmos o qual baseia-se no modelo PMC. Porém, e como a grande maioria dos algoritmos em nível de sistema para o diagnósticos de falhas são baseados em modelos de comparação de resultados, dois desses trabalhos foram escolhidos como comparativo com o algoritmo DAINGeo. Entre os vários trabalhos encontrados na literatura, foram escolhidos o modelo de Chessa e Santi [6], o qual é um dos modelos também muito referenciado na área de diagnóstico de falhas e algoritmo apresentado por Madhu Chouhan [7], por ser este um trabalho muito recente encontrado durante a confecção desse trabalho.

Esse estudo, é apresentado como o primeiro modelo para diagnóstico de falhas em nível de sistema, o qual também é baseado no modelo PMC para redes Ad Hoc e que não impõe restrições quanto a mobilidade dos nós durante todo o diagnóstico. Além disso, o DAINGeo, baseia-se em informações geográficas para diferenciar falhas de *hardware* e *software* das falhas ocorridas em função da mobilidade dos nós.

O algoritmo DAINGeo, busca classificar o estado dos nós de uma rede organizando os resultados do diagnóstico em três conjuntos distintos: (i) nós falhos, (ii) nós sem-falha e (iii) nós suspeitos. Essa classificação, é resumidamente apresentada da seguinte forma: Ao considerar a ocorrência do tempo limite *timeout* no qual um nó deixa de responder ao seu testador este é classificado como falho, caso a distância prevista entre ele e seu testador

seja menor que o alcance máximo da transmissão. Caso contrário, i.e. a distância prevista seja maior, então esse nó possivelmente estará em movimento. Nesse caso, o nó torna-se inalcançável para o envio e/ou recebimento dos testes, sendo portanto considerado como suspeito. Por fim, nós que respondem ao teste dentro do tempo limite T_{out}, são classificados como sem-falha.

Os testes teóricos apresentados nesse estudo indicam que o algoritmo consegue diagnosticar corretamente os nós da rede, desde que não haja nós suspeitos. Esses nós poderão dificultar a realização do diagnóstico durante um período de latência, tornando-se incorreto. Porém, volta a ser correto novamente após esse período. Além disso, o algoritmo mostrou ser incompleto nos casos onde houverem vértices de corte. Nesse caso, a falha desse vértice poderá dividir a rede em pelo menos dois subgrafos, (G' e H'). Embora cada subgrafo possa realizar seu próprio diagnóstico, os nós contidos em G' não conseguirão obter informações dos nós contidos em H' e vice-versa.

Ao comparar o algoritmo apresentado em [7] e [6], com o algoritmo DAINGeo, todos eles consideram a mobilidade, mas somente o DAINGeo não impõe nenhuma restrição quanto a mobilidade dos nós durante todo o diagnóstico.

Além disso, nos dois modelos apresentados em [7] *DDD Flooding* e *DDD Spanning Tree* a complexidade da mensagem é menor, porém eles custam mais em relação ao tempo. Tanto em [6] quanto no algoritmo DAINGeo, apenas uma única fase engloba todos os procedimentos para a realização do diagnóstico, isto é, ambos são mais rápidos que os modelos apresentados em [7]. Destaque para o algoritmo DAINGeo, o qual se mostrou mais rápido que os demais.

Como sugestão de trabalhos futuros, pretende-se avaliar o algoritmo DAINGeo de forma mais detalhada, isto é, realizar simulações sobre a de sobrecarga, corretude, completude e latência.

BIBLIOGRAFIA

[1] Paolo Baronti, Prashant Pillai, Vince W. C. Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, 30(7):p. 1655–1695, 2007.

[2] Ronald P. Bianchini, Júnior and Richard W. Buskens. Implementation of online distributed system-level diagnosis theory. *IEEE Transactions on Computers*, v. 41, p. 616-626, 1992.

[3] Azzedine Boukerche, Hisham Elkadiki, and Mourad Elhadef. A distributed fault identification protocol for wireless and mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, v. 68 n. 3:p. 321–335, 2008.

[4] Azzedine Boukerche, Hisham Elkadiki, and Mourad Elhadef. Diagnosing mobile ad hoc networks: two distributed comparison-based self-diagnosis protocols. p. 18-27, 2006.

[5] Azzedine Boukerche, Hisham Elkadiki, and Mourad Elhadef. Self-diagnosing wireless mesh and ad hoc networks using an adaptable comparison-based approach. p. 983-990, 2007.

[6] S. Chessa and P. Santi. Comparison-based system-level fault diagnosis in ad hoc networks. *IEEE - Symposium on Reliable Distributed Systems*, p. 257-266, 2001.

[7] Madhu Chouhan, Manmath Narayan Sahoo, and P. M. Khilar. Fault diagnosis in manet. In *Advances in Computing and Communications*, volume 190 of *Communications in Computer and Information Science*, pages 119–128. Springer Berlin Heidelberg, 2011.

[8] K. Chwa and S. Hakimi. Schemes for fault-tolerant computing: A comparison of modularly redundant and t-diagnosable systems. *Information and Control*, v. 49, p. 212-238, 1981.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[10] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison based self-diagnosis protocol for wireless ad hoc networks. pages p. 165–172, 2006.

[11] S. L. Hakimi and K. Nakajima. On adaptive system diagnosis. *IEEE Transactions on Computers*, v. 33, p. 234-240, 1984.

[12] S. Louis Hakimi and A. T. Amin. Characterization of connection assignment of diagnosable systems. *IEEE Transactions on Computers*, v. 23, p. 86-88, 1974.

[13] Pankaj Jalote. *Fault tolerance in distributed systems*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA, 1994.

[14] Elias Procópio Duarte Júnior, Luiz Carlos Pessoa Albini, Alessandro Brawerman, and André Luiz Pires Guedes. A hierarquical distributed fault diagnosis algorithm based on clusters with detours. *LANOMS*, 2009.

[15] Elias Procópio Duarte Júnior, Alessandro Brawerman, and Luiz Carlos Pessoa Albini. An algorithm for distributed hierarchical diagnosis of dynamic fault and repair events. *ICPADS*, pages p. 299–306, 2000.

[16] Elias Procópio Duarte Júnior and Takashi Nanya. A hierarchical adaptive distributed system-level diagnosis algorithm. *IEEE Transactions on Computers*, v. 47, p. 34-45, 1998.

[17] P. M. Khilar and S. Mahapatra. A novel hierarchical clustering approach for diagnosing large-scale wireless ad hoc systems. *International Journal of Computers and Applications*, 31, 2009.

[18] Pabitra Mohan Khilar, Jitendra Kumar Singh, and Sudipta Mahapatra. Design and evaluation of a failure detection algorithm for large scale ad hoc networks using cluster based approach. p. 153-158, 2008.

[19] J. Maeng and M. Malek. A comparison connection assignment for self-diagnosis of multiprocessor systems. *In Digest 11th International Symposium on Fault Tolerant Computing*, p. 173-175, 1981.

[20] Miroslaw Malek. A comparison connection assignment for diagnosis of multiprocessor systems. p. 31-36, 1980.

[21] Mathew D. Penrose. On k-connectivity for a geometric random graph. *Random Struct. Algorithms*, 15(2):p. 145–164, 1999.

[22] Franco P. Preparata, Gernot Metze, and Robert T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Computers*, v. EC-16, n. 6, p. 848-854, 1967.

[23] Elizabeth M. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Pemarsonal Communications*, 6:46–55, 1999.

[24] Abhijit Sengupta and Anton T. Dahbura. On self-diagnosable multiprocessor systems: Diagnosis by the comparison approach. *IEEE Transactions on Computers*, v. 41, p.1386-1396, 1992.

[25] Andrea Weber, Alexander R. Kutzke, and Stefano Chessa. Diagnosability evaluation for a system-level diagnosis algorithm for wireless sensor networks. p. 241-244, 2010.

[26] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computer Communications Review*, 3:p. 3–11, 1999.

[27] Roverli Pereira Ziwich, Egon Hilgenstieler, Emerson Fabiano Fontana Carara, Elias Procópio Duarte Jr., and Luis Carlos Erpen De Bona. Using distributed diagnosis to deploy highly-available web servers. In *Proceedings of the 2008 Latin American*

Web Conference, LA-WEB '08, pages 129–134, Washington, DC, USA, 2008. IEEE Computer Society.